DEVELOPING A NEURAL NETWORK MODEL TO PREDICT THE ELECTRICAL LOAD DEMAND IN THE MANGAUNG MUNICIPAL AREA

by

Lucas Bernardo Nigrini

Thesis submitted in fulfilment of the requirements for the degree

Doctoral: Technologiae: Engineering: Electrical

in the

School of Electrical and Computer Systems Engineering,

of the

Faculty of Engineering and Information Technology

of the

Central University of Technology, Free State

August 2012

Supervisor: Prof G.D. Jordaan D.Tech. (Eng)

Training a neural network involves large amounts of data, and complex relationships exist between all the different parameters, so one will never understand or spend time on the solution at all.

DECLARATION OF INDEPENDENT WORK

I, LUCAS BERNARDO NIGRINI, hereby declare that this research project submitted for the degree DOCTORATE TECHNOLOGIAE: ENGINEERING: ELECTRICAL, is my own independent work that has not been submitted before to any institution by me or anyone else as part of any qualification.

....

L.B. Nigrini **Date**

ACKNOWLEDGEMENTS

I would like to thank the following persons and institution for their help with, and contribution to the completion of this project:

The Central University of Technology, Free State, for the opportunity to do this project.

Prof G.D. Jordaan for his friendship and assistance as promoter.

Mr W.R. Kleyn of Eskom Bloemfontein - Electricity Delivery Systems Support, for supplying the load data.

SUMMARY

Because power generation relies heavily on electricity demand, consumers are required to wisely manage their loads to consolidate the power utility's optimal power generation efforts. Consequently, accurate and reliable electric load forecasting systems are required.

Prior to the present situation, there were various forecasting models developed primarily for electric load forecasting. Modelling short term load forecasting using artificial neural networks has recently been proposed by researchers.

This project developed a model for short term load forecasting using a neural network. The concept was tested by evaluating the forecasting potential of the basic feedforward and the cascade forward neural network models.

The test results showed that the cascade forward model is more efficient for this forecasting investigation. The final model is intended to be a basis for a real forecasting application. The neural model was tested using actual load data of the Bloemfontein reticulation network to predict its load for half an hour in advance.

The cascade forward network demonstrates a mean absolute percentage error of less than 5% when tested using four years of utility data. In addition to reporting the summary statistics of the mean absolute percentage error, an alternate method using correlation coefficients for presenting load forecasting performance results are shown.

This research proposes that a 6:1:1 cascade forward neural network can be trained with data from a month of a year and forecast the load for the same month of the following year. This research presents a new time series modeling for short term load forecasting, which can model the forecast of the half-hourly loads of weekdays, as well as of weekends and public holidays. Obtained results from extensive testing on the Bloemfontein power system network confirm the validity of the developed forecasting approach. This model can be implemented for on-line testing application to adopt a final view of its usefulness.

LIST OF ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Networks
BP	Back Propagation
DLC	Direct Load Control
Eskom	South African electricity public utility, established in 1923 as the Electricity Supply Commission
MAD	Mean Absolute Deviation
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
MSE	Mean Sum of squares of the network Errors
MSW	Mean of the Sum of squares of the network Weights and biases
NMD	Notified Maximum Demand
RMSE	Root Mean Squared Error
SSE	Sum of the Squared Errors
SSW	Sum of the Squared Weights
STLF	Short Term Load Forecasting

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Electric load forecasting is one of the principal functions in power systems operations. The inspiration for accurate forecasting lies in the nature of electricity as a service and trading article; bulk electricity cannot be stored. For any electric utility, the estimate of the future demand is necessary to manage the electricity production in an economically reasonable way so as to meet peak demands without unnecessary load shedding or power cuts [46].

System loads vary through daily and seasonal cycles, creating difficult operating problems. Forecasting allows a utility company to schedule load shedding without affecting the load generation capacity of the national grid.

Different statistical load forecasting models have been developed. Practically, there is a subtle difference in conditions and needs of every situation where there is a need for electric load demand. This has a significant influence on choosing the appropriate load forecasting model. The results of the methods used to forecast electric load, presented in publications, are usually not directly comparable to each other.

Some recently reported forecasting approaches are based on neural network techniques. Many researchers have presented good results. The attraction of these forecasting methods lies in the assumption that neural networks are able to learn properties of the electric load curve, which would otherwise be hardly discernable.

Research on neural network applications in load forecasting is continuing. Results using different network model structures and training algorithms is not complete. To make use of these techniques in a real application such as Bloemfontein City, a comparative analysis of the properties of two different neural models seems appropriate.

1.1 Problem statement

“To develop an efficient Artificial Neural Network-based model for Short Term Load Forecasting and apply this model to a real life case study to evaluate the performance of the proposed approach and provide a one half-hour ahead forecast for the Bloemfontein power system network”.

Any large network’s electric load to be forecasted is a pseudo-random, non-stationary process composed of thousands of individual components. The variety of possible approaches to do the forecasting is extensive. One possibility is to take a macroscopic view of the problem by trying to model the future load as a reflection of its earlier behaviour with time series prediction using an appropriate statistical tool.

This still leaves the field open to several diverse solutions. Because of the pseudo-random nature of the load, the only objective method to evaluate this approach is through experimental confirmation.

Load forecasting is an important topic in Energy Markets for planning the operations of load producers. Bloemfontein Municipality (CENTLEC (Pty) Ltd) needs to frequently review the Notified Maximum Demand (NMD), declared at 300 MW in 2009, for it to avoid possible penalty charges payable to Eskom for exceeding the contractual maximum demand threshold.

Short Term Load Forecasting (STLF) can be used to address this problem. It has been mentioned very often as a solution in power systems literature in the late 1990s. One reason is that recent scientific innovations have brought in new approaches to solve the problem. The development in computer software and technology has broadened the possibilities for solutions, working in a real-time environment. Genetic algorithms, neural networks, expert systems and fuzzy logic are some of the software tools used to find possible solutions in predicting the electric load “up front” [27, p.249].

1.2 Objectives of thesis and execution of the project

This work entails the investigation of the design of an accurate, robust neural model that can be used to estimate the future electric load by using the historical electric load data patterns to train such a network.

Analysing the objective specifications

1. Using a forecasting time horizon of one month, which means that a neural network is built and trained for each month of the year if forecasting deployment is eventually realized.

2. The same network structure, training algorithm and data input structure is to be used for forecasting the load in each of the twelve months of a year.

3. Experimenting and testing a short term dynamic, half-hour ahead updating, neural network forecaster for each of the late autumn and winter months of May, June and July. The final neural network structure can then be used repeatedly, for training and to predict the electric load for the rest of the year; one network for each month of the year.

4. Each of the three final networks must be trained with the least possible amount of historical load data, using the minimum quantity of inputs to predict the load, with a half-hour lead time. For the training and testing of each network, each week of a month starts off on a Saturday morning at 00:00:00 and ends seven days later on Friday night at 23:59:59.

5. Evaluating suitable network training algorithms and adjusting the topology elements of the neural network to keep the Mean Absolute Percentage Error (MAPE) below 5% during forecasting.

The research procedure of this project is represented by the flowchart in Figure 1.1

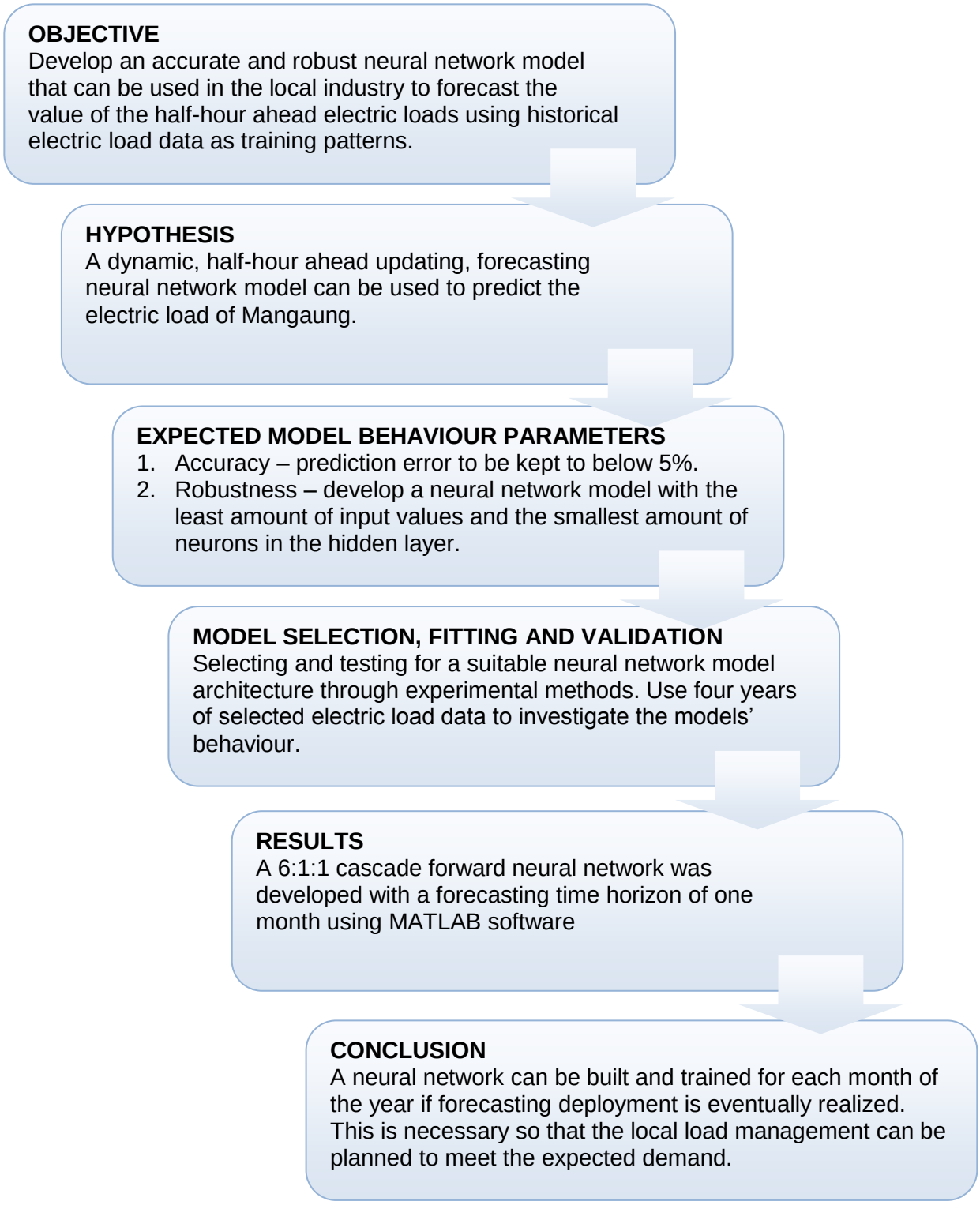


Figure 1.1: Research phases during execution of the project

1.3 Structure of this thesis

The essential layout of this project is shown in Figure 1.2

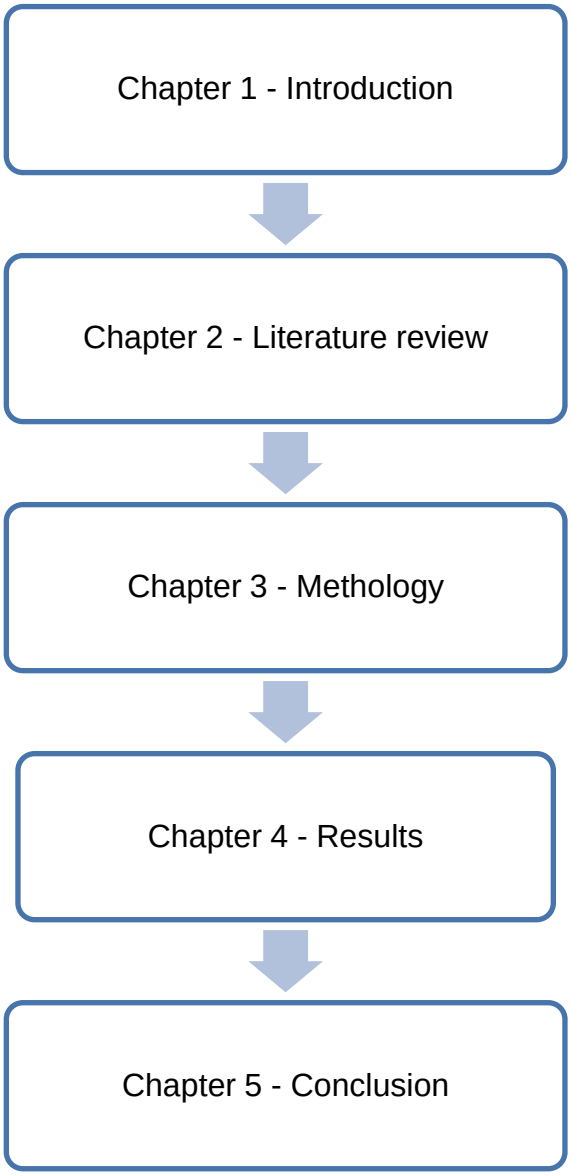


Figure 1.2: Layout of the thesis

An introduction to electric load forecasting theory with specific reference to short term load forecasting opens Chapter 2. The different shapes of load curves in a typical load pattern, planned load shedding, and a blackout are compared. Neural network theory with reference to the backpropagation and cascade correlation

model, Levenberg-Marquardt minimization training algorithm and performance validation are also considered.

Since the aim of the project was not to investigate the mathematical theory behind neural networks, but its implementation in a working load forecasting model, little theory in this regard was covered. Only those aspects which influence the load forecasting model were identified.

The feedforward and the cascade forward neural network models were chosen for comparison according to:

- their architecture;
- backpropagation and Levenberg-Marquardt training algorithms;
- and performance using the mean absolute percentage error as a benchmark.

Extensive tests were done for three months of the year: May, June and July, for the three running years 2007, 2008 and 2009, to obtain a suitable model. This process is described extensively in Chapter 3.

Chapter 4 provides an exposition of how the final 6:1:1 neural cascade forward network was trained and its forecasting potential evaluated for the three months of the year: May, June and July, for the three running years 2007, 2008 and 2009. Evaluation was done using the mean absolute percentage error (MAPE) and correlation coefficients to measure and evaluate this cascade networks' performance. The MAPE, Daily Peak MAPE and linear regression plots, using July 2009 as a case study, are discussed in more detail. All these results are presented in this chapter.

The outcome of this research is concluded in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Electric load forecasting

Numerous books have been written about electric machinery, transmission systems and the protection technology needed to produce and deliver electric power. In comparison, little has been recorded about electric load forecasting.

A forecast is a prediction of some future event. The importance of forecasting lies in the prediction of changes in load behaviour, which has a significant effect on various types of planning and decision-making processes in power systems operations and financial management.

Despite the large range of problem situations that involve forecasts, there are only two broad types of forecasting techniques: *qualitative* methods and *quantitative* methods.

Qualitative forecasts are often used in situations where there is little or no historical data on which to base the forecast. *Quantitative* forecasting techniques apply historical data to a forecasting model. The model recognizes patterns in the historical data and expresses a linear or non-linear relationship between the previous and current values of the data. Then the model projects the recognized data patterns into the future. The three most widely used *Quantitative* forecasting models are regression models, smoothing models and time series models [32, p.4]. Forecasting electric load curves are dependent on the assembly of historical load data so it will fall in the category of *quantitative* forecasting.

2.1.1 Forecasting the load curves

Normally, electric load curves alternate with time from a slow random swing to rapidly repeated pulses through daily, weekly, monthly, and annual cycles, creating continuously varying load curves from season to season. The

predictability of these load curves can be modeled with a wide range of roughly defined forecasting time periods, which includes:

Long term forecasting

The forecasting of bulk power for periods, years into the future. Usually, only peak loads are predicted.

Medium term load forecasting

The forecasting of bulk power for periods six months to one year ahead. Usually peak loads are forecasted and sometimes off-peak load values are also considered.

Short term load forecasting

This refers to much less than a “year ahead” forecast, and occasionally to a “one day ahead” forecast. The short term load forecasts differ from their long term counterparts in that peak forecasts assume less importance than forecasts at specific times; e.g., hour by hour forecasts will assume greater importance.

Load forecasting, especially Short Term Load Forecasting (STLF), is dependent on a combined load made up of different components. As an example, the load requirements can be decomposed into *residential*, *industrial* and *commercial* components [12, p.273] where the:

- *Residential* demand is proportional to a change in daily temperature;
- *Industrial* loads are sensitive to economic and political factors; and the
- *Commercial* demand is sensitive to the “day of the week” and occasional holidays [22, p.302].

2.1.2 Quantitative short term load forecasting using time series data with a neural network model

Forecasting is usually based on identifying, modeling, and extrapolating the patterns found in historical data. Because historical data do not change dramatically very quickly, statistical methods, e.g. neural networks, are useful for short term forecasting [16, p.302].

It should be noted that power system transients such as sudden overloads due to short circuits, line harmonics or blackouts cannot be predicted with a forecasting model.

Most forecasting problems involve the use of time series data [36], which are amongst the oldest methods applied in load forecasting [19, p.904]. An electric load pattern is principally a time series [3, p.2]. A time series can be described as a sequential set of hourly, daily, or weekly data measured over regular intervals. A time series forecasting takes an existing series of data x_{t-n}…,x_{t-3}, x_{t-2}, x_{t-1}, x_t and forecasts the x_{t+1} data value where:

x_{t+1} is the target value of x that we are trying to model;
x_t is the value of x for the previous observation where we use
x to indicate an observation and t to represent the index of the time period.

As mentioned in par. 2.1, *quantitative* forecasting techniques involve the use of time series data (historical) and a forecasting model, i.e. an artificial neural network capable of representing complex nonlinear relationships. The model summarises patterns in the data and expresses a statistical relationship between previous and current values of the variable x, in this case. Patterns in the data can then be projected into the future, using this model [32, pp.1-5].

In neural network time series forecasting, two approaches are available in updating forecasting models over time: the *rolling window* approach and the *sliding window* approach.

The *Rolling window approach* uses a constant starting point and all the available data to train the neural network to anticipate the next value(s) and is not emphasized here. A disadvantage of this approach is that it is not very likely to give optimum results when the time series process is changing over time.

The *Sliding window approach* uses a changing starting point and a set of the latest observations to train the neural network to predict the next value(s). The sliding window approach adds a new training value and disregards the oldest one

from the training sample set which is used to update the neural model. The input sample set size, presented to the neural network, stays fixed as it slides forward with time.

It is not easy to determine an appropriate input sample set size that is useful for training the forecasting network. Too small a sample input set may restrict the power of the neural network to do proper forecasting by not adjusting the network parameters properly. However, the sliding window approach, with the changing starting point in the training sample set, is more suitable to contemplate changes occurring in the underlying process of the time series [33, pp.3-8].

As an example of a *sliding window technique*, Figure 2.1 shows a standard, trained back propagation neural network performing time series prediction using four equally time spaced input data points, sliding over the full training set of electric load data to predict the next output data point value [6].

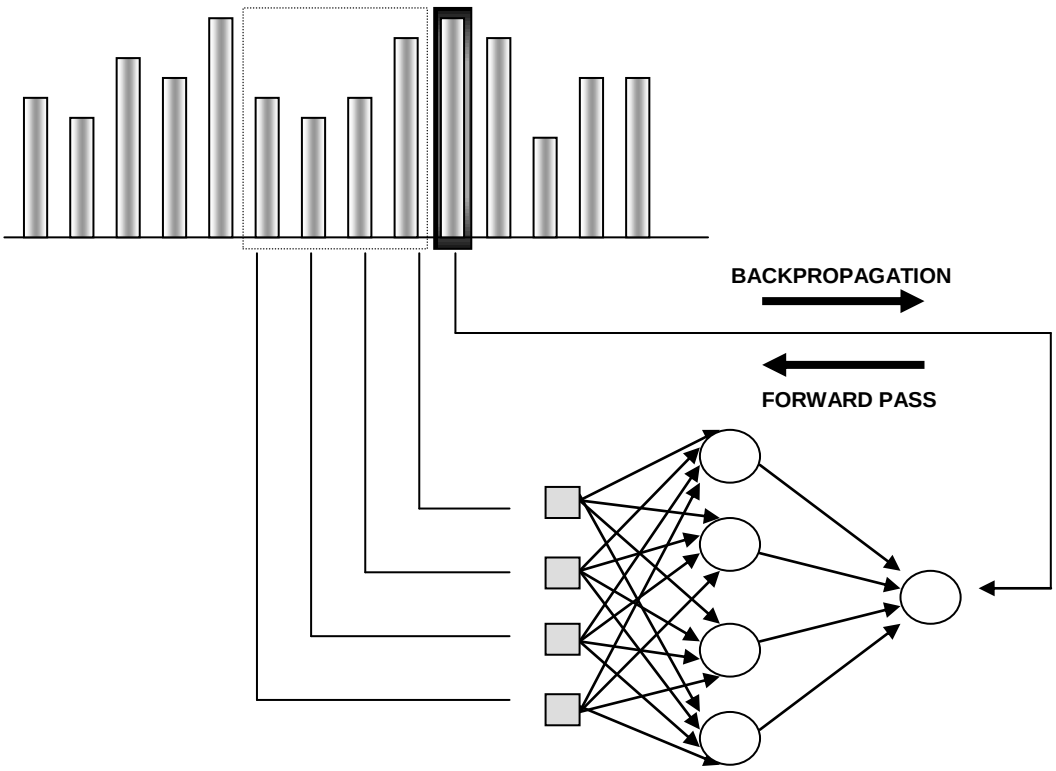


Figure 2.1: Time series prediction using the "sliding window" approach

The rate at which the samples are taken will dictate the maximum resolution of the model. Once the network is trained with this set the same technique is used

with new data points to predict the future electric load. It is not always true that the model with the highest resolution will give the best forecasting performance.

Currently, the available power generation capacity in South Africa is running in short supply. Load shedding needs to be applied frequently across the country to ensure the security of electricity supply, especially during the winter peak electricity demand of May, June and July. So, forecasting short term load can play an important role in power system operations decisions, taken when load shedding becomes a priority in the colder months of each year.

2.1.3 Load shedding

Occasionally, a misconception arises when discussing everyday load, load shedding and a blackout. One first has to understand the difference between a:

- *typical load pattern*,
- *planned load shedding* and a
- *blackout* event

before considering the useful application of electric load forecasting.

A weekly load characteristic for Bloemfontein - July 2009 will be taken as an example to graphically illustrate the difference between the three concepts:

A typical load pattern

In Figure 2.2 it can be seen that the valleys in the daily pattern fluctuate little in magnitude and the base load floats at more or less 125 MW. The peak load levels for each day, indicating the Maximum Demand used for that day varies between 250 MW and 280 MW depending on the day-type one is looking at. An underlying repeating cyclic pattern, with random magnitudes at random periods, can be observed in the data.

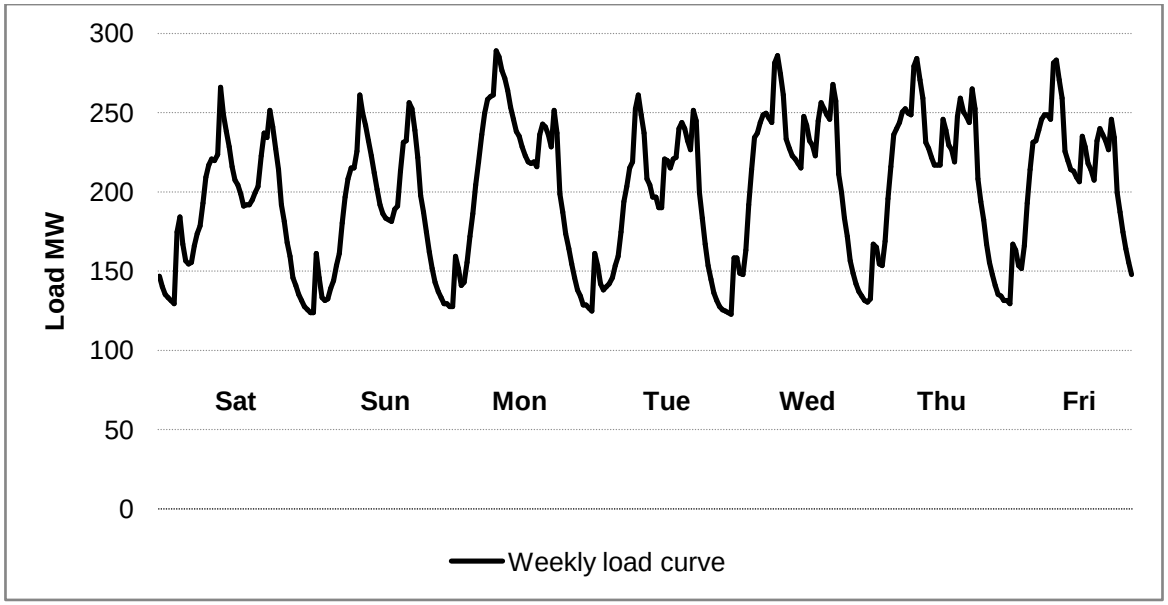


Figure 2.2: A typical weekly load pattern

Planned Load shedding

Load shedding, also known as a rolling blackout, occurs when a power company cuts off electricity to selected areas to save power. This is a controlled way of rotating generating capacity. As an example, an area blackout can vary between two to six hours before the power is restored and the next area is cut off. Hospitals, airport control towers, police stations, and fire departments are often exempt from these rolling blackouts. One would be tempted to think that the load shedding curve would have the following shape, discussed next, which for practical reasons, are not predictable.

Example of an impracticable load shedding pattern

Figure 2.3 is a “clipped” version of the previous weekly pattern. It illustrates a hypothetical situation where the Maximum Demand is continuously controlled by cutting off the peak load to a maximum level of approximately 250 MW. In reality this type of load control is impractical to achieve.

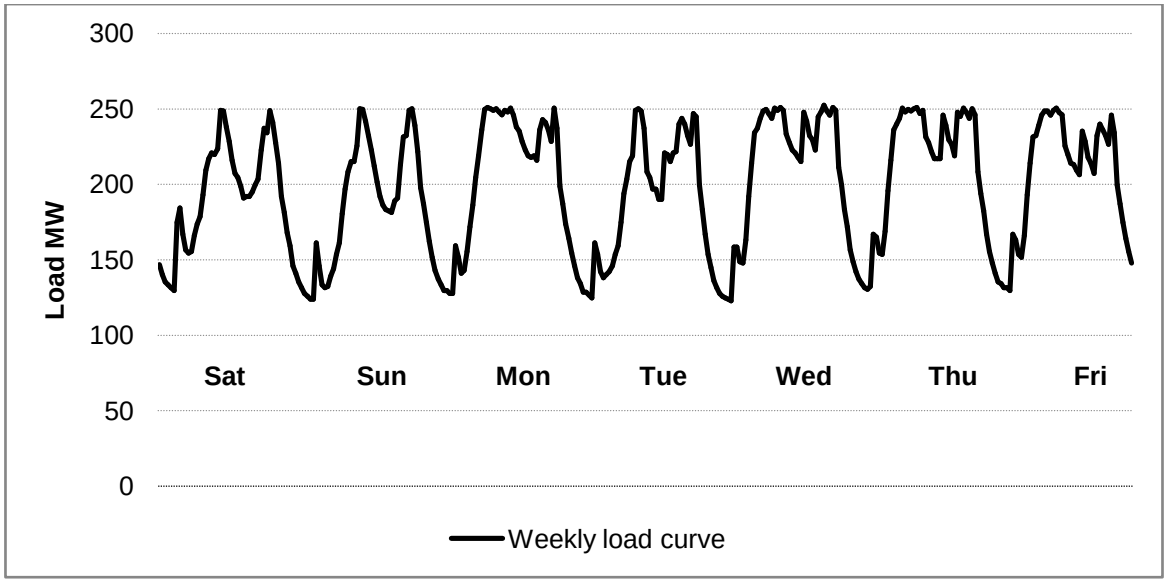


Figure 2.3: An impractical weekly load shedding pattern

Example of a realistic load shedding pattern

Figure 2.4 shows a practical situation. The load shedding curve closely follows the normal electric load demand pattern with a smaller peak magnitude in most places and can be presented to a load forecasting model for further research.

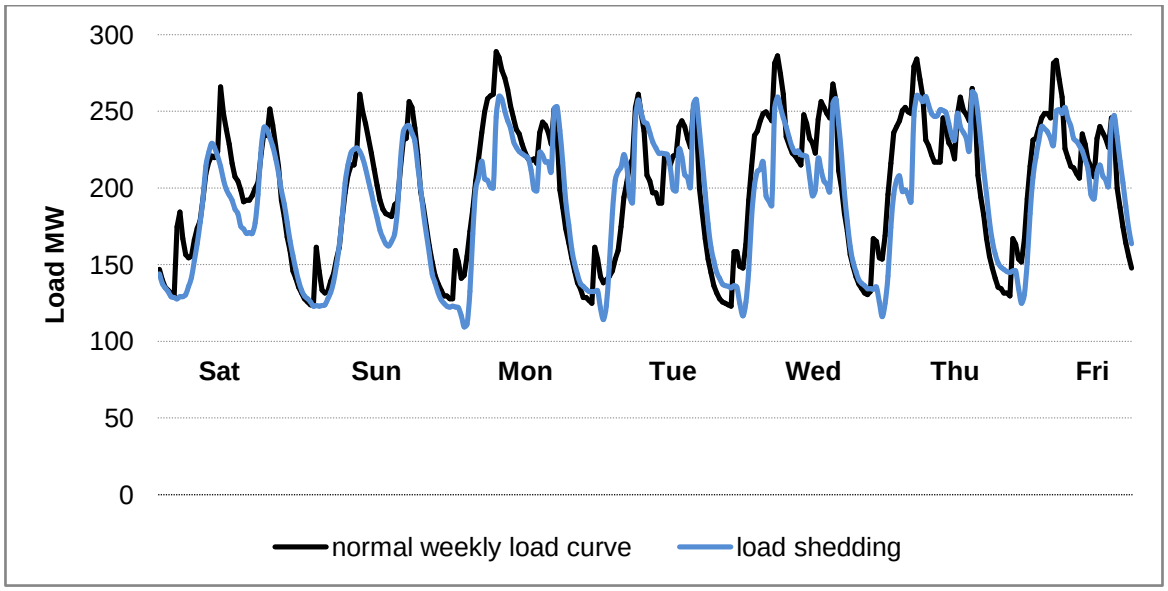


Figure 2.4: An actual weekly load shedding pattern

Example of a blackout event

Figure 2.5 shows a total power cut during a part of Sunday night and Monday. This type of uncontrolled power outage can affect a whole city or a big part of a country due to a network overload, act of Nature, terrorism or bad maintenance planning. If it lasts for more than a day or three, it can have a displeasing long term impact on transportation, water supply, communication, the local economy and the crime rate.

