11-1-2008

Development of models for short-term load forecasting using artificial neural networks

Simaneka Amakali
Cape Peninsula University of Technology

DEVELOPMENT OF MODELS FOR SHORT-TERM LOAD FORECASTING USING ARTIFICIAL NEURAL NETWORKS

by

Simaneka Amakali

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: **Discipline Electrical Engineering**

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Prof. R. Tzoneva
Co-supervisor: C. Kriger

Bellville Campus
November 2008

DECLARATION

I, Simaneka Amakali, declare that the contents of this thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed **Date**

ABSTRACT

Optimal daily operation of electric power generating plants is very essential for any power utility organization to reduce input costs and possibly the prices of electricity in general. For a fossil fuel – fired power plant for example, the benefits of power generation optimalization *(i.e. generate what is reasonably required)* extends even to environmental issues such as the subsequent reduction in air pollution. Now to generate "*what is reasonably required"* one needs forecast the future electricity demands. Because power generation relies heavily on the electricity demand, the consumers are also practically speaking required to wisely manage their loads to consolidate the power utility's optimal power generation efforts. Thus, for both cases, accurate and reliable electric load forecasting systems are absolutely required.

To date, there are numerous forecasting methods developed primarily for electric load forecasting. Some of these forecasting techniques are conventional and often less favoured.

To get a broad picture of the problem at hand, a literature survey was first conducted to identify possible drawbacks of the existing forecasting techniques including the conventional one. Artificial neural networks (ANNs) approach for short-term load forecasting (STLF) has been recently proposed by a majority of researchers. But there still is a need to find optimal neural network structures or convenient training approach that would possibly improve the forecasting accuracy.

This thesis developed models for STLF using ANNs approach. The evolved models are intended to be a basis for real forecasting application. These models are tested using actual load data of the Cape Peninsula University of Technology (CPUT) Bellville campus reticulation network and weather data to predict the load of the campus for one week in advance.

The models were divided into two classes: first, forecasting the load for a whole week at once was evaluated, and then hourly models were studied. In both cases, the inclusion of weather data was considered. The test results showed that the hour-by-hour approach is more suitable and efficient for a forecasting application. The work suggests that incremental training approach of a neural network model should be implemented for on-line testing application to acquire a universal final view on its applicability.

Keywords *– power system operations, load forecasting, artificial neural networks, training mode, accuracy*

ACKNOWLEDGEMENTS

To the Almighty Lord, for granting me the wisdom, ability, perseverance, and clarity to do this work.

The research work is partially funded by the TESP Project "*Advanced control of power systems*" and NRF project "*Substation Automation and Energy Management System*" *No. ICD2006061900021*. The financial support received from the CPUT is also appreciated.

First of all I would like to show my special thanks to my supervisors Prof. R. Tzoneva and Mr. C. Kriger for all their guidance and support. They supervised the whole process of my research, and have provided many technical suggestions. I benefited a lot from their knowledge and experience on programming and control systems.

Many thanks to the CENORED management, particularly Mr. Mburumba Appolous, for giving me the chance to study in South Africa.

I wish also to thank Mr. L. Lombard and the entire CPUT maintenance team, for their continued support and cooperation during data gathering process.

Many thanks to the South African Weather Service department in Cape Town for providing the weather data used in this work.

I wish to thank every individual at the Cape Peninsula University of Technology, especially fellow colleagues in the electrical department. Without their help and encouragement during the one year in South Africa I wouldn't have finished the Master research work.

I would also like to thank my parents and my brothers Chris, Ben, and Erastus for their support, patience, and understanding of my research in Cape Town.

Many thanks to my ever encouraging lovely fiancée, Sarah, for being patient, and supportive throughout the study period.

Cape Town, November 2008
Simaneka Amakali

DEDICATION

To my father ***Aron Jonas*** *and my mother* ***Hilen Namukwambi***

TABLE OF CONTENTS

CHAPTER 2: LITERATURE REVIEW

CHAPTER 3: POWER SYSTEM LOAD FORECASTING

CHAPTER 4: ARTIFICIAL NEURAL NETWORKS (ANNs)

CHAPTER 5: DATA COLLECTION AND PRE-PROCESSING

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

MATLAB:	A tool for doing numerical computations using existing functions and user defined programs to analyze data and visualize the output.
Simulink:	A platform for multi-domain simulation and model-based design of dynamic systems.
Genetic Algorithm:	An adaptive heuristic and global optimization technique based on the mechanics of natural selection and genetics as observed in the *biological evolution*
Artificial Neural Network:	A biological inspired network which is capable of learning a pattern from a set of training data and then able to generate the same outputs as with the learning data when presented with a new set of data which contains a similar pattern to that which the ANN was trained with.
Demand:	A rate at which electric energy is delivered to or by a system, part of a system or a piece of equipment. It is expressed usually in kilowatts at a given instant or averaged over any designated period of time (30 minutes). The primary source of "demand" is the power-consuming equipment of customers.
Distribution:	An act or process of delivering electric energy from convenient points on the transmission system (usually a substation) to consumers.
Load Profile or Curve:	A curve on a chart showing power (watts) supplied, plotted against time of occurrence, and illustrating the varying magnitude of the load during the period covered.
Demand-Side Management:	A broad term for electric load management applying to utility actions, programs, and designs for the purpose of lowering system peaks and reducing energy consumption by the consumer as well as

	the utility system. There are direct actions and controls by the utility that result in immediate results to lower peaks and reduce energy requirements. There are also indirect (passive) programs requiring the cooperation and participation by the consumer to achieve similar results.
Load Management:	An economic reduction of electric energy demand during a utility's peak generating periods.
Reliability:	A guarantee of a system performance at all times and under all reasonable conditions to assure constancy, quality, adequacy, and economy of electricity. It is also the assurance of the continuous supply of electricity for customers at the proper voltage and frequency.
Substation:	An assemblage of equipment for the purposes of switching and/or changing or regulating the voltage of electricity. Service equipment, line transformer installations or minor distribution and transmission equipment are not classified as substations.
Peak:	A greatest load on an electric system during any prescribed demand interval for a given period. The summer (or cooling) season peak is normally smaller than winter season peak.
Forecasting:	A process of estimation in unknown situations.
Perceptron:	A type of artificial neural network invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt.
Stochastic model:	A time evolution of the model that contains random elements
Eskom:	The largest South African power utility. City of Cape Town (CoCT) purchases electric energy from Eskom.
Time-of-use electricity tariff:	A tariff with three periods when energy is charged at

different rates. The least expensive rate is called the off-peak period and coincides with the time period when the system load is below a certain threshold. The most expensive rate is called the peak-period and corresponds to a period when the system load is above a certain threshold. When the load is in between the off-peak and peak period the time period is called the standard period. The rate for the standard period is higher than the off-peak energy rates but lower than the peak period energy rates.

Structured Query Language (SQL): An American National Standards Institute (ANSI) standard computer language for accessing and manipulating database systems. SQL statements are used to retrieve and update data in a dataset.

Non-essential load: refers to an electric load that can conceivably be shed for a limited period at a time without negatively impacting on the operation of the organisation as a whole.

Compact fluorescent lamps (CFLs): are lamps which offer consumers lighting that have a longer life and consume considerably less energy than conventional incandescent globes.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
AR	Auto Regression
ARMA	Auto Regression Moving Average
BL	Boltzmann Learning
BP	Back Propagation
CG	Conjugate Gradient
CL	Competitive Learning
CoCT	City of Cape Town
CPUT	Cape Peninsula University of Technology
CVP	Cross Validation Pruning
ECL	Error Correlation Learning
ECN	Engineering Council of Namibia
EMS	Energy Management System
ER	Elman Recurrent
ES	Expert System
FF	Feedforward
GA	Genetic Algorithm
HL	Hebbian Learning
ICP	Information Criterion Pruning
IPP	Independent Power Producer
IV	Input Vector
LM	Lavenberg-Marquardt
LPU	Larger Power User
MAPE	Moving Average Percentage Error

MLE	Maximum Likelihood Estimation
MLP	Multi Layer Perceptron
M-LTLF	Medium-Long Term Load Forecasting
MPE	Mean Percentage Error
MSE	Mean Squared Error
NIEIR	National Institute of Economic and Industry Research
NMD	Notified Maximum Demand
pdf	Probability Density Function
QN	Quasi-Newton
RBF	Radial Basis Function
RL	Reinforced Learning
RSBP	Rough Set Back Propagation
SSE	Sum of Squared Error
STLF	Short-Term Load Forecasting
TOU	Time-of-use electricity tariff

LIST OF SYMBOLS

$\hat{L}$	Forecast load value
L	Actual load value
u_j	Output of the j^{th} hidden neuron
L_n	Normal or standard load value
w_{ij}	Weight factor between the input layer and the hidden layer
w_j	Weighted factor between the output layer and the hidden layer
x	Model input
ε	Error term
η	Learning rate
β_i	Regression parameter
m	Input variable index
$o(n)$	Output of the output layer
$L(t)$	Load value at time t
$L(t-1)$	Load value of the previous hour
$L(t-2)$	Load value at current hour minus 2 hours
$L(t-168)$	Load value of the previous week at the same hour
$L(t-169)$	Load value before the homologous load of the previous week
$d(t)$	Day indicator
$C(t)$	Cloud rate at time t
$H(t)$	Humidity at time t
$W(t)$	Wind speed at time t

N	Number of observations or data points
L_w	Weather sensitive part of the load
L_s	Special event component that creates a substantial deviation from the usual load
a_i	Estimated slowly varying coefficients
y_i	Independent influencing factors such as weather effect
$S(t)$	Seasonal term
$R(t)$	Irregular or random component
L_s	Slope between the load at time t and load of the previous hour
B	Backshift operator
ρ	Basis function
ψ	Momentum factor
σ_1	A tan sigmoid transfer function in the hidden layer neurons
σ_2	A pure linear transfer function in the output layer
K	A coefficient of the pure linear transfer function
r	Number of hidden layer neurons
z_i	The weighted sums of the i^{th} hidden layer neuron

CHAPTER ONE

INTRODUCTION

This chapter encapsulates a brief overview of power system load forecasting, awareness of the problem, problem statement, research aim and objectives, hypothesis, assumptions, delimitations and motivation of the research. It also partially covers research design and sampling methodologies. The last part of this chapter provides the organization and outline of the thesis.

1.1 Summary

With the skyrocketing growth of power system networks and the increase in their complexity, many factors have become influential in electric power generation, demand or load management (Moulin *et al.*, 1999). Load forecasting in one of the critical factors for economic operation of power systems.

Forecasting of future loads is also important for network planning, infrastructure development and so on. However, power system load forecasting is a two dimensional concept: consumer based forecasting and utility based forecasting. Thus the significance of each forecast could be handled disjointedly. Consumer based forecasts are used to provide some guidelines to optimize network planning and investments, better manage risk and reduce operational costs.

In basic operations for a power generation plant, forecasts are needed to assist planners in making strategic decisions with regards to unit commitment, hydro-thermal co-ordination, interchange evaluation, and security assessments and so on. This type of forecast deals with the total power system loads at a given time, and is normally performed by utility companies.

Nonetheless, power system load forecasting can be classified in three categories, namely short-term, medium term and long term forecasting. The periods for these categories are often not explicitly defined in a number of literature papers. Thus different authors use different time horizons to define these categories. But roughly, short-term load forecasting covers hourly to weekly forecasts. These forecasts are often needed for day by day economic operations of power generation plants.

Medium-term load forecasting deals with predictions ranging from weeks to a year. Outage scheduling and maintenance of plants and networks are often roofed in these types of forecasts.

Long term forecasting on the other hand deals with forecasts longer than a year. It is primarily intended for capacity expansion plans, capital investments, and corporate budgeting. These types of forecasts are often complex in nature due to future uncertainties such as political factors, economic situation, per capital growth etc. Planning of new and extensions to existing power system networks for both the utility and consumer require long-term forecasts.

The accuracy of the forecast is a critical feature in power system load forecasting. A poor load forecast misleads planners and often results in wrong and expensive expansion plans. From the consumer forecast view, accurate load forecasting is important for distribution system investments, electric load management strategies. This subject also forms part of load rationing strategies i.e. load shedding, DSM (demand side management) initiatives and so on. Without replicating, short term load forecasting is an essential function in daily operational activities especially for utility companies. A negative error in the forecast could severely affect consumer's production levels, particularly for larger power users. Thus accurate forecasts are required for power system security and its overall reliability.

One of the convincing ways to predict loads that are known to be varying continuously on short-term bases is to rather minimise load sampling points to several minutes or hours. This approach is called very short-term load forecasting and is also discussed in this work.

Undoubtedly, both utility companies and consumers are challenged to accurately predict their respective loads. This challenge has been in existence for decades, thus a variety of load forecasting techniques ranging from classical to intelligent systems have been developed to date, and highlighted in a number of studies. The ultimate distinction of these methods can be drawn on the bases of forecast accuracy.

The most popular techniques used for load forecasting are time series based models, similar-day approach and intelligent system based models. Some of the conventional forecasting methods have major drawbacks especially their inability to map the non-linear characteristic of the load, thus a substitute of classical methods with intelligent system based models is to a great extent essential.

Most forecasting models use statistical techniques or artificial intelligence algorithms such as regression, neural networks, fuzzy logic, and expert systems (Feinberg *et al.*, 2005)

Amongst all other intelligent techniques, the use of ANN in STLF is very predominant. Most recent load forecasting works are based on Artificial Neural Networks, and a majority of these papers presented good estimates. Because ANNs are capable of generalization and learning non-linear relationships between variables, ANN-based approaches are often favoured for STLF problems (Lauret *et al.*, 2007). The other important feature of ANNs is their capability to iteratively adjust the synoptic weights between layers. Conventional methods on the other hand require static complex mathematical equations and but still perform poorly in comparison to intelligent-based approaches.

Another leading load forecasting method is Fuzzy logic. Its application in load forecasting is based on periodical similarity of electric load, where the input variables, output variables and the governing rules are the key points. The scope of this work does not cover this technology despite a brief introduction in following chapter. .

1.2 Scope of work of this research

This research work focuses on a specific area of load forecasting, short-term load forecasting. The forecasts are achieved by using Artificial Neural Network (ANN) based models, i.e. feed-forward and recurrent networks developed in MatLab and Simulink environment. The models are then applied to the actual load data of the Cape Peninsula University of Technology to forecast what is often referred to as "consumer own forecast". The application of the models to factual data is done merely as case study to validate the approach. Figure 1.1 below attempts to clarify the focus of this research.

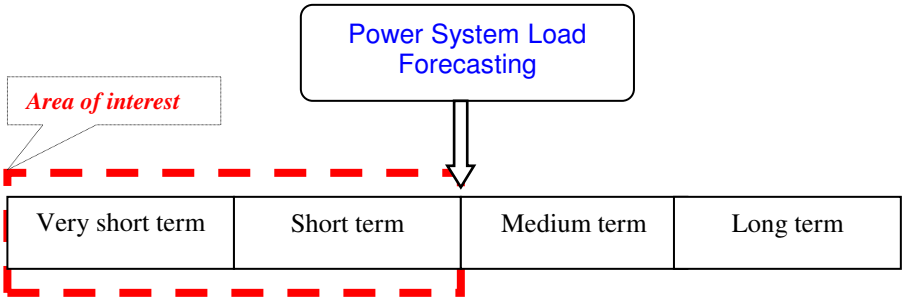


Fig 1.1: Types of load forecasting and focus of the research.

1.3 Awareness of the problem

Short-term load forecasting plays a major role in the real-time control and security functions of an energy management system (Srinivasan, 1998). Laurent (Lauret *et al.*, 2007) concluded that accurate forecasts greatly benefit power system planners to accomplishing a variety of tasks such as economic

scheduling of generating capacity, scheduling of fuel purchases etc. However, load forecasting is a difficult task because the consumption is influenced by many factors, such as weather conditions, vacations, economy status, and idiosyncratic habits of individual customers (Goa *et a*l., 2001).

Inaccurate load forecasts may increase operating costs. (Bunn *et al.,* 1985) reported that a one percent increase in forecasting error of electricity demand resulted in a £10 million increase in operating costs in the British power system. This is purely an error in the utility-type of load forecast. Evidently, a poor load forecast misleads planners and often results in wrong and expensive expansion plans.

Equally, overestimating future electric loads may result in a redundant reserve of electric power. On the contrary, underestimation of loads causes failure in providing sufficient electric power (Pai, 2006).

For a planner to neither underestimate nor overestimate the load, convenient forecasting techniques with reasonable degree of accuracy need to be developed. Although ANN based models entertain some superiority in dynamic systems, possibilities to improve associated drawbacks can not be ruled out. Amongst others, long training time, dependence on initial parameters, lack of appropriate network topology are some of the disadvantages.

Therefore there is a need for development of optimal network structures for ANN based load forecasting models to improve the forecast error. The main research question of the project can be formulated as:
How can supervised ANN-based load forecasting models be rationally trained using back propagation algorithms and possibly obtain an optimal network structure for short-term load forecasting?

1.4 Problem statement

Generally, electric load forecasting is a complex exercise. Although ANNs are proven superior methodologies for STLF compared to traditional techniques, the design of optimal network structures has not yet been successfully implemented. And the fact that an electric load is a non-linear function, traditional forecasting methods are simply not suitable for the application due to the lack of nonlinear mapping ability. Intelligent techniques on the other hand, require optimal network structure and

unified training algorithms in order to improve the accuracy of the forecast as well as the performance of the network.

Problem statement: *To develop optimised Artificial Neural Network-based models for Short-Term Load Forecasting and apply these models to a real life case study to evaluate the performance of the proposed approach and provide one week ahead forecast for the CPUT Bellville Campus power system network.*

1.4.1 Load forecasting problem decomposition

Phase I- *Approximation tool*

Analysis of the components of the existing neural network prediction structures, development of the ANN-based models, the optimization criterion needs to be reviewed.

1.4.1.1 Neural Network Architecture

A basic introduction to ANN is required to assist the reader in having a far reaching understanding on the concept, thus a chapter on this subject is an absolute requirement. General applications of ANN have also to be highlighted.

For model parameters selection, a systematic approach with regards to number of hidden layers, number of nodes, epochs, network performance, desired activation functions, and training period has to be unambiguously formulated.

Moreover, the application of gradient descent error back propagation algorithm to the network to get correct set of weights needs to be carefully analyzed.

The evolved models need to be trained, thus the selection of training algorithm and parameters needs to be explicitly understood.

1.4.1.2 Back Propagation Algorithm and Training Parameters

For the network to accomplish realistic targets, the network learning phase requires a teacher. This requirement of is often referred to as “supervised learning”. The following network development issues or points required a good understanding: learning rate, momentum factor, local and global minima, generalisation, and memorization of the network.

1.4.1.3 Development of ANN-based forecasting models

The development forecasting models is the engine of phase I and the whole project entirely. This implies that the appropriate ANN topology and training algorithm need to be identified, implemented and analyzed.

Phase II – (*Data Collection, Poor or Inadequate Data Treatment, Pre/Post Data Processing)*

1.4.1.4. Historical load data

The total electric load of the Cape Peninsula University of Technology needs to be measured preferably at the main intake substation for at least two seasons (winter and summer). Subsequently, measurements were taken at the main intake substation using (E1 Enermax) Maximum Demand Meter and data were downloaded using Optical head to RS232/USB communication options. As for departmental loads, a roving Lovato Power Meter (DMK 40 type), class 1 meter, with data logging features, was programmed and installed at various distribution boards/substations feeding the following departments: Administration, Electrical, IT and Mechanical. Similarly, data were also downloaded using the Serial port (RS232) communication option to a stationary PC.

1.4.1.5. Weather data

As described earlier, an electrical load is a non-linear function influenced by a variety of variables. Some major factors influencing the load need to be considered. The selection of input variables is a critical research component individually, thus only variables with a sound correlation with the load are considered. In this work, only past contiguous load values and weather related data (daily wind speed, humidity, minimum and maximum temperatures) are used. This decisive factor is necessary to avoid network over fitting. The weather data (both for the summer and winter period) were provided by the South African Weather Service Office based in Cape Town.

Phase III – *Computing and Training*

1.4.1.6 Modelling

The developed ANN models are trained under MatLab®, ver. 7.1, environment.

Phase IV - *Validation and Results*

1.4.1.7 Testing of Models

This part is a key component of the project where the forecasts are compared to the actual load, and the error is computed. Results obtained by using both Feedforward and Elman Recurrent ANN-based models are analyzed and compared to the actual load data. Thus, in this thesis, various input variables such as historical load data and weather data are selected.

The standard mean absolute percentage error *(MAPE) is most reported* performance measure function (Mandal *et al.,* 2006, Al-Shareef *et al.*, 2008, Sharif *et al.*, 2000, Rashid *et al.*, 2005, Conzâlez *et al.*, 2004 and Hayati *et al.*, 2007).

$$MAPE(\%) = \frac{100}{N} * \sum_{i=1}^{N} \left| (L - \hat{L}) / L \right| \tag{1.1}$$

Where L denotes the actual load value, $\hat{L}$ is the forecast load value, and N represents the number of observations or data points.

But in this work, the following performance measure functions were employed: mean squared error *(MSE)* and mean percentage error *(MPE)* to evaluate the performance of the models.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} [(L_{actual}(n) - \hat{L}_{predicted}(n))]^2 \tag{1.2}$$

$$MPE = \sum_{i=1}^{N} [(L_{actual}(n) - \hat{L}_{predicted}(n)) / L_{actual}(n)] / N \tag{1.3}$$

Where $L_{actual\ i}$ is the actual load, $\hat{L}_{actual\ i}$ is the forecast value of the load, and N is the number of data points.

Phase V – *Documentation*

The outline of the thesis is given at the end of this chapter. This work does not necessary form part of the load forecasting problem. However, enough time is allocated to document this experiment.

1.5 Research aim and objectives

The impetus of the study is mainly to develop ANN-based models for STLF, and do network assessments on the bases of the average forecast error, and network performance using different training approaches. The models are trained by using error back propagation algorithm to adjust

biases and connection weights between layers. Possibilities for minimizing the error means of other intelligent-based technique such as genetic algorithm (GA) are also explored and evaluated.

Aim: *To develop supervised ANN-based models for short-term load forecasting (STLF), and evaluate the performance of these models by applying the actual load data of the Cape Peninsula University of Technology Bellville campus to predict the load of one week in advance.*

In order to accomplish this aim, the following objectives are to be achieved:

1.4.1 The process of load forecasting requires a time series of historical data, thus it is required to gather the historical load data of the campus reticulation network.

1.4.2 Climatic conditions may affect the load greatly, especially in areas where sudden weather changes are always anticipated such as Cape Town. Thus this work takes into account the weather effect on the load. To enforce this feature, weather data such as minimum and maximum temperatures, wind speed, and humidity should to be gathered and presented as network inputs.

1.4.3 To acquire and set up a convenient local data management system (database) for storing all collected data.

1.4.4 The possible existence of bad data in the load curves as well as in the weather data cannot unfortunately be discarded. Therefore one of the objectives is to develop a strategy aimed at detecting and eliminating the bad data.

1.4.5 ANNs are vulnerable to raw data, thus it is essential to select suitable techniques for normalizing the data prior to applying them to the network for an attempt forecasting.

1.4.6 To develop two different Artificial Neural Network-based models for STLF and evaluate their performance using the mean squared error (MSE) and other supplementary performance measure functions.

1.4.7 To assess other forecasting methodologies by means of explanatory methods.

1.4.8 The CPUT power system operators with their long time working experience might have some good intuition in manual load forecasting. Therefore it is extremely important to combine their experience in convenient manner.

1.4.9 To train the evolved models under the MATLAB environment and predict the total load for CPUT Bellville campus and the loads of some selected departments with 168 hours lead time forecast.

1.4.10 The other important objective is find convenient model training approach that yields the best results.

1.4.11 Last but not least, to document the work done during the experiment and submit the thesis by November 2008.

1.6 Hypothesis

The average forecast error can be improved by developing and implementing a customized ANN – based models based on the standard error back propagation training algorithm using a convenient training approach.

1.7 Project motivation

General view: accurate short-term load forecasting in power industry assists planners to make strategic decisions on unit commitments, purchase of energy, optimal reserve capacity, system security, and reliability etc. Consumers are also required to predict the load accurately for load management and investment criterion purposes. Without replicating, load forecasting is a two dimensional concept, and this distinction is repeatedly perplexed. Figure 1.2 hereunder attempts to clear possible ambiguities on load forecasting dimensions in power system context. The first dimension, specifically deals with utilities' forecast, and the second one, fundamentally involves consumer load forecast.

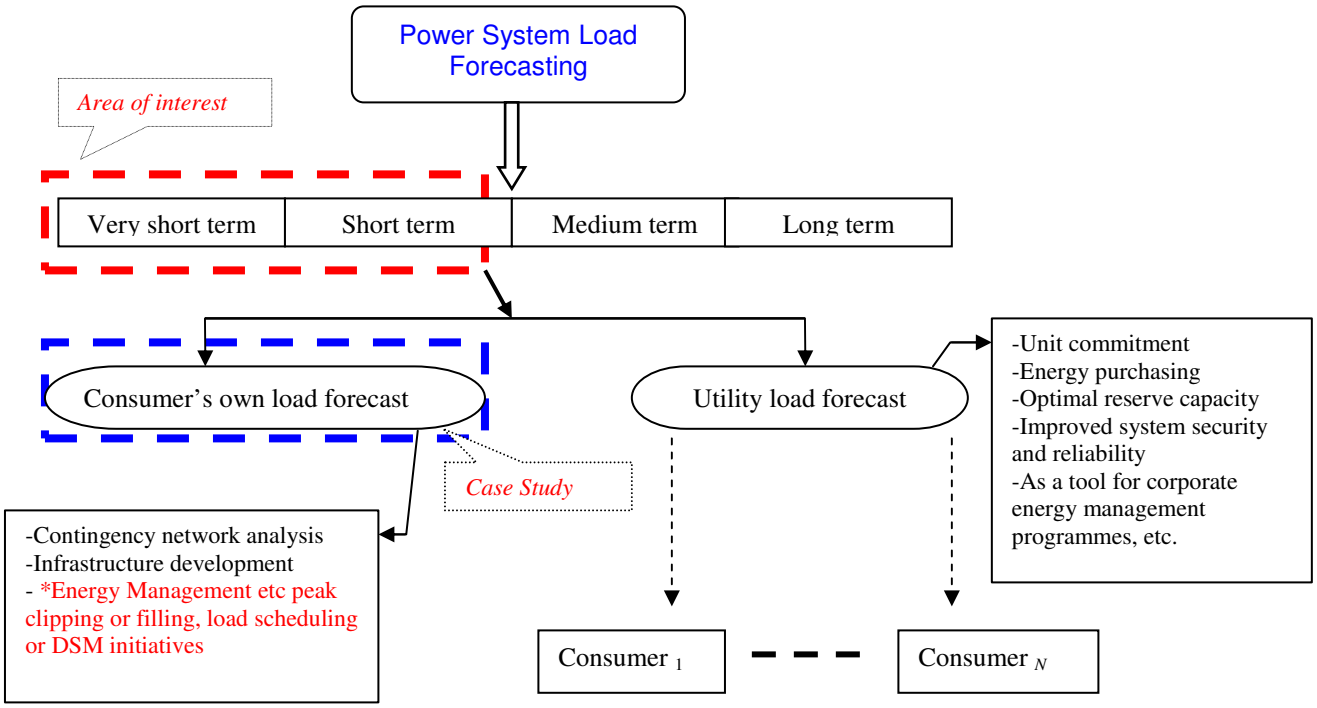


Fig 1.2: A typical load forecasting outline and some applications.

The quality of the network investment decision is based on the analysis of load growth and an economic sound decision can separate the good investments from the investments likely to end-up as "*stranded assets*".

Case study outlook: The CPUT notified maximum demand[1] has considerably increased since the inception from 0.5MVA in 1962 to 2.2 MVA in 2008.

In the absence of any similar studies, short-term future loads should certainly be investigated. As already stated, the forecast can provide the campus management with some guideline with regards to the current maximum demand, and possibly a more realistic NMD (notified maximum demand) can be declared. In this light, load management requirement partially qualifies the significance and applications of this research work. CPUT network planners greatly benefit from this work.

This process (energy management) is an essential element for both utilities and consumers. As illustrated in Figure 1.2, load forecasting is certainly the prerequisite to any load management programmes.

[1]- A *Verbal feedback from the CPUT Management-*

The developed models are applied to actual load of the CPUT – Bellville Campus to forecast the load on short-term bases. Figure 1.3 illustrates extrapolated load growth pattern of the campus since its inception in 1962.

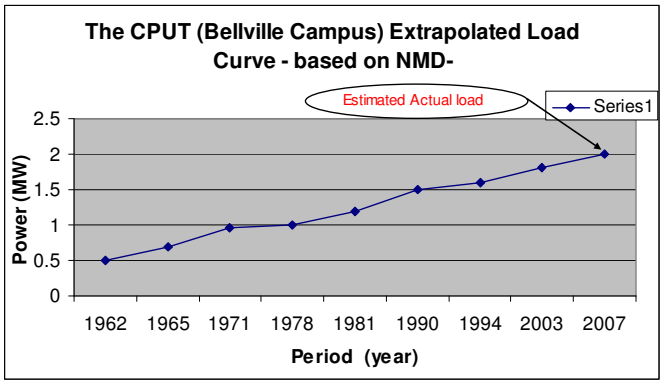


Fig. 1.3: Load growth curve –for the CPUT (Bellville-Campus) derived using extrapolated annual Notified Maximum Demand data.

1.8 Delimitation of the research

The scope of work of this work is solely based on the ANN technique and the comparisons of the results and performances are done only using ANN based models. However, other classical popular load forecasting techniques are discussed and analyzed by means of **explanatory methods**.

The performance evaluations of these models are limited to the CPUT actual load data but the models with slight modifications can be universalised for utilities' applications.

Only short-term load forecasting is modelled and analyzed in this research work.

Although seasonal load variations (winter and summer) are considered, major sudden changes of load are not easily detected by the models. Forecasting the load for holidays or special days does not form part of this work.

The training data set used in this work is limited to four months of historical load pattern for each season.

Both medium to long-term load forecasting are not covered in this work.

1.9 Assumptions

To achieve the aforementioned research objectives, the following assumptions are made:

1.9.1 Short term load can be forecasted using limited historical data and the ANN model has the ability to map the non-linear characteristic of the load given a number of inputs.

1.9.2 The optimal training of the MLP (Multi-Layered Perceptron) network is possible, and the connection weights between layers can be adjusted as desired.

1.9.3 The load profile of each building or department can be measured and assessed separately.

1.9.4 It is also assumed that the installation and commissioning of Digital Power Multi-Meter (DMK 40) for actual load measurements can be completed. Another crucial assumption is that the meter has the required data storage capability, and/or the built-in 2Mbytes (MB) non-volatile memory will be sufficient to uninterruptedly record data for at least two weeks.

1.9.5 The CPUT reticulation network operators (maintenance team) and the local meteorological service office are, upon request, willing to provide the required data.

1.9.6 The real-time simulation and modelling of the ANN models under the MatLab environment will be possible.

1.9.7 It is further assumed that the electricity price or tariff remains unchanged or the change remains within ± 10%, as the impact of the electricity price change is not considered.

1.9.8 The nature of the measured data can be presented by means of a time series load pattern with deterministic characteristics.

1.9.9 A strategy can be formulated to detect and replace possible bad data. And the implementation of an algorithm to normalize all the input data can be easily carried out.

1.9.10 The dependence of the load on the weather conditions is a deterministic one.

1.10 Research design and methodology

1.10.1 Investigation methods

Data collection is a convoluted exercise, thus a combination of research methodologies needs to be defined. In this work, quantitative research method is used as a tool to get required data. Amongst other research investigation technique, some data are obtained by means of questionnaires, personal interviews, and equipment (power meters) for numerical data measurements.

Questionnaires – were typically designed for energy end-users i.e. lecturers, students, house parents etc to provide info regarding electricity use pattern.

Personal interviews – This arrangement is ideally designed to interact with the system operators, maintenance team, and officers from meteorological department. Such liaison structures are necessary to determine critical monitoring points or priority areas.

Modelling – The selected input data are carefully interpreted and normalized before presented as model inputs to avoid, amongst others, over-fitting and input redundancy.

1.10.2 Sampling methodology

Sampling does not necessary apply to the selected method. However the forecasts need to be validated by comparing the output of the model with the actual load data. Another fact that needs to be considered is the measurement points in the campus ring network. More technical detail regarding the network is discussed in chapter 5. Due to limited data logging meters and the time frame, only limited residential and educational buildings are measured. These include and the electrical, information technology (IT), mechanical departments, air conditioning plant (ACP), and the administration building in addition to the total load of the campus recorded at the intake substation.

1.10.3 Literature review

To date, a number of load forecasting techniques have been developed, thus a comprehensive literature review needs to be conducted in order to evaluate different reported forecasting approaches, and possibly establish the associated drawbacks. This subject is exclusively covered in the following chapter. A majority of papers report that intelligent system models, particularly ANN based model is more superior and attractive approach for dynamic systems compared to the conventional one. Further, the forecasting techniques proposed and used in journal papers, databases, and books have been extensively analyzed.

1.10.4 Data collection, normalization, bad data detection and treatment

The historical load data of the CPUT (Bellville Campus) were acquired by means of measurements (i.e. using DMK 40 Digital MultiMeters). Weather data were obtained from a local meteorological office. The weather related data are essential for the model. The gathered data were then analyzed, interpreted, normalised, and assigned to the model in a simplified manner. Other factor such as seasonal change was also considered.

1.10.5 Thesis organization outline

This thesis document comprises eight chapters, arranged in a systematic manner: **the first chapter** of the document mainly discusses the background, purpose of the work and breakdown structure of the work.

The second chapter covers literature review i.e. methods used for load forecasting, comparisons of various papers, findings and remarks. In this chapter, drawbacks of different forecasting methods are also highlighted. This chapter exclusively also discusses the application of genetic algorithm to STLF.

Chapter three of this document broadly talks about power system load forecasting: i.e. the factors affecting the load, characteristics and features of the power system load. This chapter also roofs basic requirements of a good STLF system.

The concept of Artificial Neural Networks is briefly introduced in **chapter four**. This section covers a history and topologies of ANNs, trainings, associated errors, learning paradigms of ANNs etc. Moreover, technique used for teaching ANNs i.e. error back propagation(BP) algorithm, Conjugate Gradient algorithm (CG), Quasi-Newton algorithm (QN), Lavenberg- Marquardt (LM) are also discussed in this chapter.

The content of the **fifth chapter** entails description of the data, data collection methods, and pre-processing. This section also defines how the selected data were imported into the local database and then into the MATLAB workspace. Moreover, this chapter further clarifies the existence of bad data in the load curves and the method used to replace such data prior to storage. The description of the collected data is also clearly defined in this chapter. **Chapter six** generally deals with the application of ANNs in load forecasting. This chapter also encompasses some essential procedure for developing ANN-based model as well the composition of the input vectors (IV) for the developed models.

The core section of the project, the simulation results, is discussed in **Chapter Seven.** This section of the document also covers some specific discussions on different STLF models. In other words, this part of the document entails real implementation of the project i.e. Application of the duo models to STLF using the actual load data of the CPUT Bellville Campus as a CASE STUDY.

The last chapter **(Chapter 8)** of the thesis encapsulates conclusion and future outlook of the project.

1.11 Conclusion

This chapter highlights the basic introduction of the proposed research project. Thus it generally covers the scope of work, significance and objectives of the proposed research project, some brief research methodologies. The chapter also discusses some well thought-out assumptions, delimitations of the research as well as the structure of the proposed work and final organization of the thesis.

In order to establish possible shortcomings of some commonly used forecasting methods, one needs to extensively conduct a literature review on the existing forecasting techniques. This subject is covered in the following chapter and some of the commonly used forecasting techniques are discussed by explanatory means therein.
Moreover, the following chapter also covers a brief introduction of genetic-based search in STLF with a hope to minimize the error function.

The next chapter also highlights a comprehensive comparison of some previous similar works on the following basis: problem statements, methodologies, findings, drawbacks, and areas of possible improvements. The findings, remarks, and conclusions of the literature comparison are also discussed in the next chapter.

CHAPTER TWO

LITERATURE REVIEW

The subject of load forecasting has been in existence for decades, and a number of techniques have been developed. These methods are based on either classical or modern approach. This part of the research work is necessary to establish the statistical relevance of the proposed research project, establish a generalized research question, analyze existing methods, and explore areas of possible improvements. This chapter also covers the analysis of various existing load forecasting techniques, a comparative study of reviewed papers, findings and remarks.

2.1 Overview of load forecasting techniques

Load forecasting has become one of the most significant aspects of electric utility planning. The economic consequences of improved load forecasting approaches have kept development of alternate, more accurate algorithms at the forefront of electric power research (Kermanshahi *et al.*, 2001). Thus the significance of the subject in power systems has drawn alarming interests of many researchers, and to date a number of load forecasting approaches have been developed. Some of the most popular techniques are discussed in this chapter.

Generally, load forecasting models can be classified into two categories: time-of-day models and dynamic models. Time-of-day model is a non-dynamic approach and expresses the load at once as discrete time series consisting of predicated values for each hour of the forecasting period. The second classification involves the dynamic model that recognizes the fact that the load is not only a function of the time of the day, but also of the load most recent behaviour.

The load can be represented mathematically as a function of different factors such as time, weather, customer class etc. In a typical additive model, the estimated load is given by.

$$\hat{L} = L_n + L_w + L_s + L_r \qquad (2.1)$$

Where $\hat{L}$ is the total estimated load, L_n represents the normal part of the load, L_w represents the weather sensitive part of the load, L_s is a special event component that creates a substantial deviation from the usual load pattern, and L_r is a complete random term, the noise.

The previous works presented that, better forecasting results could be realized by considering the electricity pricing in mathematical models. The representation of multiplicative approach can be formulated as.

$$\hat{L} = L_n \bullet F_w \bullet F_s \bullet F_r \qquad (2.2)$$

Where L_n is the normal (base) load, and the correction factors F_w, F_s and F_r are positive numbers that can increase or decrease the overall load. These corrections are based on current weather (F_w), special events *(* F_s *)* and random fluctuation (F_r) which include factors such as electricity pricing or load growth factor.

Forecasting the load using the proposed method involves weighted inputs which are passed through nonlinear transfers. Thus the proposed technique somehow uses a combination of additive and multiplicative methodologies to predict required load values.

2.2 Classification of forecasting methods

The following methods are widely used for load forecasting: similar day approach, regression models, time series, neural networks, expert systems, fuzzy logic, statistical learning algorithms, etc. These methods can be classified in terms of their degrees of mathematical analysis used in the forecasting model and are presented into two basic types, namely: quantitative and qualitative methods.

In most cases historical data are insufficient or not available at all, yet it is anticipated for planners to accurately forecast, thus qualitative forecasting methods are generally used. Among others, these methods include: Delphi method, curve fitting and technological comparisons. Other forecasting techniques such as decomposition methods, regression analysis, exponential smoothing, and the Box-Jenkins approach are quantitative methods. These methods are considered in the thesis.

The following techniques for load forecasting are analyzed and compared in the thesis:

- Regression models
- Time series
- Time-of-day models
- Similar-day Approach
- Stochastic time series models
- Intelligent system based models (i.e. ANNs and GA)
-

2.2.1 Regression methods

Regression is one of the most widely used statistical techniques and it is often easy to be implemented. The regression methods are usually employed to model the relationship of load consumption and other factors such as weather conditions, day types, and customer classes. This method assumes that the load can be divided in a standard load trend and a trend linearly dependent on some factors influencing the load (Gross *et al.*, 1987).

The mathematical model can be written as:

$$\hat{L}(t) = L_n(t) + \sum_{i=1}^{n} a_i x_i(t) + \varepsilon(t) \tag{2.3}$$

where $L_n(t)$ is the normal or standard load at time *t*, a_i is the estimated slowly varying coefficients, $x_i(t)$ are the independent influencing factors such as weather effect, $\varepsilon(t)$ is a white noise component, N is the number of observations, usually 24 or 168.

2.2.1.1 Linear Regression

Linear Regression is the most popular method, and often used to forecast the load affected by a number of factors ranging from meteorological effects, per capital growth, electricity prices, economic growth etc. The model can be formulated as:

$$\hat{L}(t) = \beta_0(t) + \beta_1 x_1(t) + \beta_2 x_2(t) + \cdots + \beta_k x_k(t) + \varepsilon \tag{2.4}$$

where, β_i are the regression parameters with respect to x_i, and ε is an error term.

This model assumes that the error term ε has a mean value equal to zero and constant variance.

In this model, regression parameters (β_i) are estimated from observation of the load and the affecting factors. Let β_i *(i=0,1,2,…k)* be estimated in terms of b_i *(i=0,1,2,…k).* The forecast load ($\hat{L}$) becomes:

$$\hat{L}(t) = b_0(t) + b_1 x_1(t) + b_2 x_{(t)2} + \cdots + b_k x_k(t) \tag{2.5}$$

Using the least square estimates method which minimizes the Residual sum of squares (SSE) (Bowerman *et al.*, 2005) the parameters b_i can be obtained:

$$\underline{B} = [b_0 \quad b_1 \quad b_2 \cdots b_k]^T = (\underline{X}^T \underline{X})^{-1} \underline{X}^T \underline{L} \tag{2.6}$$

where $\underline{L}$ and $\underline{X}$ are the following column vector and matrix:

$$\underline{L} = \begin{bmatrix} \hat{L}_1 \\ \hat{L}_2 \\ \vdots \\ \hat{L}_n \end{bmatrix} \quad and \quad \underline{X} = \begin{bmatrix} 1 & x_{11} & x_{12} \cdots x_{1k} \\ 1 & x_{21} & x_{22} \cdots x_{2k} \\ \vdots & \vdots & \vdots \quad \vdots \\ 1 & x_{n1} & x_{n2} \cdots x_{nk} \end{bmatrix} \tag{2.7}$$

Hence the parameters are calculated, and this model can be used for prediction. Accurate load values ($\hat{L}$) are achieved if the standard error s is small.

$$s = \sqrt{\frac{SSE}{n-(k+1)}}, \quad SSE = \sum_{i=1}^{N} (L(n) - \hat{L}(n))^2, \quad L(n) - actual, \quad \hat{L}(n) - forecasted \tag{2.8}$$

2.2.2 Time series

Time series methods are based on the assumption that the data have an internal structure, such as autocorrelation, trend or seasonal variation. The first impetus of the approach is to accurately assemble a pattern matching available data, and then obtain the forecasted value with respect to time using the established model. The model characteristics can be formulated as: (Janacek *et al.*, 1993)

$$\hat{L}(t) = L_n(t) + S(t) + R(t) \quad t = ...-1,0,1,2... \tag{2.9}$$

Where $L_n(t)$ denotes the standard load value, $S(t)$ is the seasonal term, and $R(t)$ is the irregular or random component.

2.2.3 Time-of-day model

This approach presents a simplest form of load forecasting. The model uses the previous week's actual load pattern to predict the present week's load. Alternatively, a set of load patterns is stored for typical weeks with different weather conditions. These are then heuristically combined to create the forecast.

The equation of the model can be written as: (Gross *et al.*, 1987)

$$\hat{L}(t) = \sum_{i=1}^{n} a_i f_i(t) + \varepsilon(t) \tag{2.10}$$

where the load at time t, $\hat{L}(t)$ is considered to be the sum of explicit time functions, $f_i(t)$ usually sinusoids with a period of 24 hrs or 168 hrs depending on the forecast lead time, a_i are slowly time-varying coefficients, and $\varepsilon(t)$ represents the error term.

2.2.4 Similar days approach

Similar days approach can be formulated on the bases of Euclidean norm. The model assumes the load as a row vector in which weighted factors are used to assess the similarity between the forecast day and searched previous days. Suppose, the vector $\mathbf{x} = [x_1, x_2, ..., x_n] \in R^n$ the equivalent Euclidean norm can be defined in R^n space by the formula:

$$\|x\| = \sqrt{x_1^2 + ... + x_n^2} \tag{2.11}$$

Similarly, for Euclidean norm in a form of complex vector $z = [z_1, z_2, ..., z_n] \in C^n$, where C^n represents the dimensional complex space. Hence the corresponding representation can be written as:

$$\|z\| = \sqrt{|z_1|^2 + ... + |z_n|^2} \tag{2.12}$$

where $|z|$ is the amplitude of the i^{th} complex component.

It is useful to utilize the evaluation determined by the Euclidean norm, which makes us understand the similarity by using an expression based on the concept of norm. Decrease in Euclidean norm results in better evaluation of similar days. The model equation is used as Euclidean norm with weighted factors and can be formulated as: (Mandal *et al*, 2007)

$$D = \sqrt{w_1(\Delta L_t)^2 + w_2(\Delta L_s)^2 + w_3(\Delta T_t)^2} \tag{2.13}$$

$$\Delta L_t = L_t - L_t^P \tag{2.14}$$

$$\Delta L_s = L_s - L_s^P \tag{2.15}$$

$$L_s = L_t - L_{t-1} \tag{2.16}$$

where L_t is the hourly load on forecast day, L_t^P is the hourly load on historical days, ΔL_t is the load deviation between the load on forecast day and the load on historical days, L_s is the slope between L_t and L_{t-1}, L_s^P is the load slope of historical days, ΔL_s is the deviation of slope between the load on forecast day and the load on historical days, and ΔT_t is the temperature deviation.

2.2.5 Stochastic time series models

Unlike other classical forecasting techniques, the stochastic time series method expresses the current load linearly in terms of its previous value, and a white noise series with zero mean and variance. (Janacek *et al.*, 1993 and Moghram *et al.,* 1989)

This representation introduces the backshift operator and qualifies the method to partially handle the complexity of dynamic load forecasting. The model can be presented as:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + \varepsilon_t \quad (2.17)$$

where, x_t is the current load value, x_{t-1} and x_{t-2} are previous load values, and ε_t is a zero mean white noise.

The basic idea of this approach is to introduce a backshift operator. Suppose B denotes the operator, then the previous load value becomes:

$$x_{t-1} = Bx_t \quad (2.18)$$

Similarly, the m^{th} previous load value can be formulated as:

$$x_{t-m} = B^m x_t \quad (2.19)$$

Substituting equation (2.18) and (2.19), the model can be presented as:

$$\phi(B)x_t = \varepsilon_t \quad (2.20)$$

where, $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$ (2.21)

There are two popular methods used to calculate parameters ϕ_i for forecasting, namely maximum likelihood estimation (MLE) and least squares estimation.

Literature reports that MLE method is considered attractive because it has minimum variance and is unbiased, thus this approach is considered in thesis.

The MLE assumes a sample of dependent observations x_t , t= 1,...,*N* each with density function $f(x_t)$. The combined density function can be formulated as:

$$f(x_1, x_2, ..., x_N) = \prod_{t=1}^{N} f(x_t \mid \underline{x}_{t-1}) \quad (2.22)$$

where, $\underline{x}_t$ represents all observations up to and including x_t . $f(x_t \mid \underline{x}_{t-1})$ is the conditional distribution of x_t given all observations prior to *t*.

To compute the required forecasting parameters, the MLE approach further predicts that the zero mean white noise (ε_t) remains normally distributed.

Suppose the mean of the conditional distribution is $\phi_1 x_{t-1} + \phi_2 x_{t-2}$, and the variance is σ^2 . The model equation is derived as:

$$f(x_t \mid x_1, x_2, ..., x_N) = \frac{1}{\sqrt{(2\pi)}\sigma} \exp\left[\frac{-(x_t - \phi_1 x_{t-1} - \phi_2 x_{t-2})^2}{2\sigma^2} \right] \quad (2.23)$$

Similarly, set of x_1 and x_2 to be fixed and defined by the conditional likelihood as:

$$\tilde{L}(\phi) = \prod_{t=3}^{N} f(x_t \mid x_1, x_2, ..., x_{t-1}) \quad (2.24)$$

Finally, the desired parameters can be computed by minimizing $\tilde{L}(\phi)$.

2.2.6 Intelligent system based models

"The thread that unifies so many different concepts is woven from the interpretation of the intelligent system. Practically speaking, an intelligent system is one which employs Artificial intelligence (AI) to fulfil some or all of its computation requirements" (Warwick *et al.,* 1997)

Many studies have reported the superiority of the intelligent system methods in load forecasting models. The next section now briefly discusses some of the widely used artificial intelligence based techniques.

2.2.6.1 Artificial Neural Networks

An artificial neural network (ANN) is a computational model inspired by a biological nervous system. The network consists of interconnected group of neurons and processes information using a connectionist approach to computation. The distributed processing by neurons results in intelligent outcome. ANN-based model learns to perform a desired task directly from examples using special training algorithms. The basic structure of an ANN process is illustrated in Figure 2.1. This method is also often regarded as an adaptive system that changes its structure based on external or internal information that flows through the network during the training.

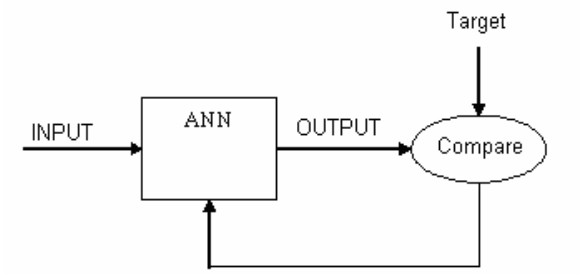


Fig 2.1: A simple ANN structure

2.2.6.1.1 Neural Network Function Approximation tool

Neural networks are essentially non-linear circuits that have the demonstrated capability to do non-linear curve fitting. The approach is also suitable to analyze dynamic systems.

The outputs of an artificial neural network are some linear or non-linear mathematical functions of its inputs.

The inputs may be the outputs of other network elements as well as actual network inputs. In practice network elements are arranged in a relatively small number of connected layers of elements between network inputs and outputs. The target output is achieved by adjusting neuron weights in a Multilayer Perceptron.

2.2.6.1.2 Proposed ANN-based Models

A number of ANN-based structures have been developed. The three common ANN family models are: Feed-forward, Radial based, and Recurrent types ANNs. The proposed research project explicitly analyzes the performance and accuracy of a feed-forward and a recurrent ANN-based model.

2.2.6.1.3 A Multilayer Perceptron (MLP)

Figure 2.2 shows a network configuration of a three-layered feed-forward network.
The inputs are fed into the input layer and multiplied by interconnection weights, and then passed through an activation function before passed on to the next layer.

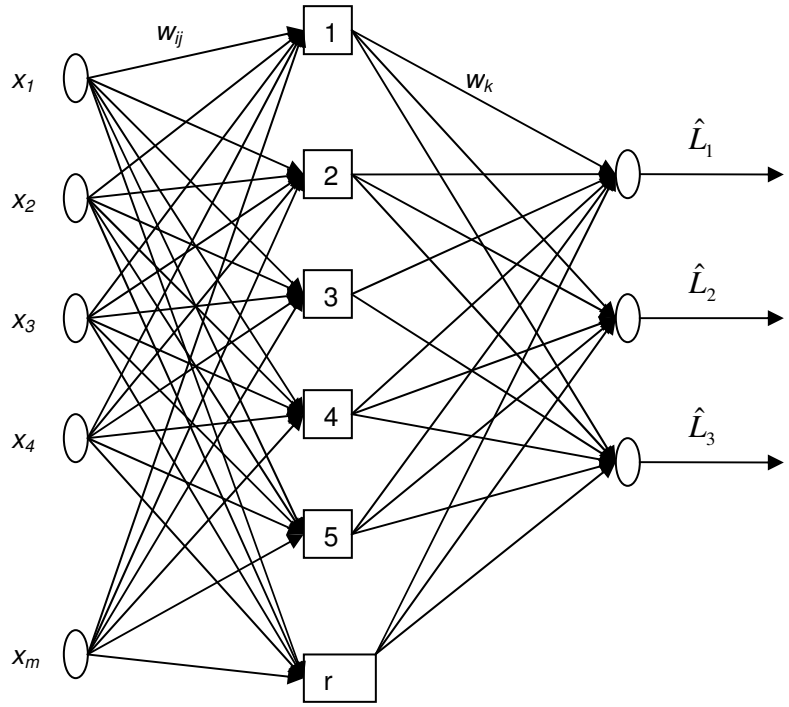


Fig 2.2: A Feedforward neural network with one hidden layer and multi-outputs

2.2.6.2 Expert Systems (ES)

This method uses Artificial Intelligence at a high level perspective. It is derived by system developers and experts through experience and close interactions. AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behavior. It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine.

The basic idea of this approach is to manipulate and encapsulate high level knowledge in an attempt to imitate the behaviour of an expert (Warwick *et al.*, 1997)

The ES does not require any particular model hierarchy or a historical trend. The forecasting process is rather encapsulated by rules derived from interviews with human expert. Once these rules are unambiguously and correctly defined, ideally the uncertainties which might influence the load would be taken care of, thus this approach should be reliable.

2.2.6.3 Fuzzy approach

The term "fuzzy logic" emerged in the development of the theory of fuzzy set which Dr. L.A. Zadeh pioneered by the mid 1960s. A fuzzy logic model is a logical-mathematical procedure based on an "IF-THEN" rule system that mimics the human way if thinking in computational form. Generally, a fuzzy rule system has four modules (Alvisi *et al.,* 2005):

a. Fuzzification of the input – process that transforms the "crisp" into a fuzzy input.
b. Fuzzy rules – IF-THEN logic statement that connects the input to the output variables.
c. Fuzzy inference – process that elaborates and combines rule outputs.
d. Defuzzification of the output – process that transforms the fuzzy output into a crisp output.

Since inception, this technique has gained a wide recognition and a variety of products ranging from washing machines, air conditioners, cameras to industrial process control, medical instrumentation, signal processing and speech recognition have been developed, despite the early scepticism view in the Western World.

In load forecasting specifically, fuzzy rules based on demand forecasts must be developed to provide domain specific information to enhance the non-linear models. The expert knowledge developed in the model can easily be incorporated using linguistic descriptions to high quality data for load forecasting (Warwick et al, 1997).

2.6.4 Evolutionary Computing

Literature of most recent STLF works reports that Genetic Algorithm (GA) is one of the suitable approaches especially for load forecasting network structure optimization (Srinivasan, 1998 and EL-Naggar *et al.*, 2005) Most of the drawbacks associated with traditional intelligent systems are the dependence on initial parameters, long training, network topology setup etc can easily be addressed by using this method (Srinivasan, 1998).

Because of the projected dominance of GAs in STLF, it was found necessary to include a brief introduction on theory of behind the approach.

2.2.7 Genetic-based optimization

The standard back propagation training-based rules can be used to solve a variety of simple optimization problems. However, as problem complexity increases, their performance falls rapidly. Other drawbacks include issues like: long training period, single point search, scaling, initial parameters dependence etc. However, genetic algorithm (GA) is regarded as an alternative approach. . This technique was initially discovered in the mid 1970s at University of Michigan by John Holland. The main idea was to design artificial systems retaining the robustness and adoption properties of natural systems. Since the inception, these methodologies were then further improved by other researchers and are now widely used in various fields (business, science, engineering etc) to solve a variety of optimization problems that lie outside the scope of the standard optimization Toolbox.

GA mimics the biological processes to perform a random search in a defined N-dimensional possible set of solutions. For an optimization problem, one needs to search and find the best solution in a given search space. The idea behind the GA's principle is inspired by Darwin's theory of evolution (*survival of the fittest*).

2.2.7.1 How does a GA work?

The algorithm is commenced with a search space containing set of solutions (or chromosomes) called *population.* Like its counterpart in nature, Chromosomes from one population are chosen based on *fitness* and then combined to form new generation. The best fit individuals are more likely to be selected and subsequently reproduce in the next generation. This approach is encouraged by a hope that the new population will be better than the old one. These procedures are carried out repeatedly until some pre-determined stopping conditions (such as generation, time limits) are met. Like its counterpart in biology, this process requires some genetic operators such recombination, crossover, and mutation (Mishra *et al.*, 2008, Davis, 1991 and Eberhert *at al.*, 1995).

To implement a genetic based search, one needs to define the objective function to be optimized. Strictly speaking, the objective function can be viewed as the input to the algorithm. This function is primarily intended to provide a measure of how individuals have performed in the problem domain. Depending on the optimization goal, most fit elements in case of a maximization problem would ideally have the largest numerical values. For instance, the objective function of a supervised ANN-based load forecasting model can be written as:

$$\min f = \frac{1}{N}\sum_{i=1}^{N}(L - \hat{L})/L \qquad (2.25)$$

where N is the total number of observations or hours, L denotes the desired load value, and $\hat{L}$ represents the predicted load.

2.2.7.2 Conventional Optimization versus GA

GA differs considerably from conventional search and optimization techniques. The four notable and most important differences as discussed earlier and also reported in the work of (Goldberg, 1989 and Maifeld *et al.*, 1994) are:

1. GAs do not require derivative information or other auxiliary knowledge but use an objective function instead.
2. GAs implement a parallel search in a population, not single point
3. GA use probabilistic transition rules, not deterministic ones.
4. Except in real-value representations, GAs work on encoding of the parameter set rather than parameters themselves.

However, the proposed models will not be trained using GA based rule, but this brief GA introduction could be useful for general knowledge. Thus the following section now brings a comparative analysis of some ANN-based forecasting models.

2.2.8 Comparative analysis of some load forecasting ANN-based models

In order to establish the shortcomings and evaluate the performance of existing methods, various papers had to be compared.

The comparisons have been made on the following bases:

- Problem statement
- Objective of the project
- Proposed or used method
- Results or findings
- Drawbacks and possibilities for improvement

2.2.8.1 Literature comparison – findings

The viewed papers illustrate a variety of solutions for load forecasting related problems using different methods, particularly for short-term load forecasting, thus a universal distinction regarding shortcomings of various techniques has been drawn and the proposed approach could be a superior attempt.

Literature shows that different researchers use different methods to address a load forecasting. **Sharif** *et al.,* **(2005**) presented a multi-layer feed-forward neural network model with the aim to compare the forecasting accuracy of a time-series and an ANN-based model. The ANN-based model gave reasonable results.

Chen *et al.,* **(2002)** evaluated the impact of electricity prices in a load forecasting model. This assessment would typically be suitable for areas where sudden electricity tariff increases are experienced as it greatly affects the forecasting accuracy.

Adepoju *et al.,* **(2007)** have used a supervised neural network –based model to forecast the load in the Nigerian power system. The study however did not consider the influences due to weather conditions, thus the accuracy could be improved.

Satish *et al.,* **(2004)** have discussed the effect of the temperature on the load trend using an integrated ANN-based method. The study concluded that the integrated model resulted in less error of prediction. Among other weather variables, only the temperature was incorporated in the model, thus a consideration of other factors would greatly improve the result.

Rashid *et al.,* **(2005)** presented a feed forward and feed back multi-context artificial neural network (FFFB –MCANN) as a practical approach for load forecasting. They have proposed the use of the rate values rather than the absolute to produce better accuracy.

Al-Saba *et al.,* **(1999)** illustrated the application of the ANN to long-term load forecasting. The model forecasted the annual peak demand of a Middle Eastern utility and repeated the process using a time-series approach. The study established that the ANN-based model produces better forecast rather classical methods (ARMA etc).

