

**HYBRID ARTIFICIAL NEURAL NETWORK FOR
SHORT-TERM ELECTRICITY LOAD FORECASTING**

BY

K. DARSHANA ABEYRATHNA

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE
(ENGINEERING AND TECHNOLOGY)**

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

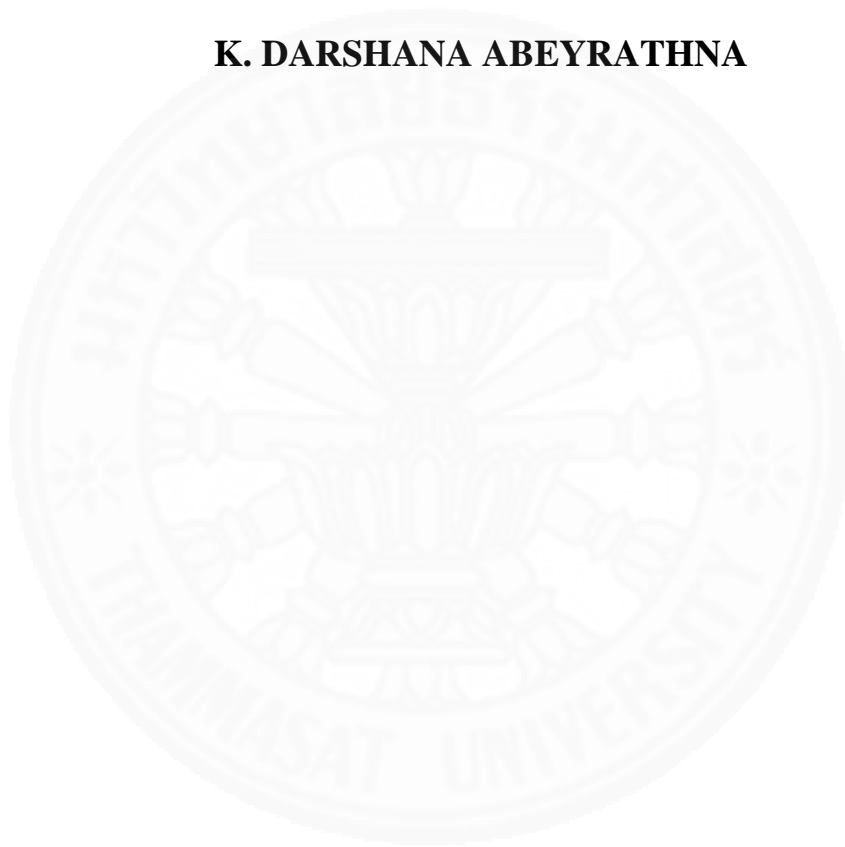
THAMMASAT UNIVERSITY

ACADEMIC YEAR 2016

**HYBRID ARTIFICIAL NEURAL NETWORK FOR
SHORT-TERM ELECTRICITY LOAD FORECASTING**

BY

K. DARSHANA ABEYRATHNA



**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE
(ENGINEERING AND TECHNOLOGY)
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2016**

HYBRID ARTIFICIAL NEURAL NETWORK FOR SHORT-TERM
ELECTRICITY LOAD FORECASTING

A Thesis Presented

By

K. DARSHANA ABEYRATHNA

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE (ENGINEERING AND TECHNOLOGY)

Approved as to style and content by

Advisor and Chairperson of Thesis Committee



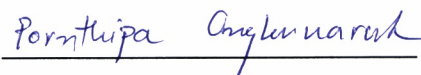
(Assoc. Prof. Dr. Chawalit Jeenanunta)

Committee Member



(Asst. Prof. Dr. Somsak Kittipiyakul)

Committee Member



(Assoc. Prof. Dr. Pornthipa Ongkunaruk)

MAY 2017

Acknowledgements

I would like to express my special appreciation and thanks to my supervisor Assoc. Prof. Dr. Chawalit Jeenanunta, for his continuous support, encouragements, motivations, ideas, and sharing his immense research knowledge from the starting date of my MSc degree to thus far. His guidance, advice, and experience were priceless and helped me to finish my MSc dissertation successfully. Most importantly, his easy nature, calmness, and the ability of handling and dealing with different kind of people will definitely shape my future.

Besides my advisor, I would also like to thank the rest of my thesis committee: Asst. Prof. Dr. Somsak kittipiyakul and Assoc. Prof. Dr. Pornthipa Ongkunaruk for their comments and encouragements. Their comments and hard questions helped to widen my research knowledge and showed the right way for reaching the research objectives.

I must appreciate the support that I had from SIIT, Thammasat University. They gave me a precious opportunity to pursue my MSc degree and supported with the EFS scholarship for my studies. All the faculty members and staffs of the department and members of my research group have to be mentioned since they helped me to complete this dissertation in many aspects.

Finally, a special thanks to my parents and other family members for all the sacrifices they have done to bring me thus far. They were always there with me when I had really hard time during my MSc studies. I appreciate their motivations and encouragements. I would also like to thank all of my friends and every person that I could not mention here for their supports in numerous ways. Without all their precious supports it would not be possible to conduct this research

Abstract

HYBRID ARTIFICIAL NEURAL NETWORK FOR SHORT-TERM ELECTRICITY LOAD FORECASTING

by

K. DARSHANA ABEYRATHNA

Bachelor of Science in Engineering, School of Engineering and Technology, Asian Institute of Technology, Thailand, 2015

Master of Science (Engineering and Technology), Sirindhorn International Institute of Technology, Thammasat University, 2017

Numerous research have been being conducted by many researchers in the field of electricity forecasting with the purpose of matching the forecasted outcomes with the actual consumption for the next day. This is important since both the under and over forecasting lead to revenue reductions. Even though the existing forecasting techniques can forecast the electricity consumption up to a considerable level, small errors can reduce the revenue in large amounts. Therefore, the electricity forecasting field is still active and strategies and improvements for the existing techniques for reducing the forecasting errors are always appreciated. Thus, the objective of this research is to improve the electricity forecasting outcomes or further reducing the electricity forecasting errors for the case of Thailand.

Based on the literatures, Artificial Neural Network (ANN) is selected as the technique to forecast the daily electricity consumption. ANN has the ability of

recognizing and learning non-linear patterns in data. However, ANN has many internal and external parameters and training methods which have to be arranged in the optimum way. These optimally arranged parameters and training algorithms can significantly improve the forecasting outcomes. Nevertheless, this research mainly focuses on finding the best training algorithm to train the ANN for electricity demand forecasting. The main experiment is supported by two other experiments where they help to find the optimum number of inputs and outputs to the ANN, the amount of data needed to train the ANN, and the optimum number of hidden units of ANN for electricity demand forecasting.

Due to the limitations of using backpropagation training algorithm to train the ANN, two stochastic optimization techniques are introduced to optimize the weights and bias of the ANN: Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Using the positive characteristic of both the algorithms, a combined algorithm is created to train the ANN. An experiment is arranged to find the best training algorithm out of these four algorithms. A sample data set is selected from the data gathered by Electricity Generating Authority of Thailand (EGAT). The sample set consist with data from 1st January, 2012 to 31st December, 2013 where one year data is used for training the ANN and one year data is used for testing. The experiment is started with cleaning the selected data set using the time window based data cleaning technique. ANN with four inputs ($L_t(d-7)$, $L_t(d-1)$, $T_t(d-1)$, $T_t(d)$) and one output ($F_t(d)$) is created to forecast each time period t of each day d . The ANN consist with one hidden layer and four hidden neurons. Backpropagation training algorithm, GA, PSO, and hybridized training algorithm are used to train the created ANN with one year data, respectively.

Forecasting outcomes from each training algorithm are evaluated in term of the percentage error (Mean Absolute Percentage Error). According to the yearly average MAPE, the hybridized training algorithm performs well compared to the other three training algorithms. Genetic algorithm also gives competitive results compared to the hybridized training algorithm. However, Particle Swarm Optimization and backpropagation training algorithms are fairly good compared to the other two training algorithms. Regardless of the training algorithm, days in December are hard to forecast

as their electricity consumption is significantly low compared to the other months. Therefore, forecasted outcomes are always higher than the actual consumption.

Findings of this research will add a great value to the electricity forecasting filed and the users of ANN. The research will help to define the inputs, outputs, and required data for electricity forecasting and most importantly for selecting the best training algorithm for the ANNs.

Keywords: Short-Term Electricity Demand Forecasting, Artificial Neural Networks, Genetic Algorithm, Particle Swarm Optimization, Backpropagation

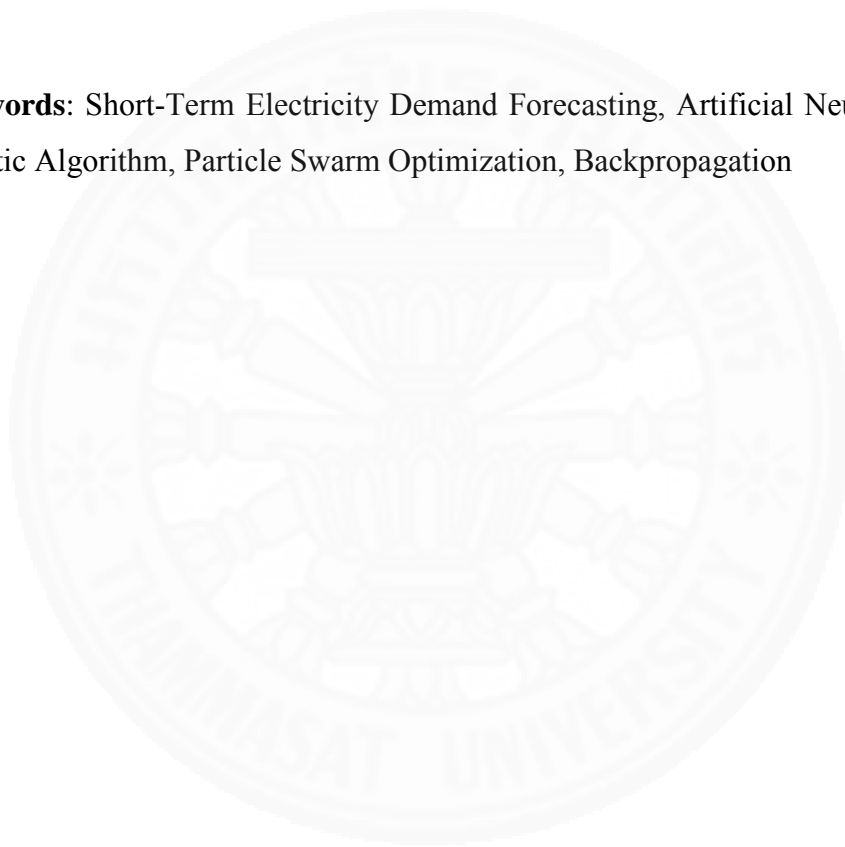


Table of Contents

Chapter	Title	Page
	Signature Page	i
	Acknowledgements	ii
	Abstract	iii
	Table of Contents	vi
	List of Tables	ix
	List of Figures	x
1	Introduction	1
	1.1 The Background of the Energy Sector of Thailand	1
	1.2 The Analysis of Actual Consumption Data	3
	1.3 Electricity Forecasting	5
	1.4 Problem Statement	9
	1.5 Objective	9
	1.6 Significance of Study	10
2	Literature Review	11
	2.1 Time Series	11
	2.2 Related Works and Models	14
3	Method and Procedure	35
	3.1 Artificial Neural Network	35
	3.2 Training Algorithms	37
	3.2.1 Backpropagation	37
	3.2.2 Genetic Algorithm	42

3.2.3 Particle Swarm Optimization	46
3.3 Data Cleaning	49
3.3.1 Detecting and Replacing Holidays	50
3.3.2 Detecting and Replacing Bridging Holidays	50
3.3.3 Detecting and Replacing Outliers	51
3.3.3.1 Detecting Outliers	51
3.3.3.2 Replacing Outliers	52
3.4 Evaluating the Accuracy	53
4 Design of Experiments	56
4.1 Case 1: Study of Neural Networks' Parameters	56
4.1.1 Data Arrangement	56
4.1.2 ANN structure	57
4.1.3 Training and Testing	57
4.2 Case 2: Effect of the Amount of Training Data and Inputs	58
4.2.1 Data Arrangement	59
4.2.2 ANN structure	60
4.2.3 Training and Testing	60
4.3 Case 3: Performances of Training Algorithms	61
4.3.1 Data Arrangement	61
4.3.2 ANN structure	62
4.3.3 Training and Testing	63
4.3.3.1 Training ANN with Backpropagation	63
4.3.3.2 Training ANN with GA	65
(1) Encoding	65
(2) Fitness Value Calculation	66
(3) The Next Generation	67
(4) Stopping Conditions	68
4.3.3.3 Training ANN with PSO	71
(1) Encoding	71
(2) Fitness Value Calculation	71

	(3) Parameters and Stopping Conditions	72
	4.3.3.4 Training ANN with PSO-GA	75
5	Results and Discussion	77
	5.1 Case 1: Study of Neural Networks' Parameters	77
	5.2 Case 2: Effect of the Amount of Training Data and Inputs	80
	5.3 Case 3: Performances of Training Algorithms	82
6	Conclusions and Recommendations	91
	6.1 Case 1: Study of Neural Networks' Parameters	91
	6.2 Case 2: Effect of the Amount of Training Data and Inputs	92
	6.3 Case 3: Performances of Training Algorithms	92
	References	94
	Appendices	101
	Appendix A – MATLAB code: Data Cleaning	101
	Appendix B – MATLAB code: Training ANN with Backpropagation	102
	Appendix C – MATLAB code: Training ANN with GA	104
	Appendix D – MATLAB code: Training ANN with PSO	107
	Appendix E – MATLAB code: Training ANN with PSO+GA	112

List of Tables

Tables	Page
Table 2.1 List of research/techniques on electricity demand forecasting	19
Table 3.1 Identifying bridging holidays	50
Table 4.1 Data arrangement for the 5-input case	57
Table 4.2 Number of electricity demand values for training each case	59
Table 4.3 Example Data arrangement for 1 input and 3 input cases	60
Table 5.1 Average of 5 replications MAPEs with different number of inputs and different NN settings	77
Table 5.2 MAPEs and computational times for different transfer functions	79
Table 5.3 Average MAPE values for Load Forecasting of Weekdays in November for all 20 Cases	80
Table 5.4 Monthly (2013) average $MAPE^p$ for different training algorithms	83
Table 5.5 Yearly average $MAPE^p$ for different categories of days by different training algorithms	86
Table 5.6 Monthly average $MAPE^p$ for different categories of days by different training algorithms	86

List of Figures

Figures	Page
Figure 1.1 Electricity consumption during 2010 to 2014 by each sector (WONGLA, 2013)	2
Figure 1.2 Electricity generation of all the months from 2011 January to 2014 April	2
Figure 1.3 Total daily load in November 2013	3
Figure 1.4 Electricity consumption pattern on 10 th Thursday, 2013	4
Figure 1.5 Demand curves for different time horizons (J. V. RINGWOOD, D. BOFELLI, F. T. MURRAY, 2001)	6
Figure 2.1 A sample time series	11
Figure 2.2 Separation of time series (Verbesselt, Jan, et al.)	12
Figure 2.3 Categorization of Forecasting Techniques	34
Figure 3.1 The basic structure of NNs	35
Figure 3.2 An NN structure for BP explanation	37
Figure 3.3 Back propagation stops at a local minimum	42
Figure 3.4 A sample optimization problem	43
Figure 3.5 Three methods for producing children for the next generation	44
Figure 3.6 A flock of birds that illustrates PSO	47
Figure 4.1 The suggested NN structure	63
Figure 4.2 Flowchart for adjusting weights with backpropagation	64
Figure 4.3 Encoding NN weights and bias into chromosomes	65
Figure 4.4 Selecting parents from the current population	67
Figure 4.5 Flowchart for hybridizing GA and NN	70
Figure 4.6 Flowchart for hybridizing PSO and NN	74
Figure 4.7 Flowchart for hybridizing PSO and GA	76
Figure 5.1 MAPEs for three cases with different number of neurons with 1 layer	78
Figure 5.2 MAPEs with 5 input case for 1 layer and 2 layers	78
Figure 5.3 Actual and Forecasted series with 2 months training data and one input	81

Figure 5.4 Actual and Forecasted series with 10 months training data and 20 inputs	81
Figure 5.5 Performances of the hybridized training algorithm	83
Figure 5.6 Actual vs Forecasted load with different training algorithms for 23 rd Monday of December, 2013	84
Figure 5.7 Actual vs Forecasted load with different training algorithms for 25 th Wednesday of September, 2013	85



Chapter 1

Introduction

1.1 Background of the Energy Sector of Thailand

Among the Southeast Asia's countries, Thailand has reached to the second of the list with its economy and the Gross domestic product (GDP) of \$404,824. Electricity had played an important role for carrying its economy to the top of the list. In 2012, Thailand Power Development Plan (PDP) had published its new policies, strategies, and targets for next 20 years (Energy Policy and Planning Office, 2012). Some of their strategies were included, energy efficiency promotions and energy saving programs around the country. They are expecting to reduce the total power demand of the country by 25% within the next 20 years. During the period of 2012 to 2022, they are planning to increase the usage of alternative energy by 25%. They will build renewable power plants so that they can replace conventional power plants with them. For all these policies, strategies, and targets, the central idea was its daily accretion of power demand. To build the policies for the next 20 years, they forecast the total power demand for 2030.

- Demand at end of 2011 – 32,395 MW
- Added capacity to the period of 2012 – 2030 – 55,130 MW
- Retired capacity to the period of 2012 – 2030 - -16,839 MW
- Total demand by end of 2030 – 70,686 MW

This is a depression of demand than they forecasted before. As they specified, 6.27% of demand has decreased as they work on a target to reduce the demand during the last 10 years. Therefore, added capacity has been reduced by 3,494 MW, than they estimated.

Figure 1.1 shows that industrial sector has consumed the highest amount of electricity during the period of 2010 to 2014. The residential and businesses sectors are the second and third highest, respectively in the electricity consumption list.

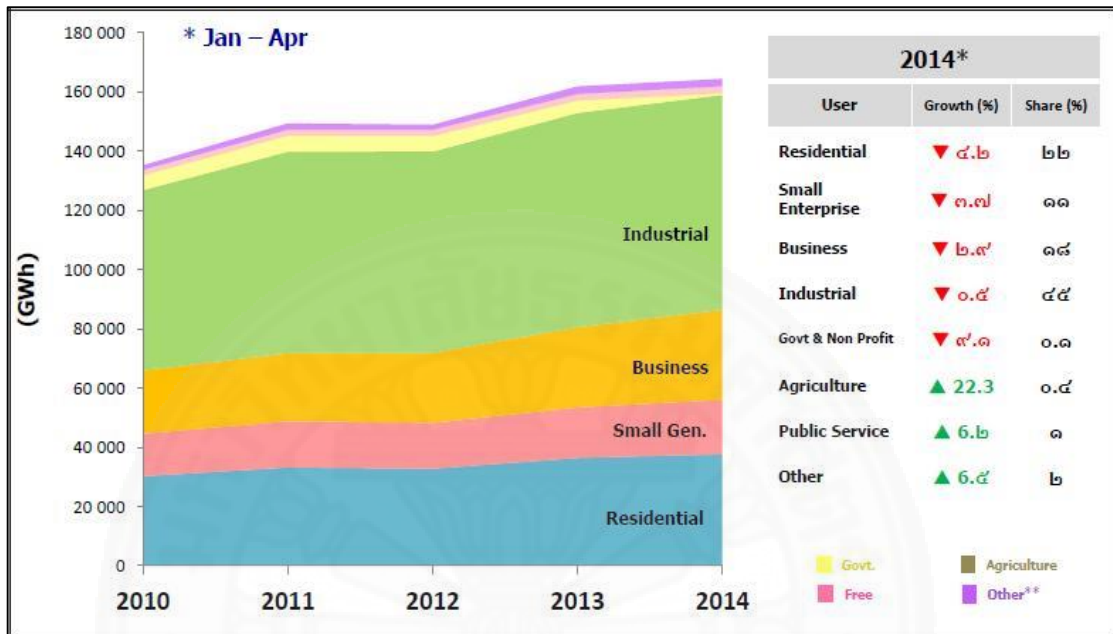


Figure 1.1 Electricity consumption during 2010 to 2014 by each sector (WONGLA, 2013)

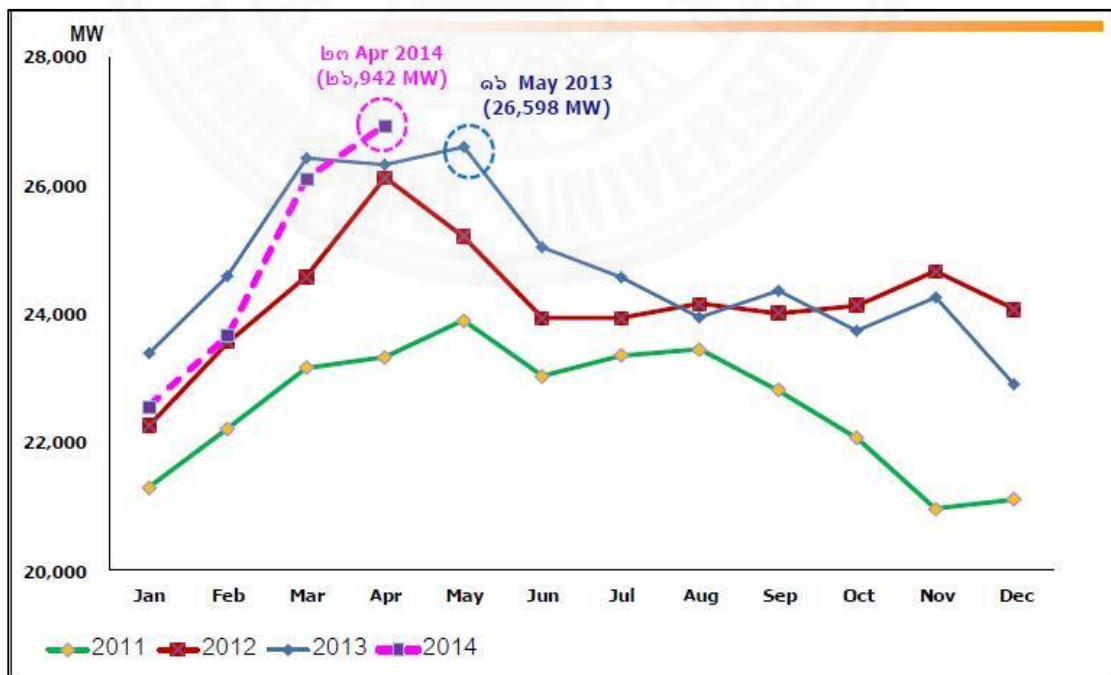


Figure 1.2 Electricity generation of all the months from 2011 January to 2014 April

Figure 1.2 explains electricity generation for each month from 2011 January to 2014 April. Electricity generations for all the months in 2012 are higher than all the months in 2011. Except August, October, and December, electricity generation is higher in 2013 than 2012. Peak electricity generation is on May for both 2011 and 2013. Peak generation for 2012 is on April. Peak electricity generation of 26,598 MW in 2013 has been already passed by peak electricity generation of 26,942 MW in 2014 April. Throughout the year, this variation is non-linear. Almost for all the months, demand is increased from one year to the next year, but unpredictable.