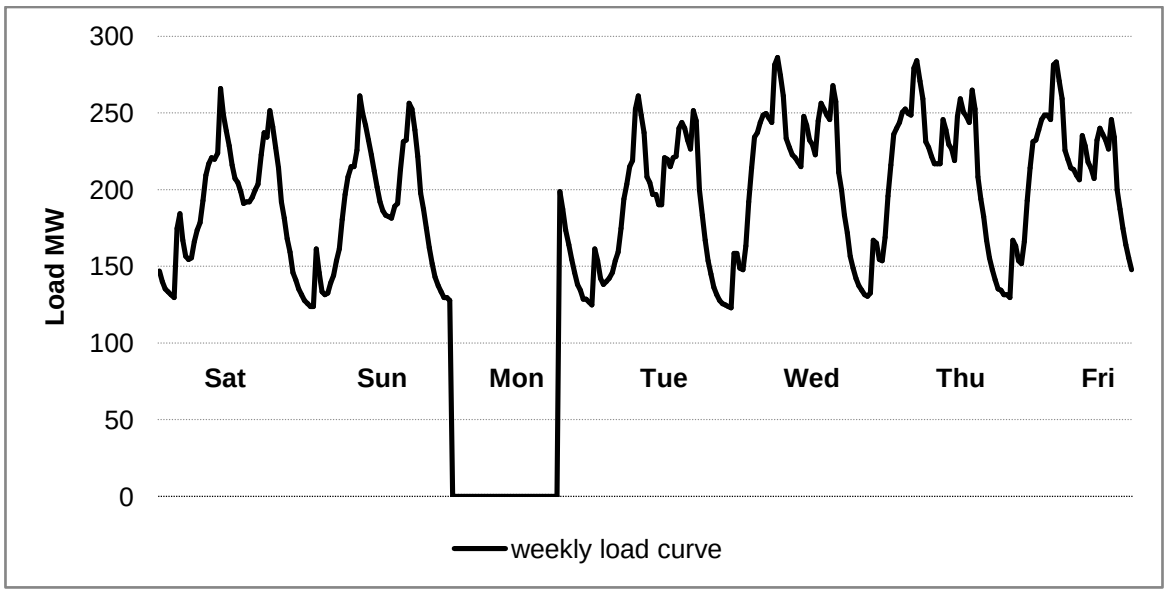


Figure 2.5: An example of a blackout occurring on a Monday

To summarise:

1. A typical daily load varies in a cyclic pattern and can be presented to a forecasting model.
2. Load shedding affects only a part of the populated area and:
   - can be classified as an example of short term load variation.
   - on average the load drops to below its previous maximum daily peak load level values.
   - a neural network can follow this type of change in load.
3. A blackout affects the whole city or region and it is totally unpredictable.

Economic and demographic factors in this region change very slowly over very long periods of time so medium- and long term forecasting is not applicable to critical factors like daily load shedding. However, short term load forecasting plays an important role when a decision is to be made during times of load shedding.

Some relatively recent forecasting models are available for research to predict the shape of electric load curves. The development of Artificial Intelligence (AI), especially Artificial Neural Networks (ANN) can be applied for research to model short term load forecasting.

Artificial neural networks have been extensively used as time series predictors; these are usually feed-forward networks that make use of a sliding window over the input data sequence. Using a combination of a time series and a neural network prediction method, the past events of the load data can be explored and used to train a neural network to predict the next load point.

There are many neural network training methods and architectures that can be used to model a forecasting problem, but only the applications and principles used in this investigation will be discussed next.

2.2 Some principles of artificial neural networks

2.2.1 Introduction

Artificial neural networks, “roughly” based on the architecture of the brain, are rising as an exciting new information-processing concept for AI systems. Their working mechanism is totally different from conventional computers in the following way:

- They do not need to be programmed, as they can learn from examples through training.
- They can produce correct outputs from noisy and incomplete data (fault tolerant), whereas conventional computers usually require correct data.

- If one or more neurons or communication lines are damaged, the network degrades ‘gracefully’ (that is, in a progressive manner), unlike sequential computers which can fail catastrophically after isolated failures.
- They are economic to build and to train.

Due to these differences in working mechanisms, considerable interest have been created in the possibilities for applying neural networks in engineering, and have resulted in a great deal of research over the last few years. Some of the claimed advantages are exaggerated, but others are certainly proven, and neural networks are slowly becoming a standard technology for engineers [26, p.95], [2, p.31].

In the 1980’s research in neural networks increased substantially because personal computers became widely available [55, p.41]. Two new mathematical tools were responsible for the rejuvenation of neural networks in the 1980s:

- The use of statistical mechanics explaining the operation of a certain class of network;
- The discovery of the back-propagation algorithm for training multi-layer perceptron networks.

These new developments recharged the interest in the field of neural networks.

2.2.2 Moving to artificial neural network structures

An artificial neural network consists of a number of very simple processors, also called neurons, which are analogous to the biological neurons in the brain. The neurons are connected by weighted links passing signals from one neuron to another.

The input signal is transmitted through the network neurons to its outgoing connection. Each neuron connection can split into a number of branches, the weighted links, to transmit the received signal. The links can amplify or weaken

each signal as it passes through, according to its allocated weight value. A simple network is illustrated in Fig 2.6.

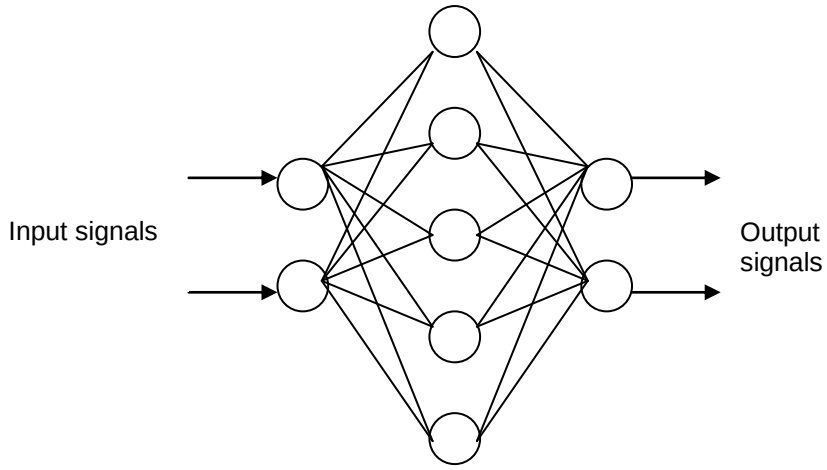


Figure 2.6: Layout of a plain artificial neural network

More complex systems will have more layers of neurons with some having increased layers of input neurons and output neurons.

Next, the behaviour of a neural network is discussed by first showing the essential features of neurons and their weighted links. However, because our knowledge of neurons is incomplete and our computing power is limited, this mathematical model is a crude portrayal of real networks of neurons.

2.2.2.1 A general mathematical neuron model - the perceptron

A specific artificial mathematical neuron, the perceptron, is considered, based on McCulloch and Pitt's model. The perceptron is the simplest form of a neural network and consists of a single neuron with adjustable weighted links and an activation function called a "hard limiter", sign- or a threshold function.

This perceptron-neuron, shown in Figure 2.7, computes the weighted sum of the input signals and adds the result to a bias value, θ. This constitutes the net input **X** that will be presented to the activation function. This activity is referred to as a linear combination.

If the net input is less than a threshold value (e.g. zero on the x-axis), the neuron output is -1. But if the net input is greater than or equal to the zero threshold value on the x-axis, the neuron becomes activated and its output value, **Y,** becomes +1.

This situation is referred to as the neuron having “fired”. The perceptron-neuron in Figure 2.7 uses the “sign” or “threshold” activation function.

$$X = \sum_{i=1}^{n} x_i w_i + \theta_i \qquad Y = \begin{cases} +1, \text{ if } X \geq 0 \\ -1, \text{ if } X < 0 \end{cases}$$

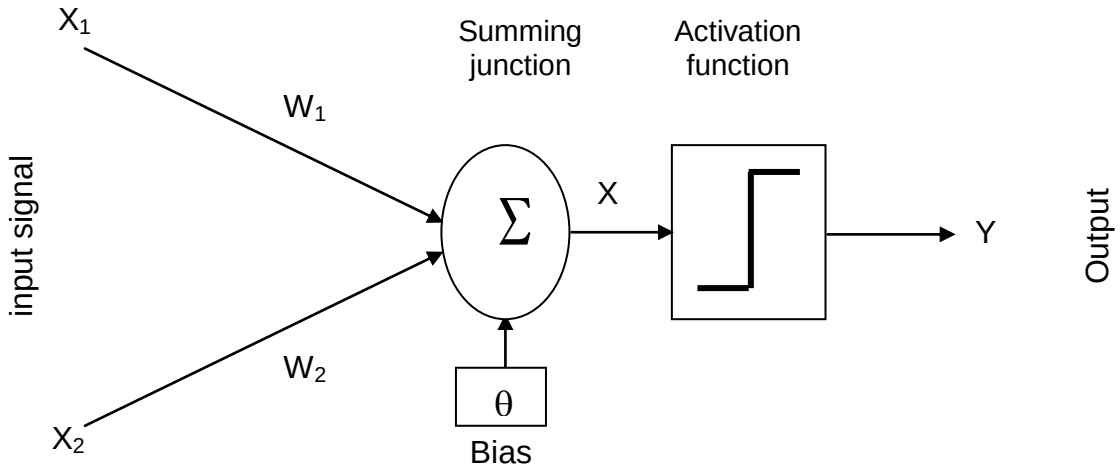


Figure 2.7: A single layer, two input, linear threshold perceptron where X is the net input and Y is the neuron output

In Figure 2.7 it can be seen that the combination of the weighted links and the activation function are used to control the amplitude of the output. An activation function is a mathematical function used by a neural network to scale numbers to a specific range.

There are many activation functions that one can choose from and each one has its own special merits [5, p.44]. The four basic activation functions are discussed next.

2.2.2.1.1 Basic activation functions

Many different types of linear or nonlinear activation functions (or limiters) can be designed [18, pp.2-6]. Choosing the correct function depends on the particular problem to be solved. A basic reference model for the presentation of activation functions is shown in Figure 2.8 [7, p.21].

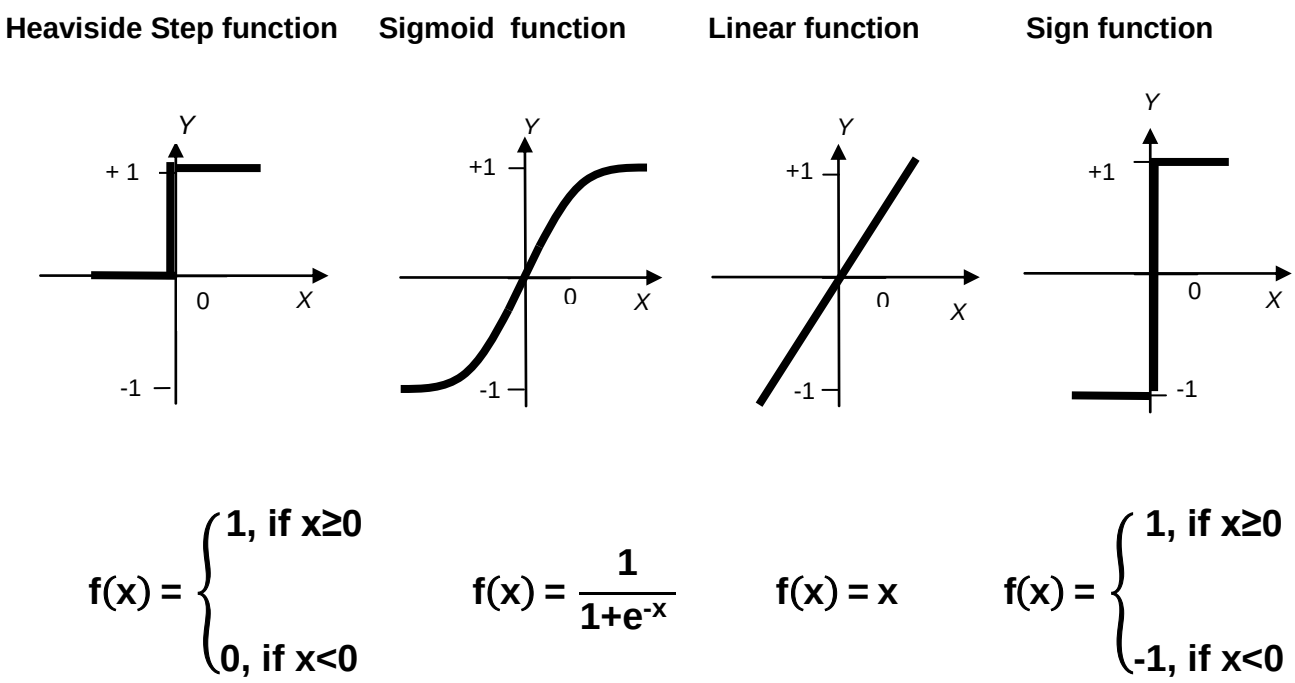


Figure 2.8: Four basic types of activation functions

1. The **Heaviside step function** and the **Sign function** are binary functions that hard limit the input to the function to either a 1 or a 0 for the binary type, and a -1 or a 1 for the bipolar type.
2. **Sigmoid functions** are the most commonly used function in the construction of artificial neural networks [21, p.14]. They are also known as “squashing functions”, thought of as a slowly softened Sign function [17]. This type of function is nonlinear and therefore differentiable everywhere, causing greater weight change activity to take place for neurons where the output is less certain (i.e. close to 0.5 where the slope

is the steepest) than those in which it is more certain (i.e. close to 0 or 1) [1, p.138]. This type of nonlinearity is very important, or else the input-output relationship of the network will be reduced to that of a single layer perceptron network [21, p.157]. Interestingly, artificial neurons containing sigmoidal functions resemble a type of biological neuron found in the brain [56, p.5].

3. The **Linear limiter** is a continuous function where the output purely reflects the input. This function is usually combined with other functions for specific tasks at hand, i.e. linear networks.

2.2.2.2 Connecting perceptrons into structures called network architectures

A single perceptron is not very useful because of its limited mapping ability. Multiple perceptrons can be connected as building blocks of a larger, much more practical structure called an artificial neural network or a complex nonlinear mapping tool. This type of neural net can deal with complex nonlinearities in a fairly general way [56, p. 2]. Many kinds of neural networks exist today and variations of older structures are invented regularly [47, p.14].

There is a large variety of neural network structures to choose from. The selection or design of a particular network depends on the uniqueness of the intended application. The following aspects should be given careful consideration:

- How well can the network model the system?
- How efficient can the network structure and parameter values be adapted?
- Amount of accuracy required;
- Overall problem complexity;
- System stability;
- Is off-line or on-line implementation required?

Other important questions to consider when setting up a neural network are:

- Should the network be fully or partially connected?
- Should there be any sub-structures within the neural network?

- How many layers are needed?
- How many neurons should be in each layer?
- What type of activation function should be used [23, p.28]?
- Can one obtain a sufficient historical data set from the system to be predicted to train the neural network?

Once the network connection structure has been decided upon, the following step would be to adjust the network biases and weights by a suitable training method so that the network input/output relationship is very close to that of the system to be modeled.

The well-known feedforward connection structure will be discussed next. The term “Feedforward” means that the input signal moves from the input to the output layer.

2.2.2.3 Multilayer feedforward network connection structure

All feedforward neural networks have an input layer and an output layer, but the number of hidden layers may vary. This class of layered network has one or more hidden layers presented in the three-layer net, as shown in Figure 2-9.

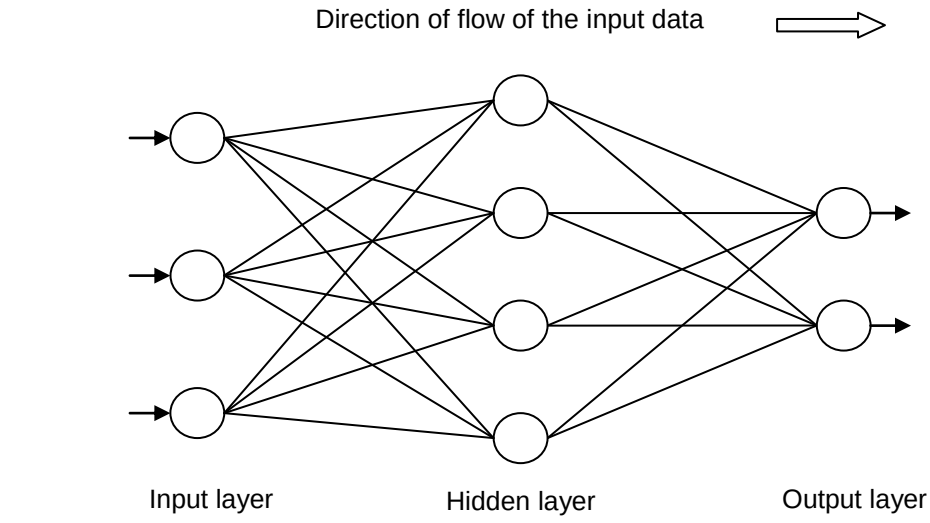


Figure 2.9: A fully connected multilayered feedforward network

This network structure is referred to as a 3:4:2 network because it has 3 inputs, 4 hidden neurons and 2 output neurons. This network is said to be fully connected in the sense that every neuron in each layer is connected to every other neuron in the next forward layer.

If some of the communication links between the neurons are not part of the designed network then it is referred to as being partially connected. The “hidden layers” can communicate backwards with the input source nodes and forward with the output layer in some determined manner. By adding a hidden layer or two, the neural net can handle some highly nonlinear problems that would be difficult to describe mathematically.

The most efficient number of hidden layers depends in a complex way on the:

- number of input and output units;
- number of training cases;
- amount of noise in the input signal;
- complexity of the function or classification to be learned;
- network’s architecture;
- type of hidden layer activation function;
- training algorithm;
- regularisation.

2.2.3 Training a neural network

Training is as essential for a neural network as programming is for a computer to function properly. The activation function of a neuron is fixed when the network is set up and the set of signals to which it is supposed to react is fixed. During the training phase, the only adjustment that can be made to individual neuron’s input/output behaviour is to its own input weights and biases [43]. Therefore, the

correct choice of a learning algorithm is an important issue in network development.

One of the problems that can occur during training is over-learning or over-fitting. The desire for the neural network is to provide a correct mapping of the test data and maintain the ability to generalize for new data. The results of over-fitting can already be visible in the test stage [24, p.274].

A critical goal during training is to find a network large enough to learn the task but small enough to generalize [20, p.53].

In this project, the neural network must be able to learn a model of the electric load which it will be predicting. It must be able to maintain the model adequately and reliably when used to predict real load variations so as to achieve the particular goals of the application of interest [21, p.24].

For different network structures, different training algorithms have been developed. Neural nets are classified according to their corresponding training algorithms:

- fixed weight networks;
- supervised networks; and
- unsupervised networks.

No learning is required for a fixed weight network, so the common training algorithms are supervised and unsupervised learning [54, p.249].

2.2.3.1 Supervised training

The prevailing current of thought of artificial neural network development has been focused on supervised learning networks. To commence learning this type of network, data needs to be gathered and consists of pairs of input/output training sets. First the training sets are presented to the network. Then, through comparison of the actual and desired response (the error) at the output stage,

adjustments of the weights and biases of the whole parameter space are made. In this way, learning or proper classification is facilitated with the help of a teacher, as shown in Figure 2.10 [37, p.14].

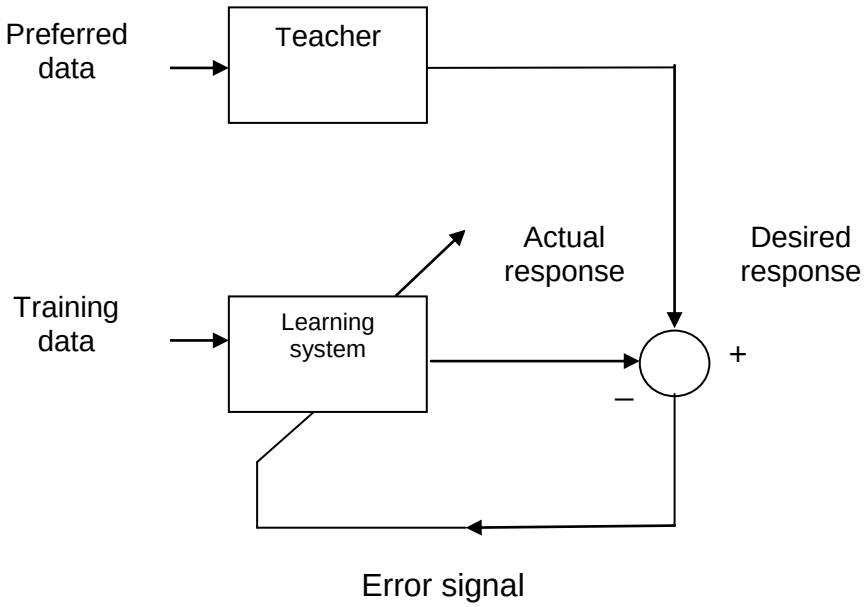


Figure 2.10: Block diagram of learning with a teacher

The special quality of a network exists in the values of the bias and weights between any two of its neurons, so we need a method of adjusting the bias and weights to solve a particular mathematical problem. For feedforward networks, the most common training algorithm is called Back Propagation (BP). A BP network learns by example, that is, we must provide a learning set that consists of pairs of data sets (input examples and the known-correct outputs for each case). So, we use these input-output examples to show the network what type of performance is expected, and the BP algorithm adjusts the weights and biases in the network after the input/output pairs are compared.

2.2.3.1.1 Supervised training using backpropagation

The *backpropagation* training algorithm was first described by Rumelhart and McClelland [42] in 1986; it was the first convenient method for training neural networks. The original procedure used the *gradient descent* algorithm to adjust the weights toward convergence, using the gradient (the direction in which the performance index will decrease most rapidly). The term “backpropagation” or

“backprop” is often used to indicate a neural network learning algorithm using the gradient descent algorithm.

To make practical forecasts for this work, the different neural networks were trained using appropriate data series sets. Training examples in the form of (*input***,** *target*) pairs of vectors are extracted from the data series. The *input* and *target* vectors are equal in size to the number of network inputs and outputs, in the order mentioned.

For every training example presented to the network, the algorithm follows a sequence of steps to adapt each neuron’s weight and bias values before the following training example is presented to it. The sequence is as follows:

1. The learning procedure works in small adjustable incremental steps: one of the example pairs is presented to the network in a forward pass, and the network produces some outputs based on the present state of its bias and weights (initially, the outputs will be random because the values of the biases and weights were chosen randomly). The output of the first layer becomes the input to the second layer and so forth until a final value is available at the output layer, the network output.
2. The final output value is compared to the target value. This is the difference (the error) between the desired value (the target that the network will need to learn) and the actual network output value obtained from the input training examples in 1.
3. This error is backpropagated from the output layer to the first hidden layer through the network, calculating the gradient (vector of derivatives) of the change in error with respect to the changes in weight values.
4. All the network neuron weights and biases in proportion to the error are updated.
5. Steps 1 to 4 is repeated until the difference, measured using a performance index (e.g. the mean square error) has been reduced to a preset value.

Each complete cycle of these steps is called an *epoch*. Training will end when a maximum number of epochs or a network output error limit is reached as programmed by the network user.

Training can be time consuming, depending on the number of example sets, network size and topology, epoch- and error limits.

Because the error information is propagated backward through the network, this type of training method is called *backward propagation*. A mathematical description of the above sequence will be explained next.

Backpropagation algorithm

Kindly abusing the much quoted work of Hagan, Demuth and Beale, “Neural Network Design” [18, pp.11-7 to 11-13], a brief explanation of a more accepted approach to the backpropagation training algorithm is shown where the notation from Hagan is represented as follows:

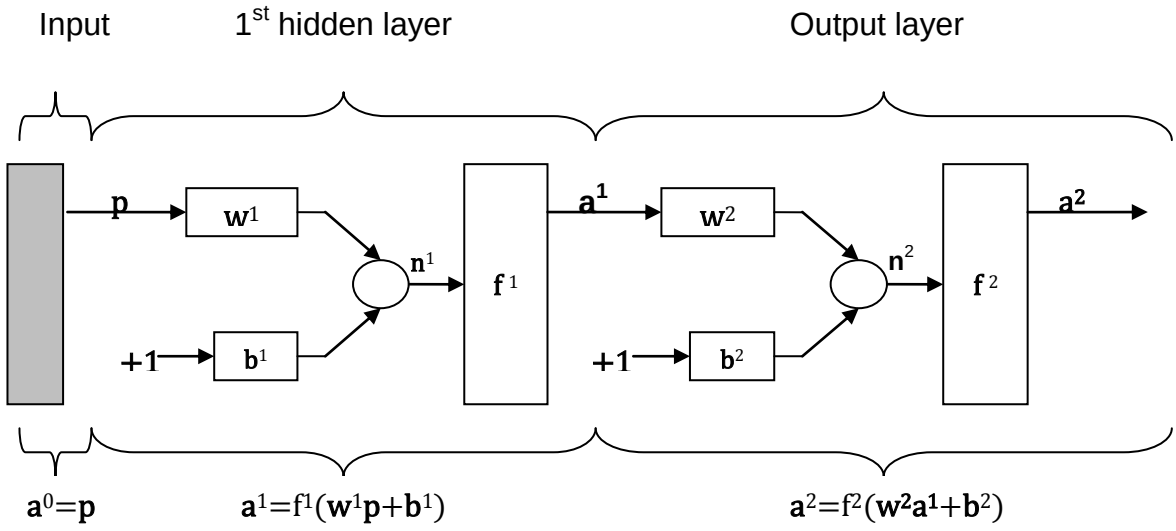
Scalars: small *italic* letters…a, b, c

Vectors: small **bold** letters...$\mathbf{a}, \mathbf{b}, \mathbf{c}$

Matrices: capital **BOLD** letters…$\mathbf{A}, \mathbf{B}, \mathbf{C}$

Vector: column of numbers.

Row vector: a row of a matrix used as a vector



$$a^2 = f^2\,(w^2\,(\,f^1(w^1p+b^1)\,)+b^2)$$

Figure 2.11: Two layer network shown in abbreviated notation where the output vector of the first (hidden layer), a^1, becomes the input vector to the second (output) layer

Consider an example of a multilayer feedforward network as in Figure 2.11.

Step 1 - Forward propagation

For an M layer network the system equations in matrix form are given by

$$a^0=p, \qquad \textbf{(2.1)}$$

for the input layer where the input vector elements a^{m+1}enter the network through the weight matrix W and

$$a^{m+1}= f^{m+1}(W^{m+1}a^m+b^{m+1}) \text{ for } m = 0,1,...,M\text{-}1, \qquad \textbf{(2.2)}$$

is the transfer function vector for the output layer, so that the last network layer outputs, considered as the network outputs, are:

$$a = a^M, \qquad \textbf{(2.3)}$$

where

p= network input vector and

$\mathbf{a}^0, \mathbf{a}^1 \ldots \mathbf{a}^M$ = network output vectors

Step 2 - Backpropagation

The task of the network is to learn associations using a training set of prototype input-output examples:

$$(p_1,t_1)\,,(p_2,t_2)\,,\ldots,(p_q,t_q)\,, \qquad \textbf{(2.4)}$$

where p_q is a network input and t_q is the matching target output. For each input p_q applied to the network, the network output (a) will be compared to the target output (t). The difference will be the error (e)**,** used to calculate the performance index $F(\mathbf{x})$ using the mean square error (MSE):

$$F(\mathbf{x}) = E\,[\,e^2\,] = E\,[\,(t-a)^2\,]. \qquad \textbf{(2.5)}$$

Where **x** is the vector of network weights and biases and E[ ] donates the expected value where the expectation is taken over all sets of input-target pairs. The algorithm adjusts the network parameters to minimize the MSE.

For a multiple output network it generalizes to vectors

$$F(\mathbf{x}) = E\,[\mathbf{e}^T\mathbf{e}] = E\,[(t-a)^T(t-a)]. \qquad \textbf{(2.6)}$$

So the approximate MSE (Single sample) or performance index for the network is

$$\hat{F}\,(\mathbf{x}) = \big(\mathbf{t}(k)-\mathbf{a}(k)\big)^T\big(\mathbf{t}(k)-\mathbf{a}(k)\big) = \mathbf{e}^T(k)\mathbf{e}(k) \qquad \textbf{(2.7)}$$

where the expectation of the squared error is replaced by the squared error at iteration k.

The *steepest descent algorithm* searches the parameter space (adjusting the weights and biases) in order to reduce the mean square error $\hat{F}\,(\mathbf{x})$. Updating the weights at each iteration k is

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \frac{\partial \hat{F}}{\partial w_{i,j}^{m}} \quad \textbf{(2.8)}$$

and the biases is

$$b_{i,j}^{m}(k+1) = b_{i,j}^{m}(k) - \alpha \frac{\partial \hat{F}}{\partial b_{i}^{m}} \quad \textbf{(2.9)}$$

Where α is the learning rate which determines the length of the step in the search direction to minimize $\hat{F}(\mathbf{x})$.