Table 2.1: Comparative analysis of some existing ANN-based forecasting models

PAPER	PROBLEM STATEMENT	OBJECTIVE	MODEL	CONSTRAINTS	FINDINGS
Sharif and Taylor, 2000 *"Short-term load forecasting by Feed forward neural networks"*	To obtain Average forecasting error using feed-forward neural network (FFNN) and a conventional time series package	To predict an hourly load of one day ahead using both methods and evaluate the corresponding performance index and forecasting error.	A trained multi-layer feed-forward neural network was used for STLF purposes.	Load forecasting accuracy mainly depends on the training and selected time period of the forecast.	The FFNN produced better results, thus the model was used to forecast the load.
Chen, Canicares, and Singh, 2002 *"ANN-based Short-term load forecasting in Electricity Markets"*	The impact of electricity prices in a short-term load forecasting model	To incorporate the electricity pricing in the model and establish influences.	The common multi-layer feed forward ANN based model was considered.	The load-price relationship is highly nonlinear and difficult to model.	The study concluded that if electricity prices are not considered in a model, the number of interactions required to achieve the same forecasting accuracy would be practically doubled.
Adepoju, Ogunjuyigbe, and Alowode, 2007 *"Application of Neural Network to load forecasting in Nigerian Power System"*	The operation and planning of a power utility company requires accurate model for electric power load forecasting.	To implement a short-term load forecasting using neural networks (ANNs) and apply it to the Nigerian electric power system.	A supervised artificial neural network model has been used to obtain the forecasts.	The number of neurons in the hidden layer must carefully be chosen – as too many neurons may lead to overspecializing and subsequently loss of generalizing capability.	After experimenting a variety of hidden layer neurons from 5 – 11, the 11 neurons were finally utilized because it offered a better model characteristic.
Lauret, Fock, Randrianarivony, Manicom- Ramsamy, 2007 *"Bayesian neural network approach to short-term load forecasting"*	Optimal Neural Network structure for load forecasting.	To improve the accuracy of ANN-based model using Bayesian method.	A probabilistic model using Bayesian Neural Network method was presented.	Uncertainty in model's input needs to be developed.	The developed model considers a probability density function (pdf) over the weight space. This arrangement handles some of the uncertainties in the modelling, thus further development could increase the quality of the forecasting models.

PAPER	PROBLEM STATEMENT	OBJECTIVE	MODEL	CONSTRAINTS	FINDINGS
Satish, Swarup, Srinivas, and Rao, 2004 "*Effect of temperature on short-term load forecasting using an integrated ANN*"	The impact of temperature on the average forecasting accuracy on short-term.	To evaluate the load variation with and without considering the temperature.	An ANN based model was used to predict the load. The model was divided in various sub-components namely; the basic ANN, the Peak or Valley Averager & Forecaster, and adaptive combiner to form an integrated model.	The evaluation was specific and did not consider other weather parameters	The average forecasting error was around 4% even with a sub-modeled system.
Rashid and Kechadi, 2005 *"A practical approach for electricity load forecasting"*	The impact of the absolute values and change in weather components, and past load on forecasting accuracy.	To utilize the change in weather components and/or historical load data rather than absolute values.	The paper presented a model called feed forward and feed back multi-context artificial neural network (FFFB – MCANN) using exogenous and endogenous variables.	Recurrent ANN models are complex and often require good training	The recurrent neural network has been used and ±1.5% a steadily mean average percentage error was achieved.
TAl-Saba, and El-Amin, 1999 *"Artificial neural networks as applied to long-term forecasting"*	Forecasting the annual peak demand for a Middle Eastern utility	To explore various forecasting techniques and compare the forecasts with the proposed method.	The paper presented different time-series load forecasting models and the ANN based approach	The future uncertainty makes long-term load forecasting an extremely challenging computational problem.	The ARMA technique has registered the maximum error of about 40%. This result proves the prior selection of the ANN based model. The ANN-based model provided very good results.
Ganzalez, and Zamarreno, 2004 *"Prediction of hourly energy consumption in building based on a feedback artificial Network.*	Prediction of energy consumption in buildings	To present an approach for load prediction with high precision.	A feedback ANN-based model was presented. The ANN training had been achieved by means of hybrid algorithms.	Only limited load variables were considered.	The model used a non-optimized data window size and hidden layers which could seriously affect the over-all forecasting accuracy.

PAPER	PROBLEM STATEMENT	OBJECTIVE	MODEL	CONSTRAINTS	FINDINGS
Srinivasan, 1998 *"Evolving artificial neural networks for short-term load forecasting "*	Shortcomings of commonly used back propagation ANN-based models	To improve the BP-based ANN model, eliminate the drawbacks, and establish an optimal network structure for better forecast.	A three-layered feedback back propagation network, trained by genetic algorithm (GA) was developed.	The model did not have the ability to detect sudden load changes.	The GA approach illustrated a superior representation, and addressed some of the problems encountered by using ordinal back propagation models. The approach also free the model from long training time weight adjustments
Madal, Senjyu, Urasaki, and Funabashi, 2006 *"A neural network based on several-hour-ahead electric load forecasting using similar days approach*"	Linking traditional load forecasting techniques with an intelligent ANN based network.	To unite similar days approach load forecasting methods with an ANN based network	A combination of an ANN-based model and a classical technique was used to forecast one-to-six hour ahead forecasting. The model used Euclidean norm and weighted factors to a simple algorithm.	A great variation in weather conditions may influence the forecast accuracy	The model indicated very good results for one hour ahead forecasting with a MAPE at 0.98% , despite the further increase to 2.4% for six-hour ahead forecasting.
Topalli, Erkme, and Topalli, 2006 "*Intelligent short-term load forecasting in Turkey"*	Inaccuracy in load forecasting and numerical instability for time-series forecasting methods.	To introduce an intelligent load forecasting method.	They have used an Elman's recurrent neural network model with integrated hybrid learning which prepares the model to combine real-time load forecasting and real time trainings.	The ability of the developed model to establish an optimal network structure and corresponding learning algorithms.	The model indicated better results compared to that of a time series. The average forecasting error recorded was approx. 1.6%
Kandil, Wamkeue, Saad, and Georges, 2006 *"An efficient approach for short-term load forecasting using neural networks"*	The impact of estimated values (Historical load data) on load forecasting.	To demonstrate neural networks – ability to predict the load with minimal input data (i.e. Temperature only)	The well-known multi-layered feed forward ANN-based model had been used to forecast for a local load. The model used Levenberg-Marguard training algorithms.	Only preliminary results were discussed in the study.	The paper presented the approach only, and the ANNs capabilities to forecast the load without using a historical load threshold.

PAPER	PROBLEM STATEMENT	OBJECTIVE	MODEL	CONSTRAINTS	FINDINGS
Xiao, Ye, Zhong and Sun, 2007 *"BP neural network with rough set for short-term load forecasting"*	Drawbacks in ANN-based load forecasting models.	To evaluate the relationship between inputs and outputs in a dynamic environment.	They used a rough set back propagation (RSBP) neural network approach, and momentum method to enhance the training speed despite improving the credibility of the Algorithms.	The development of an optimal threshold for attribute deduction was never achieved. More historical data is required.	The RSBP and BP model results were compared. The RSBP presented better results for maximum values.

Ganzalez *et al.,* **2004** have used a feedback ANN-based model to predict energy consumption in buildings with high precision. The model was train by means of hybrid algorithm. The optimal network structure was not evidently achieved.

Srinivasan (1998) has used a dominant back propagation ANN-based model and genetic algorithm as an attempt to evolve the optimal neural network structure. This approach is powerful despites the fact that the model is unable to detect sudden load changes, thus the approach still needs further development.

Madal *et al.,* **(2006)** presented a comparison of a classical load forecasting technique with an ANN-based model using actual load data. The models were used to forecast the load one-to-six hours ahead and again the MAPE showed that the ANN-based model provides reliable forecasts. Again, the optimal network structure for better forecast was never achieved.

Topalli *et al.,* **(2006)** have used a recurrent neural network method using hybrid learning scheme to offline learning with real-time to forecast Turkey's total load one day in advance. The study reported an average error of 1.6%. The forecasting accuracy could be achieved by employing good network training.

Kandil *et al.,* **(2006)** have explored the ANN capability to predict the load without necessary using the historical load trend, but only temperature instead. The study reports that using estimated load values may lead to a great degree of inaccuracy in the forecast, thus only the temperature was used as an input. Because of the ANN's input-output mapping ability, this approach could be efficient. However, the better results could be achieved by selecting other importance input variables and better network training parameters.

Xiao *et al.,* **2007** have introduced the rough set and its ability to study and remember the relationship between the inputs and outputs. A multi-layer back propagation neural network was used in the study and momentum method was also applied to decrease the sensitivity of local parts of error curve surface. This approach requires further development to attribute deduction threshold.

Lauret e*t al.,* **(2007)** have used a model called Bayesian neural network as an attempt to design an optimal network for short-term load forecasting. The Bayesian NN model is a new proposed methodology but also requires derivation of better noise models and uncertainties consideration.

2.2.9 Conclusion

A comprehensive literature review on existing load forecasting techniques is carried out in this chapter. Moreover, a comparative analysis of some specific models has been covered in this chapter. The findings are as follows: various papers compared intelligent system based models to conventional load forecasting methods. Such assessments, ideally are intended to prove the capabilities of ANN-based models to classical ones, and often pay no attention to the shortcomings of modern systems. The focus on network model optimization has never been explicitly highlighted in a majority of reviewed papers, thus there is an enormous need in evaluating similar network class models and exploring the possibilities of network optimization for load forecasting. Literature also shows that classical methods are outperformed by intelligent system based models, hence an ANN-based model is proposed for this project.

In the contrary, a majority of models that use Artificial Neural Networks have a number of drawbacks. The three major problems encountered are: long training period, the dependence on initial parameters, and the development of an optimal network structure for better load forecasting. Surely ANN-based models for load forecasting can be improved by employing good

network trainings using appropriate training algorithms (genetic algorithms, customized back propagation algorithms, rough set theory etc), as a result this topic needs to be investigated.

Since this research project is based on forecasting an electric power system load, there is a need to discuss general requirements for a good forecasting system and also the factors affecting the load at different time horizons. Thus the following chapter of this document briefly covers the aforementioned issues.

CHAPTER THREE

POWER SYSTEM LOAD FORECASTING

This chapter discusses the characteristics of power system loads and the effect of endogenous and exogenous factors on an electric load. These factors determine the amount of energy usage at a given time. In fact, the accuracy of a forecast depends predominantly on these factors. In addition, the latter part of this chapter covers general requirements of a good forecasting system.

3.1 Overview

An electric load refers to the power consumed by an electric circuit at its output terminal. Figure 3.1 shows a simple electric circuit presented by a voltage source V_s in series with an internal resistance R_s, and then the load at the circuit terminal.

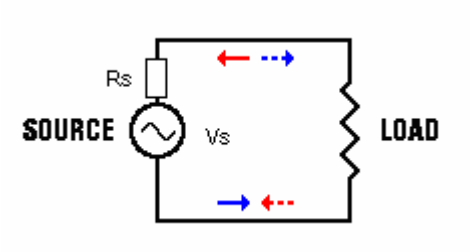


Fig 3.1: A simple AC circuit

The set-up illustrated in Figure 3.1 can readily be generalised to a complete AC circuit by introducing line parameters such as reactive and capacitive impedances.

3.2 Characteristics of a power system load

The behaviour of a power system load depends on a number of factors. Thus the load does not satisfy the superposition principle i.e. the load is not necessary the sum of linear independent variables, but it is rather a nonlinear system. As commonly known, nonlinear problems are often difficult to solve and much less understood than linear problems. Hence, power system load forecasting is a great computational problem for many researchers and network planners.

3.3 Factors influencing the load

The system load, in power operation context, is the sum of consumers' load at the time. A consumer load trend is as different as '*chalk and cheese*' and influenced by a thousand of factors. There is no any engineering rule that guides the selecting of these factors. Thus this process is mainly based on

experience gained from the correlation analysis between the load and potential influencing factors. The only existing criterion is that load forecasting in power systems can generally be divided into three different time horizons: short, medium, and long term load forecasting. However, factors affecting the load at different time horizons are not necessary the same. The variation in the short-term load depends heavily on time factors i.e. hour of the day, day of the week etc. Medium to long term load is determined by factors such as population growth, per capital income, demographic factors, gross domestic product (GDP) and so on. Most utility companies use this distinctive dependence in selecting the input variables.

Possible factors that may influence the load in various time horizons are discussed the following section.

3.3.1 Short-term load forecasting (STLF)

Generally, it is impractical to determine the major influencing factors for a certain forecast time horizon. However, some of the factors highlighted in Figure 3.2 could be considered for STLF.

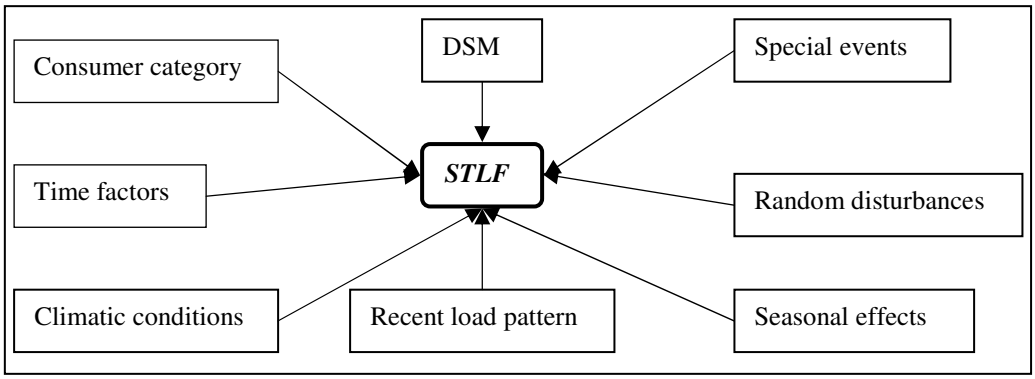


Fig. 3.2: Possible influencing factors – for STLF.

a) **Consumer category:** residential, commercial, and Industrial – the electrical usage trend is unique for customers that belong to different categories. Normally, industrial type customers consume a large part of electricity. Residential load are normally the smallest and often complex to determine due to different consumer electricity usage routines.

b) **Meteorological related-factors:** temperature, humidity, cloud rate, and wind speed. Weather conditions influence the short term load greatly. Literature shows that, weather conditions have a strong correlation with the load, especially the temperature. In fact, load forecasting can be concluded using only weather related inputs.

A general guideline on the effect of weather related conditions on the system load has been discussed in the work of (Say, 1976). Typically, the impact are as follow: a 1˚C temperature change will produce a change in load of about 1 per cent, an overcast sky as compared to a clear sky will increase the load by 3-4 per cent, while thick fog may increase it by 10-12 per cent, and an increase of 1 per cent in load for every 4km/h wind velocity. Authors (Satish *et al.*, 2004), report that STLF can be concluded by only using weather related conditions such as temperature.

c) **Recent historical load pattern:** Despite the fact that load forecasting can be concluded by using limited input variables, the recent and corresponding past load trend is mainly the backbone of the forecast. The selected factors will then define the dimension of the training data set. Ideally, the training data set should cover the whole problem space as this enables the network to identify and map input-output patterns adequately. The biggest problem here is to determine the optimal number of consumption instances, prior to the value to be predicted. Usually, the data points set is arbitrary determined, but taking into consideration the result obtained by using correlation analysis. However, the block entropy analysis performed by (Santos *et al.,* 2006), reports that the use of long chains of contiguous load values does not offer any sort of benefit in the design of the IV of the ANN. For STLF, 1 week past contiguous load values could be adequate to design an IV structure and subsequently forecast the load with reasonable accuracy (Santos *et al.,* 2007).

d) **Time of the day:** The load has a cyclic characteristic. Unlike industrial load that is slightly stable, depending on production levels, the residential one changes greatly during the day. Peaks for a residential load are frequently observed in the mornings, mid days, and in the afternoons.

e) **Day of the week:** i.e. weekdays or weekends. Generally, each day of the week has its unique electricity demand trend. Sometimes the days of the week could be selected as inputs to a forecasting model. (Adepoju *et al.,* 2007) report that two load patterns namely: weekends and week days load patterns. Their result illustrates that the greatest peak loads are often recorded on Fridays, otherwise the load is slightly constant from Monday through Thursday.

f) **Seasonal effects:** The seasonal changes have direct influence on system loads. For supply security and reliability, most utility companies consider worst case scenarios (seasonal

variation effects). The heating loads (in winter) and cooling loads (in summer) are normally viewed as maximum and minimum peak load thresholds respectively. The seasonal variation determines whether a utility company is summer or winter peaking (Gross *et al.,* 1987).

g) **Special events:** Depending on the magnitude of an episode in terms of power requirements, the associated demand could significantly influence the planned system load. For instance, the FIFA 2010 Word Soccer Cup related activities may greatly influence the load in the South African power system. Often such events are thorny to consider in a load forecasting model.

h) **Holidays:** Other factors that add uncertainties in the forecasts are religious or national holidays. Load variations due to these factors are minimal and often negligible, thus often weekend load curves are used instead.

i) **Random disturbances:** Unlike residential consumers with smooth load curves, a slight change in industrial loads causes a significant load variation. For instance, if one considers the impact on the load as a result of starting-up or shutting down large loads such as smelters, wind tunnels, or steel mill, indisputably the change is not negligible.

j) **DSM (Demand Side Management) Initiatives:** This approach entails actions that influence the quantity or pattern of use of electric energy consumed by end users. A good example would be the prevailing power shortage especially in the South African region which has certainly forced power utility companies in the region to implement ongoing and aggressive DSM initiatives such as compact fluorescent lamps (CFL) lighting, the use of energy efficient motors, implementation of time-of-use (TOU) tariffs, residential load management. Other demand management strategies include emergency generation, independent power production, and industrial cogeneration. Although DSM is a short term strategic approach, it helps the utilities to defer the expenditure associated with additional generation capacity by reducing total end-user demand (Boake, 2003). These measures are required immediately to minimise the risk of load shedding of *"non-essential load"* until the additional generation capacity can be built, but the reality is that they will be required for the foreseeable future (Ross *et al.,* 2008)

The forecast time horizon determines which factors could be selected in the design of input vector of a forecasting model. For medium-to-long term load forecasting the factors discussed below may need to be considered.

3.3.2 Medium-to-long term load forecasting (M-LTLF)

Contrary to the factors affecting STLF, medium-to-long term load forecasting involves numerous uncertainties that are often tricky to predict. Some of these factors are indicated in Figure 3.3.

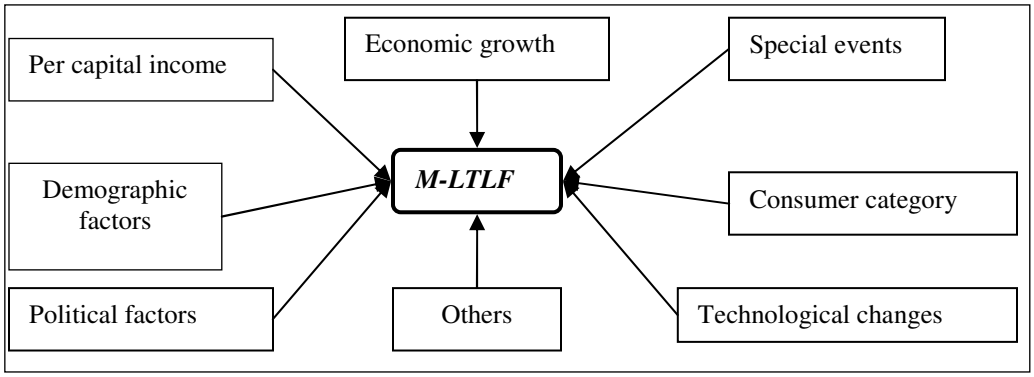


Figure 3.3: Typical influencing factors – for M-LTLF

a) **Economic growth:** Depending on the economic growth indicator such as GDP (Gross Domestic Product) of a particular state, an increase in the economic growth would ideally raise the demand (NIEIR, 2007). In fact, economic related factors influence long term load trend greatly, thus the consideration requires a special attention in a long term load modelling.

b) **Energy and environmental policies:** Energy policies (often corresponding with international laws) have direct influences on the system load. This involves policies aimed at promoting energy efficiency, and development of off-grid power generations. Among others, most of the utility companies are fully conversant of the environmental policies i.e. *greenhouse polices* and/or *emission trading schemes.* For a utility to operate within these policy bounds, it implies that additional electricity supply resources are required and thus it influences the electricity demand growth.

c) **Technological change:** The change in technology may positively or negatively influence the load. For instance, if one considers an aggressive use of gas air conditioning technology which will as result offset the electric conditioning systems, thereby reducing the system load.

d) **Per capital income:** Often income rate per household determines the corresponding space comforts. Consumer preferences (heating or cooling loads) rely heavily on real income ranges, thus the demand is either positively or negatively influenced.

e) **Demographic factors:** A raise in the population of a certain locality would obviously result in an increase in the demand. For a single locality, the impact seems to be relatively small but the lump-sum effect is surely not negligible.

f) **Customer category:** The impact of a large industrial load growth could affect the forecast in the long run, thus this should be considered.

g) **Political factors:** Politically motivated wars in a particular state can influence inhabitant's decisions to immigrate to other countries. The movement directly affects the population, and subsequently affect the load negatively and vice versa. Other stakeholders such as independent power producers (IPP) may also find it senseless to invest and operate in a peace challenged country.

Others: For example, upstream developments, such as a construction of a new generation plant or regional inter-connectors, will certainly influence electricity prices and demand through final retail prices. This implies that the electricity tariffs are highly likely to be increased, hence affordability determines usability.

3.4 Load features

3.4.1 Types of load banks

An electric load is formed by load banks. There are three common types of load banks: resistive load bank, reactive load bank, and capacitive load bank. Load banks are used for different purposes. A general example for a resistive load would be: the conversion of electric energy to heat by power resistor in a circuit. For a resistive load, the angle between the current and voltage is zero or in-phase, thus the circuit always operates at a unity power factor.

Reactive load bank includes inductive apparatus such as induction motors, transformers, lighting etc, and has a lagging power factor. Capacitive load bank is somehow similar to the reactive load bank in rating and purpose, except that a leading power factor load is introduced.

3.4.2 Load categories

Power system loads can be divided into three categories, namely: linear loads, nonlinear loads, and special loads. For linear loads, the load impedance is always constant regardless of the applied voltage and the load current increases proportionally as the voltage increases and vice versa. Typically, linear loads include motors, incandescent lighting, heating loads etc.

For instance, starting a large motor has a great impact on power system load, especially for direct on line (DOL) motors due high starting currents. For a DOL, the starting current is normally 7 times higher than the rated full load current. Thus this type of loads may contribute significantly to cost intensive high peak demands. Other soft-starting techniques (i.e. star-delta, rotor resistance, auto transformer starting etc) are widely used for economic operations of electric machines.

Load currents in electronic loads such as computers, uninterrupted power supply (UPS), variable speed motor drive, thyristor controlled equipment etc are neither proportional to the instantaneous voltage nor continuous. These loads are typical examples of nonlinear loads. Other loads, such as motors used in lift applications, have different nature of operations i.e. here motors are frequently started. Therefore, general design standards specify lower starting currents. Typically, starting current of a lift motor should be less than 75% of the rated current. The contribution of special loads and nonlinear loads to the overall power system load often falls within the average of the load base.

3.4.3 Consumer classes

Electric power is used by different types of consumers: residential, commercial and industrial. Load profile for each consumer class is unique and it depends mainly on types of appliances used, daily activity patterns. Residential consumers generally use electric energy for heating, cooling, and illumination and consume about half of the electric power. Commercial businesses purchase electricity for a variety of commercial functions i.e. office machinery, store display lighting, parking bay lighting, escalators etc. Whilst industrial facilities and plants need electricity to power processes such as compressor, conveyor motor, air conditioning, and other manufacturing applications. This distinction of various consumer classes is essential for electric utilities in determining electricity prices, because rates charged for each class are different.

The first section of this chapter gives a comprehensive introduction to the power system load. The following section now discusses some requirements for a good and reliable forecasting system.

3.5 Prerequisites of a good STLF System

Most of demand or load management programs used by electric utilities comprise STLF units. Every utility intends to have a reliable STLF system for economical operations of power systems. The reliability and robustness of the system primarily depend on the accuracy of the forecasts. Though, there are other important requirements for a good STLF system. These requirements take account of:

- Fast speed
- Accuracy
- Automatic data access
- Friendly interface
- Timely forecast
- Automatic performance evaluation of the obtained forecast
- Automatic bad data detection and forecasting report generation.

3.5.1 Fast Speed

Forecasting programs with different techniques, network topologies, training algorithm, desired goals, etc have diverse convergence periods. This means that the speed of the forecasting depends on the system structure. The idea is to put together an optimal forecasting structure that yields accurate results in the shortest period.

3.5.2 Accuracy

Accurate forecasts drastically benefit utilities and other stakeholders in a number of ways. In fact, the main goal of a majority of review papers on STLF is to produce forecasts with a reasonable degree of accuracy. In essence in power generation sector, just a simple decision with regards to price based unit commitment (PBUC) requires an accurate STLF system.

3.5.3 Automatic data access

Load forecasting models or systems normally require a number of inputs, namely: historical load data, weather related inputs, sinusoidal functions and so on. Thus there exists a need of a database where all relevant model inputs are stored. This implies the system should be able to automatically access and extract required data from the database.

The database is principally for historical data. Some inputs such as weather parameters may need to be captured automatically on-line. For this process, the use of some communication options (internet, telemetry, or external modems) could be enforced. The data base is also used to store the results from the developed forecasting models.

3.5.4 Friendly interface
The forecasting model should be easy to use and convenient.

3.5.5 Timely forecast
If a network operator gets forecasts, even accurate forecasts, from the planning department but outdated, it's highly likely that the operator will implement unrealistic decisions. Therefore, a good STLF system should yield forecasts which are timely and corresponding to the desired forecast lead periods.

3.5.6 Automatic model performance evaluation
Because accuracy is a key measure in forecasting, the design structure of a good forecasting system should specify some permissible error limits. In this work, the standard mean squared error is employed. The error bound is defined as: $-2.5\% \leq MSE \leq 2.5\%$. However, this performance measure condition is not applicable when the average forecast error is consistently smaller than these limits.

3.5.7 Bad data detection
Out of range data (outliers) is one of the major challenges in computational problems. In load forecasting for example, historical data are gathered and stored in a database. However, some these data especially estimated data, can fall beyond the reasonable data boundaries, hence a bad data detection strategy should be formulated and implemented to identify and replace the bad data. In the early days, STLF systems relied heavily on the power system operators to interpret the data and subsequently get rid of the bad data (Yang *et al.,* 2006). Detecting bad data manually is a work burden and demanding exercise for operators, thus automated bad data detection can accelerate the forecasting process.

3.5.8 Automatic forecasting report generation
Forecasting results should be presented in a manner that is easily understood and interpreted by the end-user. In simple terms, the results may need to be illustrated in numerical and graphical forms.

3.6 Conclusion
To briefly summarize the preceding chapters, the following concepts have been covered: firstly, a general overview on the subject topic was discussed, and then a review on the existing methods and drawbacks of these techniques were explored. Secondly, here the third chapter covers the basic requirements of a load forecasting model.

The next part of this document now introduces the proposed forecasting method. In other words, the following chapter (chapter four) now covers the concept of Artificial Neural Networks (ANN) in general context, network training, learning paradigms etc as an attempt to help the reader in understanding the architectural network designs of the proposed models.

CHAPTER FOUR

ARTIFICIAL NEURAL NETWORKS (ANNs)

This chapter introduces a brief history of Artificial Neural Networks (ANNs). The chapter also covers an overview of the ANN concept, simplified architecture network topologies, and its basic elements, network training, learning paradigms, generalization and training algorithms including the proposed error back propagation (BP) algorithm.

4.1 General overview of Artificial Neural Network

The novelty of Artificial Neural Network (ANN) theory could undoubtedly be attributed to the experiment performed by *McCulloh* and *Pitts* in modelling bio-systems using nets of simple logical operations back in 1943. The idea was to find a simple parametric nonlinear model for a real neuron. Ever since this innovation, there has been a great interest from various researchers and scientists, thus several ANN-based models in different fields were discovered. The technology though had lost its momentum in the late 1969 till 1986 when the back-propagation of error was discovered. To date, ANN-based models have been successfully implemented in a number of industries ranging from: Aerospace, automotive, defence, electronics, entertainment, financial and so on. Artificial neural Networks have been also successfully applied in medical fields. Crawford, 2000 reports acute appendicitis (diseases and health conditions) analysis based on ANN methodologies specifically. More information regarding the application of ANNs can be found in (Demuth *et al.*, 2008)

ANN is a part and parcel of intelligent based systems, designed distinctively to improve the performance of conventional computing techniques. The biggest drawback associated with the so called conventional methods is the inability to learn and identify patterns in dynamic systems. Thus the need to eliminate this shortcoming through learning is proven essential.

Artificial Neural Network is an information processing paradigm inspired by the way biological nervous systems, such as the brain, process information. The characteristic of the biological neuron is briefly introduced as an attempt to assist the reader to understand the concept better. The human brain has 100 billion biological neurons with about 100 000 connections per neuron. A simplified biological neuron is illustrated in Figure 4.1.

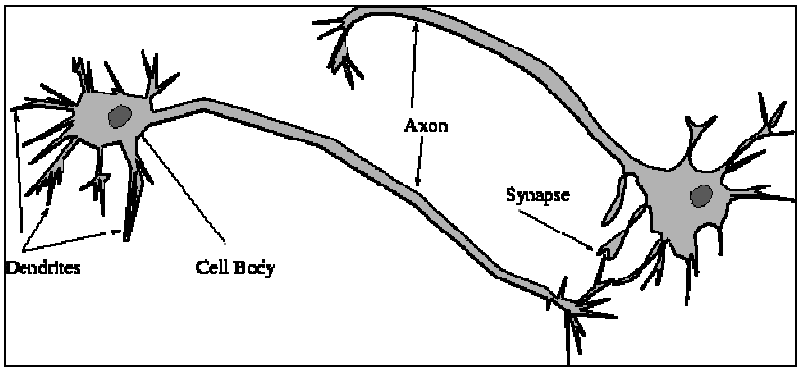


Fig. 4.1: A biological neuron

Biological neurons receive spikes through *synapses* located on the *dendrites* of the neuron. When the spikes received are strong enough and exceed a definite threshold, the neuron is *activated* and fires a signal though the *axon*. This signal travels from the body, down the axon, to the next neuron(s). Learning arises by adjusting the effectiveness of the input (synapses) so that it influences one neuron on other changes. Humans and highly trained animals use the same configuration and summing up to extremely complex networks.

Similarly, an artificial network is made up of simple interconnected processing elements called *neurons*. The neurons are arranged in a layered structure to complete a network competent of executing parallel and distributed computations. Architecture of a simple ANN is shown in Figure 4.2. The attraction of ANN-based models comes with the network's ability to learn, recognise data patterns, and adapt to a changing environment like the human brain. This adaptive characteristic is often called "the human-like reasoning".

The architecture illustrated in Figure 4.2, presents a three layered feed-forward network. ANN has a remarkable capability to develop sense from convoluted or imprecise data, extract patterns and detect trends that are too complex often only noticeable by either humans or other computer techniques. A supervised neural network can be trained as a *"guru"* in a given problem space.

In broad terms, ANN-based models offer a variety of benefits namely: adaptive learning, self organization, real time operation, fault tolerance via redundant information coding. Thus neural network processes information in the similar way the human brain does.

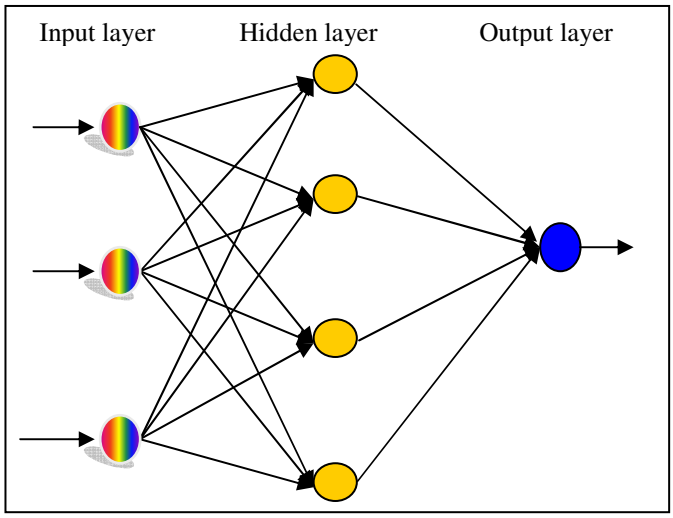


Fig. 4.2: Architecture structure of a feedforward neural network

4.1.1 Basic Architecture of a feed-forward network

The feed-forward network topology illustrated in Figure 4.2 permits signals to travel one way only, from the input through the hidden layer to the output layer. These types of networks are somehow straight forward and associate inputs with outputs. They are extensively used in pattern recognition. This kind of organisation is also referred to as bottom-up or top-down and commonly used in pattern recognition.

Figure 4.2 also shows the commonest type of artificial neural network which consists of three layers. The *hidden layer* neurons are connected to the *output layer* neurons. The functions of each layer in the network are defined below:

a) The input layer neurons represent the pre-processed data fed into the network.
b) The input of each hidden layer neuron is defined by the sum of the input vector set and the connection weights between the input layer and hidden layer.
c) The input of the output neuron is determined by the weighted sum of outputs of the hidden layer neurons.
d) The output of a neuron is defined by the type of the transfer function used in that specific layer.

This type of network is attractive because the hidden neurons are free to develop their individual representations from the input set.

4.1.2 The Perceptron – a network for decision making

The perceptron, a basic neuron, invented by Rosenblatt in 1957 at the Cornell Aeronautical Laboratory in an attempt to understand human memory, learning, and cognitive processes prior to his demonstration on the first machine that could *"learn"* to recognize and identify optical patterns in the early 1960. The mathematical model of the perceptron or artificial neuron is modelled in the similar manner of the biological architectural set-up. Again, the three major components are considered: Axons and synapses of the neuron are modelled as inputs and weights respectively.

The strength of the connection between an input and a neuron is denoted by the value of the weight. The mathematical model of this topology is illustrated in Figure 4.3. The weighted inputs are added together and passed through a nonlinear activation transfer function. Finally, the activation function controls the amplitude of the output of the neuron. The suitable scale of output is usually between 0 and 1, or -1 and 1.

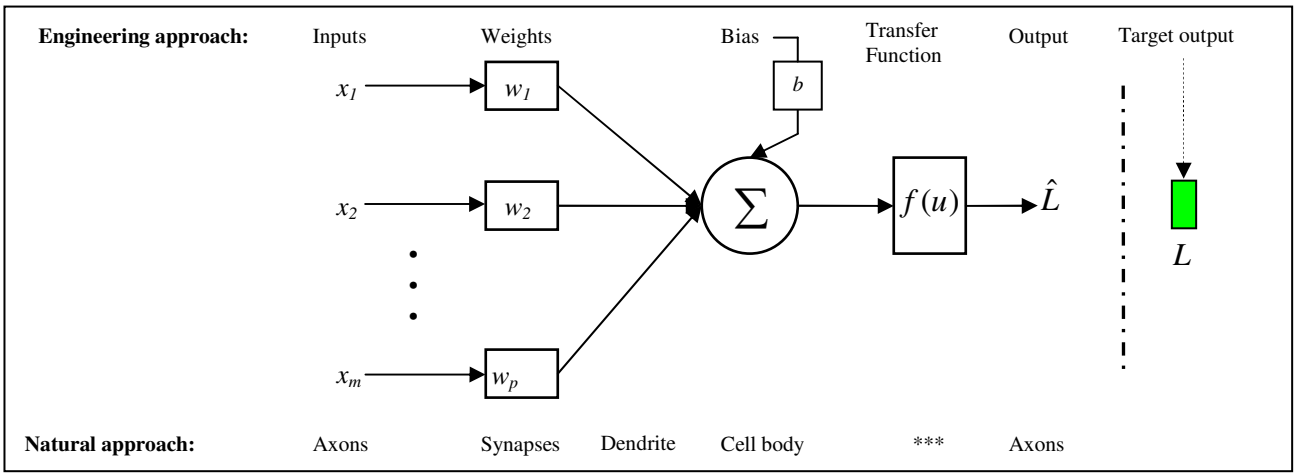


Fig. 4.3: A perceptron model

Thus the output $\hat{L}$ can be mathematically formulated as:

$$\hat{L} = f(\sum_{i}^{p}(w_i x_i + b) \tag{4.1}$$

Where w_i are the synoptic weights, x_i represents the model input vector, and b denotes the bias.

4.1.2.1 Activation functions

The activation function acts as a squashing function. In simple terms, the network activation rule is denoted as activation function. A number of transfer functions are available for selection. These activation functions include (and not limited to): step function, logistic function, tangent-hyperbolic

function, linear function, cosine function and so on. Some of these functions are illustrated in Figure 4.4.

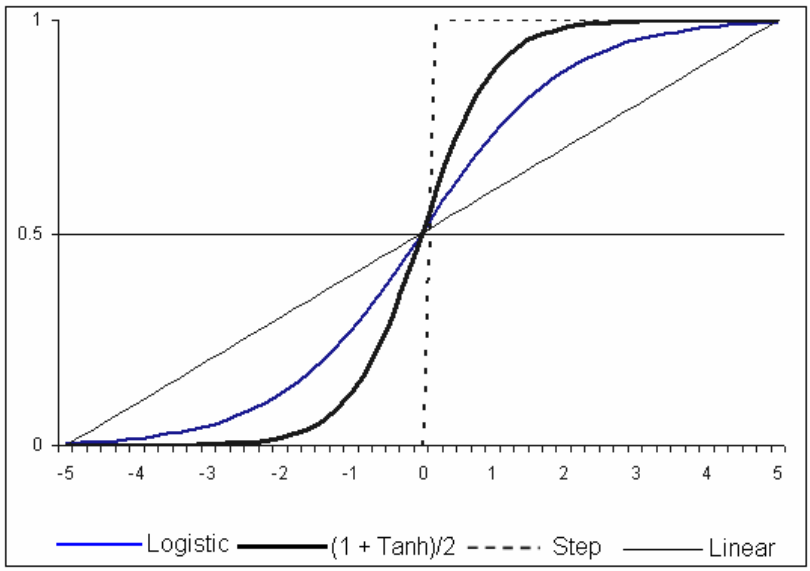


Fig. 4.4: Various types of transfer functions

However there is no engineering rule or *"rule of thumb"* for the selection of transfer function thus the choice is frequently performed arbitrarily. In fact, the effect of different transfer functions on network performance has not been explicitly reviewed in a number of literature papers.

4.2 Soft Computing using Neural Network Topologies

Generally, there are three common types of neural networks: radial basis function network (RBFN), multi-layered perceptron networks (MLP), and recurrent neural networks (RNNs). The distinction in different network topologies can perhaps be attributed to the arrangement of neurons and the connection patterns of the layers. A Multi-Layer Perceptron network is the most popular neural network type and the most of the reported neural network in STLF.

The inner structure of the processing element (neuron) in each network is interconnected differently, and the configuration set-up is often referred to as *network topology*. The behaviour of the network relies greatly on the network topology.

4.2.1 Feedforward Network

In a MLP feed-forward network, connections are unidirectional and no loops are introduced in the network, thus each neuron is linked only to neurons in the next layer. In (Mehta *et al.,* 2003) a feed-forward network is referred to as a directed cyclic graph. This implies no backward links either.

The importance of loopless networks is that computation can proceed uniformly from input neurons to output neurons. And since there are no backward links, activation functions from the preceding time step play no role in the computation. Figure 4.5 shows a simple multi-layered feed-forward network topology with inputs $x_1,...,x_p$ and output $\hat{L}$. Synoptic weights connected to the hidden layer are denoted as w_{ij} and w_k for the output layer respectively.

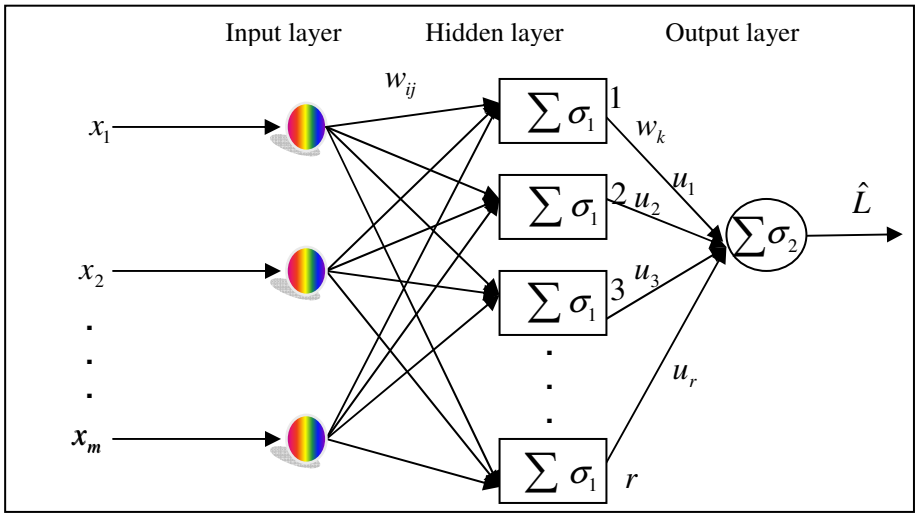


Fig. 4.5: A feedforward network with one hidden layer and one output layer.

Mathematically, the functionality of the hidden neuron is formulated as:

$$u_i = \sigma_1\left(\sum_{j=1}^{m} w_{ij}x_j + b_j\right) \tag{4.2}$$

Where weights and biases $[w_{ij}, b_j]$ are illustrated with arrows feeding into the neuron, σ_1 is the tan-sigmoid, σ_2 represents the pure linear transfer function, [$x_1,...,x_m$] denotes the input vector, and k =1,2,…,r is the number of hidden layer neurons.

In Figure 4.5, only one output layer neuron is considered and these types of networks are used for single output problem (Sjöberg, 2004). In some approximation problems, the number of output neurons equals to the number of outputs. Finally, the output, $\hat{L}$, is represented by the compact notion as $g(\phi, x)$ and it is further simplified as: (Sjöberg, 2004).

$$\hat{L} = \sum_{i=1}^{r} w_k \sigma_2\left(\sum_{j=1}^{m} w_{i,j}^1 + b_{j,i}^1\right) + b^2 \tag{4.3}$$

Where, ϕ is the parameter vector containing all weights of the network, r and m represent the number of neurons in the hidden layer and input vector respectively.

4.2.2 Elman Recurrent (ER) Network

The organisation of a recurrent network is comparable to a human brain network. Unlike MLP network, recurrent structure introduces cycles or loops and backward links in the network (Mehta *et al.*, 2003). Feedback networks are exceptionally dominant and can get extremely convoluted. The behaviour of these types on networks is known to be changing continuously until they reach an equilibrium point. This implies the state of the network remains at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations. Figure 4.6 shows a typical topology of recurrent networks.

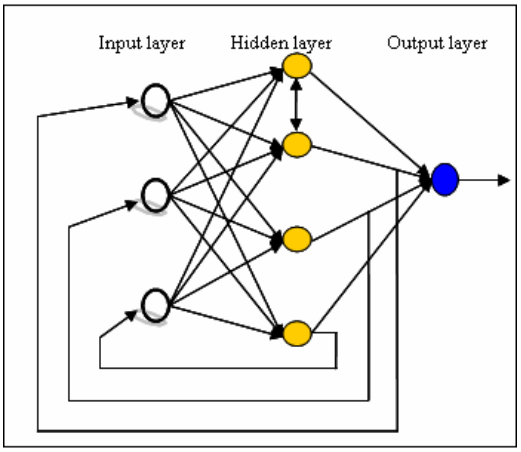


Fig. 4.6: A structure of an Elman (Recurrent) Network

4.2.3 Radial Basis Function (RBF) Network

The radial basis function network is probably the second widely used type of Artificial Neural Network in contrast to the standard Feedforward MLP network. RBF emerged as a variant of ANN in late 80's and it is commonly as a pattern recognition technique. Some of the typical applications of this network include: clustering, function approximation, spline interpolation (Tou *et al.,* 1974).

The RBF network topology is also configured like that of a feedforward network, but the distinction is conceivably can be attributed to the structure of the hidden layer and its output computation approach. Its network also comprises of input, hidden or memory, and output neurons respectively.

Figure 4.7 shows network structure of a two-layered radial basis function network topology. A linear transfer function is used in the output layer and a nonlinear transfer function (normally the Gaussian) for the hidden layer.

The idea of RBF is to execute a mapping of vector variable x into another vector variable u based on the set of learning samples. As commonly known, the estimator which minimizes the mean square error is the conditional average estimator (CAE). Mathematically, the RBF output forms a linear part given by:

$$\hat{L} = g(\phi, u) \tag{4.4}$$

Where ϕ is the parameter vector containing all weights of the network and x denotes the input vector.

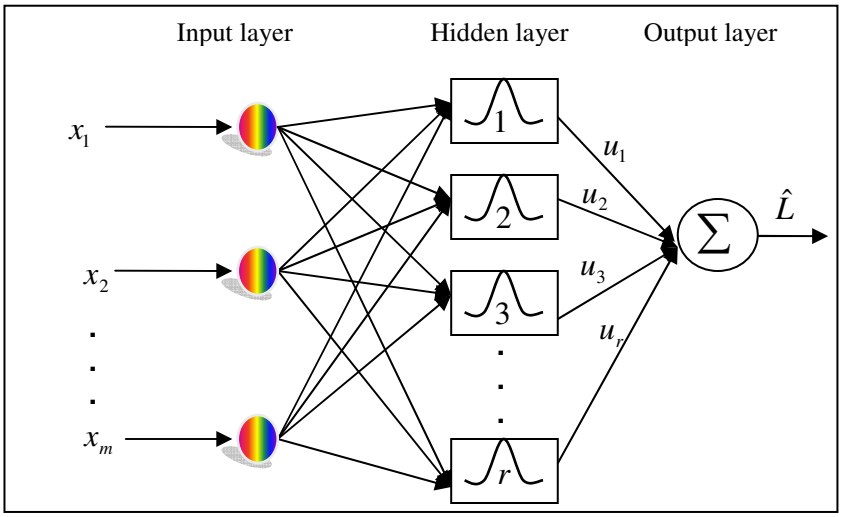


Fig. 4.7: A radial basis function network (RBF) network topology.

Hence, the output $\hat{L}$ is formulated as:

$$\hat{L}(x) = \frac{\sum_{i=1}^{R} y_i . w(x - x_i, \phi)}{\sum_{j=1}^{R} w(x - x_j, \phi)} \tag{4.5}$$

Where $w(x - x_i, \phi)$ represents a smooth approximation of delta function, as illustrated below:

$$w(x - x_i, \phi) = \exp\left[-\frac{\|x - x_i\|^2}{2\phi^2} \right] \tag{4.6}$$

Applying a Gaussian function, Equation (4.5) becomes:

$$\Phi(x - x_i) = \frac{w(x - x_i, \phi)}{\sum_{j=1}^{r} w(x - x_j, \phi)} \tag{4.7}$$

Where the parameters x_i and ϕ describe the center and the width of the receptive field of *i*-th neuron respectively and $\|.\|$ denotes the Euclidean norm.

Each hidden neuron in an RBF takes its input from all the outputs of the input layer x^i. The hidden neuron contains a "basis function" with parameters "center" and "width". The center of the basis function is a vector of numbers c^i of the same size as the inputs to the neuron and there is normally a different center for each neuron in the neural network. Thus the radial distance *'d'* between the input vector x^i and the center of the basis function is the computed as:

$$d = \sqrt{\left(\left(x^1 - c^1\right)^2 + \left(x^2 - c^2\right)^2 + ... + \left(x^n - c^n\right)^2\right)} \tag{4.8}$$

The output of the neuron in the output layer $\hat{L}$ is then calculated by applying the basis function ρ to this distance divided by the width *w*:

$$\hat{L} = \rho\left(\frac{d}{w}\right) \tag{4.9}$$

This type of network is not applied in this work, however more reading materials on RBF can be found in (Neruda *et al.,* 2004), and (Sjöberg, 2004).

4.3 ANN Training and Generalization

There are various processes involved in developing a supervised ANN-based model. Amongst others, training process and validation process are some of the vital steps. The input-output patterns are repeatedly presented to the network during the training process. Through this process, the network learns the subjected environment or patterns, and eventually yields the desired output. The desired output is attained as a result of adjusting weights of all interconnections between neurons to establish the correct set of input-output response. However, during the training process, optimal training time is required to avoid overtraining. Overtraining of the network can be prevented by employing complex stopping criterion. Early stopping (the most common), regularization, pruning, information criterion pruning (ICP), cross-validation pruning (CVP) are some of the stopping methods. Due to space limitation, only some of these methods are covered in this work. However the reader is encouraged to refer to the work of (Rech, 2002). A detailed reference on overfitting is discussed in (Paruelo *et al.,* 1997).

Training the network at infinitum normally results in a reduced error function for a given set of inputs. Though, this does not guarantee better accuracy and robustness of the network because the error could be extremely big if the network is presented with the data it has not seen before. This characteristic is explored during the validation process. Moreover, the validation process also improves the network reliability and generalisation.

4.3.1 Network Coupled Errors

Broadly speaking, there exist two types of errors during the training process: training error and generalization error. The general error presentation can be seen in Figure 4.8. At the beginning of the training, the training error is at maximum, and gradually decreases as time elapses. Optimum training time should be within the circled area in order to maintain network generalisation tendency. In essence, the desired goal should not be an error-free output, as the network will tend to memorise the input-output patterns if trained at infinitum.

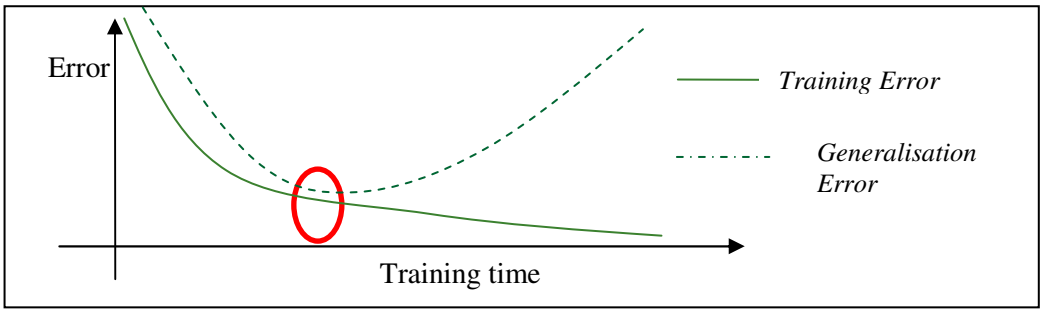


Fig. 4.8: Errors vs. Optimal network training time

4.4 ANN Learning Paradigms

Like in any other intelligent based systems, a desired output of the network does not come by chance. The network rather adapts itself to a stimulus, and ultimately yields the desired output. The main difference here perhaps could be featured to the type of learning a particular network is subjected to.

Broadly speaking, there are different learning paradigms that can be used to train neural networks. Supervised and unsupervised learning are the most common, with reinforced and competitive learning techniques also gaining considerable popularities.

The learning process is essential to adjust network weights in order to reduce the error. During the process of adapting and adjusting the synaptic weights, the network acquires knowledge similar to human-like reasoning.

However, the learning process requires sets of mathematical algorithms describing how synoptic network weights and biases are attuned. By default, MatLab® employs gradient descent based training algorithm commonly known as "*Levenberg-Marquardt*". However other training algorithms could be selected or derived.

4.4.1 Supervised Learning

This is a type of training approach where the input and the desired output are clearly specified. Suppose, the input vector is represented by [$x_1,...,x_m$] and the corresponding output vector is denoted by $y_1,...,y_m$, an optimal rule is to be determined such that:

$$[y_1,...,y_m] = f[x_1,...,x_m] + \varepsilon \quad (4.10)$$

Where ε represents the approximation error. In this example, the error could be a vector.

The idea is that the network adjusts its weights as an attempt to minimize the approximation error preferably to the smallest value possible, and thereafter the network returns a trial result. This result is then compared to the desired output. Figure 4.9 illustrates a basic network structure for a supervised ANN model. Here the error function is fed back into the system as an attempt to find a correct set of input vectors.

If the desired threshold is not attained, another update of the network weight vectors could be initiated. In dynamic systems especially, the need for a network to map the nonlinear relationship between inputs and outputs is enormous, thus this approach is commonly employed in ANN based models.

Some of the key measures to evaluate a supervised learning session are: time required per iteration, number of iterations per unit pattern, convergence points i.e. local or global minima(s) etc.

4.4.2 Unsupervised learning

In contrast to supervised learning, this training method does not require an external teacher. This implies there no explicit target output is presented to the network.

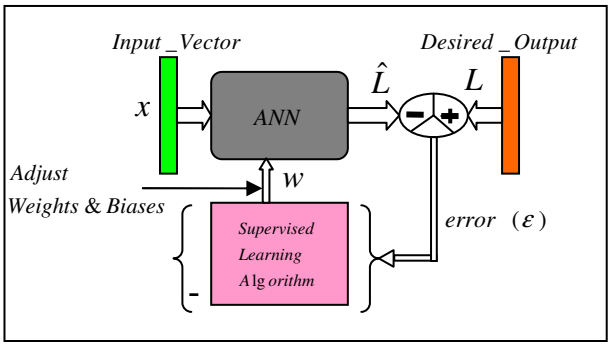


Fig 4.9: A basic layout for a supervised learning paradigm using error correction technique

Unsupervised learning studies how the system can respond to various input patterns during the training session, and subsequently presents the input patterns in a way that reflects the statistical structure of the overall collection of the input patterns. Because this learning method self-organises data offered to the network and discovers their emergent shared properties, it is also often referred to as *"self-organised learning"*. Hebbian learning is one of the common unsupervised learning paradigms.

(Kalteh *et al.*, 2007) report that self organising maps (SOMs) are promising techniques to investigate, model, and control many types of water resources processes and systems. Moreover, unsupervised learning is proved superior in a number of applications including data visualisation, pattern classification etc. In (Valova *et al.*, 2005) a parallel growing architecture for self organising maps has been investigated.

4.4.3 Other Types of Learning

Reinforced learning and competitive learning are other types of learning paradigms. In the reinforced learning an external teacher is required although still no desired output targets are presented to the network. The teacher scores or rates the output of the network with either a pass or fail. A fail score implies the network has to readjust its parameters again and again until targeted outputs are eventually observed. If the network yields a correct response, a reward is given otherwise a penalty is guaranteed for out-of-range responses. However, the response rating limits of the network should be declared so that the teacher should not continue scoring at infinitum.

Competitive learning is known to be a "*racing*" learning method. This approach uses a rule based on the idea that only one neuron in the layer fires at a time for a given iteration. In other words, neurons compete among themselves to be activated. The output neuron that wins the "racing" is then referred to as the *winner-takes-all* neuron. In the *winner-takes-all* schema, the output neuron receiving the largest input is fully credited, whereas outputs of other neurons are suppressed to a 0 value.

Competitive learning rule can be mathematically defined as:

$$w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k) \tag{4.11}$$

Where, $\Delta w_{ij}(k) = \eta(x - w_{ij})$ for $i = j$, otherwise $\Delta w_{ij}(k) = 0$, η denotes the learning rate, x is a randomly picked input, and $w_{ij}(k)$ is the current weight value.

Competitive learning is very common, and most of the feature maps are based on this method. For instance, self-organising feature maps, Kohonen's special class of neural networks introduced in the late 1980s.

4.5 Competent Learning Process for ANNs

In addition to the ANN learning paradigms discussed earlier in this chapter, here other main issues compulsory for a successful ANN training are exclusively introduced. They are: learning rule, and learning theory. Unlike learning paradigm, learning rule classifies how network weights should be adjusted in the learning trials. There are four basic types of learning rule: error-correlation learning (ECL), Boltzmann learning (BL), Hebbian learning (HL) and competitive learning (CL). Most of the supervised learning algorithms use these learning rules during training. Error-correction learning is the method of comparing the network output to the desired output value, and using that error to guide the training. In the most guiding phases, the error values can be used to directly adjust the tap weights, using an algorithm such as the backpropagation algorithm. Mathematically, the error used by this method can be formulated as:

$$\varepsilon = L - \hat{L} \tag{4.12}$$

Where, ε is the error value, L is the target output, and $\hat{L}$ denotes the network output.
At each training iteration, error correction learning algorithms attempt to minimize this error function. This method is mostly used in backpropagation learning algorithms.

The originality of the Hebbian learning rule could perhaps be attributed to a hypothesis proposed by D. Hebb back in the year 1949. The concept is currently adopted as Hebb's Law, and is one of the key ideas in biological learning. The Hebb' Law can be presented in the form of two sub-rules:

1. If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.
2. If two neurons on either side of a connection are activated asynchronously, then the weight of that

connection is decreased.

In simple terms, the Hebbian learning rule specifies how much the weight between two neurons should be increased or decreased in proportion to their activation (Hebb,1949, Mazzoni *et al.,* 1991, and Hertz *et al.,* 1991). Suppose α_i and α_j are the activations of the neurons i and j, and w_{ij} is the synoptic weight between them, then the basic form of the Hebbian rule can be formulated as:

$$\Delta w_{ij} = \eta \alpha_i \alpha_j \tag{4.13}$$

Where, Δw_{ij} is the change is in the synoptic weight, and η is the learning rate. Another powerful learning rule is the delta rule. This rule utilizes the discrepancy between the desired and actual output of each output unit to change the weights between neurons.

Another vital component for competent learning process of neural networks is the learning theory. This concept covers issues related to data quality, data quantity, and best convergence etc. i.e. training data in general. In most of the viewed papers, researchers tend to omit explanations with regards to criterion used for selecting certain variables, training samples, data quality etc. It's obvious there is no general rule in selecting historical data of the phenomena of interest. Thus the input variables are purely identified based on judgements and experiences. In most cases, variables with strong correlation with the target output(s) are suggested. The selection of training data is extremely crucial as it could improve adaptability, reliability, and robustness of an ANN. Training data with an extended phenomena space margin and free from outrange data points are normally preferred. The larger training data can increase the accuracy of network generalization but it also increases the computation time of the learning process.

4.6 Error Back Propagation (BP) Learning Algorithm.

Error back propagation is a common technique of teaching ANNs how to perform a given task. The term *back-propagation* refers to the manner in which the gradient is computed for nonlinear multilayer networks. In simple terms, the gradient is the derivatives of the error with respect to the weight ($g_k = \frac{\partial \varepsilon}{\partial w_k}$), and $k = \overline{1,r}$. This method was first described by Paul Werbos in the mid 1970s, and it only gained recognition about ten years later.

Figure 4.10 shows a simplified supervised ANN training design chart using backpropagation method. First of all, the input(s) and desired target (s) are presented to the network prior to computing the resulting output. The difference between the desired and resulting outputs is known as the error. If

the error is greater than the acceptable threshold, the network will continue to adjust the weights and biases for all neurons using the error.

Though, this process (adjusting of weights and biases) does not continue at infinitum.

The network repeats the process until the error reaches an acceptable value (typically $error \leq goal$), which means that the NN was trained successfully. If a maximum count of iterations is reached without attaining the goal, this implies that the NN training was not successful.

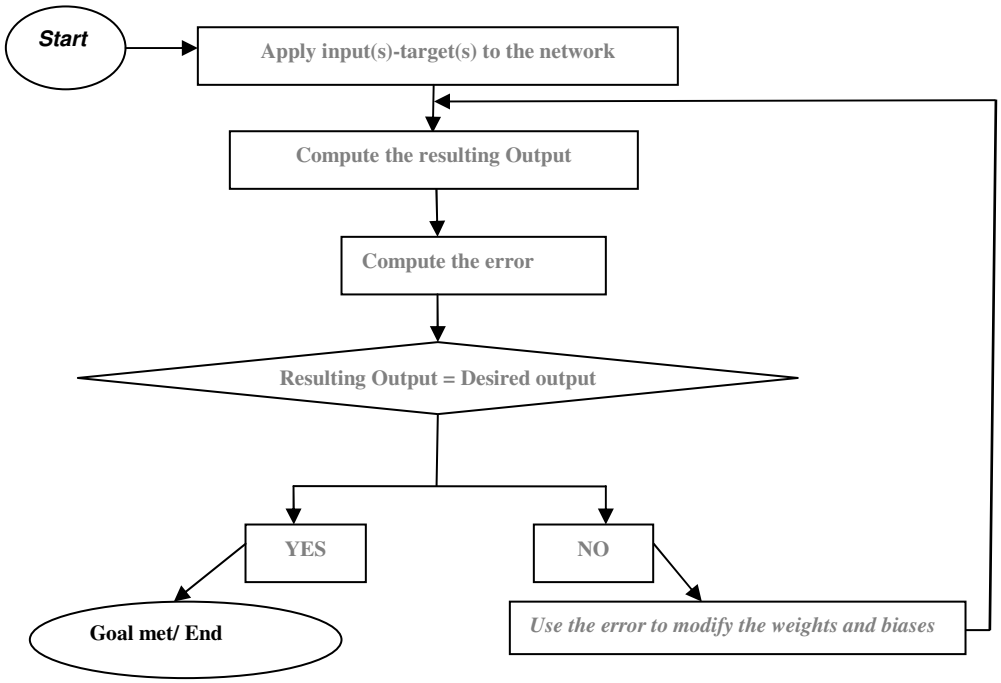


Fig 4.10: A simple design chart for a supervised BP training method

The basic implementation of backpropagation learning deals with the adjusting of network weights and biases in the direction in which the performance function decreases most rapidly, the negative of the gradient. The basic back propagation algorithms are: gradient descent and gradient descent with momentum. The simplest approach of this technique can be formulated as:

$$w_{k+1} = w_k - \eta g_k \tag{4.15}$$

Where w_k denotes the vector of current weights and biases, g_k is the current gradient, and η is the learning rate.