Even though this is highly non-linear and hard to predict, accurate forecasting of electricity demand helps to make important policies, plans, and for taking important decisions as discussed earlier. Therefore, these forecasting results directly affect to a country's economy. The following section explains the behavior of actual loads in Thailand. Patterns and some important factors which lead to alter the forecasting accuracy will be discussed as well.

1.2 Analysis of Actual Consumption Data

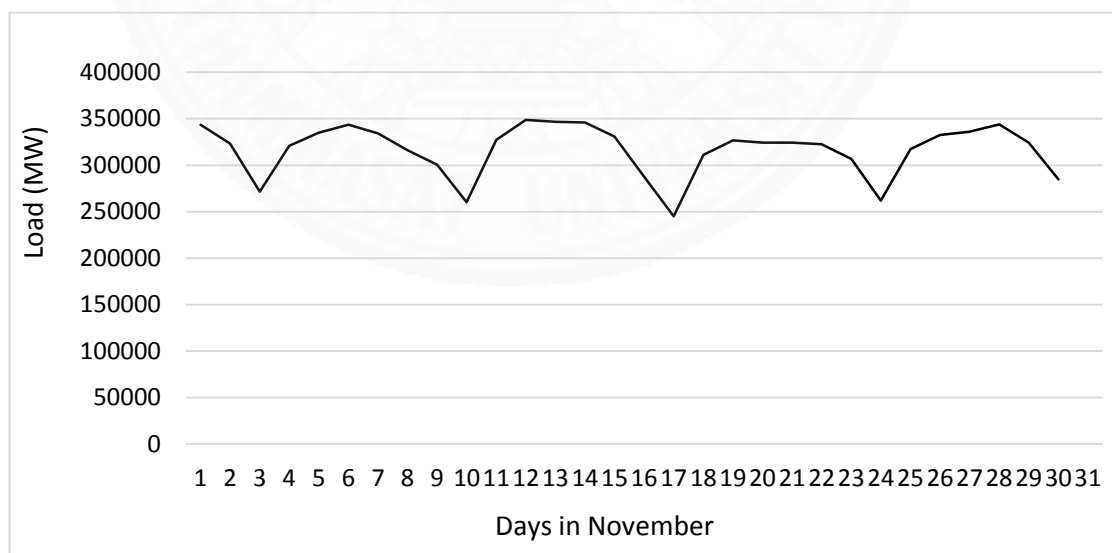


Figure 1.3 Total daily load in November 2013

When it comes to forecast the next day electricity load demand, understanding the daily and monthly patterns are important. Pattern variations over the years due to

the e.g. population and the economic growth are hardly affect the short-term load forecasting. Load consumption in Thailand can be categorized into several groups. The total daily load for each day of November, 2013 is plotted to get an idea about the weekly demand variation and given in Figure 1.3. Weekdays except Mondays have almost similar patterns throughout the day. Mondays' demand is low compared to the other weekdays' demand. Saturdays and bridging holidays give even lesser demand consumption and finally, Sundays and holidays have the lowest demand consumption. Therefore, it is very important to identify the patterns of each category and apply them while forecasting the similar days. The following figure shows how the regular electricity consumption varies from midnight to 11.30 pm on 10th Thursday, 2013.

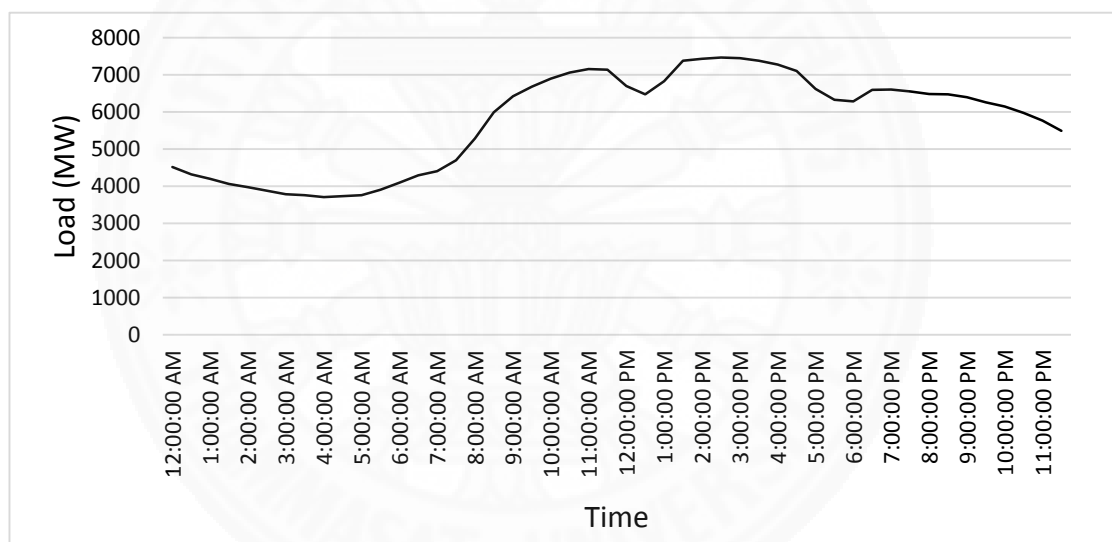


Figure 1.4 Electricity consumption pattern on 10th Thursday, 2013

The consumption starts rising from 8.00 a.m. and reaches to the peak around 12.00 p.m. Due to the lunch break, the demand suddenly drops down between 12.00 p.m. to 1.00 p.m. The consumption curve reaches to its second peak around 2.00 p.m. and stays steady till around 4.00 p.m. After 4.00 p.m., there is another drop of the consumption curve till 6.00 p.m. as people leave back to their homes after finishing their office hours. Due to the usage of electricity appliances, consumption curve reaches to its third peak between 7.00 p.m. and 9.00 p.m. With the start of the sleeping hours, consumption becomes very low and declines to flat.

However, these regular patterns can change for special cases. Holidays and the temperature are the main concerns. When we consider the total monthly electricity consumption, April and December have the lowest total monthly consumption as those months have a lot of holidays: April due to the Songkran festival and December due to the end year holidays. The total electricity consumption in May is higher compared to the other months. Higher temperature in May plays a vital role for this change. Therefore, we have to consider all the demand variations while using the historical data to forecast the next day electricity demand. The following section will give an introduction about electricity forecasting, how we can forecast with the existing methods, and how we can select a forecasting technique based on the time series pattern.

1.3 Electricity forecasting

Electricity forecasting can be divided into three main time horizons (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014): Short-Term Load Forecasting (STLF, few hours to few days), Medium-Term Load Forecasting (MTLF, few days to one year), and Long-Term Load Forecasting (LTLF, one year to few years). STLF uses in electric power dispatching and ensure optimal system utilization. MTLF uses for scheduling, renovation and installation of generating equipment, and distribution planning. Finally, LTLF uses for utility system planning, transmission and distribution planning, and revenue rate planning. Different sectors of a country may use different time horizons based on their requirements. E.g. government authors may need to forecast country's whole electricity demand for the next 20 years, electricity producers, suppliers, or retailers may want to forecast their daily electricity demand. STLF techniques wouldn't give accurate results for MTLF, or LTLF and vice versa. Therefore, different forecasting techniques are required to forecast different time horizons.

Following graphs will give a clear idea, why we need different techniques for different time horizons.

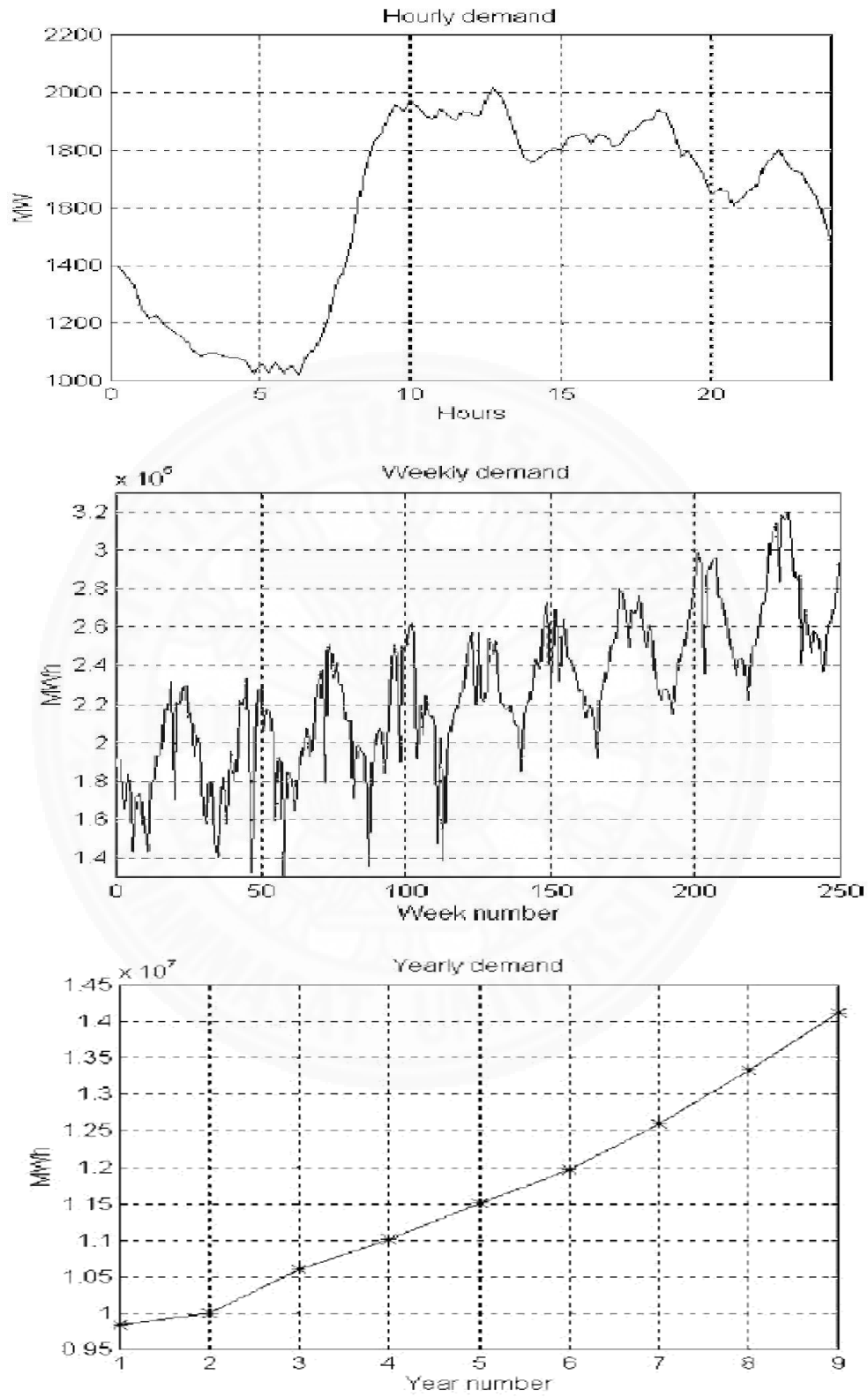


Figure 1.5 Demand curves for different time horizons (J. V. RINGWOOD, D. BOFELLI, F. T. MURRAY, 2001)

The first graph shows, how electricity consumption varies with hours in a day. It is completely nonlinear and there is no any specific pattern. For STLTF, studying of first graph's features is essential. Electricity consumption against weeks has been drawn in the second graph. It has a positive trend with seasonal variation while increasing the number of weeks. But its features are different from the first graph. For MTLF, careful consideration on second graph's features is important. Demand variation over the years is given by third graph. It has a positive trend with the number of years. Even though these three graphs are given for electricity consumption, curves have completely different patterns. Sometimes it is hard to say that these graphs are for same dependent variable. That is why we need different techniques to forecast electricity demand in different time horizons. Some techniques are better to forecast linear time series. Some techniques are better to forecast series which have seasonal variations. Therefore, first we have to think under which time horizon we are going to forecast and then we have to examine the features in it.

The other important thing about the electricity forecasting is external factors effect on electricity consumptions. A number of external factors can change the people's behaviors, so that these behaviors may increase or decrease the regular electricity consumption. Therefore, consideration about these external factors as important as selecting a forecasting technique to get accurate forecasting results. Temperature plays a vital role in electricity consumption (J. V. RINGWOOD, D. BOFELLI, F. T. MURRAY, 2001), (James W. Taylor, Roberto Buizza, 2002), (Henrique Steinherz Hippert, Carlos Eduardo Pedreira, Reinaldo Castro Souza, 2001), (Z. Ismail and F. Jamaluddin, 2008), (Mauro Castelli*, Leonardo Vanneschi, Matteo De Felice b, 2015), (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014). When the temperature is higher or lower than the regular days, electricity consumption is higher than regular days. (Henrique Steinherz Hippert, Carlos Eduardo Pedreira, Reinaldo Castro Souza, 2001). When the temperature is high, people use air conditioners in their homes and working places. It causes high electricity consumption and change the consumption curve. When the temperature is low, people use heating devices. This is also cause high electricity consumption and changed the regular consumption pattern.

Therefore, we can expect high electricity consumption during the winter and the summer. But there are some odd days suddenly temperature goes higher or lower than regular days. Predicting of electricity consumption is hard on those days. Therefore, we have to study and use the temperature variations from the past.

“Holidays” is another external factor that changes the regular electricity consumption. Most of the time it causes reducing the electricity consumption. On holidays, most of the industries are closed and people go to travel and do not stay in their homes. Because of that, usage of electric devices is very less and this causes consuming less amount of electricity during the holidays. (Mauro Castelli*, Leonardo Vanneschi, Matteo De Felice b, 2015), (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014), (James W. Taylor, Roberto Buizza, 2002), (Z. Ismail and F. Jamaluddin, 2008). Especial events (Mauro Castelli*, Leonardo Vanneschi, Matteo De Felice b, 2015), (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014), Seasonality (J. V. RINGWOOD, D. BOFELLI, F. T. MURRAY, 2001), (Henrique Steinherz Hippert, Carlos Eduardo Pedreira, Reinaldo Castro Souza, 2001), (Z. Ismail and F. Jamaluddin , 2008), Human behavior (Z. Ismail and F. Jamaluddin , 2008), Humidity (James W. Taylor, Roberto Buizza, 2002), (Henrique Steinherz Hippert, Carlos Eduardo Pedreira, Reinaldo Castro Souza, 2001), (Z. Ismail and F. Jamaluddin, 2008), Population (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014), (James W. Taylor, Roberto Buizza, 2002), (Z. Ismail and F. Jamaluddin , 2008), Economic status of the people (Mauro Castelli*, Leonardo Vanneschi, Matteo De Felice b, 2015), (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014), (Z. Ismail and F. Jamaluddin, 2008), and “Industrial growth” also cause changing regular electricity consumption.

Electricity forecasting accuracy is important for many related agencies. Both, under forecasting and over forecasting cause for additional cost and revenue reductions. Over forecasting causes producing or purchasing extra electricity. Under forecasting causes revenue reduction by not having of enough electricity to cover the customer demand. It may cause to get a bad image for the company (Z. Ismail, A.Yahya, K.A. Mahpol, 2009). Since this electricity forecasting accuracy is so important, use of good

forecasting technique along with discussed external factors can lead to get accurate forecasting results at the end.

1.4 Problem Statement

Electrical energy cannot be produced suddenly and storing electrical energy is also not possible. If the produced amount is higher than the required amount, it is an additional cost for producing extra electricity. If the produced amount is lower than the required amount, they have to purchase from other producers and it increases the cost. Therefore, the produced amount of electricity and the consumption amount by people should be equal in order to reduce the cost. This is important not only for producers, but also for distributors and retailers. Forecasting results of electricity demand can help to match the produced and required amount of electricity. Then they can plan it in advance how much they should produce.

1.5 Objective

There are many forecasting techniques available to overcome the above-mentioned problem. Many research are being conducted by researchers around the world. Considerably good forecasting results are taken for their experiments. But when they deal with huge amount of electricity consumptions, still the error cause to large additional costs. Therefore, the selected forecasting technique should have the ability of giving very accurate results. Different strategies have to be used while arranging historical data. Data from external related series have to be included to enhance the accuracy. Considering all these facts, the objective of this research is finding ways to further reduce the forecasting error for the Thailand case. This can be achieved by improving the existing forecasting techniques and also applying strategies while using the historical load demands and other available data such as temperature.

1.6 Significance of Study

The procedure for achieving the above objective and the other important components related to the procedure can be used by electricity producers, distributor, and retailers for forecasting the electricity demand based on their requirement. Accurate forecasting results help them on fuel estimation, fuel purchasing, electricity purchasing, renovation and installation of generating equipment, distribution planning, load switching, and to make many other decisions.

The technique and other supported components can be used by other researchers who are working in the same field. They could use this technique to build some other techniques and they can use these results as a benchmark to make decisions on their results. Since the electricity forecasting is just a one application under forecasting, the same approach use in this research can be used in other applications such as inventory forecasting, price forecasting, weather forecasting, sales forecasting and flood forecasting.

Chapter 2

Literature Review

2.1 Time Series

When observations, X , are gathered at particular time axis values t , these observations are called a time series. Time series can be separated into two groups: continuous time series and discrete time series (Peter J. Brockwell). In continuous time series, observations are gathered continuously for a given time period. In discrete time series, observations are gathered in equal time intervals and repeat it for a given time period. Following figure is an example for a time series. Observations have been gathered at very small time intervals. These observations have been connected by a continuous line.

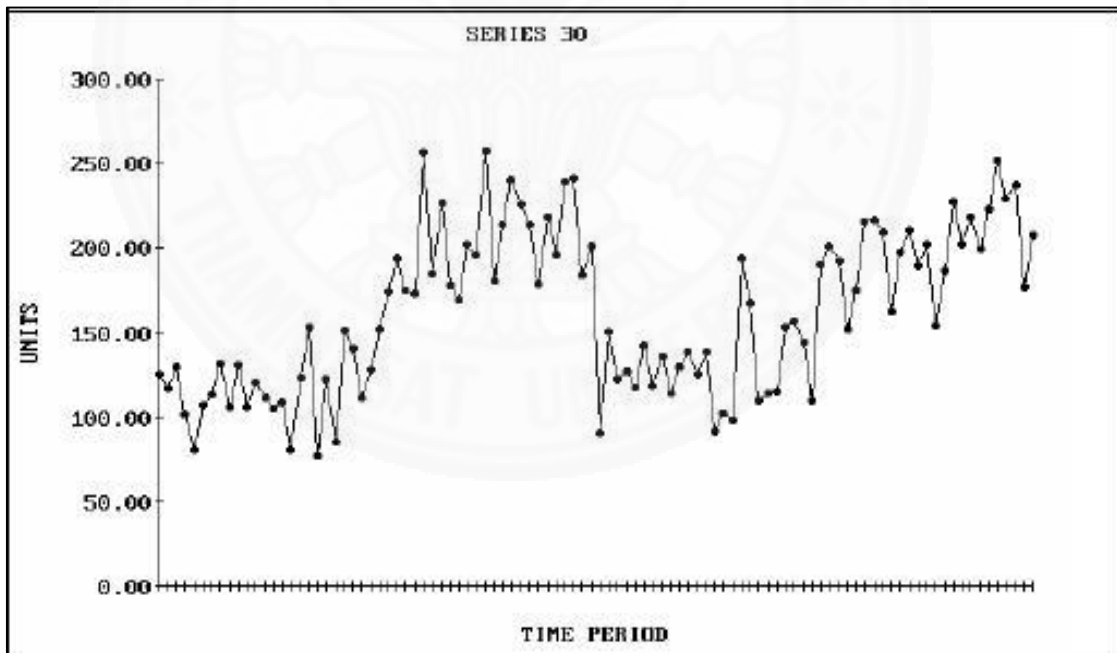


Figure 2.1 A sample time series

Therefore, discrete time series are converted to continuous time series when the time intervals which gathered the observations become very small. These discrete

observations are connected by a continuous line to get a clear idea about the series as given in Figure 2.1.