The error is an indirect function of the weights in the hidden layer, so the chain rule of calculus is used to compute the partial derivatives. The derivatives in Eq. 2.8 and Eq. 2.9 are written as:

$$\frac{\partial \hat{F}}{\partial w_{i,j}^{m}} = \frac{\partial \hat{F}}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}} \quad \textbf{(2.10)}$$

$$\frac{\partial \hat{F}}{\partial b_{i}^{m}} = \frac{\partial \hat{F}}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial b_{i}^{m}} \quad \textbf{(2.11)}$$

Since the net input to a layer m is a function of the weights and bias in that layer:

$$n_{i}^{m} = \sum_{j=1}^{s^{m-1}} w_{i,j}^{m} a_{j}^{m-1} + b_{i}^{m} \quad \textbf{(2.12)}$$

The second term in Eq. 2.10 and Eq. 2.11 will be

$$\frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}} = a_{j}^{m-1} \quad \textbf{(2.13)}$$

And

$$\frac{\partial n_{i}^{m}}{\partial b_{i}^{m}} = 1 \quad \textbf{(2.14)}$$

Specify s as the sensitivity of the performance index $\hat{F}$ to changes in the i th element of the net input at layer m:

$$s_i^m \equiv \frac{\partial \hat{F}}{\partial n_i^m} \tag{2.15}$$

Then Eq. 2.10 and Eq. 2.11 can be reduced to

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \tag{2.16}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m \tag{2.17}$$

This algorithm can now be expressed as

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \propto s_i^m a_j^{m-1} \tag{2.18}$$

For the weight and

$$b_i^m(k+1) = b_i^m(k) - \propto s_i^m \tag{2.19}$$

For the bias

The next step is to propagate the sensitivities backward through the network to compute the gradient:

Compute the sensitivity s^m which requires another application of the chain rule. This part of the routine has led to the term "*backpropagation"*. It describes a "repetitive" relationship in which the sensitivity s^m at layer m is calculated from the sensitivity s^{m+1} at layer $\mathrm{m}+1$. To derive this relationship for the sensitivities, the following Jacobian matrix is used:

$$\frac{\partial n^{m+1}}{\partial n^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{s^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & & \frac{\partial n_1^{m+1}}{\partial n_{s^m}^m} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_2^m} & & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_{s^m}^m} \end{bmatrix} \tag{2.20}$$

To find an expression for this matrix the following i,j elements of the matrix is considered:

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{l=1}^{s^m} w_{i,l}^{m+1} a_l^m + b_i^{m+1}\right)}{\partial n_j^m}$$

$$w_{i,j}^{m+1} \times \frac{\partial a_j^m}{\partial n_j^m} = w_{i,j}^{m+1} \times \frac{\partial f^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} \times f^m(n_j^m)$$

where

$$f^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m} \tag{2.21}$$

So the Jacobian matrix can be written as

$$\frac{\partial \boldsymbol{n}^{m+1}}{\partial \boldsymbol{n}^m} = \boldsymbol{W}^{m+1} \times \dot{F}^m(\boldsymbol{n}^m)) \tag{2.22}$$

where

$$\dot{F}^m(\boldsymbol{n}^m)) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \tag{2.23}$$

Using the chain rule in matrix form, the recurrence relationship for the sensitivity can be written as:

$$\boldsymbol{s}^m = \frac{\partial \hat{F}}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}} = \dot{\mathrm{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial \hat{F}}{\partial \mathbf{n}^{m+1}}$$

$$= \dot{\mathrm{F}}^m(\boldsymbol{n}^m)(\boldsymbol{W}^{m+1})^T \boldsymbol{s}^{m+1} \tag{2.24}$$

The sensitivities are computed by starting at the last layer, and then propagating backwards through the network to the first layer.

$$\boldsymbol{s}^M \rightarrow \boldsymbol{s}^{M-1} \rightarrow \cdots \rightarrow \boldsymbol{s}^2 \rightarrow \boldsymbol{s}^1$$

The starting point, $\boldsymbol{s}^m$, for the recurrence relation of Eq. 2.23 is obtained at the final network layer:

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (\boldsymbol{t}-\boldsymbol{a})^T (\boldsymbol{t}-\boldsymbol{a})}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^M} (t_j - a_j)^2}{\partial n_i^M} = -2(t_i - a_i)\frac{\partial a_i}{\partial n_i^M} \quad \textbf{(2.25)}$$

So that from Eq. 2.21

$$s_i^M = -2(t_i - a_i)\dot{f}^M(n_j^M)$$

Expressed in matrix form as

$$\mathbf{s}^{\mathrm{M}} = -2\,\dot{\mathbf{F}}^{\mathrm{M}}(\mathbf{n}^{\mathrm{M}})(\mathbf{t} - \mathbf{a}) \quad \textbf{(2.26)}$$

$$\mathbf{S}^{\mathrm{m}} = \mathbf{F}^{\mathrm{m}}(\mathbf{n}^{\mathbf{m}})(\mathbf{W}^{\mathrm{m+1}})^{\mathrm{T}}\mathbf{s}^{\mathrm{m+1}}, \text{ for } \mathrm{m} = \mathrm{M} - 1, \ldots, 2, 1 \quad \textbf{(2.27)}$$

Step 3 - Weight update

Backpropagation learning updates the network weights and biases in the direction in which the performance function diminishes most rapidly – the negative of the gradient. One iteration of this algorithm can be written as

$$\mathbf{W}^{\mathrm{m}}(\mathrm{k+1}) = \mathbf{W}^{\mathrm{m}}(\mathrm{k}) - \boldsymbol{\Delta}\mathbf{W}^{\mathrm{m}}(\mathrm{k}), \quad \textbf{(2.28)}$$

Where the change in weight is

$$\boldsymbol{\Delta}\mathbf{W}^{\mathrm{m}}(\mathrm{k}) = -\,\alpha\,\mathbf{s}^{\mathrm{m}}\left(\mathbf{a}^{\mathrm{m-1}}\right)^{\mathrm{T}}. \quad \textbf{(2.29)}$$

and

$$\mathbf{b}^{\mathrm{m}}(\mathrm{k+1}) = \mathbf{b}^{\mathrm{m}}(\mathrm{k}) - \boldsymbol{\Delta}\mathbf{b}^{\mathrm{m}}(\mathrm{k}), \quad \textbf{(2.30)}$$

Where the change in bias is

$$\Delta \mathbf{b}^{m}(k) = -\alpha \mathbf{s}^{m}. \quad \textbf{(2.31)}$$

The iteration is continued until the difference between the network response and the target function reaches some desired level set by the computer programmer.

2.2.3.2 Drawbacks of backpropagation training

The error performance surface

The mean square error (MSE) performance surface for a single-layered linear network is a quadratic function with only one single stationary point and constant curvature. Therefore, this training algorithm is guaranteed to converge to a solution that minimizes the MSE if the learning rate, α, is not too large.

The performance surface curvature for a multilayer nonlinear network, not a quadratic function, can vary widely in different regions of the weight parameter space. It would then be difficult to choose an appropriate learning rate for the steepest descent algorithm. For a flat surface, (a sigmoid transfer function output for large inputs saturates) a large learning rate would be suitable whereas for a region where the curvature is high, a smaller learning rate would be required.

Another feature of this error surface is the existence of many local minimum points. The steepest descent algorithm would effectively stop at a local minimum or even if the surface gradient is close to zero.

Also, if a multilayer network has two local minimum points with the same value of the squared error, the origin (0, 0) of the parameter space tends to be a saddle point for the performance surface. This might happen if the initial values of the weights and biases are set to zero.

In conclusion, the initial parameters should then be chosen to be small random values [18, p.12-5].

Convergence to the minimum error

In some instances the entire training set is presented to the network (batch training) before the parameters are updated. Slow or fast convergence can occur due to sudden changes in the curvature of the surface over the path of the steepest descent's trajectory towards the minimum error.

For example, changing the learning rate to speed up the convergence when it is traversing a flat section could make the algorithm's trajectory unstable when it reaches a steeper portion of the performance surface. This divergence or oscillations across the error surface can be smoothed out using a low pass "momentum" filter [18, p12-8].

Improving the gradient descent algorithm

Momentum

If one could filter the steepest descent's trajectory by averaging the updates to the parameters, this might smooth out the oscillations and produce a stable trajectory. This is done by adding a low pass filter termed momentum.

Adding a momentum coefficient γ tends to speed up convergence when the trajectory is moving in a steady direction. The larger the value of γ, the more "momentum" the trajectory has.

When the momentum modification is added to the parameter updates for the steepest descent backpropagation, Eq.(2.29) and Eq.(2.31) change to:

$$\Delta \mathbf{W}^{m}(k) = \gamma \Delta \mathbf{W}^{m}(k-1) - (1-\gamma)\alpha\, \mathbf{s}^{m} \left(\mathbf{a}^{m-1}\right)^{T}, \qquad \textbf{(2.32)}$$

And

$$\Delta \mathbf{b}^{m}(k) = \gamma \Delta \mathbf{b}^{m}(k-1) - (1-\gamma)\alpha\, \mathbf{s}^{m}. \qquad \textbf{(2.33)}$$

Where the momentum coefficient must satisfy $0 \leq \gamma < 1$

2.2.3.3 Faster Training – a numerical optimization technique

The previous section presented two backpropagation training algorithms: gradient descent, and gradient descent with momentum. These two methods are often too slow for practical problems. For improved training, a standard nonlinear least square optimization technique, the Levenberg-Marquardt algorithm, can be incorporated into the backpropagation training algorithm (See Chapter 9 of Hagan et al, for a review of basic numerical optimization [18]). It can converge from ten to one hundred times faster than the algorithms discussed previously.

Levenberg-Marquardt modification to the backpropagation training algorithm

This high performance algorithm is one of the faster methods for training moderate-sized feedforward neural networks up to several hundred weights. It was designed for minimizing functions that are the sums of squares of other nonlinear functions. This is beneficial to neural network training where the performance index is the mean square error. It is a variation of Newton's method. Newton's method requires calculation of the second derivative so it is only used when it is feasible to calculate the Hessian matrix $\mathbf{H}$ [48, p.15].

The Levenberg-Marquardt algorithm approaches second-order training speed without having to compute the Hessian matrix, which contains the second derivatives of the performance index along the network weights and biases axis. When the performance function has the form of a sum of squares, the Hessian matrix can be approximated as

$$\mathbf{H} = \mathbf{J}^{\mathbf{T}}\mathbf{J} \qquad \textbf{(2.34)}$$

and the gradient can be computed as

$$\mathbf{g} = \mathbf{J}^{\mathbf{T}}\mathbf{e} \qquad \textbf{(2.35)}$$

Where

$\mathbf{J}$ is the Jacobian matrix that contains the first derivatives of the network errors with respect to the weights and biases and,

$\mathbf{e}$ is a vector of network errors [29, p.25].

The Jacobian can be calculated using the standard backpropagation algorithm (section 2.2.3.1.1) that is less complicated than calculating the Hessian matrix.

If the weight and bias update from 2.29 and 2.30 is equal to some value δ

$$\delta = \Delta\mathbf{W}^{m}(k) + \Delta\mathbf{b}^{m}(k), \quad \textbf{(2.36)}$$

Then the Levenberg-Marquardt algorithm approximates a function that can be solved by:

$$(\mathbf{J}^{\mathbf{T}}\mathbf{J} + \mu\mathbf{I})\delta = \mathbf{J}^{T}\mathbf{e} \quad \textbf{(2.37)}$$

Where

$\mathbf{J}$ is the Jacobian matrix for the network,

μ is the Levenberg damping factor Mu,

$\mathbf{I}$ is the identity matrix,

δ is the weight update vector and

$\mathbf{e}$ is the error vector containing the output errors for each input vector used on training the network.

The weight update δ tells us by how much the network weight and bias parameters should be adjusted to achieve a near zero performance goal.

The μ damping factor is adjusted at each iteration, and influences the optimization process. If μ is equal to zero, this is just Newton's method, using the approximate Hessian matrix. When μ is increased, e.g. by a factor of 10, it is a step closer to the gradient descent direction.

Newton's method is faster and more accurate near an error $\mathbf{e}$ minimum, so the intention is to change toward Newton's method as quickly as possible. As a result, μ is decreased after each successful step (closer to a zero performance

goal) and is increased only when a tentative step would increase the performance function. In this way, the performance function (sum of the squares) is always reduced after each iteration cycle of the algorithm. The algorithm is assumed to have converged when some performance goal, in this case, zero, is reached.

Typical Levenberg-Marquardt algorithm training parameters are shown in Table 2.1, with their default values:

Table 2.1: LM training parameters

Maximum number of epochs to train	100
Performance goal	0
Maximum validation failures	5
Minimum performance gradient	1e-10
Initial μ	0.001
μ decrease factor	0.1
μ increase or adjustment factor	10

The next method to be discussed is the Cascade-Correlation architecture and supervised learning algorithm for artificial neural networks. It was developed by Scott Fahlman at Carnegie Mellon in 1990.

Cascade-Correlation training is a supervised learning architecture that grows layers of hidden neurons of fixed nonlinear activation functions (e.g. sigmoid functions), in real training time, so that the network architecture can be determined efficiently.

Instead of only adapting the weights in a network of fixed topology, Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden neurons one by one, creating a multi-layer structure [52, p.9].

Once a new hidden neuron/unit has been added to the network, its input-side weights are frozen. This unit then becomes a permanent feature-detector in the network, available for producing outputs or for creating new, more complex feature detectors.

2.2.3.4 A Variation of backpropagation training - The Cascade correlation training algorithm [11]

Cascade correlation neural networks are “self-ordering” networks, finding their own size and topology. Through the training process, new neurons are added in the hidden layer, one at a time, producing a multi-leveled network formation. Initially, a cascade correlation neural network consists of only the input and output layer neurons with no hidden layer neurons. Every input is connected to every output neuron by a connection with an adjustable weight, as shown in Figure 2.12.

Each node, depicted by the symbol ⊗ in Figure 2.12, represents a weight value between the input and the output neuron. Values on a vertical line are added together after being multiplied by their weights. So each output neuron receives a weighted sum from all of the input neurons including the bias. The output neuron sends this weighted input sum through its transfer function to produce the final output.

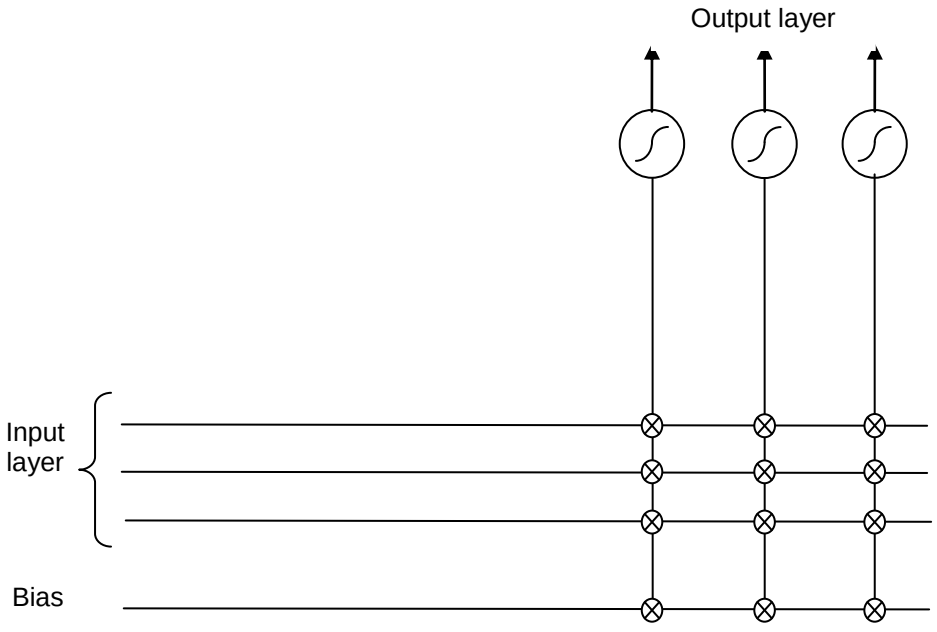


Figure 2.12: Initial network state: No neurons in the hidden layer

After the addition of the first hidden neuron, the network would have the structure shown in Figure 2.13:

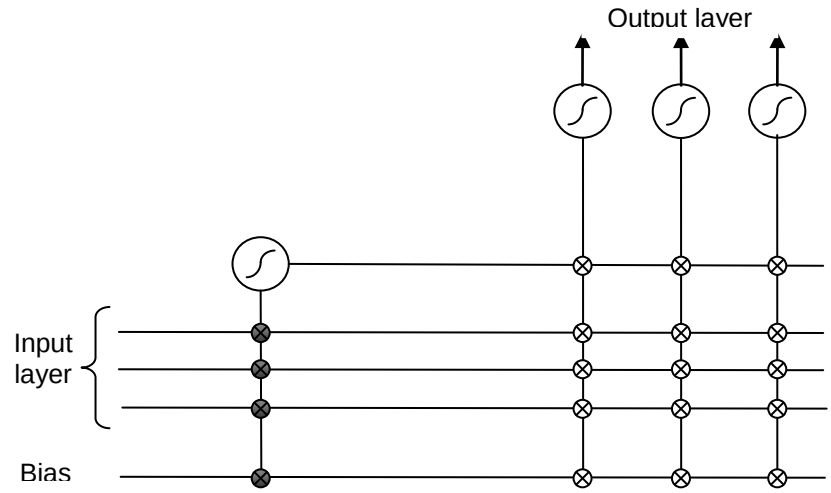


Figure 2.13: Adding the first neuron to the network's hidden layer

The input weights for the new added hidden neuron in Figure 2.13 are shown as ⊗ - nodes to indicate that they are fixed once the neuron has been added. Weights for the output neurons, shown as ' ⊗ - nodes', continue to be adjustable.

Figure 2.14 is a schematic presentation of a network with two hidden neurons.

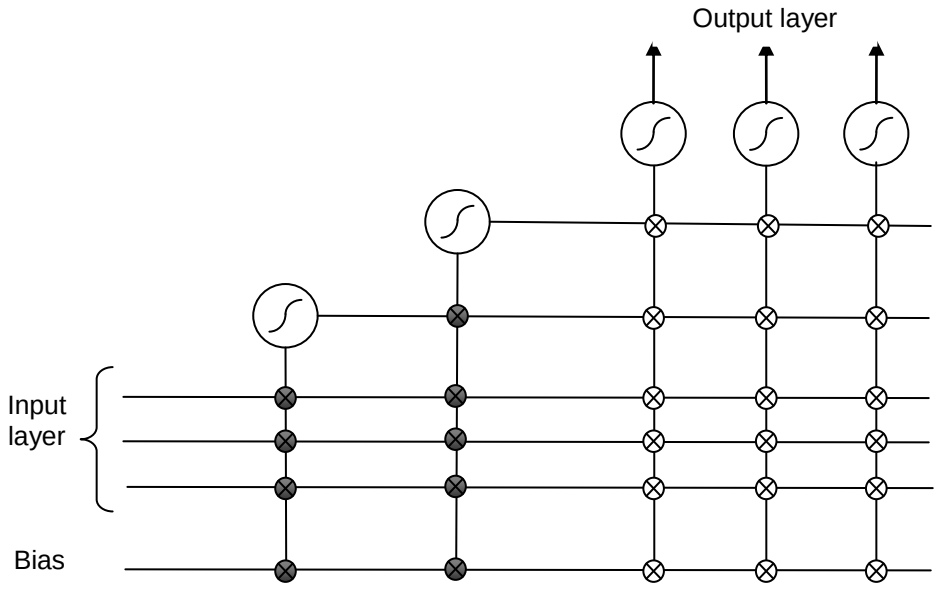


Figure 2.14: Adding a second neuron to the network's hidden layer

Note that the second neuron receives inputs from the external inputs and pre-existing hidden neurons.

According to Fahlman and Lebiere [11], the Cascade-Correlation training algorithm has several advantages over the backpropagation training algorithm:

- It is a self-organizing network and grows its own hidden layer(s) during training. The researcher does not have to be concerned with the design of the overall network architecture.
- Very fast training times – often more than 1000 times as fast as a feedforward backpropagation network for this specific application.
- Typically, cascade correlation networks are fairly compact, often having fewer than a dozen neurons in the hidden layer, retaining its structure it has built even if the training set changes.
- Cascade correlation network training is quite robust, and good results can usually be obtained with little or no adjustment of parameters.

Even a simple Cascade-Correlation network with no hidden neurons has considerable predictive power. For a fair number of problems, a cascade correlation network with just an input and an output layer provides excellent predictions.

During training, the neural network learns the model of the application in which it will be operating and must maintain it adequately and reliably to achieve the particular goals of the application of interest, e.g. load forecasting. As briefly mentioned in 2.2.3, a problem that can occur during network training is called *underfitting* or *overfitting*.

2.2.3.5 Underfitting and overfitting of artificial neural network output data

Designing a network that does not have enough hidden layers or too small a number of neurons in the hidden layer(s) and exposing it to a complex signal, might make it fail to fully detect the signal, leading to *under-fitting*. It is rather like

sampling a complex, high frequency signal at too low a rate, as found in Digital Signal Processing.

Another example is if the data set is very noisy due to a "measurement error". Then a very complex network will fit the underlying noise as well as the intended signal, leading to *over-fitting*. Under-fitting and over-fitting can produce wild predictions in networks consisting of multilayered perceptrons. As a result, the network training needs to be done in such a way that the network weights and biases are adjusted to improve generalisation.

Training neural networks to generalise

A neural network can generalise when it correctly classifies input values that are not in its training data sets. A neural network can generalise well when the accuracy of classification is high and vice versa.

A complex, nonlinear function is programmed by the neural network from its inputs. If the training data is fitted well by the trained neural network and it classifies its data rather accurately, then there is probably some well-established mathematical relationship between the input/output vectors. Estimating outputs from new inputs would probably be more accurate now and it will be said that the neural net is generalising quite well. One method used to improve generalisation is regularisation.

Regularisation

This requires changing the performance function, which is normally chosen to be the sum of squares of the network errors on the training set. The typical performance function used for training feedforward neural networks is the Mean sum of Squares of the network Errors (MSE):

$$MSE=\frac{1}{n}\sum_{t=1}^{n}\left(A_t-F_t\right)^2 \qquad \textbf{(2.38)}$$

where

A_t = actual value;

F_t = forecast value;

n = number of fitted data points.

To improve generalization one can modify the performance function by adding a term that consists of the Mean of the sum of Squares of the network Weights and biases, (MSW**)**.

The regularised MSE = γ MSE + (1 – γ) MSW **(2.39)**

Where

γ = performance ratio

For example, if the network is trained with the regularized performance function where the performance ratio γ is set to 0.5, it gives equal weight to the mean square errors and the mean square weights.

and

$$MSW=\frac{1}{n}\sum_{j=1}^{n}(w_j)^2 \quad \textbf{(2.40)}$$

Where

j = layer number

w_j = weight and bias matrix in that layer

Using the regularised performance function causes the network to have smaller values of weights and biases. This forces the network response to be smoother and less likely to overfit.

The problem with regularization is that it is difficult to determine the optimum value for the performance ratio parameter γ. If one makes this parameter too large, one might get overfitting. If the ratio is too small, the network does not sufficiently fit the training data.

A routine that automatically sets the optimal performance function to achieve the best generalization is the Bayesian framework of David MacKay [28, pp.415-447] where the weights and biases of the network are assumed to be random variables with specified distributions. The regularization parameters are related to

the unknown variances associated with these distributions. One can then estimate these parameters using statistical techniques.

One feature of this algorithm is that it provides a measure of how many network parameters (weights and biases) are being used effectively by the network. This effective number of parameters should remain approximately the same, no matter how large the number of parameters in the network becomes. (This assumes that the network has been trained for a sufficient number of iterations to ensure convergence.)

Training can be stopped if the algorithm has converged when the Sum of the Squared Error (SSE) and Sum of the Squared Weights (SSW) are relatively constant over several iterations [8, Chapter 5 - p.55].

The response of the trained network is such that it will never overfit the data, and, therefore, the network will generalize well to new inputs. This eliminates the guesswork required in determining the optimum network size.

A detailed discussion of Bayesian regularization is beyond the scope of this work. A discussion in the use of Bayesian regularization, in combination with Levenberg-Marquardt training, can be found in [14, pp. 1930-1935].

Once the chosen neural network is trained, it needs to be evaluated as a suitable forecasting model. One can build different neural network models and use the same data set on all of them to compare their forecasting performance. The final network model and training method is selected when the best model fit is obtained from the training and new validation data when measuring the neural networks performance.

2.2.4 Measuring the neural network model performance

The user of forecasts is concerned about the accuracy of future forecasts, not always about model goodness of fit [32, p.49]. There are many statistical methods that describe how well a model fits a given sample of data and a few of

these will be mentioned in 2.2.4.1. The quality of the forecasting model can also be investigated by the use of linear regression plots, briefly described in 2.2.4.2. This can provide useful guidance on how the forecasting model will perform when exposed to new data.

2.2.4.1 Choosing the Mean Absolute Percentage Error (MAPE)

The most important measure of a trained neural network's performance is its forecasting accuracy using data other than the training data. An acceptable method of measuring the accuracy of any ANN forecaster is argued over by forecasting academics and practitioners. The ANN's performance is often defined in terms of the forecasting error which is the difference between the actual value A_t and the forecasted value F_t.

$$\text{Forecasting error} = A_t - F_t \qquad \textbf{(2.41)}$$

The most frequently used "measures of accuracy" in the forecasting literature [57, p.51] are the:

MAD - **M**ean **A**bsolute **D**eviation:

$$MAD=\frac{1}{n}\sum_{t=1}^{n}\left|A_t-F_t\right| \qquad \textbf{(2.42)}$$

SSE - Sum of the **S**quared **E**rrors:

$$SSE=\sum_{t=1}^{n}\left(A_t-F_t\right)^2 \qquad \textbf{(2.43)}$$

MSE - Mean **S**quared **E**rror:

$$MSE=\frac{1}{n}\sum_{t=1}^{n}\left(A_t-F_t\right)^2 \qquad \textbf{(2.44)}$$

RMSE - **R**oot **M**ean **S**quared **E**rror:

$$RMSE=\sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(A_t-F_t\right)^2} \qquad \textbf{(2.45)}$$

MAPE - **M**ean **A**bsolute **P**ercentage **E**rror:

$$MAPE=\frac{1}{n}\sum_{t=1}^{n}\frac{|A_t-F_t|}{A_t}\times 100\ \% \qquad \textbf{(2.46)}$$

Where

A_t = actual value and $A_t \neq 0$;
F_t = forecast value;
n = number of fitted data points.

Equations 2.42 to 2.45 are scale–dependant measures of forecasting accuracy. So if one were forecasting electric load demand for Bloemfontein during the winter, the units would be expressed in Megawatts (MW). If the MAD, from Eq. 2.42, for a month in the winter was 6 MW, one might not know whether it was a large or a small forecasting error. Furthermore, accuracy measures that are scale dependent do not facilitate comparison of a single forecasting technique across different time series or comparisons across different time periods.

To achieve this, one needs a measure of relative forecast error such as Eq.2.46. Knowing that the percentage electric load forecasting error or the MAPE is 4 % can be much more meaningful than knowing that the MAD is 6 MW [32, pp.48-51].

When measuring the forecasting performance of the different ANN's, the MSE is not used as it has the disadvantage of heavily weighting outliers when squaring each term. This weighs large errors more heavily than small ones and makes its use undesirable in many applications.

Therefore, researchers rather use alternatives such as the MAPE, which can be a valuable approach for discriminating between competing forecasting models [44, p.6], [30, p.370].

The MAPE can be used to decide on practical network architectures for future analysis of the possibility of using it to build a short term load forecasting model.

In addition, besides the MAPE criterion other statistical measures are available to further evaluate the network's performance, using linear regression or scatter plots.

2.2.4.2 Linear regression plots

The success of a trained network can be considered to some degree by evaluating the MAPE on the training, validation and test sets, but it is often useful to look into the network response in more detail. One option is to perform a regression analysis [4, pp.298-318], [31, p.64] between the network output response F and the actual corresponding target A in the form

$$\text{Output} = m.\text{Target} + \text{y}. \qquad \textbf{(2.47)}$$

The MATLAB routine '*plotregression',* as described below*,* perform this analysis [8, p.5-64]. It returns three parameters. The first is m*,* representing an estimate of the slope of the linear regression line for which the formula is

$$m = \frac{\sum_{t=1}^{n}(A_t-\bar{A})(F_t-\bar{F})}{\sum_{t=1}^{n}(A_t-\bar{A})^2} \qquad \textbf{(2.48)}$$

Where

A_t = actual value;

$\bar{A}$ = actual data average;

F_t = forecast value;

$\bar{F}$ = forecasted data average;

t = data point number;

n = number of fitted data points.

and y represents an estimate of the *y*-intercept of the best linear regression, relating targets to network outputs.

The intercept y, can be calculated from

$$\mathrm{y} = \bar{\mathrm{F}} - m\bar{\mathrm{A}} \qquad \textbf{(2.49)}$$

Where

$\bar{\mathrm{F}}$ = forecasted data average;

$\bar{\mathrm{A}}$ = actual data average;

m = estimate of the slope.

If there were a perfect fit (outputs exactly equal to targets), the slope would be 1, and the *y*-intercept would be 0.