Gradient descent algorithm are reported too slow for practical problems and highly likely to get stuck in a shallow local minima. Thus a momentum term is introduced to enable the network to respond not only to the local gradient, but also to global minima.

With the standard descent algorithms, the arbitrarily selected learning rate is kept constant throughout training. And since the performance of the algorithm is very sensitive to the proper setting of the learning rate, optimal setting of the learning rate is extremely essential. If the learning rate is set too high, the algorithm can oscillate and become unstable. If the learning rate is too small, the algorithm takes too long to converge. (Alam *et al.,* 2007) conducted a study on network structure parameters and concluded that the learning rate, number of hidden layers and neurons have impact on network training. To improve the learning process, other algorithms with variable learning rate such as: Conjugate Gradient algorithm (CG), Quasi-Newton algorithm (QN), Lavenberg- Marquardt (LM) could be used. These types of algorithms are designed primarily for high performance, optimalization, and faster convergence and are often recommended for real world problems.

4.7 Conclusion

This chapter contains the backbone theory of the ANNs. It commences with a brief history of ANNs, some applications and also the biological background for the methodology. Moreover, the chapter also highlights the commonly used Neural Network topologies, and some essential training issues such as input vector (IV), various learning paradigms, stopping criterion, generalisation, and some basic information about training algorithms (including the proposed error back propagation).

It can be seen that prior to training, one needs to present an IV containing training data of the phenomenon of interest to the network. Thus there is a call for data collection. The process of data collection is fairly involving and often time consuming in a sense that the data have to be preprocessed to get rid of out-of-range data. In addition, all data should be normalized or scaled to values ranging from 0 to 1 due to ANN's vulnerability to raw data and to avoid convergence related problems. These important topics are exclusively covered in the next chapter (Chapter 5).

CHAPTER FIVE

DATA COLLECTION AND PRE-PROCESSING

The purpose of this chapter is primarily to discuss types of data used as inputs to the proposed forecasting models and data collection methods used in the research. The chapter also covers the importance of data processing, particularly pre-processing of data prior to forecasting. Moreover, this section talks about the configuration and setting-up of the database engine, and retrieval of data from the database to various forecasting models.

5.1 Data Description

Data collection is a significant function of any type of research study. Inaccurate or insufficient data can impact the results of a study and ultimately lead to invalid or skewed results (Lendasse *et al.*, 2000). In this work, various data collection methods had been employed to ensure adequate historical samples of the load.

As discussed in previous chapters, a power system load is affected by a number of factors. These include exogenous and endogenous variables. A good forecasting system requires variables which have a strong correlation with the load. Thus the evolved models will be presented with the following inputs: load and weather data to forecast the load in different time horizons. As previously discussed, the selection of the input variables is rather done arbitrary, based on the correlation analysis, and sometimes on recommendations made in previous works and experience. Literature reports that, amongst all other weather related factors, temperature has the most notable dependency in load variation.

The actual load data for the CPUT Bellville campus were used to evaluate the effectiveness of the models. The campus 11kV ring connected reticulation network comprises 13 substations mainly feeding educational buildings, residential buildings, and a sport field. The network has 8 x 500kVA, 11/0.4 kV and 6 other 1000kVA, 11/0.4kV distribution transformers. A single line diagram of this network is shown in Figure 5.1. Two types of loads were measured: total load and departmental loads. The measurements for departmental loads were taken at the LT side of transformers. As for the total (combined) load, measurements were taken at the main in-intake sub station (*labelled as Pentech in Fig. 5.1*). The geographical layout of these sub stations as well as some measurement points are also shown in the Figure 5.1.

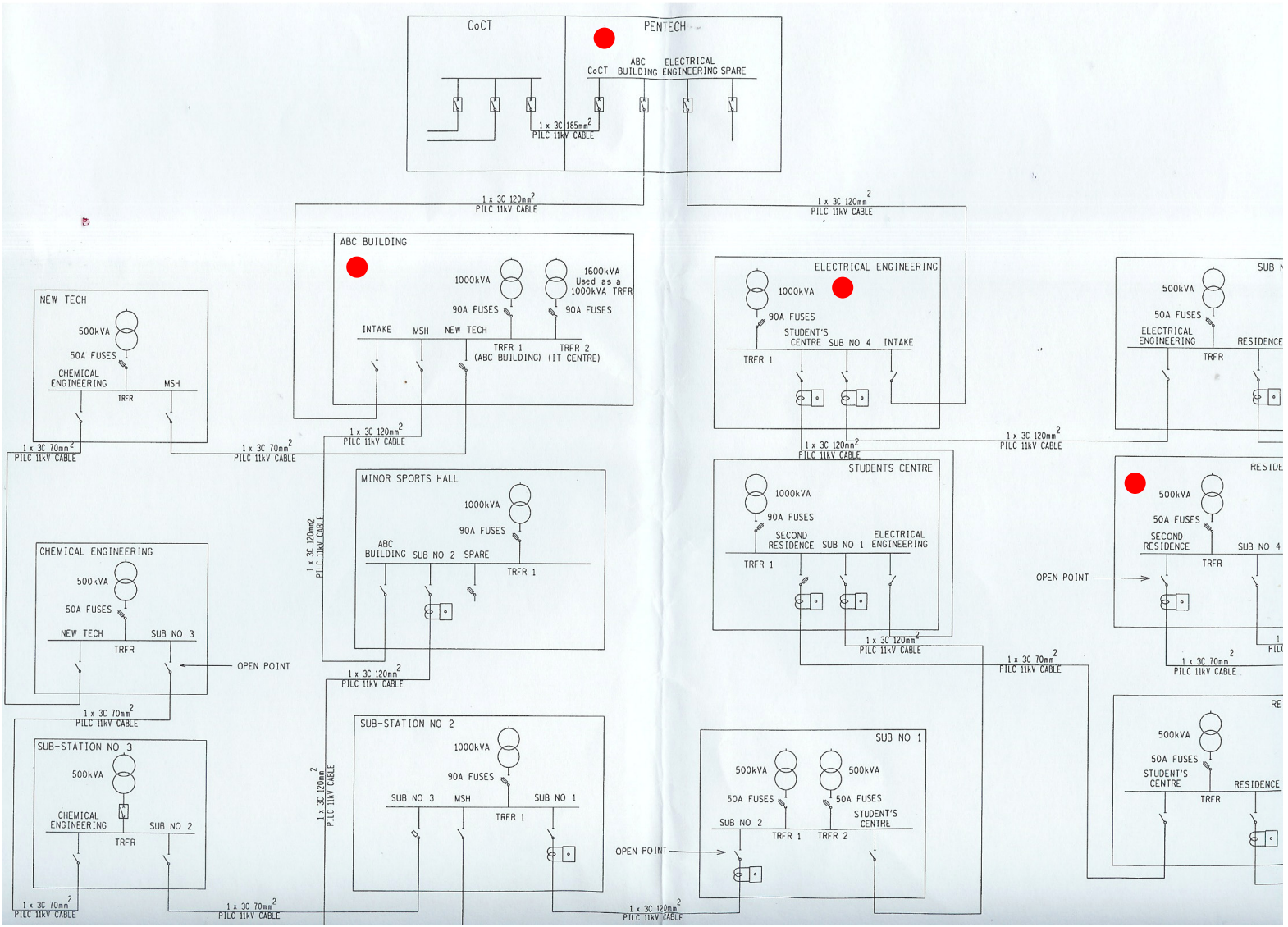


Fig. 5.1: Single line diagram – 11kV network reticulation of the campus

During data collection process, power meters were installed at designated substations to measure power consumptions at different integration periods i.e. quarterly, sub-hourly, and hourly bases. Weather data were obtained from a local meteorological office.

5.2 Data collection methods

In this work, various quantitative data gathering strategies were utilized. Some of the strategies used to gather relevant information are given below:

- Taking measurements - (i.e. the actual load for the entire campus, and departmental loads)
- Observing and recording well-defined load trend (e.g., recess break periods, special events, anomalous days etc)
- Obtaining relevant data from management information systems (the contractual maximum demand threshold – 2.2 MW declared by the institution to the CoCT).
- Administering surveys with closed-ended questions with network operators (e.g., face-to face and telephone interviews, questionnaires)

Based on the Audience, Behaviour, Conditions, and Degree (ABCD) rule on measurable objectives, several inception meetings with audiences or stakeholders were held. Inter *alia*, the widely used face-to-face interactions approach with stakeholders i.e. network operators, CPUT management, and suppliers was initiated to establish rapport and thereof possibly gain their cooperation. Contrary to the said strategy, telephone interviews were also used not only to optimally utilise the resources but also because this communication method is less time consuming. The data collection is an involving exercise and often consumes a lot of time and resources. Table 5.1 – shows a chronological data gathering schedule for this research. Two historical load data namely: combined load and departmental loads can be observed. The combined load constitutes the cumulative actual load for the entire campus recorded by a statistical or check meter at the main in-take substation. As per Eskom/CoCT Maximum demand metering requirements, this meter has been configured to record the load using a 30 minutes integration period.

For departmental loads, the scope of this research was as result extended to record power consumptions for different departments and building within the campus premises at a lower integration sampling period. These data were recorded by a roving Lovato DMK40 power meter for a period of 2 weeks per locality. The DMK40 meter is class 1 meter to IEC/EN 61036 and IEC/EN 61268 standards and it can measure and display more than 251 power parameters.

Data type	Data Source	Recording Integration Period	Parameter	Location/Building	Mobility	Duration
Total load	E-max Max. Demand Meter	30 mins	kW	Main In-take substation	Stationary	9 months
Departmental loads	DMK 40 Lovato Meter	15 mins	kW	Admin. BLD, Elec. ENG, IT, Mech. ENG, and Air Conditioning Plant (ACP)	Roving	2 weeks/ department
Weather	Meteorological office	daily	Min, Max Temp (°C), Wind Speed, Humidity, and Cloud rate	Nearest weather service point (Cape Town)	**	Jan – August 2008. Daily records

Table 5.1: A typical integrated schedule – for data gathering paradigm

5.3 Description of the load data

5.3.1 The combined (total) load data

As previously mentioned, the combined load data were measured at the main in-take substation of the CPUT Bellville Campus for a period of four months per season. Because the load is greatly influenced by seasonal changes, the data set for the total load used has been divided into two categories (seasons): winter and summer. These seasons were subsequently defined as follow: winter - a window period from February till May and the latter season from June – September.

Figure 5.2(a) shows the highest and lowest load curves recorded in winter. With respect to the present notified maximum demand (NMD) of 2.2MW, it can be seen that the load has staggeringly exceeded the contractual threshold. One now begins to think about the subsequent penalty charges, normally billed per exceeded kVA, payable by CPUT to CoCT. These charges can obviously to be avoided if the NMD is reasonably declared. Thus the NMD needs to be redefined to a value close to at least 3 MW.

The entire load profile for the total load, excluding power interruptions due to Eskom load shedding, for the selected winter period is shown in Figure 5.2 (b).

The peaks and valleys shown in Fig.5.2 (b) represent typical power consumptions for weekdays and weekends respectively. It is extremely important to analysis the historical load data to determine input-output data structure.

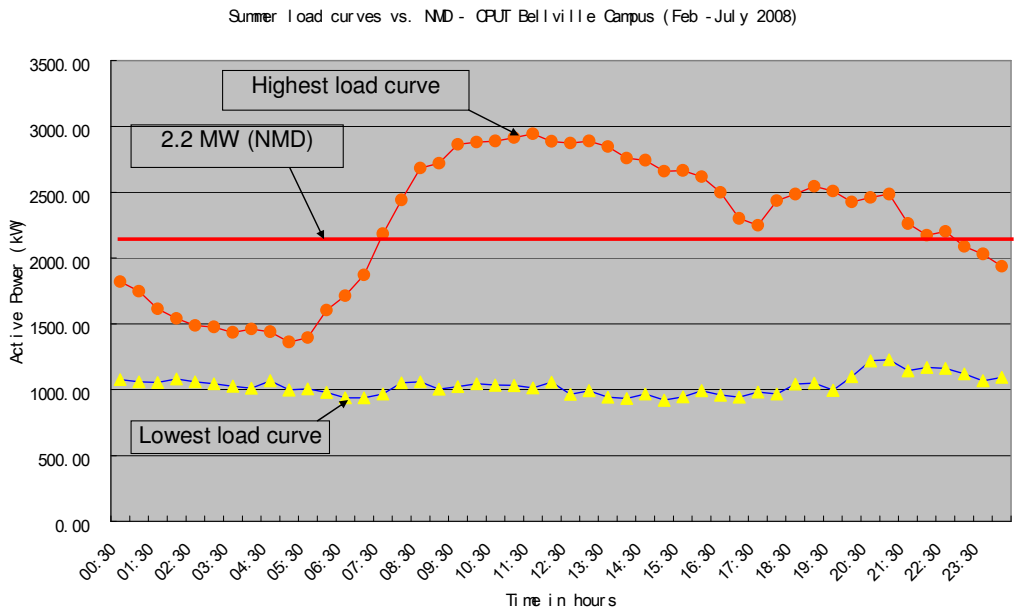


Fig 5.2 (a): Load curves (Lowest & highest recorded load patterns in summer)

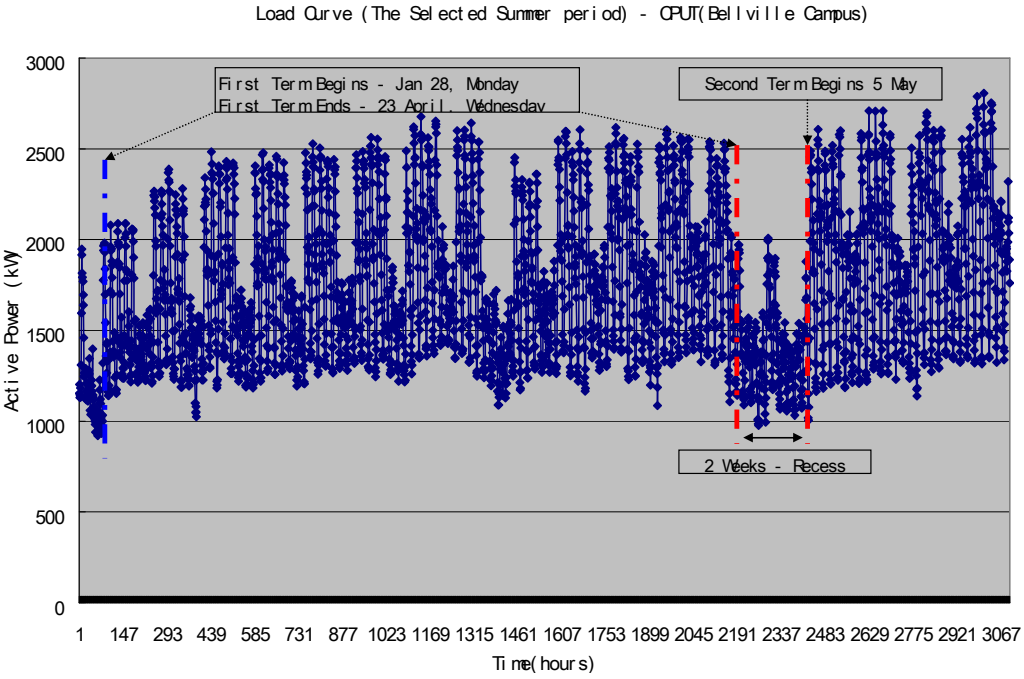


Fig 5.2 (b): Time series – load trend for the selected summer period. The first day is a Friday (during holiday).

For instance, if one uses the past load values for the weekend load pattern to predict the load for a week day, the network is likely to yield wrong results. In this work, the input-output data structure is also determined on the basis of historical load trend analysis. Fig. 5.2 (c) shows the selected past load values and modelled as network IV to forecast the load for a week ahead on hour-by-hour basis and collectively.

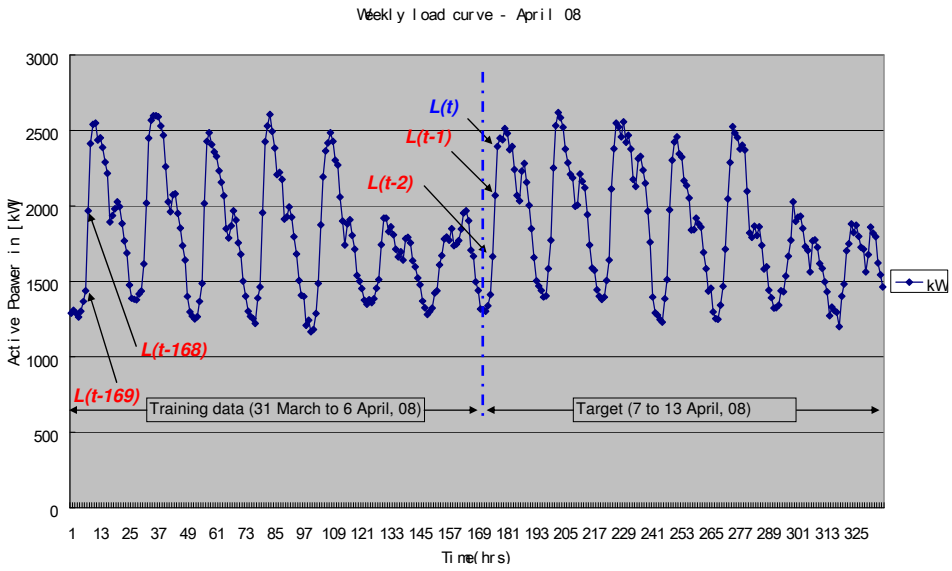


Fig. 5.2 (c): Structure of IV for the non weather sensitive model.

The total load of the campus is a sum of many departmental and residential loads. The following section introduces the descriptions of the load for the selected educational buildings and departments.

A typical load profile for the selected winter period (June – September) is shown in Figure 5.2 (d).

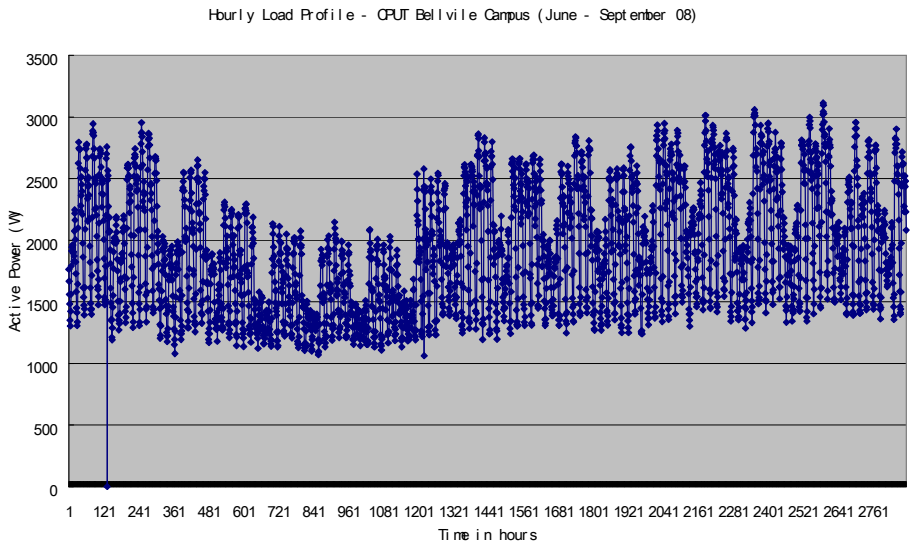


Fig. 5.2 (d): Load profile – for the selected winter period including Eskom' load rationing

5.3.2 Departmental loads

As per measurement schedule illustrated in table 5.1, individual loads for different departments were also measured. A typical daily load profile for the Administration building (for a non-

holiday period) is shown in Figure 5.3(a). The first measurement was taken on Thursday, 5th June for a period of six days.

In Figure 5.3(a) one observes that the load for reaches the peak (approx. 145 kW) at about 08h30 every week day. Usually, this department gets occupied at around 08h30 till 16:30 for weekdays. Thus the average load recorded after 16:30 indicates a reduction of approx. 50% i.e. 65 kW. As for weekend load profiles, an average 78 kW was recorded and practically speaking, this big power consumption reduction seems to be valid because the building would ideally be unoccupied on weekends.

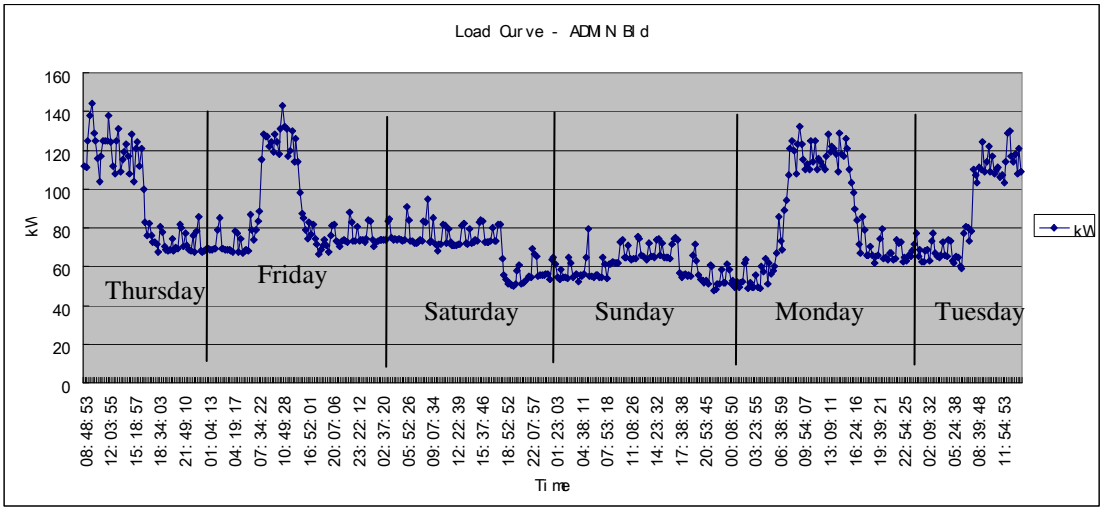


Fig. 5.3 (a): The load over the period June 02 – 15, 2008 for the Administration Building. The first day is a Friday.

As shown in figure 5.3(a), two different load patterns during day time and after hours can be clearly observed. Convincingly, the load reaches the peak during the day and then gradually reduces immediately after hours. Typical weekly load profiles for the IT, electrical engineering, and the administration building are shown in Figure 5.3 (b).

The highest peak of more than 300kW was recorded in the IT Building.

The other interesting load profile, with unique features in terms of fluctuations, was measured at the air conditioning plant (ACP). A typical load profile for the ACP is shown in Figure 5.3 (c).

A load with this type feature is extremely difficult to predict because of the lack of consistence in the load pattern. However, the IV structure of this particular load needs to be reviewed. Possibly,

instead of using the entire past load values defined in the structure of the previous IV, only most recent past load values could be considered.

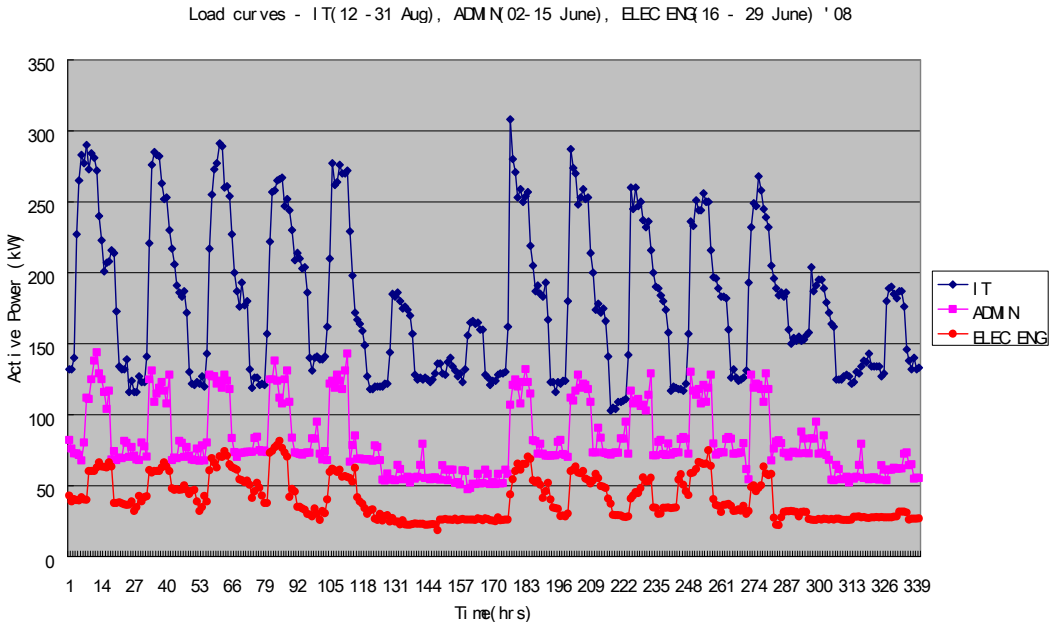


Fig. 5.3 (b): Load profiles – for departmental loads (IT, ADM, and Electrical department).

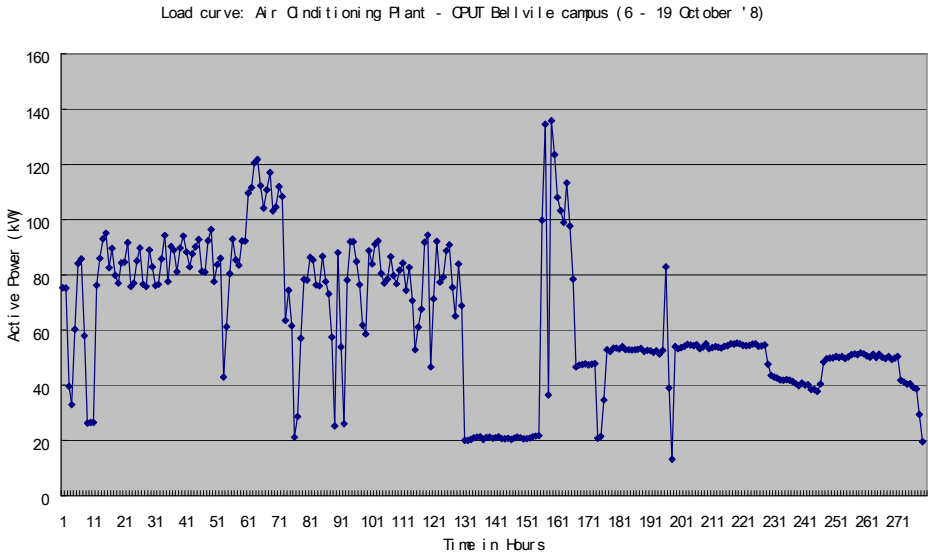


Fig. 5.3 (c): Load profile – for the Air Conditioning Plant (First day is a Monday).

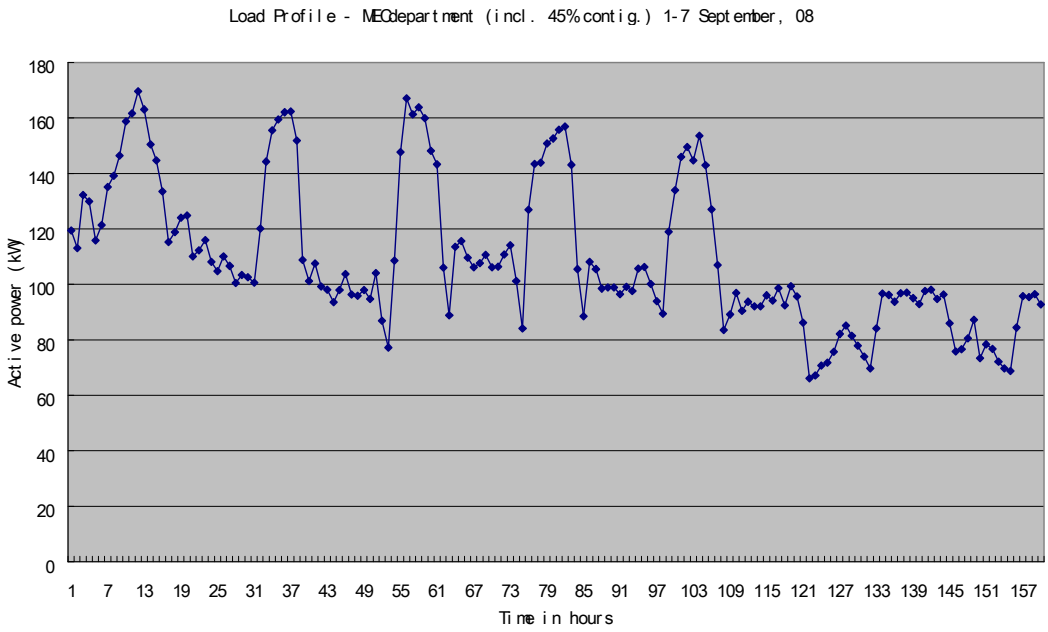


Fig. 5.3 (d): Load profile for the Mechanical Department including the surrounding dwellings.

The load profile for the Mechanical department as shown in Figure 5.3 was recorded for a week period (1st till 7th September 2008). However, some machines were not in use or operation when measurements were taken. Please refer to Appendix F. Thus it was found necessary to add a 45% load to the measured load to cater for the worse case scenarios (if machines were to be running collectively). Again the peaks and valleys shown in the profile indicate the different load patterns on weekdays and weekends. It must however be noted that the load profile shown in Figure 5.3 (d) also includes the electric energy consumed at the buildings located in the Mechanical department geographical zone (including the Craft Tech building).

5.3.3 Characteristic of the total load curve

As clearly shown in Figure 5.4, the lowest power consumption of the campus is normally at night and then the load picks up early in the morning around 07:30. This Figure indicates the load profile for Monday 17th, 2008. All other weekdays would ideally have similar load curves when the institution is fully operational. However load profiles during holidays would obviously be different from the pattern shown in Figure 5.4.

From these graphs, one learns that there is a need for pattern classification. Thus the day-type classification in accordance with their corresponding load patterns reflects more or less a precise relation with the target load. The recording sampling period is 30 minutes, thus the daily

maximum threshold ends at 48. The load profiles for Saturdays and Sundays in winter are shown in Figure 5.5 (a) and 5.5 (b) respectively.

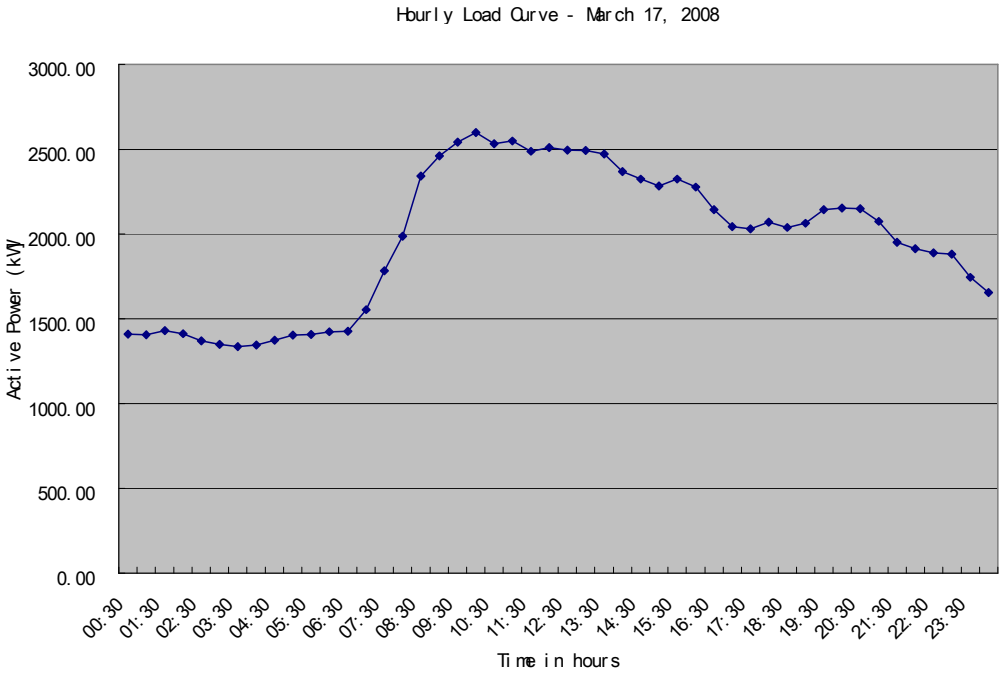


Fig 5.4: Hourly load curve –March 17, 2008 (Monday)

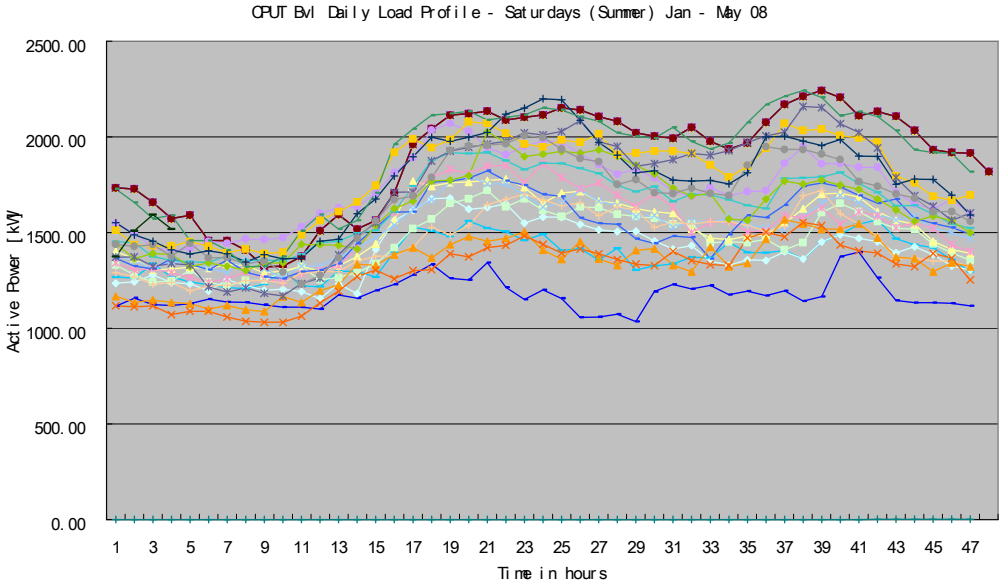


Fig. 5.5(a): Cumulative daily load curve (Saturdays in summer) - Jan to May 2008.

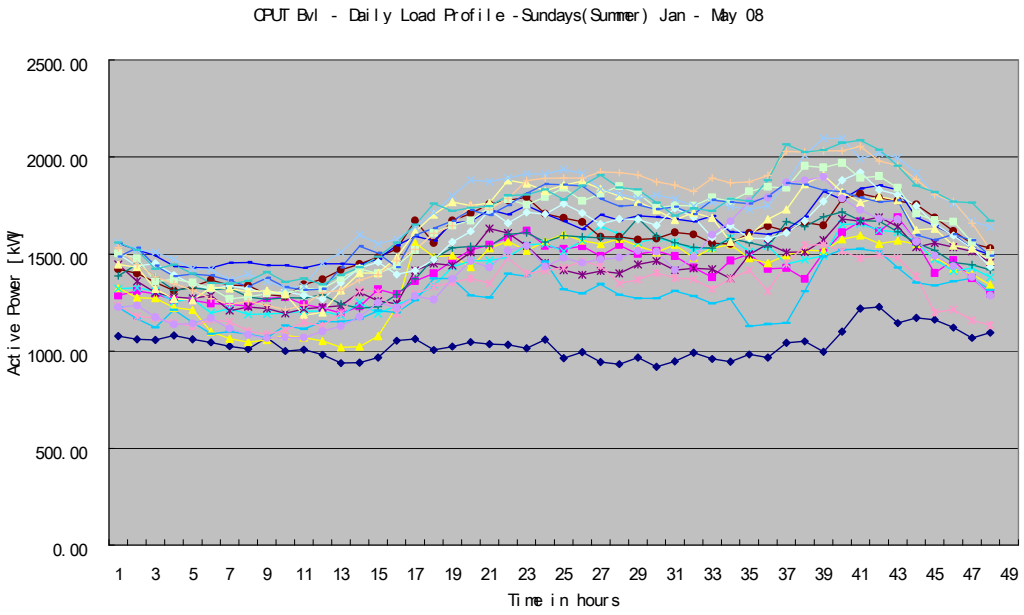


Fig. 5.5(b): Cumulative daily load curve (Sundays in summer)

The weekend load profiles show that the power consumption is the lowest on Sundays.

5.4 Correlation analysis between the load and weather data

Weather conditions influence the load greatly. Without replicating, the weather data used in this work were provided by the South African Weather Service office based in Cape Town. Thus the correlation between weather related conditions and the consumption is conclusively essential, particularly in regions where significant changes in weather conditions are experienced. The graph in figure 5.6 shows a scatter plot between the load and wind speed in summer. The other weather related variables such as daily average minimum and maximum temperatures, humidity, and wind speed were also incorporated into a local database.

The correlation analysis between climatic and load data was also performed to measure the statistical association between variables. Table 5.2 summarizes the closeness of linear relations defined by correlation coefficients. Mathematical formulation can be defined as follow: suppose, x denotes the consumption and y represents a weather variable, then the population correction between two variables x and y is given as (Shen, 1999)

$$\rho(x,y) = Co\,\mathrm{var}iance(x,y)/\{\mathrm{var}iance(x) * \mathrm{var}iance(y)\}^{1/2} \qquad (5.1)$$

Where ρ is called the product moment correction coefficient or simply the correlation coefficient.

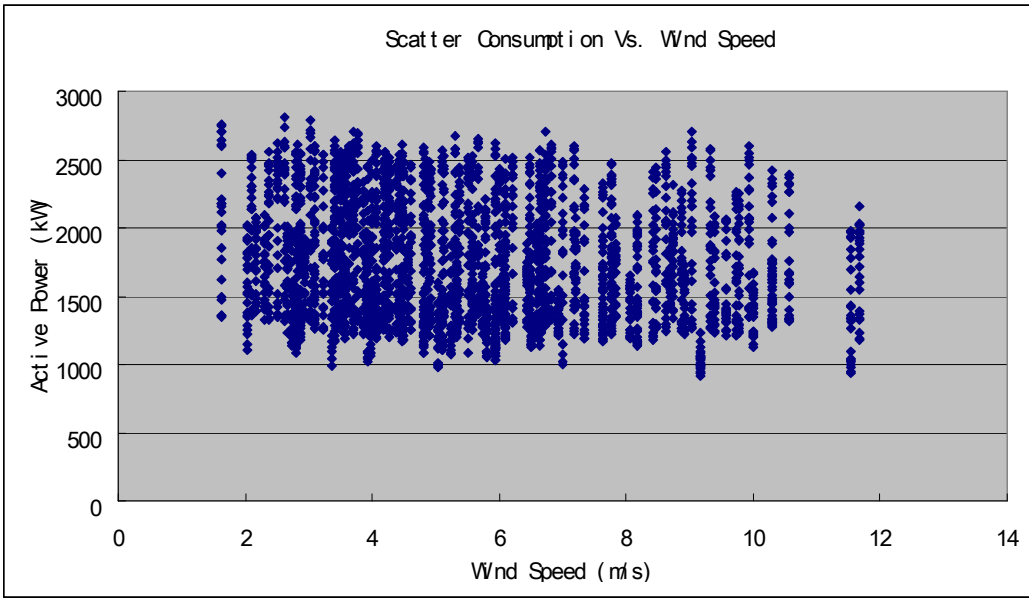


Fig 5.6: Scatter plot between wind speed and the consumption in summer.

Consumption	***Climatic data***	$\rho(x, y)$
Load	Wind speed	-0.1443
	Humidity	0.0934
	Min_temp	-0.1603
	Max_temp	-0.1631

Table 5.2: Correlation analysis between the load and exogenous variables - summer data set

The sign (+ or -) indicates the direction of the relationship. For this particular analysis, all weather variables excluding humidity indicate inverse relationships. This implies that if one variable increases the other variable decreases.

5.5 Data Storage

The proposed ANN based forecasting models will be trained off-line using past load data and weather forecast data stored in a local database. A general practice for a good STLF system is that the relevant data from database are accessed automatically by the model. For this configuration, the desired database needs to be identified, and subsequently be correctly installed in a local personal computer (PC). There are a number of freeware databases available for data storage namely: IBM, DB2, Informix, Ingres, Microsoft Access, Microsoft Excel, Microsoft SQL Server,

MySQL, Oracle, PostgreSQL etc. However, if one intends to use MATLAB Database Toolbox, the selecting of an appropriate data management system is required because this Toolbox mainly supports ODBC/JDBC protocol compliant systems. A comprehensive procedure on how to set up MySQL Database and conversion of data into required format can be found in (Kriger, 2007).

In this work, all historical data are stored in MySQL DB Manager Database and then imported into MATLAB as desired via ODBC communication protocol. The DB (MySQL) Manager Professional 3.0 software is a freeware data management system and it is primarily designed for research purposes.

Since the models need to automatically access the input data stored in the Database, some MATLAB Database Toolbox functions (Structured Query Language (SQL)) syntax are used to extract the required data from the Database into MATLAB Workspace.

Figure 5.7(a) gives a general layout of MySQL Database. MySQL Database Software allows a user to create one or more databases and tables. For the purpose of this research, 1 database ('*combinedloaddata*') was created and then configured. In this database, various table storing data for different models have been created. Figure 5.7 (b) shows some of the tables that were created in the Database. The required data were then extracted from these tables using Standard Query Language (SQL) MATLAB functions into MATLAB workspace.

5.5.1 Importing Data into MySQL Database

The DBManager can import, export structure data to and from some known databases. Depending on the format and structure of the required data, the DAO or Text import wizard can be used to import the data into MySQL Database.

The DAO wizard uses the MS DAO drivers to convert and import data from the supported files: MS-Access, MS-Excel, Dbase/Clipper/FoxPro, ODBC data sources etc. To use this wizard one must have MS Office or at least the DAO drivers installed. This import wizard often has some conversion problems, especially if one deals with data in different formats.

Unlike the DAO, the text wizard imports fixed or delimited text files into a database table. Here data are kept or imported in designated delimited columns as shown in Figure 5.7 (c). Again, depending on the format of the data and types, the suitable delimiter i.e. tab, semicolon, comma

or spaces should be selected. The text wizard method appears to be the most convenient approach particularly if one plans to import data in different formats.

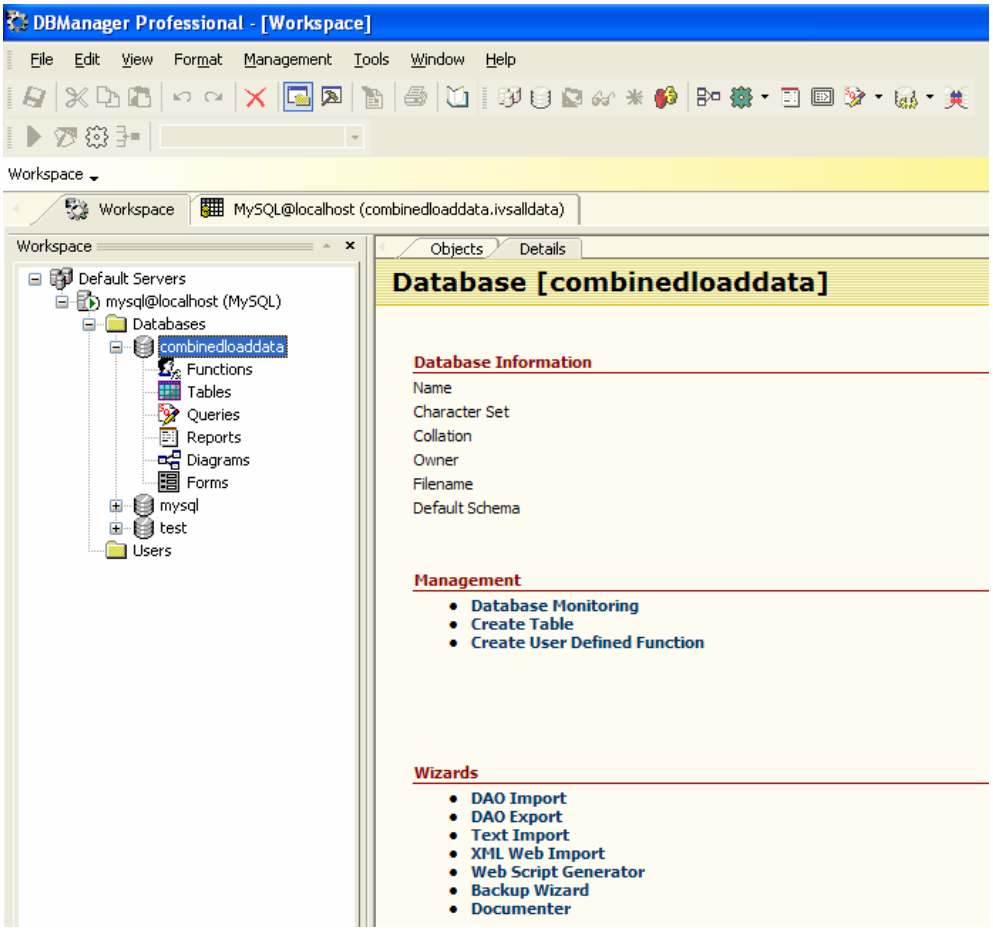


Fig. 5.7 (a): A general overview of DB Manager – Workspace (where historical load and weather data are stored).

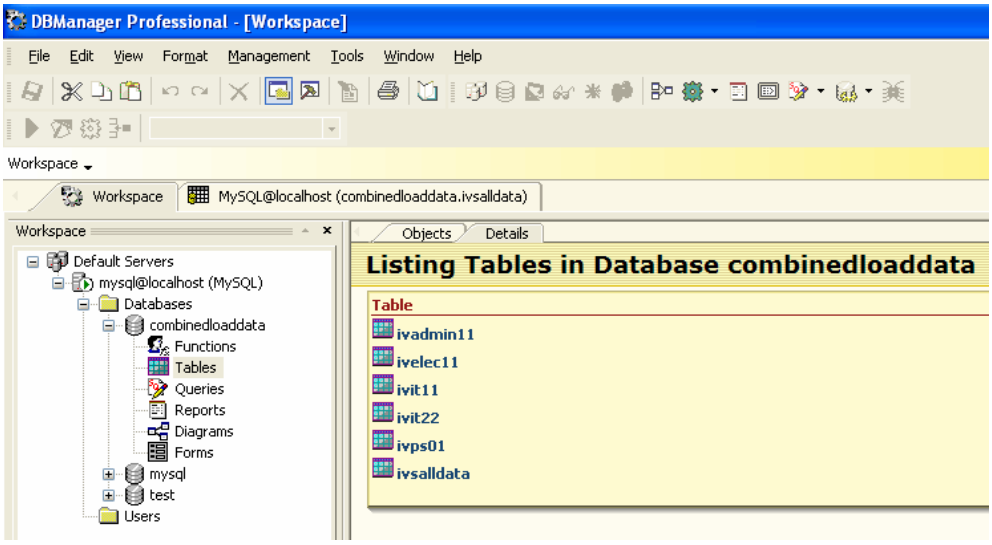


Fig. 5.7(b): A typical layout of some of the created tables.

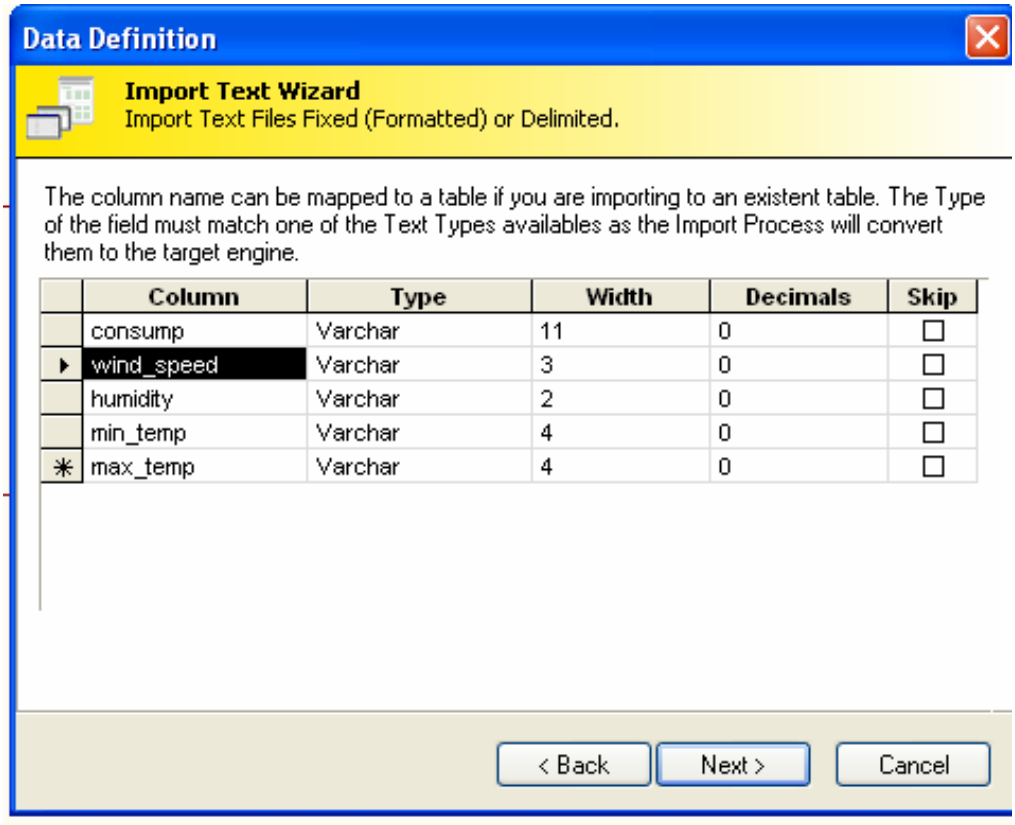


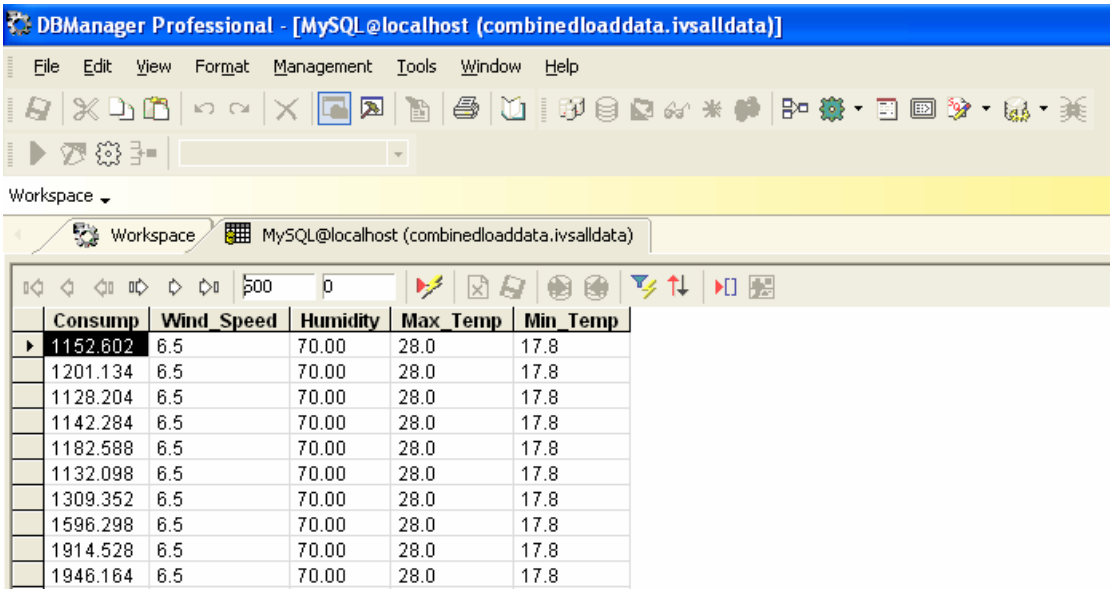
Fig. 5.7(c): Importing data from an Excel spread sheet into MySQL DB

Alternatively, the use of enterprise software for converting Excel files/MSAcess to MySQL databases is promoted. This application is designed primarily to eliminate time-consuming data entry and the errors that accompany it. The commonly-known Navicat's Database Transfer Tool Software or Excel-MySQL Converter could be used instead. Navicat is a third party product for MySQL and is the leading MySQL GUI for MySQL server management and development. For this application one often needs to procure a commercial license.

5.6 Data Pre-processing

The database stores raw or pre-processed weather and load data as shown in Figure 5.7(d). However, the data should be normalized prior to presenting them to a model for training or any forecasting attempt.

Data scaling is essential due to the fact that neural networks are often vulnerable to raw data, it's extremely important that data are scaled (typically values between 0 and 1, or -1 and 1) to avoid convergence problems. There are several methods that can be used for data normalization as defined in Equation (5.2) to (5.2).



Consump	Wind_Speed	Humidity	Max_Temp	Min_Temp
1152.602	6.5	70.00	28.0	17.8
1201.134	6.5	70.00	28.0	17.8
1128.204	6.5	70.00	28.0	17.8
1142.284	6.5	70.00	28.0	17.8
1182.588	6.5	70.00	28.0	17.8
1132.098	6.5	70.00	28.0	17.8
1309.352	6.5	70.00	28.0	17.8
1596.298	6.5	70.00	28.0	17.8
1914.528	6.5	70.00	28.0	17.8
1946.164	6.5	70.00	28.0	17.8

Fig. 5.7 (d): Raw load and weather data stored in the *"ivalldata"* Table for the IV # 1.

5.6.1 Data Scaling Methods

Data can be normalized using one of the following equations:

$$Normalized\ value = \frac{\text{actual value} - \text{minimum value}}{\text{maximum value} - \text{minimum value}} \tag{5.2}$$

$$Normalized\ value = \frac{\text{actual value}}{\text{sum of the day values}} \tag{5.3}$$

$$Normalized\ value = \frac{\text{actual value}}{\text{maximum value}} \tag{5.4}$$

$$Normalized\ value = \frac{\text{actual value} - \text{average value}}{\text{maximum value} - \text{average value}} \tag{5.5}$$

The scaling of the data is done in order to improve interpretability of network weights and the uniform scaling equalizes initially the importance of variables. In this work, Equation (5.4) has certainly been adopted and implemented to normalize both the historical load data and weather data.

5.7 Bad data detection and replacement

As perfectly defined in (Yang, 2006)'s work, the existence of bad data in historical load curve affects the precision of load forecasting result.

There are two kinds of bad data in the daily load curve: false channel bad data and abnormal event bad data. False channel bad data are due to the measurement and transmission mistakes, and often they are far from their real physical values. Abnormal event bad data come from some unexpected sudden incidents, such as short circuit and equipment overhaul, which cause unnatural sudden changes of the load curve trend.

With no exception, some bad data were detected in the total load curves as well as in the weather data. Figure 5.8 shows daily load curves for six days including valleys due to bad data. The valleys are as a result of Eskom's load management strategies to temporarily reduce cost intensive peaks by shedding "non essential loads". The impact of a load shedding process could be viewed as an absolute inconvenient power interruption method as it affects daily production activities. However, this partial power interruption helps to prevent a total power black-out.

The nature of bad data discussed in Fig 5.8 is neither false channel bad data nor abnormal event bad data type, but rather bad data as result of a planned strategy. Nonetheless, a strategy for detecting and replacing these data needs to be formulated prior to importing data into the Database.

The longest power interruption period (as shown in figure 5.8(b) because of load shedding was recorded on 16th April for a period of 2 hours in total. For more information on other load shedding recorded in summer, please refer to the records in Appendix B.

In this work, a similar technique to replace bad data, proposed in the work of (Santos *et al.*, 2005) has been adopted. The rule used for this purpose has been, briefly: if the lost or bad data period is longer than an hour, then the whole day is replaced with the average load profile of the two homologous days of the two previous weeks; else the average load of three previous hourly loads is considered instead; if the bad data corresponds to a special or holiday – the closest similar day is used.

Applying this rule to all recorded bad data as shown in Figure 5.8(a), the load profile for the 16th April was subsequently replaced with the average load profile recorded for the two homologous days of the two previous weeks (i.e. averaging the profiles for past two similar day - 26th and 19 March).

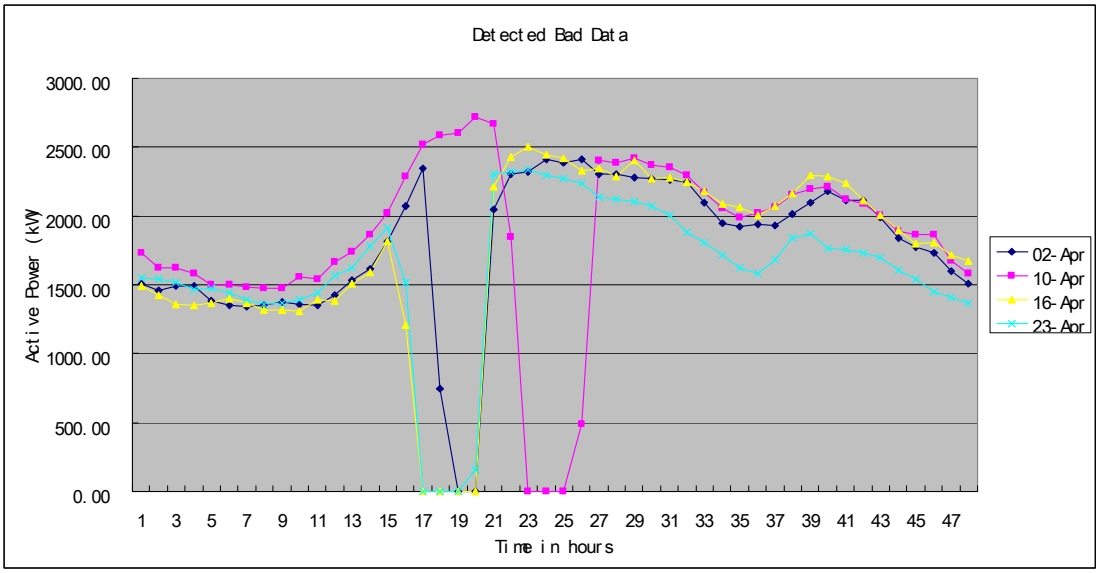


Fig. 5.8(a): The effects of bad data on the load curves resulted from Eskom's load shedding. The total supply interruption period is 7 hours.

Because the supply interruption period of other days (02 April, 10 April, and 23 April) is shorter than an hour, the average load of three previous hourly loads was taken. No bad data corresponding to a special or holiday was recorded.