Time series have been created by combining different components: seasonal, trend, and remainder (noises). Figure 2.2 shows how to separate time series into seasonal, trend, and remainder. The trend can be positive or negative. Both positive and negative trends can appear in the same time series. Therefore, these trends are non-linear over the time. In terms of electricity consumption, both positive and negative trends appear in short time intervals (in daily consumption curves). As an example, the negative trend which gives just after starting of lunch hour becomes a positive trend just before finishing the lunch hour. A small positive trend can appear over long period in electricity consumption curves (in yearly consumption curves). This positive trend could be due to the population and industrial growth of a country. In the seasonal component, the same observation appears in equal time intervals. This time interval in seasonal series is called as a period. For electricity consumption, period may equal to 24 hours, when we have one week data. Period may equal to one week, when we have one month data. Likewise, period of seasonal series can be changed with the available amount of data. Because of this remainders or the noises in time series, forecasting outputs are always different than the actual values. It can be a positive or negative variation for the regular series.

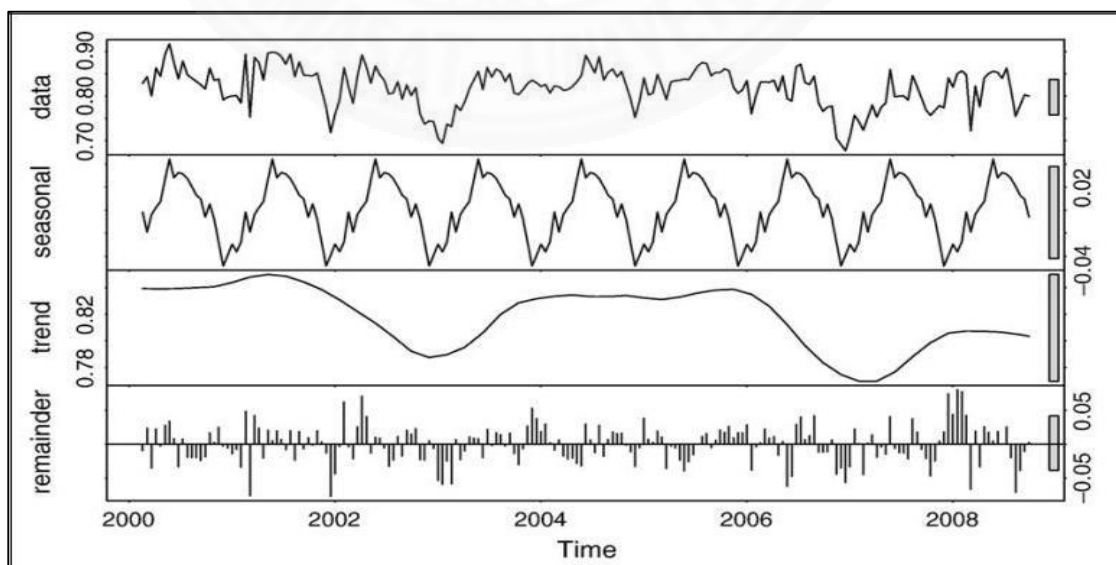


Figure 2.2 Separation of time series (Verbesselt, Jan, et al.)

These three components are combined to create time series. Sometimes only one component appears in time series. Sometimes two components or all three components could appear in time series. As a result, different time series are given when different components are combined.

When seasonal and trend components are not appearing in the time series, it calls a 'constant time series'. This series is parallel to the time axis with small positive and negative changes due to noise component. Therefore, constant series have the same mean over the time. Seasonal and trend components are combined to create the 'seasonal-trend' series. Noise component is always there with all the time series. Seasonal-trend series is look similar to the seasonal component. But values are increased if there is a positive trend and values are decreased if there is a negative trend. Trend component along with noise component creates 'trend series' while seasonal component along with noise component creates the 'seasonal series'.

Since different time series exhibit different characteristics, the same forecasting technique would not give the accurate results for different time series. When one forecasting technique predict trend series accurately, the same technique would give bad results for seasonal series forecasting. To overcome this problem, (Nahmais, 2005) brings different 'time series techniques' for different time series. For constant time series forecasting, Moving Average and Exponential Smoothing are suggested. These both techniques are specially created to forecast constant time series. For trend series forecasting, Regression Analysis and Double Exponential Smoothing using Holt's Method are suggested. These both techniques can track the characteristics of trend series and then forecast accurately. Winter's Method is used to forecast seasonal-trend series. Calculations in Winter's Method are quite complex than other time series techniques. Most of the time, this technique can be used to forecast seasonal series as well. But all these time series techniques deal only with the historical data of the time series. Less accurate forecasting result can be given while forecasting electricity demand since electricity consumption is affected by many external factors such as temperature, holidays, etc. Therefore, researchers are focusing on causal forecasting

techniques: Multiple Regression, Vector Auto Regression, Neural Networks, Fuzzy logic, etc. The main advantage of causal forecasting techniques is, related external series can be used to forecast the future. (James W. Taylor, Roberto Buizza, 2002), (R. Ramakrishna, Naveen Kumar Boiroju, M. Krishna Reddy, 2011), (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014) say that, Neural Network (NN) has the ability of recognizing and learning the non-linear patterns in time series. Therefore, they suggest NN to forecast all these different time series.

2.2 Related works and Models

In almost all electricity forecasting research, the main objective is to get accurate results by introducing new techniques with data arrangement strategies. Their reason for conducting this research is, both over and under forecasting cause reducing revenue by increasing additional costs. Therefore, researchers are trying to conduct experiments on electricity forecasting with the purpose of reducing the error between actual and the forecasted data.

Taylor, 2010 mention three methods in his research which are commonly used in electricity forecasting: Exponential Smoothing, Holt-Winters Exponential Smoothing, and Double Seasonal ARMA. But only causal forecasting techniques can work with related external series such as temperature. Electricity consumption mostly vary with the weather variations. History of the weather and weather forecasting results can include in forecasting techniques to get accurate electricity forecasting outputs. Disadvantage when we use these external series is, those are costly to get and hard to find. Therefore, he suggests univariate methods for electricity forecasting which can perform better without external series. His suggested triple seasonal method is demonstrated with Great British and French load data. Half an hour, five years data from 2001 to 2005 is used to train the technique. Then one year (2006) data is used check the accuracy of the forecasted results using the suggested technique.

(Cheng-Ting Lin, Li-Der Chou, Yi-Ming Chen, Li-Ming Tseng, 2014) discuss two approaches for short-term electricity forecasting: Statistic based and Artificial Intelligence based. With their results, they conclude that Artificial Intelligence (AI) based approaches perform better than statistic based approaches since AI based approaches have the ability of recognizing and learning the relationship between the main series and non-linear external series. (Mauro Castelli*, Leonardo Vanneschi, Matteo De Felice b, 2015) also bring the same conclusion that statistic based approaches cannot recognize non-linear patterns and cannot find the relationships with external series. Therefore, they also suggest to use AI for electricity forecasting.

Since AI based approaches need a lot of historical data to train the technique and this may lead to the overfitting problem which gives wrong outputs, (Cheng-Ting Lin, Li-Der Chou, Yi-Ming Chen, Li-Ming Tseng, 2014) bring a new integrated technique for short-term load forecasting: Hybrid Economic Indices for Short Term Load Forecasting (HEI-STLF). They use two years of data to train their technique. They have separated their data in to two sections as working day demands and non-working day demand. Then one month data is used to check the accuracy of results which they obtained from their technique. Mean Absolute Percentage Error (MAPE) had been used to analyze the output.

As a solution for disadvantages in AI such as over-fitting, required a lot of historical data in the training phase, difficulty in parameter selection, and etc. (A.S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, B. VenkateshDepartment, 2015) suggest another integrated technique which is based on Neural Network: Bagging technique. New England historical load data from 2004 to 2006 is used to demonstrate the suggested technique. Artificial neural network is used to recognizing and learning the pattern appearing in the historical data. Bagging is used to improve the accuracy and the stability of the output. Simply, this helps to avoid the overfitting problem. MAPE is calculated between real and forecasted data. These results are compared with results which were obtained using only the NN.

These days, many researches use NN based forecasting techniques for short-term load forecasting. They believe that it helps to get the advantage of NN and avoid the disadvantage by other combined methods. (Krzysztof Gajowniczek, Tomasz Ząbkowski*, 2014) use Multi-Layer Perceptron (MLP) and Support Vector Machines (SVM) which are based on NN for their experiments in electricity forecasting. Neural Network and the Box – Jenkins methods for electricity forecasting are compared by (R. Ramakrishna, Naveen Kumar Boiroju, M. Krishna Reddy, 2011). Data from a transmission company in Hyderabad, India is used in this research. Results of their research show that NN perform better in electricity demand forecasting than Box–Jenkins method does. They also mention, NN’s recognizing and learning ability in the training phase. (J. V. RINGWOOD, D. BOFELLI, F. T. MURRAY, 2001) also use Artificial Neural Network (ANN) for forecasting Ireland electricity demand. They find that ANN is better for forecasting all three time horizons: short-term, Medium, and Long-term. Finally, (Henrique Steinherz Hippert, Carlos Eduardo Pedreira, Reinaldo Castro Souza, 2001) compare all these NN based techniques. They discuss advantages and disadvantages of each technique, where, when, and how we can use these techniques.

This shows, integrated techniques are included one of major techniques such as Neural Network, Regression. Therefore, researchers expect to keep the advantages of that main technique while removing the disadvantages by combining with another technique. This is proved by the results. When they compare the results of original techniques with the integrated techniques, most of the time, results by the integrated techniques are better than the original techniques.

Other integrated techniques such as Moving Average and Combined procedures with Weighted Hybrid model (MA-C-WH) (Suling Zhu, 2011), self-similarity theory with fractal interpolation theory (Zhai, 2015) and individual techniques such as Rule Based Approach (Z. Ismail, A.Yahya, K.A. Mahpol, 2009), Time Series Regression Model (Z. Ismail and F. Jamaluddin, 2008) also have good results for their own experiments in electricity forecasting.

When we compare the above techniques, it is important to consider about the amount of training data researchers used to demonstrate their techniques, external factors they used with their techniques, and the interpreted way of results by their techniques. With these factors, results can be varied for each technique. It is hard to say which technique is the best when researchers have done their research under different conditions. Some researchers had used temperature as an external input for their techniques, some had used only the historical data with their techniques. Therefore, we cannot conclude which technique is the best when the conditions are different.

Researchers who used temperature as an external input may have better results than who didn't use. Some of researchers had used a large amount of data (4 – 7 years) to train their techniques, but some researchers have used less amount of data (few months – 1 year) to train their techniques. This can vary the output even for the same technique. Therefore, consideration about the amount of data they used to train their techniques is important before deciding which technique is the best. The other important thing is the interpreting way of results. This helps reader to get an idea that not only how good the proposed technique is, but also the effort of the researcher on the research and how good his data arrangement is. The most common ways of interpreting results are Mean Absolute Percentage Error (MAPE), Mean Absolute Deviation (MSE), Root Mean Squared Error (RMSE), and etc. Researchers can use one of these methods to calculate the error between real and forecasted data from their techniques and interpret it numerically. Some of selected research with their pros and cons are summarized in Table 2.1.

Finally, all of these forecasting techniques are summarized in the following tree diagram (Figure 2.3). Forecasting techniques can be divided into two groups: Subjective forecasting techniques and Objective forecasting techniques (Nahmais, 2005). Subjective forecasting techniques simply analyze the people's opinions. Therefore, these techniques are not important and not used in electricity demand forecasting. Some examples for subjective forecasting techniques are, Sales force composites, Delphi method, Jury of executive opinion, and Customer surveys. All other

techniques discussed above belong to the objective forecasting techniques. Objective forecasting techniques analyze the numerical values and able to give numerical results. Therefore, readers can easily get an idea about the technique or about the research. Basically, these techniques use historical data in the training phase. Objective forecasting techniques can be divided into two groups: Time series techniques and Causal techniques. Time series techniques analyze the history of the same data set. Therefore, they forecast the future based only on the historical data. But more related external series can be included in Causal techniques. Therefore, these techniques can identify how the main series varies with other external series and predict accurately. Some examples for time series techniques are, Moving Average, Naïve method, Exponential Smoothing, Simple Regression, and etc. Some examples for causal forecasting techniques are, Multiple Regression, Dynamic Regression, Artificial Neural Network, and etc.

Table 2.1 List of research/techniques on electricity demand forecasting

No.	Title	Author	Country	Year	Technique(s)	Comments (*), Pros (+), and cons (-)
01	Forecasting Electricity Demand on Short, Medium and Long Time Scales Using Neural Networks	J. V. RINGWOOD, D. BOFELLI, F. T. MURRAY	Netherlands	2001	Artificial Neural Networks (ANNs)	(*) Results are better with its own experiments (-) Difficulty in parameter selections
02	Forecasting daily electricity load using neural networks	R. Ramakrishna, Naveen Kumar Boiroju, M. Krishna Reddy	India	2011	Neural Networks and Box-Jenkins methods	(*) To forecast daily electricity load demand, NN is better than Box-Jenkins method
03	Neural Network Load Forecasting with Weather Ensemble Predictions	James W. Taylor, Roberto Buizza	UK	2002	Neural Networks (NN) with weather ensemble predictions	(*) Perform better than traditional methods (+) Has the ability to model an unspecified non-linear relationship between load and weather variables.
04	Time series regression model for forecasting Malaysian electricity load demand	Z. Ismail, F. Jamaluddin, F. Jamaludin	Malaysia	2008	Multiple regression model (time series regression model with auto regression error)	(*) Input holiday results are important for load forecasting (+) MAPE with holiday variables is 1.72% for their experiment
05	Forecasting Peak Load Electricity Demand Using Statistics and Rule Based Approach	Z. Ismail, A.Yahya, K.A. Mahpol	Malaysia	2009	Rule-based method	(*) Results are more accurate than the standard moving average and exponential smoothing method (+) Can be applied forecaster

						expertise and domain knowledge
06	A new method for short-term load interpretation and wavelet analysis	Ming-Yue Zhai	China	2015	Combined with self-similarity theory and fractal interpolation	(+) Easy to use, less-sample demands, high-precision
07	Forecasting short-term electricity consumption using a semantics-based genetic programming framework: The South Italy case	Mauro Castelli, Leonardo Vanneschi, Matteo De Felice	Italy	2015	Genetic programming integrate with sematic methods	(*) Performs better than other traditional techniques (+) Can capture the non-linearity in demand signals under the market conditions
08	Short term electricity forecasting using individual smart meter data	Krzysztof Gajowniczek, Tomasz Ząbkowski	Poland	2014	Neural Networks and Support Vector Machines	(*) Both methods are good in electricity forecasting with their data arrangements
09	A seasonal hybrid procedure for electricity demand forecasting in China	Suling Zhu, Jianzhou Wang, Weigang Zhao, Jujie Wang	China	2011	Moving Average and Combine procedures with Weighted Hybrid model (MA-C-WH)	(*) Results are compared with results from SARIMA model: shows that proposed method is better (+) Can capture the trends and seasonal adjustments
10	A hybrid economic indices based short-term load forecasting system	Cheng-Ting Lin, Li-Der Chou, Yi-Ming Chen, Li-Ming Tseng	Taiwan	2014	Hybrid Economic Indices based Short-Term Load Forecasting (HEI-STLF)	(*) Compared to the traditional SVR load forecasting, suggested method perform better in forecasting (+) Can reduce the impact of dynamic of economy

11	Improved short-term load forecasting using bagged neural networks	A.S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, B. Venkatesh	Canada	2015	Bagged Neural Networks (BNNs).	(*) BNNs perform better in terms of lowering mean absolute percentage error for their experiments (+) Bagging process reduces variation range of errors and estimation errors
12	Forecasting Electricity Demand in Thailand with an Artificial Neural Network Approach	Karin Kandananond	Thailand	2011	Autoregressive Integrated Moving Average (ARIMA), Artificial Neural Network (ANN), Multiple Linear Regression (MLR)	(*) MAPE with ANN - 0.996% (*) MAPE with ARIMA - 2.80981% (*) MAPE with MLR - 3.2604527% (+) ARIMA needs information only for one variable (+) MLR takes less calculation time than other two (-) ANN computation time is much higher than other two (-) ANN need many training epochs (-) Difficulties in ANN parameter selections
13	An evaluation of Bayesian techniques for controlling model complexity and selecting inputs in a	Henrique S. Hippert, James W. Taylor	UK	2010	Artificial Neural Network model with Bayesian techniques	(*) Technique is compared with the cross-validation (+) Helps to define the model complexity (+) Helps to select input

	neural network for short-term load forecasting					parameters (+) Helps to avoid overfitting
14	Electrical consumption forecasting in hospital facilities: An application case	A. Bagnasco, F. Fresi, M. Saviozzi, F. Silvestro, A. Vinci	Italy	2015	Multi-Layer Perceptron ANN, based on a back propagation training algorithm	(*) Shows better results for their case study (*) Suggest for other types of time series forecasting
15	A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids	Nian Liu, Qingfeng Tang, Jianhua Zhang, Wei Fan, Jie Liu	China	2014	Hybrid model with Empirical Mode Decomposition (EMD), Extended Kalman Filter (EKF), Extreme Learning Machine with Kernel (KELM) and Particle Swarm Optimization (PSO)	(*) Gives acceptable results for their case study
16	A novel hybrid model for bi-objective short-term electric load forecasting	JinXing Che	China	2014	Hybrid model: multiple linear regression with support vector regression	(*) Proposed method outperform MLR, OTS-SVR, ANN-APSO and OTSSVR-ARIMA
17	Monthly electric demand forecasting	Miguel A. Jaramillo-Morán,	Spain	2013	Multilayer Perceptron NN	(*) Multilayer Perceptron is able to perform both filtering and

	with neural filters	Eva González-Romera, Diego Carmona-Fernández				forecasting (* Capable of predicting monthly load demand for the next 12 months (* Shows the capability of NN in signal processing
18	A combined model based on data pre-analysis and weight coefficients optimization for electrical load forecasting	Liye Xiao, Jianzhou Wang, Ru Hou, Jie Wu	China	2015	Combined NN based techniques: BPNN, RBFNN, GRNN, and GA-BPNN	(* Combined method is better than individual techniques based on their results (* Can use of advantages of each individual technique (+) Data pre-analysis is used to reduce the interferences from the data
19	A novel random fuzzy neural networks for tackling uncertainties of electric load forecasting	Chin Wang Lou, Ming Chui Dong	China	2015	Novel integrated Technique: Random Fuzzy NN (RFNN)	(+) Remedy for uncertainties of inputs and network parameters (+) Eliminate data pre-processing (+) Can tackle more volatility on historical data (-) Computing time is high
20	Electricity demand forecasting over Italy: Potential benefits using numerical weather prediction models	Matteo De Felice, Andrea Alessandri, Paolo M. Ruti	Italy	2013	Time series models (ARIMA/ARIMAX) used with numerical weather prediction (NWP)	(* Results are compared with results from naive predictor (* Use of weather data improve the forecasting results
21	Pattern similarity-based methods for	Grzegorz Dudek	Poland	2015	Kernel estimation-based model,	(* Results are competitive with other univariate techniques

	short-term load forecasting				nearest neighbor estimation-based models and pattern clustering-based models	(+) Simple to use (+) Small number of parameter are used (+) Outputs can be increased without complicating the structure
22	A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm	Hong-ze Li, Sen Guo, Chun-jie Li, Jing-qi Sun	China	2013	Model combining fruit fly optimization algorithm (FOA) and generalized regression neural network (GRNN)	(*) Outperform, GRNN model with default parameter, least squares support vector machine with simulated annealing algorithm (SALSSVM), GRNN model with particle swarm optimization (PSOGRNN) (+) Training time is shorter
23	Next-day MV/LV substation load forecaster using time series method	Ni Ding, Yvon Bésanger, Frédéric Wurtz	France	2015	Dummy variable regression models for separately track the trend, seasonal and random error	(*) Results are compared with the naive model (*) Shows good results for their own experiment
24	Short term and medium term power distribution load forecasting by neural networks	T. Yalcinoz , U. Eminoglu	Turkey	2005	Neural Network (NN)	(*) Shows good results for their own experiments (*) Suggests to use in other forecasting requirements
25	Forecasting next-day electricity demand and price using nonparametric functional methods	Juan M. Vilar, Ricardo Cao, Germán Aneiros	Spain	2012	Nonparametric regression techniques with functional explanatory data and	(*) Results are compared with results from naïve method and with ARIMA (+) Outperform naïve and ARIMA techniques

					a semi-functional partial linear model	(+) Flexible for new data (-) Complex than other techniques
26	Performance Evaluation of Different Optimization Algorithms for Power Demand Forecasting Applications in a Smart Grid Environment	Ashraf Ul Haque, Paras Mandal, Julian Meng, Ricardo L. Pineda	USA	2012	Hybrid intelligent algorithm: combination of wavelet transform (WT) and fuzzy ARTMAP (FA) network, that's optimized using genetic algorithm (GA), particle swarm optimization (PSO), and firefly (FF) algorithm	(*) Results are compared with the results from BPNN, RBFNN, ANFIS methods (*) FF is the best out of three optimization methods
27	Hybrid filter-wrapper feature selection for short-term load forecasting	Zhongyi Hua, YukunBao, TaoXiong, RaymondChiong	China	2015	Support vector regression with hybrid filter-wrapper approach	(*) Shows good results for the designed experiment and outperform other well-known methods (+) Can identify few important features with high forecasting Accuracy
28	A hybrid Genetic Radial Basis Function Network with Fuzzy Corrector for Short Term Load Forecasting	W. T. Ghareeb, E. F. El Saadany	Canada	2013	Genetic Algorithm optimized Radial Basis Function network (GA-RBF) with a fuzzy corrector	(*) Results are compared with results from four methods: multi-layer feed forward neural network, the RBF network, the adaptive neuro-fuzzy inference System and the genetic programming (*) Outperform the other

						techniques
29	Support Vector Machine Optimized with Genetic Algorithm for Short-term Load Forecasting	Lihong Ma, Shugong Zhou, Ming Lin	China	2008	Support Vector Machines (SVM) with Genetic Algorithm to optimize the parameters (GA-SVM)	(+) Has greater improvement in both accuracy and velocity of convergence for SVM (+) The model is practical and effective and provides an alternative for load forecasting
30	Short Term Load Forecasting using a Robust Novel Wilcoxon Neural Network	Sanjib Mishra, Sarat Kumar Patra	India	-	Wilcoxon neural network (WNN) with Wilcoxon norm cost function and Multilayer perceptron neural network (MLPNN) with least mean square (LMS) cost function	(*) MAPE is used to calculate the errors (*) MLP NN provides better performance than WNN (+) Can train with dew data sets
31	Short Term Load Forecasting using a Neural Network trained by A Hybrid Artificial Immune System	Sanjib Mishra, Sarat Kumar Patra	India	2008	Artificial immune system (AIS) use with back-propagation to train an NN	(+) Lesser historical data are required (+) Results are better than pure back-propagation training, GA and PSO
32	Short Term Load Forecasting using Genetically Optimized Radial Basis Function Neural	Navneet Kumar Singh, Asheesh Kumar Singh, Manoj Tripathy	Australia	2014	Radial Basis Function Neural Network (RBFNN) is optimized using Genetic Algorithm	(*) Results are compared with the results from Feed Forward Neural Network (FFNN) (*) Results of proposed method are significantly better