The third variable returned by MATLAB's *plotregression* is Pearson's correlation coefficient (the **R** value) between the outputs and targets. The **R** value is an indication of the relationship between the outputs and targets. It is a measure of how well the variation in the output is explained by the targets. The formula for **R** is

$$\mathbf{R} = \frac{\sum_{t=1}^{n}\left(\mathrm{A_t}\text{-}\bar{\mathrm{A}}\right)\left(\mathrm{F_t}\text{-}\bar{\mathrm{F}}\right)}{\sqrt{\sum_{t=1}^{n}\left(\mathrm{A_t}\text{-}\bar{\mathrm{A}}\right)^2}\sqrt{\sum_{t=1}^{n}\left(\mathrm{F_t}\text{-}\bar{\mathrm{F}}\right)^2}} \qquad \textbf{(2.50)}$$

Where

$\mathrm{A_t}$ = actual value;

$\bar{\mathrm{A}}$ = actual data average;

$\mathrm{F_t}$ = forecast value;

$\bar{F}$ = forecasted data average;

t = data point number;

n = number of fitted data points.

If **R**=1, this indicates that there is an exact linear relationship between outputs and the actual values (targets). If **R** is close to zero, then there is no linear relationship between outputs and targets.

In Figure 2.15, it is difficult to distinguish the best linear fit (blue line) from the perfect fit (dashed line) because the fit is 1, which illustrates a perfect fit. The network outputs are plotted versus the targets as open circles.

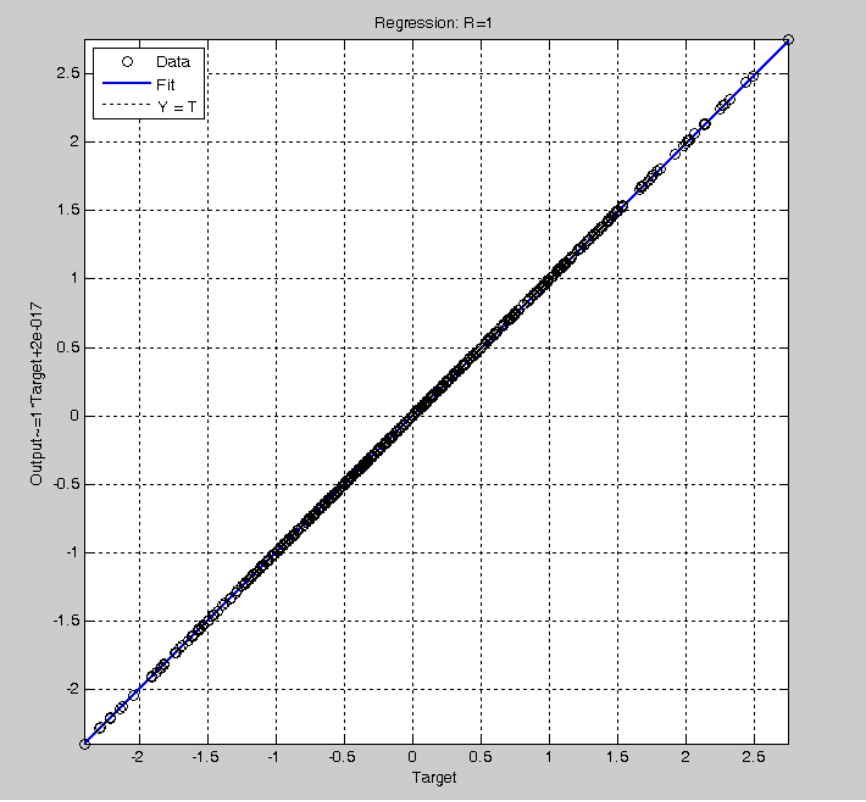


Figure 2.15: An example of a perfect linear regression plot where the input values = output values so that the correlation coefficient R=1

A more practical example is illustrated by Figure 2.16. The perfect fit (output equal to targets) is indicated by the dashed line representing the perfect result where the network outputs = actual targets. The solid blue line represents the best fit linear regression line between outputs and targets.

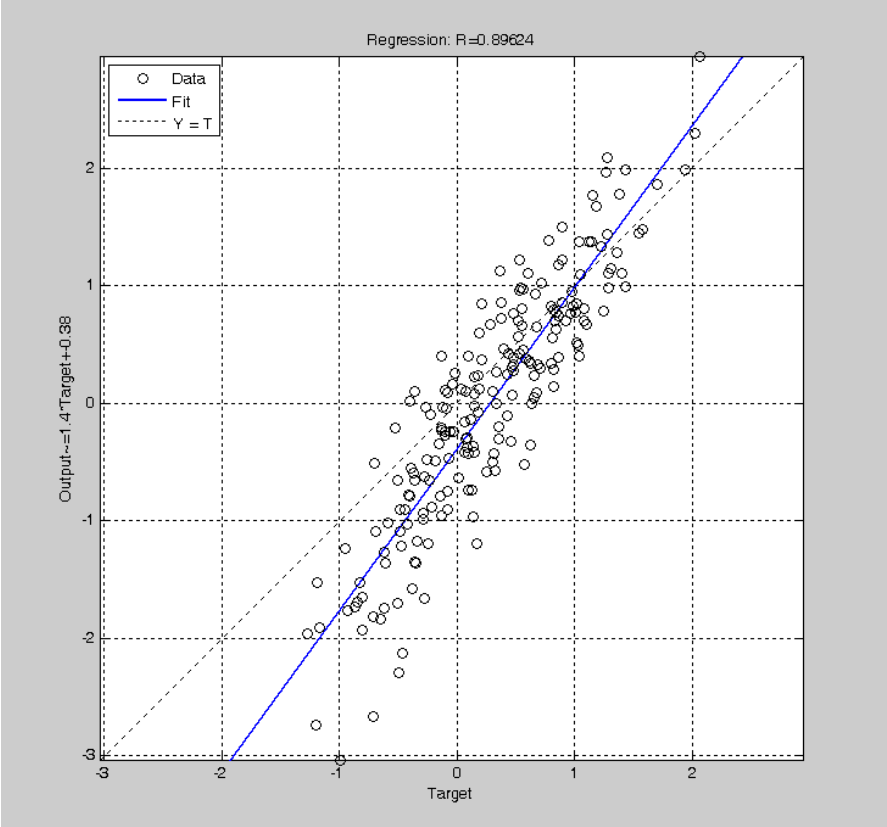


Figure 2.16: A more realistic example of a linear regression plot where the input values ≠ output values and the correlation coefficient R=0.89624

2.2.5 Summary

Load curve prediction can be modelled using a neural network time series forecasting model. Further to this, supervised training can be implemented to train a neural network. Moreover, the Levenberg-Marquardt modification can be applied to the backpropagation algorithm to improve the training performance.

From the exposition above it can be concluded that a self organising Cascade-Correlation network is faster and more efficient than the feedforward network.

Also, generalisation can be improved by using the Bayesian framework. To conclude, the neural network's performance can be measured by making use of the MAPE and linear regression statistics.

CHAPTER 3

METHODOLOGY

For this project, electric load data consisting of active and reactive power, was obtained from the two 132 kV feeder voltage sources at Harvard substation, west of Bloemfontein. The active power component was extracted from the load data. This was used to develop an artificial neural model to predict power demand with a lead time of half an hour.

Due to the economic growth of the Bloemfontein Municipal area, older power demand data was of decreasing relevance as it does not reflect the changes in the composition of the recent load demand accurately. Later data sets from 2006, 2007, 2008 and 2009, where the change in the Notified Maximum Demand (NMD) was less than ten per cent, was used to develop the neural network forecasting model. This type of model can be used to capture complex relationships between inputs and outputs or to find patterns in the electric load data.

An important aspect of load forecasting is the relationship it has with load planning. Forecasting can be described as predicting what the future *will* look like, whereas planning predicts what the future *should* look like. There is no single correct forecasting method to use. Selection of a procedure should be based on the investigators' objectives and conditions (such as data, model and method, etc.). A good place to start is to identify the underlying factors that might influence the variable that is being forecasted. The procedure used to execute the investigation is then set out.

3.1 Procedure used to develop the neural network forecasting model

A procedure is a series of linked actions that convert one or more inputs into one or more outputs [32, p.12]. All work activities are conducted in specific procedures. Forecasting is no exception. The actions in this forecasting procedure are executed in the following sequence:

- Specification of the aim and objectives.
- Data collection.
- Data analysis.
- Neural network model selection, training and testing.

3.2 Specification of the aim and objectives

Aim:

To build, train and test a short term dynamic, half-hour ahead updating, neural network forecaster using the data for each of the late autumn and winter months of May, June and July 2007 to 2009. Once the network's performance has been verified, it could be extended to an annual forecasting system for future years.

Objectives:

- Each of the three networks for the three months must be trained with the least possible amount of historical data, using the minimum quantity of inputs to predict the weekly load, with a half-hour lead time.

- Minimising the error during training by evaluating and adjusting the topology elements of the neural network to keep the weekly MAPE below 5% during forecasting. This was decided on to represent an accepted forecasting accuracy.

- The neural networks functional reliability would be estimated experimentally [15, p.305]. Final evaluation of the STLF network's performance would be done by comparing weekly MAPE and **R** values (correlation coefficients), taken from the regression plots, to analyse the forecasting accuracy for each month. This is done by checking if the prediction error from week 1 to week 4 for each month is repeated consistently. Accuracy is limited by systematic (repeatable) errors [39, p.347].

- The forecasting time horizon decided upon will be one month which means that a network can be built and trained for each month of the year if on-line forecasting deployment is eventually realised.

3.3 Data collection

Historical load demand data for the years 2006 to 2009 from Parkwest Feeder 1 and Feeder 2 at the Harvard 132 kV substation was obtained from Eskom, Bloemfontein in the format shown in Table 3.1. Initially, the reliability and integrity of the data for the months of May, June and July was examined graphically, comparing each week of the same month of each year with the weekly data obtained from each other year e.g. the data of week 1, May 2006 was checked with the data for week 1, May 2007, week 1, May 2008 and week 1, May 2009, etc…

Table 3.1: An example of load data obtained from ESKOM

HARVARD 132 kV SUBSTATION

LOAD DATA FROM PARKWEST FEEDER 1 & FEEDER 2

Recorder	Date	Time	kW	kvar	kW	kvar	TOTAL Kw	TOTAL Kvar	Recorder
"BF-00000000673	7/1/2006	30	78720	10560	72960	9600	151680	20160	"BF-00000000675
"BF-00000000673	7/1/2006	100	74880	9600	69120	8640	144000	18240	"BF-00000000675
"BF-00000000673	7/1/2006	130	72960	9600	67200	7680	140160	17280	"BF-00000000675
"BF-00000000673	7/1/2006	200	71040	8640	66240	8640	137280	17280	"BF-00000000675
"BF-00000000673	7/1/2006	230	70080	8640	64320	8640	134400	17280	"BF-00000000675
"BF-00000000673	7/1/2006	300	69120	8640	64320	7680	133440	16320	"BF-00000000675
"BF-00000000673	7/1/2006	330	70080	9600	64320	9600	134400	19200	"BF-00000000675
"BF-00000000673	7/1/2006	400	69120	8640	63360	8640	132480	17280	"BF-00000000675
"BF-00000000673	7/1/2006	430	70080	8640	64320	8640	134400	17280	"BF-00000000675
"BF-00000000673	7/1/2006	500	71040	7680	66240	7680	137280	15360	"BF-00000000675
"BF-00000000673	7/1/2006	530	73920	8640	69120	8640	143040	17280	"BF-00000000675
"BF-00000000673	7/1/2006	600	76800	8640	71040	7680	147840	16320	"BF-00000000675
"BF-00000000673	7/1/2006	630	82560	9600	76800	8640	159360	18240	"BF-00000000675
"BF-00000000673	7/1/2006	700	89280	11520	82560	10560	171840	22080	"BF-00000000675
"BF-00000000673	7/1/2006	730	96960	12480	89280	10560	186240	23040	"BF-00000000675
"BF-00000000673	7/1/2006	800	104640	13440	96960	12480	201600	25920	"BF-00000000675
"BF-00000000673	7/1/2006	830	114240	17280	105600	14400	219840	31680	"BF-00000000675
"BF-00000000673	7/1/2006	900	119040	19200	111360	16320	230400	35520	"BF-00000000675

3.4 Data analysis

3.4.1 Load curve characteristics

The load curve information for Bloemfontein, collected for the years 2006 to 2009, consisted of discrete data points, accumulated every half-hour, taking into account that 24 hours equals 48 half-hours. As a result, the *daily* load curves comprised 48 observations, which is the sampling rate of the data. As a result, a total of 17520 data points per year are available for investigation.

As an example, the electric power expenditure in the year 2009 is shown in Fig 3.1.

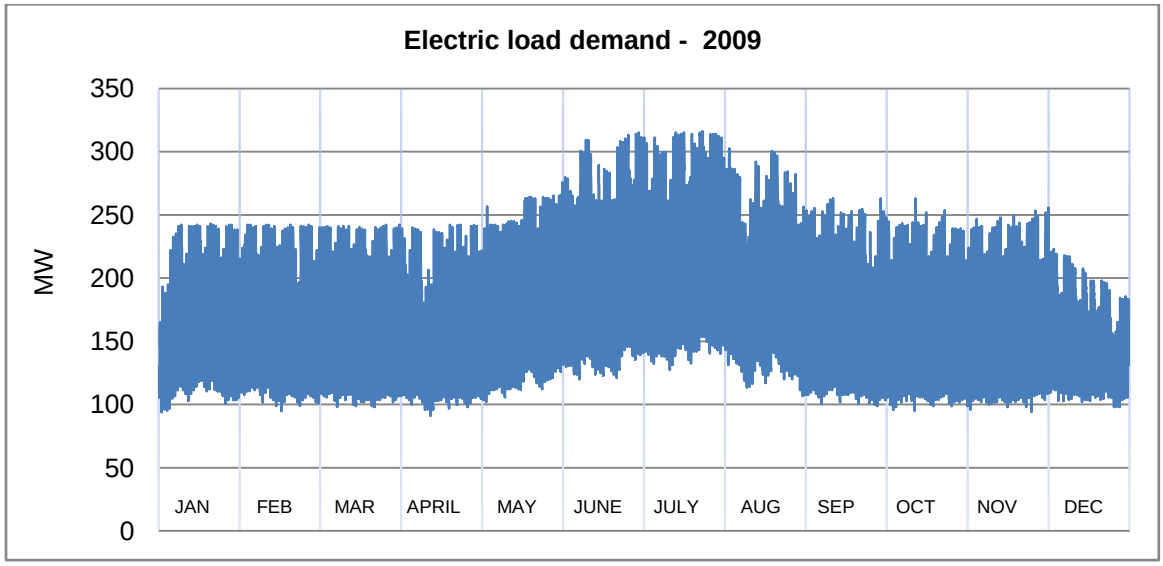


Figure 3.1: Annual load profile used from January 1 - 2009 to December 31 - 2009

The load starts to pick up middle January as industry starts up and schools start after the Christmas holidays. A small drop is seen during April, depending on the span of the Easter holiday window. Energy demand reaches a maximum during winter. This can either be May, June or July depending on the type of winter season experienced during the specific year under observation. The load drops off as the season changes to Spring and stays constant until the December holidays during which a rapid drop to below 200 MW is observed (for the duration of the holiday).

The following seasonal variation can be observed from this load profile:

- A maximum consumption of less than 250 MW during the summer and autumn season;
- A maximum consumption of more than 250 MW during the winter and early spring season;
- The base load level barely drops below 100 MW throughout the year.

Time series plots of the electric load data were created and visually inspected for familiar patterns e.g. trends, seasonal components and abrupt changes in magnitudes of peaks and valleys. Abnormal data points or outliers were checked for and removed if necessary. The purpose of this initial data examination was to obtain a ‘feel’ for the data. In the region of the Free State, South Africa, the months in the seasons follow the pattern shown in Figure 3.2.

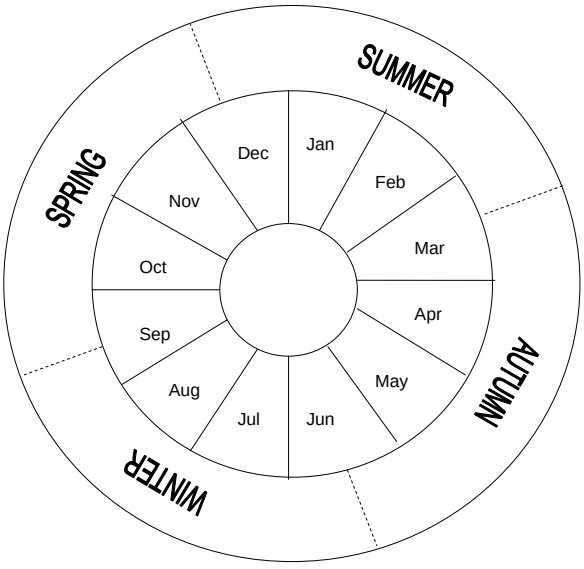


Figure 3.2: Segmentation of the months in the different seasons of the year

Usually, during the winter season, electric load consumption reaches an annual peak. Also, in the months of May, June and July, extensive load shedding is applied to save the considerable penalties induced when exceeding the NMD.

To predict the electric load during the winter period, a neural network was built, trained and validated for each of the three months of May, June and July, mentioned previously as the late autumn and winter months.

Through the course of a month in a specific season, the weekdays and weekend load shape usually repeat in a fixed pattern. The weekly characteristics consist of five working days and two weekend days.

The next objective to consider would be the load curve shape of the different day types of the week of each month to present to the neural network for training. From Figure 3.3, days can be separated into different day types, each having its own distinctive features. The daily rhythm changes throughout the year. There are normal and special days, weekdays and weekends.

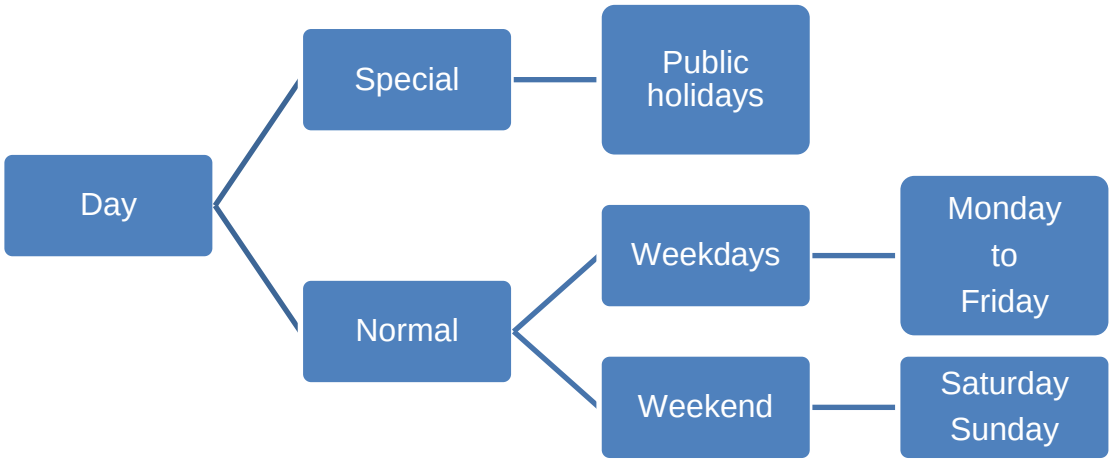


Figure 3.3: Layout of possible day configurations for the neural net to learn

In Figure 3.4 the load over two consecutive weeks with three weekends included in July 2009 is shown.

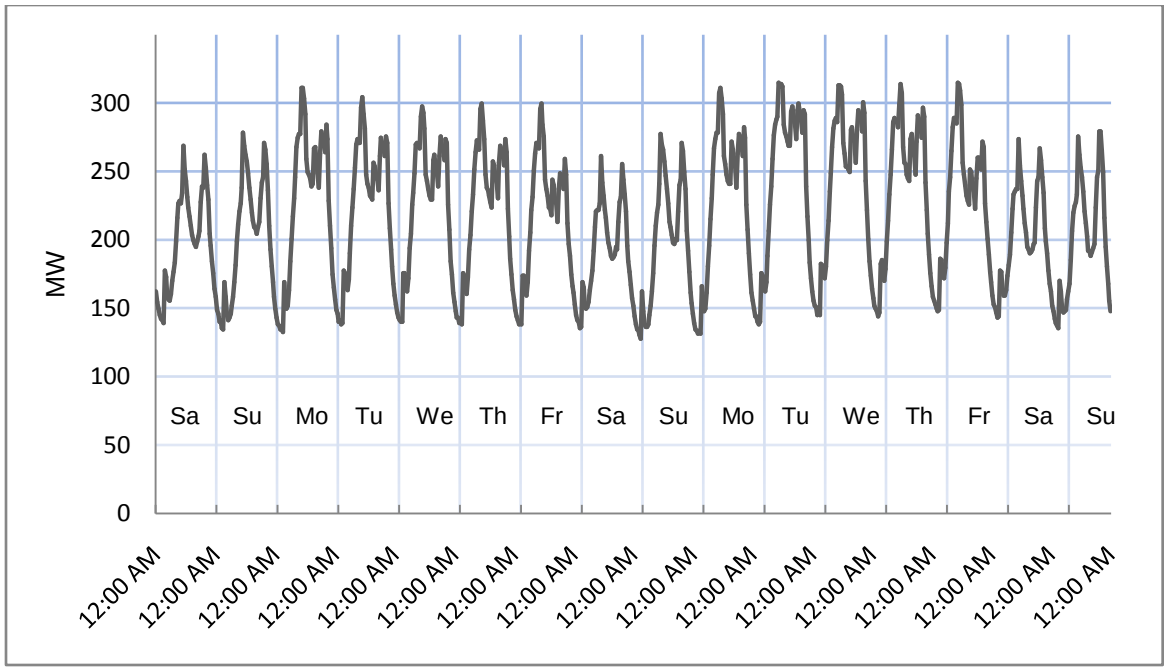


Figure 3.4: Load profile from Midnight Saturday, 4th July 2009, to Midnight, Sunday 19th July 2009.

The 12:00 AM's below the x-axis in the graph actually represents midnight at 0:00 Hours. The load profile in Figure 3.4 shows the following features:

- It is observed that the Saturday- and Sunday load patterns differ in shape from the rest of the week with lower peak load values. Households are usually more inactive over the weekend.

- The Monday to Friday peak power use is higher compared to peak loads during Saturdays and Sundays due to a higher communal activity. An increase in "ramping of load" is required for Monday mornings as the community starts up after the weekend. (It is necessary that the forecast be as accurate as possible during these ramping periods to avoid any unnecessary interruptions to customers because of load shedding).

- The Tuesday to Thursday profiles are very similar under “normal” conditions, making it the easier days to forecast.

- The load starts dropping off from a Friday evening peak, which is lower than the Friday morning peak, as the weekend starts.

- The base load stays at approximately 140 MW over the two weeks.

- The morning peaks are between 11:00 and 12:00 and the evening peaks are between 6:30 and 7:30.

- A slight peak at 3:30 am each morning is due to geyser “load control”.

As seen from Figure 3.4, there are differences between the seven days in the same week of the month and even between the “day types” of two consecutive weeks. So each day type has its own characteristic load pattern [34, pp.10-14].

From plotting these graphs one can explore the historical daily energy usage data to develop a forecasting model. Statistics and visual inspections may reveal that there are usage trends throughout each day, and these trends seem to depend on the day type of the week.

This knowledge can be used to build and test a neural network to forecast the weekly load curves. Another possibility would be to use a separate network to forecast each day type.

Using preliminary testing, the network forecasting results can be used to analyse these historical forecasting errors for daily, weekly, or monthly behaviour.

The neural network’s forecasting performance can be defined by the previous week for the next week, last year’s monthly data for next year or include all the weekly day types.

In Figure 3.5 a typical day type profile is illustrated. Load consumption curves of a typical Wednesday in different seasons of the year 2009 are shown.

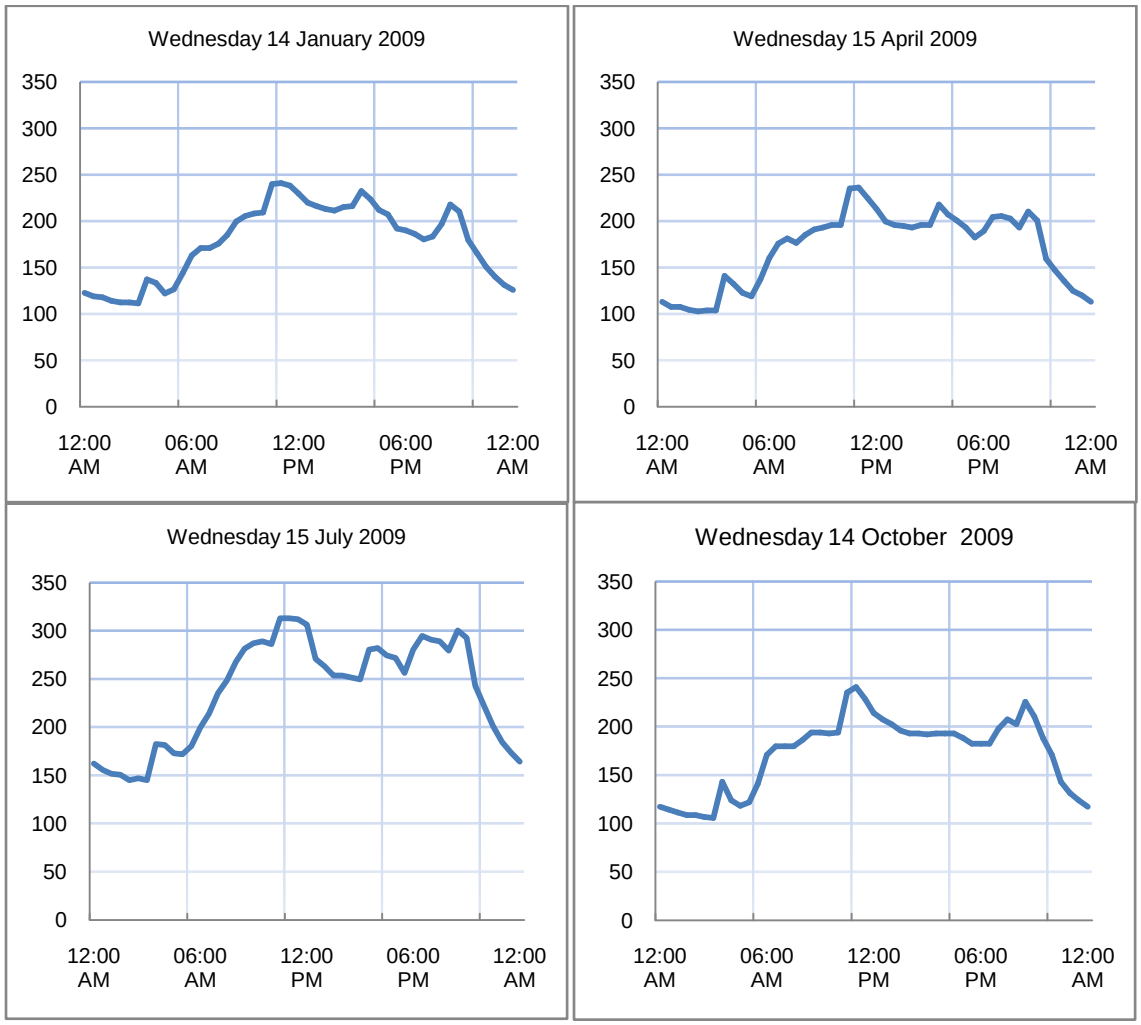


Figure 3.5: Load curves of four Wednesdays in the four different seasons

The special days vary at random and one must also look at its load curve shapes to see if one has to train the network especially for this occurrence.

Investigating the profile of the electric load data plotted in Figure 3.6 to Figure 3.8 for the year of 2009 it is seen that on the special days of the year (see Fig. 3.3) the load consumption curves follow roughly the same pattern as on a Sunday in that same week,

so it was not deemed necessary to additionally train a network particularly for the special days.