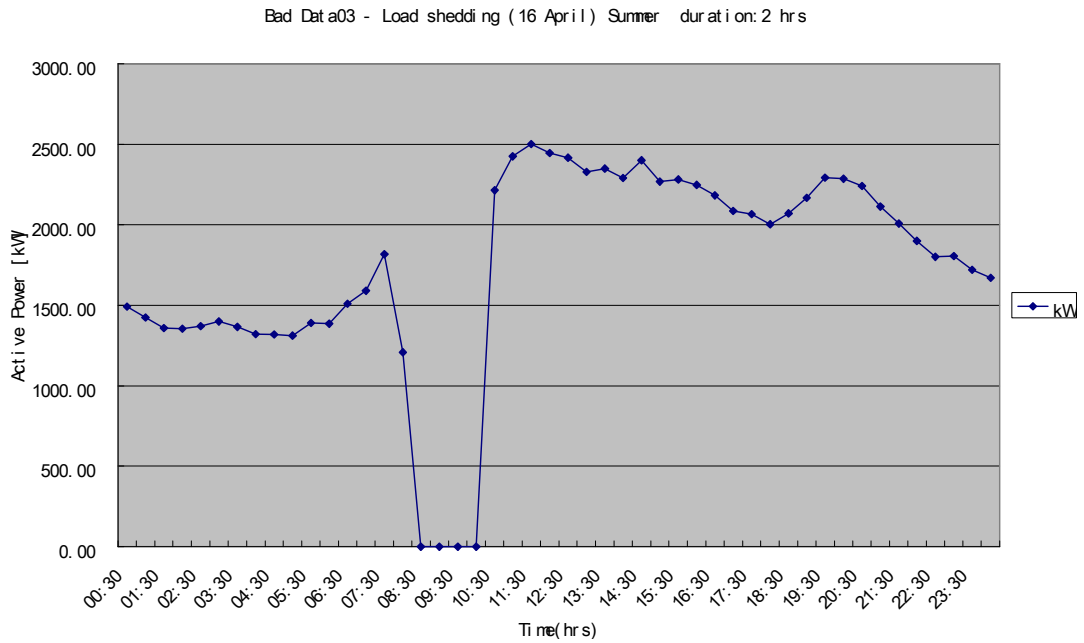


Fig 5.8 (b): The longest load shedding period - recorded in summer for the CPUT Bvl Campus.

5.8 Composition of the Input Vector (IV) of the prediction models

The structure of the input vector specifies the selected endogenous and exogenous variables. In this work, only past load values and weather related data have been used. Some times sinusoidal functions and binary codes could be presented to the network to provide the model with the cyclic behaviour of the load. (Chen, 1991) applied autocorrelation analysis on the historical load to determine the input variables and reported that loads of the same hours have very strong correlation with the load measured at the same hour and day of the previous weeks. In the work of (Monohar *et al.,* 2008) only five inputs selected from the previous day and five each from the previous weeks on the same day to predict the load of the next day.

In (Santos, *at al.,* 2008, and Hippert *et al.,* 1093), the construction of the IV structures were based on some contiguous past values of the load. The most convoluted task in time series forecasting is to determine an adequate data points for training. If data points are insufficient, the forecasting will be poor. If on the contrary, data points are useless or redundant, modeling will be difficult or even skewed (Lendasse *et al.*, 2000).

In this work, two forecasting approaches were considered, namely: first, forecasting the load for a whole week at once (batch) was evaluated, and then hourly (incremental) forecasting was studied. In both cases, the inclusion of weather data was considered.

The IV for the weather sensitive model is:

$$[Input\_Vector\#1] = L(t-1); L(t-2); L(t-168); L(t-169); ave\_wind\_speed(W(t)); \\ ave\_humidity(H(t)); \max\_temp(T_{\max}(t)); \min\_temp(T_{\min}(t)) \quad (5.6)$$

The predicted output is a nonlinear function of the IV # 1 and is defined as:

$$\hat{L}(t) = f\left(L(t-1); L(t-2); L(t-168); L(t-169); (T_{\min}(t); T_{man}(t); W(t); H(t)\right) \quad (5.7)$$

where $W(t)$ denotes daily average wind speed, $H(t)$ is the daily average humidity, $T_{\min}(t)$ and $T_{\max}(t)$ represent daily maximum and minimum temperatures on the forecast day.

The IV and the predicted output for the non-weather sensitive model are:

$$[Input\_Vector\#2] = L(t-1); L(t-2); L(t-168); L(t-169) \quad (5.8)$$

$$\hat{L}(t) = f\left(L(t-1); L(t-2); L(t-168); L(t-169);\right) \quad (5.9)$$

The IV structure used in this work, as discussed earlier, consists of two previous hours active power values, *L(t-1)* and *L(t-2)* and some homologous consumption past load values of the previous week *L(t-168)* and *L(t-169)*. The inclusion of these values provides information regarding the consumption trend in the past homologous periods (Santos, *at al.,* 2008). These inputs are presented to the network to predict the load $\hat{L}(t)$.The structure of IV # 1 and 2 are given in Figure 5.9 and 5.10 respectively.

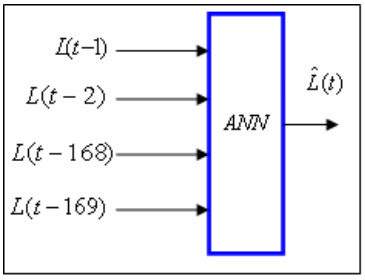


Fig. 5.9: Composition of the input vector (IV) # 2 (non-weather sensitive model)

For weather sensitive models, the structure of the input vector is slightly modified to take into account the effect of weather conditions on the load as illustrated in Figure 5.10.

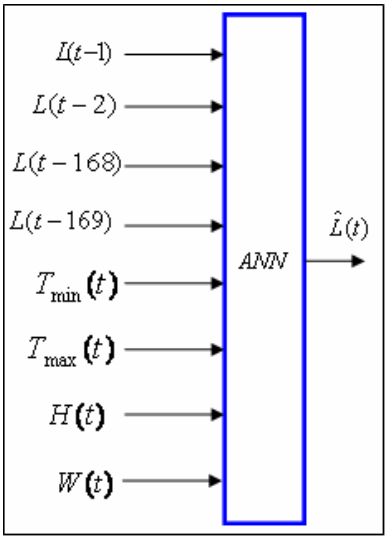


Fig. 5.10: Structure of the IV # 1.

In this work, the period chosen for load memory series analysis was defined from February to May 2008. However only 1 week past load values were used in addition to the weather-related data point, producing a set of N = 1352 points. The load data were taken every hour for a period of one week. The training data was defined from Monday, 31st March till 6th April 2008 and the corresponding target was defined for the period from 7th to 13th April 2008. The models were also

evaluated for generalization capability, thus testing data set was defined from the first week of March (3rd to 9th). Figure 4 shows weekly load profile from which the training data and target were derived.

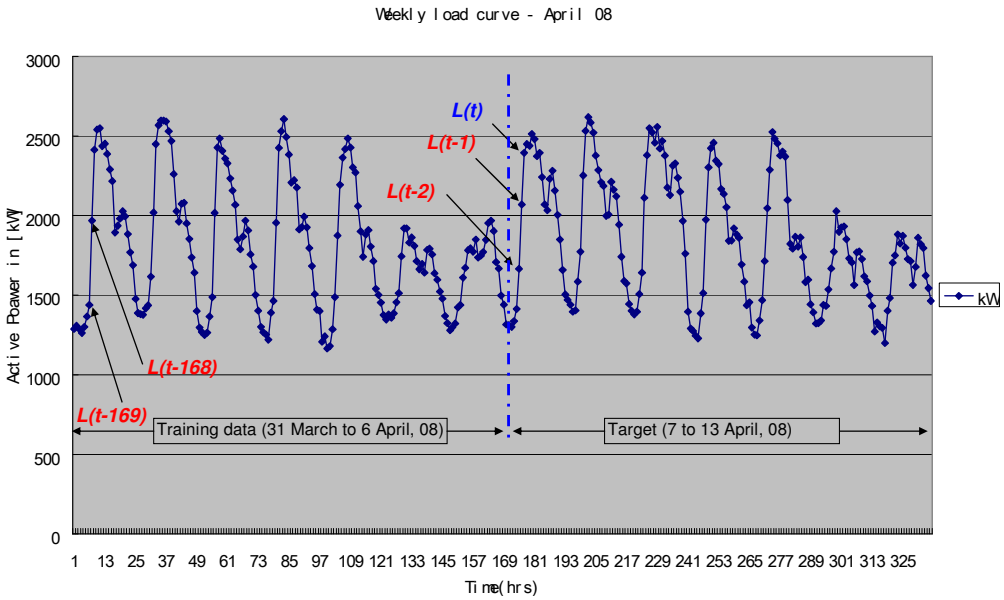


Fig. 5.11: Two weeks load profile for the CPUT Bellville Campus in summer (31st March to 13th April 2008)

These models were also evaluated for generalization, thus the testing data were selected from the first week of March (3rd to 9th).

More network architectural details on the developed models are given in the following chapter, where a fully-fledged application of ANN in load forecasting is introduced.

5.9 Conclusion

This chapter discusses the methodologies used to gather the required data, the descriptions of these data, and some results from the correlation analysis between the load and exogenous variables. In addition, the chapter also gives some details with regards to the storage of input data and results. Some detected bad data and subsequent replacement strategies are also highlighted. The other important topic covered in this chapter is the composition of the input vectors (IVs) of the proposed models.

The context of the previous chapters has explicitly covered some good introduction to a load forecasting problem. Now it is time to experiment. Thus the following chapter will discuss real application of ANN in STLF.

CHAPTER SIX

APPLICATION OF ANNs IN LOAD FORECASTING

This chapter presents a comprehensive approach to power system load forecasting using Artificial Neural Network methodologies. Furthermore, the chapter covers some important steps that need to be followed prior to developing a forecasting model. And the last part of this chapter deals with the network structures, input vectors and various STLF models developed in this work.

6.1 ANNs for load forecasting

As highlighted in the previous chapters, there are different types of neural networks, namely: radial basic function, feedforward, recurrent, dynamic, Hopfield, perceptron, vector quantization, unsupervised. Some networks perform better than others in specific applications. For function approximation, classifications, and for dynamic system modelling: radial basis function, feedforward, and recurrent are dominant. Unsupervised networks are normally used for clustering. The structure of *feedforward* and *radial basis functions* neural network types can be modified to suit a specific problem. For example, the neuron activation function can be changed to some other suitable functions.

The basic idea of ANN in load forecasting is that the load can be observed as a non-linear function influenced by a number of factors, i.e. historical load, weather-related, time factors etc. The relationship between the load and these factors is often difficult to model using conventional methods. ANNs can map complex, non-linear relations, thus many researchers and planners frequently use these methodologies in power system load forecasting (Mandal *et al.*, 2005).

The weather-related factors (daily maximum temperature, minimum temperature, humidity, and wind speed on the forecast day) were selected and then modelled as network inputs. The selection of network parameters is made rather arbitrary. However, presenting a neural network with less significant inputs may result in convergence problems and subsequently affect the performance of the network.

6.2 Designing an Input Vector (IV) for a STLF model

The accuracy of the forecast relies on the configuration of the model's input vector. Thus the correlation analysis between the load and exogenous variables was carried prior to selecting

appropriate input variables. The following section talks about how the change in weather conditions and the load can be modelled mathematically.

6.2.1 Variation in Weather-Related Inputs

The change in load and weather variables (temperatures, wind speed, and humidity) can be described as follow:

$$\begin{aligned}
&\Delta L(t) = L(t) - L(t-1) \\
&\Delta T_{\min}(t) = T_{\min}(t) - T_{\min}(t-1) \\
&\Delta T_{\max}(t) = T_{\max}(t) - T_{\max}(t-1) \\
&\Delta H(t) = C(t) - C(t-1) \\
&\Delta W(t) = W(t) - W(t-1)
\end{aligned} \tag{6.1}$$

Where $L(t)$ is the load at time t, $L(t-1)$ is the load of the previous hour, $T_{\min}(t)$ is the minimum temperature at time t, $T_{\min}(t-1)$ is the minimum temperature of the previous hour, $T_{\max}(t)$ is the maximum temperature at time t, $T_{\max}(t-1)$ is the maximum temperature of the previous hour, $H(t)$ is the humidity at time t, $H(t-1)$ is the humidity of the previous hour, $W(t)$ is the wind speed at time t, $W(t-1)$ is the wind speed of the previous hour.

Generally, weather variables are considered to be key inputs especially for short-term load forecasting and often small errors in weather data could affect the prediction.

6.3 Development of a load forecasting ANN-based Model

Load forecasting could be viewed as a non-linear function approximation problem. Most forecasting models reported in literature, were subjected to different training approaches, and subjected to different data quality (Sharif *et al.,* 2000, Medal *et al.,* 2006, Lauret *et al.,* and Srinivasan, 1998). However, it's extremely important, especially for intelligent systems, to undertake some crucial model development steps. The flow chart in figure 6.1 attempts to illustrate some essential model developmental steps. The first one involves the type of forecast and data collection. Here one needs to define the system requirements in terms of forecast time horizon, and then gather the required data. Thereafter, a correlation analysis between variables and the load, data preparation and processing should be carried out. The latter step covers details with regards to network architecture and suitable model parameters i.e. defining type and size of network, learning paradigm, network performance function etc.

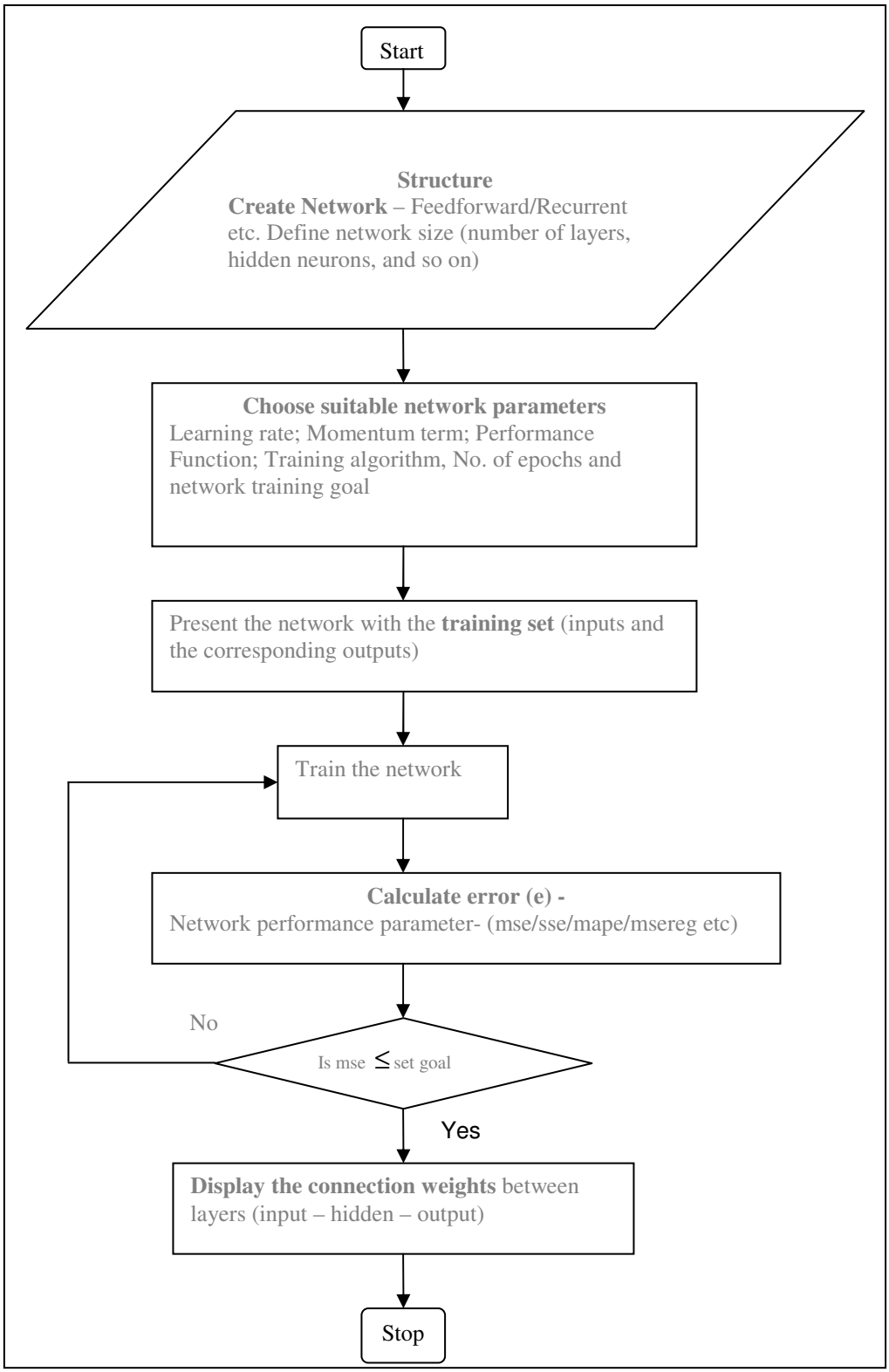


Fig. 6.1: Flow chart for development of a supervised ANN-based model

6.3.1 Forecasting using a Feedforward (FF) Neural Network

The back propagation multi-layered perceptron (MLP) type neural network is most reported and suitable network structure for STLF (Dillion *et al.,* 1991, Lee, 1992 and Djukanvic, 1993). A multilayer perceptron is a family of feedforward neural networks that maps sets of input data onto a set of appropriate output. These networks use three or more layers of neurons with nonlinear activation functions. The superiority of this type of networks, in contrast to the conventional methods, is that it can distinguish data that are not linearly separable and thus it is commonly used in STLF.

The multilayer perceptron consists of an input and an output layer with one or more hidden layers of nonlinearly-activating neurons.

Each neuron in one layer connects with a certain weight to every other neuron in the following layer. In this work, only the Feedforward neural network is considered. However, different sub-models (weather sensitive and non-weather sensitive models) were developed to evaluate different cases. The topology of the Feedforward neural MLP network used in this work excluding biases is shown in Figure 6.2.

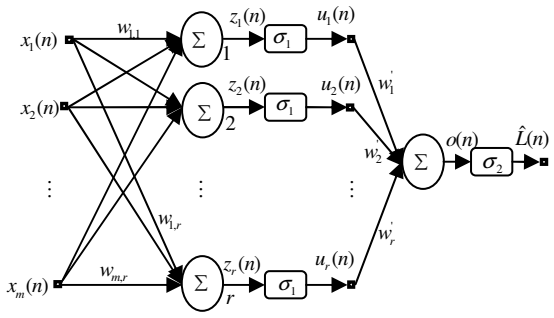


Fig. 6.2: Architecture of the feedforward neural network

where w denotes a vector of the synaptic weights, x and u are vectors of the inputs to the layers, m is the number of input variables, and r is the number of neurons in the hidden layer.

The weighted sums for the hidden and the output layers are:

$$z_k(n) = \sum_{l=1}^{m} w_{l,k} x_l \tag{6.6}$$

and

$$o(n) = \sum_{k=1}^{r} w_k' u_k(n) \tag{6.7}$$

where $k = \overline{1, r}$ and $n = \overline{1, N}$, where N is the number of data points used for training of the model.

The outputs of the neurons in the hidden layer and output layer are computed by passing the weighted sum of inputs through the tan sigmoid and pure linear transfer functions respectively.

Mathematically, the outputs of the hidden layer and the output layer can be defined as:

$$\sigma_1(z_k(n)) = \frac{1}{1 + e^{-z_k(n)}} = u_k(n) \tag{6.8}$$

$$\sigma_2 o(n) = K * o(n) = \hat{L}(n) \tag{6.9}$$

Where K is a coefficient of the pure linear transfer function.

If the desired output is not attained, the network can reiteratively be trained until it yields reasonable results. However, over-training should be avoided to ensure network generalisation. By default MATLAB trains with Lavenberg-Marquardt (LM) training algorithm. Thus any different training algorithm needs to be specified prior training. During the training, network parameters (weights and biases) are iteratively adjusted on the bases of the derivatives of the error w.r.t. The derivative is then multiplied by a step size or learning rate. For a standard gradient algorithm the learning rate is held constant throughout the training, and this is often problematic for network optimalization. Thus a training algorithm with varying learning rate is considered appropriate as an alternative.

Mathematically, the approach for a standard gradient based search algorithm can be formulated as:

$$\partial w_{ij} = \eta \frac{\partial \varepsilon}{\partial w_{ij}} \tag{6.10}$$

$$\partial w_j = \eta \frac{\partial \varepsilon}{\partial w_j} \tag{6.11}$$

Where w_{ij} is the weight between the input layer and the hidden layer, w_j is the weight between the output layer and the hidden layer, ε is the network error, and η is the learning rate.

Contrary to the network topology, a neural network can have some interconnections and feedback paths between layers. This type of configuration is known as Recurrent Network. The following section discusses the architecture of the Elman Recurrent Neural Network designed for the prediction model.

6.3.2 Forecasting using an Elman Recurrent (ER) Neural Network

The Elman network commonly, is a two back propagation network with feedback (delay) connections as shown in Figure 6.3. The Elman prediction model is designed based on this network topology.

Custom ER networks can be configured to have internal connections as desired. Unlike the conventional MLP networks, the first layer of this Elman network has recurrent connections. The delay $(D^i \; i = 1,2,...,r)$ in this connection stores values from the previous time step, which can be used in the current step. The same transfer functions used in the Feedforward prediction model were also used here. This implies the tag sigmoid (σ_1) in the hidden layer and pure linear (σ_2) activation function in the output layer. This configuration is special in that two-layer networks with these transfer functions can approximate any function with discontinuities.

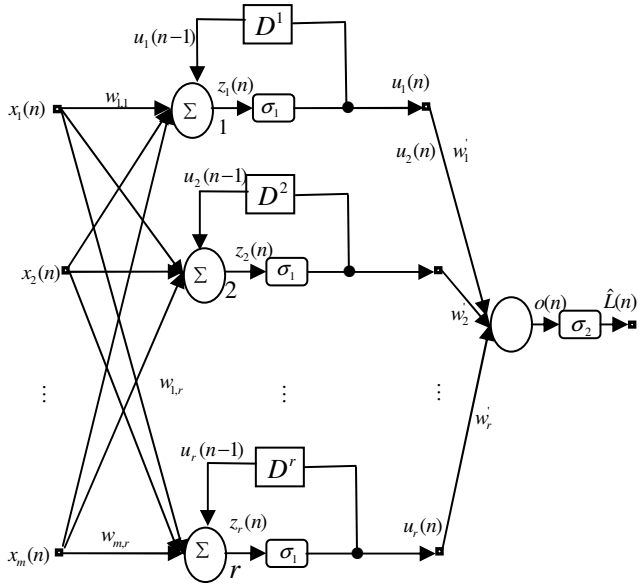


Fig. 6.3: Elman recurrent neural network topology

where w denotes a vector of the synaptic weights, x and u are vectors of the inputs to the layers, m is the number of input variables, and r is the number of neurons in the hidden layer.

The weighted sums for the hidden and the output layers are:

$$z_k(n) = \sum_{l=1}^{m} w_{l,k} x_l + w_{D^k} u_k(n-1) \tag{6.12}$$

and

$$o(n) = \sum_{k=1}^{r} w_k^{'} u_k(n) \tag{6.13}$$

where $k = \overline{1,r}$ and $n = \overline{1,N}$, where N is the number of data points used for training of the model.

The outputs of the neurons in the hidden layer and output layer are computed by passing the weighted sum of inputs through the tan sigmoid and pure linear transfer functions respectively. Mathematically, the outputs of the hidden layer and the output layer can be defined as:

$$\sigma_1(z_k(n)) = \frac{1}{1+e^{-z_k(n)}} = u_k(n) \tag{6.14}$$

$$\sigma_2 o(n) = K * o(n) = \hat{L}(n) \tag{6.15}$$

Where K is a coefficient of the pure linear transfer function.

Other training parameter that was considered is the momentum factor as an attempt to prevent the network to get stuck in a shallow local minimum. Equation (6.16) shows how the synoptic weights are adjusted. The network determines the value of the increment ($\Delta w(n)$ on the basis of the previous value of the increment.

$$\Delta w(n) = \frac{\partial \varepsilon(n)}{\partial w} + \psi \Delta w(n) \tag{6.16}$$

The effect of the momentum factor to the performance of the algorithm is discussed in (Phansalkar, 1994). The other important training parameter is the learning rate which controls the amount of change imposed on connection weights during training and to provide faster convergence also needs to be defined.

Mathematically, the weights are updated using the equation:

$$w(n+1) = w(n) - \eta \Delta w(n) \tag{6.17}$$

Where (η) is the learning rate.

A comparison of the accuracy of neural network using different learning algorithms and momentum factors is explicitly discussed in (Ho, 1992).

Finally, the amount of change in weights of the network topology illustrated in Fig. 6.2 for the hidden and output layers are defined as:

$$\Delta w_{ij}(n) = [L(n) - \hat{L}(n)]\hat{L}(n)[1-\hat{L}(n)] * u_i(n)[1-u_i(n)]w_i'(n)x_j(n) + \psi \Delta w_{ij}(n-1) \qquad i = 1,...,r,\ j = 1,...,m \tag{6.18}$$

and

$$\Delta w_i'(n) = [L(n) - \hat{L}(n)]\hat{L}(n)[1 - \hat{L}(n)] * u_i(n) + \psi \Delta w_i'(n-1) \quad i = 1,..., r \tag{6.19}$$

6.4 Description of the evolved forecasting models

In this work, a number of ANN-based models have been developed and presented with actual load data to forecast the load a week in advance. Firstly, two Neural Network models, Feedforward (FF) and Elman Recurrent(ER), were developed and then applied to the total load of the campus. Some of the evolved models are non-weather sensitive, but the effect of climatic conditions on load has also been analyzed in this work. Only two seasons (summer and winter) have been considered for the total load.

Secondly, the same models were applied to the other sub-loads of various departments. In Table 6.1 a list of different models as well as the associated IVs for the two seasons is given.

Model	**Season**	**Input Vector (VI)**
FFNN_TL_NWS01	Summer (February – May 2008)	Past Contiguous Consumption Values (PCCV): $L(t-1); L(t-2); L(t-168); L(t-169)$
ERNN_TL_NW01		PCCV
FFNN_TL_WS01		PCCV; Wind speed; Humidity; Max Temp; Min Temp
ERNN_TL_WS01		PCCV; Wind speed; Humidity; Max Temp; Min Temp of the forecast day
FFNN_TL_NWS02	Winter (June – September 2008)	PCCV
ERNN_TL_NWS02		PCCV
FFNN_TL_WS02		PCCV; Wind speed; Humidity; Max Temp; Min Temp of the forecast day
ERNN_TL_WS02		PCCV; Wind speed; Humidity; Max Temp; Min Temp of the forecast day
***Note:** model number 01 – is for the summer model and model # 02 represents a winter model*		

Table. 6.1: Developed forecasting models and the corresponding IV for total load prediction.

Without replicating, two forecasting approaches were considered, namely: first, forecasting the load for a whole week at once (batch) was evaluated, and then hourly (incremental) forecasting was studied. In both cases, the inclusion of weather data was considered. The developed models for forecasting the total load and departmental loads were bases of these approaches.

For network performance comparison purposes, it was found necessary that the configuration of both networks remains unchanged in both cases. The network structure for the non-weather sensitive model was defined as follows: (4-10-1) i.e. 4 inputs for past contiguous consumption instances, 10 hidden layer neurons, 1 output neuron for hour by hour forecast. Similarly, the latter network

structure was slightly modified to include the effects of the weather related variables. Thus the network structure was then changed to (8-10-1) - as shown in figure 6.4.

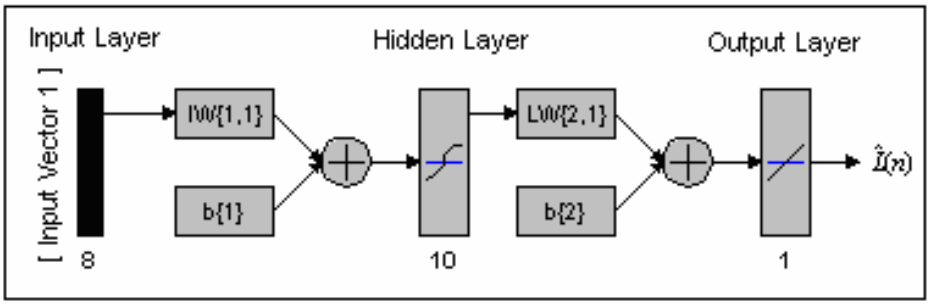


Fig. 6.4: Network architecture for the weather sensitive load forecasting model.

This network uses a tan sigmoid and pure linear transfer functions in the hidden layer and output layer respectively.

The mathematical models of the input vectors are defined as in Equation (5.6), (5.7), (5.8) and (5.9) in chapter 5

The other forecasting models for predicting some departmental loads have also been developed with the similar network configuration. However, the hidden layer neurons for neural networks for predicting departmental loads were reduced to 5. Table 6.2 shows a list of the other 10 sub-models, developed to forecast departmental loads.

The following departments or units were considered: Information Technology (IT), Mechanical, and Electrical Engineering departments, Air Conditioning Plant (ACP), and the Administration building. The measurements were taken at every department/building for a period of two to three weeks with the exception of the Mechanical department load data which were recorded for only a week period.

Sub Models	**Department/ Unit**	**IV structure**
FFNN_IT ERNN_IT	Information Technology	*Input _ Vector#* 1
FFNN_ADM ERNN_ADM	Administration	*Input _ Vector#* 1
FFNN_MEC ERNN_MEC	Mechanical	$L(t-1)$ *and* $L(t-2)$
FFNN_ELEC ERNN_ELEC	Electrical	*Input _ Vector#* 1
FFNN_ACP ERNN_ACP	Air Conditioning Plant (ACP)	$L(t-1)$ *and* $L(t-2)$

Table 6.2: Schedule of the developed departmental load forecasting models

6.5 Conclusion

The application of Artificial Neural Network in STLF and different developed forecasting models are described in this chapter. The chapter also covers some important steps to be considered when developing a supervised ANN model for function approximation problems. Moreover, the architecture and mathematical modelling of the selected neural networks have also been presented in this chapter.

And now that the ANN approach to electric load forecasting has been clearly demonstrated, the following chapter (*chapter 7*) finally discusses the results obtained from different developed forecasting models and some specific findings.

CHAPTER SEVEN

RESULTS AND DISCUSSIONS

This chapter presents the simulation results of various forecasting models discussed in the previous chapter. Few discussions pertaining specific forecasting results have also been incorporated in this chapter. The first part of this chapter talks about forecasts for the combined load and the latter part encapsulates departmental forecasts. Responses from the two forecasting approaches (i.e. batch and incremental training modes) have also been clearly indicated. Results emerged from non-weather sensitive models are also included and discussed.

Preamble

The models discussed in chapter 4 were subsequently trained under MATLAB®, *ver.* 7.1, environment using Neural Network Toolbox, *ver.* 4.0.6. And this chapter now presents forecasts emerged from these forecasting models. Comparisons between the developed weather sensitive and non-weather sensitive models are also included. Another important component of this chapter to note is that, these models were simulated with the same network architectural layout. Thus no modification in terms of network parameters and structure of these models were made. Both input vector #1 and # 2 (as specified in the previous chapter), were individually and collectively presented to the models, and simulated.

Firstly, the models were trained by applying all the inputs to the network before the weights and biases are updated. This approach is also known as *"batch training mode"*. And secondly, the models were trained using the incremental training mode (i.e. the weights and biases of the network are iteratively adjusted every time an input is applied to the network).

In addition, the following performance measure functions were employed: mean squared error *(MSE)* and mean percentage error *(MPE)* to evaluate the performance of the models. The actual error in kW is also compared.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}[(L_{actual}\ (n) - \hat{L}_{predicted}\ (n))]^2 \qquad (7.1)$$

$$MPE = \sum_{i=1}^{N}[(L_{actual}\,(n) - \hat{L}_{predicted}\,(n))/\,L_{actual}\,(n)]/\,N \qquad (7.2)$$

Where $L_{actual}\ (n)$ is the actual load, $\hat{L}_{actual}\,(n)$ is the forecast value of the load, and N is the number of data points.

7.1 Network architectural design for the prediction models

The network parameters for the models are given in the table below. The network comprises 10 hidden layer neurons and 1 output neuron. For activations, a tan sigmoid transfer function was used for the hidden layer and pure linear one in the output layer. Both momentum factor and learning rate were held constant throughout the training at 0.75 and 0.35 respectively.

Network	Hidden Layer	Hidden neurons	Output nodes	Training function	Transfer functions	Momentum factor	Learning rate
Feedforward and Elman Recurrent	1	10	1	Gradient descent algorithm	Tan sigmoid and Pure linear	0.75	0.3

Table 7.1: Network Topology for the total load models

The parameters shown in the table 7.1 defines the ANN topology. All models were firstly trained using this arbitrary designed structure.

7.2 Forecasting the total load using the non-weather sensitive FF model

The FFNN_TL_NWS01 model, as discussed in chapter six, was specifically designed to ignore any climatic related influences on the load. Thus the model merely uses past contiguous load values (input vector # 2) to predict hourly loads for at least a week ahead. The input vector was firstly applied to a Feedforward Neural Network and eventually the model was trained using summer training data. In Figure 7.1, one observes an example of some summer forecasts produced by this Feedforward forecasting model. This model is trained using Scaled Conjugate Gradient backpropagation algorithm and the actual error presented in Figure 7.2 (b). The performance of the network is shown in Figure 7.3 (c). The same procedures were then repeated for an Elman Recurrent Network based model (ERNN_TL_NWS01) in both seasons.

The forecasts produced by both networks (Feedforward and Elman Recurrent) are approximately the same and thus the results not further discussed here. The reader is however encouraged to view the records in Appendix E or forecasts emerged from Elman Recurrent based forecasting model (non-weather sensitive) as well as other application results and error comparative analysis.

The network response (forecasts) for the Feedforward network using batch training mode is shown in Figure 7.1 (a). In the same Figure, one notices that the first peak for the CPUT Bellville Campus

normally occurs at about 08:00 to 09:00 and the latter peak in the evening from 19:00 to 21:00. This "*m-shaped*" daily load characteristic confirms the non-linear features of the load and often affects the accuracy negatively. The "*m-shaped*" load feature in essence symbolises daily peaks, namely: morning and evening peaks.

The highest error (more than 10%) was recorded on Wednesday evening. Otherwise an average error of $\pm 100\ kW$ was maintained throughout the forecast lead time. Figure 7.1 (c) shows the performance of the model. It can be seen that the network has successfully failed to meet the test goal *(1e-3)* after 1000 iterations.

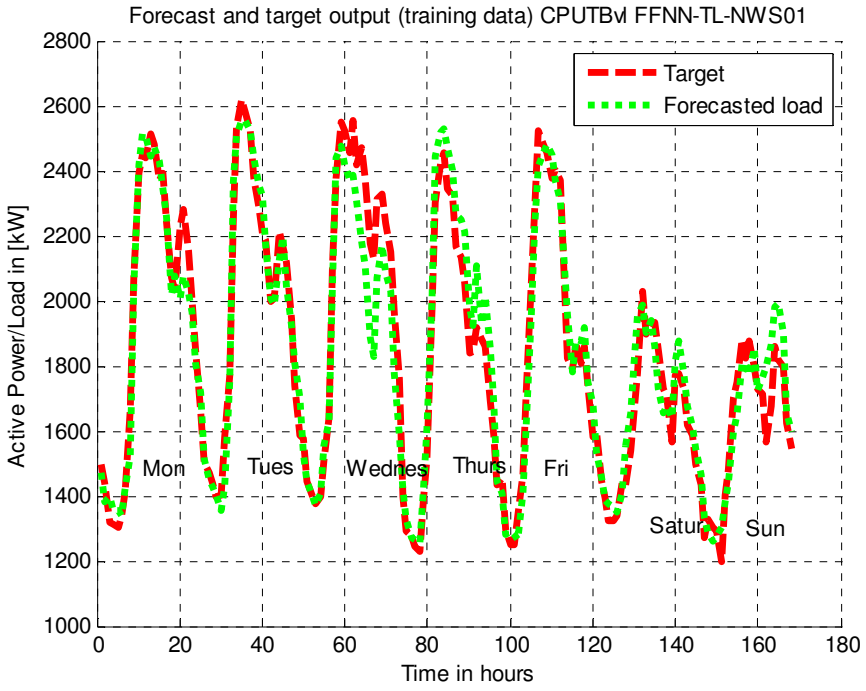


Fig 7.1 (a): De-normalized forecasts using the Feedforward Neural Network (non-weather sensitive model) for the summer period.

7.2.1 Simulating the network using testing data

To evaluate generalization capability of the network, the model was again re-simulated using the data it has not seen before. An example of the response of the network for a non weather sensitive model is shown in Figure 7.2. Despite the fact that the network has not met the goal, it can be seen that its generalisation is surely not very bad as shown in Figure 7.2.

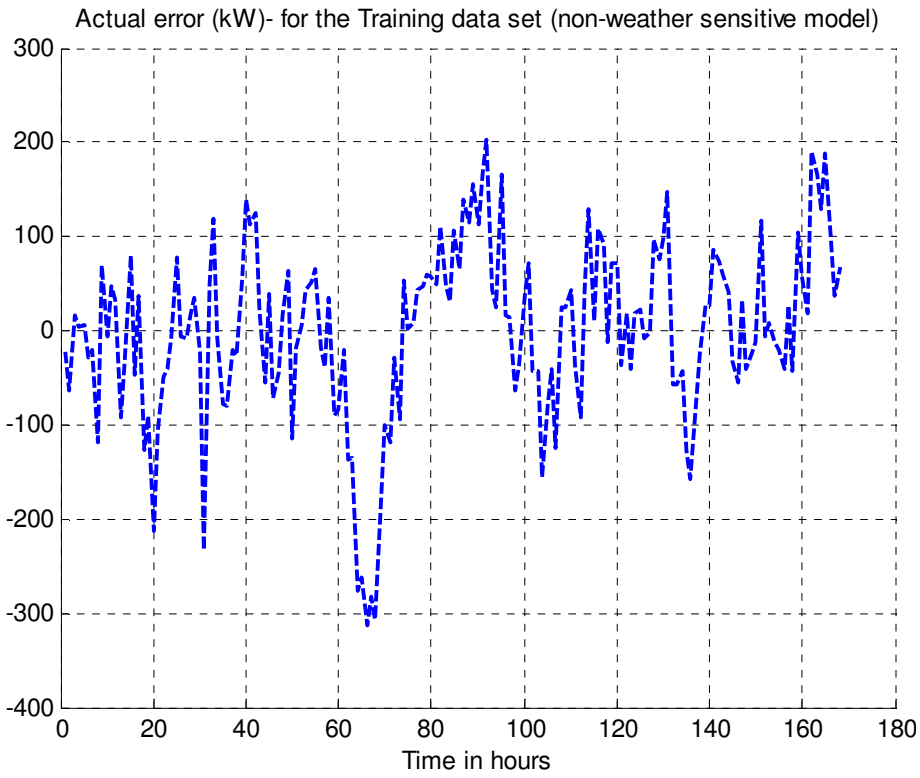


Fig. 7.1 (b): The associated actual error (kW) for the FFNN_TL_NWS01 model

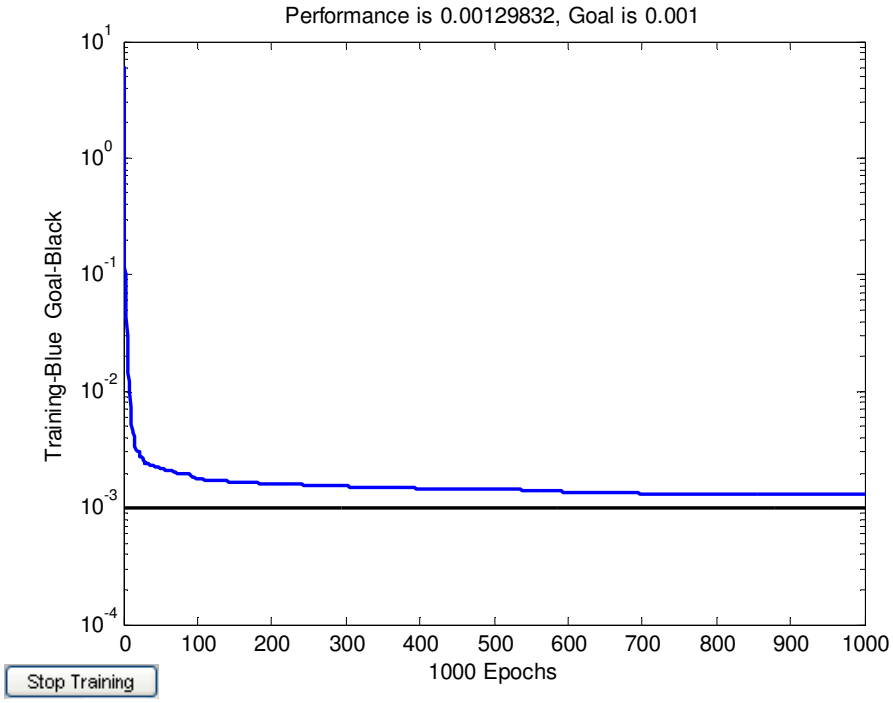


Fig. 7.1 (c): Network performance for a non-weather sensitive model. The goal was never met in 1000 iterations.

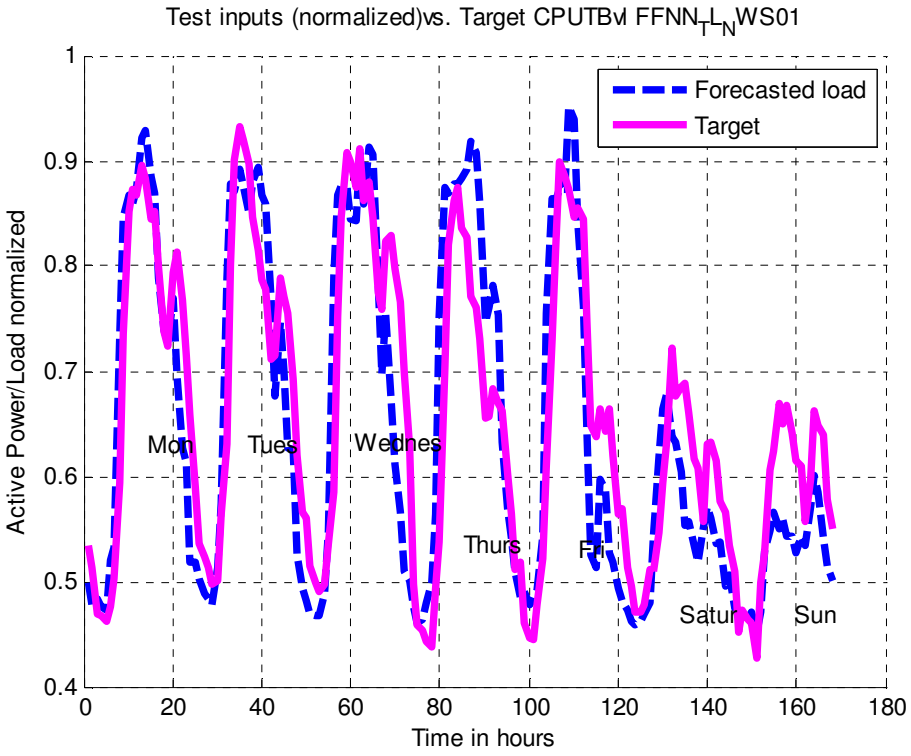


Fig 7.2: Normalized forecasts using testing data – Feedforward ANN (non-weather sensitive) in summer.

The following models now introduce the effect of the weather conditions.

7.3 Forecasting the total load using the weather sensitive ER model

Similarly, the simulation was again repeated, but then using the input vector # 1. As previously defined, this vector takes into account the effect of weather conditions. The following weather variables were used: daily average wind speed, humidity, and maximum and minimum temperatures.

And as duly specified in chapter six, the two models (FFNN_TL_WS01 and ERNN_TL_WS01) which represent Feedforward and Elman Recurrent Neural Networks were trained. Some obtained results and actual error in kW in Figure 7.3 (a) and (b) respectively. These particular results were obtained by training the Elman Recurrent Neural Network based model (ERNN_TL_WS01). The performance of this model is shown in Figure 7.3 (c). In Figure 7.3 (b), one observes that the weather sensitive forecasting model performs slightly better than the first one. The actual error for this model has improved to approximately 50% in comparison with the model discussed earlier.

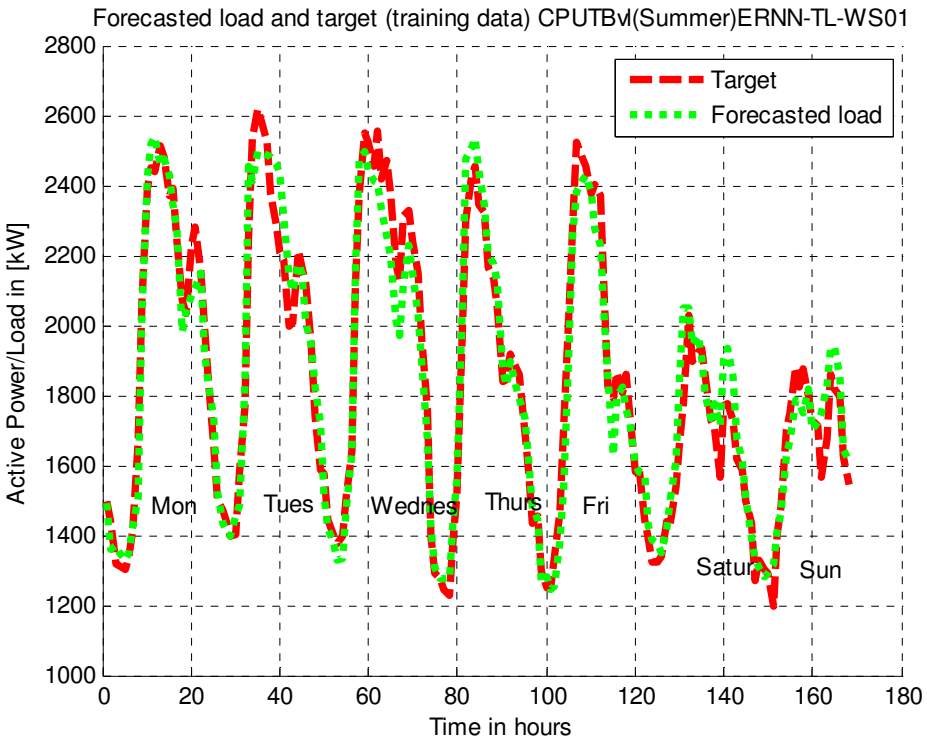


Fig. 7.3 (a): De-normalized forecast for the CPUT combined load (summer) - using Elman Recurrent Artificial Neural Network (weather sensitive model)

The performance goal for the Recurrent forecasting in Figure 7.3 (c) was successfully met only in 218 epochs.

7.3.1 Simulating the network using testing data

Again to evaluate the model's degree of generalization, this model was again simulated using the testing data, and the simulation results are shown in Figure 7.4.

7.4 Forecasting the total load using the weather sensitive FF model

Inline with the objectives of this research project, particularly the objective that was primarily designed to assess the performance of two different Artificial Neural Networks, it was decided to again present the same input vector to the Feedforward Neural Network model (earlier defined as FFNN_TL_WS01) and simulate the model. The subsequent simulation results, corresponding error, and network performance for this model are shown in Figure 7.5 (a), (b), and (c) in the sequence order. The goal was also successfully met in 354 epochs. And finally, the network response after presenting the model with the testing data is illustrated in Figure 7.6.

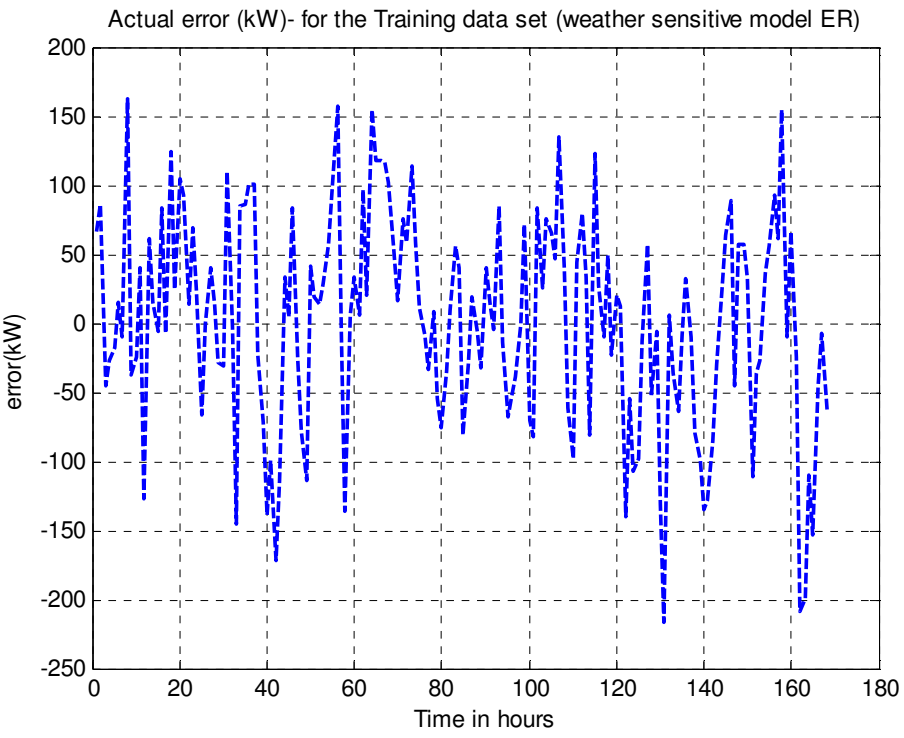


Fig. 7.3 (b): The corresponding mean squared error for ERNN_TL_WS01 model.

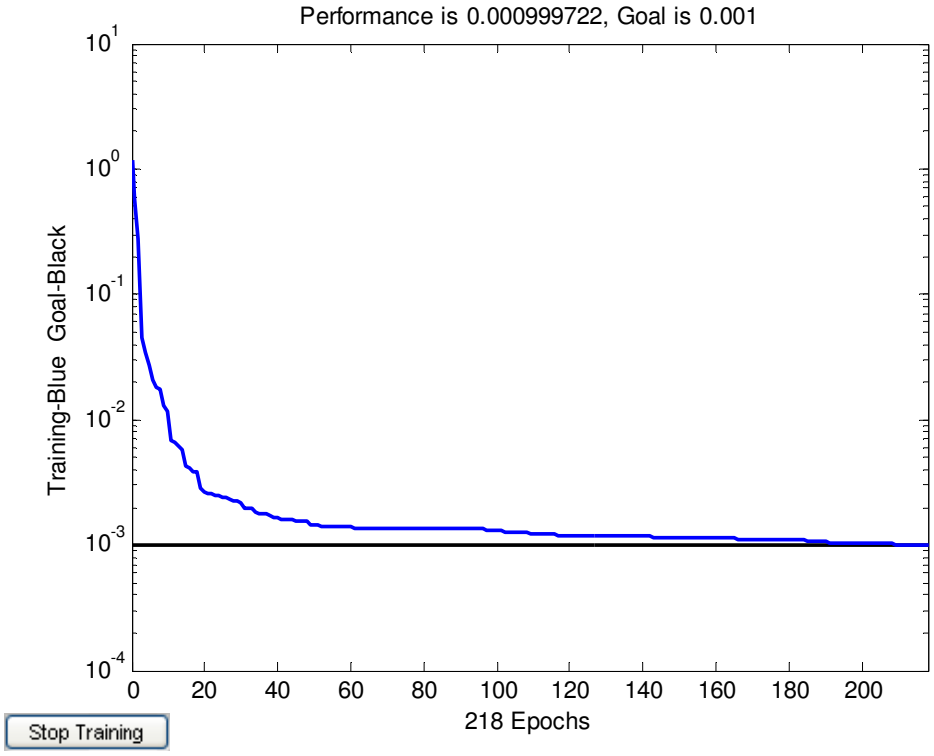


Fig. 7.3 (c): Network performance of the model (ERNN_TL_WS01).

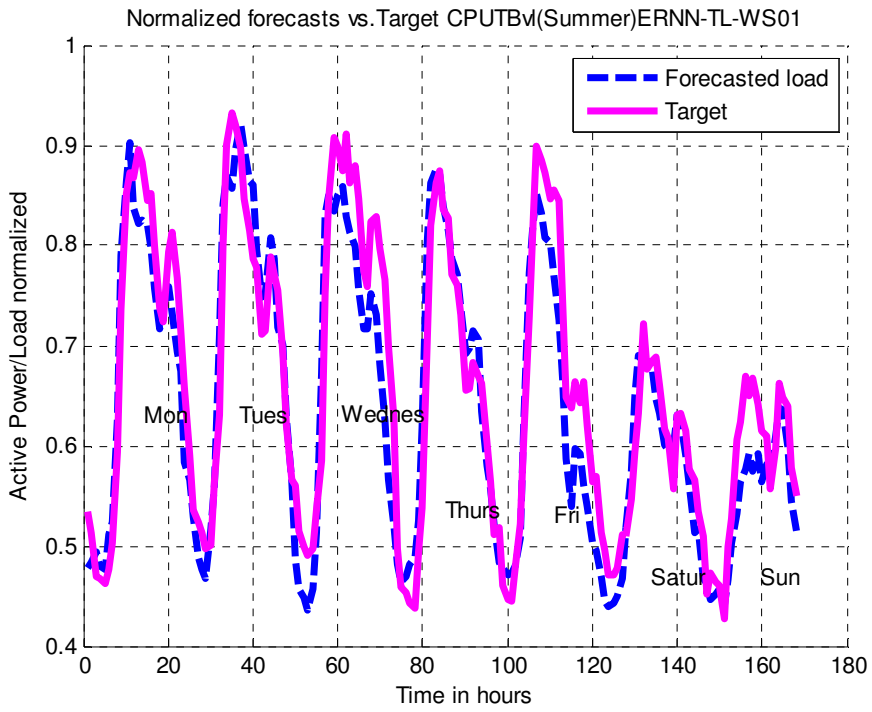


Fig.7.4: CPUT combined load normalized forecasts (in summer) using Elman Recurrent Neural Network (ERNN_TL_WS01). Only testing data were used.

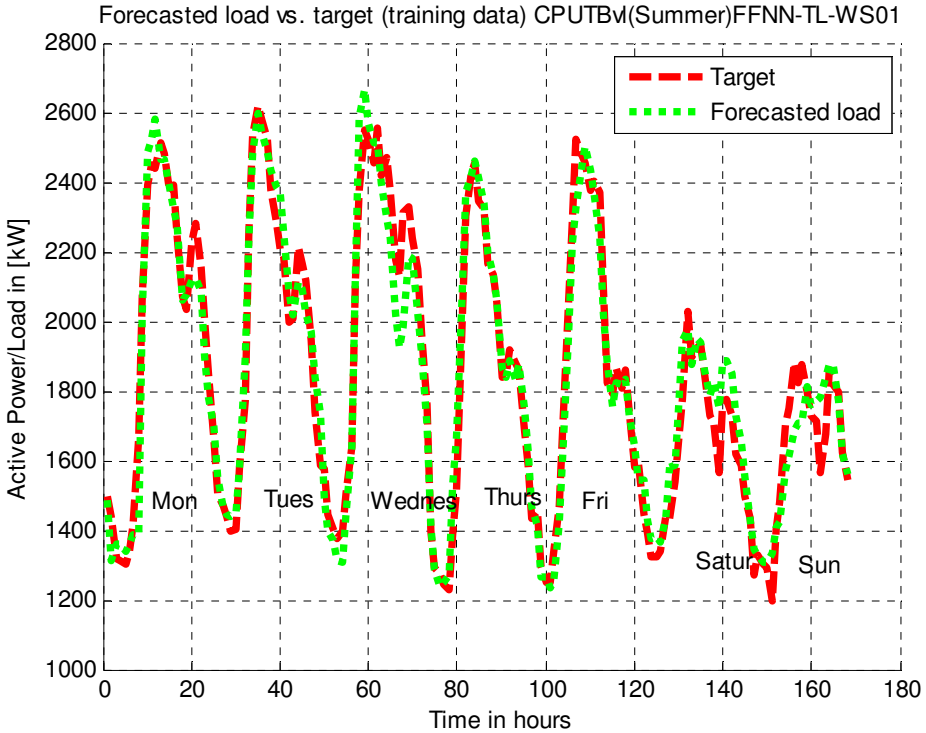


Fig. 7.5 (a): De-normalised forecasts (combined load) – using the Feedforward Neural Network weather sensitive model.

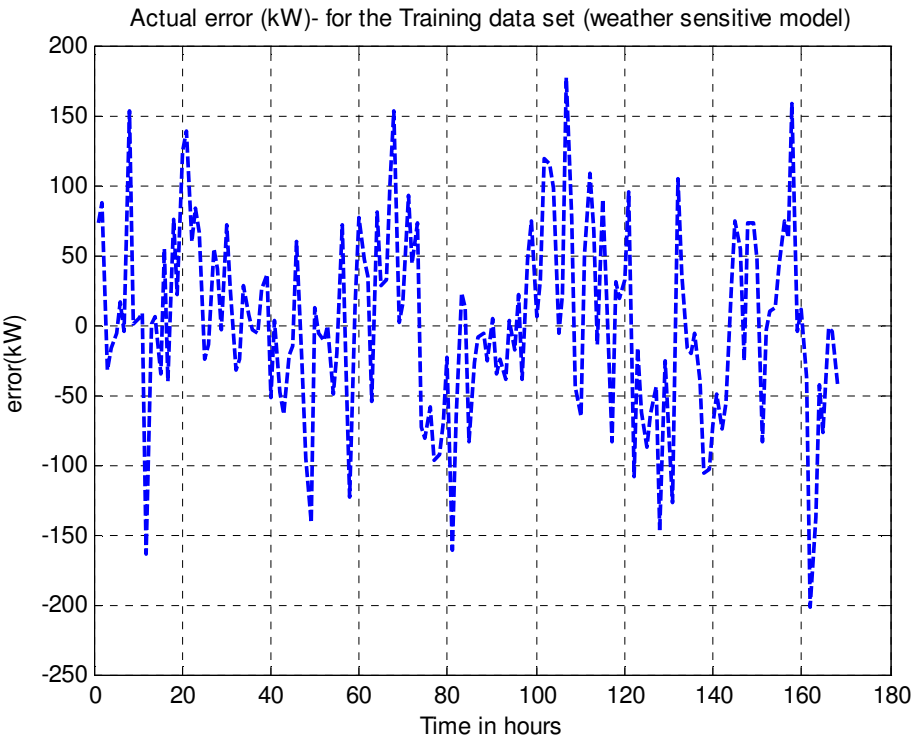


Fig. 7.5 (b): The corresponding actual error (kW) for the FFNN_TL_WS01 model

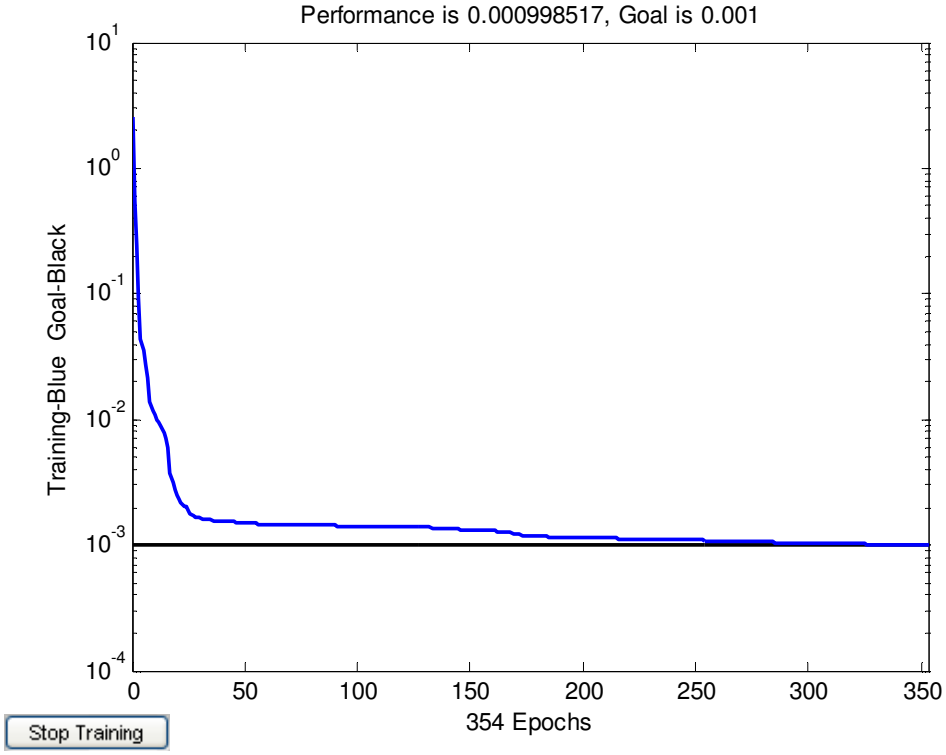


Fig. 7.5 (c): Network performance of the model (FFNN_TL_WS01).