	Network				(GA)	(+) Reduction in input parameters
33	Short Term Load Forecasting using Neural Network trained with Genetic Algorithm & Particle Swarm Optimization	Sanjib Mishra, Sarat Kumar Patra	India	2008	An NN is trained with BP, GA and PSO	(+) The GA training gives better accuracy than BP training (-) GA takes much longer time (+) PSO is faster than both BP and GA (-) PSO slight compromise in accuracy
34	A Hybrid Particle Swarm Optimization Neural Network Approach for Short Term Load Forecasting	WANG Xuan, LV Jiake, WEI Chaofu, XIE Deti	China	2008	Neural network with particle swarm optimization training algorithm	(*) ARMA model and NN with PB is used to compare the results (+) Results are better with given experiment (-) PSO parameters have to be selected by trial and error method
35	Short term load forecasting in Mauritius using neural network	C. Bhurtun, I. Jahmeerbacus, C. Jeewoath	Mauritius	2011	Artificial Neural Network (ANN) with back-propagation	(*) Better to forecast next hour, next day, and next week demand (+) Able to determine the non-linear relationships of historical data.
36	Back Propagation Neural Network for Short-term Electricity Load Forecasting with Weather Features	Yong Wang, Dawu Gu, Jianping Xu, Jing Li	China	2009	Back propagation neural network with weather feature	(*) Shows good results for their own experiments (+) Can use for long-term electricity forecasting with required external information
37	Forecasting day-ahead electricity load using a	A.E.Clements, A.S.Hurn, Z.Li	Australia	2016	Multiple equation time-series model	(*) MAPE is used to evaluate the outputs (*) Has the ability of outperform well-known forecasting techniques

	multiple equation time series approach					(+) Applicable for any load forecasting problem (+) Flexibility of handling with a lot of data
38	Electricity Load Forecasting Based on Support Vector Machines and Simulated Annealing Particle Swarm Optimization Algorithm	Jingmin Wang, Yamin Zhou and Xiaoyu Chen	China	2007	SVM model with Stimulated Annealing Particle Swarm Optimization Algorithm (SAPSO) for SVM parameter selection	(*) Provides a promising alternative for forecasting electricity load (*) Outperforms BPNN and PSO algorithm for training SVM (+) Reduce the operating time (+) Other parameter selection techniques can be combined with this technique
39	A new method for short-term electricity load forecasting	Jing-Min Wang, Li-Ping Wang	China	2008	Integrate the support vector machines (SVMs) forecasting technique and rough sets (RSs) with reduced attributes using evolutionary algorithms (EAs)	(*) The technique is compared with the traditional SVM and SVMs combine with simulated annealing algorithms (SVMSA) (+) Reduce the input variables (+) High prediction accuracy (+) Better speed the convergence (+) Require less computational effort
40	Wavelet-GA-ANN Based Hybrid Model for Accurate Prediction of Short-Term Load Forecast	Nidul Sinha, Loi Lei Lai, Palash Kumar Ghosh, Yingnan Ma	UK	-	Integrate of wavelet transforms, floating point GA and artificial neural networks	(*) Results are compared with results from Radial Basis Function (RBF) (+) Can capture the global trend and hidden templates in loads (+) Can use for multiple steps ahead forecasting

41	Forecasting Short-term Electricity Demand in Residential Sector Based on Support Vector Regression and Fuzzy-rough Feature Selection with Particle Swarm Optimization	Hyojoo Son, Changwan Kim	Korea	2015	Support vector regression and fuzzy-rough feature selection with particle swarm optimization algorithms	(*) Results are evaluated using MAPE, MAE, RMSE, MBE, and UPA values (*) Results are compared with the results from artificial neural network, auto-regressive integrated moving average, multiple linear regression models (+) Automatically determines appropriate and necessary variables
42	Power System Short - term Load Forecasting Based on Improved Support Vector Machines	Wenbin Ma	China	2008	Support vector machines (SVM) with different training algorithms	(+) Has a high prediction accuracy and faster computing speed (-) No specific way of selecting parameters (-) Required large number of data
43	An ensemble approach for short-term load forecasting by extreme learning machine	Song Li, Lalit Goel, Peng Wang	Singapore	2016	Combine wavelet transform, extreme learning machine (ELM) and partial least squares regression	(*) Numerical results show great improvements (+) Solve the problem of overtraining (+) Solve the problem of wavelet parameter determination
44	Application of an Adaptive Network-based Fuzzy Inference System Using Genetic Algorithm for Short Term Load Forecasting	Zhang Honghui, Li Yongqiang	China	2012	Genetic algorithm (GA) optimized Adaptive Network-based Fuzzy Inference System (ANFIS)	(*) Better results for their own experiments (+) Can include many relevant series for electricity consumption (+) Require a shorter training time than ANN

45	A hybrid approach of neural networks and grey modeling for adaptive electricity load forecasting	Cheng-Chin Chiang, Ming-Che Ho, Jen-An Chen	Taiwan	2006	Neural networks combine with grey modeling	(*) Outperforms the individual ones and the statistical autoregressive methods (+) Helps to choose important variables among many input variables (+) Reduces the interference of irrelevant inputs
46	Short term forecasting using neural network approach	Dipti Srinivasan, A. C. Liew, John S. P. Chen	Singapore	1991	Neural network with back propagation	(*) Results are compared with results from conventional smoothing methods and stochastic methods (*) Gives better results for own experiments
47	Core vector regression with particle swarm optimization algorithm in short term load forecasting	Yuancheng Li, Kewen Liu	China	2010	Integrate Core Vector Regression (CVR) with particle swarm optimization (PSO)	(*) Compare the results with results from Support Vector Regression (SVR) (+) Can be used with large scale data sets (+) Training time is much faster (+) Do not need to manually select the input parameters, therefore reduce the workload
48	Artificial Neural Network based Short Term Load Forecasting for Restructured	Mohan Akole, Barjeev Tyagi	India	2009	Multi-layer feed forward (FF) neural network trained with Back propagation (BP) algorithm	(*) MAPE, MSE and percentage error are used to calculate the errors (*) Results are compared with the results from Multiple Regression

	Power System					(MR) forecasting model
49	Mathematical modelling and short-term forecasting of electricity consumption of the power system, with due account of air temperature and natural illumination, based on support vector machine and particle swarm	Nadtoka I.I, Al-Zihery Balasim M	Iraq	2015	Support vector machine (SVM) with particle swarm optimization (PSO)	(*) MAPE is used to evaluate the results (*) Prediction error of summer and winter is less than of spring and autumn (*) Overall better results for own experiments
50	Support Vector Machines with PSO Algorithm for Short-Term Load Forecasting	Changyin Sun, Dengcai Gong	China	2006	Support vector machine (SVM) with particle swarm optimization (PSO) to determine the free parameters of SVM	(*) Provides a promising alternative for forecasting electricity load (*) Outperforms the BPNN and FI Models (-) Longer iteration time
51	Short Term Electrical Load Forecasting Using Back Propagation Neural Networks	S. Surender Reddy, James A. Momoh	-	2014	Artificial neural networks (ANN) with Back Propagation Algorithm (BPA)	(*) Shows good results for the given experiment (+) Has the ability of including many external series (+) Has high forecasting accuracy.
52	Substation short term load forecasting using neural network with genetic algorithm	TAYATI Worawit, CHANKAIPOL Wanchai	Thailand	2002	Neural network (NN) with a generic algorithm (GA)	(*) Technique is compared with the BPNN (*) Error is 0.77 % lower than BPNN

						(+) NN weights and bias are easily optimized
53	Short-term electricity load forecasting of buildings in micro-grids	Hamed Chitsaz, Hamid Shaker, Hamidreza Zareipour, David Wood, Nima Amjady	Iran	2015	Self-Recurrent Wavelet Neural Network (SRWNN) is trained by Levenberg–Marquardt (LM) learning algorithm	(*) Shows good results with given case studies (+) Better captures nonlinear complexities of volatile time series
54	Short –term Electricity Load Forecasting Based on SAPSO-ANN Algorithm	Jingmin Wang, Yamin Zhou	China	2008	Artificial neural network is trained with Stimulated Annealing Particle Swarm Optimization Algorithm (SAPSO)	(*) The technique is compared with PSO-BP, BPNN and PSO-SVM models (*) Shows lower prediction error than other methods (+) Enhances the accuracy (+) Improve the convergence ability (+) Reduce operation time
55	A Short-term Load Forecasting Approach Based on Support Vector Machine with Adaptive Particle Swarm Optimization Algorithm	Huang Yue, Li Dan, Gao Liqun, Wang Hongyuan	China	2009	Support vector machine (SVM) is combined with adaptive particle swarm optimization (APSO) algorithm	(*) Proposed technique outperformed SVM, BPNN and regression model (+) APSO easily and successfully identify the optimal values of parameters of SVM
56	An Adaptive BP-Network Approach to Short Term Load	Liang Haifeng, Li Gengyin, Zhou Ming	Hong Kong	2004	BP-network is combined with Genetic Algorithm	(*) Gives the percentage error 3% (+) The NN structure is made by GA

	Forecasting				(GA)	(+) Can be used with other forecasting applications
57	Short-term forecasting of the Abu Dhabi electricity load using multiple weather variables	Luiz Friedrich, Afshin Afshari	United Arab Emirates	2015	Transfer Function (TF) model	(*) Accuracy of the model is compared with the Autoregressive Integrated Moving Average (ARIMA) model and to an Artificial Neural Network (ANN) model (*) Gives better results for one and two day forecasting
58	Grouping Model Application on Artificial Neural Networks for Short-term Load Forecasting	Shunhua Zhang, Jingjing Lian, Zhiying Zhao, Huijun Xu	China	2008	Artificial neural networks with back propagation	(*) Matlab neural network toolbox has been used (*) Grouping data into three sections: workday data, weekend data and festival data helps improving forecasting accuracy
59	BP Neural Network Optimized with PSO Algorithm for Daily Load Forecasting	ZhaNg Caiqing, Lin Ming, Tang Mingyang	China	2008	BP neural network is optimized by particle swarm optimization (PSO)	(+) Both accuracy and velocity of convergence for BP neural network has a great improvement (+) Model is practical, effective, a better alternative for STLF techniques

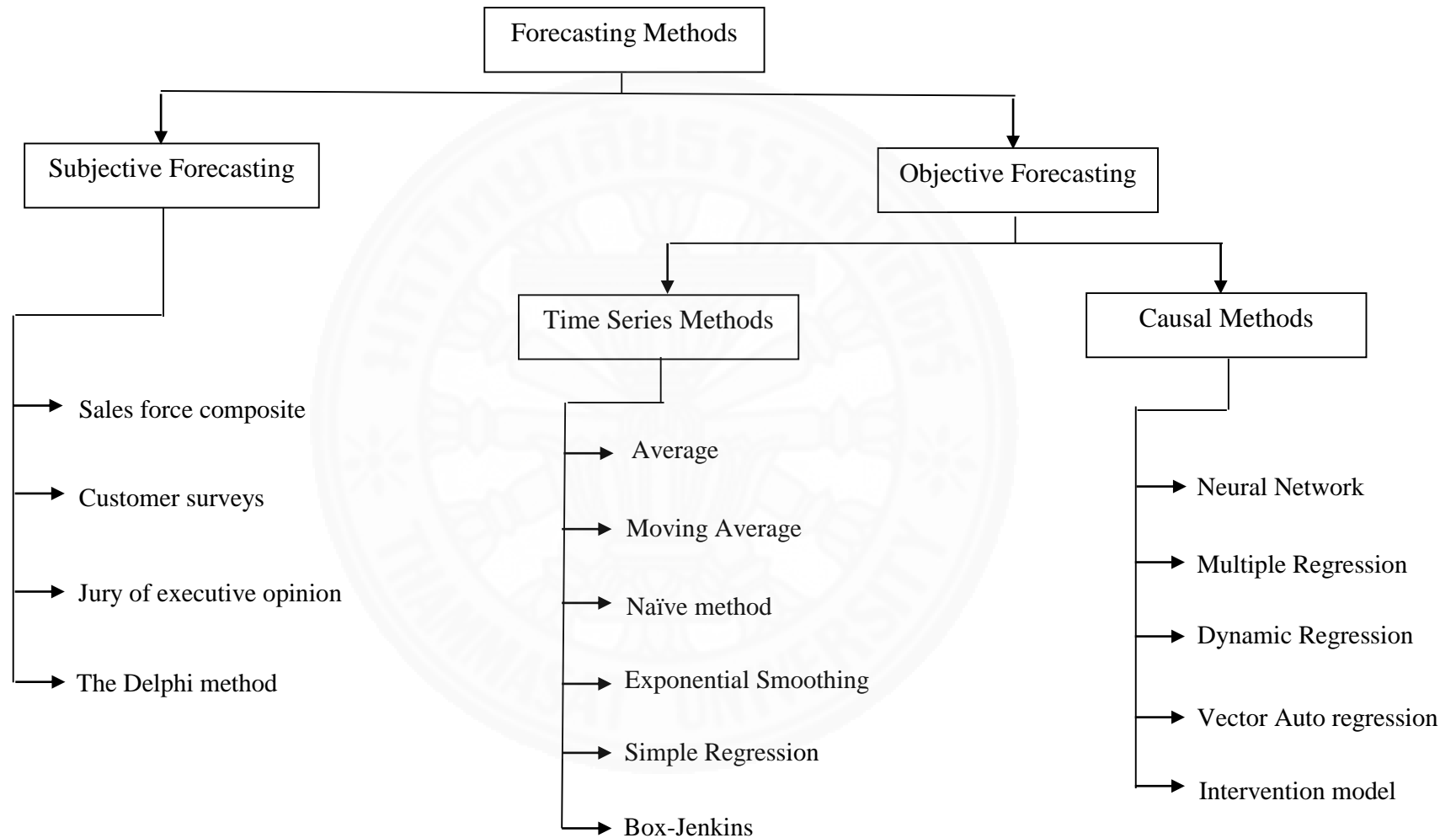


Figure 2.3 Categorization of Forecasting Techniques

Chapter 3

Method and Procedure

3.1 Neural Network (NN)

Many researchers use NN for electricity load demand forecasting since its ability of recognizing and learning inherent patterns in historical data. As a causal forecasting technique, many external series can be included in NN such as temperature and seasonal variations. The basic structure of NNs consist with input layer, output layer, and hidden layers. Input nodes are given to represent the external inputs to the network. Output nodes represents the output from the NNs or the NNs' results. Hidden layers consist with neurons and all the layers are connected by their specific weights from one neuron to another. The following figure is given to get an idea about the NN structure.

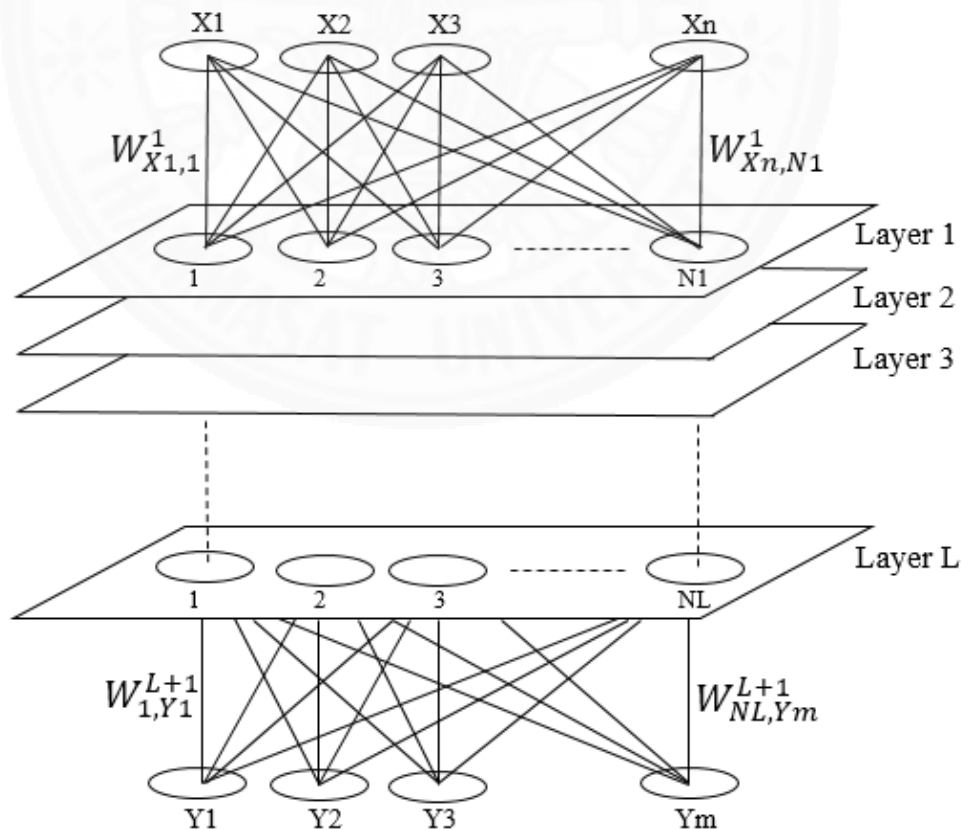


Figure 3.1 The basic structure of NNs

Layer 1, 2, and up to L are the hidden layers in the given structure. The structure has n input nodes such that $X1, X2, \dots, Xn$. The output nodes are indicated as $Y1, Y2, \dots, Ym$. Hidden layer 1 to L of the network have $N1, N2, \dots, NL$ hidden neurons, respectively. These neurons from one layer to another are connected by specific weight values. Weights which connect input nodes and neurons in the 1st hidden layer are indicated as W^1 . Likewise, weights that connect neurons of layer L and output nodes are indicated as W^{L+1} . The weight from first input node, $X1$ to the first neuron of the layer 1 is indicated by $W_{X1,1}^1$. When the number of neurons in the hidden layers are increased, the number of weights in the structure are also increased.

Each neuron consists an activation function and a bias in it. All the inputs to the neuron are multiplied with its corresponding weight values before come into the neuron. These multiplied inputs are added up along with neuron's bias before go through the transfer function. When the input series to the neuron is X , the corresponding weight value matrix is W , and the bias of the neuron is b , the input to the transfer function can be written as,

$$\text{transfer function input} = \sum_{i=1}^n (W_i \times X_i) + b$$

The transfer function takes its input to a different domain based on the type of the function. Commonly used transfer functions are the Log-Sigmoid Transfer Function, Linear Transfer Function, and Hard-Limit Transfer Function.

The weights that connect each node are adjusted during the training phase. The job of the training algorithm is adjusting the weights of the network so that it gives the minimum error between the network output for the training inputs and the target data. The weights adjusting procedures are different from one training algorithm to another. These procedures are explained in the following section in details.