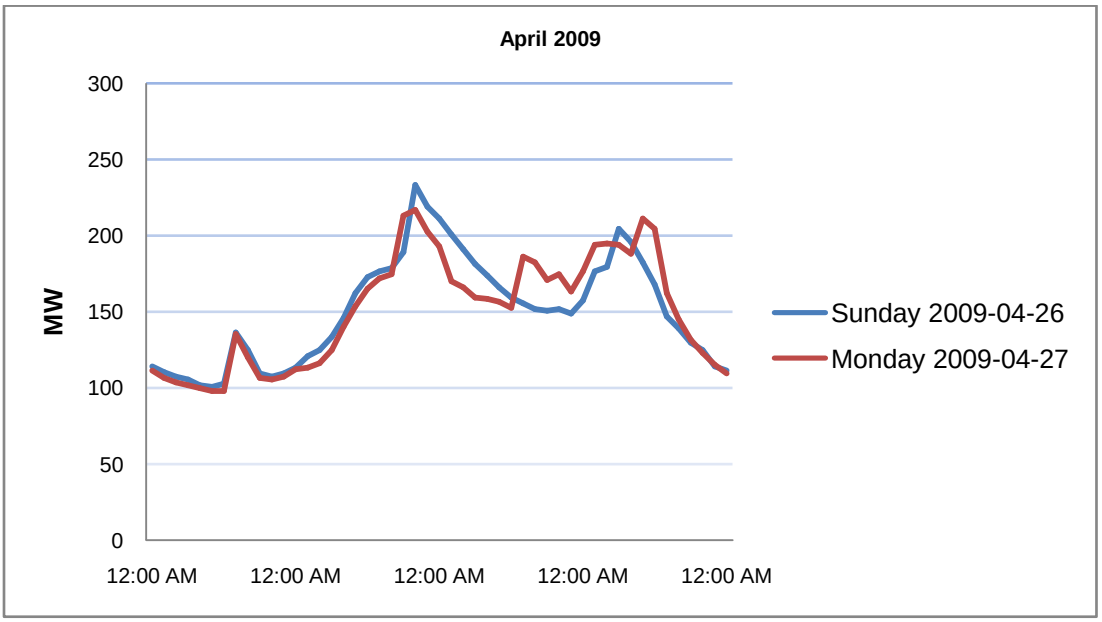


Figure 3.6: Freedom Day on a Monday

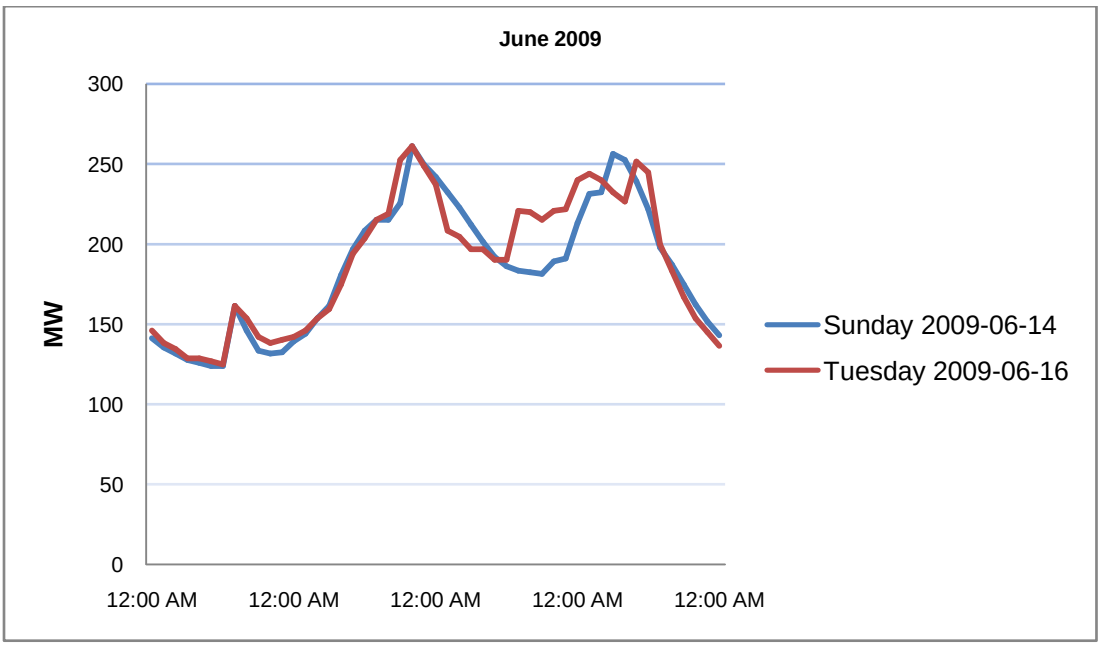


Figure 3.7: Youth Day on a Tuesday

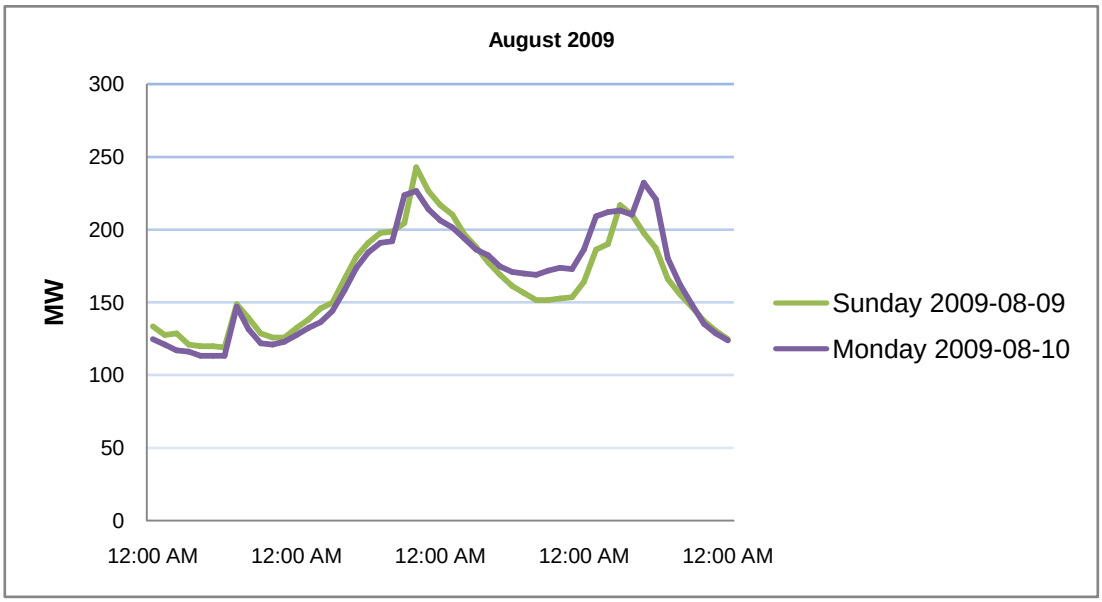


Figure 3.8: National Women's Day on a Monday

3.4.2 Data pre-processing

Data pre-processing was done in three phases:

1. Selected data was copied to files in EXCEL format to make importing and exporting the data to MATLAB more efficient.

2. The data was checked for outliers and missing data. Minimal adjustments were identified and discarded or rectified.

3. Data entering the MATLAB environment was normalised to the range [- 1; +1] for the ANN to facilitate training and prevent "squashing" by the sigmoid activation function [35, p.465].

3.5 Neural network model selection, training and testing

All the computer simulations and development were done using MATLAB's Neural Network Toolbox™ for building, training, validating and testing neural network applications.

3.5.1 Method of selection of a network architecture or topology for further analysis

The network topology describes the arrangement of the neural network. Choosing the proper neural network topology is regarded as a key aspect in optimisation and reliability of neural network performance.

Selecting the topology of the neural network is a difficult decision. There are no specific directions for developing a neural network for a particular task. ‘Guidelines are either heuristic or based on simulations derived from limited experiments. Hence the design of an ANN is more of an art than a science’ [57, p.42].

The network topologies available are numerous; each with its inherent advantages and disadvantages. Some networks sacrifice speed for accuracy, while some are capable of handling static variables and not continuous ones.

For this project, the data sets were not large and did not consist of multiple arrays of input variables such as temperature, active power, reactive power and day type which would necessitate the use of complex topologies. Consequently, in order to arrive at an appropriate robust network topology, the multilayer feedforward network was investigated first. Next, the performance of the cascade-forward network topology was analysed.

To develop an understanding of and gain in experience of how this load forecasting model would develop, an initial, experimental arrangement of the multilayer feedforward network architecture was set up to train and test the different load data configurations.

This network’s ability to model a set of input examples was measured using the MAPE performance function as discussed in section 2.2.

Test runs, presented in this section, would serve as a baseline against which to choose between the two architectures. It includes input and hidden layer size testing for the

feedforward network and input layer size testing for the cascade forward network. Once the input set size and the hidden layer size had been determined, training tests were done to decide between the Levenberg-Marquardt with and without the Bayesian regularization training algorithm to fine-tune the final network's performance.

3.5.2 Selecting the multilayer feedforward network configuration.

The feedforward back propagation network structure is problem dependant. Therefore it is assumed that a network layout that is used for this load forecasting system will not automatically be suitable for another forecasting system.

The architecture of the feedforward ANN was experimentally estimated using the Levenberg-Marquardt training method in conjunction with Bayesian regularization for improving generalization. The following criteria were considered:

1. The size of the output layer.
2. Interconnection of neural nodes.
3. Type of activation functions used in each layer.
4. The number of hidden layers.
5. Determine the size of the input layer.
6. Determine the size of the hidden layer.

These criteria will be discussed in the following section. The question to be asked is which parameter is determined first and why? As mentioned, this is not a simple task. The first four criteria were first kept constant, to practical values, as mentioned in the next discussion, while criteria 5 and 6 were being adjusted simultaneously to get a "feel" for the rhythm of the problem.

Criteria 5 and 6 were evaluated using the MAPE performance measurement. Four weeks of input data in July 2008 were prepared and presented to train this preliminary

network and the 4th week in July 2009 was used to validate the forecasting accuracy of this trained feedforward network.

A decision, based on the results of Table 3.2, was then used to consider the next step in finding a suitable network for the investigation of a robust short term load forecasting model.

3.5.2.1 Discussing the criteria used to estimate the initial feedforward network architecture experimentally

Size of the network output layer

The output layer is one only as dictated by the nature of this problem to be solved.

Interconnection of the network nodes

The nodes are fully connected.

Activation functions used in the hidden and output layer

The log sigmoid transfer function was initially used for test runs in the hidden layer. It squashes the input value from plus and minus infinity to an output value of a range of between zero and one. It is differentiable as mentioned in Chapter 2.2 so the derivative is easy to calculate, which is helpful for calculating the weight updates in the back propagation training algorithm. The linear transfer function was used in the output layer.

Number of the network's hidden layers

It has been found empirically that a single hidden layer is sufficient for modelling most data sets. Additional hidden layers allow the neural network to model more complex functions. Usually just one hidden layer is required to solve most nonlinear problems [50, p.83], [9, p.393].

Size of the network input layer

It should be noted that 3 half-hour data points equals 1.5 hours and 30 half-hour data points equals three quarters of a day (which could be lost for starting training). So it was

important to keep the input layer size as small as possible if one wants to keep within the constraints set out to use the minimum amount of input variables used to predict the load in 3.2.1

Number of neurons in the network hidden layer(s)

Too many neurons in the hidden layer(s) will make the model overfit the data. In addition, if the net is too large, it will memorize rather than learn [40, p.684]. Use too few, then the model may not have sufficient power to fit the data. The neurons in the hidden layer(s) allow the network to:

- detect the feature
- capture the pattern in the data and
- perform the complicated non-linear mapping between the actual and forecasting variables [38, pp.143-149].

The appropriate number of hidden neurons is system-dependent [41], based on the investigator's experience.

From the above criteria, the first stage was to determine the size of the network input layer and the number of neurons in the hidden layer heuristically using the following procedure:

- The electric load data was pre-processed to fall in the range [-1, 1]. Then the training set of 336, half-hour sampled, electric load values were presented to the network sequentially, training the network for one week at a time, for the four weeks of July 2008. The weights and biases where updated after each input vector was presented to the network using supervised learning. Post-processing was done to convert the electric load data back to its standard value of kW units.
- The fourth week in July 2009 (Saturday 25th July 2009 12:00 am to Friday 31st July 2009 12:00 pm) was arbitrarily chosen to measure the performance of the network architecture.

- To estimate the input layer size experimentally, the number of input data points presented to the network was varied incrementally from 3 to 30 points while an additional neuron was added to the hidden layer at the end of the incremental period every time the whole of the input sliding window was presented to the ANN, up to the point where there were 3 neurons in the hidden layer.

The results are shown in Table 3.2.

Table 3.2: Simultaneous recording of the network input size and the number of neurons in the hidden layer using a feedforward network

Network Input size	Network Prediction error using MAPE with 1 neuron hidden layer (%)	Network Prediction error using MAPE with 2 neurons hidden layer (%)	Network Prediction error using MAPE with 3 neurons hidden layer (%)
3	4.01	4.2	4.41
6	3.82	4.1	4.29
9	3.7	4.14	4.57
12	3.65	4.3	4.31
15	3.66	4.91	4.7
18	3.61	4.26	5.26
21	3.62	4.05	4.99
24	4.09	4.42	4.63
27	4.11	4.64	5.29
30	4.1	4.5	4.7

Table 3.2 shows that as more neurons are added to the hidden layer, the MAPE values become larger. Using these results, a network with one neuron in the hidden layer was decided upon for further investigation. From figure 3.9 it can be seen that there is little fluctuation in the MAPE when using one neuron/node in the hidden layer when the network input size increases from 3 to 30 data points.

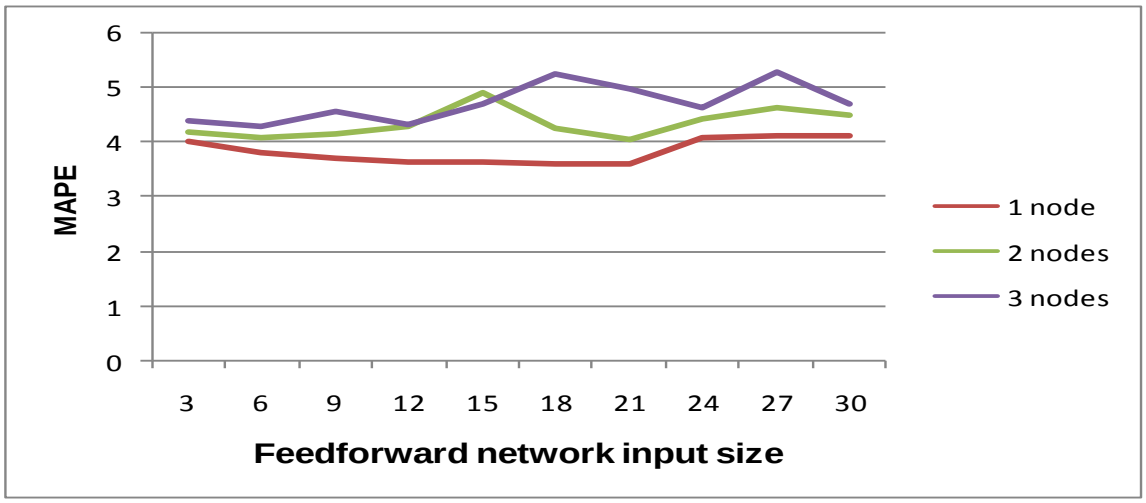


Figure 3.9: Plotting the results in Table 3.2 where the hidden layer of 1, 2 and 3 nodes (or neurons) were used for comparison

For further illustration, the graphs in Figure 3.10, Figure 3.11 and Figure 3.12 are shown for a visual inspection of the difference between the actual and predicted peaks and valleys of the two loads.

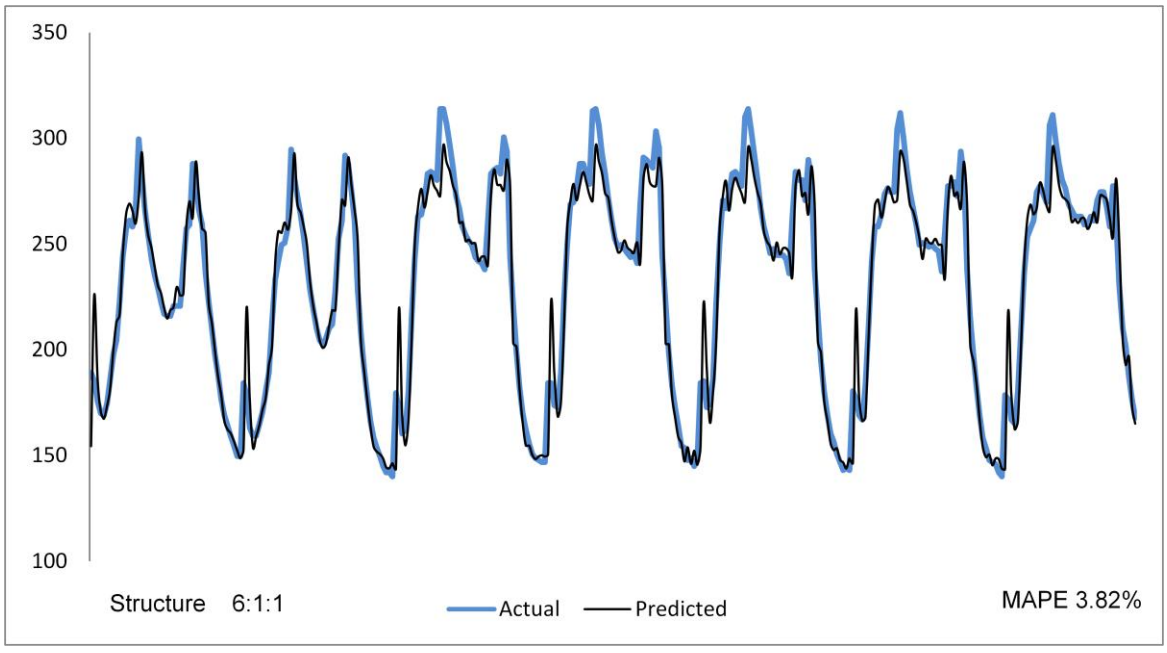


Figure 3.10: Plotting the forecasting results of the 6 input feedforward network with a hidden layer of one neuron

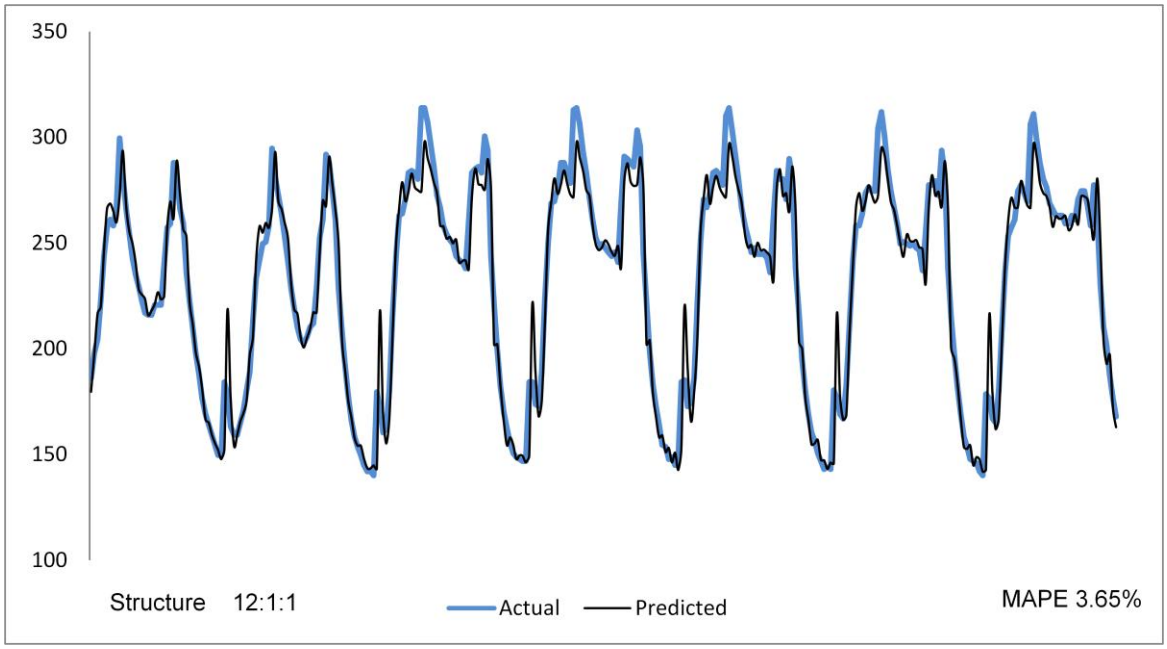


Figure 3.11: Plotting the forecasting results of the 12 input feedforward network with a hidden layer of one neuron

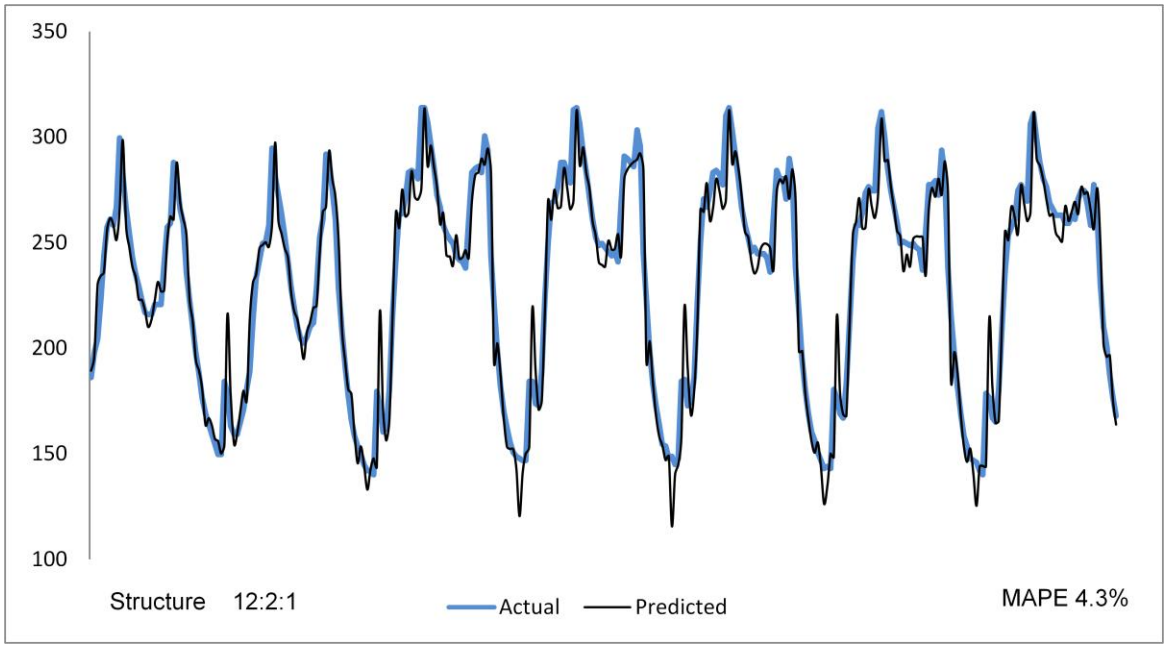


Figure 3.12: Plotting the forecasting results of the 12 input feedforward network with a hidden layer of two neurons

One cannot use the MAPE values alone to look for the most suitable network. Also, the MAPE value of two sets of tabled results can both be more or less the same but the overall “fit” of the predicted values on the actual values can look very different.

From the accompanying MAPE values in Table 3.2 and comparing Figures 3.10, 3.11 and 3.12 using visual inspection of the peaks and valleys, Figure 3.11 shows the best fit of Figures 3.10, 3.11 and 3.12.

The 12:1:1 architecture of the feedforward network with a MAPE of 3.65 % illustrated in Figure 3.11 would be chosen for future comparison with the next network to be investigated, the cascade forward network.

3.5.3 Selecting the cascade forward network

The next stage was to apply the same training and testing data sets (July 2008 and week 4 - July 2009) used previously, to a new cascade forward network to optimize its network input layer size.

Figure 3.13 shows the results when using a cascade forward network with the same log-sigmoid transfer function in the hidden layer and a linear transfer function in the output layer.

Levenberg-Marquardt training with Bayesian regularization was used to determine the size of the input window. Once more, for further illustration, the MAPE results from Figure 3.13 and the graphs in Figure 3.14 to Figure 3.16 are shown as a guideline for visual inspection to decide on a suitable cascade forward network for comparison with the feedforward network in the next step of this study.

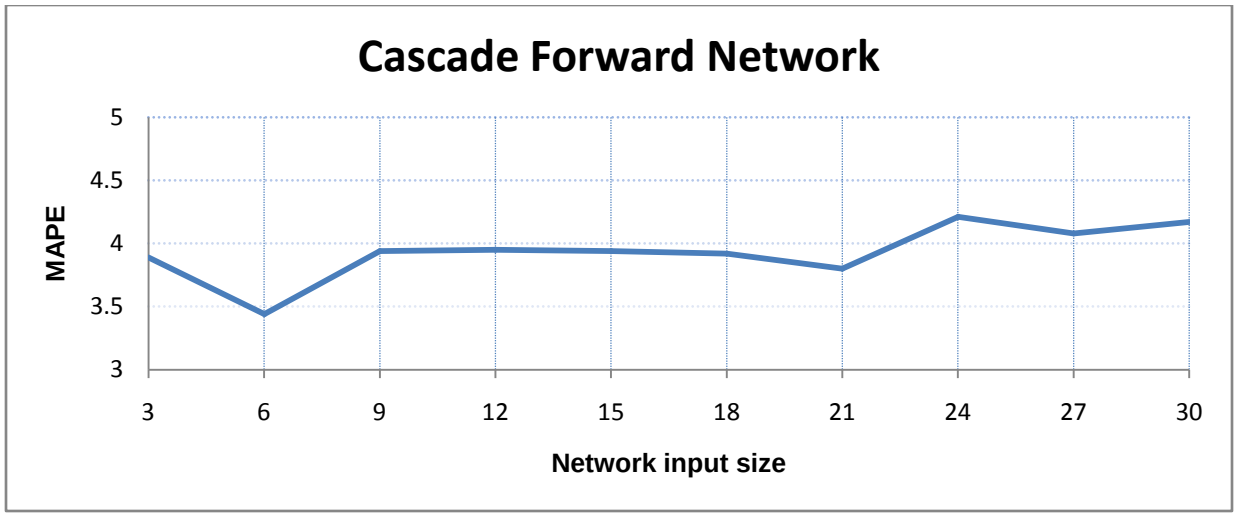


Figure 3.13: The graph shows the lowest MAPE at a network input size of 6 data points and one neuron in the hidden layer

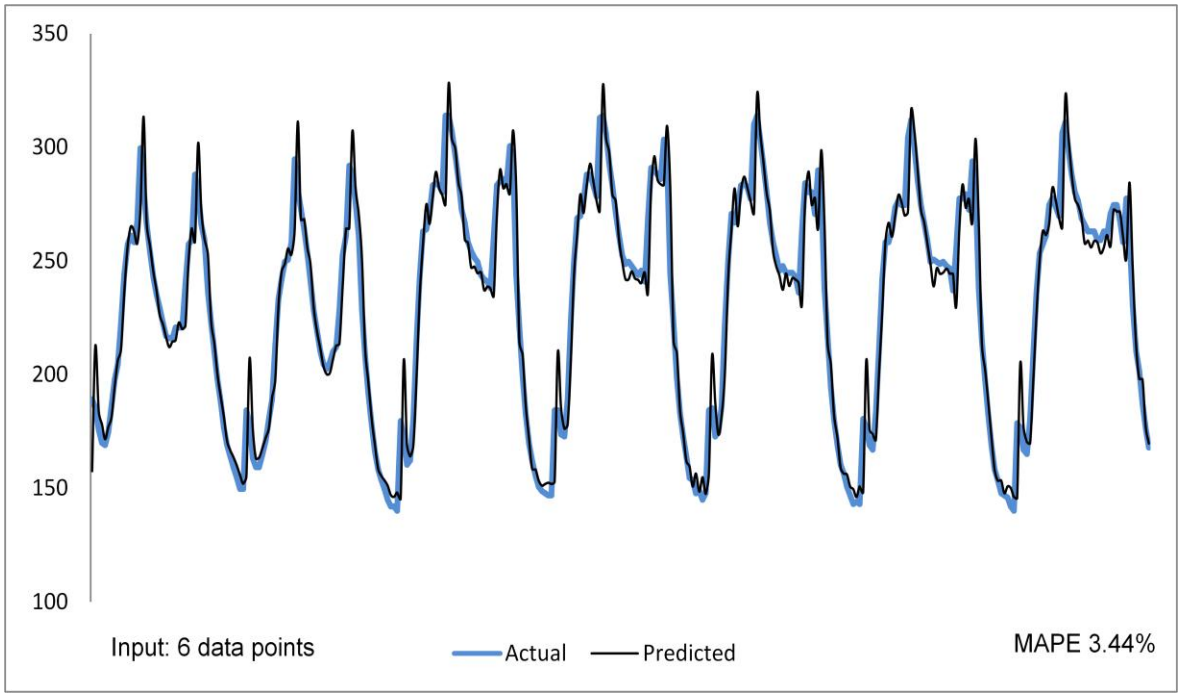


Figure 3.14: Plotting the forecasting results of the 6 input cascade forward network with a single neuron in the hidden layer

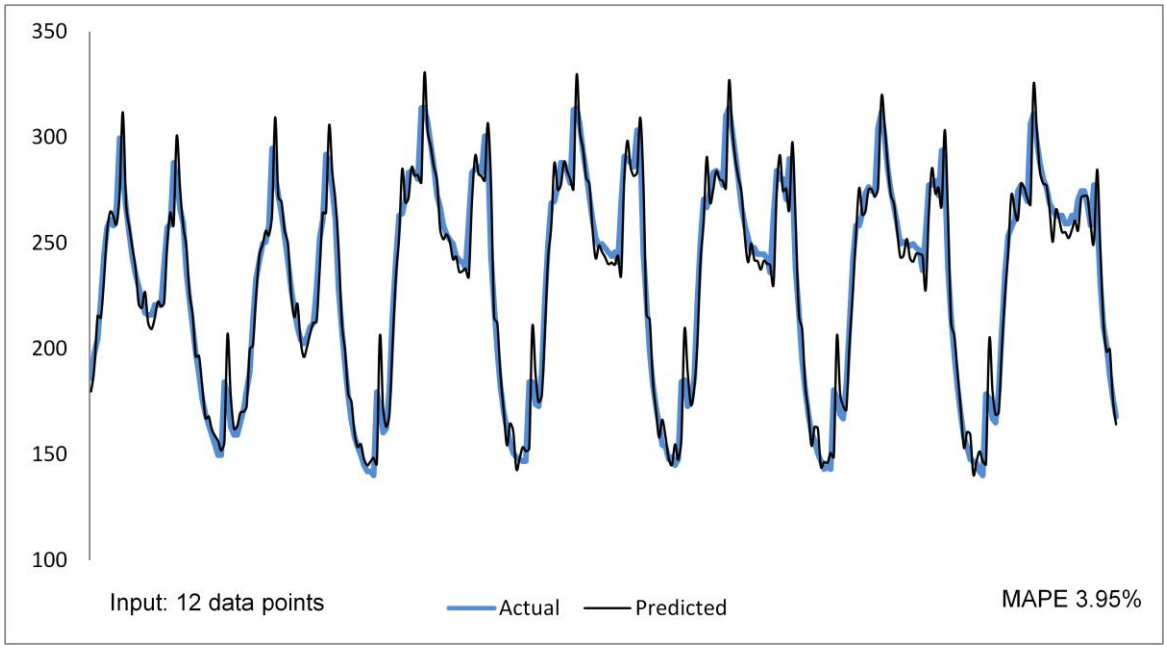


Figure 3.15: Plotting the forecasting results of the 12 input cascade forward network with a single neuron in the hidden layer

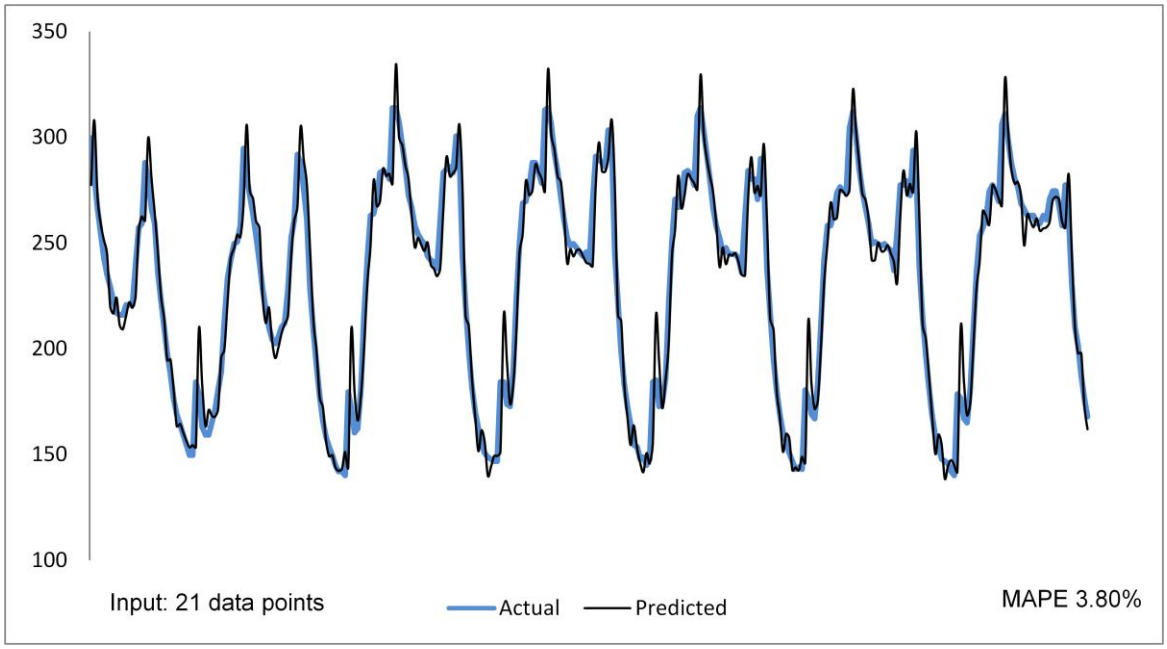


Figure 3.16: Plotting the forecasting results of the 21 input cascade forward network with a single neuron in the hidden layer

Having decided on the feedforward configuration as shown in figure 3.11 the next step would be to choose the **cascade** configuration by comparing Figures 3.14, 3.15 and 3.16 using visual inspection of the peaks and valleys. The structure in Figure 3.14, a 6:1:1 cascade forward network with a MAPE value of 3.44 % was chosen for further comparison with the 12:1:1 feedforward network from Figure 3.11 with a MAPE value of 3.65 %. This was decided from the lowest MAPE value obtained, shown in Figure 3.13.