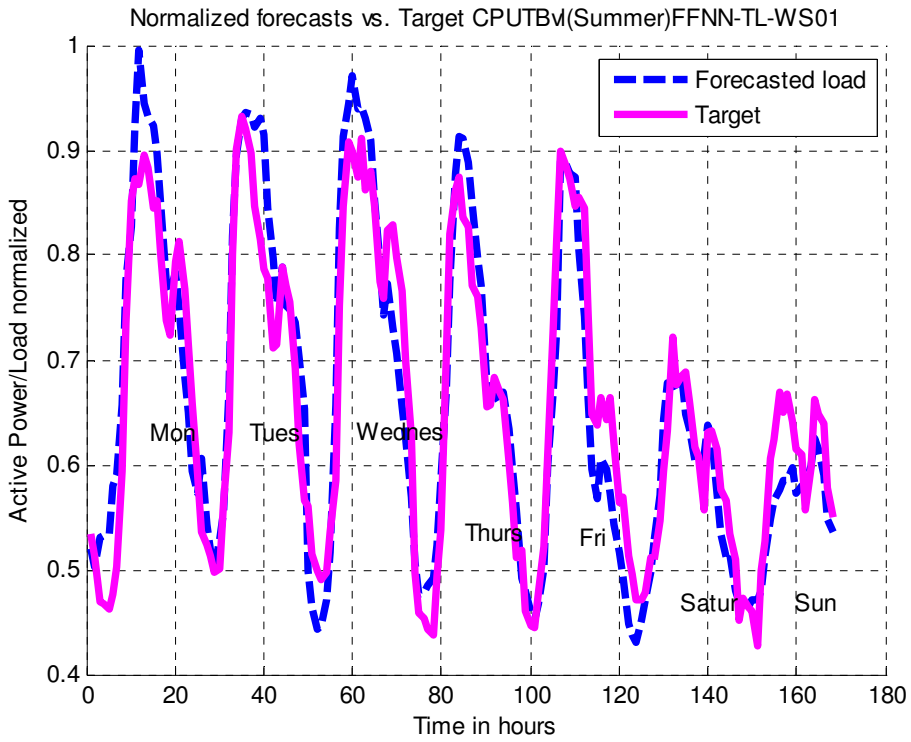


Fig. 7.6: Normalized forecasts using Feedforward Neural Network weather sensitive model

As clearly indicated in Figure 7.2 (c), the non-weather sensitive models have successfully failed to meet the defined goal (1e-3). However, the models presented with the IV that brings the effect of the weather conditions on the load have surprisingly met the goal as shown in Figure 7.3(c) and 7.5 (d). The very same models have also performed better than the non-weather sensitive ones when presented and simulated with data which where not used during training. Figure 7.4 shows the typical response of the weather sensitive models when simulated with testing data.

However, it should be noted that the defined performance goal (1e-3) was arbitrarily selected for preliminary model's performance evaluation. In this work, the absolute performance goal of 1e-5 was used to ensure improved forecast accuracy.

The following section now covers the responses emerged from different forecasting models when subjected to the new performance goal. Another important topic also covered in the next section is the implementation of the incremental training mode. This means that the connection weights between layers and biases are iteratively computed and adjusted every time an input is applied to the network. The subsequent errors and model performance results presented in both tabular and graphical forms are also included in the following section.

7.5 Model responses to the new performance goal (1e-5)

Firstly, the following models were evaluated using the batch training mode: FFNN_TL_NWS01, FFNN_TL_WS01, ERNN_TL_NWS01, and ERNN_TL_WS01. In all cases, none of these models manage to meet the goal.

Some network performances obtained from these models are shown in Figure 7.7 (a), (b), and in Appendix E. The network performance of the Elman Recurrent Neural Network weather sensitive model (ERNN_TL_WS01) is shown in Figure 7.7. This type of response also emerged from all other forecasting models including the forecasting models which predict the load during the winter period. Evidently, it can be seen that presenting a neural network with a lot of input data points does not guarantee improved accuracy or a better network performance.

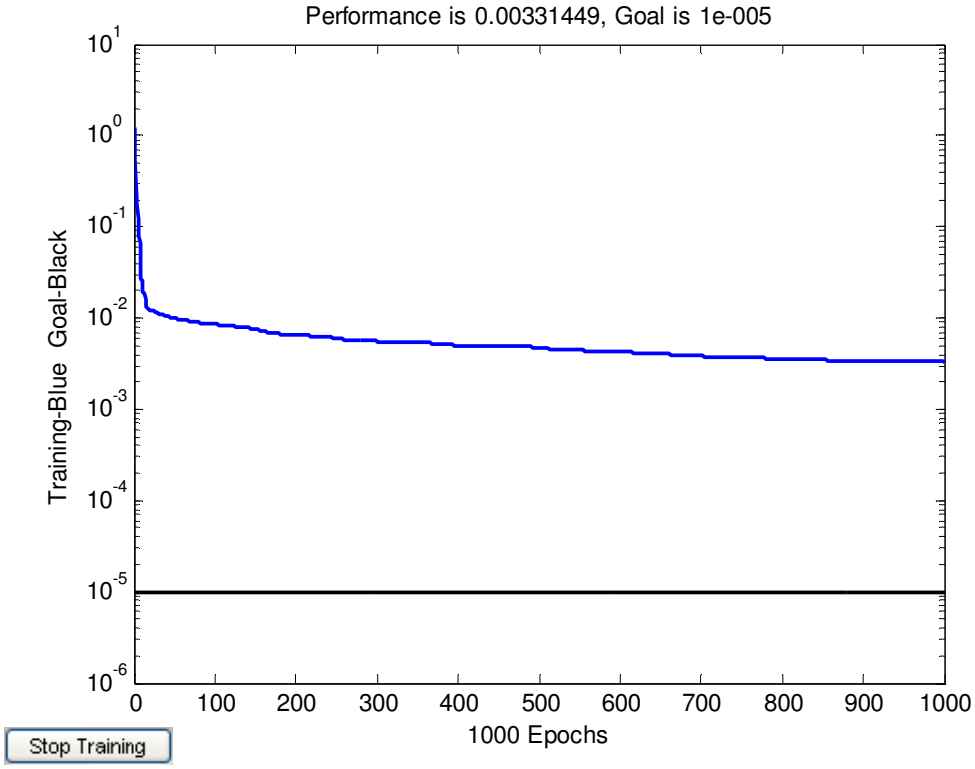


Fig. 7.7 (a): The network response of the Feedforward Neural Network Model (weather sensitive)

The following section now discusses the network responses obtained from different forecasting models after the implementation of the incremental training mode.

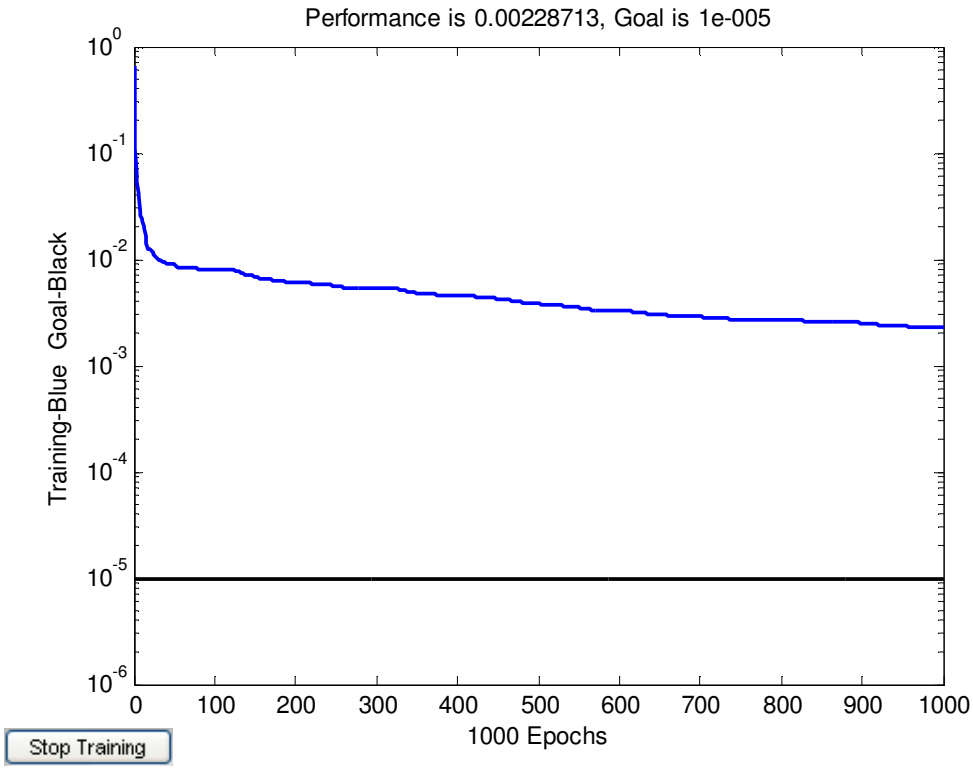


Fig. 7.7 (b): The network response of the Elman Recurrent Neural Network Model (non-weather sensitive)

7.6 Forecasting the total load using the hour-by-hour approach

The evolved Feedforward and Elman Recurrent Neural Network models for predicting the load in both seasons (winter and summer) were trained incrementally. In all cases, the input vector (IV) # 1 and # 2 were applied to the networks to predict the load with 168 hours lead time. As stated in the previous chapters, the IV # 2 is specifically designed to ignore the effect of weather related conditions on the load. The composition of the latter IV brings the effect of climatic conditions.

Figure 7.8 (a) shows 1 week ahead forecast for the CPUT total load emerged from the Feedforward Neural Network non-weather sensitive model (ERNN_TL_NWS01). The forecast and network performance came into sight after predicting the load using the Elman Recurrent weather sensitive model (ERNN_TL_WS01) are given in Figure 7.9 (a) and 7.9 (b) respectively.

In these figures, it can be seen that the incremental training mode is undoubtedly superior compared to the performance of the batch trained models.

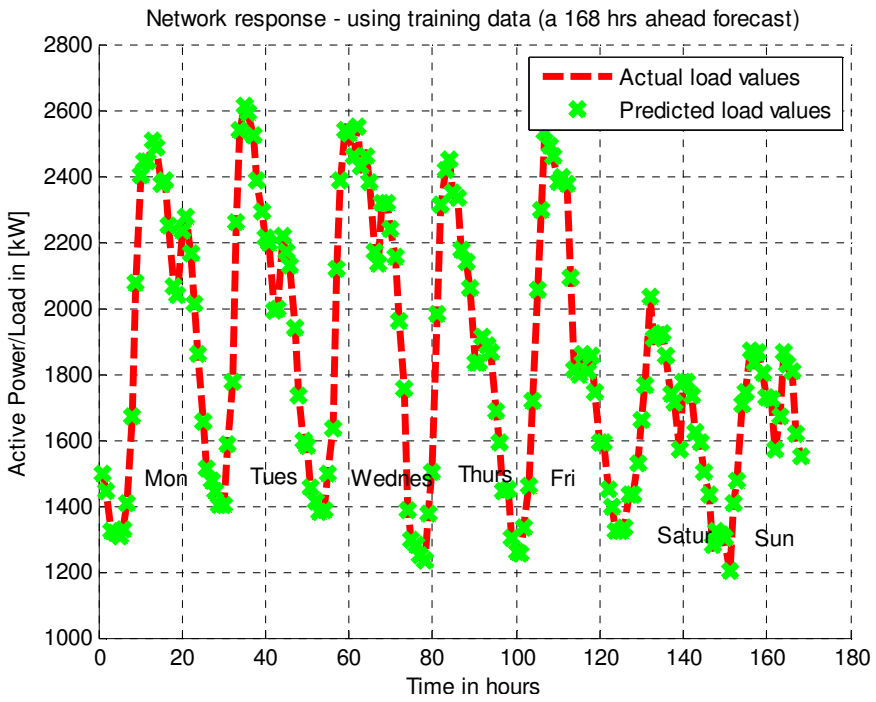


Fig. 7.8(a): 1 week ahead forecast (summer) – FFNN_TL_NWS01 (Feedforward non-weather sensitive model)

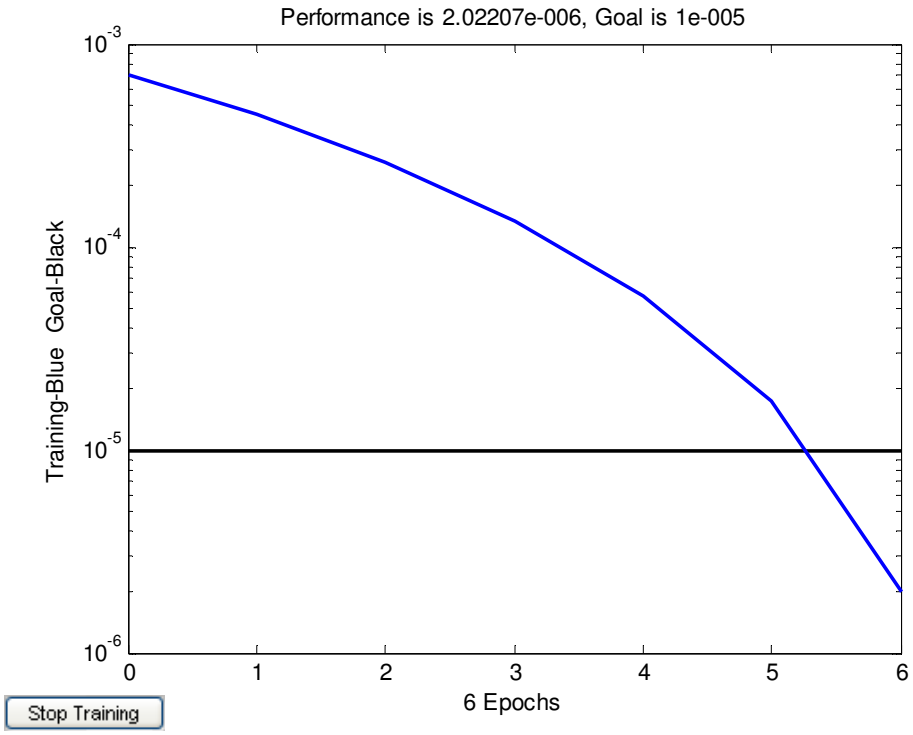


Fig. 7.8(b): The corresponding network performance- FFNN_TL_NWS01 (Feedforward non-weather sensitive) model)

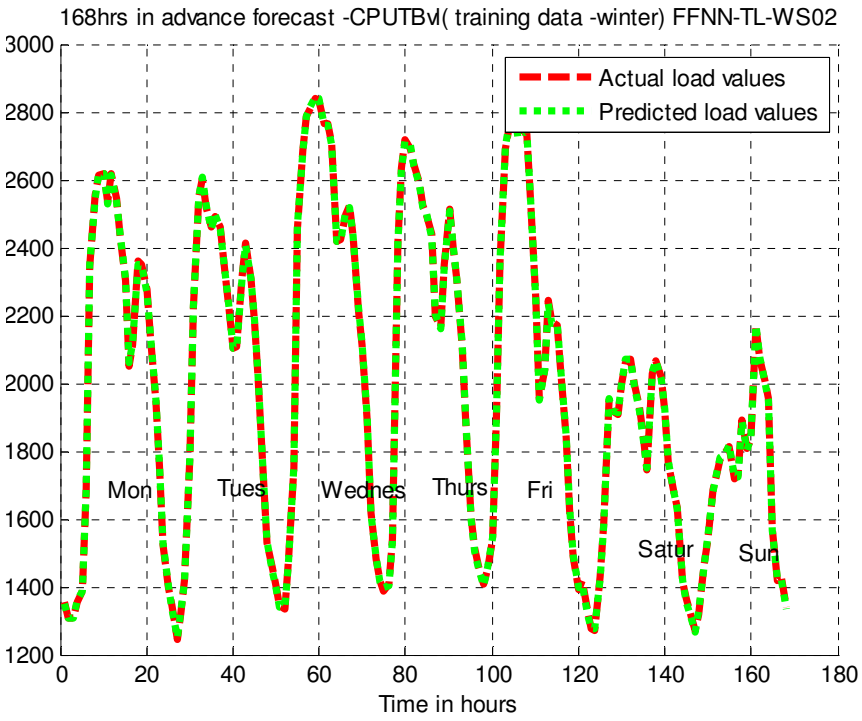


Fig. 7.8(c): 1 week ahead forecast (winter) – FFNN_TL_NWS01 (Feedforward non-weather sensitive model)

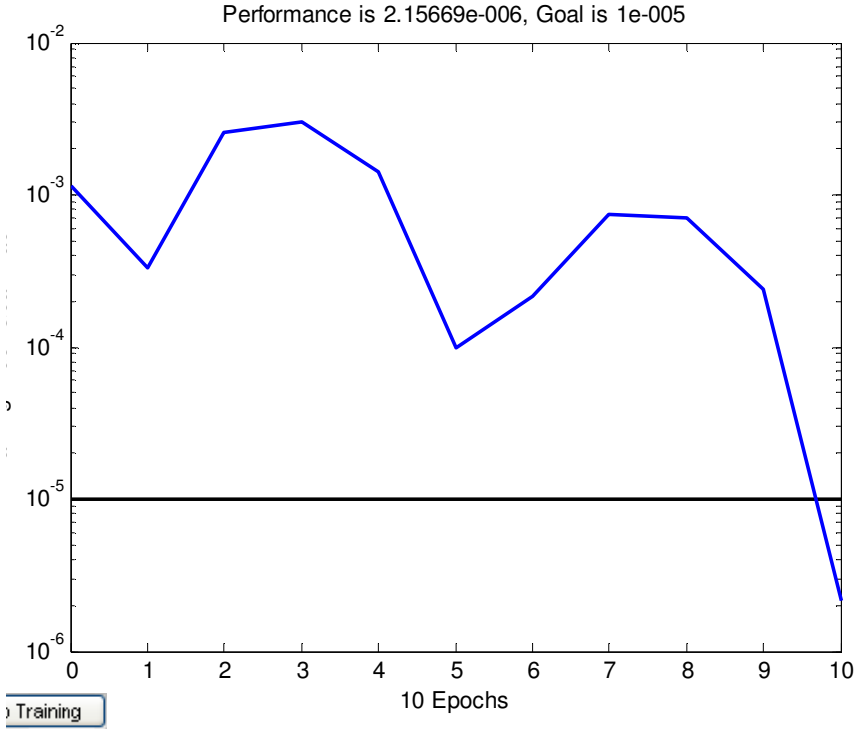


Fig. 7.8(d): The corresponding network performance of the FFNN_TL_WS02 (using incremental training mode) – for the last forecast value (168h)

Figure 7.8 (a) and (b) show a 1 week ahead forecast in winter. The forecast period is 04th – 10th August, 2008. Other subsequent forecasts and performance responses can also be found in Appendix E. Similarly, these models were also presented with the matching testing data. The results are summarized in Table 7.1.

7.7 Error comparative analysis for the total load forecasting models.

The developed models were evaluated using the standard mean squared error (MSE). The percentage mean error (MPE) was also used as a supplementary performance measure. The outcome of the analysis is also illustrated in Table 7.1. The result in the table shows that forecasting the whole week at once (batch training approach) can severely affect the accuracy of the forecast compared to the hour-by-hour approach. The testing result also indicates that the incremental training technique improves the degree of generalisation for a neural network.

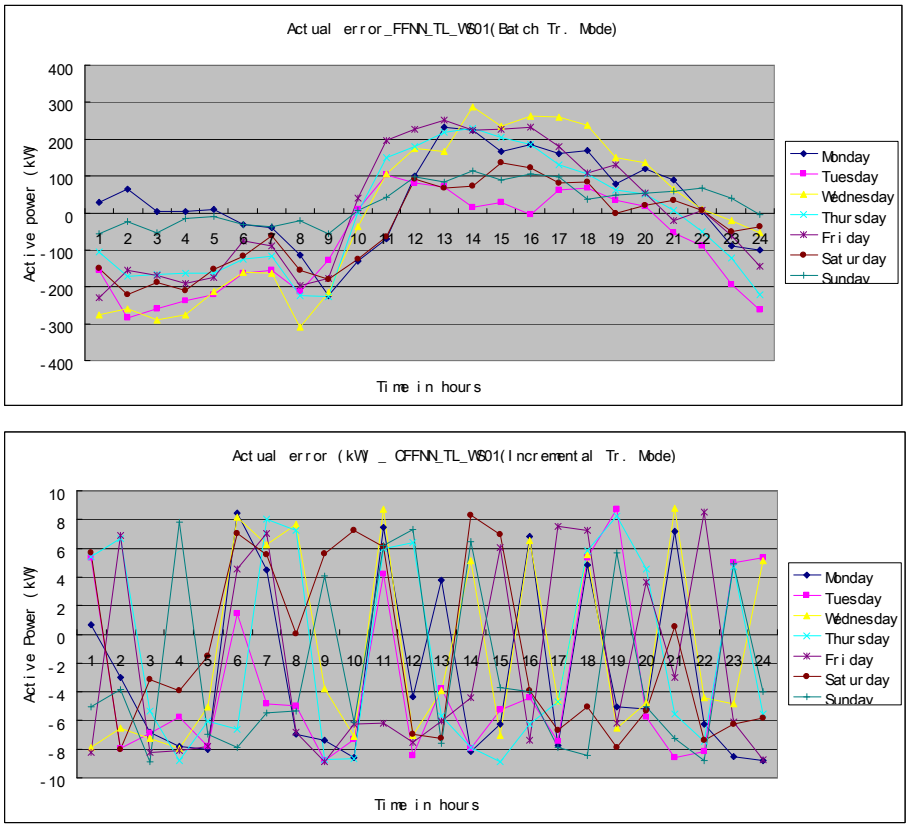


Fig. 7.10: Comparison of actual error – batch vs. incremental training mode. The forecast is for the period from 7th -13th April, 2008.

Figure 7.10 shows the actual daily errors for the Feedforward Neural Network model trained in the batch manner and incrementally respectively. Some more comparisons of the actual error are given in tabular form in Appendix E.

Model	Input Vector (IV)	Network	training mode	*MSE*		*MPE*		Season
				training	**testing**	**training**	**testing**	
FFNN_TL_NWS01	*IV # 2*	Feedforward	batch	0.001364707	0.006668	5.78E-06	0.001575	summer
ERNN_TL_NWS01		Elman Recurrent	batch	0.001175932	0.005098	4.41E-06	0.000498	
FFNN_TL_WS01	*IV # 1*	Feedforward	batch	0.000667856	0.014004	8.67E-07	0.004432	
ERNN_TL_WS01		Elman Recurrent	batch	0.000753235	0.025275	5.92E-07	0.000394	
CFFNN_TL_NWS01	*IV # 2*	Feedforward	incremental	5.40E-06	8.98E-03	7.03E-07	3.50E-04	
CERNN_TL_NWS01		Elman Recurrent	incremental	2.07398E-06	0.001477	2.73E-09	3.48E-05	
CFFNN_TL_WS01	*IV # 1*	Feedforward	incremental	5.40215E-06	6.54E-02	7.03E-07	9.11E-06	
CERNN_TL_WS01		Elman Recurrent	incremental	3.41951E-06	1.95E-03	2.17E-09	5.21E-05	
FFNN_TL_NWS02	*IV # 2*	Feedforward	batch	0.005585235	0.011151	6.77E-05	0.000205	winter
ERNN_TL_NWS02		Elman Recurrent	batch	0.005636492	0.013949	7.34E-05	0.000308	
FFNN_TL_WS02	*IV # 1*	Feedforward	batch	0.003314488	0.020662	2.36E-05	0.000169	
ERNN_TL_WS02		Elman Recurrent	batch	0.002287125	0.018256	1.37E-05	0.000479	
CFFNN_TL_NWS02	*IV # 2*	Feedforward	incremental	4.85248E-06	7.25E-05	1.17E-07	3.86E-04	
CERNN_TL_NWS02		Elman Recurrent	incremental	3.63666E-06	4.48E-04	1.61E-08	2.23E-06	
CFFNN_TL_WS02	*IV # 1*	Feedforward	incremental	4.85E-06	6.30E-02	1.17E-07	3.36E-05	
CERNN_TL_WS02		Elman Recurrent	incremental	3.15444E-06	1.66E-04	2.25E-08	4.24E-04	

Table 7.1: Error comparative analysis on the evolved forecasting models for the total load

7.8 Forecasting the departmental loads

Since the characteristics of the department load profiles are already highlighted in chapter 5, there is no need to replicate or further discuss these profiles. Thus the following topic directly talks about forecasts obtained from the developed forecasting models. One more thing that needs to be understood here is that the batch training mode is no longer considered due to the associated high degree of inaccuracy. This implies that all forecasting models for different departments would be incrementally trained to forecast respective loads as desired.

The forecast lead time (168 hours) remains unchanged for the IT, Electrical departments and the Administration building. However, the characteristic of the load for the Air Conditioning Plant (ACP) is uniquely noticeable and needs to be handled differently. This load lacks consistence in its pattern due to fast load fluctuations. Thus the forecast lead time is reduced to 24 hours as an attempt to possibly use more meaningful training data. The 24hrs lead time forecast horizon was also used to predict the for the Mechanical Engineering department.

Some networks for departmental load forecasting models were only presented with the IV # 2, because the effect of weather related variables was not explicitly observed in the previous model performance evaluation. In simple terms, all departmental load forecasting are non-weather sensitive one and incrementally trained.

Moreover, the architecture of the neural networks for the departmental forecasting models has been slightly modified as clearly indicated in Table 7.2.

Network	Hidden Layer	Hidden neurons	Output nodes	Training function	Transfer functions	Momentum factor	Learning rate
Feedforward & Recurrent	1	5	1	Gradient Descent Algorithm with momentum	Tan sigmoid and Pure linear	0.75	0.3

Table 7.2: Network Topology for the departmental load forecasting models

The error comparison analysis for the departmental load forecasting models is summarized in Table 7. 6.

7.8.1 Forecasting the load for the IT department

The load measurements of the IT department were taken from 12th August – 01 September 08. Thus the data set was defined as: training data (25th August – 1st September), target data (17th – 24th August), and testing data specified as the training data set. And subsequent 1 week ahead forecast of the weather sensitive IT load forecasting models is shown in Fig. 7.11(a) and (b).

The performance of this model is shown in Figure 11 (c). And a comparison in terms of the actual errors of the incrementally and batch trained forecasting Recurrent models described in section (7.5.1) is given in Table 7.3 (a) and (b). The result shows that the batch trained model gave some good forecast at specific hours of the forecast day (hr 7 is a good illustrative case) compared to the incremental one.

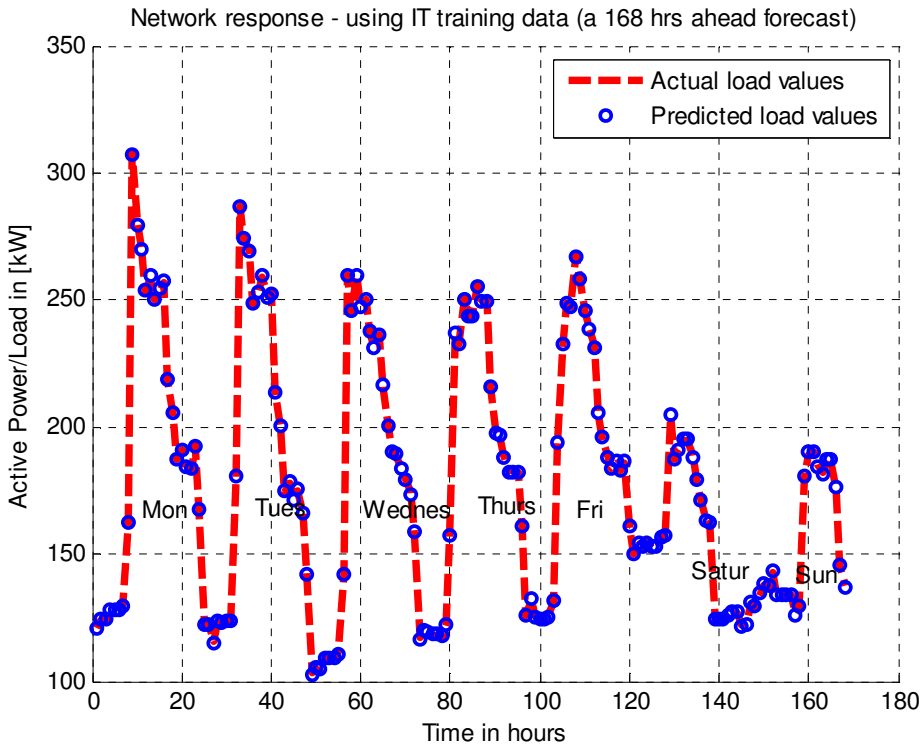


Fig. 7.11(a): One week ahead forecast for the IT department (the forecast is for the week of 17th – 24th August, 08) – FFNN_IT_WS, incrementally trained.

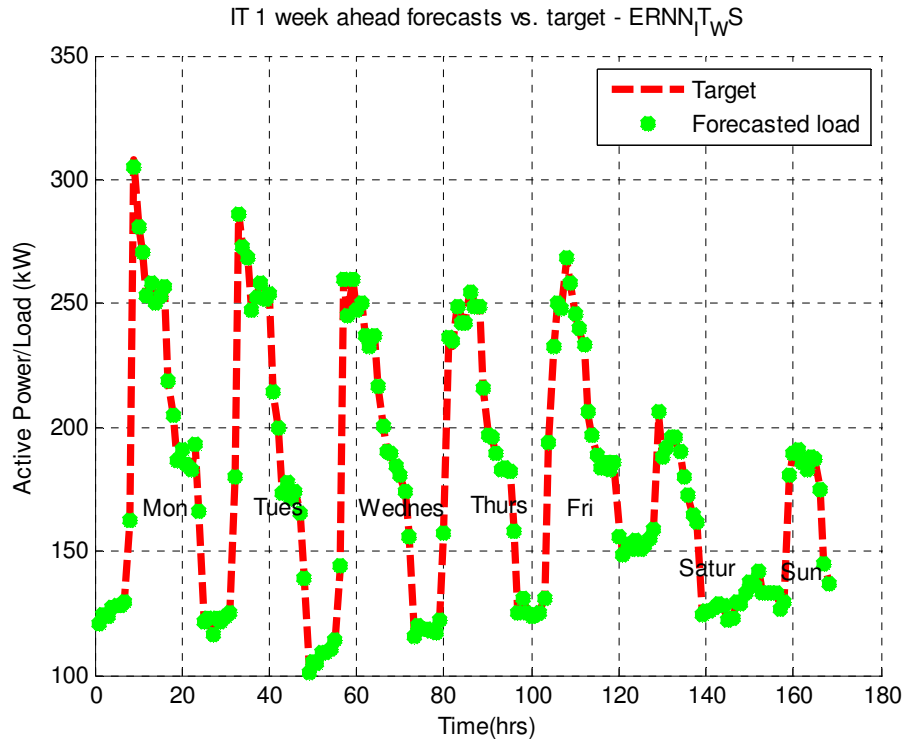


Fig. 7.11(b): One week ahead forecast for the IT department (the forecast is for the week of 17th – 24th August, 08) – FFNN_IT_WS, trained in batch mode. Forecast week is from 17th -24th August, 2008.

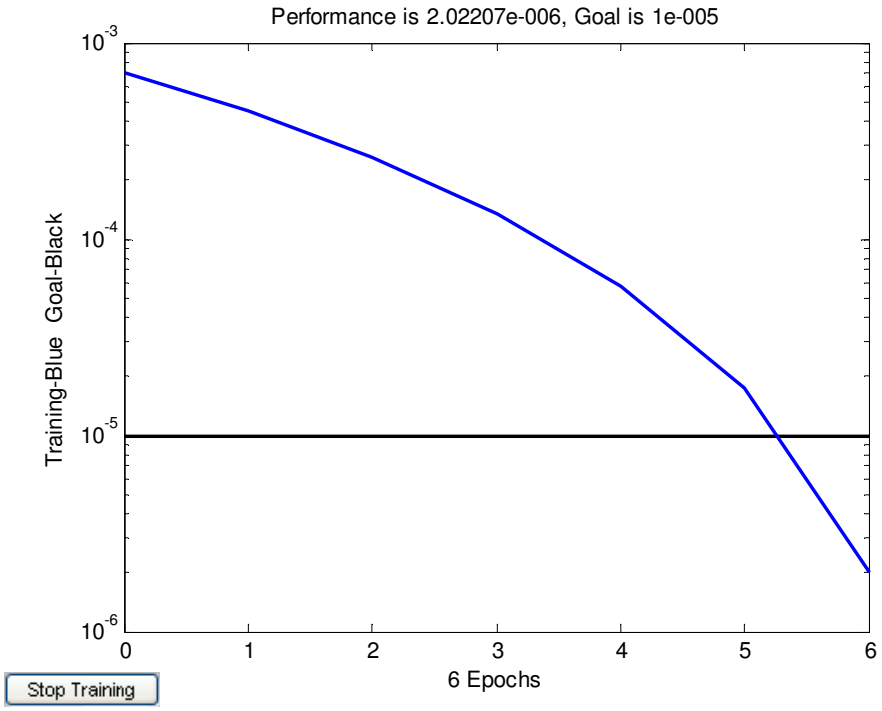


Fig. 11 (c): Performance of the feedforward IT load forecasting model

	Actual error (kW) - FFNN Prediction model (FFNN_IT_WS) – IT Depart.						
		Incremental training mode, IV # 1					
Hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	-0.95922	0.804706	0.693268	0.778046	0.900866	-0.49954	-0.82276
2	0.891821	0.804706	0.768659	0.594778	0.369755	0.551078	0.177239
3	0.891821	-0.33195	-0.23134	-0.40522	-0.88637	-0.88384	0.384564
4	0.770119	0.357281	0.75073	-0.85719	-0.82377	0.924761	-0.61544
5	0.773542	-0.64272	0.75073	-0.85719	0.17623	-0.90688	0.690262
6	0.773542	0.827941	0.765069	-0.81159	0.940984	0.09312	0.738388
7	0.775038	0.827941	0.771335	0.659381	0.935289	0.882777	-0.26161
8	-0.88443	-0.4505	-0.85734	-0.96226	-0.49892	0.822744	0.903394
9	0.066452	-0.84466	-0.92864	-0.61864	-0.71533	-0.86994	-0.43382
10	0.753781	-0.74181	-0.85347	-0.56968	0.872549	-0.95749	-0.43382
11	-0.44184	-0.7465	0.757985	0.933934	-0.90196	0.868333	-0.43382
12	0.497649	0.61381	-0.65594	-0.95498	-0.54234	0.766422	-0.43382
13	0.348151	0.883782	0.668995	-0.95498	-0.56487	0.766422	-0.39831
14	-0.46357	0.36884	-0.66071	0.591803	-0.72682	-0.82392	0.699938
15	0.556739	-0.3553	-0.89117	-0.85141	-0.36043	-0.57996	-0.96056
16	0.559937	0.644696	0.887261	-0.85141	-0.39438	-0.40614	0.430761
17	0.695615	0.71262	0.515753	0.940458	0.739232	-0.45041	0.872765
18	-0.71286	-0.71195	-0.82966	**-0.96765**	-0.44262	-0.69935	-0.64975
19	0.504963	0.720817	-0.58026	-0.85986	-0.39878	0.700432	-0.57458
20	0.709216	0.743198	-0.96396	-0.42114	-0.84993	0.700432	0.696694
21	-0.83293	-0.82758	-0.93891	-0.34405	0.920056	0.700432	0.696694
22	-0.80852	0.949348	-0.77753	-0.34405	-0.90894	0.770703	-0.55206
23	0.492489	-0.43136	-0.36315	-0.81987	0.913795	0.773797	0.824441
24	0.688315	0.68626	-0.87675	0.61579	0.676946	-0.2262	-0.38447

Table 7.3(a): Daily actual errors (kW) for the **total load** in summer – for the incrementally trained Feedforward model

However, the incrementally trained model, again, convincingly gave the smallest error through out the forecasting period. As indicated in these Tables, the highest actual error (2.029 kW) for the batch trained model was obtained Thursday, 21st August at hour 11:00. While the latter model has the highest error (0.967 kW) also recorded on the same Thursday at hour 18:00.

	Actual error (kW) - ERNN Prediction model (ERNN_IT_WS)						
		batch training mode, IV #1					
Hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	0.149638	1.223561	1.660549	1.012544	0.520114	1.311398	0.002132
2	-0.53001	0.213081	-0.7514	-0.19228	0.636313	0.783025	-0.02204
3	0.255152	-0.26234	-0.91741	0.216178	0.454916	0.904801	0.219287
4	0.445805	-0.309	-0.3348	-0.27605	0.329877	0.834964	0.31462
5	0.758461	0.306812	0.003725	-0.15065	0.340497	0.926882	0.503543
6	0.807855	0.364453	-0.73107	-0.22648	0.460012	1.051486	0.632146
7	**0.0665**	**-0.873**	**-3.26473**	**-0.48492**	**-0.03401**	**0.541077**	**0.658862**
8	-0.55545	-0.18761	-1.98104	-0.30779	-1.24355	-0.51694	0.611654
9	2.724826	0.913159	0.053649	-0.5775	-0.9788	-2.26558	0.544614
10	-0.56628	0.99162	-0.39992	-1.81375	-1.25437	-1.24268	0.509075
11	0.083661	1.260355	0.197767	**2.029063**	-1.31592	-0.90452	0.494106
12	0.067505	0.57129	-0.02848	1.708496	-0.65242	-1.26235	0.556299
13	0.588693	0.638934	-0.29831	1.463113	-0.37503	-1.33406	0.441867
14	0.089182	0.641629	-0.33242	1.235145	-0.49898	-1.23709	-0.46426
15	0.567706	0.034676	-0.62884	1.280102	-1.01618	-0.81763	-0.64704
16	0.255615	-0.82382	-0.80222	1.098009	-1.22382	-0.67205	-0.79084
17	0.250436	-0.42794	-0.4879	0.059361	-1.21163	-0.46398	-0.94612
18	0.140326	0.472933	-0.55465	-0.10456	-1.01218	0.16661	-0.58569
19	0.250231	0.434557	-0.58545	-0.35829	0.071418	0.193609	-1.05459
20	-0.21827	0.276391	-0.55177	-0.45309	0.046667	-0.8051	-0.72501
21	0.006146	0.782483	-0.4912	-0.25263	-0.20298	-0.61597	-0.45221
22	0.16186	0.71347	-0.6831	-0.37219	0.208327	-0.60896	1.102097
23	0.158397	0.8536	-0.44208	-0.18312	0.272807	-0.54808	1.194572
24	0.716072	1.858276	2.152413	1.565315	3.994725	-0.85638	1.198676

Table 7.3(b): Daily actual errors (kW) – for the batch trained Elman Recurrent model (**total load** in summer)

7.8.2 Forecasting the load for the Electrical department

The electric load for the Gold field (Electrical) department was record for a two week period from 16th – 30 June, 08. The training data and target were defined from 23rd – 29 of June and 16th – 22nd of June respectively. The evolved models were also simulated with the training data. The results and performance obtained from these Feedforward and Recurrent weather sensitive models are shown in Figure 7.12 (a), (b), (c), and (d).

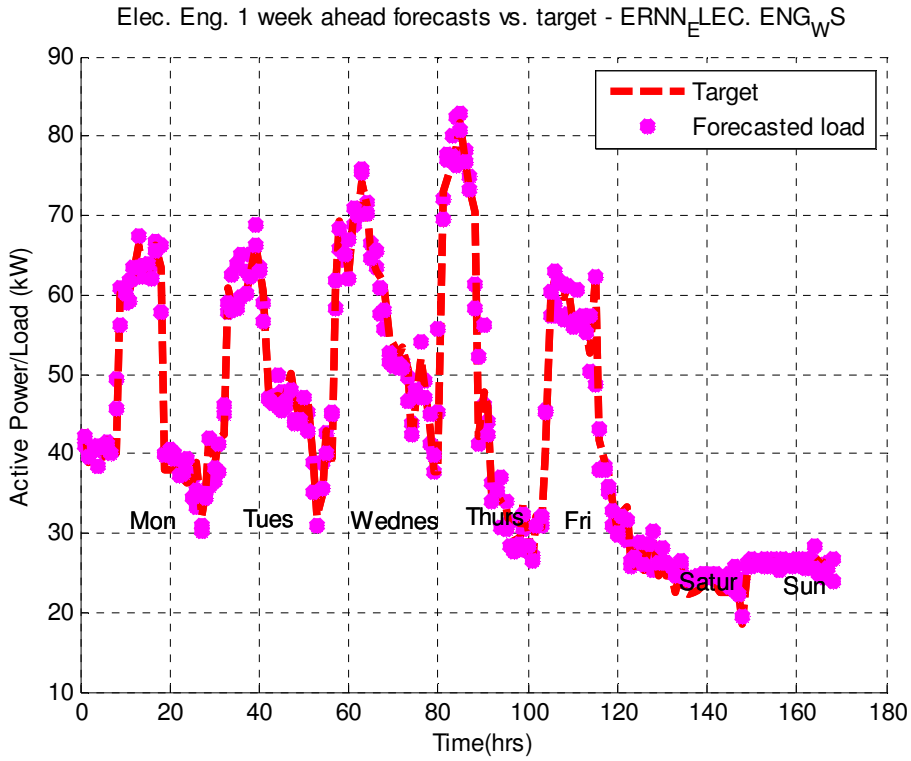


Fig. 7.12(a): One week ahead forecast for the Electrical Engineering department (ERNN_ELEC_WS-trained in the batch mode. Forecast week is from the 16th – 22 June, 2008.

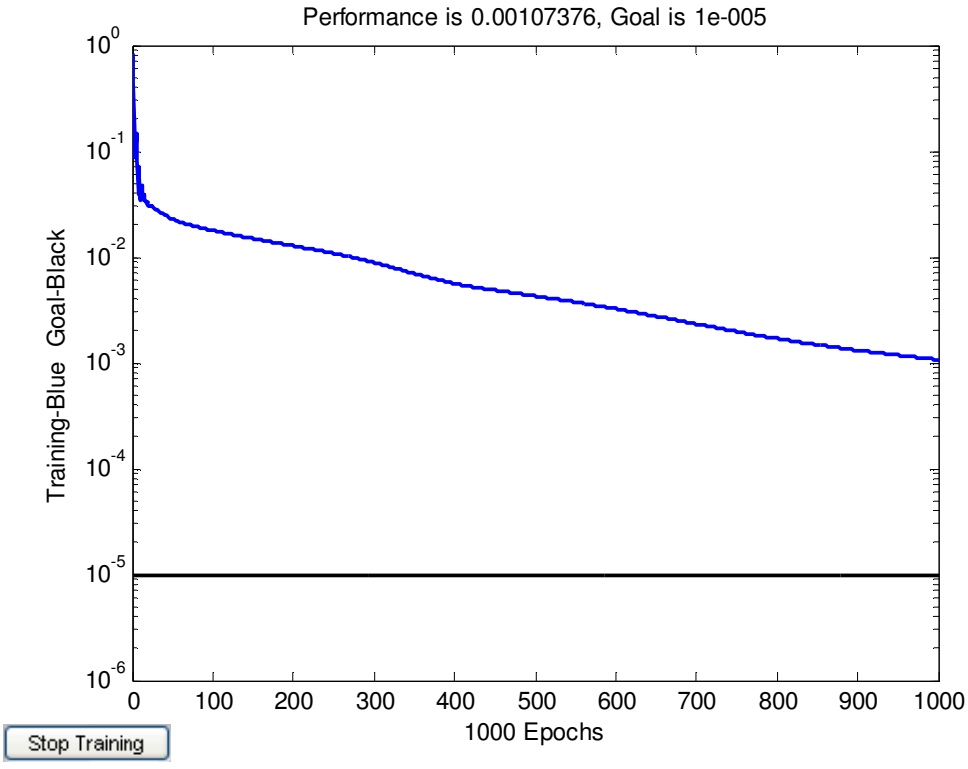


Fig. 7.12(b): Corresponding network performance for the Elman model (ERNN_ELEC_WS)-

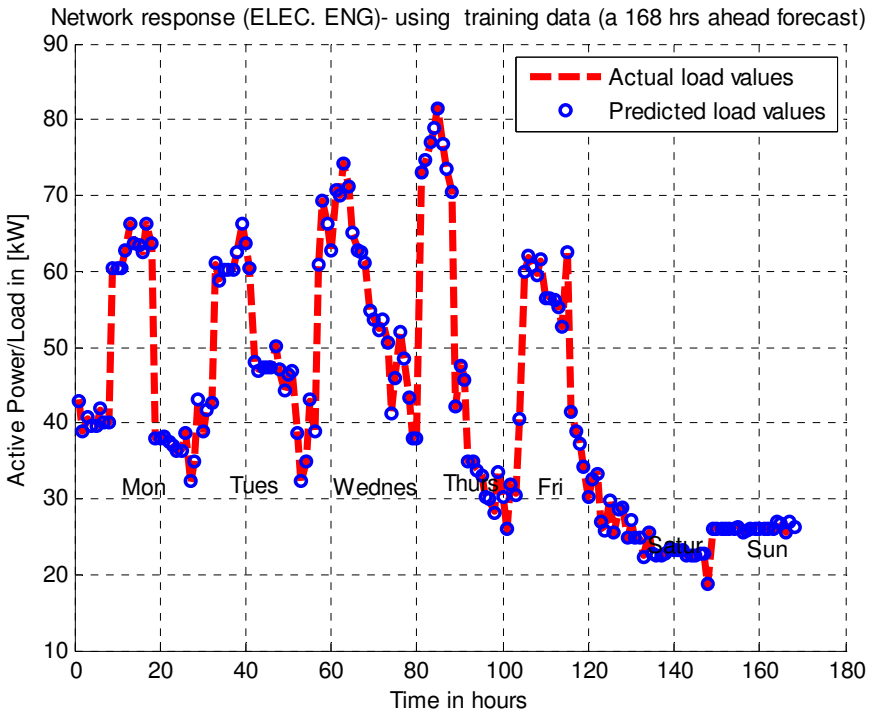


Fig. 7.12(c): One week ahead forecast for the Electrical Engineering department (FFNN_ELEC_WS-incrementally trained. Forecast week is from the 16th – 22 June, 2008.

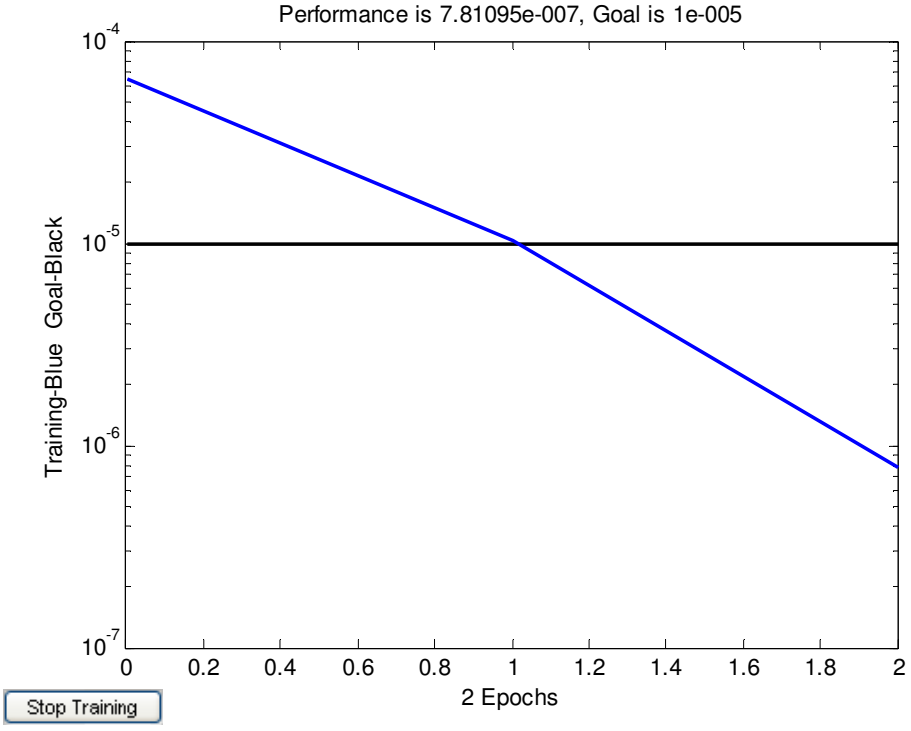


Fig. 7.12(d): The corresponding network performance for the Feedforward model (FFNN_ELEC)

7.8.3 Forecasting the load for the Mechanical department

The recording of the total load for the Mechanical department was exceptionally tricky because some of the big machinery (lathe machines) were not in operation when measurements were taken from 01st – 10th of September 2008. However, for forecasting purposes it was found necessary to add up a *45%* to the measured load to cater for contingences (the worse case scenario here is when all machines and other big loads are in use at the same time). Since the recorded data is only for a week period, which is insufficient for the defined model input vector (IV) structure in terms of the required past contiguous load values, it was found appropriate to reduce the forecast lead time (168 hours) to a 24 hours forecast instead. This implies that only most recent past load values $(L(t-1)$, and $(L(t-2))$ of the previous similar day are used to the load for the next hour. Figure 7.13 (a) shows a 24hrs forecast for the department. The corresponding performance of the model and actual error (kW) are given in Table 7.4 and Figure 7.13 (b) respectively. The forecast day was Tuesday, 2nd of September. The load data for Monday, 01st September were used to train the model.

7.8.4 Forecasting the load for the Administration Building

The Lovato power meter was installed and commissioned at the *Sacco* min-substation to record the load for the administration building for a two week period (02nd – 15th June, 2008). Similarly, the second week recorded data were used as the training data and the other remaining data were defined as the target. Figure 14 (a), (b), (c), and (d) show the obtained forecasts and performance of different models. Some actual error comparisons of the develop load forecasting models for this building are given in Table 7.5 (a) and (b).

7.8.5 Forecasting the load for the Air Conditioning Plant (ACP)

The load for the ACP is the most convoluted one. Not only that this load fluctuates unexpectedly, but also because it has a *"z-shaped"* unique pattern which makes it extremely difficult to predict. The approach used here briefly is as follow: the forecast lead time was also reduced to 24 hours and only two past contiguous load values (as discussed in section 7.53) were used to predict the load of the next hour.

	Actual error (kW) - FFNN Prediction model (FFNN_MEC) - MECHANICAL. ENG Department - for 24hrs lead time forecast. Forecast day is Tuesday, 2nd of Sep, 08																							
kW	0.39	0.2	0.35	-0.21	-0.28	-0.41	0.25	-0.23	-0.28	0.39	-0.48	-0.23	0.14	-0.37	0.5	-0.26	0.22	-0.29	0.16	0.47	-0.53	0.19	-0.26	-0.28
Hr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Table 7.4: Daily actual errors (kW) for the Feedforward Neural Network incrementally trained Model (FFNN_MEC)

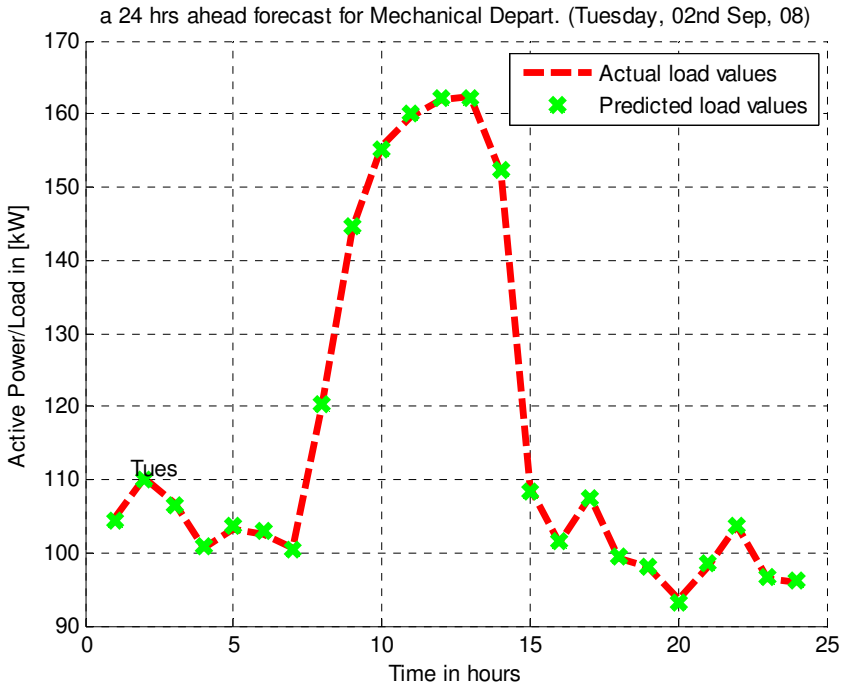


Fig. 7.13(a): One day forecast for the Mechanical Engineering Department electric load

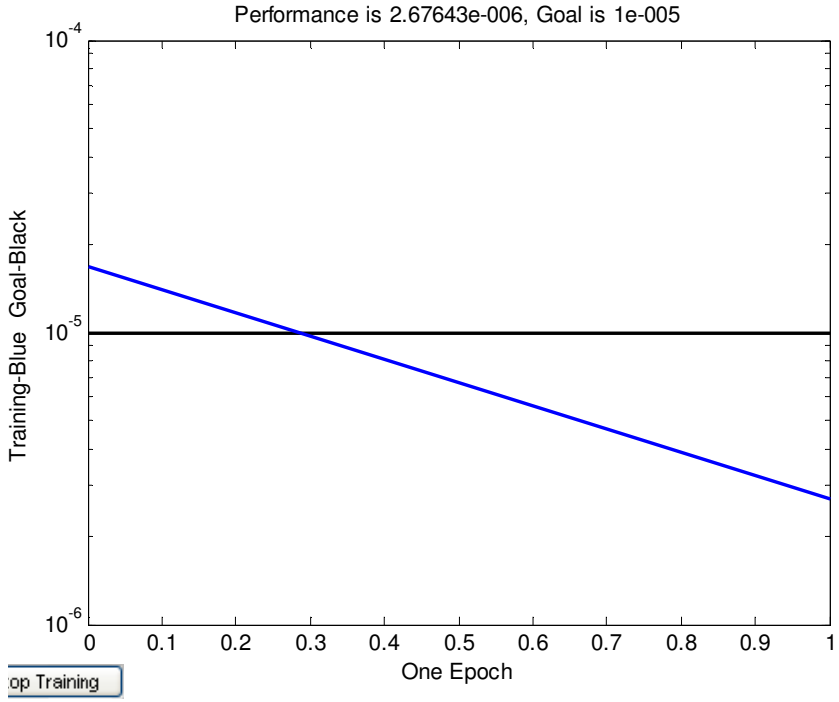


Fig. 7.13 (b): The corresponding model performance

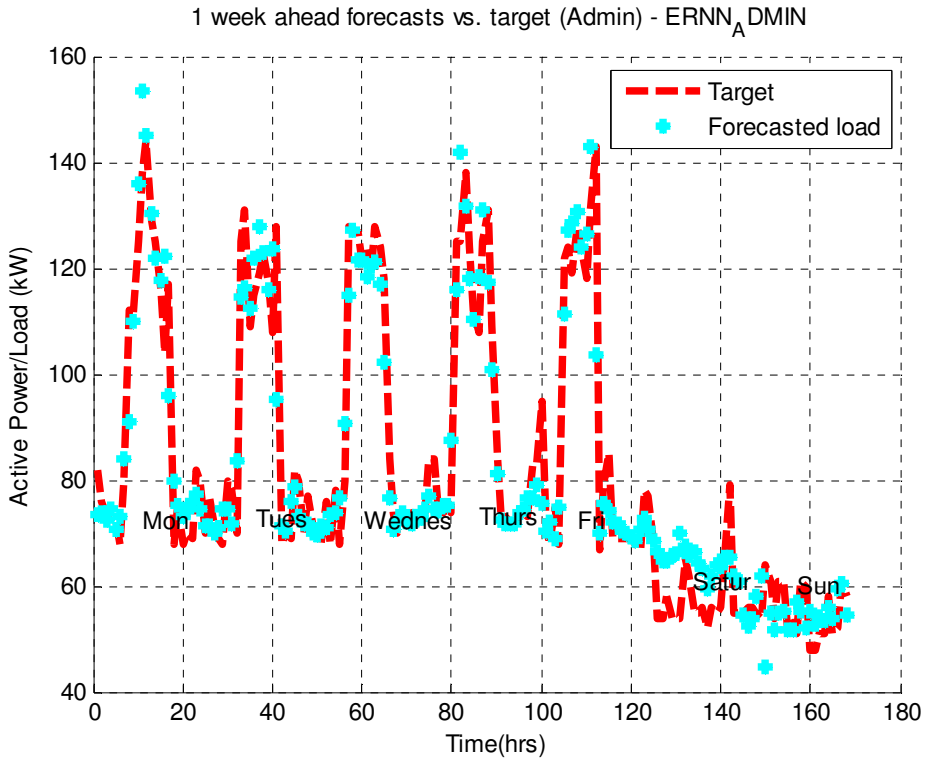


Fig. 7.14(a): One week ahead forecast for the Administration Building (ERNN_ADM_WS- batch trained). Forecast week is from the 2nd – 8th of June 2008.

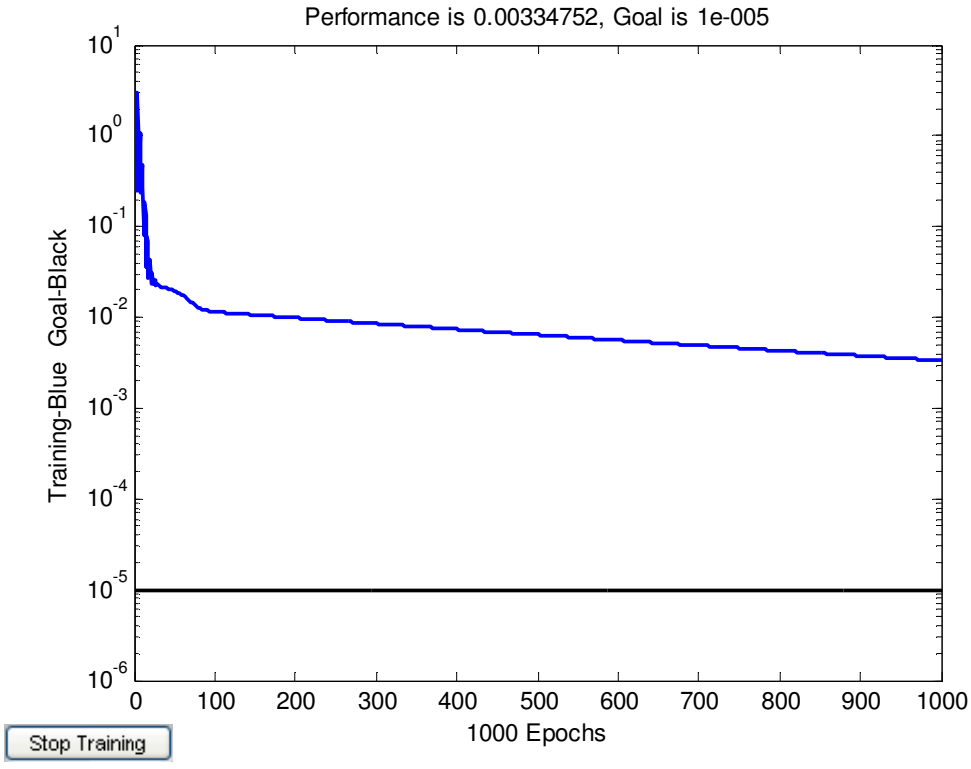


Fig. 7.14 (b): The corresponding model performance.