3.2 Training algorithms

3.2.1 Back propagation

“Backward Prorogation of Errors” has been abbreviated to Back Propagation. Back propagation is one of training methods for adjusting weights in NNs and it is the most common training method among researchers. Weights are adjusted based on the given target series. Therefore, this method is called as supervise training algorithm.

Initially, the algorithm starts giving random values for weights. Inputs pass through these weights and hidden neurons to get the network outputs. These network outputs are compared with the given target values to calculate the errors. These errors are propagated to the previous layers and during the process, weights are adjusted to minimize the errors. This process runs until the output error becomes less than a predetermined value or stop after running predetermined number of cycles (epochs). After stop the weights adjusting process, the network is ready to work with unseen data. These weights adjusting process is simply explained below. For an example, a network with one hidden layer, two inputs (IN_1 , IN_2), two neurons in the hidden layer (H_1 , H_2), and two outputs (OUT_1 , OUT_2) are given in the following figure. H_1 , H_2 , OUT_1 , and OUT_2 have their bias in it.

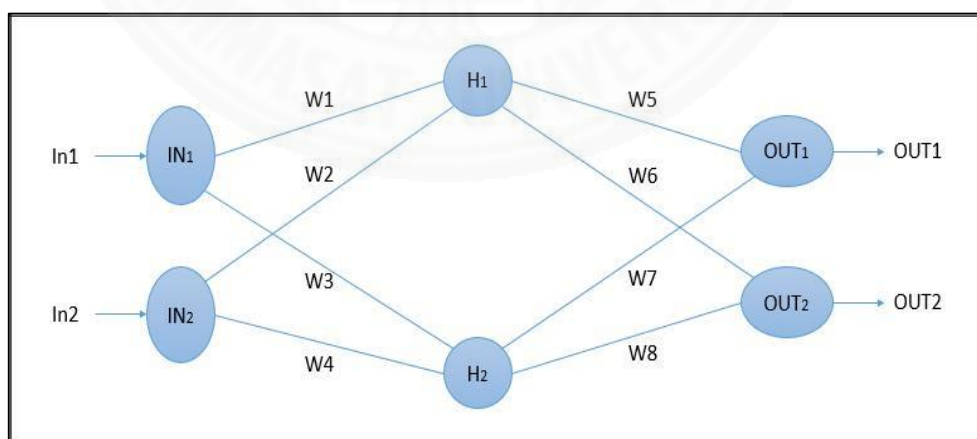


Figure 3.2 An NN structure for BP explanation

Each of these nodes and neurons are connected by weights, from W_1 to W_8 . Now, the back propagation is ready to optimize these weights so that given inputs (In_1 ,

In_2) can pass through these weights to the given outputs ($OUT1$, $OUT2$). To get the error between network outputs and given target values, first, network is fed with the given inputs assuming neurons consist with logistic transfer function.

Transfer function inputs for both $H1$ and $H2$ can be written as follows.

$$\begin{aligned} In_{H1} &= In_1 \times W1 + In_2 \times W2 + b_{H1} \\ In_{H2} &= In_1 \times W3 + In_2 \times W4 + b_{H2} \end{aligned}$$

After go through the logistic transfer function, outputs from $H1$, and $H2$ can be written as,

$$\begin{aligned} H1_{out} &= \frac{1}{1 + e^{-(In_{H1})}} \\ H2_{out} &= \frac{1}{1 + e^{-(In_{H2})}} \end{aligned}$$

Hidden neuron outputs are the inputs for output nodes. Therefore, Transfer function inputs of $OUT1$, and $OUT2$ can be written as,

$$\begin{aligned} In_{OUT1} &= H1_{out} \times W5 + H2_{out} \times W7 + b_{OUT1} \\ In_{OUT2} &= H1_{out} \times W6 + H2_{out} \times W8 + b_{OUT2} \end{aligned}$$

These inputs go through the transfer functions in the output nodes and outputs from the network can be calculated as,

$$\begin{aligned} OUT1_{out} &= \frac{1}{1 + e^{-(In_{Out1})}} \\ OUT2_{out} &= \frac{1}{1 + e^{-(In_{Out2})}} \end{aligned}$$

Since now we have the network outputs, we can easily calculate the error of each output node. For interpreting this error of each node, squared error function is used.

$$E_{out1} = \frac{1}{2}(OUT1 - OUT1_{out})^2$$

$$E_{out2} = \frac{1}{2}(OUT2 - OUT2_{out})^2$$

Total error from all the output nodes are equal to the summation of individual errors from each node.

$$E_{total} = E_{out1} + E_{out2}$$

Now, back propagation uses this total error to update weights which connect the output layer with the adjacent layer. The change on $W5$ due to the total error is the partial derivative of total error with respect to $W5$ and can be written as,

$$\text{Change on } W5 = \frac{\partial E_{total}}{\partial W5}$$

Using the chain rule, change on $W5$ can be written in the following form,

$$\frac{\partial E_{total}}{\partial W5} = \frac{\partial E_{total}}{\partial OUT1_{out}} \times \frac{\partial OUT1_{out}}{\partial In_{OUT1}} \times \frac{\partial In_{OUT1}}{\partial W5}$$

Each of these terms can be derived and given below.

Partial derivative of total error with respect to $OUT1_{out}$,

$$E_{total} = \frac{1}{2}(OUT1 - OUT1_{out})^2 + \frac{1}{2}(OUT2 - OUT2_{out})^2$$

$$\frac{\partial E_{total}}{\partial OUT1_{out}} = 2 \times \frac{1}{2}(OUT1 - OUT1_{out})^{2-1} \times -1$$

$$\frac{\partial E_{total}}{\partial OUT1_{out}} = OUT1_{out} - OUT1$$

Partial derivative of $OUT1_{out}$ with respect to In_{OUT1} ,

$$OUT1_{out} = \frac{1}{1 + e^{-(In_{out1})}}$$

$$\frac{\partial OUT1_{out}}{\partial In_{OUT1}} = OUT1_{out} (1 - OUT1_{out})$$

Partial derivative of In_{OUT1} with respect to $W5$,

$$In_{OUT1} = H1_{out} \times W5 + H2_{out} \times W7 + bOUT1$$

$$\frac{\partial In_{OUT1}}{\partial W5} = 1 \times H1_{out} \times W5^{1-1}$$

$$\frac{\partial In_{OUT1}}{\partial W5} = H1_{out}$$

Therefore, these results help to get the partial derivative of total error with respect to $W5$, $(\partial E_{total})/(\partial W5)$. Now we can calculate the new value of $W5$ as follows,

$$W5^+ = W5 - \eta \times \frac{\partial E_{total}}{\partial W5}$$

η is the learning rate (η , between 0 and 1) and $W5^+$ is the new or the updated weight value. Therefore, $W5$ is replaced by $W5^+$. Likewise, other weight values which are adjacent to the output layer ($W6$, $W7$, and $W8$) are updated in the same way.

After that, weights between the input layer and hidden layer are updated. As an example, change on the weight $W1$ can be written in the same way,

$$\frac{\partial E_{total}}{\partial W1} = \frac{\partial E_{total}}{\partial H1_{out}} \times \frac{\partial H1_{out}}{\partial In_{H1}} \times \frac{\partial In_{H1}}{\partial W1}$$

But, output from the $H1$, affects both network outputs, $OUT1_{out}$ and $OUT2_{out}$. Therefore, $\frac{\partial E_{total}}{\partial H1_{out}}$ has to be calculated slightly different way than before. Since it affects both outputs, the term can be written in the following way,

$$\frac{\partial E_{total}}{\partial H1_{out}} = \frac{\partial E_{out1}}{\partial H1_{out}} + \frac{\partial E_{out2}}{\partial H1_{out}}$$

Then we can apply the same procedure (derivatives, as did in the $W5$ case) to find all the terms and can easily update the weight values between the input and hidden layers ($W1$, $W2$, $W3$, and $W4$). After finishing the update for the first round, same inputs are sent again through the updated weights and weight are updated for the second time. This process runs until the error comes to a minimal point or we can specify how many times this process should run to adjust the weights.

But, there are some disadvantages while using the back propagation. One of the most conversant problems is, it does not guarantee of finding the global minimum from the solution space. Since the back propagation acts as an optimization algorithm and it does not find solutions from the global space, it travels to a local minimum depending on from where it starts finding solutions. The following figure shows the solution space with corresponding errors. For one solution space, there are many local minimums (except some special cases) and only one global minimum. Therefore, finding the global minimum is vital since the error difference between the global and some of local minimums are really high.

Since the back propagation starts with random weights, the solution can be started from any place in the solution space. If the back propagation starts finding solutions from a point where the above figure has given, it would never reach to the global minimum. Because of this reason, back propagation algorithm is sometimes called as a “local optimization algorithm”. Therefore, to get accurate results with neural networks, a global optimization algorithm is needed so that it adjusts the weights of the network in the optimal way. Genetic Algorithms, Evolution Strategy, Learning

Classifier Systems, Particle Swarm Optimization, Differential Evolution, and Ant Colony Optimization are some of well-known optimization algorithms which we can try to use with Neural Networks.

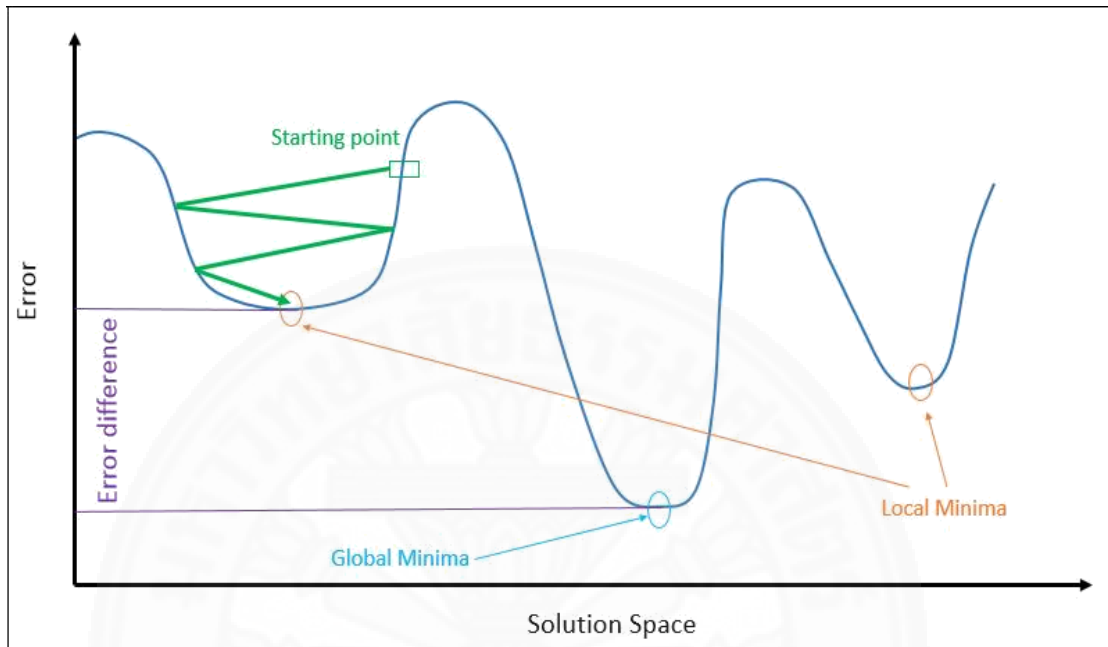


Figure 3.3 Back propagation stops at a local minimum

3.2.2 Genetic Algorithm (GA)

The Genetic algorithm (GA) is a method that we use to solve optimization problems. This method is based on the natural selection or this method has been driven from biological evolution (Survival of the Fittest) which was first introduced by Charles Darwin. Optimization problems consist of an objective function with a set of constraints which defined the boundaries of the variables in the objective function. The following figure (Figure 3.4) shows an example for an optimization problem.

The objective of the problem could be maximized or minimized the given objective function. In the following example, the objective function is indicated by $f(W)$. The objective of this problem is to minimize $f(W)$. $W1$ and $W2$ are the variables that vary the objective value and these variables are bounded by four constraints. Now, we need an optimization method to find the minimum $f(W)$. A number of optimization

methods are available to solve this kind of problems. But Genetic Algorithm (GA) has the ability of solving any kind of problems with a number of variables and problems which have non-differentiable objective functions, stochastic objective functions, discontinues objective functions, or highly nonlinear objective functions.

$$\begin{aligned} \text{Min } f(W) &= 100 * (W1^2 - W2)^2 + (1 - W1)^2 \\ \text{Subject to,} \\ W1 * W2 + W1 - W2 + 1.5 &\leq 0, && \text{(nonlinear constraint)} \\ 10 - W1 * W2 &\leq 0, && \text{(nonlinear constraint)} \\ 0 \leq W1 &\leq 1, && \text{(bound)} \\ 0 \leq W2 &\leq 13 && \text{(bound)} \end{aligned}$$

Figure 3.4 A sample optimization problem

Basic steps used in GA start with a random population. This population includes a set of individual solutions. These individual solutions are called Chromosomes. Random values for $W1$ and $W2$ in the above example bring different objective values for $f(W)$. These random values in chromosomes are called as Genes. Hence, the number of variables in the objective function should be equal to the number of genes in a chromosome. In the next step, GA calculates fitness values of each chromosome. Simply, the values of $W1$ and $W2$ are put in $f(W)$ to get the fitness value of each chromosome in the population. Then, GA can find the best chromosomes based on their fitness values and name as parents. If the optimization problem wants to minimize the given objective function, chromosomes with lowest fitness values have a higher chance for selecting as parents. If the optimization problem wants to maximize the objective function, chromosomes with highest fitness values have a higher chance for selecting as parents. GA uses these parent in the current population to produce children for the next generation. Therefore, children in the next generation have the genes from the parents of the current population. This is how GA expects to get better and better fitness values in each generation. Over successive generations, chromosomes in the populations evolve toward an optimal solution.

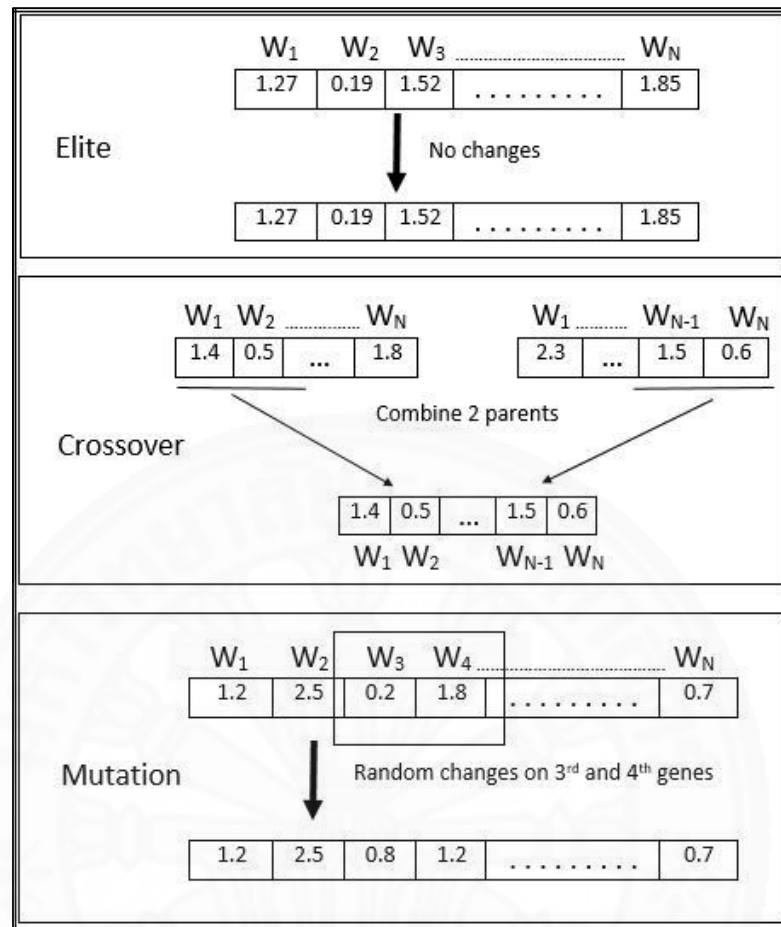


Figure 3.5 Three methods for producing children for the next generation

Three methods are used to produce children for the next generation from the selected parents: Elite, Crossover, and Mutation. Chromosomes which have the best fitness values use as the elite children. Without making changes on them, they are directly put in the next generation. Therefore, they have the same fitness values in the next generation as they have the same genes from the current population. Two random parents are combined to produce crossover children. Therefore, these children have genes from both parents and are expected perform better in the next generation. The third and the last method for producing children is mutation. Random changes are made on genes of randomly selected parent from the current population. Sometimes, changes are applied for all the genes of the selected parent and sometimes changes are applied only on few genes. All these three methods are graphically illustrated by Figure 3.5.

Here in these examples, all the chromosomes are consisted with “ N ” number of genes. Which means that objective function has “ N ” variables. When GA produces new children, they also should consist with the same number of genes same as their parents. Therefore, children in the above examples have “ N ” genes.

Still, there are some limitations while using GA for adjusting weights in NNs. The hardest thing is creating a convenient fitness function or the objective function which is optimized by GA. In this research, we simply calculate the Mean Squared Error (MSE) between training outputs and target values. But, there are many ways of calculating errors: MAPE, RMSE. Sometimes other functions can be better than MSE, sometime MSE can be the best one. Therefore, we have to go for all the possible functions and select the best one.

Other problem with GA is choosing parameters. We don't have exact values for the number of generations we need, population size, how many parents we should select from the population, percentages for different children, functions use to produce children, and stopping conditions. Therefore, for all these parameters, trial and error method is suggested. It takes some times for adjusting all these parameters in the optimal way or sometimes it is not possible to adjust all the parameters in the optimal way.

Experiments which we did so far takes large number generations and the number of chromosomes in a generation (population size) is also high for giving considerable results. Therefore, it is too slow for running or takes much time for giving results. That is one point where we have to decide whether we need more accurate results consuming more time or that current results are enough with consuming lesser time.

3.2.3. Particle Swarm Optimization (PSO)

Particle swarm optimization has some similarities with GA which we discussed in the previous section. Both methods start with random solutions for the given objective function (fitness function). In PSO, individual solutions in the population are called particles (in GA, individual solutions are called chromosomes). Both methods check the conditions of each individual by calculating the fitness value of them. Based on the calculated fitness values, they produce the next population until they meet the stopping conditions. But PSO does not have evolution operators like in GA: producing crossover and mutation children. Its individuals (particles) follow the best particles from their solution space and run until they reach to the target. Since it has few parameters to adjust, it is easier to use and control than GA.

As many researchers introduce, PSO can be described as a flock of birds (or school of fish) who are finding a food source in a particular area. The one who closest to the source makes a noise so that others can come closer to him. In this case, a new bird can come closer to the source in the next step than the previous one. Then other birds start to follow the new one who is closest to the source. Likewise, all of them follow individuals until one can find the exact place. Following figure shows a same kind of flock of birds who are following individuals as described early. Therefore, each particles or individual solutions in PSO are equal to individual birds in a flock. Any bird can reach to the source of food. But it depends on their current location and how fast they can fly.

As stated before, PSO algorithm works as a flock of birds finding a food source. At the beginning, no one knows where it is. The strategy they use is, just follow the one who is currently closest to the target. The same concept is used by particles where they travel through the solution space following the current best particle. Each particle has its own velocity for reaching to the target. Velocity is another parameter addition to the fitness value of each particle which decides how fast it travels to the target. After PSO calculates the fitness of each particle using MSE, it updates two variables.



Figure 3.6 A flock of birds that illustrates PSO

The first one is “pbest”. The pbest is the abbreviated name of “personal best”. Each particle has its own personal best value. That is the best fitness value that each particle has obtained so far. In this research, since we need to minimize the objective function (MSE), the pbest of a given particle is the lowest fitness value that particle has achieved so far. The other variable is “gbest”. The gbest is the abbreviated name of “global best”. The gbest is the best fitness value that any particle has achieved so far from the whole population. Therefore, after each iteration, gbest has to be updated if there is any fitness value better than the current gbest. When the fitness value of the i^{th} particle is equal to F ,

$$F = \text{fit}(i).$$

If the personal best fitness of the same particle is equal to $\text{fit}(p)$, and

$$\text{if } \text{fit}(p) < F, F = \text{pbest}$$

Otherwise, $\text{pbest} = \text{pbest}$ (does not change)

Likewise, if the fitness value of global best equal to $\text{fit}(g)$ and

$$\text{if } \text{fit}(g) < F, F = \text{gbest}$$

Otherwise, $\text{gbest} = \text{gbest}$ (does not change)

Next step is to update the velocities and positions of all the particles from the current population to the next population. So far, we stated the fitness function, started with random solutions, calculated the fitness of each particle and finally updated two variables: pbest and gbest. These two updated variables are used to update the velocity and position of each particle. First, the following equation is used to update the velocity of each particle.

$$V^i(i + 1) = w \times V^i(t) + c1 \times rand1 \times (p - X^i(t)) + c2 \times rand2 \times (g - X^i(t))$$

$V^i(i + 1)$ is the updated velocity of the i^{th} particle. This velocity is calculated using the current velocity $V^i(t)$, the difference between the current position ($X^i(t)$) and the personal best position (p) of that particle: ($p - X^i(t)$), and the difference between the current position and the global best position (g) from the whole population, ($g - X^i(t)$). w is the inertia weight. The velocity of the i^{th} particle in the next generation is varied according to the inertia weight. If w is a small value, the velocity of the i^{th} particle in the next generation has higher impact from pbest and gbest positions. But if w is high, the velocity of the i^{th} particle for the next generation has a higher impact from the current velocity. $rand1$ and $rand2$ are random values between 1 and 0. $c1$ and $c2$ are learning factors and most of the time $c1 = c2 = 2$.