This completed the tests to select the input set size and the hidden layer size for the 12:1:1 feedforward network and the 6:1:1 cascade forward network.

3.5.4 Final selection of the feedforward or the cascade forward network

The final selection was done again using the same training set (four weeks in July 2008) and testing week 4 in July 2009.

Training the two neural networks involved the following actions:

- The two networks' respective input layer sizes were kept constant;
- Training occurred according to default neural toolbox parameters that were left constant for all the trials;
- The hidden layer transfer functions were altered between Log-sigmoid and Tan-sigmoid for each network;
- Selecting the Levenberg-Marquardt training algorithm with and without Bayesian regularisation; and
- Finally, comparing the forecasting errors of the two networks.

The results of this training are shown in Table 3.3.

Table 3.3: Results when training the two different networks with the data of week 1,2,3 and 4 of July 2008 and forecasting week 4 of July 2009

Network Tested	Size of Input layer	Hidden Layer Transfer Function	Hidden layer size	Output Layer Transfer Function	Output layer size	Network Training Algorithm	% Error (MAPE)
Newcf	6	Logsig	1	Purelin	1	Trainlm	**4.04 %**
Newcf	6	Tansig	1	Purelin	1	Trainlm	**4.44 %**
Newff	12	Logsig	1	Purelin	1	Trainlm	**3.69 %**
Newff	12	Tansig	1	Purelin	1	Trainlm	**3.69 %**
Newcf *	6	Logsig	1	Purelin	1	Trainbr	**3.44 %**
Newcf	6	Tansig	1	Purelin	1	Trainbr	**3.92 %**
Newff **	12	Logsig	1	Purelin	1	Trainbr	**3.65 %**
Newff	12	Tansig	1	Purelin	1	Trainbr	**3.65 %**

Where the MATLAB functions used:

Newff is the feedforward back propagation network architecture.

Newcf is the cascade forward network architecture.

Tansig is the tan-sigmoid transfer function.

Logsig is the log-sigmoid transfer function.

Purelin is the linear transfer function where the input = output.

Trainlm is the Levenberg-Marquardt training function.

Trainbr is the Bayesian regularization training function.

In Table 3.3 the following can be observed:

- Empirically there is little difference between using the different network algorithms with different hidden layer size transfer functions and different network training algorithms if one looks at the MAPE results.
- Both networks delivered MAPE results below 5 % as considered.
- Comparing the best overall performance and fit between the 6:1:1 cascade forward network with a MAPE of 3.44 % and the 12:1:1 feedforward network with a MAPE of 3.65 % (marked with a single and a double asterisk in Table 3.3), the cascade forward network was selected because of the better MAPE value and a better visual fit (Compare the amplitudes of the peaks and valleys in Figure 3.11 and Figure 3.14).

3.5.5 Training with the correct set of data

The next stage that was investigated concerned the following two questions:

1. How far ahead can the neural network forecast when it is trained with last year's monthly data?

2. Would adding a fifth "average week" to the four weeks of training data, increase the network's forecasting flexibility by reducing over fitting?

To test question 1 involved training and testing the cascade forward network's forecasting accuracy over a period of three successive years and then over two successive years as follows:

- Electric load data for the months of May, June and July 2007 to 2009 was structured in four week sets for presentation to the neural network to be trained and tested.
- The data sets were divided into three. The training set for the month in year 2007 and the two test sets for the months in 2008 and 2009.
- Four weeks in the three months of May 2007, June 2007 and July 2007 (one week at a time) were used to train three cascade forward networks.
- Four weeks in the three months of May 2008, June 2008 and July 2008 (one week at a time) and four weeks in the three months of May 2009, June 2009 and July 2009 (one week at a time) were now used to test the forecasting accuracy of the three trained cascade forward networks.

The results for one year's training and two year's testing are shown in Tables 3.4, 3.5 and 3.6.

Table 3.4: MAPE results when testing May 2008 and May 2009 with the 2007 trained network

MONTH YEAR	MAY 2007	MAY 2008	MAY 2009
ACTION	TRAIN	FORECAST	FORECAST
WEEK 1	○	3.77 %	5.88 %
WEEK 2	○	3.93 %	5.89 %
WEEK 3	○	3.84 %	4.68 %
WEEK 4	○	3.67 %	5.03 %

Table 3.5: MAPE results when testing June 2008 and June 2009 with the 2007 trained network

MONTH YEAR	JUNE 2007	JUNE 2008	JUNE 2009
ACTION	TRAIN	FORECAST	FORECAST
WEEK 1	○	3.55 %	3.71 %
WEEK 2	○	3.38 %	3.73 %
WEEK 3	○	3.21 %	4.64 %
WEEK 4	○	3.36 %	4.5 %

Table 3.6: MAPE results when testing July 2008 and July 2009 with the 2007 trained network

MONTH YEAR	JULY 2007	JULY 2008	JULY 2009
ACTION	TRAIN	FORECAST	FORECAST
WEEK 1	○	3.37 %	4.42 %
WEEK 2	○	3.53 %	4.36 %
WEEK 3	○	3.59 %	4.43 %
WEEK 4	○	3.56 %	3.97 %

Observing the MAPE results in Tables 3.4, 3.5 and 3.6 there is a slight increase in error from 2008 to 2009 when applying the same network to predicting the load for two consecutive years in a row. The results for one year's training and the next year's testing are shown in Tables 3.7, 3.8 and 3.9:

Table 3.7: MAPE results when training May 2008 and testing May 2009 with the 2008 trained network

MONTH YEAR	MAY 2008	MAY 2009
ACTION	TRAIN	FORECAST
WEEK 1	○	4.24 %
WEEK 2	○	4.21 %
WEEK 3	○	3.85 %
WEEK 4	○	3.78 %

Table 3.8: MAPE results when training June 2008 and testing June 2009 with the 2008 trained network

MONTH YEAR	JUNE 2008	JUNE 2009
ACTION	TRAIN	FORECAST
WEEK 1	○	2.71 %
WEEK 2	○	2.87 %
WEEK 3	○	2.94 %
WEEK 4	○	3.12 %

Table 3.9: MAPE results when training July 2008 and testing July 2009 with the 2008 trained network

MONTH YEAR	JULY 2008	JULY 2009
ACTION	TRAIN	FORECAST
WEEK 1	○	3.57 %
WEEK 2	○	3.70 %
WEEK 3	○	3.63 %
WEEK 4	○	3.44 %

Using Table 3.10 to compare the MAPE results in Table 3.7, 3.8 and 3.9 to the results in Table 3.4, 3.5 and 3.6 there is a slight reduction in the error when training the same network with only the previous year's data. The error gets slightly worse when using the 2007 trained network for forecasting two consecutive years in a row. It was then decided that the previous year's monthly data is enough to train the network to predict the next year's monthly load.

Table 3.10: Comparing the MAPE results in Table 3.4, 3.5 and 3.6 to the results in Table 3.7, 3.8 and 3.9

MONTH AND YEAR	TRAINED MAY 2007 FORECAST MAY 2009	TRAINED MAY 2008 FORECAST MAY 2009	TRAINED JUNE 2007 FORECAST JUNE 2009	TRAINED JUNE 2008 FORECAST JUNE 2009	TRAINED JULY 2007 FORECAST JULY 2009	TRAINED JULY 2008 FORECAST JULY 2009
WEEK 1	5.88 %	4.24 %	3.71 %	2.71 %	4.42 %	3.57 %
WEEK 2	5.89 %	4.21 %	3.73 %	2.87 %	4.36 %	3.70 %
WEEK 3	4.68 %	3.85 %	4.64 %	2.94 %	4.43 %	3.63 %
WEEK 4	5.03 %	3.78 %	4.5 %	3.12 %	3.97 %	3.44 %

To test question 2 (increasing the networks forecasting flexibility by reducing over fitting?) involved adding a fifth "average week" to the four weeks of training data.

To test this question, the procedure used to train the cascade forward network involved the following stages:

- Electric load data was structured in five week sets (four weeks plus the monthly average as a fifth week), in the month of July 2006 to 2008, and separately presented to the neural network for training.
- The training set for the network containing the data only for each month of a year e.g. four weeks plus the average week value in July 2006, one week at a time, were presented to the network.
- Once the network was trained with the 5 sets of weekly parameters of July 2006, it was tested using the four weeks of July 2007 and so on for 2008 and 2009.

The training set = week 1+week 2+week 3+week 4+ weekly average

where the

$$\text{weekly average} = \frac{\text{week 1+week 2+week 3+week 4}}{4}$$

Then:

Train using 4 weeks July 2006 + average ► forecast July 2007
Train using 4 weeks July 2007 + average ► forecast July 2008
Train using 4 weeks July 2008 + average ► forecast July 2009

Training July 2006, 2007 and 2008 without the average values then testing 2009 with the 2008 trained network

week 1+week 2+week 3+week 4

Then:

Train July 2006 with no average ► Forecast week 4 July 2007
Train July 2007 with no average ► Forecast week 4 July 2008
Train July 2008 with no average ► Forecast week 4 July 2009

The results when training and testing the cascade forward network with and without the average week, measuring the forecasting accuracy using MAPE, are shown in Figures 3.17 to 3.19.

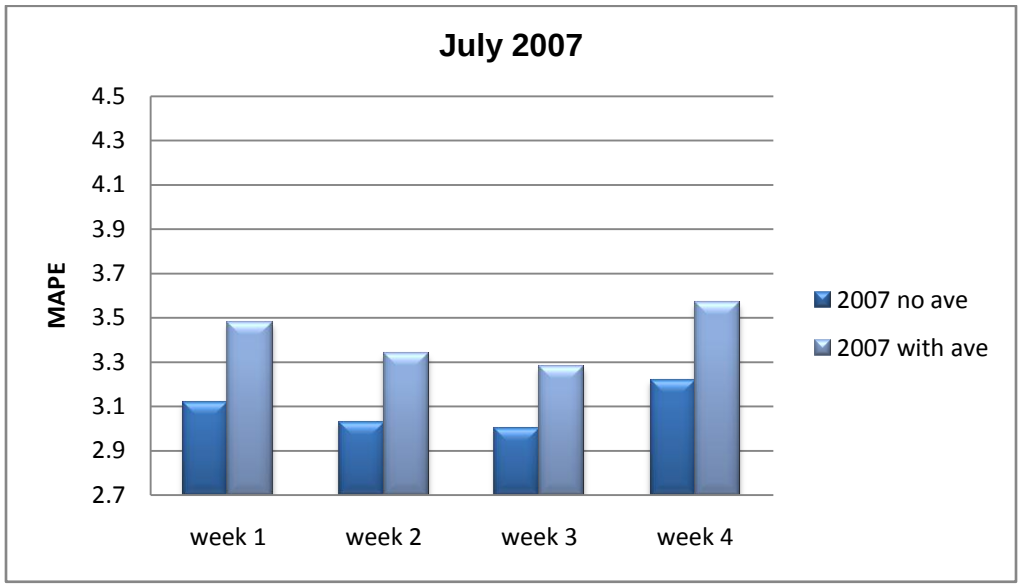


Figure 3.17: MAPE values for July 2007 when training without and with the average

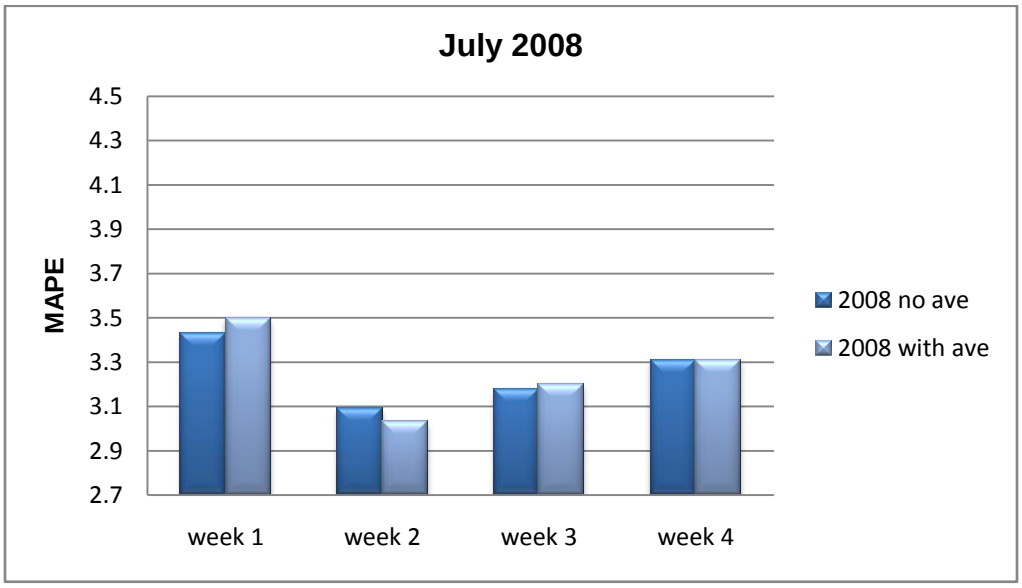


Figure 3.18: MAPE values for July 2008 when training without and with the average

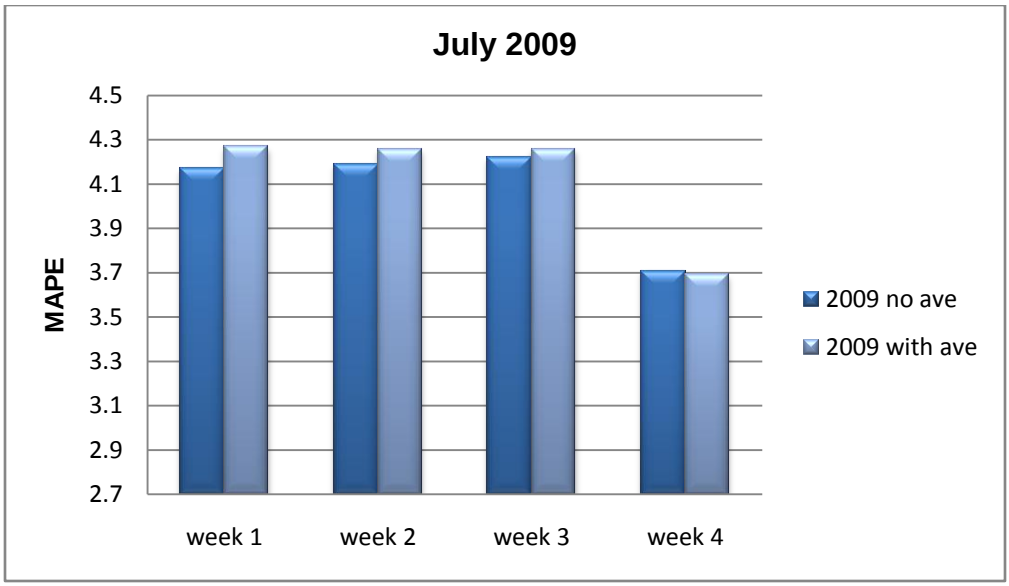


Figure 3.19: MAPE values for July 2009 when training without and with the average

From Figure 3.17 to Figure 3.19, comparing the difference between the network trained with an average value and the same network trained with no average value, the overall difference in the MAPE is smaller than 0.5% for July 2007, 2008 and 2009. So in this instance, the influence of averaging the monthly data can be disregarded for training purposes.

3.6 Summary

A forecasting procedure, mentioned in 3.1, was used to develop and test the performance of an ANN to predict the annual peak electric load during the late autumn and winter period of the three months of May, June and July in 2007, 2008 and 2009.

Time series plots of the electric load data was created and visually inspected for familiar patterns, e.g. trends, seasonal components and abrupt changes in magnitudes of peaks and valleys. Next, the load curve shape of the different day types of the week of each month was separated, each day type having its own distinctive features. The daily rhythm changes throughout the year, comprising continuous changes in normal and special days, weekdays and weekend load curve cycles.

The special days, i.e. public holidays (see Fig. 3.3), differ from weekdays and one also had to look at its load curve shapes to see if the network needed to be trained especially for this occurrence. Investigating the profile of the electric load data plotted in Figure 3.6 to Figure 3.8 for the year of 2009 it can be seen that on the special days of the year the load consumption curves follows roughly the same pattern as on a Sunday in that same week, so it was not deemed necessary to additionally train a network particularly for the special days.

As it was pointed out above, the load data sets which would be presented to the neural networks were not large and did not consist of multiple arrays of input variables such as temperature, active power, reactive power and day type necessitating the use of building complicated network architectures. So, in order to arrive at an appropriate robust network topology, the structure and performance of a multilayer feedforward network was investigated first.

In order to develop an understanding of and gain experience in how this load forecasting model would develop, an initial, experimental arrangement of the multilayer feedforward network architecture was set up to train and test the different load data configurations. It included determining input and hidden layer size, utilising a MAPE of less than 5% as a benchmark. Initial load forecasting simulations served as a reference against which to select the developing forecasting models' architecture.

The next configuration looked at was a cascade forward network with six electric load parameters as input units. One neuron was used in the hidden layer and one neuron in the output layer. The log sigmoid activation function was used in the hidden layer and the pure linear transfer function was used in the output layer respectively. This network input layer size was tested systematically, again using a MAPE of less than 5% as a benchmark.

Each of the two ANN topologies were developed using three stages: a training, testing and evaluation stage.

Once the input set size and the hidden layer size of the two neural network models were determined, more performance tests were done to select between the Levenberg-Marquardt with and without the Bayesian regularization training algorithm, to fine-tune the final network's performance (indicated in Table 3.3).

The 6:1:1 cascade forward network was finally selected for the next phase of investigation since it was better at detecting the peak load levels and capturing the pattern in the data.

The next stage that was investigated concerned the following two questions:

1. How many years ahead can the neural network forecast when it is trained with a year's monthly data?

2. Would adding a fifth "average week" to the four weeks of training data, increase the networks forecasting flexibility by reducing over fitting?

To test question 1 involved training and testing the cascade forward network's forecasting accuracy over a period of three successive years and then over two successive years. The MAPE results in Table 3.10 differ only slightly when comparing the 6:1:1 cascade forward networks performance for forecasting one year and then for two consecutive years in a row. So it was decided that the one year's historical monthly data was enough to train the network to predict the next year's monthly load.

To test question 2 involved adding a fifth "average week" to the four weeks of training data. From Figure 3.17 to Figure 3.19, comparing the difference between the network trained with an average value and the same network trained with no average value, the overall difference in the MAPE is smaller than 0.5% for July 2007, 2008 and 2009.The influence of averaging the monthly data was disregarded for training purposes as the overall difference in the simulated MAPE results was smaller than 0.5% for July 2007, 2008 and 2009.

The method developed here considered factors such as the time frame, periodic characteristics of the load pattern of data, desired accuracy, availability of data, ease of operation and understanding.

From the test runs made, and the results tabulated in Chapter 3, the 6:1:1 cascade forward network produced a better fit. So, it was used for the forecasting performance evaluation in Chapter 4 to obtain the final results.

CHAPTER 4

DATA-BASED RESULTS

As mentioned in section 3.2, the aim and objectives was to build a dynamic, half-hour ahead updating, forecasting model with a MAPE of less than 5%. The method used to examine the neural network's performance was to use one data set to train and test the neural network model and another set to validate it**.** (In this instance, the word "set" means four weeks of data selected in a month of choice, chosen for investigation).

To obtain the experimental results shown in this chapter, the performance of the neural model selected in Chapter 3 was measured by training the network with the four week data set of May 2006 and evaluating it against the four week data set of May 2007, measuring the MAPE, one week at a time, shown in the sequence:

1. Input May 2006 data ► train a new network ► compare forecasted and actual May 2007 data to validate the trained network.
2. Input May 2007 data ► present to the taught network ► compare forecasted and actual May 2008 data to validate the trained network.

Then, the same procedure was used for May 2008, May 2009 and repeated for the months of June, from 2007 to 2009 and July, from 2007 to 2009.

4.1 An examination of the data presented to the neural forecasting model

A new network with a new set of random weights was trained for each of the nine months used for testing. In this manner, the results obtained from the different networks' performance of nine sets of four weeks of forecasted load curves was measured: May 2007, 2008, 2009, June 2007, 2008, 2009 and July 2007, 2008 and 2009. As an illustration, Figure 4.1 shows the data presented to the July 2009 neural network for training. The vertical lines separate the weekly data sets from each other.

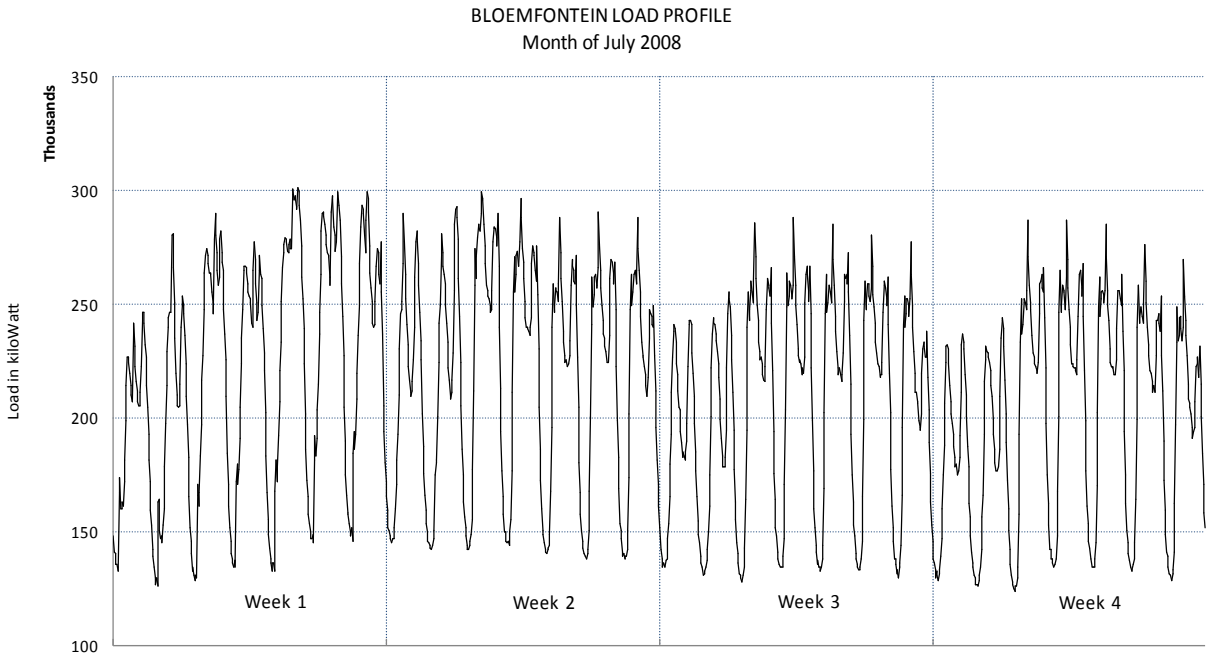


Figure 4.1: Four weeks of data for the month of July 2008 is shown, which would be used for training the network for forecasting the month of July 2009

Figure 4.2 shows the data used to measure the performance of the trained network's forecasting capability.

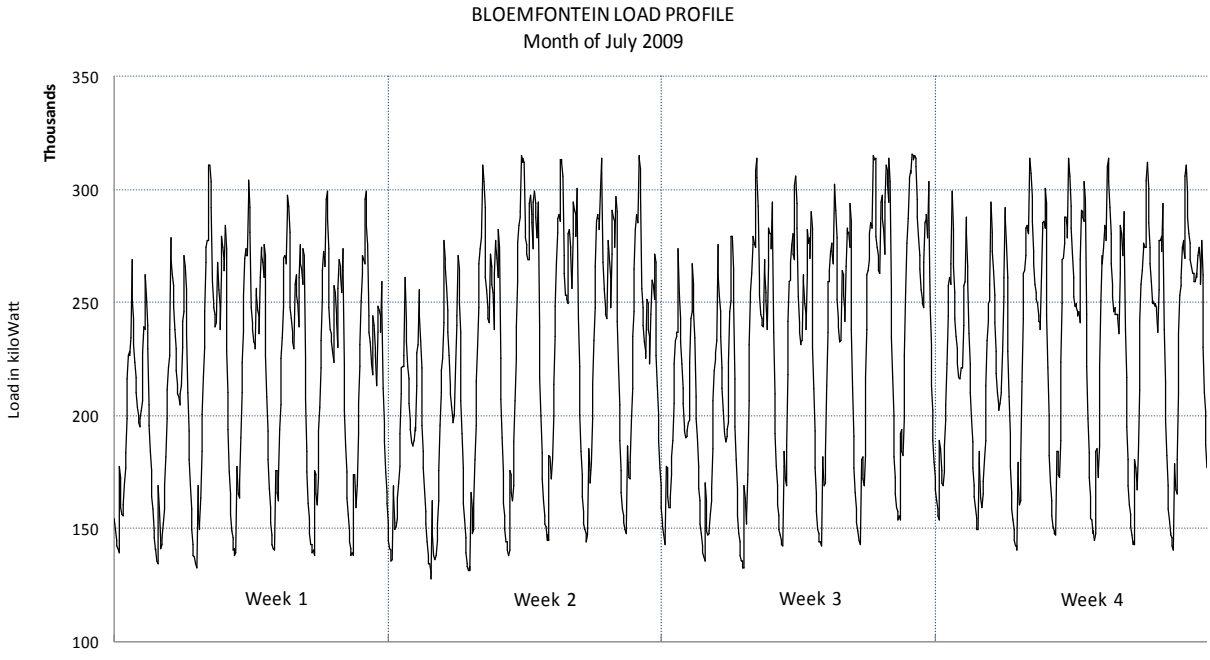


Figure 4.2: All the actual output data for July 2009 week 1, 2, 3 and 4 is shown, which would be used to validate the trained network's forecasting potential

Three examples of the actual data used to train, test and validate the neural networks for week 1 in the month of May 2009, week 3 in the month of June 2007, and week 2 in the month of July 2008 are shown below. These examples would indicate the different spatial attributes and the daily peak load level differences between the training and the actual weekly load curves to be predicted, to the reader.

Figure 4.3 shows all the training data of the month of May 2008 and the actual week 1, May 2009. Most of the training and actual data is “masked in each other” and “seems” much easier for the network to learn when doing the actual forecasting.