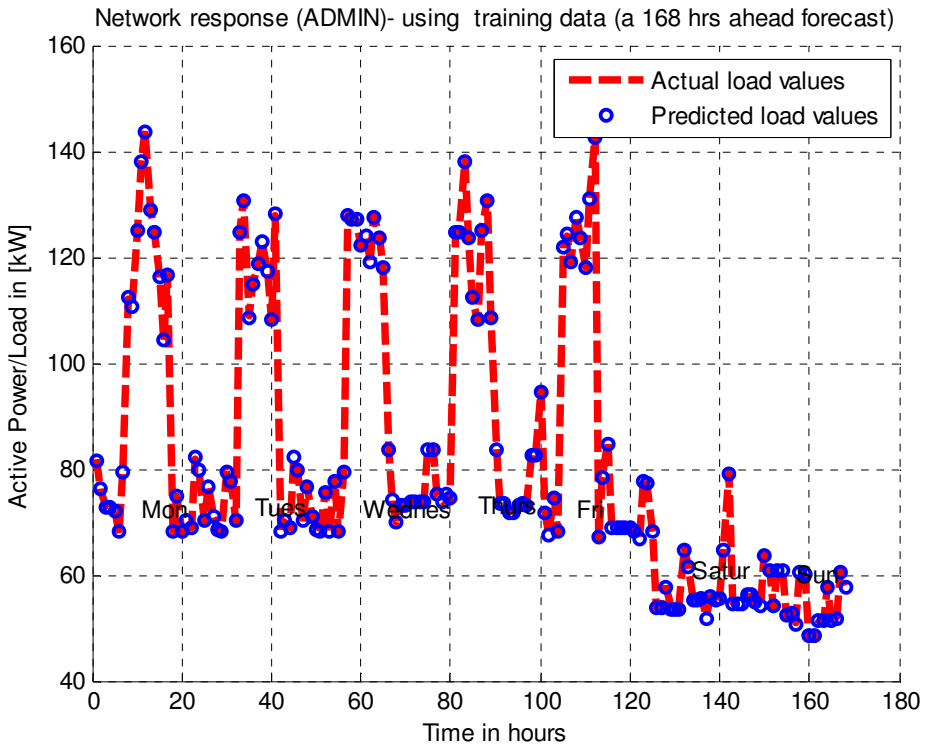


Fig. 7.14(c): One week ahead forecast for the Administration Building (ERNN_ADM_WS-incrementally trained). Forecast week is from the 2nd – 8th of June 2008.

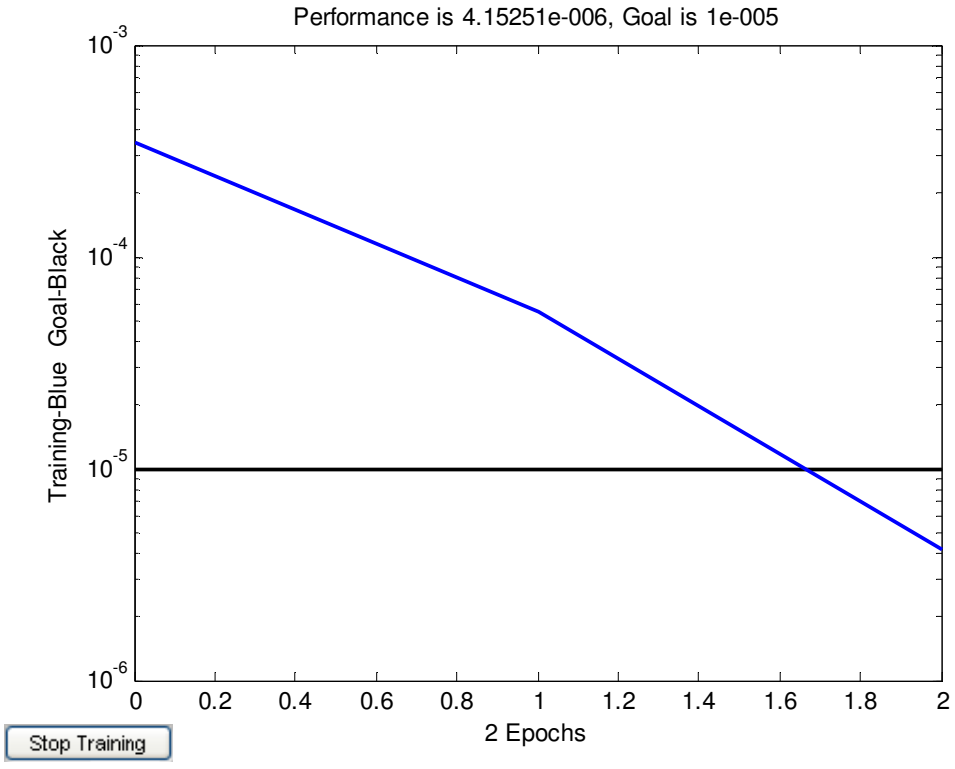


Fig. 7.14 (d): The corresponding model performance.

Some results and performance indicators obtained from the developed ACP load forecasting models are shown in Figure 7.15 (a) and (b).

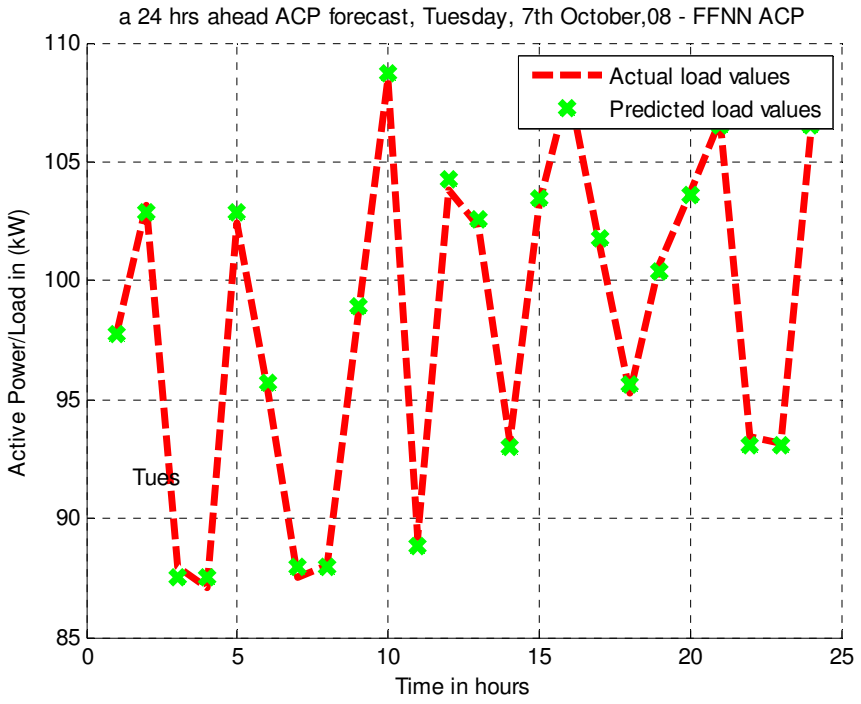


Fig. 7.14(a): One day forecast for the Air Conditioning Plant (ACP) electric load.

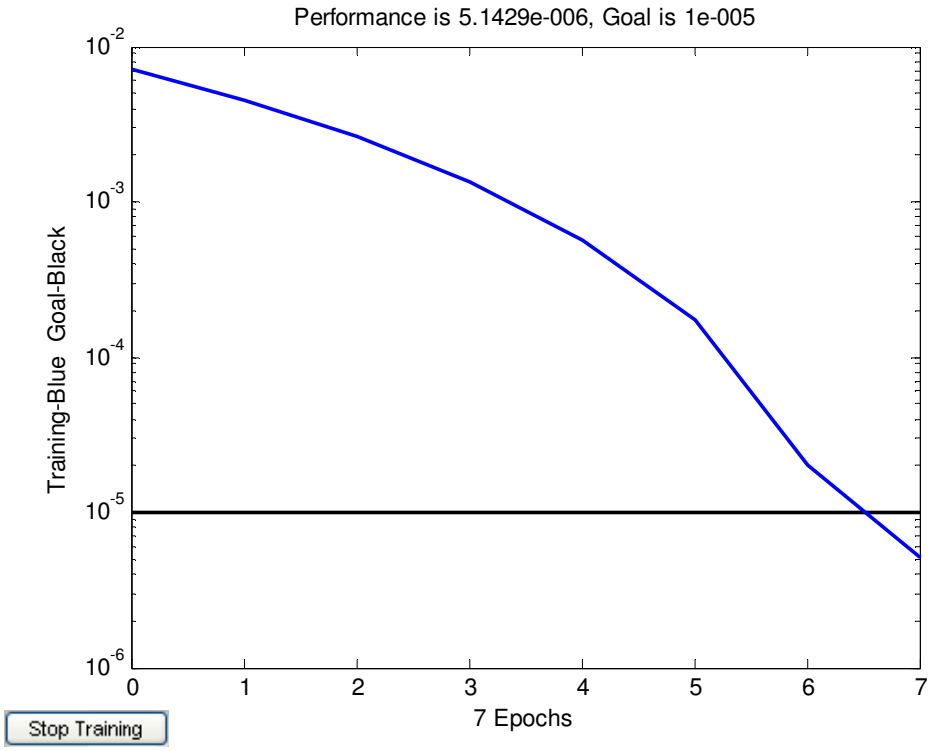


Fig. 7.14 (b): The corresponding model performance.

Actual error (kW) - Elman Recurrent Prediction model (ERNN_ADMIN BLDG) Recurrent Model for predicting the load for the Administration Building							
		batch training mode, V # 1					
Hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	8.57	-1.36	0.88	1.28	-3.75	-0.61	0.61
2	2.12	5.58	-0.73	-0.69	6.27	-3.88	3.37
3	0.64	1.10	-3.43	6.94	3.80	5.18	2.23
4	-1.70	-1.93	4.90	9.37	19.41	6.40	-3.07
5	1.46	-6.41	-5.42	0.88	1.68	-0.53	-8.07
6	-5.28	5.35	4.23	-1.25	-4.24	-12.33	19.43
7	-4.13	6.39	-8.74	-0.14	6.00	-10.78	6.23
8	21.02	-13.57	-10.63	-13.62	-7.09	-6.87	2.27
9	1.11	10.42	12.99	8.86	10.58	-11.85	6.09
10	-10.86	14.71	-0.26	-16.79	-3.32	-12.14	5.82
11	-15.39	-3.41	5.58	6.08	-9.72	-16.03	0.41
12	-0.95	-6.84	0.45	5.90	-2.77	-2.54	1.16
13	-1.25	-8.83	5.62	1.69	0.05	-3.64	-5.93
14	3.22	-0.14	-1.54	-10.52	-8.59	-11.43	5.64
15	-1.68	1.07	6.93	-6.17	-11.84	-9.45	7.73
16	-18.23	-15.71	7.00	13.46	39.21	-7.36	-7.21
17	21.18	32.70	15.76	8.20	-2.94	-7.45	-4.77
18	-11.66	-3.57	7.25	2.76	3.23	-5.82	-3.31
19	-0.09	-0.33	3.35	0.71	10.63	-9.05	-2.45
20	-5.05	-6.93	-2.19	1.30	-2.98	-7.74	2.00
21	-4.08	3.05	-0.69	0.26	-2.74	-0.38	-2.92
22	-7.02	6.93	0.83	0.14	-1.36	13.58	-7.59
23	4.55	-1.85	2.27	-0.75	-0.98	-7.02	0.63
24	5.29	5.29	0.95	-0.73	-0.25	-5.88	3.59

Table 7.5(a): Daily actual errors (kW) for the Feedforward Neural Network batch trained Model (FFNN_ADM)

7.8.6 Error comparative analysis for the departmental load forecasting models

Precisely as the preceding error analysis performed on the total load forecasting models summarized in Table 7.1, the same assessment (i.e. *the MSE and MPE*) was also carried to evaluate the performance of the developed forecasting models for the departmental loads. The result of this analysis is summarized in Table 7.6. The evaluation result also indicates that the incremental training mode is the best approach compared to the latter.

Actual error (kW) - FFNN Prediction model (FFNN_ADMIN) - ADMIN BLDG Feedforward Model for predicting the load for the Administration Building							
		Incremental training mode, IV # 1					
hour	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	0.308401	-0.3488	-0.20587	0.274827	-0.26028	-0.32685	0.280213
2	-0.20334	0.23762	0.242645	0.274827	0.34796	0.145954	-0.14082
3	0.352367	-0.20587	-0.30294	0.367077	0.34796	0.398199	-0.14082
4	0.352367	0.242645	0.274984	0.367077	0.441141	-0.24072	0.125491
5	-0.25905	-0.30294	-0.27598	-0.30842	0.261576	-0.33013	-0.3498
6	-0.15216	0.417887	0.347399	0.143926	0.411227	0.166162	0.344761
7	0.423274	0.174032	-0.34485	-0.12583	0.264772	0.166162	0.292076
8	-0.37596	-0.27959	0.41639	-0.45033	-0.24062	0.14884	-0.23965
9	0.151355	0.205803	0.182094	0.190117	0.202189	0.423628	0.241519
10	-0.16409	0.221708	-0.32716	0.190117	-0.24224	0.423628	0.241519
11	-0.14884	0.252525	-0.32716	-0.15294	-0.18728	0.423628	-0.3129
12	0.209038	0.223377	-0.19032	0.164107	0.314842	0.408119	0.274838
13	0.171505	0.150884	-0.19907	-0.42285	0.405367	0.285107	0.189768
14	0.421134	0.148294	-0.18574	-0.15801	-0.19987	-0.2399	0.364038
15	-0.30649	-0.20906	0.314897	-0.19529	-0.14842	-0.2399	-0.25438
16	-0.43966	-0.329	0.405361	0.207378	0.423408	0.304042	-0.4378
17	0.448729	-0.22809	-0.19987	0.252691	-0.28424	0.406555	-0.4378
18	-0.1826	-0.22652	0.396557	0.286952	0.418555	0.157428	-0.28184
19	0.243557	-0.19508	-0.34309	-0.38273	0.229308	-0.33703	-0.28184
20	-0.24138	0.131459	-0.15516	-0.38273	0.182865	0.265188	0.240012
21	-0.19345	-0.15226	-0.31293	0.152101	0.182865	0.331007	-0.24151
22	0.131279	0.236749	-0.31293	0.152101	0.182865	-0.16617	0.303397
23	-0.15226	-0.3488	0.274827	-0.12673	0.182865	0.280213	0.332372
24	0.236749	0.23762	0.274827	0.349308	0.182865	0.280213	0.293439

Table 7.5(b): Daily actual errors (kW) for the Feedforward Neural Network incrementally trained Model (FFNN_ADM)

7.9 Conclusion

This chapter visibly demonstrates the application of Artificial Neural Networks to a short-term load forecasting problem. The simulation results obtained from different forecasting models are self-explanatory. It is also evident in this chapter that STLF can accurately implemented using only past load values. Most importantly one needs to pay attention to the training approach of the model.

All in all, the error analysis comparison result for both cases (total & departmental loads) shows that the developed hour-by-hour models are reliable, efficient and have a high degree

of generalization as these models responded unconditionally very well to the presented testing data. The hour-by-hour approach also eliminates the need to forecast the load for each day separately.

The context of this chapter also has brought this research work to an end. However, one more epigrammatic section to summarize the work done and highlight the future outlook of this project is discussed in the following chapter.

Model	**Network**	**training mode**	***MSE***		***MPE***	
			training	***Testing***	***training***	***Testing***
FFNN_IT	Feedforward	incremental	5.34E-06	7.34E-04	2.73E-08	4.72E-08
ERNN_IT	Elman Recurrent	batch	8.64E-06	0.000456	9.88E-09	8.53E-09
FFNN_ADM	Feedforward	incremental	3.84E-06	6.84E-03	1.57E-07	3.52E-08
ERNN_ADM	Elman Recurrent	batch	0.003348	0.004357	4.49E-05	6.07E-04
FFNN_MEC	Feedforward	incremental	3.7E-06	5.70E-05	5.46E-07	4.13E-05
ERNN_MEC	Elman Recurrent	batch	0.00118	0.003142	2.87E-05	0.000534453
FFNN_ELEC	Feedforward	incremental	3.51E-06	1.94E-06	1.67E-07	2.83E-06
ERNN_ELEC	Elman Recurrent	batch	0.001074	0.002763	5.23E-06	8.48E-04
FFNN_ACP	Feedforward	incremental	4.01E-06	8.28E-06	2.03E-06	3.50E-09
ERNN_ACP	Elman Recurrent	batch	0.00278	0.0061	4.23E-05	3.85E-04

Table 7.6: Error comparative analysis for departmental load forecasting models

CHAPTER EIGHT

CONCLUSIONS AND FUTURE OUTLOOK

This chapter now concludes the work, application results and findings of the project. In addition, the chapter summarises the comparative analyses of various evolved models and gives a summary of the deliverables of the project. Possibilities to improve the obtained results and to better the features of the developed models are also highlighted.

8.1 Deliverables of the project

The main objective of this research project is to provide power system planners with an accurate and reliable short-term load forecasting (STLF) system which may assist to economically optimize power system operations. In the deregulated electricity market, the key power system operational activities such as priced-based unit commitment (PBUC), energy interchange, and adequate power reserves rely heavily on a forecasting result with reasonable accuracy. Equally, STLF is also essential, especially for large power users (LPUs), to duly manage their notified maximum demands (NMD) and perhaps even better their expansion plans. These requirements are the main impetus of this work.

This thesis is sequentially arranged, commencing with the general introduction of STLF, literature review, some basic requirements of a convenient STLF system, a detailed introduction of the selected technique (ANNs), data collection, and processing, development of models, and finally the obtained application results for the CPUT, Bellville campus reticulation network.

In this work, a number of ANN-based models for STLF have been developed and studied. The development of these models was based on two different neural networks (Feedforward and Elman Recurrent) and trained using real life data as a case study. Table 8.1 presents a complete schedule of network parameters of the developed models.

The load was divided into two categories namely: total load and departmental loads and subsequently forecasted individually. Thus 18 sub models were developed to provide one week in

advance forecasts in two seasons (summer and winter) for the total load. Ten more sub models were developed to hourly and daily forecasts of the departmental loads. As clearly shown in Table 8.1, 10 neurons were used in the hidden layer and 1 neuron in the output layer for the total load forecasting models.

However, the number of hidden layer neuron was later reduced to merely 5 for the departmental load forecasting models. For activations, a tan sigmoid and pure linear transfer functions were used in the hidden layer and output layer respectively. The error is computed by means of the negative of the gradients and conjugate directions using the standard gradient descent (GD) algorithm with momentum and the scaled conjugate gradient (SCG) algorithm. The mean squared error (MSE) has been used to evaluate the performance of the models. Mean percentage error (MPE) was also used as a supplementary measure.

Model type	**Input Vector**	**Network**	**No. of hidden layer neurons**	**No. of output layer neuron**	**Training algorithm**	**Transfer functions**	**Training mode**	**Performance Measure**
NWS and WS	Past load values *and/or* weather data	FF and ER	10	1	Gradient descent algorithm with momentum	Tan sigmoid and pure linear	Batch and Incremental	MSE and MPE
Note: NWS – Non weather sensitive model, WS - Weather sensitive model, FF – Feedforward, ER – Elman Recurrent, MSE – Mean squared error and MPE – Mean percentage error								

Table 8.1: Composition of parameters for the total load models

The forecasting approach was divided into two categories: models for forecasting the load of one week at once (or the batch training approach), and models providing hourly forecasts (previously referred to as incrementally trained models). This differentiation enables one to observe basic properties of different forecasting modules. Generally, the models trained using the batch approach provided fast responses in terms of the total time taken for the forecasting process.

However, these models have some drawbacks particularly in a case of real time implementation. Inter *alia,* some of the disadvantages include the inability of the batch trained models to allow hourly updating or logging of historical load and weather data. This limitation could severely affect the accuracy of the forecast.

Another major shortcoming of these models is the poor accuracy in the forecast, caused by the manner in which the connection weights and biases are computed and adjusted. As explicitly pointed out in chapter 7, the performance of a neural network also depends on how the training and testing data are presented to the network. The result shows that the batch trained approach yields very poor network performance.

The latter training approach (the hour-by-hour), as discussed in chapter 7, section 7.4, looks more attractive despite the delay of extra 60 minutes in obtaining a complete 1 week forecast. The hour-by-hour forecasting approach is not only superior in terms of the accuracy compared to the first approach, but it offers hourly data logging flexibility. And this option is the key feature of STLF system for real-time applications.

The effect of climatic conditions on the load has also been considered in this work. Firstly, the IV # 2, which ignores any weather related effect, was presented to different models using both forecasting approaches. Again, the hour-by-hour trained models outperformed the batch trained ones.

And secondly, the other IV (which brings the effect of weather on the load) was then applied to the networks. Surprisingly, the batch trained models presented with a combination of load and weather data performed slightly better than the non-weather sensitive models (as indicated in section 7.2 of the previous chapter). Nonetheless, the result of the incrementally trained models shows that the inclusion of weather effect has no noticeable impact on the accuracy of the forecasts. Thus it can be reported that STLF can accurately be concluded using only past contiguous load values as defined by the structure of the input vector (IV) # 2.

This finding is also in support with the conclusion made by (Santos *et al.,* 2007) pertaining the optimal number of required training and testing data points. The Authors report that using a large amount of data points does not offer any sorts of advantage with regards to the accuracy of the forecast.

From the case study view, the Cape Peninsula University of Technology (CPUT) management (or the concerned department) needs to urgently review the notified maximum demand (NMD), presently declared at 2.2MW, for the campus to avoid possibly penalties charges payable to City of Cape Town (CoCT) for exceeding the contractual maximum demand threshold.

All in all, various forecasting models have been developed to predict the total load and other departmental loads of the Cape Peninsula University of Technology on hourly to daily basis. The deliverables of this project is summarized in Table 8.2. However, it should be noted only main developed models are illustrated in this Table. For the total load alone, 16 models based on the two training modes have been developed and studied. In all cases, the incremental training approach proved to be extremely superior to the latter training mode.

8.2 Application of the models and results

Although the evolved models are trained using a consumer load data, these models can be slightly modified to accurately forecast the total load of a power utility company. In an area with a prevailing power shortage, like South African region, the first step for utilities to ensure continuity of supply in the region is to acquire good forecasting systems which provide accurate, efficient, and reliable forecasts.

By so doing, power system planners and operators would economically operate complex power system networks by making good strategic decisions in executing main operational tasks such as generation scheduling, optimal energy interchange and risk assessment purposes. It is a known fact that a simple thermal-hydro coordination decision or energy importing/exporting schedule can cost a utility company millions of dollars if poorly derived. And all these decisions are mainly based on the forecast, thus accurate forecasting systems are undeniably essential.

Network	Model type	Type of load	Season	Training mode	Model ID
Feedforward	Weather sensitive	*Campus' total load*	Summer	Batch	FFNN_TL_WS01
				Incremental	
			Winter	Batch	FFNN_TL_WS02
				Incremental	
Feedforward	Non-weather sensitive	*Campus' total load*	Summer	Batch	FFNN_TL_NWS01
				Incremental	
			Winter	Batch	FFNN_TL_NWS02
				Incremental	
Elman Recurrent	Weather sensitive	*Campus' total load*	Summer	Batch	ERNN_TL_WS01
				Incremental	
			Winter	Batch	ERNN_TL_WS02
				Incremental	
Elman Recurrent	Non-weather sensitive	*Campus' total load*	Summer	Batch	ERNN_TL_NWS01
				Incremental	
		Campus' total load	Winter	Batch	ERNN_TL_NWS02
				Incremental	
		Departmental loads			
Feedforward	Weather sensitive	*IT department*	Summer	Incremental	FFNN_IT
Elman Recurrent				Incremental	ERNN_IT
Feedforward	Weather sensitive	*Electrical department*	Summer	Incremental	FFNN_ELEC
Elman Recurrent				Incremental	ERNN_ELEC
Feedforward	Non-weather sensitive	*Mechanical department*	Summer	Incremental	FFNN_MEC
Elman Recurrent				Incremental	ERNN_MEC
Feedforward	Weather sensitive	*Administration building*	Summer	Incremental	FFNN_ADM
Elman Recurrent				Incremental	ERNN_ADM
Feedforward	Non-weather sensitive	*Air Conditioning Plant (ACP)*	Summer	Incremental	FFNN_ACP
Elman Recurrent				Incremental	ERNN_ACP

Table 8.2: A schedule of the developed ANN-based electric load forecasting models

As for power consumers, especially LPUs, it is extremely important to manage their load to consolidate demand management strategies enforced by utility companies to address the known power crisis situation in South African region. This implies forecasting the load accurately will aid to facilitate the power demand management process. This exercise also prevents possible penalty charges for exceeding the declared maximum demand payable by consumers. For the CPUT in particular, the obtained forecasts can be used as a guideline to formulate internal load management strategies, such as emergency power generation and implementation schedule for demand side management initiatives.

The developed models can also used as a part of the postgraduate course materials for soft computing and further research in the fields of ANNs, prediction and control.

8.3 Project outlook

The scope of work of this project was specifically limited to off-line training. But for real-time applications, some matters still need to be carefully addressed during model development process. To start with, the existence of bad data (outliers) in the historical load curve as well as in the weather data can affect the accuracy of the forecast negatively. In this work, a manually based strategy for detecting and replacing bad data was employed. However, this approach is inappropriate for real-time implementation, thus a technique aimed at identifying and replacing abnormal data in the input variable curves needs to be automated.

For a comprehensive model performance evaluation, it would be a good idea to develop a genetic-based model and then compare the subsequent errors as well as network performance indices for further assessment or benchmarking purposes.

Another matter that needs to be reviewed is the use of more precise hourly weather data rather than the daily average values. For this, measurement apparatus may need to be installed on the campus premises to record the required weather data on hourly basis.

Evidently, the evolved models have numerous features equivalent to a good forecasting system, but a user friendly graphical user interface (GUI) can be developed as an attempt to guide the user throughout the forecasting process.

8.4 Publications in connection with the thesis

1. S.Amakali and R. Tzoneva "*A consumer-based short-term load forecasting using artificial neural networks"*, 2008 – submitted to SAIEE Africa Research Journal.
2. S.Amakali and R. Tzoneva "S*hort-term electric load forecasting: Review of the methods and some results of a distribution network"*, 2008 – submitted to PAC magazine.

REFERENCES:

E.A. Feinberg and D. Genethliou (2005) *"Load forecasting In: Applied Mathematics for Restructured Electric Power Systems": Optimization, Control, and Computational Intelligence*, J.H. Chow et al. (eds.), Springer.

M. EL-Naggar, and A. AL-Rumaih (2005)"*Electric Load Forecasting using Genetic Based Algorithm, Optimal Filter Estimator and least error squares Technique", Comparative study*, PWASET, pp. 138 -142

Ping-Feng Pai (2006) *"Hybrid ellipsoidal fuzzy systems in forecasting regional electricity loads" Energy Conversion and Management*, *Volume 47, Issues 15-16*, *September 2006*, *Pages 2283-2289*

Janacek G and Swift L (1993) *Time Series Forecasting, Simulation, Application*. Ellis Horwood, New York.

D.W. Bunn and E.D. Farmer (1985) *"Comparative models for electrical load forecasting"*, John Wiley and Sons, New York. pp.232

Gross, G and Galiana, F.D (1987), "*Short-Term Load Forecasting",* Proceedings of the IEEE, Vol.75, No.12, pp. 1558-1572.

Mohsen Hayati and Yazdan Shirvany (2007) *"Artificial Neural network Approach for Short-term load forecasting for Illam Region",* Volume1 Number 2, *WASET ORG*

S.S. Sharif and J.H. Taylor (2000) *"Short-term Load Forecasting by Feed Forward Neural Networks"*, Proc. IEEE ASME First Internal Energy Conference (IEC), Al Ain, United Arab Emirate, *http://www.ee.unb.ca*

S.S. Sharif and J.H. Taylor (2000) *" Real-time Forecasting by Artificial Neural Networks" Proc. IEEE Power Engineering Society Summer Meeting (PES), Seattle, Washington*

K. Y. Lee and J. H. Park (1992) "*Short-Term Load Forecasting Using an Artificial Neural Network*" IEEE Trans. Power Systems, vol. 7, no. 1

Hong Chen, Caludiom A. Canicares, and Ajit Singh, (2002) *"ANN-based Short-term load forecasting in Electricity Markets" Proc. IEEE Power Engineering Society Winter Meeting.*

T, Rashid and T. Kechadi (2005) *"A practical approach for electricity load forecasting"* Proceedings of world academy of science, engineering and technology volume 5, *http://www.waset.org*

Mandal P, Senjyu T, Urasaki N, Funabashi T, Srivastava AK. A novel approach to forecast electricity price for PJM using neural network and similar days method. *IEEE Transactions on Power Systems* 2007; **22**(4): 2058-2065.

Tawfig Al-Saba, and Ibrahim El-Amin (1999) *"Artificial neural networks as applied to long-term forecasting, http://www.elsevier.com*

Pedro. A. Ganzalez, and Jësus M. Zamarreno (2004) *"Prediction of hourly energy consumption in building based on a feedback artificial Network. Energy and Buildings*, *Volume 37, Issue 6*, *June 2005*, *Pages 595-601 http://www.elsevier.com*

Dipti Srinivasan (1998) *"Evolving artificial neural networks for short-term load forecasting ", Neurocomputing*, *Volume 23, Issues 1-3*, *7 December 1998*, *Pages 265-276 www.ee.nus.edu.sg*

Paras Madal, Tomonobu Senjyu, Naomitsu Urasaki, and Toshihisa Funabashi (2006) *"A neural network based on 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 http://www..elsevier.com*

Ayca Kumluca Topalli, Ismet Erkme, and Ihsan Topalli (2006) "*Intelligent short-term load forecasting in Turkey" International Journal of Electrical Power & Energy Systems*, *Volume 28, Issue 7*, *September 2006*, *Pages 437-447 http://www..elsevier.com*

Nahi Kandil, Rene Wamkeue, Maarouf Saad, and Semaan Georges (2006) *"An efficient approach for short-term load forecasting using neural networks" International Journal of Electrical Power & Energy Systems*, *Volume 28, Issue 8*, *October 2006*, *Pages 525-530 http://www..elsevier.com*

Zhi Xiao, Shi-Jie Ye, Bo Zhong and Cai-Xin Sun (2007) *"BP neural network with rough set for short-term load forecasting" Expert Systems with Applications*, *In Press, Corrected Proof.*

Philippe Lauret, Eric Fock, Rija N. Randrianarivony, Jean-Francois Manicom- Ramsamy (2007) "Bayesian neural network approach to short time load forecasting" *Energy Conversion and Management*, *Volume 49, Issue 5*, *May 2008*, *Pages 1156-1166*

J. Al-Shareef, E. A. Mohamed, and E. Al-Judaibi (2008) "*One Hour Ahead Load Forecasting Using Artificial Neural Network for the Western Area of Saudi Arabia"* Proceedings of world academy of science, engineering and technology volume 27

Mohsen Hayati, and Yazdan Shirvany (2007) "*Artificial Neural Network Approach for ShortTerm Load Forecasting for Illam Region*" International journal of electrical, computer, and systems engineering volume 1 number 2.pp.1307-5179

B. Satish, K. S. Swarup, S. Srinivas and A. Hanumantha Rao (2004) "*Effect of temperature on short term load forecasting using an integrated ANN*" *Electric Power Systems Research*, *Volume 72, Issue 1*, *15 November 2004*, *Pages 95-101*

Luciano S. Moulin and Alexandre P. Alves da Silva, 1999 *"Neural Network based short-term eleteric load forecasting with confidence intervals", proceedings of the IV Brazilian conference on neural networks, pp. 007 – 012*

Bahman Kermansllahi and Hiroshi Iwamiya, 2002 *"Up to year 2020 load forecasting using neural nets",* International Journal of electrical power & energy systems, volume 24, pp:789-797

Rang Gao and Lefteri H. Tsoukalas, 2001 *"Neural-wavelet Methodology for load forecasting"* Journal of intelligent and robotic systems, pp: 149-157

Axay J. Mehta, Hema A. Mehta, and T.C. Munjunath, 2003 *"A multi layer artificial neural network architecture design for load forecasting in power systems",* Internal journal of applied mathematics and computer sciences, volume 4, number 4, pp:227-240

Moghram I, Rahman S (1989) Analysis and evaluation of five short-term load forecasting techniques. IEEE Trans Power Syst, 4:1484–1491

A.M. Kalteh, P. Hjorth and R. Berndtsson, 2007 "*Review of the self-organizing map (SOM) approach in water resources: Analysis, modelling and application",* Environmental Modelling & Software, Volume 23, Issue 7, July 2008, Pages 835-845
Iren Valova, Daniel Szer, Natacha Gueorguieva, and Alexandre Buer, 2005 *"A parallel growing architecture for self-organizing maps with unsupervised learning"* NeurocomputingVolume 68, October 2005, Pages 177-195

Gianluigi Rech, 2002 *"Forecasting with artificial neural network models"* SSE/EFI Working paper Series in Economics and Finance, No 491, pp: 1 – 36

Peter Raicevic and Christopher Johansson, 2001 "*Biological Learning in Neural networks"* Neurovetenskap – från jonkanal till kognition Del II

J.B. Alam, C.K. Sarkar and E.U. Ahmed, 2007, "Study on thickness of two-way slab" ASIAN JOURNAL OF CIVIL ENGINEERING (BUILDING AND HOUSING) VOL. 8, NO. 5, PAGES 573-579

Purnia, V.S. and Patodi, S.C., A Counterpropagation Neurocomputing Approach for Analysis and Optimum design of Slab. *IE (I) Journal*, 82(2001).

Lendasse, J. Lee, V. Wertz, M. Verleysen, 2000 *"Time Series Forecasting using CCA and Kohonen Maps - Application to Electricity Consumption"*, ESANN'2000 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 329-334.

J. N. Fidalgo, "*Previsão de carga em saídas de subestações- resultados perliminares,"* presented at the 4º encontro Luso-Afro-Brasileiro de Planejamento e Exploração de Redes de Energia- ELAB'99, Rio de Janeiro, Jun 7-10, 1999, Paper ST 7-2.

P.J. Santos, A.G. Martins, A.J. Pires, J.F.Martins, and R.V. Mendes, "*Short Term load forecast using trend information and process reconstruction*", International Journal of Energy Research, 2006; 30:811-822

H. S. Hippert, C. E. Pereira and R. Castro Souza, "*Neural networks for short-term load forecasting:* A review and evaluation," IEEE Trans. on Power Systems, vol. 16, pp. 44-55, Feb. 2001.

T.Gowri Monohar and V.C. Veera Reddy, *"Load forecasting by a novel technique using ANN"*, ARPN Journal of Engineering and Applied Sciences, VOL. 3, NO. 2, April 2008

Sanjib Mishra and Sarat Kumar Patra *"Short Term Load Forecasting using Neural Network trained with Genetic Algorithm & Particle Swarm Optimization"* First International Conference on Emerging Trends in Engineering and Technology, IEEE Computer Society, Vol.94, DOI 10.1109/ICETET,2008

C. Kriger, "Prediction of the influent wastewater variables using neural network" Master Thesis, 2007

L. Davis, "*Handbook of Genetic Algorithms",* New York: Van Nostrand Reinhold, 1991.

C. B. Lucasius and G. Kateman, "Towards Solving Subset Selection Problems with the Aid of the Genetic Algorithm", In *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick, (Eds.), pp. 239-247, Amsterdam: North-Holland, 1992.

R. C. Eberhert and J. Kennedy, "A new optimizer using particle swarm theory", *Proceeding of the Sixth International Symposium on Micro Machine and Human Science*, page 39-43,, Nagoya, Japan, 1995, IEEE Service Center, Piscataway, Nj.

R. B. Holstien, *Artificial Genetic Adaptation in Computer Control Systems*, PhD Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1971.

D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization", *Proc. ICGA 2*, pp.41-49, 1987.

H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive Models for the Breeder Genetic Algorithm", *Evolutionary Computation*, Vol. 1, No. 1, pp. 25-49, 1993.

M. F. Bramlette, "Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization", *Proc ICGA 4*, pp. 100-107, 1991.

K. A. De Jong, *Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, PhD Thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.

W. M. Spears and K. A. De Jong, "An Analysis of Multi-Point Crossover", In *Foundations of Genetic Algorithms*, J. E. Rawlins (Ed.), pp. 301-315, 1991.

L. Booker, "Improving search in genetic algorithms," In *Genetic Algorithms and Simulated Annealing*, L. Davis (Ed.), pp. 61-73, Morgan Kaufmann Publishers, 1987.

D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Publishing Company, January 1989.

Tomothy Maifeld and Gerald Sheblé, *"Short-term load forecasting by a neural network and a refined genetic algorithm*", Electric Power Systems Research 31 (1994) 147 – 157

G.A.Adepoju, S.O.A Ogunjuyigbe, and K.O Alawode *"Application of Neural Network to Load Forecasting in Nigerian Electrical Power System*" The Pacific Journal of Science and Technology, Volume 8, Number 1, May 2007.

P.J.Santos, A.G. Martins, and A.J. Pires *"Designing the input vector to ANN-based models for short-term load forecast in electricity distribution systems"* International Journal of Electrical Power and Energy Systems, 2007 - Elsevier

J.T. Tou and Gonzalez, RC, (1974)"Pattern Recognition Principles", Reading, MA:- Addison-Wesley

D.Shen *"Computation of correlation coefficient and its confidence interval in SAS"*, WCI, SUGI31, posters, 1999

V.V. Phansalkar, *"Analysis of the Back-Propagation Algorithm with Momentum",* IEEE Trans. on Neural Networks,Vol. 5, No. 3, May 1994.

K.L. Ho, *"Short Term Load Forecasting Using a Multilayer neural Network with an AdaptiveLearning Algorithm",* IEEE Trans.on Power Systems, Vol. 7, No. 1, pp. 141-149, Feb. 1992.

J.M. Zurada, *"Introduction to Artificial Neural Systems",* West Publishing Company, 1992.

Mazzoni, P., R.A. Andersen, and M.I. Jordan, *A more biologically plausible learning rule for neural networks.* Neurobiology, 1991. **88**: p. 4433-4437.

Lendasse, J. Lee, V. Wertz, M. Verleysen, 2000 *"Time Series Forecasting using CCA and Kohonen Maps - Application to Electricity Consumption"*, ESANN'2000 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 329-334.

Hollstein, R.B. (1971): *Artificial genetic adaptation in computer control systems*, PhD Thesis, Univ. of Michigan

Ackley, D. H. (1997): *A Connectionist Machine for Genetic Hillclimbing*, Boston, MA: Kluwer Academic Publishers.

Joe Ross and Graeme Millis, 2008, *Demand Side Management System*, A power meter technics White Paper, *www.meteringonline.com,*

Say, M.G *"Electrical engineer's reference book",* 13th ed., 1973

Boake, Ian Gordon, *"A strategy for electricity load management in the south African mining industry"*, Doctoral Thesis, 2003

Hebb, D.O., *The Organization of Behavior*. 1949, New York: John Wiley Inc.

National Institute of Economic and Industry Research (NIEIR), The demand forecasts and new consumer usage, 2007, *www.nieir.com.au/*

Yang, Jingfei and Juergen Stenzel, *"Short-term load forecasting with increment regression tree",* Electric power systems research, Volume 76, Issues 9-10, June 2006, pages 880-888

Howard Demuth, Mark Beale and Martin Hagan, *"Neural Network Toolbox ver. 6.0, User's Guide "*, The MathWork Inc, march 2006

Crowford, William Jeffrey *"An Ensemble and Modular Neural Network Approach to the Diagnosis of Acute Appendicitis",* Master Thesis, 2000

MySQL AB. Why MySQL?, (http://www.mysql.com/why-mysql)

J.M. Paruelo and F. Tomasel, *Prediction of functional characteristics of ecosystems: a comparison of artificial neural networks and regression models*, *Ecol. Model.* **98** (1997), pp. 173–186.

Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*. 1991: Addison-Wesely.

T.S. Dillon, *Short-term load forecasting using anadaptive neural network*, Electrical Power & Energy Systems, pp. 186-191, 1991.

K.Y. Lee, *Short-Term Load Forecasting Using an Artificial neural Network, IEEE Trans. On Power Systems,* Vol. 7, No. 1, pp. 124-131, Feb.1992.

M.Djukanvic,*Unsupervised/supervised learning concept for 24-hour load forecasting,* IEEE Proc.-C, Vol. 140, No. 4, pp. 311-318, July,1993.

Bahman Kermanshahi and Hiroshi Iwamiya, *"Up to year 2020 load forecasting using neural nets"* International Journal of Electrical Power & Energy Systems,Volume 24, Issue 9, November 2002, Pages 789-797

K.L. Ho, *"Short Term Load Forecasting Using a Multilayer neural Network with an Adaptive Learning Algorithm"*, *IEEE Trans.on Power Systems,* Vol. 7, No. 1, pp. 141-149, Feb. 1992.

S.Alvisi, G. Mascellani, M. Franchini and A. Bardossy, "*Water level forecasting through fuzzy logic and artificial neural network approaches"*, Hydrology and Earth System Science Discussions, 2, pp 1107 – 1145, June 2005

BIOGRAPHY

Simaneka Amakali, born in Etope, Oshakati, Namibia on July 18, 1982 is a registered incorporated engineer (in-training) with the Engineering Council of Namibia (ECN). He received the B.Tech Degree in Electrical Engineering from the Polytechnic of Namibia in 2006 in First Class Distinction.

He has worked in the electricity distribution industry as a project engineer for 2 years, with majority of work concentrating on providing new consumer supplies on the medium voltage (11kV, 22kV, 33kV) networks including single wire earth return (SWER) network.

He has been actively involved in the projects starting with preliminary planning, estimation, budgeting, design and specification, tendering, tender allocation, construction and contractor management, payment certification up to final delivery. Besides this he has also been involved in project supervision and planning of urban distribution and supply networks. During this time he has also developed a closer understanding of tariff structuring as well as operations and maintenance issues.

His research interests are in the areas of power systems planning, with particular emphasis in electric load forecasting.

APPENDICES

APPENDIX A

CPUT Bellville Campus-

Substations geographical layout and single line diagram for the 11kV reticulation network

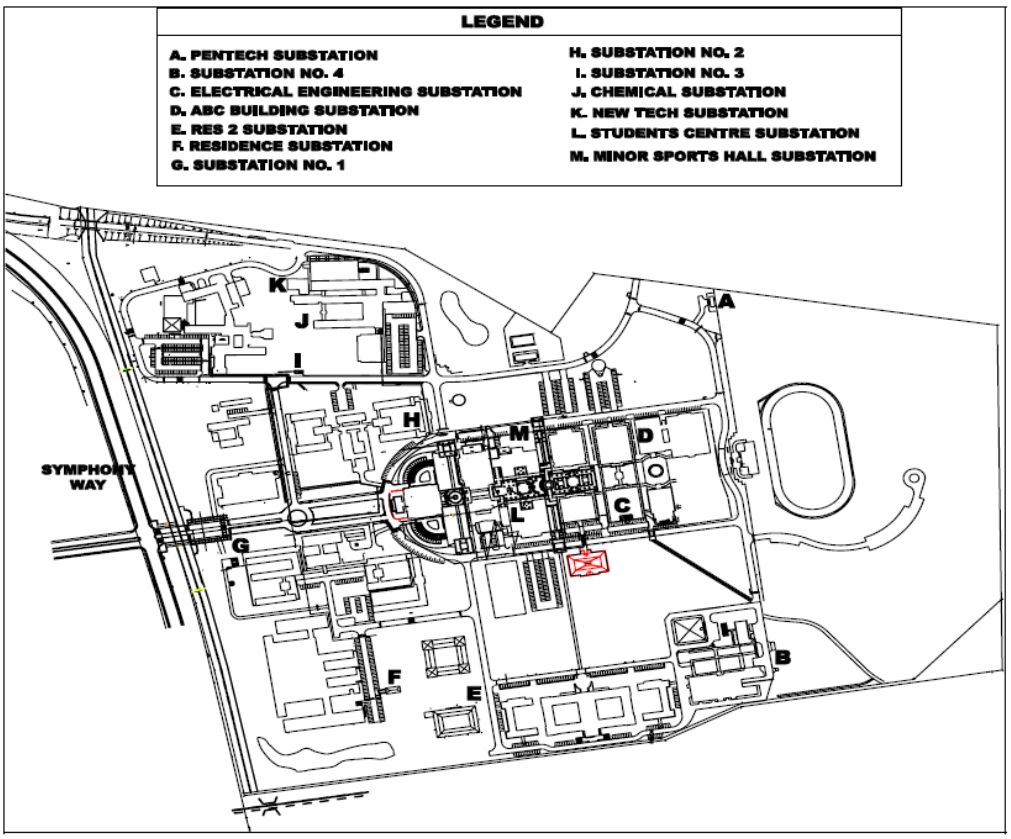
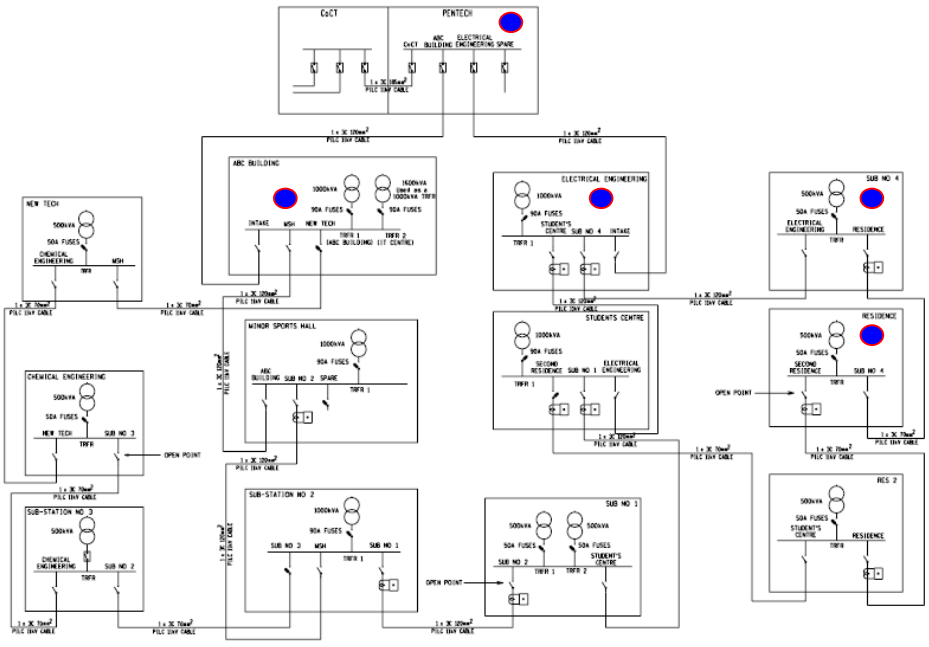


Fig. A 1: Substations geographical layout



- *symbolises a designated point for measurements where historical profile data were captured-*

Fig. A 2: single line diagram – for the 11kV reticulation network

APPENDIX B

Detected Bad Load and Weather Data

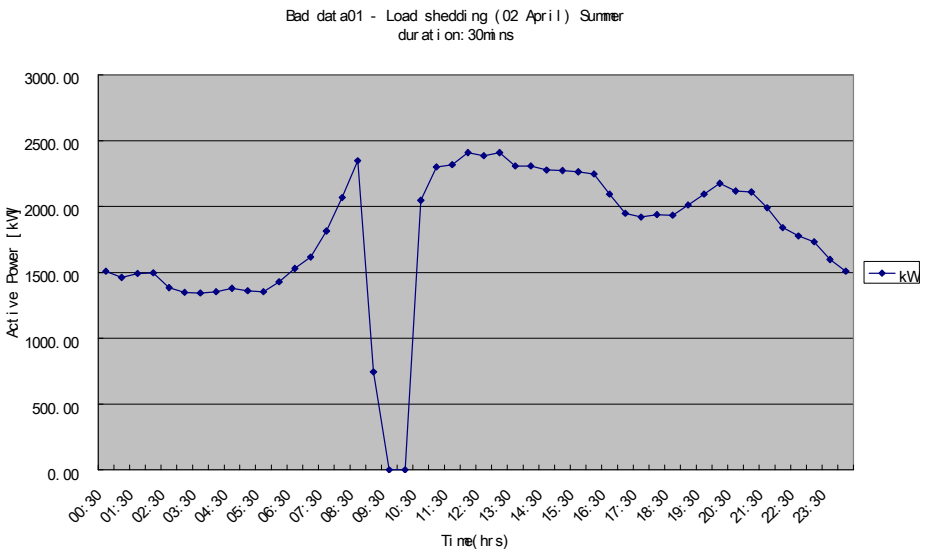


Fig. B.1: Bad data recorded on 02 April, 08 at 9h30.

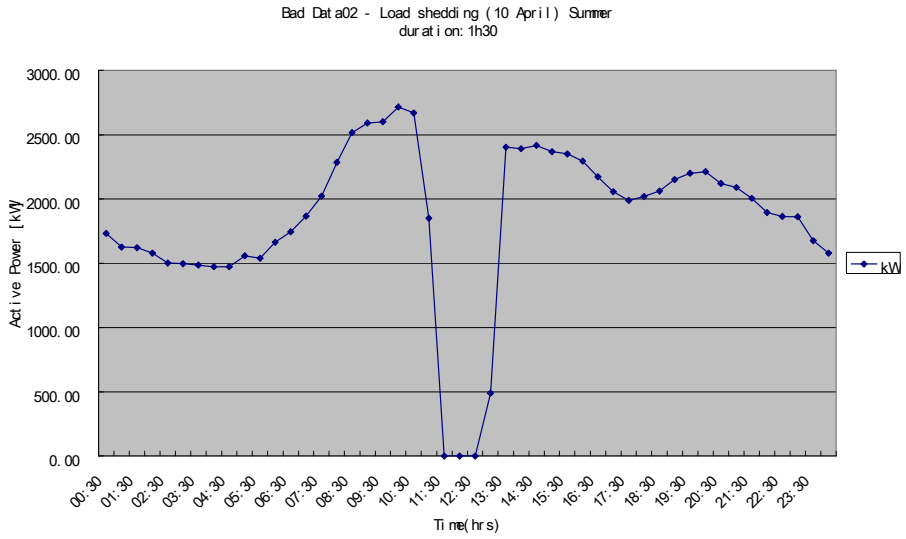


Fig. B 2: Bad data recorded on 10 April, 08 at 11h30.

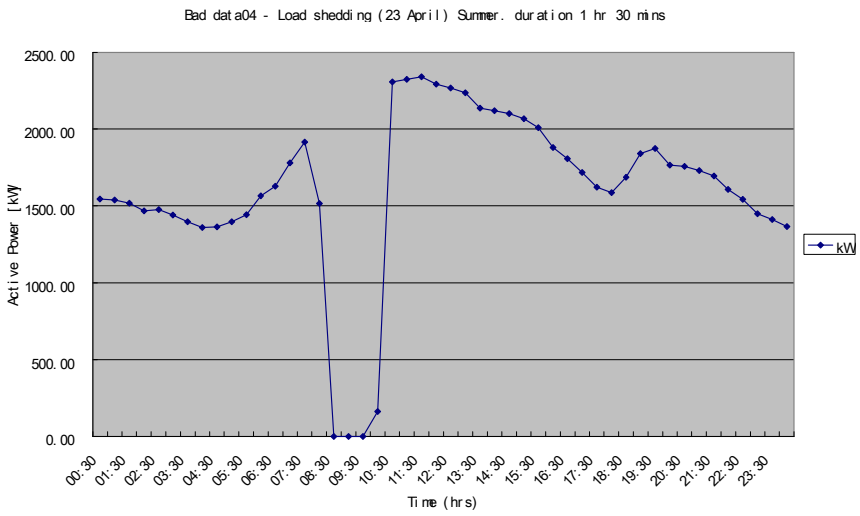


Fig. B 3: Fig. Bad data recorded on 23 April, 08 at 8h30.

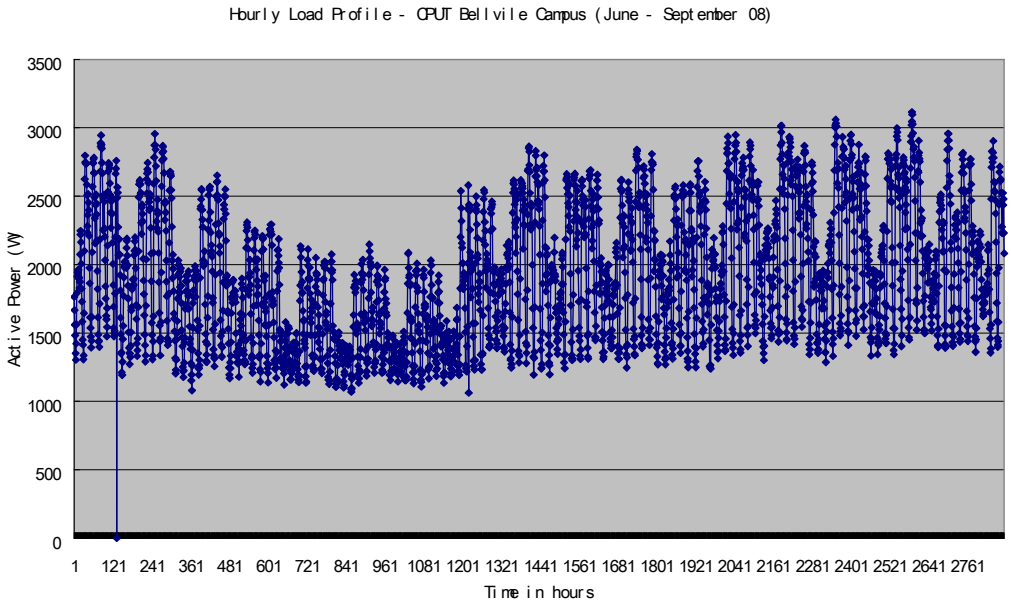


Fig. B 4: Bad data recorded on 06 June, 08 at 11h30.

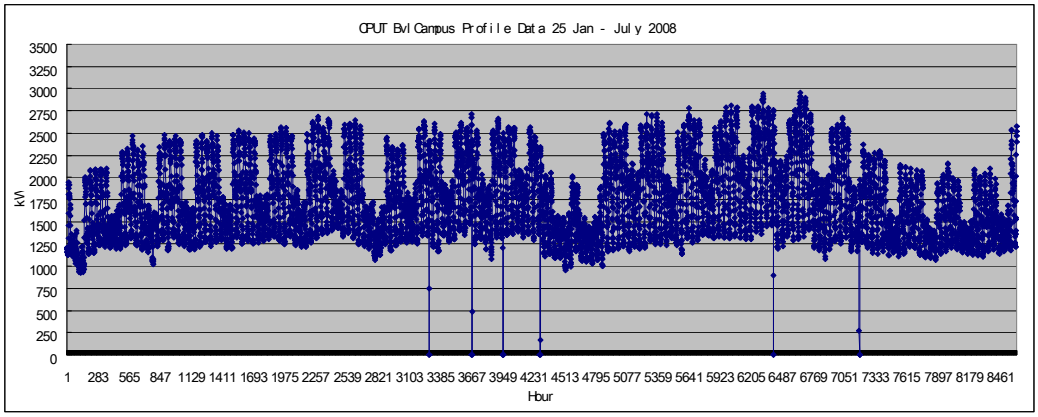


Fig. B 5: Total load profile – including bad data as a result of Eskom' load shedding

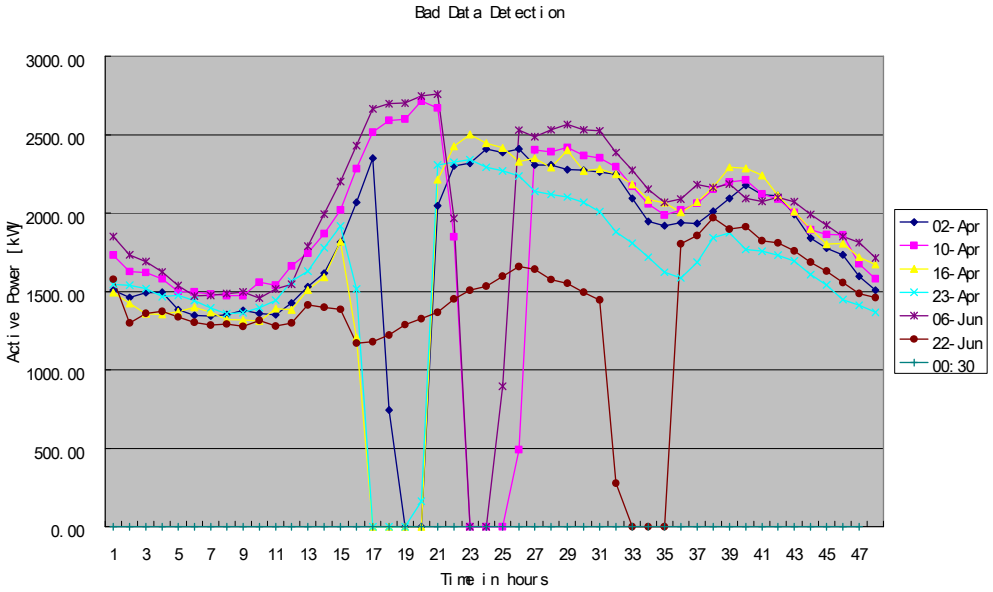


Fig. B 5: Some selected bad data for the total load.