The following equation has been given to update the position of the i^{th} particle.

$$X^i(t + 1) = X^i(t) + V^i(t + 1)$$

To update the position of the i^{th} particle ($X^i(t + 1)$), the current position ($X^i(t)$) and the updated velocity ($V^i(t + 1)$) is used.

Same like in GA, there are few limitations while we using the PSO for adjusting weights in NNs. First, we have to create an objective function for the optimization

problem which we use to calculate the fitness value of each particle. Same as in GA, MSE is calculated between NN's training outputs and target series. But, there might have different function which we can use as the objective function or the fitness function in PSO. Therefore, we have to check different functions such as MAPE, RMSE for the objective function and select the best one.

Other than this, still there are no exact values for the PSO parameters such as learning factors, the number of particles in a population, random values which we use to update velocity of each particle and so on. Therefore, most of the time we have to go for trial and error method to find the optimal values for these parameters. It takes more time and sometimes hard to find the optimal values for all the parameters.

Stopping conditions also have effects on PSO results. Sometimes we need large amount of iterations for adjusting weights and bias in the optimal way. Or else we can set very precise target values such that training can stop when the fitness value reach to the given target. But both of these consume lot of time to train the weights in NNs.

3.3 Data Cleaning

Data gathered by Electricity Generating Authority of Thailand (EGAT) for the years from 2009 to 2013 has repeated patterns due to seasonality, weekly and daily electricity consumptions. As consumption vary with the temperature, weather, and holidays, these repeated patterns are interrupted. Also, data is corrupted or missed while gathering them due to unavoidable reasons. Therefore, this hardens the forecasting process for regular days using the regular data from the history. As a solution for the problem of having unusual patterns and the outliers in the data set, both load and temperature data is cleaned using the time window based data cleaning technique. Temperature data has only outliers and missing values as they do not vary with the other factors. The specialty of the proposed technique is, unusual data are identified at three different stages. All these identified unusual data are replaced with the estimated data using the Moving Average method.

3.3.1. Detecting and Replacing Holidays

The common type of pattern changers are holidays. Since all the companies are closed and stop their works on holidays, there is a huge reduction in the electricity consumption on holidays compared to the regular working days. Including these low electricity demands in the training data set for forecasting regular days can reduce the forecasting accuracy. Therefore, calendar holidays of the sample data set which is from 2012 (for training) to 2013 (for testing) are identified and replaced using the following weighted moving average equation. $L_t(d)$ of the equation represent the load demand of day d at time period t .

$$L_t(d) = [w_1 * L_t(d-7) + w_2 * L_t(d-14)]$$

Two days from the most recent two weeks are selected to replace the holiday's data. The highest priority has been given to the previous week by giving the w_1 as 0.7. The weight w_2 is set to be 0.3.

3.3.2 Detecting and Replacing Bridging Holidays

The following table is given to get an idea about the bridging holidays that we consider in this research. Only the given three cases from many other cases are considered as other cases are hard to identify and have minor effect on the demand variations.

Table 4.1 Identifying bridging holidays

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Case 1		B'Holiday	Holiday				
Case 2					Holiday	B'Holiday	
Case 3			Holiday	B'Holiday	Holiday		

If there is a holiday on Tuesday and Monday is not a holiday, we consider Monday as a bridging holiday (B'Holiday). Likewise, when there is a holiday on Thursday and the Friday is not a holiday, we consider that Friday as a bridging holiday (B'Holiday). The last case we consider is, when there is a non-holiday between two holidays, we take that non-holiday as a bridging holiday (B'Holiday). Therefore, after we recognize all the bridging holidays in the sample data set, all of them are replaced with the same equation that used to replace the holidays' data.

3.3.3 Detecting and Replacing Outliers

Due to measurement errors, some data change the regular behavior of the load consumption. These data have to be identified as outliers and replace them with estimated data.

3.3.3.1 Detecting Outliers

Some strategies have to be used for identifying the outliers in the sample data set. If we use the complete data set directly for identifying the outliers, it complicates the process as outliers can hide among the regular patterns. Therefore, similar consumption behaviors of similar time intervals of similar days are identified. Then data are separated into different time windows, considering their similar patterns, to have less demand variation within each time window. This strategy helps to divide the sample dataset into 336 (7*48) time windows as we have data for each day for each 30-minute. The time window for time t , of day d for m weeks is given by the following equation.

$$V_t(d) = [L_t(d'), L_t(d' - 7), L_t(d' - 14), \dots, L_t(d' - 7 \times m)]$$

Where $d \in \{d', d' - 7, \dots, d' - 7m\}$ and d' represents the last 7 days in the selected data set. The variable m varies with the type of the day as different days of the weeks have different number of weeks within the sample data set.

Using the standard deviation of the time window $V_t(d)$ and the moving average for k periods ($k = 4$), a filtering band is created to detect the outliers of each time window. The k -period filtering band is given by the following equation.

$$B_t(d) = \frac{[\sum_{i=1}^k L_t(d-7 \times i)]}{k} \pm N \times SD(V_t(d)); \quad t = 1, \dots, 48, \forall d$$

The variable N defines the width of the k -period filtering band. While large N detects higher number of outliers, small N detects small number of outliers. However, the optimum N used in this research is 1.6.

3.3.3.2 Replacing Outliers

The detected outliers in the previous section using the k -period time window filtering band are replaced by moving average method. The average of the data from similar time periods from similar days of last 2 weeks is used to replace the detected outliers. The moving average equation for replacing the outliers are giving by the following equation.

$$L_t^p(d) = \frac{[\sum_{i=1}^n L_t(d - 7 \times i)]}{n}$$

Where $L_t^p(d)$ is the estimated load for the outlier detected at time period t of day d and n is set to 2.

Once the data of the sample set is clean, they are ready to feed into the neural network. The network arrangement and the training it with the cleaned data using the training algorithms discussed in the section 3.2 are described in the following sections.

3.4 Evaluating the Accuracy

With the suggested NN, there is one output node to forecast each time period of the day separately with separate training data sets. In this research, forecasted results for day d is denoted by $F_t(d)$. The t is used to denote the time intervals of day d . Therefore, the forecasted value for 12.00 a.m. is denoted by $F_1(d)$ and it belongs to the demand day d . Likewise, all the 48 demand values are forecasted separately and given as a time series, $F_t(d)$ where $t = 1, 2, 3, \dots, 48$. However, after cleaning the data with the time window based cleaning technique, cleaned data are used to train and test the performance. In that case, forecasted outputs are denoted as $F_t^p(d)$. These forecasted series can be compared with two actual demand series for day d , which are $L_t(d)$ and $L_t^p(d)$. Where $L_t(d)$ is the actual load data of time t on day d before cleaning the data and $L_t^p(d)$ is the actual load data of time t on day d after cleaning the data. Therefore, two error series can be calculated as given below using the time series created above.

$$e_t(d) = L_t(d) - F_t(d)$$

$$e_t^p(d) = L_t^p(d) - F_t^p(d)$$

$e_t(d)$ and $e_t^p(d)$ also have 48 values each in their time series. These values sometimes can be positive or negative based on the forecasted values of day d . But, by looking at this error series, reader cannot get a clear idea about the performance of the technique. Therefore, this error series has to be interpreted with a single numerical value so that everyone can come to a conclusion about the used technique. For this purpose, there are many techniques have been introduced. Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Mean Absolute Deviation (MAD), and Root Mean Squared Error (RMSE) are some of well-known techniques for interpreting forecasting performance.

The MAPE gives the error value in a percentage form. When the error series is given as $e_1(d), e_2(d), e_3(d), \dots, e_{48}(d)$, MAPE can be calculated by the following equation.

$$MAPE = \left[\left(\frac{1}{48} \right) \sum_{t=0}^{48} \left| \frac{e_t(d)}{L_t(d)} \right| \right] \times 100$$

However, when the error series is given as $e_1^p(d), e_2^p(d), e_3^p(d) \dots \dots \dots e_{48}^p(d)$, MAPE can be calculated by the following equation and denoted as $MAPE^p$.

$$MAPE^p = \left[\left(\frac{1}{48} \right) \sum_{t=0}^{48} \left| \frac{e_t^p(d)}{L_t^p(d)} \right| \right] \times 100$$

Many researchers use this MAPE in their research for interpreting errors as readers are more comfortable when they see a percentage value in the error. But MAPE should not be used when the actual demand series consists with very small values. As an example, if the actual demand value at time t is equal to zero, the MAPE at time t is not defined as the error is divided by zero. Likewise, small actual demand values may lead to have high MAPE values, which is a negative insight for the technique.

All these error interpretation techniques give positive values for their outcomes. When MAPE is calculated, the absolute term is used to get a positive value in the result. But in MSE and RMSE, calculations are done without using absolute terms. As they directly get the squared value of all errors, all negative errors are converted to a positive value and outputs from MSE and RMSE are also positive. Equations for both MSE and RMSE are given below.

$$MSE = \left(\frac{1}{48} \right) \sum_{t=0}^{48} e_t(d)^2$$

$$RMSE = \sqrt{\left(\frac{1}{48} \right) \sum_{t=0}^{48} e_t(d)^2}$$

MAD is the easiest way of taking the error series into a single numerical value. It directly takes the average of the absolute error. For calculating the MAD for the given error series, following equation can be used.

$$MAD = \left(\frac{1}{48}\right) \sum_{t=0}^{48} |e_t(d)|$$

In all these techniques, t goes from 0 to 48 in the sum function as it has 48 demand values for a given day. Likewise, MAPE, MSE, RMSE, or MAD can be calculated for any given day, d . Sometimes we have to take the average values for many days or sometimes we can directly put these values in a table to find how good those used methods for specific days in forecasting.

Chapter 4

Design of Experiments

4.1 Case 1: Study of Artificial Neural Networks' Parameters

There are many parameters in ANNs which can be adjusted to enhance the forecasting performance, such as the number of hidden layers, the number of neurons in hidden layers, epochs, etc. Adjusting parameters should be done carefully since it causes not only on enhancing performances, but also can reduce the performances. Therefore, this experiment is conducted to find the relationships of the number of hidden layers and the number of neurons in hidden layers with the number of inputs to the ANN. Later, the performance of transfer functions is analyzed to select the best transfer function for electricity forecasting.

4.1.1 Data Arrangement

To follow the main objective, a sample data set is selected from the population and data are arranged accordingly. Bangkok region's loads in 2013 are selected as the sample set, since, it has a great demand variation over time. All weekend and holiday demands are removed from the sample set to characterize the objective clearly. Three data sets are arranged from the selected sample set so that, each has 5, 10, and 20 inputs to the ANN. For all cases, two months of data (from September and October) are used to train the ANN and November is used as the testing month. At end of the training phase, testing inputs are used to forecast working days in November. MAPE is used to calculate the percentage error between actual and forecasted series. MAPE for 48 elements are calculated since each day has 48 demand values. Likewise, MAPEs for all the testing days are calculated under each case. Average MAPE of the testing days represents the goodness of that specific case: 5 inputs, 10 inputs, and 20 inputs. As an example, the data arrangement for 5 inputs case is given in Table 4.1. The simple

concept for 5 input case is, for forecasting the next day electricity demand, electricity demands of the most recent 5 days are used.

Table 4.1 Data arrangement for the 5-input case

		5-Input case				
		Inputs				Targets
Training Data	Mon	Tue	Wed	Thu	Fri	Mon
	Tue	Wed	Thu	Fri	Mon	Tue
	Wed	Thu	Fri	Mon	Tue	Wed
	⋮	⋮	⋮	⋮	⋮	⋮
	Thu	Fri	Mon	Tue	Wed	Thu
Testing data set	Fri	Mon	Tue	Wed	Thu	Fri

4.1.2 Artificial Neural Network Structure

After arranging data for the above three cases, the next step is to set the parameters in the ANN to find relationships as specified in the objective. Each case is tested with different parameter arrangements in the ANN. In the first arrangement, the ANN has only one hidden layer and each time it has different number of neurons such that they are equal to 1, 3, 5, 10, 15, 20, 30, and 40. In the second arrangement, the ANN has two hidden layers where the number of neurons are varied the same way as in the first arrangement. In the second arrangement, the number of neurons of both hidden layers are always equal.

4.1.3 Training and Testing

After setting up data and the NN structure, each case is tested separately with different parameter arrangements. As an example, the data set of 5 inputs is tested with parameter arrangement of one hidden layer ($L = 1$) and one neuron ($N_l = 1$) to get

MAPEs of working days in November. But MAPEs are varied for each run, even with same data set, due to the random initialization of weight values. Therefore, average MAPEs are taken after running minimum of 5 replications. A secondary experiment is used to show that 5 MAPEs are enough for talking average values. To obtain MAPEs of one month, the NN takes about 8 minutes. Therefore, around 40 minutes are required for 5 replications.

Neurons in each layer can consist of different transfer functions. Since different transfer functions perform different tasks, it differently affects forecasting results. Also, these transfer functions take different computational times to finish their tasks. Therefore, to check the variations of forecasting accuracy, and to analyse the computational times, the NN is run with different transfer functions with the same data set. For that, four commonly used transfer functions are used: Tan-Sigmoid Transfer Function (tansig), Log-Sigmoid Transfer Function (Logsig), Satlin Transfer Function (satlin), and Linear Transfer Function (purelin). Other conditions such as, the number of hidden layers, number of neurons, and the number of epochs are same to be fair for all functions. One hidden layer with 3 neurons and 100 epochs are given for each case. Data set with 20 inputs is used with all four functions.

First, transfer functions in 3 neurons in the hidden layer are changed to Tan-Sigmoid Transfer Function and run with 20 inputs data set. After 5 replications, average MAPE and computational times are noted down. Likewise, each transfer function is tested with the same conditions and with the same data set. Average MAPEs and computational times are summarized in the results section.

4.2 Case 2: Effect of the Amount of Training Data and Number of Inputs

The performance of the ANN changes with many factors. The amount of training data and the number of inputs to the neural network mainly effect to the forecasting outcomes. Therefore, an experiment is designed to determine the optimum amount of training data and the number of inputs to the neural network.

4.2.1 Data Arrangement

This experiment focuses on forecasting weekday demands as they contain more variations throughout the day. Therefore, all weekends and public holidays are removed from the available data set. For consistency of the testing phase for all cases, the load demand in November is used as the testing month. As an example, when 6 months of data used in the training phase, May, June, July, August, September, and October data are used to train the network. November data is used for testing performance.

Since NN takes more time with a large amount of data in the training phase, the maximum amount of data to train the network is limited to 10 months. Training data sets consist of 4 types: 2-month, 4-month, 6-month and 10-month. Moreover, the maximum number of inputs are limited to 20 days. This experiment has been designed to include 1, 5, 10, 15, and 20 inputs. Therefore, a total of 20 cases are tested, as shown in Table 4.2. As data have been gathered every 30 minutes, there are 48 demand periods per day. Each cell of the Table 4.2 is given demand values which were available in each case to train the network. As an example, for 1 input with 2-month training data case, it covers a total of 42 days from September 1st to October 31st, except weekends and holidays. Therefore, the input series of training phase includes 2016 (42×48) of electricity demand values and same amount of demand values in the target series as well. These two values are combined to train the network for this specific case and it is equal to 4032 ($2016 + 2016$) values.

Table 4.2 Number of electricity demand values for training each case

		Training Data Size (Electricity Demand Values)			
		2 Months	4 Months	6 Months	10 Months
Number of Inputs	1	4,032	8,160	12,000	19,680
	5	12,096	24,480	36,000	59,040
	10	23,232	45,936	67,056	102,960
	15	33,792	66,816	97,536	146,688
	20	44,352	87,696	128,016	185,472

Likewise, after arranging all training data sets, the network is trained with different inputs, as specified in Table 4.2. Selection of inputs, training of the network, and testing performances are discussed in the next section.

4.2.2 Artificial Neural Network Structure

To train all cases in Table 4.3, network was consisted with two hidden layers except input and output layers. Neurons of those two hidden layers are changed according to the number of inputs of each case so that, the number of neurons greater than or equal to the number of inputs. As an example, if the number of inputs to the network are 10, neurons for both layers are given 10 or higher and same amount for both layers. As it changes weight values at each epoch to minimize the error between inputs and target series in the training phase, 100 epochs are set for all 20 cases.

4.2.3 Training and Testing

Next day demand forecasting using previous day demand is the primary experiment in this section. But, it becomes more complex when it uses more than one previous days to predict the next day demand. Following table is used to demonstrate the concept of arranging training data sets.

Table 4.3 Example Data arrangement for 1 input and 3 input cases

No.	1 Input		3 Inputs			
	Input	Target	Inputs			Target
1	Mon	Tue	Mon	Tue	Wed	Thu
2	Tue	Wed	Tue	Wed	Thu	Fri
3	Wed	Thu	Wed	Thu	Fri	Mon
.
.
.
m	Wed	Thu	Fri	Mon	Tue	Wed
Testing data set	Thu	Fri	Mon	Tue	Wed	Thu

In Table 4.3, the first column is for 1 input or next day demand forecast using previous day load values. Therefore, the network is built with one input and one output for testing phase. Target series in training phase is always lead by one day than input series in 1 input case, as it shows in Table 4.3. Each row represents pair of input and target series for training. Thus, there are “ m ” pairs in the training data set in Table 4.3. Likewise, for 3 inputs case given in Table 4.3, it uses three days of historical data to predict the next day demand. It has three inputs and one target series to train the NN.

After giving these inputs and target series to train the NN, the next step is to test the accuracy of the trained network. Therefore, according to the 1-input case in Table 4.3, the testing input is Thursday demands (48 elements) and it forecasts Friday demands. Also, this testing procedure can be applied for 3- inputs case as shown in Table 4.3. NN’s performance is evaluated in the testing phase by forecasting unseen data and calculating the error between these forecasted series and actual demand series using MAPE.

4.3 Case 3: Performances of Different Training Algorithms

4.3.1 Data Arrangement

This experiment is arranged to forecast each time period t of each day d separately for the year of 2013. Therefore, for forecasting the next day electricity demand, the network has to be trained 48 times as each day has been divided into 48 time periods. The similar load behavior of the same time periods of the previous day and the same day of the week is considered. Also, the concept of similar temperature behavior of the same time period of the previous day and the forecasted temperature values for the forecasting day are used. As an example, for forecasting the load demand of time period t of day d , actual load of the previous day at the same time period, $L_t(d-1)$, actual load of the same day of the week from the previous week at the same time

period, $L_t(d-7)$, actual temperature of the previous day at the same time period, $T_t(d-1)$, and the forecasted temperature for the same day at same time period, $T_t(d)$ are used.

One year data is used to train the network each time for forecasting each time period of each day in 2013. However, the strategy of using the similar load behavior of the same day of the week lessen the number of training sets to train the ANN. Considering the Monday as an example, for forecasting the time periods of Mondays in 2013, the ANN is trained with one year data so that only Mondays are in the target series. Therefore, only 52 training sets are used to forecast each Mondays in 2013. Likewise, 52 training sets are used to forecast each Sunday in 2013 and for forecasting all the other days have 51 training sets each.

4.3.2 Artificial Neural Network Structure

For this research, a neural network with 4 input nodes and 1 output node is created. Each time period t of the forecasting day d is forecasted separately. Therefore, electricity demand values from two days ($L_t(d-7)$ and $L_t(d-1)$) of the same time period t and temperature values from two days ($T_t(d-1)$ and $T_t(d)$) of the same time period t are given as inputs to the network as given in Figure 4.1.

One hidden layer with four neurons is added to the network. This helps preventing overfitting of data. The network has 16 weight values to connect the input nodes with hidden neurons. Likewise, network has another 4 weight values to connect hidden neurons with the output node. All hidden and output nodes have bias. Therefore, 5 bias can be found in the network (4+1). All these weights and bias are variables and adjusted during the training phase to achieve the given target. Therefore, total of 25 (16 + 4 + 5) variables can be found from the suggested network. This suggested NN structure is illustrated by Figure 4.1. $L_t(d)$ represents the load of day d , at time t . Likewise, $T_t(d)$, and $F_t(d)$ represent temperature and forecasted load of day d , at time t , respectively.