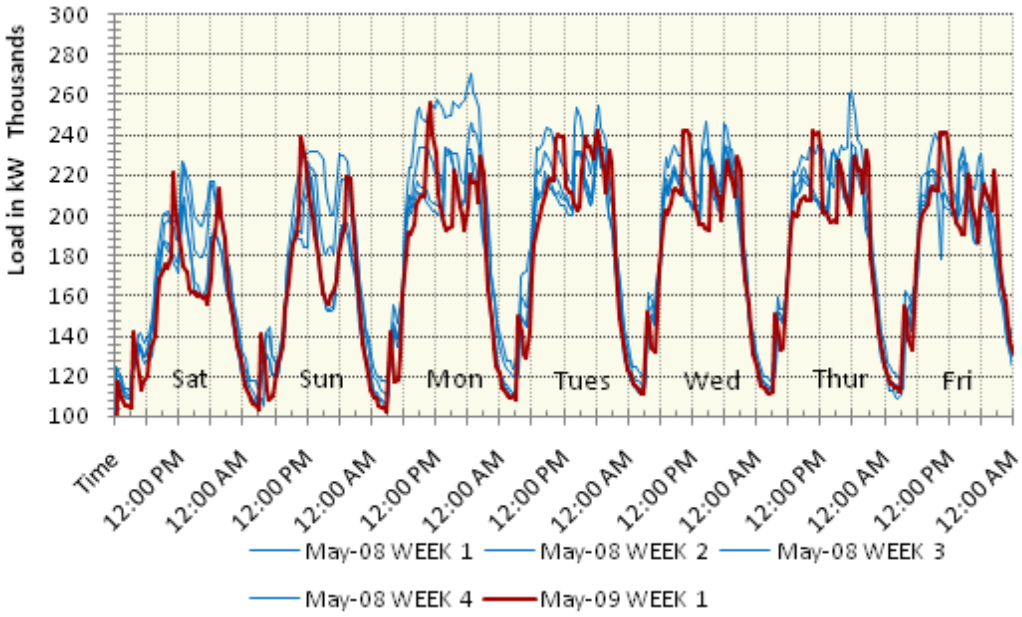


Figure 4.3: All the training load data for May 2008; week 1, 2, 3 and 4 is shown in comparison with the actual load data for week 1 in May 2009

The forecasted results of the 6:1:1 cascade forward network for week 1 of May 2009 can again be seen in figure 4.4. Week 1 in Fig. 4.4 is from the 2nd of May 2009 12:30 AM to the 8th of May 2009 24:00 PM.

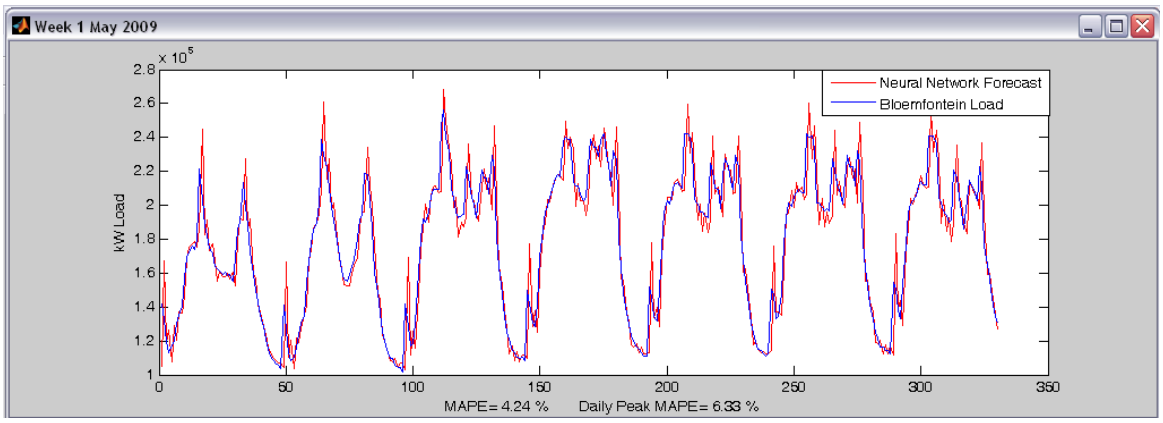


Figure 4.4:The actual and forecasted values for week 1 in May 2009

Figure 4.5 shows the shapes of all four load curves used to train the forecasting network for June 2007. It can be seen that the network, once it has been trained, may have to predict a load curve with higher peak demand values than the peak values of the load curves with which it has been trained.

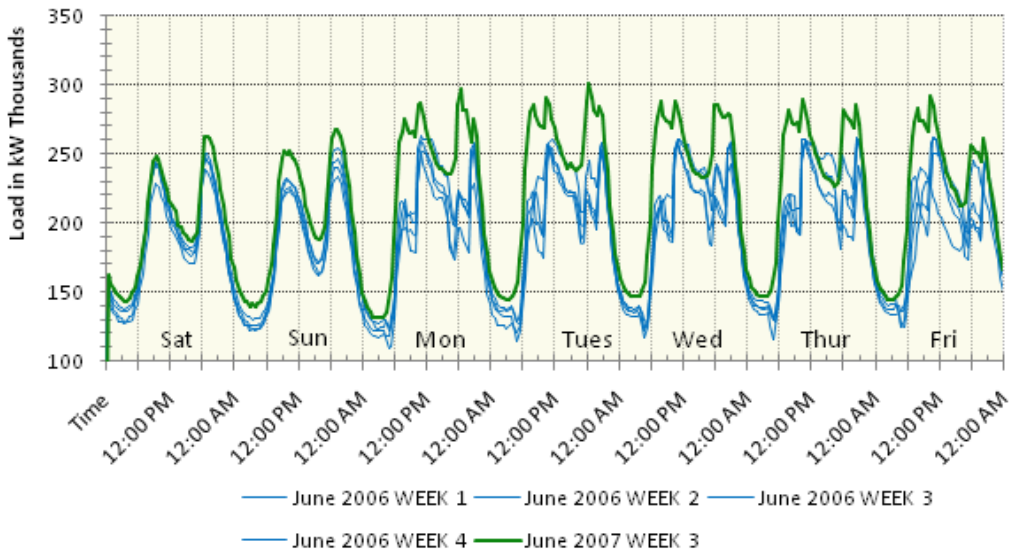


Figure 4.5: All the training data for June 2006; week 1,2,3 and 4 is shown for comparison with the actual data for Week 3 in June 2007

The actual performance of the 6:1:1 cascade forward network, forecasting week 3 of June 2007, can be seen in Figure 4.6. In this case, it seems that the network can estimate data points that lie outside the training set as shown in Fig. 4.5, e.g. Sunday, Monday, Tuesday, Wednesday, Thursday and Friday.

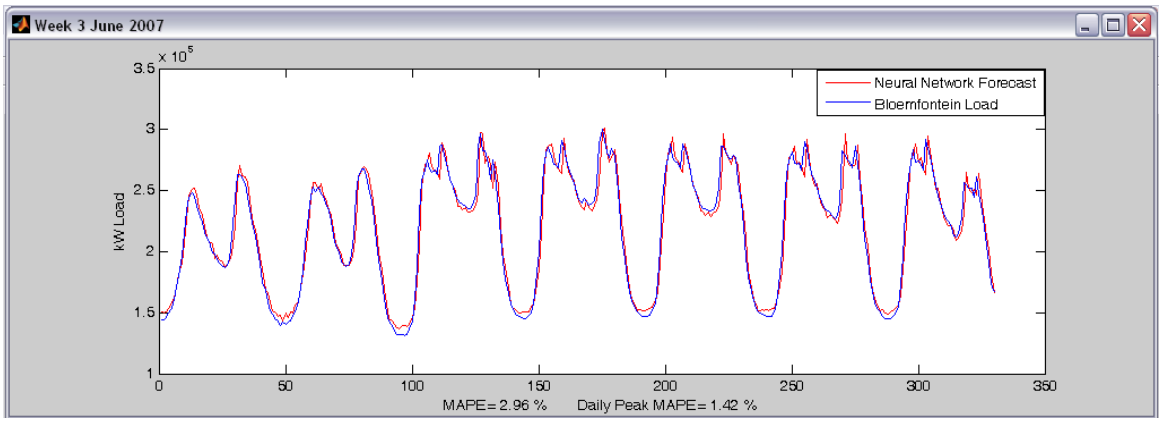


Figure 4.6: The actual and forecasted values for week 3 in June 2007

Week 3 in Fig. 4.6 is from the 16th of June 2007 12:30 AM to the 22nd of June 2007, 24:00 PM**.**

Figure 4.7 shows all the training data of the month of July 2007 and the actual week 2 of July 2008. The training data amplitudes for the first two days, Saturday and Sunday, are lower than the actual data to be predicted and the training data for Monday to Friday, all plotted in blue and the actual data, plotted in red, is again “masked in each other”

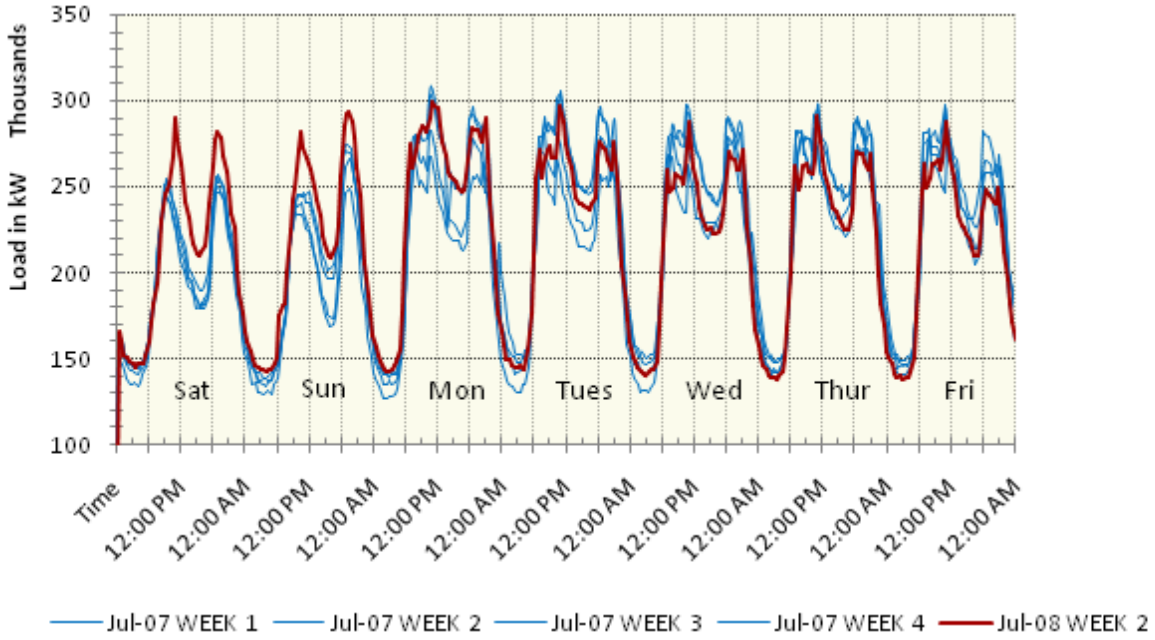


Figure 4.7: All the training data for July 2007 week 1, 2, 3 and 4 is shown for comparison with the actual data for week 2 in July 2008

The forecasting result of the 6:1:1 cascade forward network for week 2 July 2008 is shown below in Figure 4.8. Week 2 in Fig. 4.8 is from the 12th of July 2008 12:30 AM to the 18th of July 2008, 24:00 PM. Again, it seems that the network can estimate data points that lie outside the training set as shown for Saturday and Sunday in Fig. 4.7.

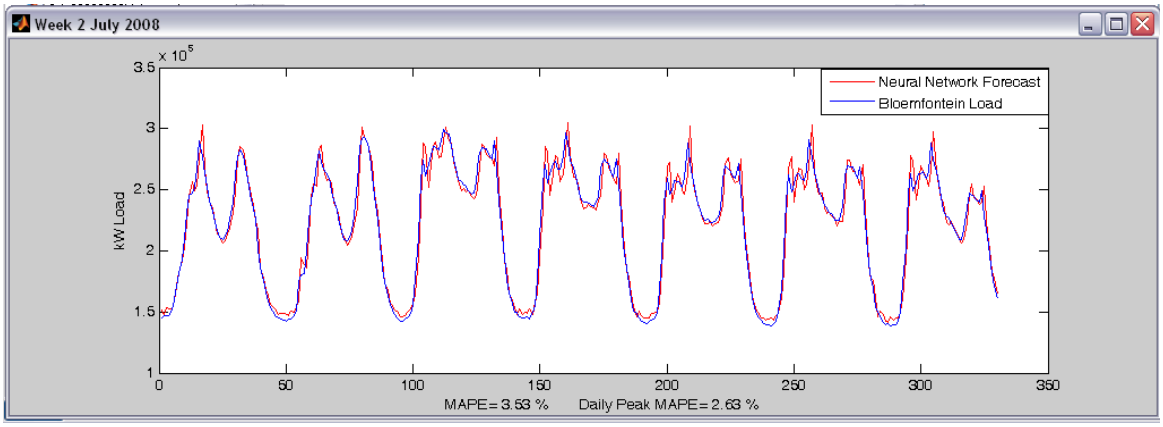


Figure 4.8: The actual and forecasted values for week 2 in July 2008

There is little correlation between the training patterns presented to the network and the forecasting results produced by the network. The three examples show that data points that lie outside the training set are predictable with the 6:1:1 cascade forward network.

From the above three examples it can be observed that visually or empirically it is very difficult to compare the network's performance to the depth of training it receives, meaning that the proper training sequence or training data set's magnitude is very difficult to relate to the network's forecasting accuracy.

After discussing the case study, the weekly MAPE and correlation coefficient (R) results for each of four weeks of May, June and July, 2007, 2008 and 2009 will be shown as a group in Figures 4.9, 4.10, 4.11 and 4.13 below for examination, to note the similarities or differences in performance of each monthly forecasting network built for the nine sets of results.

4.2 Case study

4.2.1 The forecasting model performance for July 2009

A visual check of the accuracy of the forecasts is often the most powerful method for determining whether the configured artificial neural network model fits the data adequately. Scientifically this has little meaning. In addition, it would be meaningless to talk about "overall or the average value of this or that…" Instead, the MAPE and regression graphs were used to analyse and verify the 6:1:1 cascade forward network's mapping of the shape of the load curves' performance in this case study.

Seven days contains 48 half–hour data points per day, which gives 336 data points available per week. The first six data points of each week were used to start off the sequential training sequence of the neural network. Of the 336 data points available, only 330 data points were plotted in each graph and used to calculate the displayed MAPE values and daily peak MAPE values, using the MAPE equation (2.46), adapted from [10].

The purpose of this case study was to observe the following:

- Measure the trained network's performance, i.e. the forecasting accuracy, using the MAPE, which will be to check that the *prediction error is repeatable from week to week*. Accuracy is limited by systematic (repeatable) errors as mentioned in Chapter 3, Section 3.2.

- How well the network detects the daily peak load levels [53, p.1397]. From a practical point of view, the forecasting error is usually less critical at off-peak load levels, e.g. Direct Load Control (DLC) levels compared to peak load levels.

- Using linear regression to compare the correlation coefficients (equation 2.50) for week 1 to week 4, obtained from the scatter plots, for July 2009's actual and forecasted results.

The plotted graphs on the following pages show the sequence in which the 6:1:1 cascade forward network's performance was investigated:

1. Training the network

Four weeks of the training data from July 2008 which were used for the training set, and the four weeks of actual load consumed, which will also be used to do the forecasting, in July 2009, are shown in Fig, 4.9, for comparison. Note that the daily peak load levels for example in week three in Fig 4.9 differs considerably if one compares the training data (red plot) with the actual load data (blue plot).

2. Testing the trained network

Next, the same four weeks of training data from July 2008 were presented as a test set to the trained cascade forward network. The network's predicted results were compared with the training data using the MAPE and daily peak MAPE performance as a measure of its training capability, shown in Fig.4.10.

3. Validating the trained network

The four weeks of data from July 2009 were presented as a validation set to the trained network to check its MAPE and daily peak MAPE performance, shown in Fig.4.11.

Day type and hour of the day details are irrelevant and therefore absent on the x-axis of each weekly plot in Figures 4.9, 4.10 and 4.11.

The y-axis in Figures 4.9, 4.10 and 4.11 represents the kW Load used in each week.

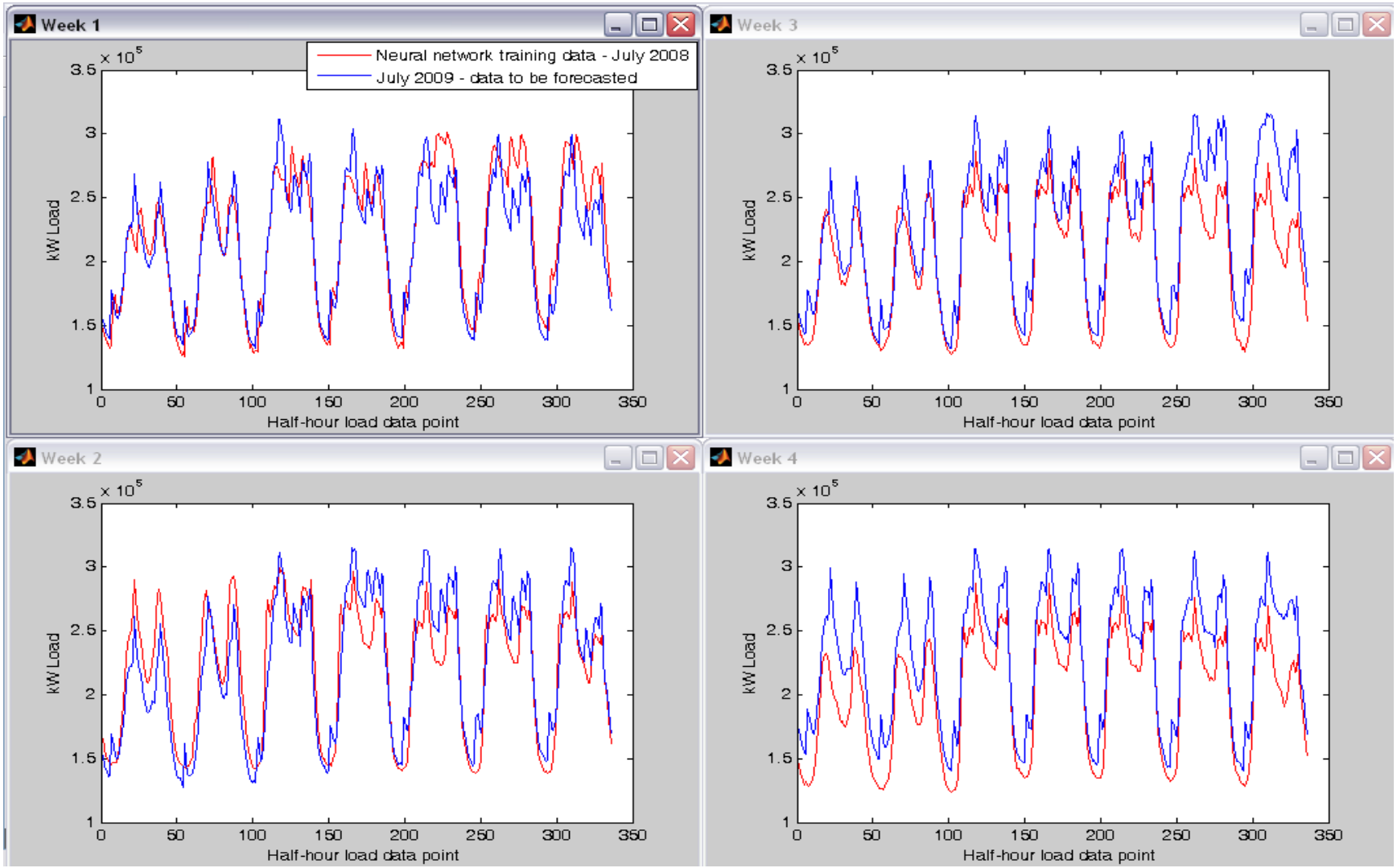


Figure 4.9: Plotting the actual load used in four weeks from the 5th of July 2008 to 1st of August 2008 (red plot) and the actual load data to be forecasted for the four weeks from the 4th of July 2009 to 31st of July 2009 (blue plot)

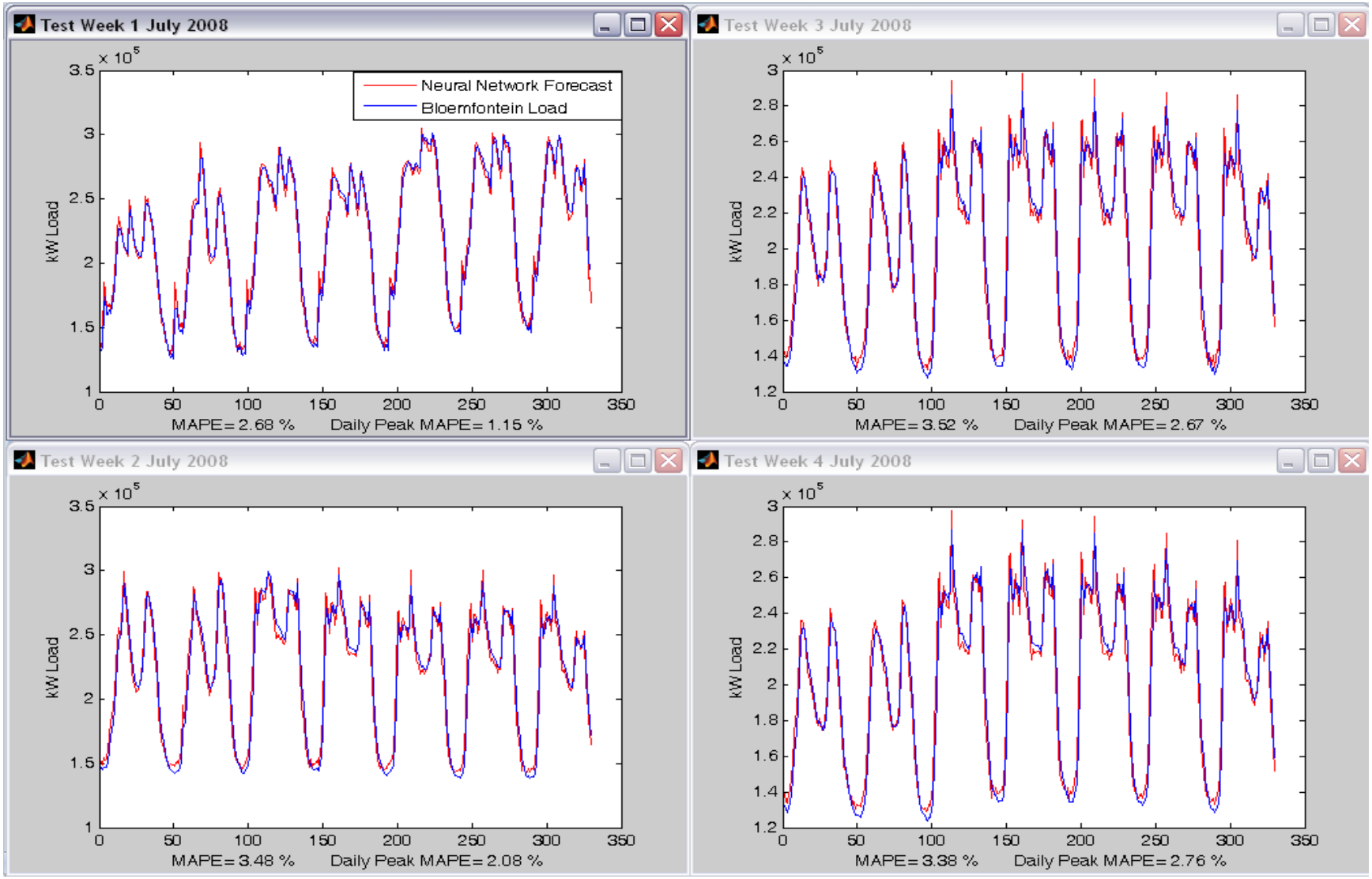


Figure 4.10: Plotting the test results of the neural network for the four weeks from the 5th of July 2008 to 1st of August 2008. The training period is the same four weeks from the 5th of July 2008 to 1st of August 2008

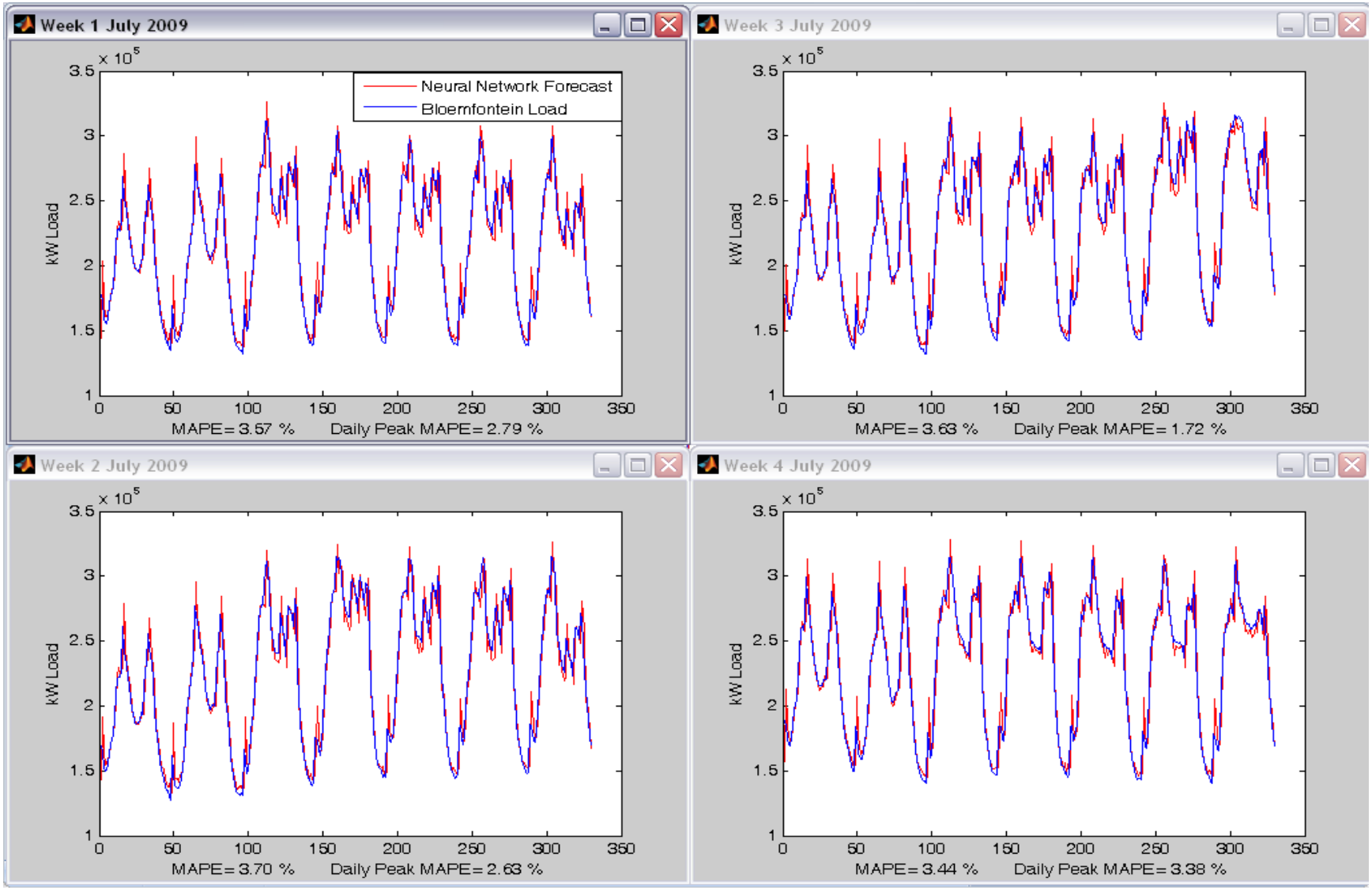


Figure 4.11: Plotting the forecasting results of the neural network for the four weeks from the 4th of July 2009 to 31st of July 2009 (red plot) and the actual load used during the same period (blue plot)

In Figure 4.9, week 1 to week 4 presents the graphical behaviour of the half-hourly load curves of July 2008 and 2009 on a weekly sub-period of the forecasting horizon of one month. Looking at the peak load levels of the four weeks shown in Figure 4.9, especially weeks 2, 3 and 4, it follows evident that the training data (red) contain peak load levels that are significantly lower than the peak load levels needed to be forecasted by the network. So, forecasting of data points that lie outside the training set was required to take place.

Figure 4.10 shows the superimposed plots of the actual load, plotted in blue, used in July 2008 and the same July 2008 load forecasted by the trained neural network, plotted in red, for four consecutive weeks. Note that the forecasted daily peak load levels overshoot the actual daily peak load levels. This is indicated in the daily peak load level MAPE, which varies between 1.15 % and 2.76 %, shown in each sub-graph in Figure 4.10. The goodness of fit of the rest of the forecasted and actual load graphs, superimposed on each other, is clear.

Figure 4.11 presents the graphical behaviour of the half-hourly actual load used in July 2009 (plotted in blue), and the same load forecasted (plotted in red), superimposed. Note that the forecasted daily peak load levels again overshoot the actual daily peak load levels. This is indicated in the daily peak load level MAPE, which varies between 1.72 % and 3.38 %, shown in each sub-graph in Figure 4.11. Again, the goodness of fit of the rest of the forecasted and actual load graphs, superimposed on each other, is acceptable.

The July - week 4 graph, taken from Figure 4.11 as an excerpt, are shown zoomed in, as Figure 4.12. In it, the morning daily peak for the Bloemfontein weekly load demand normally occurs between 11:00 to 12:00 and the evening daily peak electric consumption varies between 19:00 and 21:00. This "*twin peak"* shaped load levels, in essence, symbolises the main daily peaks, namely: morning and evening peaks, typical also in a country like Portugal [45, p.8] and Canada [25, p.529].