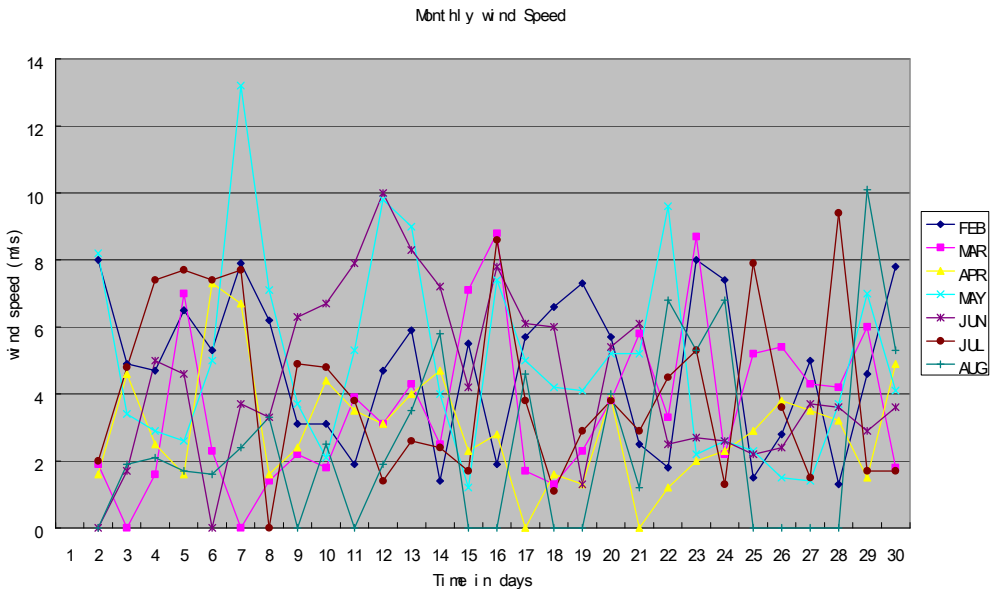


Fig. B 6: Detected bad data – wind speed

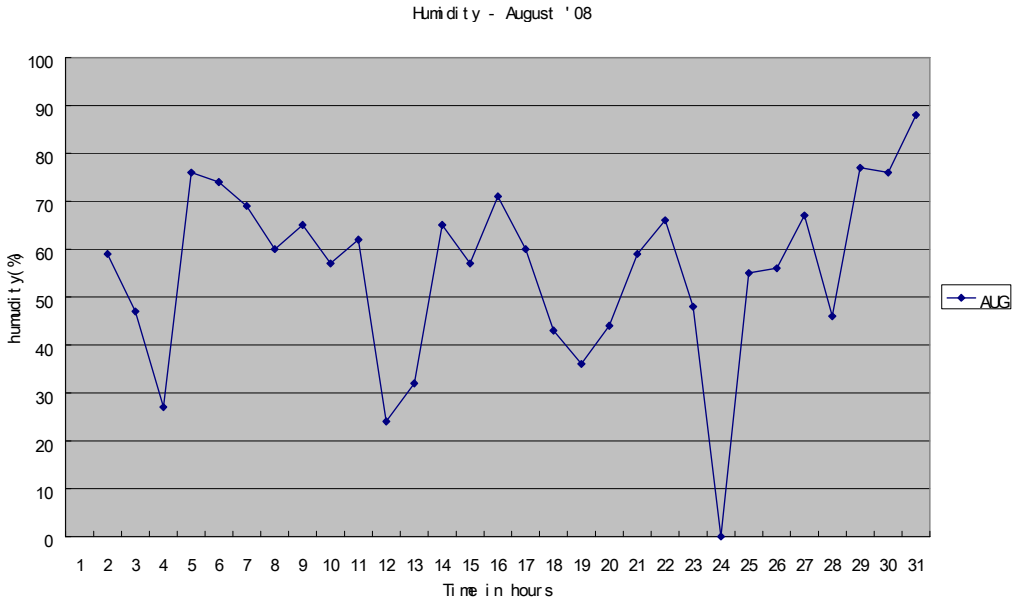


Fig. B 7: Detected bad data – humidity

APPENDIX C

Setting up and programming a DMK 40 Lovato and

Technical specifications

(Digital Meter for load measurements)

Setting up and programming a DMK 40 Lovato Digital Meter –

Measurement for departmental loads

1. Software installation

The DMK 40 Digital meter with **data logger options** comes by default with the software.

The installation of this very simple-

2. Communication options

Use Serial port RS232/RS485 communication interface – to connect the meter to a local PC.

Please also ensure that the Aux supply is connected and functional.

3. Access or Password

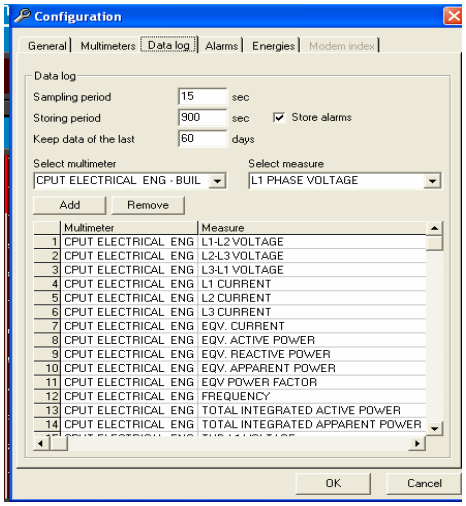
For a meter reader – no password is required.

Engineering password (full rights) to reset and re-program the meter – please click on *password* in the main menu and type in "*lovato*"

4. Programming – record data

To specify measurement parameters - Click on *Configuration* in the tool bar and the *options*

The window shown below will appear. Click on *Data log* in the tool bar



In this window you can then specify what exactly you want measure and the period of measurements etc. And then click *ok*.

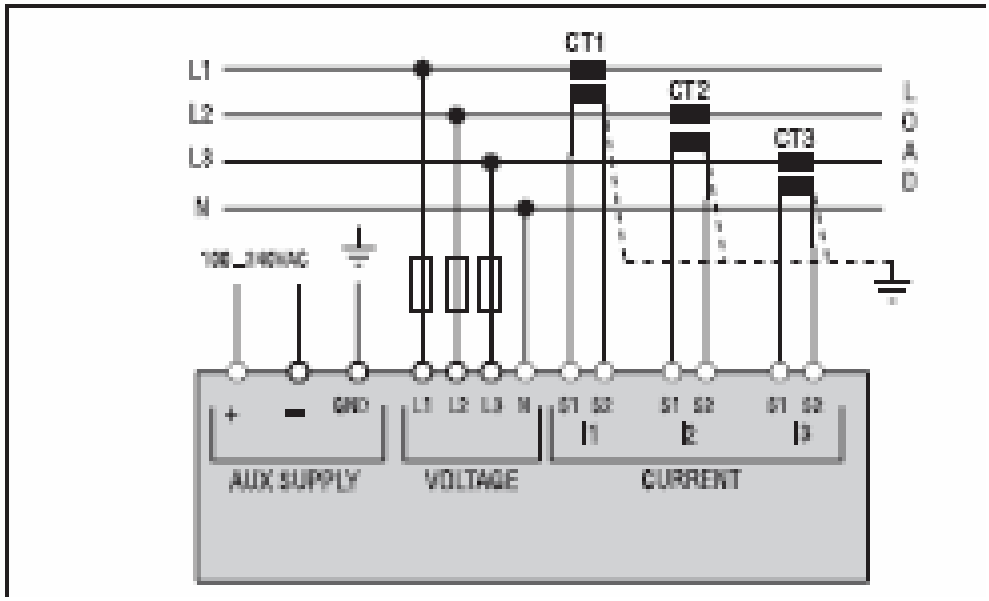
5. Programming – technical data

To set up the installation – please click on *parameters* in the main menu, and then click on *base setup*.

The DMK 40 requires CT ratio programming (typically 1-5A max). For voltage measurements less than 830VAC/DC – no need for PT ratio programming. More technical data for this meter – please refer to the enclosed technical data sheet.

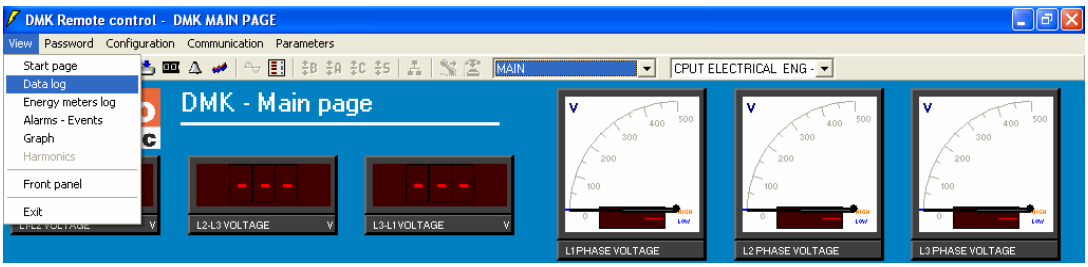
6. Installation and Commissioning

As shown in the figure below – the following connections were made to present voltage and current signals to the meter. Moreover, the meter also requires a separate AUX supply: 100 – 240V AC / 110 – 250V DC.

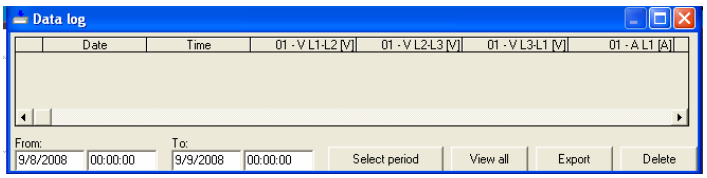


7. Data Logging and Storage

To view the recorded data – please click on *View* in the main menu and then click on *Data Log*.

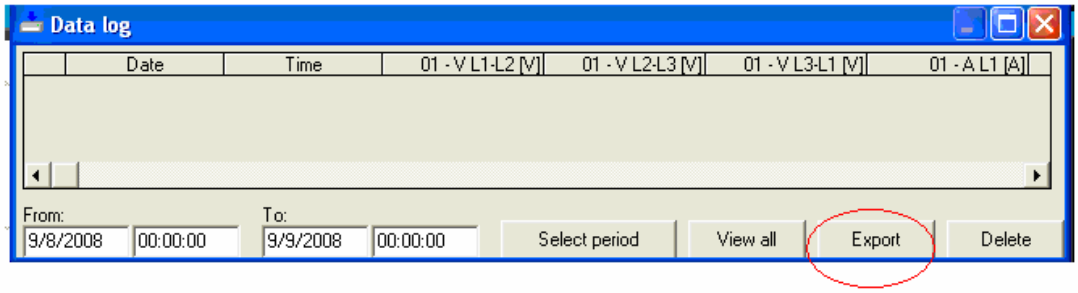


A window showing all records will then pop-up.

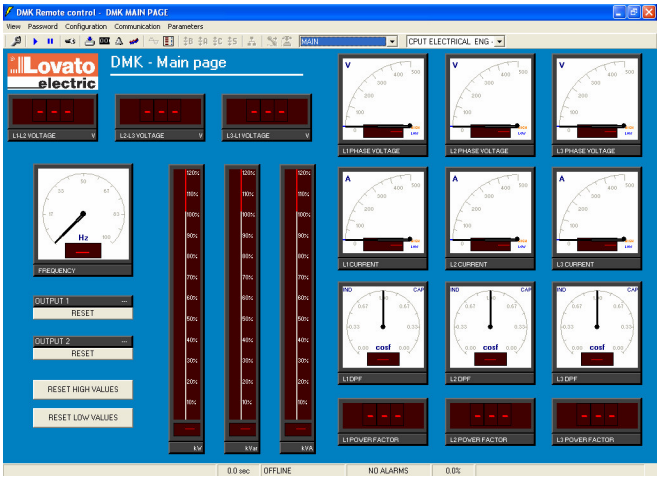


8. Uploading Logged Data

Before exporting saved data, click (View all) – to display all records and then export to either a text file or Excel spread sheet.



9. DMK Main Page



10. Data retrieval note

Because this meter has no internal memory backup – please ensure that the connection to a local PC is always available. For this, one needs to have a roving PC merely for measurements. Otherwise, if one disconnects the meter before uploading the data – all data will be lost.

The author hopes that you will find this guide useful – good luck.

By: S. Amakali, CPUT, 2008

Digital multimeters 251 electric parameters

DMK 40

DMK 40	Version with data-logger, RS232 and RS485 opto-isolated ports, 100-240VAC / 110-250VDC	1	0.470

General characteristics

These multimeters comprise excellent features, superior to devices of the same category currently on the marketplace. Distorted waveform conditions, such as very disturbed electric lines having voltage and currents with high harmonic content and variable frequency, do not influence the high accuracy DMK multimeter readouts because of rigorous design in addition to the use of the latest generation of microprocessor technology.
Measurement of the phase angle (cosφ) in addition to power factor, harmonics analysis and HIGH-LOW-MAX functions are just a few of those which are difficult to find on higher category equipment.
The DMK 40 version includes a reliable data-logger system, extremely easy to use.
DMK 3..., DMK 40 and DMK 6... digital multimeter can display more than 251 measurements; a few of these are listed below.

- Voltage: phase, line and system values
- Current: phase and system values
- Power: active, reactive, apparent phase and total values
- Energy: import, export, inductive and capacitive values
- P.F.: power factor per phase
- cosφ: angle displacement, i.e. power factor related to the harmonic fundamental only
- Frequency of measured voltage value
- Harmonics (HARM.): residual and total harmonic content for each harmonic order up to the 22° per phase, both for voltage and current values
- HIGH / LOW: maximum / minimum values of phase voltage and current and ΣW, Σvar and ΣVA power
- Maximum (MAX): maximum current and total active power values, both calculated on programmable integration time.

The technical features of the DMK 40 data-logger are:

- 2Mbyte (MB) non-volatile memory for data logging
- Real Time Clock (RTC) with replaceable back-up lithium battery
- Sampling time, 1s to 24h configurable
- Number of sampling measures, 1 to 32 configurable at a time
- Communication protocols: Modbus®-RTU and ASCII
- Data logging of one electric parameter in continuous format or with begin and end by programmable thresholds
- Suspension of data acquisition at full memory or refreshing of oldest data.

Operational characteristics

- Operating range: 85-265VAC; 93.5-300VDC
- Voltage measure range: 20-830VAC phase-phase
  10-480VAC phase-neutral
- PT ratio programming: 1.0-5,000
- Frequency measure range: 45-65Hz
- CT ratio programming: 1.0-2000
- Voltage, current, frequency and harmonic distortion accuracy: ±1 digit
  Energy measure accuracy: Class 1 to IEC/EN 61036 and IEC/EN 61268
- HIGH and LOW value functions to detect and log instantaneous voltage, current and power values
- Averaging function to slow down repetitive fluctuations to obtain more stable readouts
- Current connection in ARON configuration via 2 current transformers only
- Single, two, three phase with or without neutral and balanced three-phase connection via 1 current transformer only
- Usage with voltage transformers for voltages >830VAC
- TRMS measurements up to 22° harmonic order, class 1 accuracy
- Power factor and cosφ measure
- Voltage and current harmonic analysis per phase up to 22° harmonic order
- Electric meters of active energy (import-export)
- Electric meters of reactive energy (inductive-capacitive)
- Housing: flush mount 96x96mm for DMK 3...and DMK 40 modular for DMK 6...
- Degree of protection:
  IP54 on front for DMK 3... and DMK 40
  IP41 on front for DMK 6...
  IP 20 at rear.

Certifications and compliance

Certifications obtained: cULus, GOST.
Compliant with standards, IEC/EN 61010-1, IEC/EN 61000-6-2, CISPR11/EN 55011.

APPENDIX D

Software Routines (MATLAB Pseudo Codes)

D.0: GENERAL MATLAB SCRIPT LAYOUT & SOME MAIN FUNCTIONS

The structure below is used to describe MATLAB source code for every model

```
%     OVERVIEW OF SCRIPT DESCRIPTION
1.    File name
2.    Aim of the model
3.    Forecasting methodology
4.    Input Vector
5.    Output Vector
Notes:

CONNECTING TO THE DATABASE:
%     conn          -          connects a MATLAB session to a database via the specified ODBC driver
%                        returns a connection object
%     database      -       MATLAB function used to connect to an ODBC/JDBC database
%
%     exec          -          returns the cursor object to the variable "curs"
%     fetch         -          imports data from the cursor
%     curs.Data     -     returns the data at the cursor object
 %    corrcoef      -     calculates a matrix of correlation coefficients for an array.
% tansig          -     tan sigmoid transfer function
% purelin         -     pure linear transfer function
%     tic           -       Starts a stopwatch timer
% WS              -       Weather sensitive model
%NWS             -       Non-weather sensitive model
%FFNN            -       Feedforward Neural Network
%ERNN            -       Elman Recurrent Neural Network

%                        VERIFYING THE CONNECTION TO THE DATABASE
%    isconnection  -     MATLAB function that detects if database connection is valid,
%                        and returns a 1 if the DB connection is valid, otherwise a 0 will be displayed.

%    setdbprefs    -       set preferences for data retrieval format. Often numeric format
%                        uses less memory and offer better performance than the cellarray format

%                     CLOSING THE CONNECTION TO THE DATABASE
% close(connect)      -       closes the database connection.
% Defining the training functions and performance function
% traingdm      - gradient descent with momentum backpropagation.
% mse           -       mean squared error performance function.

%Author:                    S. Amakali
%Institution:               Cape Peninsula University of Technology
%Date:                      September 2008
```

D.1: MATLAB Script: Feedforward non-weather sensitive model for predicting the total load in summer

1. **File name** : FFNN_TL_NWS01.m
2. **Aim of the model** : This model is designed specifically to ignore any weather related effect on the load. Thus the model uses only past load values to predict the load as discussed in chapter 6 (section 6.3.1 and 6.4) and chapter 7 (section 7.2)
3. **Methodology** : This model predicts the load for the whole week at once (batch training mode)
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has failed to meet the performance goal (1e-5)

```
%Clearing the screen and memory
clc
clear
 tic;
%    establishing the connection to the DB and extracting the required data
%    Inserting the username and password to access the DB
%    importing the required data into MATLAB using SQL Syntax from a table named 'ivsalldata'
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT Humidity FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT Wind_Speed FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Wind_Speed = str2double(m);
clear m;
```

```
curs= exec(conn,'SELECT Max_Temp FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Max_Temp = str2double(m);
clear m;

curs= exec(conn,'SELECT Min_Temp FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Min_Temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,Wind_Speed);
Consump_Humidity = corrcoef(Consump,Humidity);
Consump_Max_Temp = corrcoef(Consump,Max_Temp);
Consump_Min_Temp = corrcoef(Consump,Min_Temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
norm_Consump = (Consump/MaxConsump)';

%           Preparing Data
%****************************
% defining data set dimensions
n = 1584:1762;              % Training data
k = 1752:1919;              % Target
s = 913:1081;               % Testing data
x = 1752:1919;              %Target - testing data
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-168);norm_Consump(n-169)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
P1 = mat(1:4,1:168);
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
l = 1584:1751;
Pw = [P1];
Tw = norm_Consump(k);
Ptst = [norm_Consump(s);norm_Consump(s-1);norm_Consump(s-168);norm_Consump(s-169)];
[mat,padded] = vec2mat(Ptst,168);
[r,s] = size(mat);
```

```
P2 = mat(1:4,1:168);
Pwtstd = [P2];
Tw1 = norm_Consump(x);

%      Defining ANN Topology
%*****************************
%Designing a custom Feedforward Neural Network with 1 input layer, 1 hidden layer with 10 neurons
and 1 output neurons

net = network;
net.numInputs = 1;
net.inputs{1}.size = 4;
net.numLayers = 2;
net.layers{1}.size = 10;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';

%Defining which layers have biases
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);                %Initialise the weights and biases
net.trainFcn = 'traingdm';
net.performFcn = 'mse';
net.trainParam.lr = 0.3;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,Pw,Tw);
forecast = a;
delta = Tw – forecast;
% Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'
```

```
% simulating the network using the testing data

   a2 = sim(net,Ptst,Tw1);
    forecast1 = a2;
    delta2 = (Tw - forecast1;
    e1 = delta2;

N=168;
MPE = sum((((Tw - forecast)/Tw)*100)/N)
MSE = sum(((Tw - forecast).^2)/N)
MSE_test = mse(Tw - forecast)

% saving the forecast for the training data
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% calculating the sum squared error
num=168;
MSE = (sumsqr(delta))/num;
fprintf('MSE = %d', MSE*100)
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

%       Plotting the Graphs
%**************************
figure(1)
hold on
title('Network response - using training data (a 168 hrs ahead forecast)');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'gx', 'LineWidth',3, 'MarkerSize',10);
legend ('Actual load values', 'Predicted load values');

% Plotting simulation results - using testing datatest.
figure(2)
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
```

```
grid on
plot(forecast1, 'b--', 'LineWidth',3, 'MarkerSize',12);
plot(T, 'm*', 'LineWidth',3, 'MarkerSize',10);
legend('Predicated load values', 'Actual load values');

 %Plotting the error:      using training data
%****************************************
figure(3)
%Actual_error = delta*MaxConsump;
plot(error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (W)- (training data');
xlabel('Time(hrs)');
ylabel('Error(Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
a = delta*MaxConsump;
[mat,padded] =vec2mat(a,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(W)_CERNN_TL_WS01_tr',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_CERNN_TL_WS01' , c, 'A1');

%     closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
forecast_time = toc

%     End
```

D.2: MATLAB Script: Elman Recurrent weather sensitive model for predicting the total load in summer

1. **File name** : ERNN_TL_WS01.m
2. **Aim of the model** : This model brings the effect of climatic conditions on the load on the load. More details about this model can be found in the thesis, chapter 6 (section 6.4) and chapter 7 (section 7.3)
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has successfully met the performance goal (1e-5) in only few iterations.

```
%Clearing the screen and memory
clc
clear
 tic;
%    Establishing the connection to the DB and extracting required data
%    Inserting the username and password to access the DB
%    importing the required data into MATLAB using SQL Syntax from a table named 'ivsalldata'
%**************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT Humidity FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT Wind_Speed FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Wind_Speed = str2double(m);
clear m;
```

```
curs= exec(conn,'SELECT Max_Temp FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Max_Temp = str2double(m);
clear m;

curs= exec(conn,'SELECT Min_Temp FROM ivsalldata');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Min_Temp = str2double(m);
clear m;

%Correlation Analysis between the load exogenous variables
Consump_Wind_Speed = corrcoef(Consump,Wind_Speed);
Consump_Humidity = corrcoef(Consump,Humidity);
Consump_Max_Temp = corrcoef(Consump,Max_Temp);
Consump_Min_Temp = corrcoef(Consump,Min_Temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(Wind_Speed);
MaxHumidity = max(Humidity);
MaxMax_Temp = max(Max_Temp);
MaxMin_Temp = max(Max_Temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (Wind_Speed/MaxWind_Speed)';
norm_Humidity = (Humidity/MaxHumidity)';
norm_Max_Temp = (Max_Temp/MaxMax_Temp)';
norm_Min_Temp = (Min_Temp/MaxMin_Temp)';

%          Preparing Data
%****************************
%defining data set dimensions
n = 1584:1762;        %Training data
k = 1752:1919;        % Target - training data
s = 913:1081;         % Testing data
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-168);norm_Consump(n-169)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
P1 = mat(1:4,1:168);
```

```
Temp1 = [norm_Max_Temp(k);norm_Min_Temp(k)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
Temp = mat(1:2,1:168);
l = 1584:1751;
Humid = [norm_Humidity(k)];
Wind = [norm_Wind_Speed(k)];
Pw = [P1;Temp;Humid;Wind];
Tw = norm_Consump(k);
Ptst = [norm_Consump(s);norm_Consump(s-1);norm_Consump(s-168);norm_Consump(s-169)];
[mat,padded] = vec2mat(Ptst,168);
[r,s]=size(mat);
P2 = mat(1:4,1:168);
Pwtstd = [P2;Temp;Humid;Wind];
%Target - testing data
Tw1 = norm_Consump(x);

  %      Defining ANN Topology
%*******************************
nhn = 10;                                   % Number of hidden layer neurons
net = newelm([minmax(Pw)],[nhn 1 ],{'tansig','purelin'},'traingdm');
net.layers{1}.initFcn='initnw';
net.layers{2}.initFcn='initnw';
net = init(net);                            % Initialization of network
net.performFcn='mse';
net.trainParam.epochs = 1000;               % Maximum no. of epochs
net.trainParam.show = 50;                   % No. of epochs to display
net.trainParam.lr = 0.3;                    % Learning rate
net.trainParam.mc = 0.75;                   % Momentum term
net.trainParam.goal = 1e-5;                 %Defining the goal
init_in_weights = net.IW{1,1};              % Determining the initial input weights
init_out_weights = [net.LW{2,1}(:)];        % Determining the initial output
init_in_bias = net.b{1};                    % Determining the initial input bias
init_out_bias = net.b{2};                   % Determining the initial output bias
for i = 1:168;
      P = Pw(:,i);
      T = Tw(1,i);
  [net,tr,a] = train(net,P,T);
      forecast(1,i) = a;
      delta(1,i)= Tw(1,i) - forecast(1,i);
end

% Saving the forecast for the training data
saveas(gcf, 'trainscg_err Training', 'fig');
```

```
save tr tr;

  % Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'
% Calculating the sum squared error
num=168;
MSE = (sumsqr(delta))/num;
fprintf('MSE = %d', MSE*100)

MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% simulating the network using the testing data
%***********************************************
  for i = 1:168;
     Pts = Pwtstd(:,i);
       T = Tw1(1,i);
[a2 pf af]= sim(net,Pts,T);
O_Pwtst(1,i) = a2;
forecast_tstdata = O_Pwtst(1,i);
  delta_tst = T - forecast_tstdata;
end

%       Plotting the Graphs
%**************************
figure(1)
hold on
title('Network response - using training data (a 168 hrs ahead forecast)');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'gx', 'LineWidth',3, 'MarkerSize',10);
legend ('Actual load values', 'Predicted load values');
```

```
% Plotting forecast for the testing data
figure(2)
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(forecast_tstdata, 'b--', 'LineWidth',3, 'MarkerSize',12);
legend('Predicated load values', 'Actual load values');

%Plotting the error:
%***************
figure(3)
Actual_error = delta*MaxConsump
plot(Actual_error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (W)- (training data');
xlabel('Time(hrs)');
ylabel('Error(Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
a = delta*MaxConsump;
[mat,padded] =vec2mat(a,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_ERNN_TL_WS01_tr',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_ERNN_TL_WS01' , c, 'A1');

%      closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
forecast_time = toc
%       End
```

D.3: MATLAB Script: Feedforward weather sensitive model for predicting the total load in winter)

1. **File name** : FFNN_TL_WS02.m
2. **Aim of the model** : This model is designed specifically to ignore any weather related effect on the load. Thus the model uses only past load values to predict the load as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.6)
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has successfully met the performance goal (1e-5) in only few iterations.

```
%Clearing the screen and memory
clc
clear
 tic;
%    establishing the connection to the DB and extracting required data
%    Inserting the username and password to access the DB
%    importing the required data into MATLAB using SQL Syntax from a table named 'iv_tl_winter'
%**************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT consump FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
consump = str2double(m);
clear m;

curs= exec(conn,'SELECT Humidity FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT Wind_Speed FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
wind_speed = str2double(m);
clear m;
```

```
curs= exec(conn,'SELECT Max_Temp FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
max_temp = str2double(m);
clear m;

curs= exec(conn,'SELECT Min_Temp FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
min_temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(consump,wind_speed);
Consump_Humidity = corrcoef(consump,humidity);
Consump_Max_Temp = corrcoef(consump,max_temp);
Consump_Min_Temp = corrcoef(consump,min_temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(consump);
MaxWind_Speed = max(wind_speed);
MaxHumidity = max(humidity);
MaxMax_Temp = max(max_temp);
MaxMin_Temp = max(max_temp);
norm_Consump = (consump/MaxConsump)';
norm_Wind_Speed = (wind_speed/MaxWind_Speed)';
norm_Humidity = (humidity/MaxHumidity)';
norm_Max_Temp = (max_temp/MaxMax_Temp)';
norm_Min_Temp = (min_temp/MaxMin_Temp)';

%          Preparing Data
%****************************
%defining the data set dimensions
n = 1345:1514;                  %Training data
s = 1513:1682;                  % Target - training data
k = 1681:1850;                  % Testing data
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-168);norm_Consump(n-169)];
Ptst = [norm_Consump(s);norm_Consump(s-1);norm_Consump(s-168);norm_Consump(s-169)];
P1 = P(:,1:168);
P2 = Ptst(:,1:168);
Tw1 = norm_Consump(k);
```

```
Tw = Tw1(:,1:168);
Temp1 = [norm_Max_Temp(k);norm_Min_Temp(k)];
Temp = Temp1(:,1:168);
Humid1 = [norm_Humidity(k)];
Wind1 = [norm_Wind_Speed(k)];
Humid = Humid1(:,1:168);
Wind = Wind1(:,1:168);
Pw    =    [P1;Temp;Humid;Wind];
Pwtstd    =    [P2;Temp;Humid;Wind];

%Designing a custom Feedforward Neural Network Model with 1 input, 1 hidden layer with 5 neurons
and 1 output neurons
net = network;
net.numInputs = 1;
net.inputs{1}.size = 8;
net.numLayers = 2;
net.layers{1}.size = 5;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';

%Define which layers have biases
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);%Initialise the weights and biases

% Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'

% Declare the hour by hour forecasting Algorithm and other training parameters
for i = 1:168;
      Pin = Pw(:,i);
      T = Tw(1,i);
net.trainFcn = 'traingdm';
net.performFcn = 'mse';
```

```
net.trainParam.lr = 0.3;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,Pin,T);
        forecast(1,i) = a;
        delta(1,i)= Tw(1,i) - forecast(1,i);
end
  % saving the forecast for the training data
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% simulating the network with the same inputs used to train the network
O_Pw = sim (net,Pw);
% calculating the sum squared error
  e = (Tw-O_Pw);
MSE = mse(e);
num=168;
% calculating the mean percentage error
MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))

%Denormalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;
ae = Tw_denorm - O_Pw_denorm;
%       Plotting the Graphs
%********************
figure(1)
hold on
title('168hrs in advance forecast -CPUTBvl( training data -winter) FFNN-TL-WS02');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'g:', 'LineWidth',3, 'MarkerSize',10);
legend ('Actual load values','Predicted load values');
% simulating the network using the testing data set
%*******************************************
O_Pwtst = sim (net,Pwtstd);
% saving the forecast
saveas(gcf, 'testscg_err', 'fig');
save O_Pwtst O_Pwtst;
```

```
e1 = Tw - O_Pwtst;
MSE2 = mse(e1)
MPE_ts = ((1/num)* sum((e1/Tw).^2))*100
MSE_ts = (1/num)* (sum(e1.^2))

% Plotting the graph for the testing data
figure(2)
hold on
title('Normalized 168hrs ahead forecast CPUTBvl(TestingData winter)FFNN-TL-WS02');
xlabel('Time in hours');
ylabel('Active Power/Load normalized');
grid on
plot(O_Pwtst, 'b--', 'LineWidth',3, 'MarkerSize',12);
plot(Tw, 'm-', 'LineWidth',3, 'MarkerSize',10);
legend('Predicted load values', 'Actual load values');

% plotting the error
%***************
figure(3)
plot((ae), 'b--', 'LineWidth',2, 'MarkerSize',10);
title('error (training data');
xlabel('Time in hours');
ylabel('error(W)');
legend ('error (FFNN-TL-WS02)');
grid on

% Saving the MSE,MPE and actual error(kW) to an Excel spread sheet
a = ae;
[mat,padded] =vec2mat(a,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_FFNN_TL_WS02_tr',e_tr');
b = e1*MaxConsump;
[mat,padded] =vec2mat(b,24);
e_ts = mat(1:7,1:24);
actual_error_ts = xlswrite('error(kW)_FFNN_TL_WS02_ts',e_ts');
  c = {'MSE_tr','MSE_ts','MPE_tr','MPE_ts';MSE_tr MSE_ts MPE_tr MPE_ts};
d = xlswrite('MSEMPE_FFNN_TL_WS02' , c, 'A1');

%closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
%       End
```

D.4: MATLAB Script: Elman Recurrent non-weather sensitive model for predicting the total load in winter

1. **File name** : ERNN_TL_NWS02.m
2. **Aim of the model** : This model is also designed specifically to ignore any weather related effect on the load. Thus the model uses only past load values to predict the load as discussed in chapter 6 (section 6.4) and Appendix E.
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : IV # 2.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has also successfully met the performance goal (1e-5) in only few iterations.

```
%Clearing the screen and memory
clc
clear
%    Establishing the connection to the DB and extracting required data
%    Inserting the username and password for the DB
%    Importing required data into MATLAB using SQL Syntax from a table named 'iv_tl_winter'
%*************************************************************************
tic;
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT consump FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT humidity FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT wind_speed FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Wind_Speed = str2double(m);
clear m;
```

```
curs= exec(conn,'SELECT max_temp FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Max_Temp = str2double(m);
clear m;

curs= exec(conn,'SELECT min_temp FROM iv_tl_winter');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Min_Temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,Wind_Speed);
Consump_Humidity = corrcoef(Consump,Humidity);
Consump_Max_Temp = corrcoef(Consump,Max_Temp);
Consump_Min_Temp = corrcoef(Consump,Min_Temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(Wind_Speed);
MaxHumidity = max(Humidity);
MaxMax_Temp = max(Max_Temp);
MaxMin_Temp = max(Max_Temp);
norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (Wind_Speed/MaxWind_Speed)';
norm_Humidity = (Humidity/MaxHumidity)';
norm_Max_Temp = (Max_Temp/MaxMax_Temp)';
norm_Min_Temp = (Min_Temp/MaxMin_Temp)';

%          Preparing Data
%****************************
% defining data set dimensions
n = 1345:1514;                  % Training data
k = 1681:1850;                  % Target
s = 1513:1682;                  % Testing data
Tw1 = norm_Consump(k);
Tw = Tw1(:,1:168);
Humid = Humid1(:,1:168);
Wind = Wind1(:,1:168);
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-168);norm_Consump(n-169)];
```

```
Ptst = [norm_Consump(s);norm_Consump(s-1);norm_Consump(s-168);norm_Consump(s-169)];
Temp1 = [norm_Max_Temp(k);norm_Min_Temp(k)];
P1 = P(:,1:168);
P2 = Ptst(:,1:168);
Temp = Temp1(:,1:168);
Humid1 = [norm_Humidity(k)];
Wind1 = [norm_Wind_Speed(k)];
Pw = [P1];
Pwtstd = [P2];

%      Defining ANN Topology
%*****************************
%Designing a Elman Recurrent Neural Network with 1 input, 1 hidden layer with 10 neurons and
1output neurons
nhn = 10;                                   % Number of hidden layer neurons
net = newelm([minmax(Pw)],[nhn 1 ],{'tansig','purelin'},'traingdm');
net.layers{1}.initFcn='initnw';
net.layers{2}.initFcn='initnw';
net = init(net);                            % Initialization of network
net.performFcn='mse';
net.trainParam.epochs = 1000;               % Maximum no. of epochs
net.trainParam.show = 50;                   % No. of epochs to display
net.trainParam.lr = 0.3;                    % Learning rate
net.trainParam.mc = 0.75;                   % Momentum term
net.trainParam.goal = 1e-5;                 %Defining the goal
init_in_weights = net.IW{1,1};              % Determining the initial input weights
init_out_weights = [net.LW{2,1}(:)];        % Determining the initial output
init_in_bias = net.b{1};                    % Determining the initial input bias
init_out_bias = net.b{2};                   % Determining the initial output bias
for i = 1:168;
      P = Pw(:,i);
      T = Tw(1,i);
  [net,tr,a] = train(net,P,T);
      forecast(1,i) = a;
      delta(1,i)= Tw(1,i) - forecast(1,i);
end
Saving of the forecast for the training data
saveas(gcf, 'trainscg_err Training', 'fig');
save tr tr;

% Calculating the sum squared error
num=168;
MPE_tr = ((1/num)* sum((delta/Tw).^2))*100;
MSE_tr = (1/num)* (sum(delta.^2));
```

```
perf = mse(delta);

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% Simulating the network using the testing data
%************************************************
 for i = 1:168;
    Pts = Pwtstd(:,i);
     T = Tw(1,i);
[a2 pf af]= sim(net,Pts);
O_Pwtst(1,i) = a2;
forecast_tstdata = O_Pwtst(1,i);
 delta_tst = T - forecast_tstdata;
end

%       Plotting the graphs
%**************************
figure(1)
hold on
title('Network response - using training data (a 168 hrs ahead forecast)');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'gx', 'LineWidth',3, 'MarkerSize',10);
legend ('Actual load values', 'Predicted load values');

% Plotting simulation results - using testing data test.
figure(2)
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(forecast_tstdata, 'b--', 'LineWidth',3, 'MarkerSize',12);
legend('Predicated load values', 'Actual load values');
 % Saving the MSE,MPE and actual error(kW) to an Excel spread sheet
a = delta*MaxConsump;
[mat,padded] =vec2mat(a,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(W)_CERNN_TL_NWS02_tr',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
```

```
d = xlswrite('MSEMPE_ERNN_TL_NWS02' , c, 'A1');

%      closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
Forecast_Time = toc
%          End
```

D.5: MATLAB Script: Feedforward Model for predicting the load of the IT Department

1. **File name** : FFNN_IT.m
2. **Aim of the model** : This model also brings the effect of climatic conditions on the load. Thus it uses a combination of the past load values and weather data to predict the load for the IT department as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.1).
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has also successfully met the performance goal (1e-5) in only few iterations. The target vector used during the training was later defined as the testing data.

```
%Clearing the screen and memory
clc
clear
  tic;
%      Establishing the connection to the DB and extracting required data
%      Inserting the username and password to access the DB
%      importing the required data into MATLAB using SQL Syntax from a table named 'ivit22'
%% The model is trained using the batch incremental mode
%*************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
  DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;
  curs= exec(conn,'SELECT Humidity FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Humidity = str2double(m);
clear m;
  curs= exec(conn,'SELECT Wind_speed FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Wind_Speed = str2double(m);
```

```
clear m;
  curs= exec(conn,'SELECT Max_temp FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Max_Temp = str2double(m);
clear m;
  curs= exec(conn,'SELECT Min_temp FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Min_Temp = str2double(m);
clear m;

  %Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,Wind_Speed);
Consump_Humidity = corrcoef(Consump,Humidity);
Consump_Max_Temp = corrcoef(Consump,Max_Temp);
Consump_Min_Temp = corrcoef(Consump,Min_Temp);

%             Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(Wind_Speed);
MaxHumidity = max(Humidity);
MaxMax_Temp = max(Max_Temp);
MaxMin_Temp = max(Max_Temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (Wind_Speed/MaxWind_Speed)';
norm_Humidity = (Humidity/MaxHumidity)';
norm_Max_Temp = (Max_Temp/MaxMax_Temp)';
norm_Min_Temp = (Min_Temp/MaxMin_Temp)';

%           Preparing Data
%****************************
%defining the data set dimensions
n = 313:480;              %Training data
s = 145:312;              %Target
  P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-168);norm_Consump(n-169)];
Tw = [norm_Consump(s)];
```

```
%defining weather data dimensions
 Temp1 = [norm_Max_Temp(s);norm_Min_Temp(s)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
Temp = mat(1:2,1:168);
Humid = [norm_Humidity(s)];
Wind = [norm_Wind_Speed(s)];
%Defining the final input vector # 1.
Pw = [P;Temp;Humid;Wind];

%      Defining a custom ANN Topology
%*****************************
%Designing a custom Feedforward Neural Network Model with 1 input, 1 hidden layer with 5 neurons
and 1 output neurons
net = network;
net.numInputs = 1;
net.inputs{1}.size = 8;
net.numLayers = 2;
net.layers{1}.size = 5;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
%Define which layers have biases
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);%Initialise the weights and biases
 for i = 1:168;
      P = Pw(:,i);
      T = Tw(1,i);
net.trainFcn = 'traingdm';
net.performFcn = 'mse';
net.trainParam.lr = 0.1;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,P,T);
      forecast(1,i) = a;
      delta(1,i)= Tw(1,i) - forecast(1,i);
end
```

```
% saving the forecast -
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% calculating the sum squared error
num=168;

MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% Simulating the network using the same data used during training-
%**************************************************
for i=1:168;
[a2 pf af]= sim(net,Pw(:,i));
end
%       Plotting the Graph
%***************************
figure(1)
hold on
title('Network response - using IT training data (a 168 hrs ahead forecast)');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'bo', 'LineWidth',2, 'MarkerSize',5);
legend ('Actual load values', 'Predicted load values');

% Plotting simulation results – for the testing data set.
figure(2)
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(a2, 'b--', 'LineWidth',3, 'MarkerSize',12);
%plot(T(1,i), 'm*', 'LineWidth',3, 'MarkerSize',10);
legend('Predicated load values', 'Actual load values');
```

```
%Plotting the error:      for the training data
%**************************************
figure(3)
Actual_error = delta*MaxConsump;
plot(Actual_error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (kW)- (training data');
xlabel('Time(hrs)');
ylabel('Error(kilo Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
 error    = delta*MaxConsump;
[mat,padded] =vec2mat(error,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_FFNN_IT',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_FFNN_IT' , c, 'A1');

%      closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
Forecast_Time = toc
DB_connection = isconnection(conn)
%             End
```

D.6: MATLAB Script: Elman Recurrent weather sensitive model for predicting the load of the IT Department

1. **File name** : ERNN_IT.m
2. **Aim of the model** : This model ignores the effect of climatic conditions on the load. Thus it merely uses the past load values and weather data to predict the load for the IT department as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.1).
3. **Methodology** : This model predicts the load for the whole week at once
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has failed to meet the performance goal (1e-5). The target vector used during the training was later defined as the testing data.

```
%Clearing the screen and memory
clc
clear
 tic;
%    Establishing the connection to the DB and extracting required data
%    Inserting the username and password to access the DB
%    importing the required data into MATLAB using SQL Syntax from a table named 'ivit22'
%    The model is trained using the batch training mode
%**************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT Humidity FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT Wind_speed FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
```

```
Wind_Speed = str2double(m);
clear m;

curs= exec(conn,'SELECT Max_temp FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Max_Temp = str2double(m);
clear m;

curs= exec(conn,'SELECT Min_temp FROM ivit22');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Min_Temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,Wind_Speed);
Consump_Humidity = corrcoef(Consump,Humidity);
Consump_Max_Temp = corrcoef(Consump,Max_Temp);
Consump_Min_Temp = corrcoef(Consump,Min_Temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(Wind_Speed);
MaxHumidity = max(Humidity);
MaxMax_Temp = max(Max_Temp);
MaxMin_Temp = max(Max_Temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (Wind_Speed/MaxWind_Speed)';
norm_Humidity = (Humidity/MaxHumidity)';
norm_Max_Temp = (Max_Temp/MaxMax_Temp)';
norm_Min_Temp = (Min_Temp/MaxMin_Temp)';

%          Preparing Data
%****************************
n = 313:480;              %Training data
s = 145:312;              %target
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-168);norm_Consump(n-169)];
Tw = [norm_Consump(s)];
```

```
%Defining the weather data
  Temp1 = [norm_Max_Temp(s);norm_Min_Temp(s)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
Temp = mat(1:2,1:168);
Humid = [norm_Humidity(n)];
Wind = [norm_Wind_Speed(n)];
Pw = [P;Temp;Humid;Wind];

%      Defining ANN Topology
%*******************************
nhn = 5;                                        % Number of hidden layer neurons
net = newelm([minmax(Pw)],[nhn 1 ],{'tansig','purelin'},'traingdm');
net.layers{1}.initFcn='initnw';
net.layers{2}.initFcn='initnw';
net = init(net);                                % Initialization of network
net.performFcn='mse';
net.trainParam.epochs = 1000;                   % Maximum no. of epochs
net.trainParam.show = 50;                       % No. of epochs to display
net.trainParam.lr = 0.3;                        % Learning rate
net.trainParam.mc = 0.75;                       % Momentum term
net.trainParam.goal = 1e-5;                     %Defining the goal
init_in_weights = net.IW{1,1};                  % Determining the initial input weights
init_out_weights = [net.LW{2,1}(:)];            % Determining the initial output
init_in_bias = net.b{1};                        % Determining the initial input bias
init_out_bias = net.b{2};                       % Determining the initial output bias
[net,tr] = train (net,Pw,Tw);

% simulating the network with the same inputs used to in the training
O_Pw = sim (net,Pw);
e = (Tw - O_Pw);

% Calculating the sum squared error
num=168;
  MPE_tr = ((1/num)* sum((e/Tw).^2))*100
MSE_tr = (1/num)* (sum(e.^2))
perf = mse(e)
%Data De-normalization
O_Pw_denorm = O_Pw*MaxConsump;
Tw_denorm = Tw*MaxConsump;

%       Plotting the Graphs
%*************************
figure(1)
```

```
hold on
title('IT 1 week ahead forecasts vs. target - ERNN_IT');
xlabel('Time(hrs)');
ylabel('Active Power/Load (kW)');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',8);
plot(O_Pw_denorm, 'g*', 'LineWidth',3, 'MarkerSize',7);
legend ('Target', 'Forecasted load');


% Saving the MSE,MPE actual error(W) to an Excel spread sheet
 error    = e*MaxConsump;
[mat,padded] =vec2mat(error,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_ERNN_IT',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_ERNN_IT' , c, 'A1');

%Plotting the error
%**********************

figure(3)
plot((e), 'b--', 'LineWidth',2, 'MarkerSize',10);
title('error (training data)');
xlabel('Time(hrs');
ylabel('actual error(kW)');
legend ('error (ERNN IT)');
grid on

%closing the connection to the DB
close(curs)
close(conn)
DB_connection = isconnection(conn)

%            End
```

D.7: MATLAB Script: Elman Recurrent weather sensitive model for predicting the load of the Electrical Engineering Department

1. **File name** : ERNN_ELEC.m
2. **Aim of the model** : This model also brings the effect of climatic conditions on the load. Thus it uses a combination of the past load values and weather data to predict the load for the Electrical engineering department as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.2).
3. **Methodology** : This model predicts the load for the whole week at once
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. **Note** : This model has failed to meet the performance goal (1e-5). The target vector used during the training was later defined as the testing data.

```
%Clearing the screen and memory
clc
clear
 tic;
%    Establishing the connection to the DB and extracting required data
%    Inserting the username and password to access the DB
%    Importing the required data into MATLAB using SQL Syntax from a table named 'ivelec1'
%*******************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT humidity FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT wind_speed FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
wind_speed = str2double(m);
```

```
clear m;

curs= exec(conn,'SELECT max_temp FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
max_temp = str2double(m);
clear m;

curs= exec(conn,'SELECT min_temp FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
min_temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,wind_speed);
Consump_Humidity = corrcoef(Consump,humidity);
Consump_Max_Temp = corrcoef(Consump,max_temp);
Consump_Min_Temp = corrcoef(Consump,min_temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(wind_speed);
MaxHumidity = max(humidity);
MaxMax_Temp = max(max_temp);
MaxMin_Temp = max(max_temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (wind_speed/MaxWind_Speed)';
norm_Humidity = (humidity/MaxHumidity)';
norm_Max_Temp = (max_temp/MaxMax_Temp)';
norm_Min_Temp = (min_temp/MaxMin_Temp)';

%          Preparing Data
%****************************
n = 169:336;          %Training data
s = 1:168;            %Target
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-167);norm_Consump(n-168)];
Tw = [norm_Consump(s)];

%Defining the weather data
```

```
Temp1 = [norm_Max_Temp(s);norm_Min_Temp(s)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
Temp = mat(1:2,1:168);
Humid = [norm_Humidity(s)];
Wind = [norm_Wind_Speed(s)];

Pw = [P;Temp;Humid;Wind];

%     Defining ANN Topology
%*****************************
nhn = 5;                                        % Number of hidden layer neurons
net = newelm([minmax(Pw)],[nhn 1 ],{'tansig','purelin'},'traingdm');
net.layers{1}.initFcn='initnw';
net.layers{2}.initFcn='initnw';
net = init(net);                                % Initialization of network
net.performFcn='mse';
net.trainParam.epochs = 1000;                   % Maximum no. of epochs
net.trainParam.show = 50;                       % No. of epochs to display
net.trainParam.lr = 0.3;                        % Learning rate
net.trainParam.mc = 0.75;                       % Momentum term
net.trainParam.goal = 1e-5;                     %Define the goal
init_in_weights = net.IW{1,1};                  % Determining the initial input weights
init_out_weights = [net.LW{2,1}(:)];            % Determining the initial output
init_in_bias = net.b{1};                        % Determining the initial input bias
init_out_bias = net.b{2};                       % Determining the initial output bias
[net,tr] = train (net,Pw,Tw);

% simulating the network with the same inputs used to in the training
O_Pw = sim (net,Pw);
e = (Tw-O_Pw);

 % calculating the sum squared error
num=168;
 MPE_tr = ((1/num)* sum((e/Tw).^2))*100
MSE_tr = (1/num)* (sum(e.^2))
perf = mse(e)

%Data De-normalization
O_Pw_denorm = O_Pw*MaxConsump;
Tw_denorm = Tw*MaxConsump;

%       Plotting the Graphs
%*************************
```

```
figure(1)
hold on
title('Elec. Eng. 1 week ahead forecasts vs. target - ERNN_ELEC. ENG_WS');
xlabel('Time(hrs)');
ylabel('Active Power/Load (kW)');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',8);
plot(O_Pw_denorm, 'm*', 'LineWidth',3, 'MarkerSize',7);
legend ('Target', 'Forecasted load');

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
 error    = e*MaxConsump;
[mat,padded] =vec2mat(error,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_ERNN_ELEC',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_ERNN_ELEC' , c, 'A1');

%Plotting the error
%**********************
 figure(3)
plot((e), 'b--', 'LineWidth',2, 'MarkerSize',10);
title('error (training data)');
xlabel('Time(hrs');
ylabel('actual error(kW)');
legend ('error (ERNN ELEC. ENG WS)');
grid on

%closing the connection to the DB
close(curs)
close(conn)
DB_connection = isconnection(conn)

%         End
```

D.8: MATLAB Script: Feedforward weather sensitive model for predicting the load of the Electrical Engineering Department.

1. **File name** : FFNN_ELEC.m
2. **Aim of the model** : This model also brings the effect of climatic conditions on the load. Thus it uses a combination of the past load values and weather data to predict the load for the Electrical Engineering department as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.2).
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. ***Note*** : This model has successfully met the performance goal (1e-5) in only few iterations. The target vector used during the training was later defined as the testing data.

```
%Clearing the screen and memory
clc
clear
 tic;
%    Establishing the connection to the DB and extracting required data
%    Inserting the username and password to access the DB
%    importing the required data into MATLAB using SQL Syntax from a table named 'ivelec1'
%*************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT humidity FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT wind_speed FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
```

```
wind_speed = str2double(m);
clear m;

curs= exec(conn,'SELECT max_temp FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
max_temp = str2double(m);
clear m;

curs= exec(conn,'SELECT min_temp FROM ivelec11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
min_temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,wind_speed);
Consump_Humidity = corrcoef(Consump,humidity);
Consump_Max_Temp = corrcoef(Consump,max_temp);
Consump_Min_Temp = corrcoef(Consump,min_temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(wind_speed);
MaxHumidity = max(humidity);
MaxMax_Temp = max(max_temp);
MaxMin_Temp = max(max_temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (wind_speed/MaxWind_Speed)';
norm_Humidity = (humidity/MaxHumidity)';
norm_Max_Temp = (max_temp/MaxMax_Temp)';
norm_Min_Temp = (min_temp/MaxMin_Temp)';

%          Preparing Data
%****************************
n = 169:336;          %Training data
s = 1:168;            %Target
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-167);norm_Consump(n-168)];
Tw = [norm_Consump(s)];
```

```
%Defining the weather data
Temp1 = [norm_Max_Temp(s);norm_Min_Temp(s)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
Temp = mat(1:2,1:168);
Humid = [norm_Humidity(s)];
Wind = [norm_Wind_Speed(s)];

Pw = [P;Temp;Humid;Wind];

%      Defining ANN Topology
%*******************************
%Designing a custom Feedforward Neural Network Model with 1 input, 1 hidden layer with 5 neurons
and 1 output neuron
net = network;
net.numInputs = 1;
net.inputs{1}.size = 8;
net.numLayers = 2;
net.layers{1}.size = 5;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
%Define which layers have biases
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);                    % Initializing the weights and biases

% Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'

% declaring the hour by hour forecasting Algorithm and other training parameters
for i = 1:168;
      P = Pw(:,i);
      T = Tw(1,i);
net.trainFcn = 'traingdm';
```

```
net.performFcn = 'mse';
net.trainParam.lr = 0.3;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,P,T);
        forecast(1,i) = a;
        delta(1,i)= Tw(1,i) - forecast(1,i);
end

% Saving the forecast for the training data
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% Calculating the sum squared error
num=168;
  MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% simulating the network using the same data - no testing data available
%************************************************
for i=1:168;
[a2 pf af]= sim(net,Pw(:,i));
end
%       Plotting the Graph
%**************************
figure(1)
hold on
title('Network response (ELEC. ENG)- using    training data (a 168 hrs ahead forecast)');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'bo', 'LineWidth',2, 'MarkerSize',5);
legend ('Actual load values', 'Predicted load values');

% Plotting simulation results - using testing datatest.
figure(2)
```

```
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(a2, 'b--', 'LineWidth',3, 'MarkerSize',12);
legend('Predicated load values', 'Actual load values');


%plotting the error of the training data
%****************************************
figure(3)
Actual_error = delta*MaxConsump;
plot(Actual_error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (kW)- (training data');
xlabel('Time(hrs)');
ylabel('Error(kilo Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
 error    = delta*MaxConsump;
[mat,padded] =vec2mat(error,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_FFNN_ELEC ENG',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_FFNN_ELEC ENG' , c, 'A1');

%     closing the connection to the DB
close(curs)
close(conn)
DB_connection = isconnection(conn)

%         End
```

D.9: MATLAB Script: Feedforward model for predicting the load of the Mechanical Engineering Department.

1. **File name** : FFNN_MEC.m
2. **Aim of the model** : This model also ignores the effect of climatic conditions on the load. Thus it merely uses the past load values to predict the load for the Mechanical Engineering department as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.3).
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : Two past load values (L (t-1)) and (L (t-2)).
5. **Output Vector** : a 24 hours ahead forecast
6. ***Note*** : This model has successfully met the performance goal (1e-5) in only few iterations. The target vector used during the training was later defined as the testing data.
    a) An additional 45% was added to the measured load to cater for machines which were not in operation when load measurements were taken.

```
%Clears memory and screen
clc
clear

%     Establishing the connection to the DB and extracting required data
%     Inserting the username and password for the DB
%     Importing required data into MATLAB using SQL Syntax form a table named 'iv_mec'
%*************************************************************************
tic;
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT consump FROM iv_mec');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

%            Input Data Preprocessing
%*************************************
MaxConsump = max(Consump);

norm_Consump = (Consump/MaxConsump)';

%          Preparing Data
%****************************
```

```
n = 2:25;                   %Training
k = 25:48;                  %Target -
s = 48:72;                  % Testing data
P = [norm_Consump(n);norm_Consump(n-1)];
P1 = P(:,1:24);
Pw = [P1];
Tw = norm_Consump(k);
Ptst = [norm_Consump(s);norm_Consump(s-1)];
P2 = Ptst(:,1:24);
Pwtstd = [P2];

%      Defining ANN Topology
%*******************************
%Designing a custom Feedforward Neural Network with 1 input layer, 1 hidden layer with 10 neurons
and 1 output neuron
net = network;
net.numInputs = 1;
net.inputs{1}.size = 2;
net.numLayers = 2;
net.layers{1}.size = 5;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);                                %Initializing the weights and biases
  % Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'

  % defining the hour-by-hour algorithm
for i = 1:24;
      P = Pw(:,i);
      T = Tw(1,i);
net.trainFcn = 'traingdm';
net.performFcn = 'mse';
```

```
net.trainParam.lr = 0.3;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,P,T);
        forecast(1,i) = a;
        delta(1,i)= Tw(1,i) - forecast(1,i);
end

% Saving the forecast
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% Calculating the sum squared error
num=24;
MSE = (sumsqr(delta))/num;
fprintf('MSE = %d', MSE*100)
MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% Simulating the network using the testing data
%************************************************
for i=1:24;
[a2 pf af]= sim(net,Pwtstd(:,i));
end
%       Plotting the Graph
%**************************
figure(1)
hold on
title('a 24hrs ahead forecast for Mec. Eng.- for Tuesday the 2nd Sep, 08');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'gx', 'LineWidth',3, 'MarkerSize',10);
legend ('Actual load values', 'Predicted load values');
% Plotting simulation results - using the testing data set.
figure(2)
```

```
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(a2, 'b--', 'LineWidth',3, 'MarkerSize',12);
legend('Predicated load values', 'Actual load values');

figure(3)
Actual_error = delta*MaxConsump;
plot(Actual_error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (W)- (training data');
xlabel('Time(hrs)');
ylabel('Error(Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
error    = delta*MaxConsump;
e_tr = error';
actual_error_tr = xlswrite('error(kW)_FFNN_MEC',e_tr);
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_FFNN_MEC' , c, 'A1');

%     closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
Forecast_Time = toc

%              End
```

D.10: MATLAB Script: Feedforward model for predicting the load of the Air Conditioning Plant (ACP).