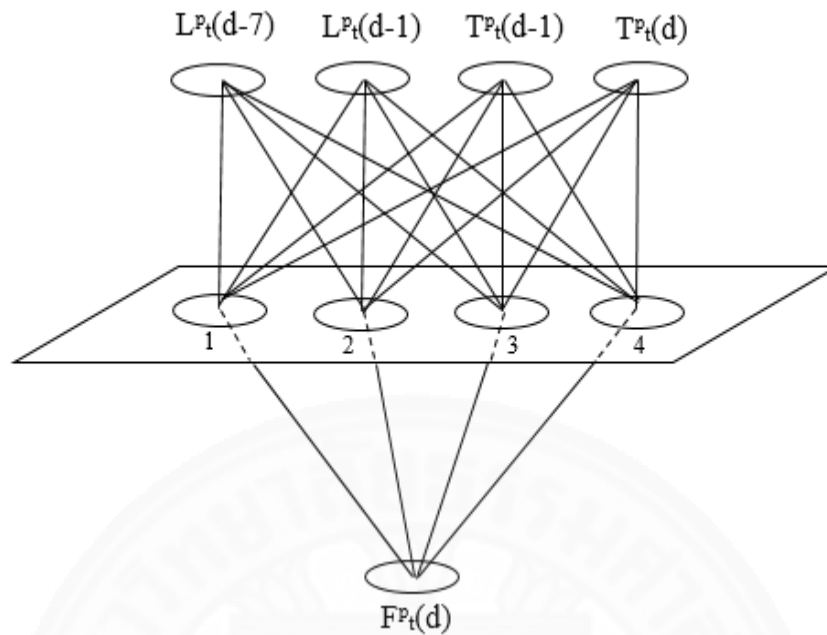


Figure 4.1 The suggested NN structure

Electricity demand values and also the temperature values for a given time period of the same day of the week do not have a big variation. Therefore, each input node receives steady input series and it is easy for weights and bias to adjust accordingly. That is the advantage of this structure has. Since the network is trained to forecast the same time period as in training inputs, this helps both input and layer weights to learn fast with less amount of data. And as a unit, this structure can easily recognize and learn complex non-linear patterns from both load and temperature series.

4.3.3 Train and Testing

4.3.3.1 Training ANN with Backpropagation (ANN-BP)

When the network and the data set is already prepared, the backpropagation algorithm is already ready to adjust the weights and bias. Only the stopping conditions have to be set for determining the stopping position of the training process. These stopping conditions are namely the minimum error (MSE) between the network outputs

and the target series and the maximum number of epochs that the algorithm should run. Until one of these conditions are met, the backpropagation adjusts the weights and bias based on the current error. Therefore, expecting the maximum performance, the minimum or the target error set to be zero and the maximum number of epochs is set to be 1000. The complete process of the backpropagation training algorithm is illustrated by the following flowchart (Figure 4.2)

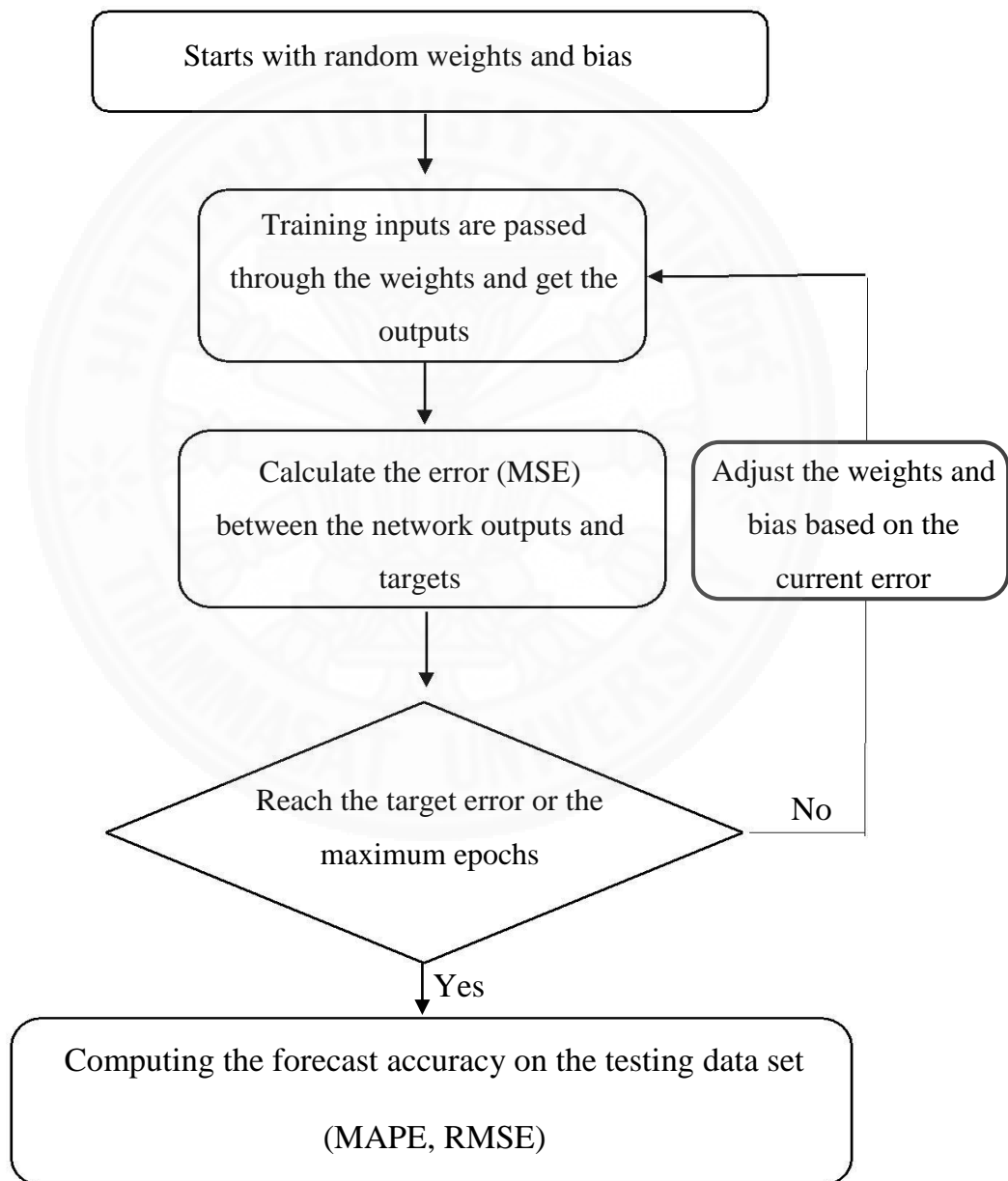


Figure 4.2 Flowchart for adjusting weights with backpropagation

4.3.3.2 Training ANN with Genetic Algorithm (ANN-GA)

(1) Encoding

Gene	Weights/Bias
1	$W_{1,1}^1$
2	$W_{1,2}^1$
.	.
.	.
.	.
.	.
16	$W_{4,4}^1$
17	$b_{(1,1)}$
.	.
.	.
.	.
20	$b_{(1,4)}$
21	$W_{1,1}^2$
.	.
.	.
.	.
24	$W_{4,1}^2$
25	$b_{(2,1)}$

Figure 4.3 Encoding NN weights and bias into chromosomes

Before train the weights and bias of the ANN using GA, first we should go for the encoding. Simply, we have to take all the weights and bias into chromosomes. Then, these weights and bias in the NN are represented by specific genes in the chromosomes. According to the suggested structure in the previous section, 25 genes should have in each chromosome. The first gene of each chromosome belongs to the weight that connect the first input node with the first hidden neuron. The next three genes belong to the weights that connect the first input node with the other three neurons in the hidden

layer. Likewise, all weights and bias have their specific places in chromosomes. Figure 4.3 helps to get an idea about the encoding process.

First 16 genes of the chromosomes belong to the weights that connect the input nodes and hidden neurons as shown in the figure. The bias of the hidden neuron ($b_{(1,1)}$ - $b_{(1,4)}$) are placed just after input weights. Since there is 1 output node, there are 4 weights and 1 bias. These weights are placed after the bias of the hidden neurons and then placed the bias of output nodes.

(2) Fitness values calculation

After the encoding process, next step is to calculate the fitness values of each chromosome. For this purpose, an objective function has to be created so that it can minimize the forecasting error. After calculating the fitness values of each chromosome, GA can select the parents from the current population to generate the children for the next generation.

Mean Squared Error (MSE) between the target series ($L_t(d)$) and the training output ($F_t(d)$) is used as the objective function in the optimization problem. This objective function is the fitness function in the GA to calculate fitness values of each chromosome. Therefore, weights and bias are adjusted and updated at every generation while minimizing the error between the target and training output series. MSE between target series and the training output is given by,

$$MSE = \frac{1}{n} \sum_{t=1}^n (L_t(d) - F_t(d))^2, \quad \text{for } n \text{ training sets.}$$

n represents the number of data sets we use to train the network or simply the number of days we have given in the target series for the specific time period t . Genes of the current chromosome are placed at their specific positions in the NN. Then, the training sets are fed into the network to get the outputs. MSE between the target series and training

output is calculated. The fitness value of the K^{th} chromosome in the G^{th} generation, $f(C_k^g)$ is equal to this MSE, Likewise, fitness vales of all the chromosomes (for all K) in the current generation is calculated. Chromosomes with the best fitness values are selected for producing the children for the next generation.

After calculating the fitness values of all the chromosomes, GA ranks these chromosomes based on their fitness values. In this case, the objective function of the optimization problem is the error between the target and training outputs (MSE). GA wants to minimize the objective function as it minimizes the error between target series and the training outputs. Therefore, chromosomes with the lowest fitness values have the highest chance for selecting as the parents. In the following figure, there are 100 chromosomes in the population and best 30% is selected as the parents.

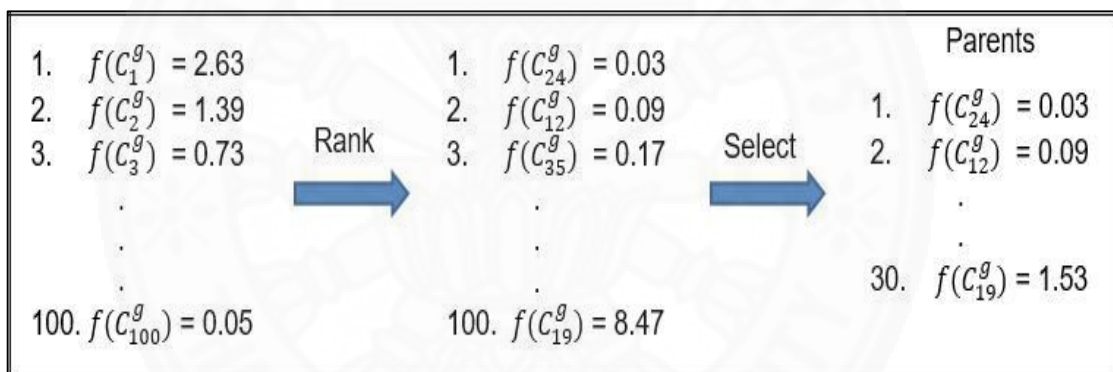


Figure 4.4 Selecting parents from the current population

(3) Next Generation

Since the population size of the GA is 100, the same number of chromosomes have to be created for the next generation using the parents from the previous generation. The composition of the next generation is 5% from elite children, 50% from the crossover children, and 45% from the mutation children.

- **Elite Children**

Without making changes on genes, the top 5 chromosomes of the ranked parents' list are selected directly for the next generation. As a result, still these elite children can select as parents in the next generation and also can contribute to make crossover and mutation children.

- **Crossover Children**

For producing a crossover child, two parents are selected randomly. As each parent has 25 genes in their chromosome, first 12 genes of the crossover child are added from one parent and the remaining 13 genes are added from the other parent. This process is run until it creates 50 crossover children.

- **Mutation Children**

Mutation children are produced adding random values which are selected from Gaussian distribution to the randomly selected 20% of genes. Therefore, in this research, as each chromosome has 25 genes, 5 randomly selected genes are changed by adding the random values from the Gaussian distribution. Total of 45 mutation children are produced for the next generation.

(4) Stopping conditions

GA uses these selected parents for producing children for the next generation as explained before. The process of producing children or generating new populations is stopped when GA fulfills one of following conditions.

- The number of generations – The maximum number of generations can be specified before start the GA algorithm. After GA produces this amount of generations, algorithm stops automatically.

- Time limit – An amount of time in seconds can be specified before start the GA algorithm. After GA runs this amount of time, algorithm stops automatically.
- Fitness limit – A minimum value for the fitness function (objective function) can be given before start the GA algorithm. After GA reaching the minimum fitness value, algorithm stops automatically.
- Stall generation – A generation length can be specified before start the GA algorithm. When there is no improvement in the fitness value between the specified generation length, algorithm stops automatically.
- Stall time limit – An amount of time in seconds can be specified before start the GA algorithm. When there is no improvement in the fitness value during the given time limit, algorithm stops automatically.

Processes from random initiation of weights to forecasting electricity demand is summarized in the following flow chart.

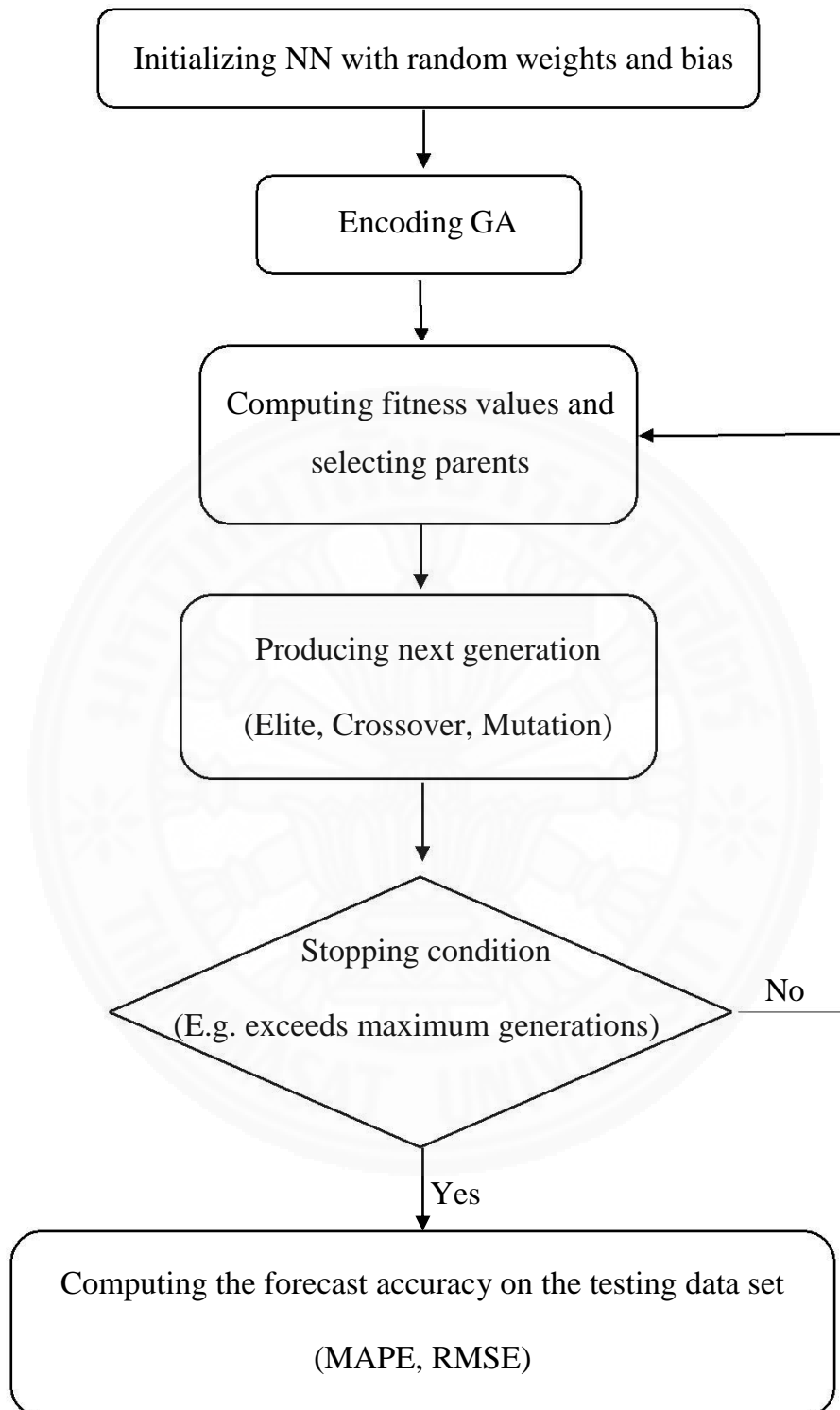


Figure 4.5 Flowchart for hybridizing GA and NN

4.3.3.3 Training ANN with Particle Swarm Optimization

(1) Encoding

Encoding and the fitness values calculation for PSO are same with GA. The neural network given in Figure 4.1 has 4 inputs, one hidden layer with 4 neurons, and 1 output node. All these nodes are connected with 25 weights and bias. Therefore, particles in PSO should have 25 elements. Then these elements represent the variables in the objective function. Same as in GA, each weight and bias have their specific places in particles (Figure 4.3). First 16 elements in particles are belong to weights that connect input nodes and hidden neurons. Bias of the hidden neuron comes after first 16 weights. Four weights, that connect hidden neuron with the output node, are put after the bias of the hidden neuron and finally bias of output node is placed. This encoding process have been clearly explained 4.3.4.1 section with Figure 4.1.

(2) Fitness values calculation

After creating places for each weight and bias of the network in particles of PSO, these particles start with random values. The next step is calculating the fitness of each particle. As did in GA, Mean Squared Error (MSE) between training outputs and target series is calculated. As an example, each element in a particle is placed in its belonged position in the NN and run with some training inputs. Training outputs are compared with target series and calculated the MSE. For calculating MSE, the same equation which used in GA can be used,

$$MSE = \frac{1}{n} \sum_{t=1}^n (L_t(d) - F_t(d))^2, \quad \text{for } n \text{ training sets.}$$

Recall, n is the number of training sets we use to train the network, otherwise, MSE is calculated for n number of training outputs. Again, this MSE is equal to the $f(C_k^g)$, where, $f(C_k^g)$ is the fitness value of k^{th} particle at g^{th} iteration. Likewise, fitness values

for all the particles are calculated. As elements in each particle are updated at each iteration, fitness values are changed and it minimize the error between the training outputs and target series. At each iteration, fitness values of particles help to update the personal best (pbest) and global best (gbest). These two values take all the particles in the population close to a given target or solution. This process is different than GA does: GA selects the parents after each generation based on their fitness values.

(3) Parameters and Stopping Conditions

In PSO, there are only few parameters which we have to be tuned for getting good results. The number of elements in each particle is already defined when the NN structure is made. Therefore, we do not need to adjust the number of elements in a particle and it is equal to the number of weights and bias in the NN. The number of particles in a population are varied on how difficult the problem is. With small number of variables in the objective function, a small number of particles can be used in the population. But for more difficult problems such as more than 100 variables in the objective function need 100 to 200 particles in the population. There is no specific range for the elements in particles to start. Sometimes some researchers use a large range for them, but simplest way is giving just random values to start them. We have already defined the learning factors (c_1 and c_2) for updating the velocity of each particle. Most of the time, c_1 and c_2 both are equal to 2. But still we can change these learning factors and there is no any specific range of that.

Stopping conditions in PSO are somewhat equal to the stopping conditions in GA. Anyhow, PSO algorithm is stopped when it meets any of following condition.

- Maximum number of iterations – The maximum number of iterations can be specified before start the PSO algorithm. After PSO runs this amount of iterations, algorithm stops automatically.
- Time limit – An amount of time in seconds can be specified before start the PSO algorithm. After PSO runs this amount of time, algorithm stops automatically.

- Target fitness value – A minimum value for the fitness function (objective function) can be given before start the PSO algorithm. After PSO reaching the target fitness value, algorithm stops automatically
- Stall iterations – A iteration number can be specified before start the PSO algorithm. When there is no improvement in the fitness value between the specified iteration numbers, algorithm stops automatically.
- Stall time limit – An amount of time in seconds can be specified before start the PSO algorithm. When there is no improvement in the fitness value during the given time limit, algorithm stops automatically.