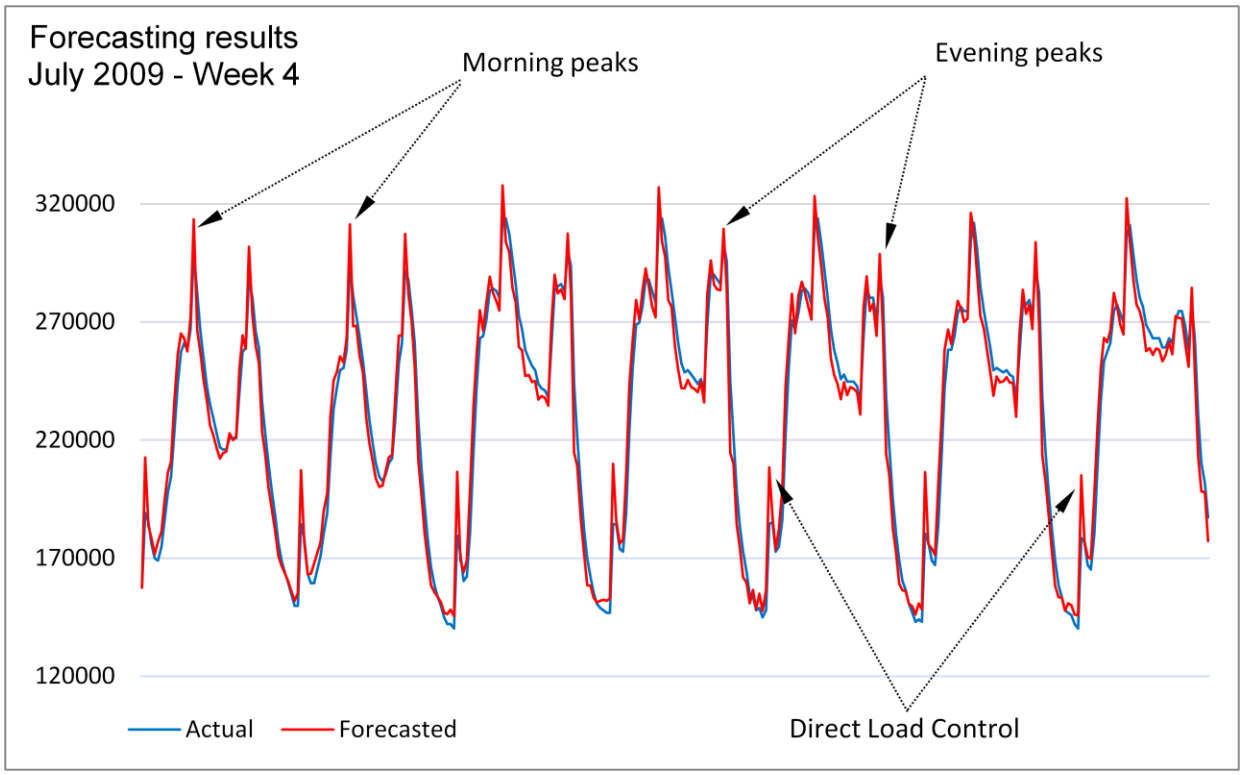


Figure 4.12: Showing the different peaks during a weeks load consumption, using Week 4, July 2009, as an example

The *"twin peaks"* confirm the non-linear features of the load and often affects the forecasting accuracy negatively due to very sharp transitions in the load curve shape.

Visually, observing the DLC- and daily peak load levels from Figure 4.10 and 4.11, it follows that the neural network seems to predict the bulk of the load data accurately, but also seems to over-predict these peak load levels as shown in Figure 4.12. However, these over-predicted DLC peaks shown in Figure 4.12 are not significant for forecasting, as they are not driven by consumer demand.

Considering the fact that neural networks can only "model" data given to it within its training limits, these peak load level errors can be expected during performance measurements [51, p.109], [49, p.249].

4.2.1.1 Comparing the training and the validation sets using MAPE results

Comparing the performance results for the same network in Table 4.1, the difference in the MAPE performance of the test set is less than 1%. The MAPE accuracy of prediction on the validation data also varies less than 1%. This indicates a reasonable forecasting consistency. The small discrepancy between the forecasting results indicates that overfitting is a minimum [13, p.731].

Table 4.1: The MAPE values obtained from Figures 4.10 and 4.11, showing the trained and forecasted MAPE results for the months in July 2008 and July 2009

	July	Week 1	Week 2	Week 3	Week 4
Test set	**2008**	**2.68%**	**3.48%**	**3.52%**	**3.38%**
Validation set	**2009**	**3.57%**	**3.70%**	**3.63%**	**3.44%**

All the MAPE results in Table 4.1, in terms of forecasted consumption, are encouraging, with a mean absolute error of less than 4%, measured as repeatable for each week of the same set.

4.2.1.2 Comparing the training and the validation sets using correlation coefficient results from the weekly scatter plots

The scatter plot is helpful in showing that certain data points have poor fits. Figure 4.13 shows the four graphs of week 1, 2, 3 and 4 of July 2009 grouped together for comparison. The scatter plots of the network outputs versus the targets (plotted as open circles) are superimposed on the best linear fit (blue line) and the perfect fit (dashed line). The Target (x-axis) and Output (y-axis) represents the actual and predicted kW Load.

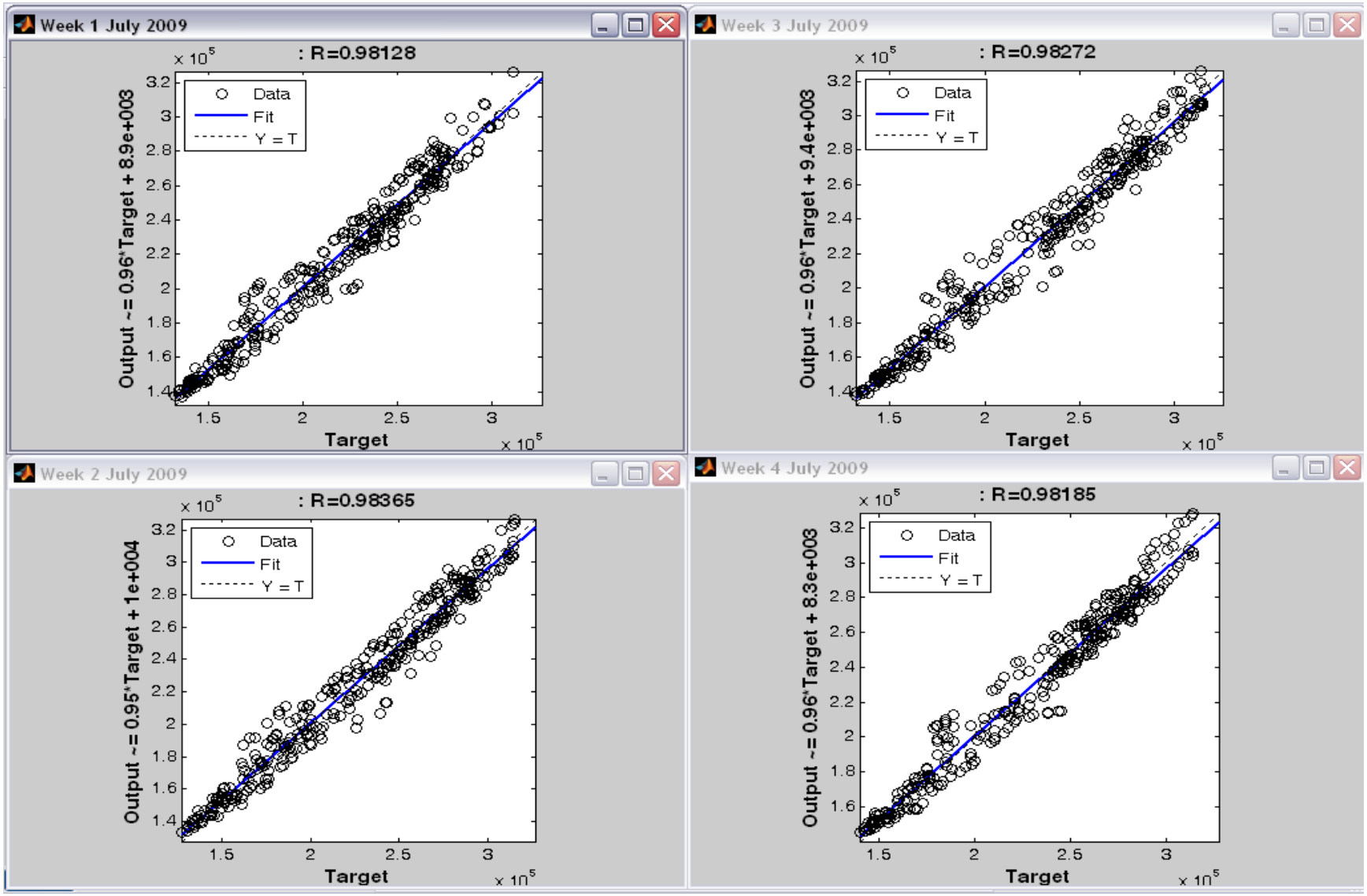


Figure 4.13: Plotting of the Target values (actual kW load used) versus the Output values (neural network forecast) for weeks 1, 2, 3 and 4 in July 2009

The **R** values (correlation coefficients) taken from Figure 4.13 are shown grouped in Table 4.2.

Table 4.2: The correlation coefficients taken from Figure 4.13, rounded to two decimal places

Week in July 2009	Correlation coefficient (R)
Week 1: 04 July 2009 12:30 AM to 10 July 2009 24:00 PM	**0.98**
Week 2: 11 July 2009 12:30 AM to 17 July 2009 24:00 PM	**0.98**
Week 3: 18 July 2009 12:30 AM to 24 July 2009 24:00 PM	**0.98**
Week 4: 25 July 2009 12:30 AM to 31 July 2009 24:00 PM	**0.98**

For a perfect fit, the data should fall along a 45° line, where the network outputs are equal to the targets. In Figure 4.13, week 1 to week 4, it is difficult to distinguish the best linear fit line from the perfect fit line, which indicates that the trained network has a good performance. Therefore, the forecasting performance of the neural network is satisfactory, since the dispersion around the 45° line is limited.

The neural network outputs track the targets very well for the validation set of four weeks and the R-value is over **0.97 for the total response.** For this case study, the validation data indicates a good fit.

In conclusion, a visual inspection and the high R values both confirm an overall acceptable linear relationship in the four graphs.

4.3 Overall forecasting model performance

The forecasting results over a period of the three years, 2007, 2008 and 2009, for the three months of May, June and July amount to nine data sets (four weeks per month was chosen as a set) which was analysed using the weekly calculated MAPE values and the linear regression statistics as used in the case study.

The two techniques used for the purpose of this analysis were to:

- Measure, using MAPE, each trained network's performance, i.e. the forecasting accuracy that is limited by prediction errors. The prediction error should be repeatable from week to week.

- Use linear regression to compare the correlation coefficients (**R**) for week 1 to week 4, obtained from the scatter plots for each month of May, June and July 2007 to 2009.

4.3.1 The MAPE results

Each of the nine neural networks used was trained with four full weeks of a specific month of the previous year. The trained network was then used to predict each of four weeks of the same month of the next year.

The waveform of each week of the same month contains different harmonic components so the forecasting performance of each week cannot be compared with the other three weeks, it can only be compared by itself (actual versus predicted).

The overall forecasting performance in the forecasting set is measured according to the aim in section 3.2 "Minimising the error during training by evaluating and adjusting the topology elements of the neural network to keep the MAPE below 5 % during forecasting".

The MAPE measures the accuracy of the fitted time series values using the same number of samples regardless of the model, so one can compare MAPE values across models and therefore compare the accuracy of different models. Smaller values indicate better fitting models.

Tables 4.3, 4.4 and 4.5 show the results for May, June and July 2007 to 2009.

Table 4.3: The MAPE values obtained when forecasting the months in May 2007, 2008 and 2009

May	Week 1	Week 2	Week 3	Week 4
2007	3.41%	3.46%	3.50%	3.40%
2008	3.77%	3.93%	3.84%	3.67%
2009	4.24%	4.21%	3.85%	3.78%

Table 4.4: The MAPE values obtained when forecasting the months in June 2007, 2008 and 2009

June	Week 1	Week 2	Week 3	Week 4
2007	3.12%	2.95%	2.96%	3.07%
2008	3.55%	3.38%	3.21%	3.36%
2009	2.71%	2.87%	2.94%	3.12%

Table 4.5: The MAPE values obtained when forecasting the months in July 2007, 2008 and 2009

July	Week 1	Week 2	Week 3	Week 4
2007	2.89%	2.88%	2.83%	2.96%
2008	3.37%	3.53%	3.59%	3.56%
2009	3.57%	3.70%	3.63%	3.44%

The MAPE results obtained from Tables 4.3 to 4.5 show that the forecasting error is repeatable from each month in May, June and July 2007 to 2009. The trial to trial accuracy is limited to a MAPE of below 5% by systematic (repeatable) errors. This meets the criterion set out in Chapter 3.

4.3.2 The correlation coefficient results

The regression correlation coefficients are shown below. Tables 4.6, 4.7 and 4.8 show the results for May, June and July 2007 to 2009.

Table 4.6: Correlation Coefficient (R) results for May 2007, 2008 and 2009 rounded to two decimal places

May	Week 1	Week 2	Week 3	Week 4
2007	0.98	0.98	0.99	0.98
2008	0.98	0.97	0.98	0.98
2009	0.98	0.98	0.98	0.98

Table 4.7: Correlation Coefficient (R) results for June 2007, 2008 and 2009 rounded to two decimal places

June	Week 1	Week 2	Week 3	Week 4
2007	0.99	0.99	0.99	0.99
2008	0.98	0.98	0.99	0.98
2009	0.99	0.99	0.99	0.99

Table 4.8: Correlation Coefficient (R) results for July 2007, 2008 and 2009 rounded to two decimal places

July	Week 1	Week 2	Week 3	Week 4
2007	0.99	0.99	0.99	0.99
2008	0.99	0.98	0.98	0.98
2009	0.98	0.98	0.98	0.98

For this investigation, the fit is reasonably good for all the data sets, with consistent **R** values of 0.97 or above as shown in Table 4.6, 4.7 and 4.8.

CHAPTER 5

CONCLUSION

As the load forecasting literature evolved over the years, some interest was shown in time-series forecasting using neural network modelling. Since neural networks have not been developed for handling a recurrent pattern of data inputs, either the input has to be pre-processed or the model has to be adapted to temporal tasks. Pre-processing is the easier of the two strategies because it turns a sequence of time-series elements into a single input. This can be achieved by sliding a so-called "time window" over the load data sequence.

Huge amounts of data are needed for large neural network forecasting. It is more efficient to develop a compact model, with fewer degrees of freedom, that requires less training data and still obtain reliable results. The hidden layers should be small enough to allow generalisation, and large enough to produce the required mapping.

The initial and final neural networks were developed using the electric power system data, obtained from ESKOM, Bloemfontein. The load data from the substation was directly accessed after preliminary screening for anomalous or missing data. It was not split up into residential, commercial and industrial components. The data used was collected over four years: 1 January 2006 to the 30th of December 2009. From this data, the winter months of May, June and July for the years 2007 to 2009 were used to train and develop the final neural network and evaluate its forecasting potential.

5.1 General assessment

The contributions of this thesis are discussed based on the objectives set out in Section 3.2 on p.51. The main objective was to build, train, test and validate a short term, dynamic, half-hour ahead updating, time series forecaster, with data

for each of the late autumn and winter months of May, June and July 2007 to 2009, using a neural network model.

First, the model finding process consisted of three phases:

- model selection,
- parameter estimation, and
- performance testing.

From the case study the next half-hour load data points were predicted for the four weeks of the month of July 2009, using the 6:1:1 cascade forward neural network model. All of the four MAPE values were below 4% for the four weeks of July 2009 and this was below the objective of a MAPE value of 5%.

From Fig 4.11, the daily peak MAPE values vary between 1.72% and 3.38%. Visually it can also be seen that the forecasting model was slightly over-predicting the daily peak load levels during each week of a month. The monitoring of the daily peak load levels is critical (as penalty charges are payable for exceeding the NMD payable by consumers).

Although the high weekly R values appear satisfactory and consistent in Table 4.2, generalisation was not complete, given that there is some dispersion around the 45° line as seen in the scatter plots in Fig.4.13.

***Finally, the neural network's functional reliability over a period of three consecutive months was estimated experimentally*.** The STLF network's performance was done by comparing weekly MAPE and **R** values (correlation coefficients), taken from the regression plots, to analyse the forecasting accuracy for each month of May, June and July in 2007, 2008 and 2009.

From the weekly MAPE results in Tables 4.3, 4.4 and 4.5 in Chapter 4 it can be seen that the prediction error calculated for week 1 to week 4 in each of the three

months was repeated consistently and below 5% during forecasting. It indicates that a good performance level was maintained throughout.

The high values of the correlation coefficient results in Tables 4.6, 4.7 and 4.8 confirm a strong linear relationship between the actual and forecasted load curves. In this case, the network's overall response was satisfactory, and it can be tested on new monthly input/output pairs of data sets.

Considering the evaluation of the final 6:1:1 cascade forward artificial neural network model developed during the course of the project, a number of definitive conclusions regarding its performance can be made:

- The practical load data from Bloemfontein City in the Free State in South Africa was used to illustrate the proposed method, and the results indicate that the proposed method can obtain an acceptable accuracy that is effective for forecasting the short term load of this power system.

- Future implementation of this type of forecasting model can prevent possible penalty charges for exceeding the NMD payable by Bloemfontein City Municipality.

- This neural network represents a simple alternative to modelling short term electricity load since it is easy to compute, significantly reduces the number of variables to be considered, and generally contributes to greater accuracy of electric load forecasts.

- This approach resulted in an economical forecasting model that not only has an acceptable short input data sample forecasting performance, but is easily constructed and applicable for day-to-day load forecasts for other territories with a similar load profile to Bloemfontein City.

In conclusion, this investigation led to an approach suitable for constructing a single neural network model that has the advantage of circumventing the problem of forecasting weekends, special holidays, day of the week and off-peak/peak

models separately. The input vector and number of neurons in the hidden layer was kept to a minimum to avoid model over-parameterization. The performance of the final STLF neural network has met the specifications of the aim and objectives set out in Section 3.2 in Chapter 3.

It is the view of the author that the progress in load forecasting at this institution can move forward in the following direction:

1. For future research the performance of the developed model can be validated with the latest annual data to be obtained from Eskom, Bloemfontein and other regions in South Africa.

2. Acquire a better understanding of short term electric load forecasting dynamics and its statistical properties to investigate other appropriate ANN models using time series prediction.

3. The developed model can be used as a part of a postgraduate course for further research into the field of medium- and long term load forecasting using time series prediction with artificial neural networks.

REFERENCES

1. **Aleksander, I. and Morton, H.** An Introduction to Neural Computing. 2nd. UK: ITCP, 1995.

2. **Alfares, H.K. and Nazeeruddin, M.** Electric load forecasting: literature survey and classification of methods. International Journal of Systems Science 33, no. 1 (2002): 23-34.

3. **Almeshaiei, E. and Soltan, H**. A methodology for Electric Power Load Forecasting, Alexandria Engineering Journal, In Press, Corrected Proof, Available online 27 July 2011, ISSN 1110-0168, DOI: 10.1016/j.aej.2011.01.015. http://www.sciencedirect.com/science/article/pii/S1110016811000330 (accessed August 2011)

4. **Berk, K.N. and Carey, P.** Data Analysis with Microsoft Excel. Toronto: Brooks/Cole, 2004.

5. **Cichocki, A. and Unbehauen, R.** Neural Networks for Optimization and Signal processing. UK: John Wiley & Sons, Inc., 1996.

6. **Crone, S.F.** EVIC'05 Slides - Forecasting with Neural Networks Tutorial. 15 December 2005. (accessed July 2008).

7. **Davalo, E.** and Naim, P. Neural Networks. UK: MacMillan Education, Limited., 1991.

8. **Demuth, H. and Beale, M.** Neural Network Toolbox Guide: For Use with MATLAB. The MathWorks, Inc, Massachusetts. 2002.

9. **Djukanovic, M., Ruzic, S., Babic, B., Sobajic, D.J. and Pao, Y-H**. A neural-net based short term load forecasting using a moving window procedure, International Journal of Electrical Power & Energy Systems, Volume 17, Issue 6, December 1995, Pages 391-397, ISSN 0142-0615, DOI: 10.1016/0142-0615(94)00009-3.

http://www.sciencedirect.com/science/article/pii/0142061594000093

10. **du Plessis, L.** System Optimisation and the Impact of the Short Term Load Forecast. http://www.eepublishers.co.za/article/system-optimisation-and-impact-of-short-term-load-forecast.html. (accessed 14/6/ 2011).

11. **Fahlman, S.E. and Lebiere, C.** "The Cascade-Correlation Learning Architecture (CMU-CS-90-100)." Pittsburgh, 1991.

12. **Feinberg, E.A. and Genethliou, D.** LOAD FORECASTING. Chap. 12 in *APPLIED MATHEMATICS FOR POWER SYSTEMS*. 2005. http://www.ams.sunysb.edu/~feinberg/public/lf.pdf (accessed 21/11/2010).

13. **Fidalgo, J.N. and Matos, M.A.** Forecasting Portugal Global load with Artificial Neural Networks. Vol. 2, in Artificial neural networks - ICANN 2007: 17th International Conference, edited by J. Marques de Sá, 731. Berlin: Springer-Verlag, 2007.

14. **Foresee, F.D. and Hagan, M.T.** Gauss-newton approximation to Bayesian regulasization. Proceedings of the 1997 International Joint Conference on Neural Networks . 1997. 1930-1935.

15. **Galushkin, A.I.** Neural Networks Theory. Berlin: Springer, 2007.

16. **Ghiassi,M., Zimbra, D.K. and Saidane, H**. Medium term system load forecasting with a dynamic artificial neural network model. *Electric Power Systems Research*, *Volume 76, Issue 5*, *March 2006*, *Pages 302-316.*
    a. ISSN 0378-7796, DOI: 10.1016/j.epsr.2005.06.010.
    http://www.sciencedirect.com/science/article/pii/S0378779605001951 (accessed July 2011).

17. **Gurney, K.** Neural Nets by Kevin Gurney. 2004. http://www.shef.ac.uk/psychology/gurney/notes (accessed June 2010).

18. **Hagan, M.T., Demuth, H.B. and Beale, M.H.** Neural Network Design. Boston: PWS Publishing Company, 1996.

19. **Hahn, H., Meyer-Nieberg, S., and Pickl, S**. Electric load forecasting methods: Tools for decision making, European Journal of Operational Research, Volume 199, Issue 3, 16 December 2009, Pages 902-907, ISSN 0377-2217, DOI: 10.1016/j.ejor.2009.01.062. http://www.sciencedirect.com/science/article/pii/S0377221709002094. (accessed May 2010).

20. **Hammerstrom, D.** Working with neural networks. *IEEE Spectrum*, 1993: 46-53.

21. **Haykin, S.** Neural Networks: A Comprehensive Foundation. Upper Saddle river, New Jersey: Prentice Hall, Inc, 1999.

22. **Heydt, G.T.** *Computer* Analysis Methods for Power Systems. New York: Macmillan Publishing Company, 1986.

23. **Hunt, K.J., Irwin, G.R., Warwick, K.** Neural Network Engineering in Dynamic Control Systems: Advances in Industrial Control. London: Springer-Verlag London Limited, 1995.

24. **Jones, M.T.** Artificial Intelligence: A Systems Approach. Sudbury, MA: Jones & Barlett Publishers, 2009.

25. **Kandil, N., Wamkeue, R., Saad, M. and Georges, S.** An efficient approach for short term load forecasting using artificial neural networks, International Journal of Electrical Power & Energy Systems, Volume 28, Issue 8, October 2006, Pages 525-530, ISSN 0142-0615, DOI: 10.1016/j.ijepes.2006.02.014. (http://www.sciencedirect.com/science/article/pii/S0142061506000676) (accessed May 2011).

26. **Kung, S.Y.** Digital Neural Networks. USA: PTR Prentice-Hall, Inc., 1993.

27. **Lai, L.** Intelligent System Applications in Power Engineering:evolutionary programming and neural networks. UK: Wiley & Sons Ltd, 1998.

28. **MacKay, D.J.C.** Bayesian Interpolation. *Neural Computation* 4, no. 3 (1992): 415-447.

29. **Madsen, K., Nielsen, H.B. and Tingleff, O.** Methods for Non-linear Least Squares Problems. 2nd Edition. Informatics and Mathematical Modelling, Technical University of Denmark, DTU. April 2004. www2.imm.dtu.dk/pubdb/views/edoc_download.../imm3215.pdf. (accessed December 2010).

30. **Mandal, P., Senjyu, T., and Urasaki, N.** Toshihisa Funabashi, A neural network based several-hour-ahead electric load forecasting using similar days approach, International Journal of Electrical Power & Energy Systems, Volume 28, Issue 6, July 2006, Pages 367-373, ISSN 0142-0615, DOI: 10.1016/j.ijepes.2005.12.007. http://www.sciencedirect.com/science/article/pii/S0142061506000275 (accessed March 2011).

31. **Middleton, M.R.** Data Analysis Using Microsoft Excel. 3. Toronto: Brooks/Cole, 2004.

32. **Montgomery, D.C., Jennings, C.L. and Kulahci, M.** Introduction to Time Series Analysis and Forecasting. Hoboken: John Wiley & Sons,Inc, 2008.

33. **Morantz, B.H., Whalen, T.G., Zhang, P.** A Weighted Window Approach to Neural Network Time Series Forecasting. Georgia State University, 2003. 3-8.

34. **Murto, P.** Neural Network Models for Short Term Load Forecasting. Helsinki University Of Technology, Helsinki, Finland, 1998.

35. **Palmer-Brown, D.,Draganova, C., Pimenidis, E. and Mouratidis, H.** Engineering Applications of Neural Networks. *11th International Conference, EANN.* London: Springer, 2009. 465.

36. **Plummer, E.A.** Time Series Forecasting with Feed-Forward Neural Networks:Guidelines and Limitations. University of Wyoming, Laramie, Wyoming, USA. 2000.

37. **Priddy, K. L. and Keller, P. E.** Artificial neural networks: an introduction. Bellingham, Washington, USA: SPIE-The International Society for Optical Engineering, 2005.

38. **Principe, J.C., Euliano, N.R. and Levebvre, W.C.** Neural And Adaptive Systems: Fundamentals through Simulations . New York: John Wiley & Sons, Inc, 2000.

39. **Rabunal, J.R. and Dorado, J.** Artificial Neural Networks in Real-Life Applications. Hershey: Idea Group Publishing, 2006.

40. **Reusch, B.** Computational Intelligence: Theory and Applications. Edited by B. Reusch. Dortmund, Germany: Springer, 1999.

41. **Rui, Y. and Keib, A.A**. A Review of ANN-based Short-term Load Forecasting Models. http://research.microsoft.com/en-us/um/people/yongrui/ps/review95.pdf (accessed March 2011).

42. **Rumelhart, D.E. and McClelland, J.L**. Parallel Distributed processing: Explorations in the Microstructure of Cognition. Cambridge, MA. MIT Press. 1986.

43. **Russel, I.** Learning in a Neural Network. 2004. http://uhaweb.hartford.edu/compsci/neural-networks-Learning.html (accessed July 2011).

44. **Santos, P, J, Martins, A. G. and Pires., A.J.** Designing the input vector to ANN-based models for short term load forecast in electrical distribution sytems. 2004. http://www.inescc.pt/documentos/14_2004.pdf (accessed June 2011)

45. **Santos, P.J., Martins, A.G., Pires, A.J., Martins, J. F. and Mendes, R. V.** Short-Term Load Forecast Using Trend Information and Process Reconstruction. www.inescc.pt/documentos/14_2004.pdf

(accessed May 2009).

46. **Sargunaraj, S., Sen Gupta, D.P. and Devi, S.** Short Term Load Forecasting for Demand Side Management. *IEEE Xplore*, no. Release 2.2 (1997): 1-8.

47. **Sarle, W.S.** Neural Network FAQ. 2004. ftp.sas.com/pub/neural/FAQ.html.

48. **Shan, S.** A Levenberg-Marquardt Method For Large-Scale Bound-Constrained Nonlinear Least-Squares. The University of British Columbia, Computer Science. Vancouver. Canada (2008): 1-8.

49. **Slade, P. and Gedeon, T.D.** Bimodal Distribution Removal. In New Trends in Neural Computation: International Workshop on Artificial Neural Networks, IWANN ,93 Sitges Spain June 1993, edited by Joan Cabestany, Alberto Prieto José Mira, 249. Berlin: Springer-Verlag, 1993.

50. **Sumathi, S. and Paneerselvam, S.** Computational Intelligence Paradigms: Theory & Applications Using MATLAB . Boca Raton, Florida: CRC Press, Taylor and Francis Group, 2010.

51. **Swingler, K.** Applying Neural Networks: A Practical Guide. San Francisco: ACADEMIC PRESS, 1996.

52. **Teng, C.C.** Mixed-Mode Supervised Learning Algorithms for Multilayered Feed-Forward Neural Networks. University of Illinois, Urbana-Champaign, Illinois, 1993.

53. **Yalcinoz, T. and Eminoglu, U.** Short term and medium term power distribution load forecasting by neural networks**,** Energy Conversion and Management, Volume 46, Issues 9-10, June 2005, Pages 1393-1405, ISSN 0196-8904, DOI: 10.1016/j.enconman.2004.07.005. http://www.sciencedirect.com/science/article/pii/S019689040400192X (accessed January 2011).

54. **Yang, J. and Chen, C.** Dominant Neuron Techniques. Vol. 2, in *Optimization Techniques*, edited by C.T. Leondes, 249. San Diego: Academic press, 1998.

55. **Zaknich, A.** Neural Networks for Intelligent Signal Processing. Singapore: World Scientific Publishing Co. Pte. Ltd, 2003.

56. **Zalzala, A.M.S. and Morris, A.S.** Neural Networks for Robotic Control: Theory and Applications. UK: Ellis Horwood, 1996.

57. **Zhang, G., Patuwo, B.E. and Hu, M.Y.** Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting Volume 14, Issue 1, 1 March 1998, Pages 35-62, ISSN 0169-2070, DOI: 10.1016/S0169-2070(97)00044-7.
http://www.sciencedirect.com/science/article/pii/S0169207097000447
(accessed August 2008).