1. **File name** : FFNN_ACP.m
2. **Aim of the model** : This model also ignores the effect of climatic conditions on the load. Thus it merely uses the past load values to predict the load for the Air Conditioning Plant as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.5).
3. **Methodology** : This model predicts the load for the whole week using the hour-by-hour approach.
4. **Input Vector** : Two past load values (L (t-1)) and (L (t-2)).
5. **Output Vector** : a 24 hours ahead forecast
6. ***Note*** : This model has successfully met the performance goal (1e-5) in only few iterations. The target vector used during the training was later defined as the testing data.
   An additional 15% was added to the measured load to cater for some machines which were not in operation when load measurements were taken.

```
%Clears memory and screen
clc
clear

%     Estabilishing the connection to the DB and extracting required data
%     Inserting the username and password for the DB
%     Importing required data into MATLAB using SQL Syntax form a table named 'iv_acp'
%*************************************************************************
tic;
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT consump FROM iv_acp');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
norm_Consump = (Consump/MaxConsump)';

%           Preparing Data
%****************************
%Defining data set dimensions
n = 2:25;                       %Training data
```

```
k = 25:48;                  %Target
s = 48:72;                  % Testing data
P = [norm_Consump(n);norm_Consump(n-1)];
P1 = P(:,1:24);
Pw = [P1];
Tw = norm_Consump(k);
Ptst = [norm_Consump(s);norm_Consump(s-1)];
P2 = Ptst(:,1:24);
Pwtstd = [P2];

%     Defining ANN Topology
%*****************************
%Designing a custom Feedforward Neural Network with 1 input layer, 1 hidden layer with 10 neurons
and 1output neuron
net = network;
net.numInputs = 1;
net.inputs{1}.size = 2;
net.numLayers = 2;
net.layers{1}.size = 10;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);       %Initializing the weights and biases

 % Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'

% Declaring the hour by hour forecasting Algorithm and other training parameters
for i = 1:24;
      P = Pw(:,i);
      T = Tw(1,i);
net.trainFcn = 'traingdm';
net.performFcn = 'mse';
```

```
net.trainParam.lr = 0.1;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,P,T);
      forecast(1,i) = a;
      delta(1,i)= Tw(1,i) - forecast(1,i);
end

% Saving the forecast for the testing data
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% Calculating the sum squared error
num=24;
MSE = (sumsqr(delta))/num;
fprintf('MSE = %d', MSE*100)
MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% Simulating the network using the testing data
%************************************
for i=1:24;
[a2 pf af]= sim(net,Pwtstd(:,i));
End

%       Plotting the Graph
%**************************
figure(1)
hold on
title('a 24 hrs ahead ACP forecast, Tuesday, 7th October,08 - FFNN ACP');
xlabel('Time in hours');
ylabel('Active Power/Load in (kW)');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'gx', 'LineWidth',3, 'MarkerSize',10);
legend ('Actual load values', 'Predicted load values');
```

```
% Plotting simulation results - using testing datatest.
figure(2)
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(a2, 'b--', 'LineWidth',3, 'MarkerSize',12);
legend('Predicated load values', 'Actual load values');

%Plotting the error:      using training data
%****************************************
figure(3)
Actual_error = delta*MaxConsump;
plot(Actual_error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (W)- (training data');
xlabel('Time(hrs)');
ylabel('Error(Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
error    = delta*MaxConsump;
e_tr = error';
actual_error_tr = xlswrite('error(W)_FFNN_ACP',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_FFNN_ACP' , c, 'A1');

%     closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
Forecast_Time = toc

%                       End
```

D.11: MATLAB Script: Elman Recurrent weather sensitive model for predicting the load of the Administration Building

1. **File name** : ERNN_ADM.m
2. **Aim of the model** : This model also brings the effect of climatic conditions on the load. Thus it uses a combination of the past load values and weather data to predict the load for the Administration Building as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.4).
3. **Methodology** : This model predicts the load for the whole week at once
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts

Note : This model has failed to meet the performance goal (1e-5).

```
% Clearing the screen and memory
clc
clear

%    Estabilishing the connection to the DB and extracting required data
%    Inserting the username and password for the DB
%    Importing required data into MATLAB using SQL Syntax form a table named 'ivadmin11'
%*************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT humidity FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT wind_speed FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
wind_speed = str2double(m);
clear m;
```

```
curs= exec(conn,'SELECT max_temp FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
max_temp = str2double(m);
clear m;

curs= exec(conn,'SELECT min_temp FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
min_temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,wind_speed);
Consump_Humidity = corrcoef(Consump,humidity);
Consump_Max_Temp = corrcoef(Consump,max_temp);
Consump_Min_Temp = corrcoef(Consump,min_temp);

%            Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(wind_speed);
MaxHumidity = max(humidity);
MaxMax_Temp = max(max_temp);
MaxMin_Temp = max(max_temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (wind_speed/MaxWind_Speed)';
norm_Humidity = (humidity/MaxHumidity)';
norm_Max_Temp = (max_temp/MaxMax_Temp)';
norm_Min_Temp = (min_temp/MaxMin_Temp)';

%          Preparing Data
%****************************
n = 169:336;              %Training data
s = 1:168;                %target
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-167);norm_Consump(n-168)];
Tw = [norm_Consump(s)];
Temp1 = [norm_Max_Temp(s);norm_Min_Temp(s)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
```

```
Temp = mat(1:2,1:168);
Humid = [norm_Humidity(s)];
Wind = [norm_Wind_Speed(s)];

Pw = [P;Temp;Humid;Wind];

%      Defining ANN Topology
%*******************************
nhn = 5;                                                % Number of hidden layer neurons
net = newelm([minmax(Pw)],[nhn 1 ],{'tansig','purelin'},'traingdm');
net.layers{1}.initFcn='initnw';
net.layers{2}.initFcn='initnw';
net = init(net);                                        % Initialization of network
net.performFcn='mse';
net.trainParam.epochs = 1000;                           % Maximum no. of epochs
net.trainParam.show = 50;                               % No. of epochs to display
net.trainParam.lr = 0.3;                                % Learning rate
net.trainParam.mc = 0.75;                               % Momentum term
net.trainParam.goal = 1e-5;                             %Define the goal
init_in_weights = net.IW{1,1};                          % Determining the initial input weights
init_out_weights = [net.LW{2,1}(:)];                    % Determining the initial output
init_in_bias = net.b{1};                                % Determining the initial input bias
init_out_bias = net.b{2};                               % Determining the initial output bias
[net,tr] = train (net,Pw,Tw);

% Simulating the network with the same inputs used to in the training
O_Pw = sim (net,Pw);
e = (Tw-O_Pw);

% Calculating the sum squared error
num=168;
  MPE_tr = ((1/num)* sum((e/Tw).^2))*100
MSE_tr = (1/num)* (sum(e.^2))
perf = mse(e)

%Data De-normalization
O_Pw_denorm = O_Pw*MaxConsump;
Tw_denorm = Tw*MaxConsump;

%       Plotting the Graphs
%**************************
figure(1)
hold on
title('1 week ahead forecasts vs. target(Admin) - ERNN_ADMIN');
```

```
xlabel('Time(hrs)');
ylabel('Active Power/Load (kW)');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',8);
plot(O_Pw_denorm, 'c+', 'LineWidth',3, 'MarkerSize',7);
legend ('Target', 'Forecasted load');


% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
 error    = e*MaxConsump;
[mat,padded] =vec2mat(error,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_ERNN_ADMIN',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_ERNN_ADMIN' , c, 'A1');

%Plotting the error
%**********************
 figure(3)
plot((e), 'b--', 'LineWidth',2, 'MarkerSize',10);
title('error (training data)');
xlabel('Time(hrs');
ylabel('actual error(kW)');
legend ('error (ERNN ADMIN)');
grid on

%closing the connection to the DB
close(curs)
close(conn)
DB_connection = isconnection(conn)

%         End
```

D.11: MATLAB Script: Feedforward weather sensitive model for predicting the load of the Administration Building

1. **File name** : FFNN_ADM.m
2. **Aim of the model** : This model also brings the effect of climatic conditions on the load. Thus it uses a combination of the past load values and weather data to predict the load for the Administration Building as discussed in chapter 6 (section 6.4) and chapter 7 (section 7.8.4).
3. **Methodology** : This model predicts the load for the whole using the hour-by-hour approach
4. **Input Vector** : IV # 1.
5. **Output Vector** : one week ahead forecasts
6. ***Note*** : This model has also successfully met the performance goal (1e-5).

```
%Clearing the screen and memory
clc
clear

%   Estabilishing the connection to the DB and extracting required data
%   Inserting the username and password for the DB
%   Importing required data into MATLAB using SQL Syntax from a table named 'ivadmin11'
%**************************************************************************
conn = database('combinedloaddata', 'root', 'password1')
DATABASE_CONNECTION = isconnection(conn)
curs= exec(conn,'SELECT Consump FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
Consump = str2double(m);
clear m;

curs= exec(conn,'SELECT humidity FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
humidity = str2double(m);
clear m;

curs= exec(conn,'SELECT wind_speed FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
wind_speed = str2double(m);
clear m;
```

```
curs= exec(conn,'SELECT max_temp FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
max_temp = str2double(m);
clear m;

curs= exec(conn,'SELECT min_temp FROM ivadmin11');
setdbprefs('DataReturnFormat','cellarray')
curs = fetch(curs);
m = curs.Data;
min_temp = str2double(m);
clear m;

%Correlation Analysis between the load and exogenous variables
Consump_Wind_Speed = corrcoef(Consump,wind_speed);
Consump_Humidity = corrcoef(Consump,humidity);
Consump_Max_Temp = corrcoef(Consump,max_temp);
Consump_Min_Temp = corrcoef(Consump,min_temp);

%             Input Data Preprocessing
%**************************************
MaxConsump = max(Consump);
MaxWind_Speed = max(wind_speed);
MaxHumidity = max(humidity);
MaxMax_Temp = max(max_temp);
MaxMin_Temp = max(max_temp);

norm_Consump = (Consump/MaxConsump)';
norm_Wind_Speed = (wind_speed/MaxWind_Speed)';
norm_Humidity = (humidity/MaxHumidity)';
norm_Max_Temp = (max_temp/MaxMax_Temp)';
norm_Min_Temp = (min_temp/MaxMin_Temp)';

%           Preparing Data
%****************************
n = 169:336;              %Training data
s = 1:168;              %target
P = [norm_Consump(n);norm_Consump(n-1);norm_Consump(n-167);norm_Consump(n-168)];
Tw = [norm_Consump(s)];
Temp1 = [norm_Max_Temp(n);norm_Min_Temp(n)];
[mat,padded] = vec2mat(P,168);
[r,s]=size(mat);
Temp = mat(1:2,1:168);
```

```
Humid = [norm_Humidity(n)];
Wind = [norm_Humidity(n)];
 Pw = [P;Temp;Humid;Wind];

%     Defining ANN Topology
%*****************************
%Designing a custom Feedforward Neural Network Model with 1 input layer, 1 hidden layer with 5
neurons and 1 output neurons
net = network;
net.numInputs = 1;
net.inputs{1}.size = 8;
net.numLayers = 2;
net.layers{1}.size = 5;
net.layers{2}.size = 1;
net.inputConnect(1) = 1;
net.layerConnect(2,1) = 1;
net.outputConnect(2) = 1;
net.targetConnect(2) = 1;
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'purelin';
net.biasConnect(1) = 1;
net.biasConnect(2) = 1;
net = init(net);                                    %Initializing the network

 % Reset weight and biases - Optional for trial only
%net.initFcn = 'initlay';
%net.layers{1}.initFcn = 'initnw';
%net.layers{2}.initFcn = 'initwb';
%net.inputWeights{1,1}.initFcn = 'rands';
%net.biases{1}.initFcn = 'rands'
%net.layerWeights{1,2}.initFcn = 'rands'

% Declare the hour-by-hour forecasting Algorithm and other training parameters
for i = 1:168;
      P = Pw(:,i);
      T = Tw(1,i);
net.trainFcn = 'traingdm';
net.performFcn = 'mse';
net.trainParam.lr = 0.3;
net.trainParam.mc = 0.75;
net.trainParam.epochs = 1000;
net.trainParam.show = 50;
net.trainParam.goal = 1e-5;
[net,tr,a] = train(net,P,T);
```

```
        forecast(1,i) = a;
        delta(1,i)= Tw(1,i) - forecast(1,i);
end

% Saving the forecast for the training data
saveas(gcf, 'traingdm_err Training', 'fig');
save tr tr;

% Calculating the sum squared error
num=168;
MPE_tr = ((1/num)* sum((delta/Tw).^2))*100
MSE_tr = (1/num)* (sum(delta.^2))
perf = mse(delta)

% De-normalization
O_Pw_denorm = forecast*MaxConsump;
Tw_denorm = Tw*MaxConsump;

% simulating the network using the same data - no testing data available
%************************************************
for i=1:168;
[a2 pf af]= sim(net,Pw(:,i));
end
%       Plotting the Graphs
%**************************
figure(1)
hold on
title('Network response (ADMIN)- using    training data (a 168 hrs ahead forecast)');
xlabel('Time in hours');
ylabel('Active Power/Load in [kW]');
grid on
plot(Tw_denorm, 'r--', 'LineWidth',3, 'MarkerSize',10);
plot(O_Pw_denorm, 'bo', 'LineWidth',2, 'MarkerSize',5);
legend ('Actual load values', 'Predicted load values');

% Plotting simulation results - using testing data set
figure(2)
hold on
title('Network response using testing data');
xlabel('Time(hrs)');
ylabel('Active Power - normalized');
grid on
plot(a2, 'b--', 'LineWidth',3, 'MarkerSize',12);
legend('Predicated load values', 'Actual load values');
```

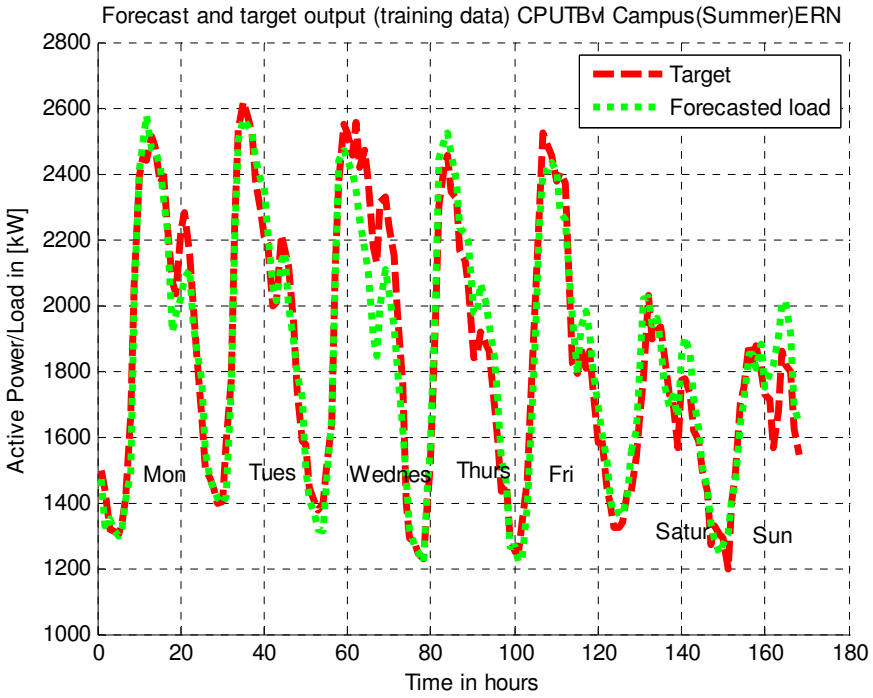
```
 %Plotting the error:
%**************
figure(3)
Actual_error = delta*MaxConsump;
plot(Actual_error, 'b--', 'LineWidth',2, 'MarkerSize',10);
title('Hourly actual error (kW)- (training data');
xlabel('Time(hrs)');
ylabel('Error(kilo Watts)');
legend ('e - ttr');
grid on

% Saving the MSE,MPE and Actual error(kW) to an Excel spread sheet
 error    = delta*MaxConsump;
[mat,padded] =vec2mat(error,24);
e_tr = mat(1:7,1:24);
actual_error_tr = xlswrite('error(kW)_FFNN_ADMIN',e_tr');
c = {'MSE_tr','MPE_tr';MSE_tr MPE_tr};
d = xlswrite('MSEMPE_FFNN_ADMIN' , c, 'A1');

%     closing the connection to the DB
close(curs)
close(conn)
isconnection(conn)
toc;
Forecast_Time = toc
DB_connection = isconnection(conn)
```

% End of MATLAB Pseudo codes

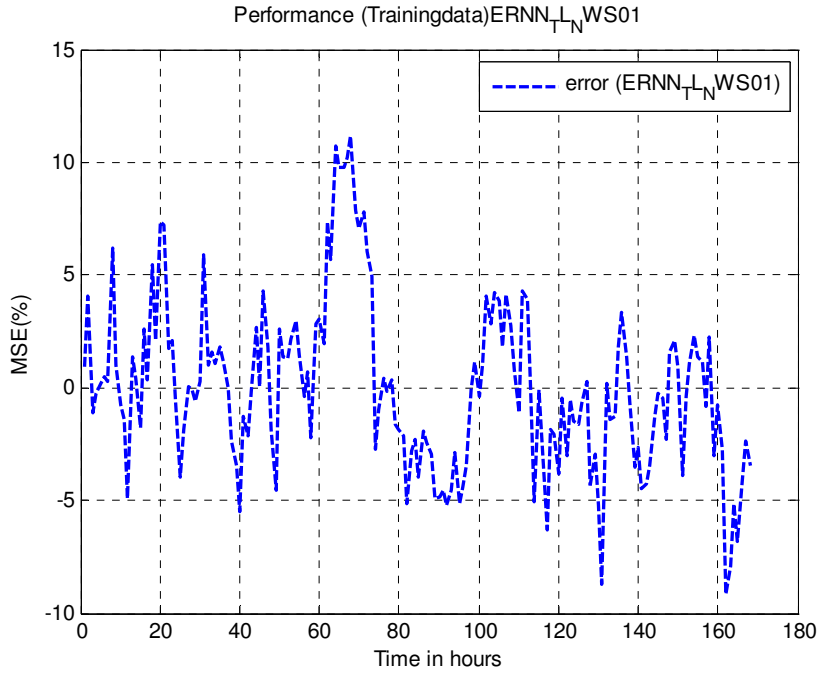
APPENDIX E

Selected Application Results

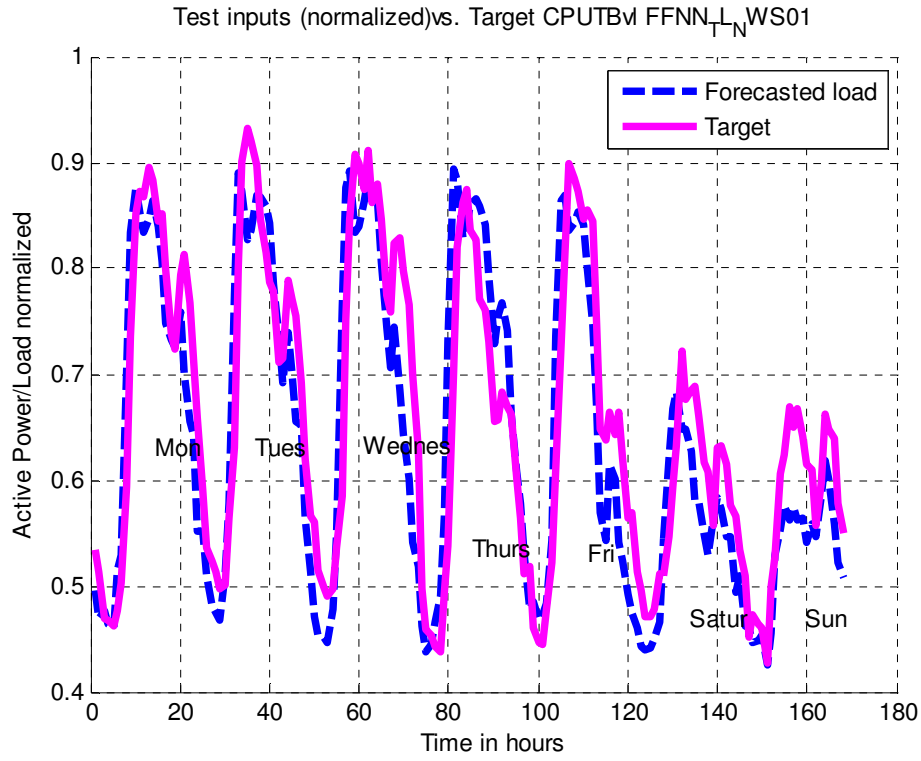
Some Simulation Results



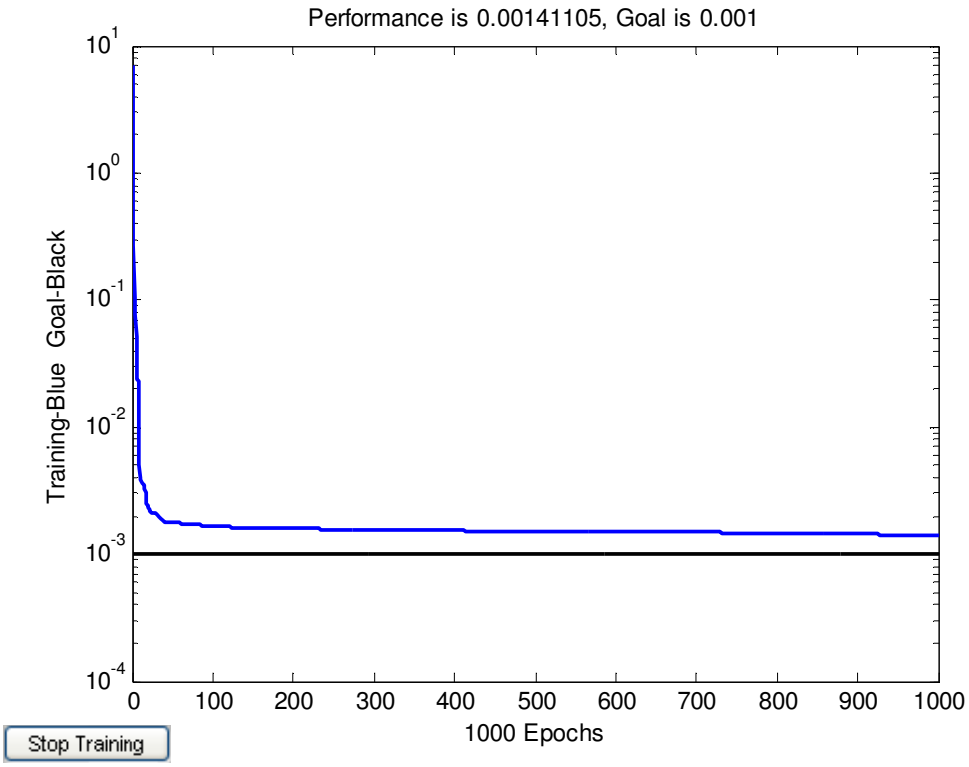
E.1: De-normalized forecasts for the combined load using Elman Recurrent Neural Network based model (non-weather sensitive) – in summer



E.2: Associated error for Elman Recurrent based model – ERNN_TL_NWS01



E.3: Forecasts after simulating ERNN_TL_NWS01 with the testing data



E.4: Goal vs. training response for ERNN_TL_NWS01 model. The was also not met in 1000 epochs.

The performance goal was then changed to 1e-5.

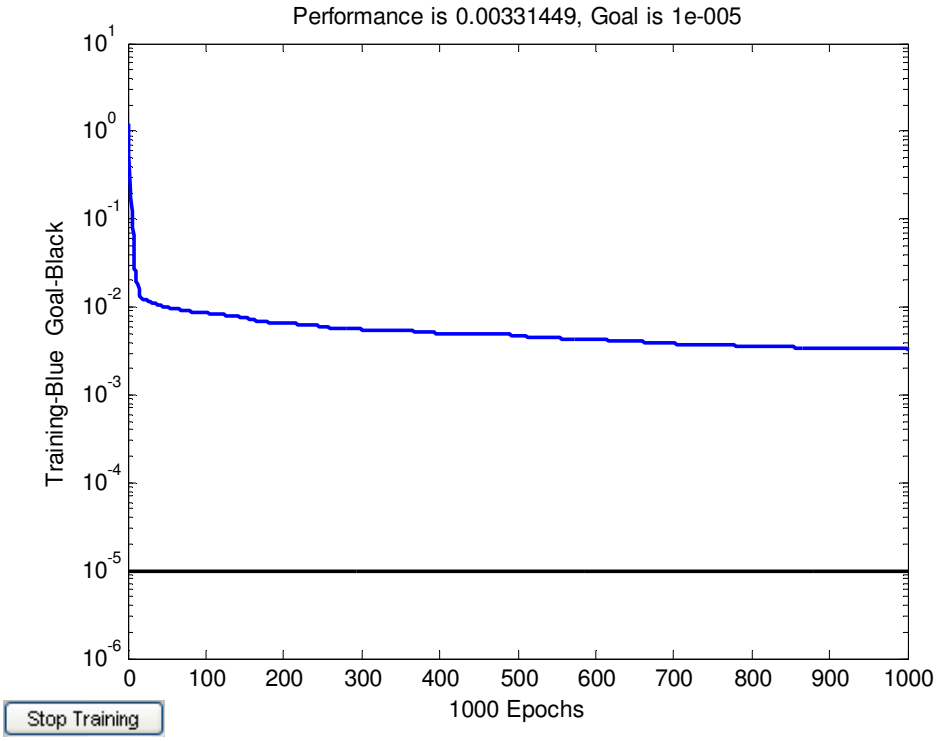


Fig. E.5: The network response of the FFNN_TL_WS01 (using batch training mode)

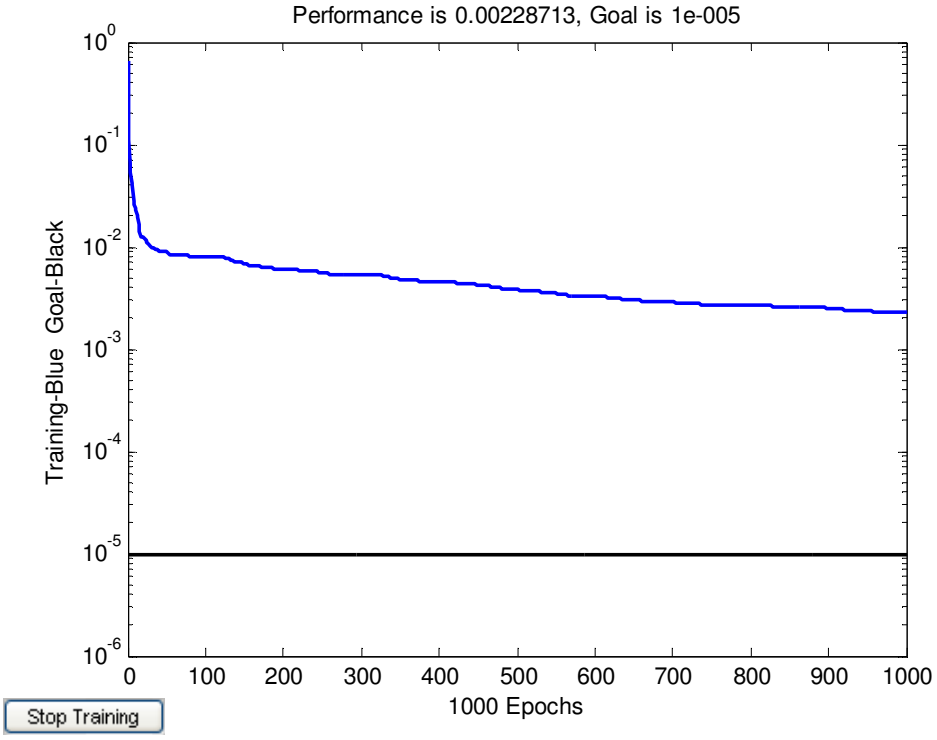


Fig. E.6: The network response of the ERNN_TL_NWS01 (using batch training mode)

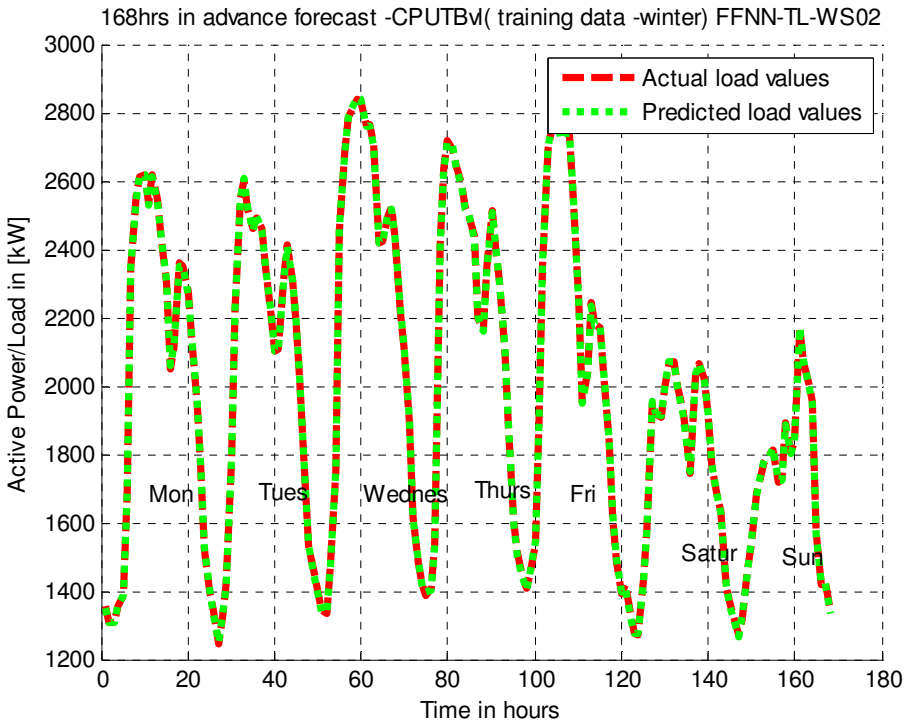


Fig. E.7: The network response of the FFNN_TL_WS02 (using incremental training mode) - winter

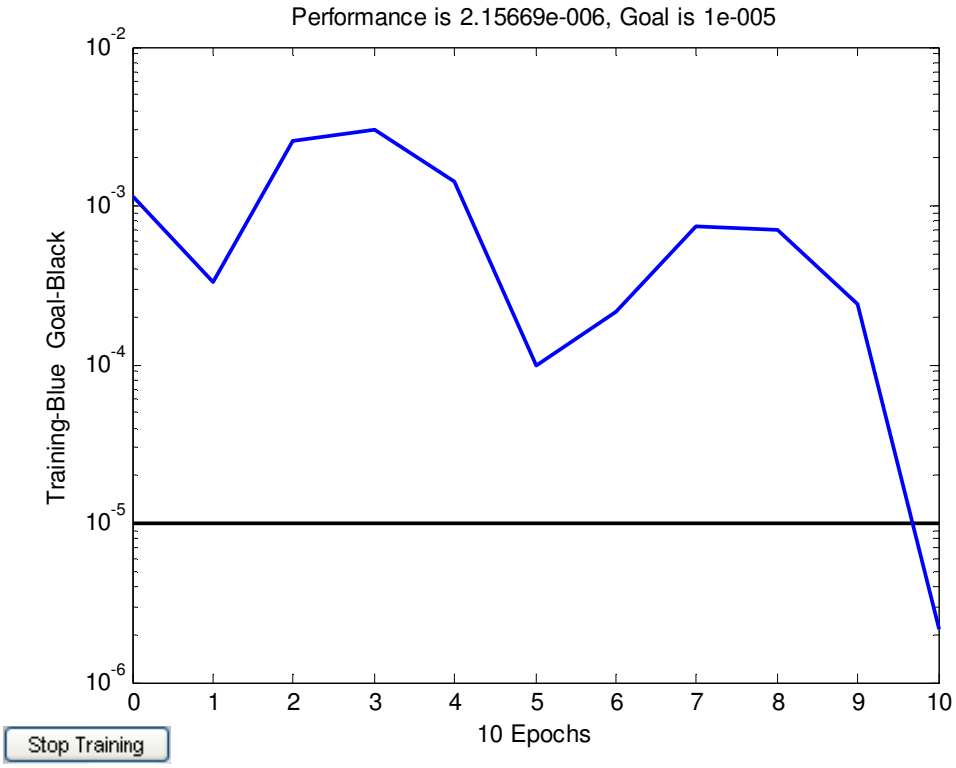


Fig. E.8: The corresponding network performance of the FFNN_TL_WS02 (using incremental training mode) – for the last forecast value (168h)

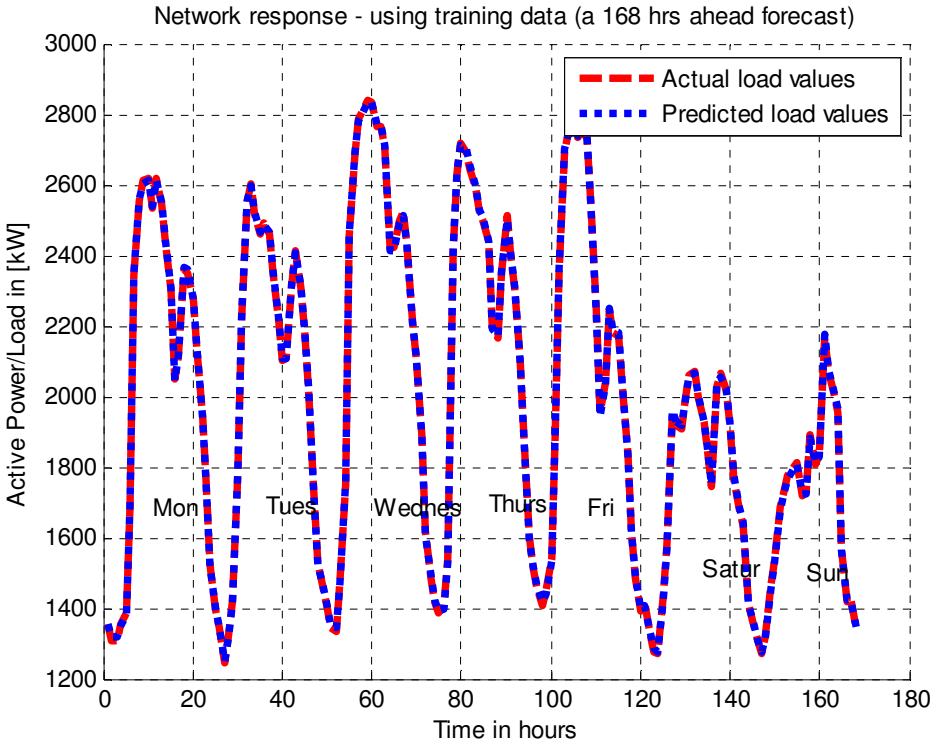


Fig. E.9: The network response of the ERNN_TL_WS02 (using incremental training mode) - winter

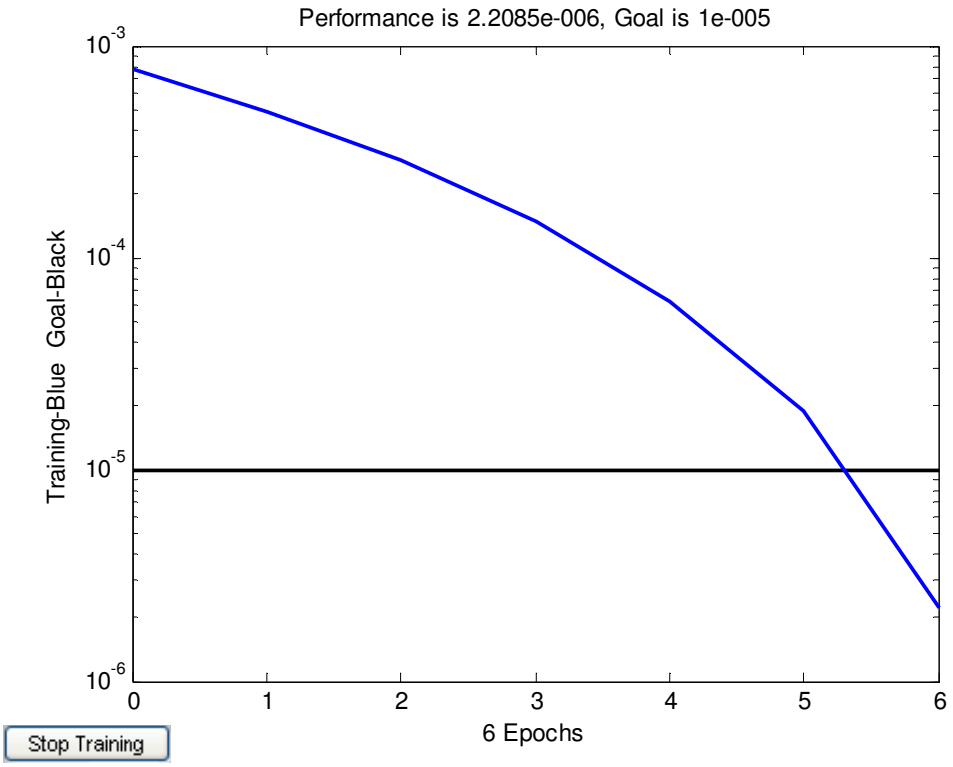


Fig. E.10: The corresponding network performance – for the last forecast value (168h) of the ERNN_TL_WS02 (using incremental training mode)

	Actual error (kW) - Elman Recurrent Prediction model (ERNN_TL_WS01)						
		batch training mode					
Hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	29.41	-155.48	-275.08	-105.28	-229.54	-149.52	-55.16
2	65.35	-284.95	-259.09	-172.16	-154.76	-220.10	-22.42
3	4.72	-258.92	-290.75	-166.33	-169.28	-189.64	-53.76
4	4.18	-238.11	-277.04	-163.82	-190.99	-209.69	-15.04
5	9.88	-222.68	-213.45	-162.58	-174.52	-151.55	-9.13
6	-30.94	-164.74	-160.67	-123.76	-75.28	-117.03	-32.67
7	-39.47	-155.24	-162.75	-115.66	-90.46	-62.48	-38.06
8	-113.78	-212.86	-310.30	-224.69	-196.97	-155.34	-20.72
9	-223.41	-128.05	-215.11	-228.03	-177.29	-179.65	-55.87
10	-130.68	9.60	-37.47	1.70	41.15	-126.00	8.23
11	-69.02	103.33	105.99	150.35	195.86	-64.52	42.82
12	99.45	82.07	173.53	180.36	227.16	91.83	97.10
13	233.57	69.03	165.33	219.37	251.83	67.33	84.18
14	223.61	15.23	286.77	229.43	225.36	74.19	113.21
15	166.39	29.93	235.04	203.70	226.93	135.14	90.58
16	185.83	-5.00	262.41	186.23	232.26	122.02	106.10
17	161.05	61.79	260.26	129.84	180.52	80.92	97.72
18	167.79	68.48	238.98	105.14	109.96	84.14	35.93
19	79.07	34.70	150.56	61.68	129.64	-0.53	47.89
20	120.33	16.96	137.23	51.35	54.74	20.21	54.57
21	88.41	-53.19	64.87	6.56	-21.88	33.69	59.31
22	3.35	-88.18	9.83	-51.71	8.17	7.81	67.75
23	-88.35	-194.77	-21.69	-123.58	-62.91	-51.86	40.37
24	-100.87	-262.34	-53.33	-222.63	-143.29	-38.16	-3.19

Fig. E.11: Daily actual errors (kW) for the batch trained Elman Recurrent model (**total load** in summer)

	Actual error (kW) - Elman Recurrent Prediction model (ERNN_TL_WS01) Total load						
		Incremental training mode					
Hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	-8.37	6.34	7.73	-5.43	3.21	7.44	3.12
2	4.40	-7.04	7.47	-8.85	4.95	7.59	-2.90
3	-3.22	2.65	-3.83	-0.70	1.08	-5.43	2.01
4	6.88	-7.32	5.22	2.35	0.89	0.43	1.45
5	5.06	5.26	0.36	5.76	3.37	-2.24	-7.05
6	1.52	4.58	-1.27	-6.30	-2.31	-8.87	-1.33
7	6.29	5.47	1.92	8.71	-2.23	3.22	2.75
8	3.12	-4.33	2.33	-8.79	6.89	-5.75	5.19
9	8.47	3.12	-3.37	2.97	7.75	-5.59	5.58
10	6.37	-4.50	7.32	8.57	-8.62	-8.17	-6.43
11	-7.19	-6.47	0.75	-2.17	-6.16	-7.12	-5.60
12	-8.54	7.65	0.56	0.39	7.75	-1.77	-3.49
13	-2.68	4.57	2.70	5.39	6.85	-6.49	5.85
14	4.03	-0.64	3.07	-7.03	-8.55	-1.85	-6.48
15	4.50	-5.10	5.69	3.93	-6.94	-4.59	7.79
16	7.86	7.48	-4.82	6.04	-0.86	5.16	-5.89
17	-6.46	0.07	0.29	-6.35	-3.18	5.35	6.64
18	2.95	-0.70	-0.88	3.29	-7.53	6.83	-8.53
19	-5.44	4.58	-3.57	-6.22	2.75	7.95	-8.31
20	-7.77	-0.82	-0.83	7.08	-2.15	-1.65	-6.34
21	2.60	3.38	-8.42	-0.29	7.08	-8.85	-6.50
22	-0.23	5.54	-2.82	7.24	-5.52	0.70	3.32
23	-7.92	-6.38	7.40	-5.50	-1.11	8.45	-3.08
24	-7.60	-2.68	-3.74	2.99	-4.90	5.73	-4.12

Fig. E.12: Daily actual errors (kW) for the incrementally trained Elman Recurrent model (**total load** in summer)

Actual error (kW) - ERNN Prediction model (ERNN_TL_NWS01) - TOTAL LOAD (WINTER)								
		incremental training mode, IV # 1						
hour	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	0.62	-6.38	-3.68	-6.77	-9.09	8.14	-6.62
2	-6.69	-3.27	-9.09	-3.85	-9.18	-4.31	-3.39
3	-7.79	-5.27	-3.49	-5.45	8.83	-8.60	-6.58
4	6.00	3.71	-7.62	8.31	5.20	-7.93	7.17
5	5.73	5.27	7.08	7.48	-6.83	8.02	8.15
6	-8.24	-6.84	-7.60	-7.01	-8.33	8.73	4.16
7	-5.16	-8.14	-5.54	-5.76	-8.87	-7.04	8.23
8	-5.53	-8.54	-6.08	4.13	7.99	-5.95	8.45
9	8.44	7.27	5.12	-8.38	3.42	-8.52	8.51
10	7.01	-4.03	7.12	-8.81	-3.74	4.44	9.17
11	-4.16	-3.31	6.25	-3.45	8.76	3.67	7.16
12	4.14	7.48	-3.78	-3.85	-6.97	8.01	-4.66
13	-3.69	-6.99	-3.66	-8.38	5.15	-3.54	3.17
14	-6.12	-6.36	-5.16	-3.62	7.95	-4.01	8.84
15	-7.47	-7.61	-3.28	6.62	7.75	-4.63	-3.83
16	6.93	-6.44	7.92	-6.49	5.10	-5.55	6.29
17	3.76	3.52	8.98	-4.83	-5.64	-7.55	-9.12
18	-6.75	-5.60	4.44	8.26	-3.69	7.52	-5.69
19	-6.92	4.95	9.27	-5.62	-8.73	-5.87	-3.64
20	-4.54	-5.42	-4.47	-5.20	-8.01	-5.73	-9.15
21	-7.79	-6.98	5.96	5.49	5.46	-8.40	7.29
22	5.36	-9.17	-5.16	5.78	5.73	-5.13	-7.94
23	4.96	8.25	6.06	7.64	-6.73	-8.08	5.15
24	6.10	4.94	7.30	-4.30	-5.51	6.22	-4.39

Fig. E.13: Daily actual errors (kW) for the incrementally trained Elman Recurrent model (**total load** in winter)

Actual error (kW) - FFNN Prediction model (FFNN_TL_NWS02) - TOTAL LOAD WINTER							
		incremental training mode, IV # 1					
hour	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	5.09	-4.78	8.84	-4.65	4.86	8.21	6.16
2	4.18	6.54	4.97	7.68	4.88	8.84	5.55
3	3.08	-3.17	5.61	-9.23	-4.95	4.56	3.61
4	-5.39	-7.84	-3.39	3.94	-9.17	-4.99	-6.13
5	-2.75	3.10	4.87	4.83	-4.10	5.45	4.96
6	-3.44	-4.24	-3.24	3.31	-5.23	5.73	-8.19
7	-7.60	-5.11	-8.15	8.00	-3.85	-4.36	5.04
8	7.23	-3.70	7.95	3.24	-6.13	3.86	-5.55
9	-7.30	-5.61	3.92	-8.02	-5.47	-3.88	-4.41
10	0.29	-3.15	-2.61	6.12	8.66	3.13	7.69
11	9.28	6.70	-3.30	5.44	8.46	-7.39	-2.60
12	3.24	-3.98	-5.33	6.93	-7.27	2.07	-3.46
13	7.67	3.95	7.68	-7.43	-6.83	7.92	4.37
14	-3.68	-3.85	6.19	7.33	3.55	6.96	5.94
15	-4.90	-4.98	5.48	-8.38	3.38	8.30	8.17
16	-9.09	-4.20	3.30	4.15	3.24	-3.24	-3.75
17	-5.95	5.76	6.60	6.77	7.40	-3.27	-3.80
18	8.60	7.49	-8.86	5.92	6.15	-3.37	-3.60
19	-0.49	3.77	7.67	-3.83	-1.07	5.30	7.24
20	8.62	-3.66	-3.04	-3.40	-5.07	-3.42	5.20
21	-4.72	-4.59	-7.84	-7.20	-7.12	-5.52	4.38
22	-6.99	-6.03	-3.93	-8.11	-8.04	-3.32	-5.39
23	-7.00	3.51	-7.97	3.45	-4.98	5.44	7.70
24	-8.51	-6.72	3.30	9.29	-3.61	-7.75	8.73

Fig. E.14: Daily actual errors (kW) for the incrementally trained Feedforward model (**total load** in winter)

	Actual error (W) - FFNN Prediction model (FFNN_IT_WS01) – for the IT Department						
	Incremental training mode, IV # 1						
hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	-959.22	804.7064	693.268	778.0462	900.8662	-499.539	-822.761
2	891.8208	804.7064	768.6587	594.7778	369.7553	551.0776	177.2389
3	891.8208	-331.95	-231.341	-405.222	-886.369	-883.843	384.5636
4	770.1193	357.2808	750.7299	-857.186	-823.77	924.7608	-615.436
5	773.5422	-642.719	750.7299	-857.186	176.2302	-906.88	690.2619
6	773.5422	827.9413	765.069	-811.59	940.9841	93.12046	738.3879
7	775.0379	827.9413	771.3351	659.3814	935.2891	882.7766	-261.612
8	-884.428	-450.5	-857.343	-962.261	-498.916	822.7444	903.3944
9	66.45194	-844.665	-928.641	-618.639	-715.328	-869.94	-433.824
10	753.7812	-741.811	-853.473	-569.682	872.549	-957.489	-433.824
11	-441.841	-746.503	757.9851	933.9338	-901.961	868.3326	-433.824
12	497.6494	613.8098	-655.939	-954.981	-542.342	766.4216	-433.824
13	348.1506	883.7821	668.9949	-954.981	-564.869	766.4216	-398.312
14	-463.575	368.8399	-660.708	591.8032	-726.819	-823.922	699.9378
15	556.7394	-355.304	-891.165	-851.412	-360.43	-579.957	-960.555
16	559.9375	644.6958	887.2615	-851.412	-394.379	-406.142	430.7615
17	695.615	712.6196	515.7534	940.4584	739.2318	-450.409	872.7645
18	-712.862	-711.951	-829.661	-967.648	-442.62	-699.347	-649.751
19	504.9628	720.8167	-580.264	-859.862	-398.783	700.4323	-574.58
20	709.2163	743.1982	-963.961	-421.139	-849.93	700.4323	696.6938
21	-832.928	-827.578	-938.906	-344.051	920.0557	700.4323	696.6938
22	-808.518	949.3483	-777.532	-344.051	-908.936	770.7034	-552.061
23	492.4888	-431.362	-363.147	-819.871	913.7951	773.7974	824.4413
24	688.3151	686.2602	-876.754	615.7897	676.9462	-226.203	-384.474

Fig. E.15: Daily actual errors (W) for the incrementally trained Feedforward Neural Network model (**IT department load forecasting model)**

	Actual error (W) - ERNN Prediction model (ERNN_IT_WS) – For the IT Department						
		batch training mode, IV # 1					
Hours	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	149.64	1,223.56	1,660.55	1,012.54	520.11	1,311.40	2.13
2	-530.01	213.08	-751.40	-192.28	636.31	783.03	-22.04
3	255.15	-262.34	-917.41	216.18	454.92	904.80	219.29
4	445.81	-309.00	-334.80	-276.05	329.88	834.96	314.62
5	758.46	306.81	3.72	-150.65	340.50	926.88	503.54
6	807.85	364.45	-731.07	-226.48	460.01	1,051.49	632.15
7	66.50	-873.00	-3,264.73	-484.92	-34.01	541.08	658.86
8	-555.45	-187.61	-1,981.04	-307.79	-1,243.55	-516.94	611.65
9	2,724.83	913.16	53.65	-577.50	-978.80	-2,265.58	544.61
10	-566.28	991.62	-399.92	-1,813.75	-1,254.37	-1,242.68	509.08
11	83.66	1,260.35	197.77	2,029.06	-1,315.92	-904.52	494.11
12	67.51	571.29	-28.48	1,708.50	-652.42	-1,262.35	556.30
13	588.69	638.93	-298.31	1,463.11	-375.03	-1,334.06	441.87
14	89.18	641.63	-332.42	1,235.14	-498.98	-1,237.09	-464.26
15	567.71	34.68	-628.84	1,280.10	-1,016.18	-817.63	-647.04
16	255.61	-823.82	-802.22	1,098.01	-1,223.82	-672.05	-790.84
17	250.44	-427.94	-487.90	59.36	-1,211.63	-463.98	-946.12
18	140.33	472.93	-554.65	-104.56	-1,012.18	166.61	-585.69
19	250.23	434.56	-585.45	-358.29	71.42	193.61	-1,054.59
20	-218.27	276.39	-551.77	-453.09	46.67	-805.10	-725.01
21	6.15	782.48	-491.20	-252.63	-202.98	-615.97	-452.21
22	161.86	713.47	-683.10	-372.19	208.33	-608.96	1,102.10
23	158.40	853.60	-442.08	-183.12	272.81	-548.08	1,194.57
24	716.07	1,858.28	2,152.41	1,565.31	3,994.72	-856.38	1,198.68

Fig. E.16: Daily actual errors (W) for the batch trained Elman Recurrent Neural Network model (**IT department load forecasting model)**

Actual error (W) - ERNN Prediction model (ERNN_ELEC_WS) – for the ELECTRICAL ENGINEERING Department							
		batch training mode, IV # 1					
hour	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
1	760.85	1,970.83	-111.01	3,837.95	1,391.44	474.69	-618.99
2	-683.15	3,429.56	-800.63	-962.05	-336.94	1,723.78	116.31
3	-208.41	1,931.77	4,069.21	-1,981.71	1,503.31	184.49	339.50
4	1,182.45	474.96	3,730.08	4,974.60	1,521.62	-1,211.36	-974.66
5	-1,224.57	1,042.60	1,311.56	1,352.66	-692.11	1,246.95	260.13
6	546.93	545.87	-662.67	-1,771.66	1,350.35	-538.28	347.91
7	132.08	629.18	2,852.52	-2,049.31	-1,407.86	193.58	214.85
8	-5,594.92	-3,568.73	-6,190.41	-17,934.60	-4,787.47	-1,087.68	121.34
9	3,818.99	1,828.64	-1,096.91	3,513.68	-500.16	-1,674.62	274.97
10	83.37	1,024.67	927.77	-2,598.72	-1,119.21	-1,075.37	154.94
11	-1,523.04	1,739.88	1,085.49	-747.93	-1,093.92	-1,516.41	429.46
12	-786.39	507.05	-4,248.54	2,190.06	-1,749.94	-1,210.31	176.14
13	-1,183.73	171.50	-305.55	850.68	264.45	-2,006.99	300.65
14	-236.46	290.82	14.87	-300.32	-453.93	-920.25	326.69
15	-609.31	-56.14	-1,069.47	137.95	-176.76	-1,670.86	196.93
16	546.16	419.95	795.57	11,979.18	-411.20	-1,854.44	271.08
17	678.25	3,600.28	309.68	-10,113.83	72.60	-1,713.56	287.45
18	5,760.62	1,358.80	-2,514.91	-8,362.70	-4,607.35	-1,775.83	441.25
19	-1,901.44	669.15	1,288.86	3,331.33	64.22	-1,394.13	342.19
20	-2,603.59	1,475.49	3,281.31	977.66	-1,486.95	-1,558.07	1,274.34
21	-997.19	1,543.70	1,851.23	-395.88	937.50	-1,658.21	1,455.65
22	303.44	492.22	2,719.85	3,314.37	1,630.97	-1,712.31	1,106.98
23	-721.67	2,216.84	1,070.07	2,560.69	1,489.04	-1,822.57	1,347.60
24	-2,956.74	2,613.45	2,586.47	1,633.63	117.36	-1,897.00	-385.78

Fig. E.17: Daily actual errors (W) for the batch trained Elman Recurrent Neural Network model (**Elec. Engineering department load forecasting model**) –

Actual error (W) - FFNN Prediction model (FFNN_ELEC_WS) – for the ELECTRICAL ENGINEERING Department							
		Incremental training mode, IV # 1					
hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	161.7226	171.5153	-104.107	-108.095	-201.782	83.94322	182.7599
2	-144.271	88.29698	-219.548	106.7704	176.1961	-86.2338	193.104
3	-171.13	-232.636	232.1807	171.7259	202.7869	-242.444	-106.896
4	106.8243	88.14866	92.38903	220.4901	-128.513	114.6688	-146.722
5	6.824322	-93.7843	-232.49	-124.308	-151.067	154.1452	-84.1037
6	-242.751	-153.399	88.15389	-193.788	219.6758	-158.832	115.8963
7	191.7026	108.8424	-93.7843	-196.27	140.8357	112.2209	215.8963
8	-108.297	-77.9727	-153.399	-196.27	-114.107	204.8884	15.89627
9	-231.809	-212.449	-252.158	132.0248	-223.625	-153.447	15.89627
10	-131.809	221.3694	-97.955	-168.523	-206.401	-247.121	-184.104
11	-31.8092	-134.351	-114.25	-234.462	154.7041	-90.9981	166.3585
12	-249.501	-34.3506	-132.695	-172.209	103.9825	-190.998	80.70056
13	123.272	65.64937	-92.3831	101.0252	-198.438	241.0098	232.2802
14	-95.6284	84.51483	76.16214	-175.02	-185.719	115.7878	172.9121
15	-158.251	135.205	152.7695	-117.003	214.2814	-103.041	-127.088
16	83.40765	-95.2021	-108.865	-114.93	-234.287	-161.216	72.91209
17	135.1654	-117.724	-225.39	-106.868	91.77162	38.78363	-27.0879
18	-95.2035	142.8274	233.7928	199.8194	-89.6086	135.5135	-27.0879
19	-96.6394	105.289	-226.483	198.0199	-111.435	-80.9065	-130.835
20	-96.6394	202.1156	123.9131	122.9328	241.3142	-152.363	-106.608
21	121.3442	-197.884	-238.509	222.9328	-94.9838	-152.363	-242.643
22	74.65214	2.115596	103.236	107.4774	164.4482	-152.363	114.6908
23	-250.139	107.2531	142.644	76.17749	-108.448	82.75989	-155.616
24	71.51531	-114.064	-125.691	-104.456	-150.35	-17.2401	72.11776

Fig. E.18: Daily actual errors (W) for the batch trained Elman Recurrent Neural Network model (**Elec. Engineering department load forecasting model) –**

Actual error (kW) - ERNN Prediction model (ERNN_MEC_WS) - MECHANICAL. ENG Department - for 24hrs lead time forecast. Forecast day is Tuesday, 2nd of Sep, 08																							
kW	4.43	8.07	1.31	-2.38	1.54	-0.46	-9.92	-1.26	-1.60	-4.89	-5.17	-0.85	8.20	17.99	-5.98	-2.23	6.29	-3.21	-5.08	-6.56	-0.69	3.63	-1.56	0.37
Hr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Fig. E.15: Daily actual errors (kW) for the **batch trained** Elman Recurrent Neural Network model (**Mec. Engineering department load forecasting model**) –

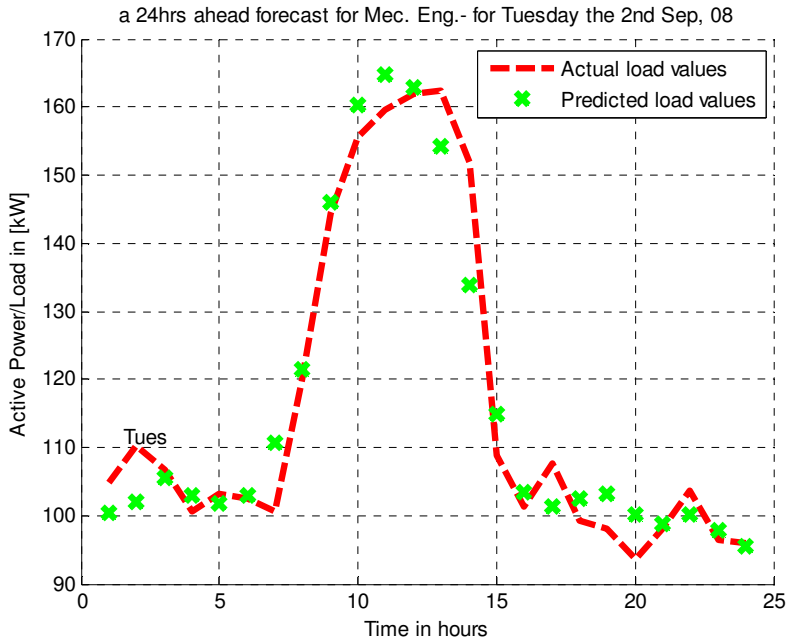


Fig. E.19(a): One day forecast for the Mechanical Engineering Department electric load

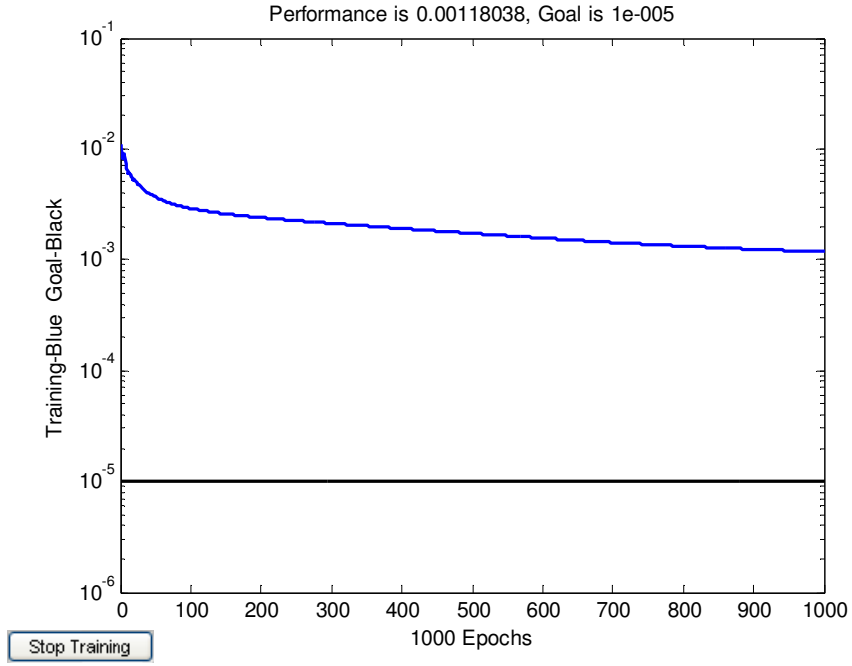


Fig. E.19 (b): The corresponding model performance

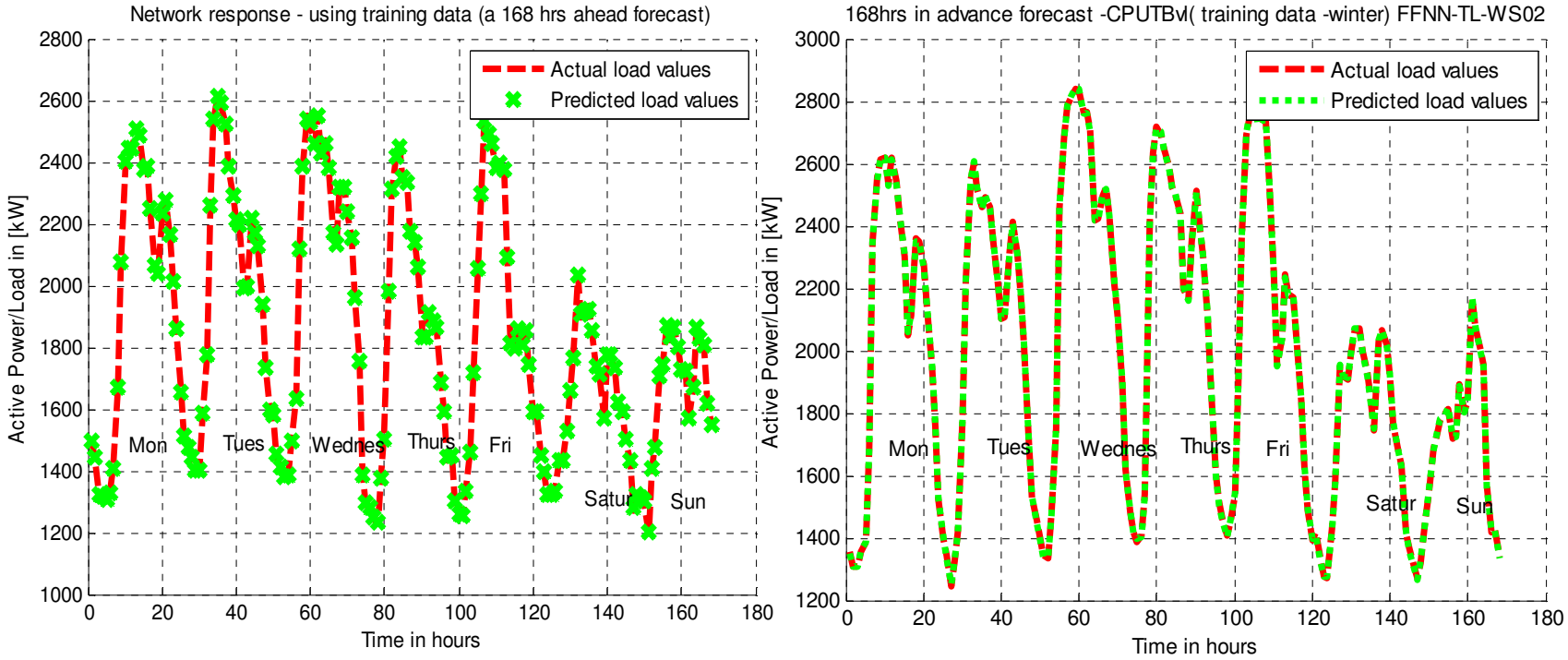


Fig. E.20: 1 week ahead forecast in both seasons – summer(left) and winter (right).

APPENDIX F

List of Machines (Mechanical Engineering Department)

G.1 List of Machines

Note: *These machines were not in use when electric load measurements were taken at the department.*

LIST OF MACHINES (MECHANICAL ENGINEERING DEPARTMENT)			
Main Workshop			
Name	Qty	Rating	Type
Colchester 1800 lathe machine	9	3HP	3-phase
High speed-gap bed lathe machine	6	3kVA	3-phase
Others	4	6kVA	3-phase
Thermodynamic Lab			
Diesel engine test bench		no details obtained	
Compressor			
Gas turbine			
Boiler			
Steam turbine			
Air conditioner unit			
Refrigeration unit			
CNC Workshop			
Fanuc lathe machine	2	20kVA	3-phase
Universal cylindrical grinder	1	8kVA	3-phase
63 Ton Press brake	1	7.55kW	3-phase
Milling machine	2	20kVA	3-phase