Following flow chart has been given with all the noticeable processes when PSO combine with NNs

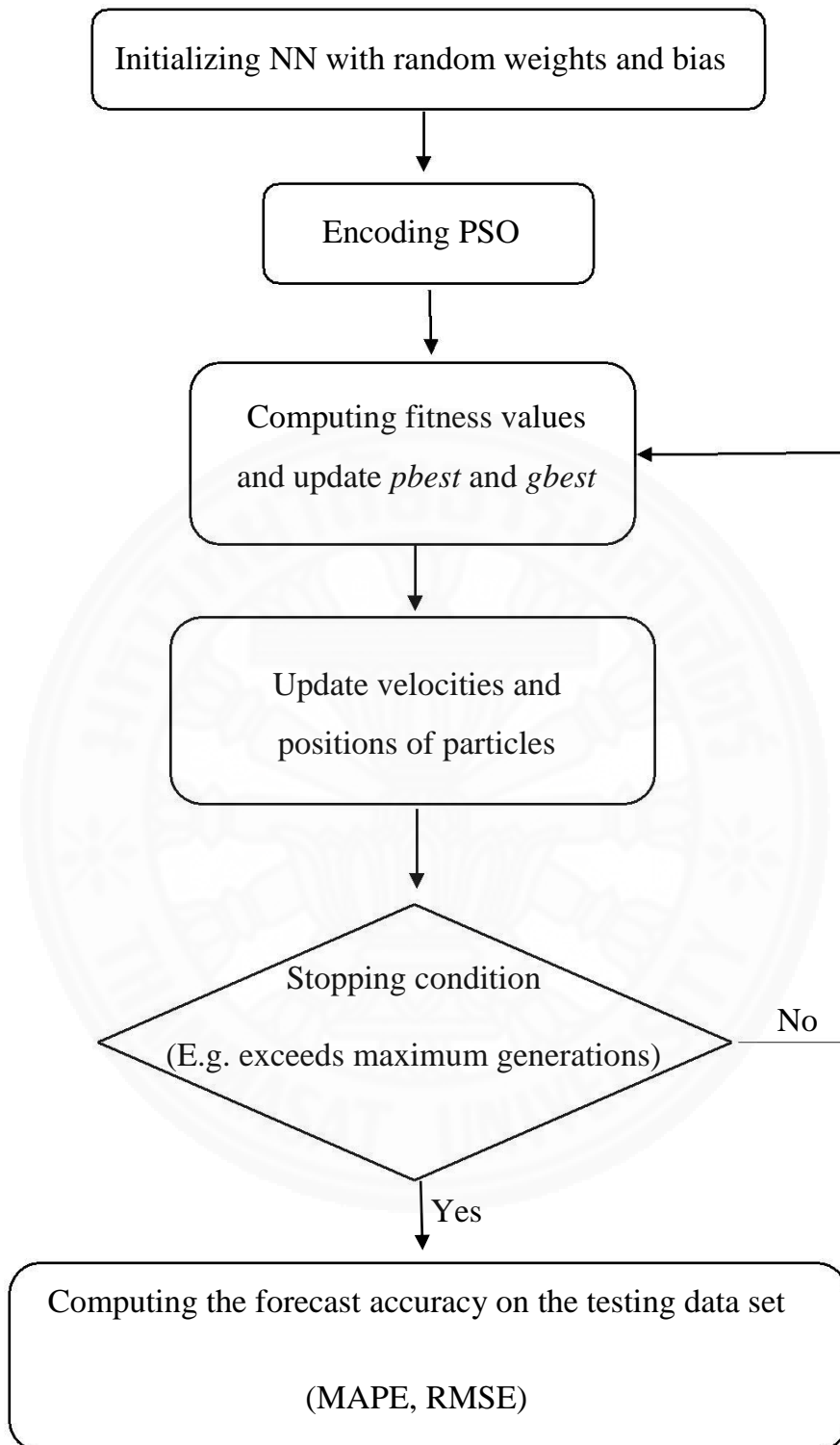


Figure 4.6 Flowchart for hybridizing PSO and NN

4.3.3.4 Training ANN with PSO-GA

However, still PSO shows poor training performances as it can still stick in a local minimum of the solution space. Since all the particles follow its own personal best and the global best particle, there is no way they can get out of local minima in case if they trap in. As a solution to the problem, GA operations (crossover and mutation) which are used to produce the next generation are applied in PSO. The strategy use in this training algorithm is, the time we need to apply these GA operations is decided by the stall generations. When there is no reduction or only a minor improvement of the fitness value given over 20 generations (20 stall generations), it assumes that PSO already has reached to the global minimum or has stuck in a local minimum. Since there is no way of finding whether PSO has reached a local minimum or the global minimum, whenever the number of stall generation reaches to 20 for one training round, crossover and mutation functions are applied. After the 1st attempt of applying these GA operations, the count of the stall generations becomes zero and start over. Likewise, crossover and mutation functions are applied two times per one solution.

Since the size of the population is 100, fifty mutation and fifty crossover children are produced when the number of stall generation reaches to 20. Fifty particles from the last generation are selected randomly for producing the mutation children. Random values between -1 and 1 are added to the random 20% of the genes of each selected particle. For producing a crossover child, the same method applied in GA is used where a new particle is created using the first 12 genes of a randomly selected particle from the previous generation and remaining 13 genes are selected from another randomly selected particle.

This helps PSO to create completely different particle set while they keep the previous personal best and global best values. In case if the newly created particle can beat the current personal best or the global best, these values are updated and carry to the next generation. After applying these GA operations, PSO starts performing its regular activities for finding the global best. The complete flowchart of the new training algorithm (GA-PSO) is given below.

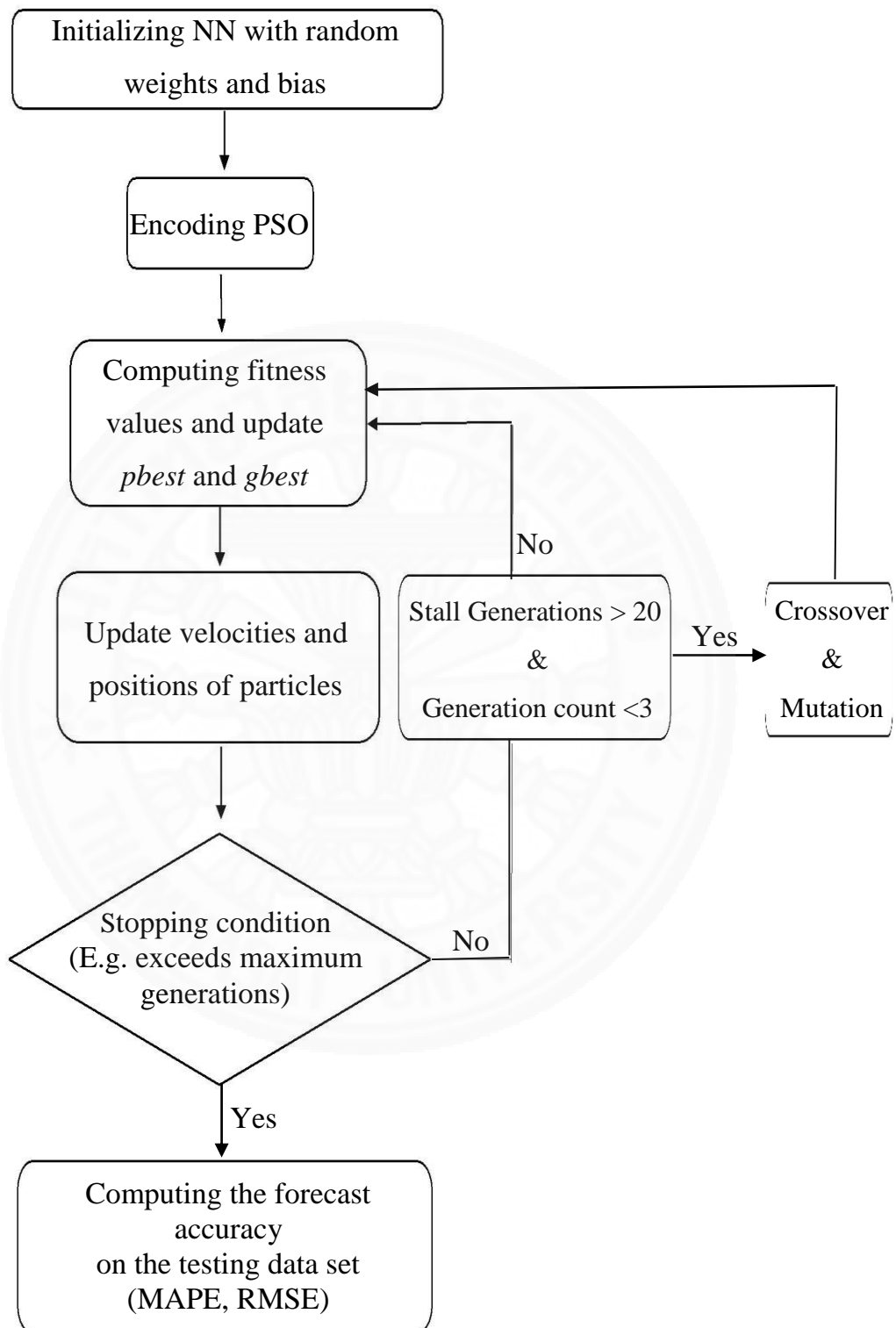


Figure 4.7 Flowchart for hybridizing PSO and GA

Chapter 5

Results and Discussion

5.1 Case 1: Study of Neural Network's Parameters

The average MAPEs for the given three training data sets are summarized in Table 5.1. All these are average MAPE values after running 5 times for each case.

Table 5.1 Average of 5 replications MAPEs with different number of inputs and different NN settings

No of Neurons	MAPE with 5-inputs		MAPE with 10-inputs		MAPE with 20-inputs	
	1 layer	2 layers	1 layer	2 layers	1 layer	2 layers
1	3.8414	3.8411	3.3302	3.3874	2.6279	2.6279
3	3.5747	3.6512	3.2174	3.2426	2.3879	2.4508
5	3.4557	3.5233	3.1883	3.2199	2.3048	2.4210
10	3.5378	3.5285	3.1750	3.2012	2.2886	2.3573
15	3.4751	3.4892	3.3409	3.2819	2.2325	2.2775
20	3.5161	3.5411	3.2308	3.3456	2.2453	2.2641
30	3.5249	3.3411	3.3452	3.3520	2.2239	2.2975
40	3.4575	3.3946	3.3160	3.3480	2.1856	2.2610

Average MAPE values are decreased from 5 inputs column to 20 inputs column with the effect of the number of inputs to the network. However, MAPE value at each column starts with a large number and suddenly reduces with increases of neurons. Then, at one point, the reduction starts to be stable. Anyhow, the minimum MAPE for 5 inputs case with 1 layer is given with 5 neurons and it is equal to 3.4557. But 5 inputs case with 2 layers has a different pattern. Its minimum MAPE comes with 30 neurons.

For the 10 inputs case, the minimum MAPE for both 1 layer and 2 layers come with 10 neurons. Those values are 3.1750 and 3.2012, respectively. For the case with 20 inputs, the NN obtains its minimum MAPEs with 40 neurons and equal to 2.1856 and 2.2610 for both 1 layer and 2 layers, respectively. When consider all the MAPEs, the minimum value is given with 20 inputs case with the number of layers equal to 1

and the number of neurons equal to 40. There is no any clear difference between MAPE values of one layer and two layers for each given data set. These MAPE variations are conclusively illustrated by Figure 5.1 and 5.2.

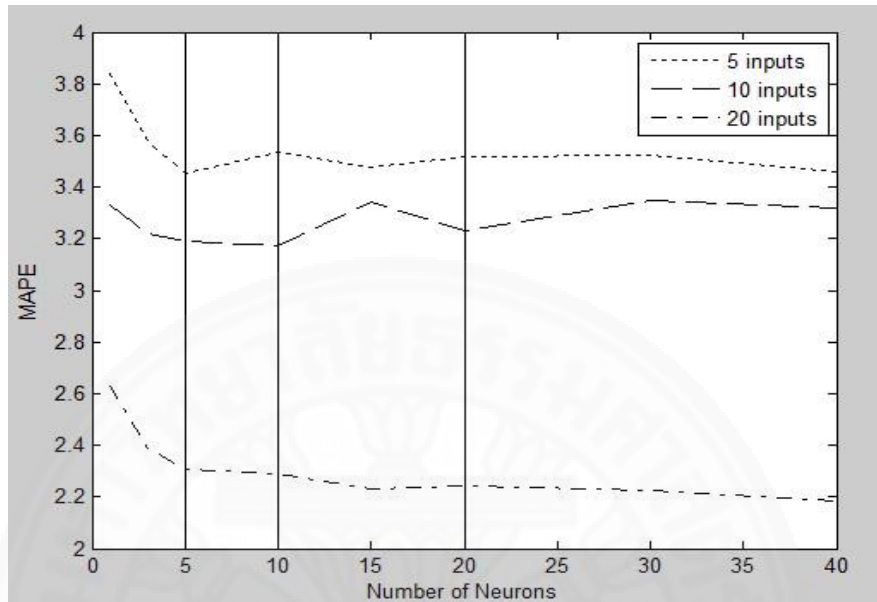


Figure 5.1 MAPEs for three cases with different number of neurons with 1 layer

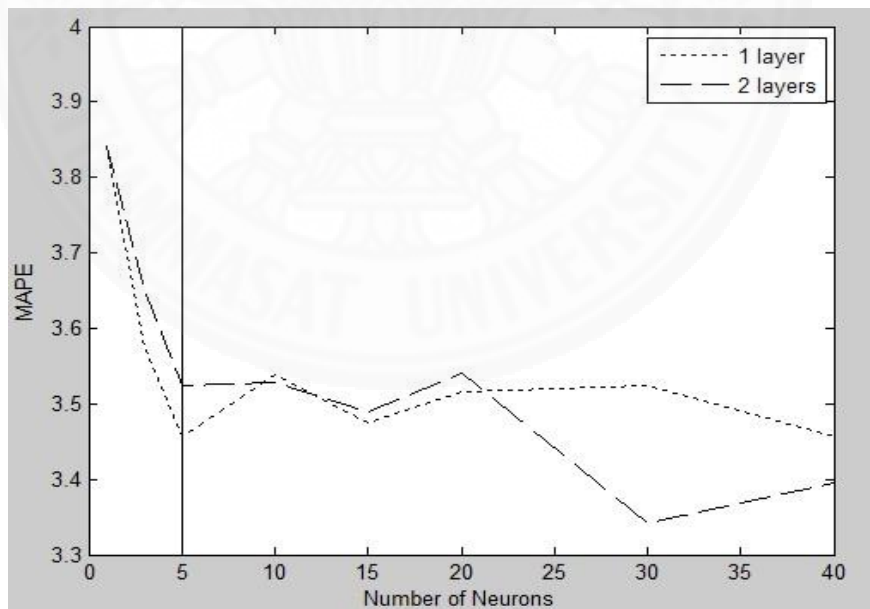


Figure 5.2 MAPEs with 5 input case for 1 layer and 2 layers

As illustrated by Figure 5.1, MAPEs are reduced while increasing the number of neurons. For 5 inputs case, MAPEs suddenly reduce from the start and change the rate at 5 neurons. For 10 inputs case and 20 inputs case also, MAPEs are changed their

reduction rates at 10 and 20 neurons, respectively. After those points, all MAPEs are tend to be stable along with x-axis. Figure 5.2 shows MAPE variations for the 5-inputs case with 1 layer and 2 layers. Both lines suddenly reduce from the starting point and change the reduction rate at 5 neurons. Then MAPEs of 1-layer tend to be stable while the MAPEs of 2-layers still fluctuate even after 5 neurons. But there is no evident relationship between 1 layer and 2 layers from Figure 5.2.

MAPE values and computational times are recorded for each transfer function as given in Table 5.2. These transfer functions are used inside neurons and other conditions such as, the number of hidden layers, number of neurons, and the number of epochs are the same for all cases. MAPEs and processing times (in minutes) are also average values of five tests. These MAPEs are also different for each run due to the random weight initialization. The lowest average MAPE of 2.3879 is given with Tan-Sigmoid Transfer Function (tansig). Satlin Transfer Function (satlin) also performs almost same as with tansig. Its MAPE is 2.4029. When consider about computational time, except Log-Sigmoid Transfer Function (Logsig), other three functions have almost same computational time which are less compared to the computational time of Logsig.

Table 5.2 MAPEs and computational times for different transfer functions

Transfer Function	Average MAPE	Average Computational Time (minutes)
Tan-Sigmoid Transfer Function (tansig)	2.3879	4.22 min
Log-Sigmoid Transfer Function (Logsig)	2.5428	5.54 min
Satlin Transfer Function (satlin)	2.4029	4.42 min
Linear Transfer Function (purelin)	2.6918	4.44 min

5.2 Case 2: Effect of the Amount of Training Data and Number of Inputs

Average MAPE values for all weekdays in November 2013 are summarized in Table 5.3. Average values for each column and for each row have been calculated to summarize the results for all the 20 cases. Average values of each column give an idea that how these MAPEs vary with the different amount of training data sets. Column 1 to column 3, MAPE values decrease as increase of training data to the network. But the average MAPE value of last column which is for 10 months of data to train the network has a higher MAPE value than other cases. As it uses 10 months of data to train the network, from January to October data are included in training data sets. Therefore, some awkward patterns of demands due to seasonal variation over the year appear in training data. But once it uses 6 months or less amount of data to train the network, MAPEs comparatively low to indicate that the most recent data are applicable to train a Neural Network. However, 6 months of data to train the network has the lowest MAPE column average.

Table 5.3 Average MAPE values for Load Forecasting of Weekdays in November for all 20 Cases

		Training Data Size (months)				Average of each row
		2	4	6	10	
Number of Inputs	1	4.10	4.01	4.03	4.08	4.06
	5	3.50	3.14	3.03	3.32	3.23
	10	3.32	3.07	3.00	3.15	3.14
	15	3.21	3.00	3.02	3.24	3.12
	20	2.18	2.59	2.56	2.76	2.52
Average of each Column		3.26	3.16	3.13	3.31	

Average values of each row show how these number of inputs effect on electricity forecasting accuracy. Average MAPEs of rows are reduced when the number of inputs are increase to train the network. When the network has higher number of inputs, or when inputs are increased to train a network, it positively effects on forecasting accuracy and on MAPE values.

The best MAPE value from all 20 cases has obtained when it uses 2 months of training data with 20 inputs. Looking at the 2-month training input data column, it matches with the existing pattern which MAPEs decrease as increase of the number of inputs. But for the row which uses different amount of training data with 20 inputs does not have a specific pattern and this MAPE value is the minimum for that row as well.

Figure 5.3 and Figure 5.4 show actual and forecasted series for two selected sample cases. Forecasted series of Figure 5.3 has been obtained by giving 2 months of training data and 1 input for the NN.

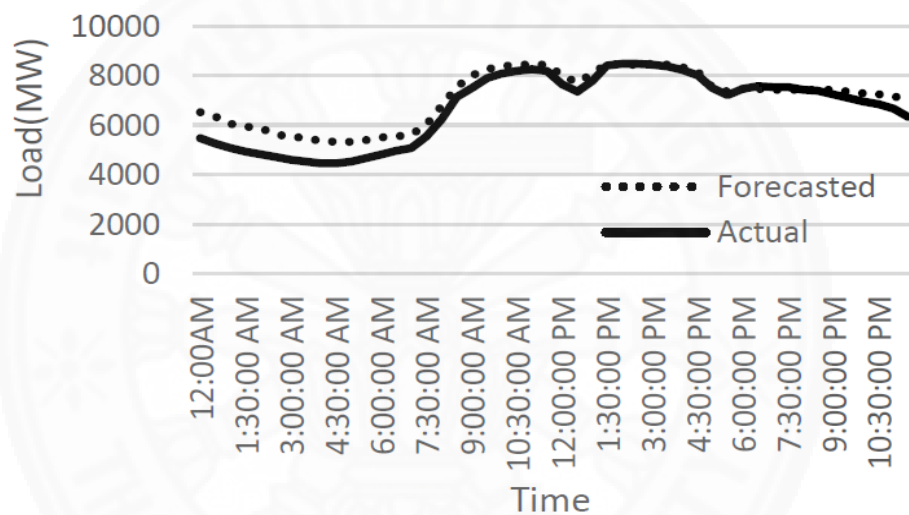


Figure 5.3 Actual and Forecasted series with 2 months training data and one input

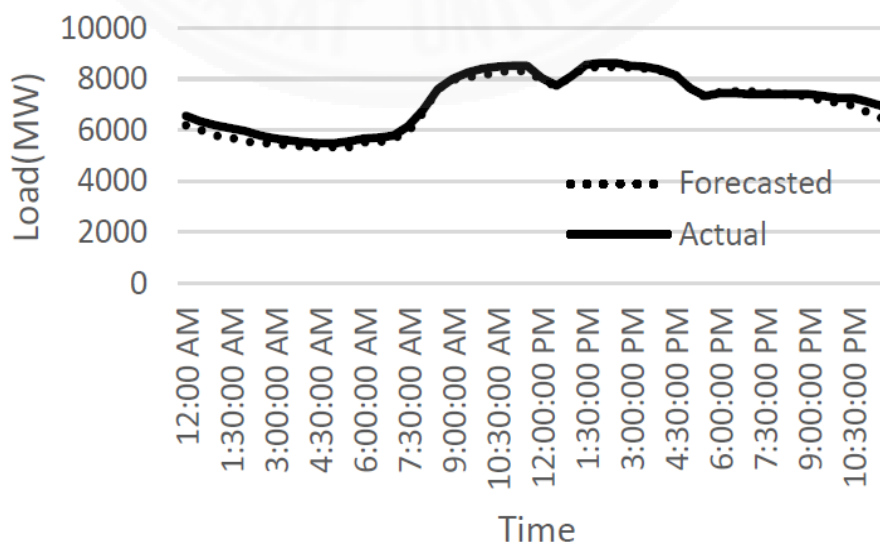


Figure 5.4 Actual and Forecasted series with 10 months training data and 20 inputs

Forecasted series of Figure 5.4 has been obtained with 10 months of training data and 20 inputs for NN. The thin line in both graphs are to indicate the actual demand series and the dotted line for the forecasted series for that specific day. For these two specific cases, forecasted series with more training data and more inputs is greatly aligned on the actual series.

5.3 Case 3: Performances of Different Training Algorithms

Since the experiment was conducted with the cleaned data, $MAPE^p$ are given in the following tables (Table 5.4, Table 5.5, and Table 5.6) instead of $MAPE$. Table 5.4 gives the monthly average $MAPE^p$ for different training algorithm which used to train the neural network. Average for each column has been calculated to get the average $MAPE^p$ for year 2013 by different training algorithms. Average yearly $MAPE^p$ helps to select the best training algorithm to train the neural networks for forecasting short term load demand. Therefore, considering the yearly average $MAPE^p$, the training algorithm which combines genetic algorithm and particle swarm optimization methods performs well compared to the other training algorithm. The yearly average $MAPE^p$ with the hybridized (PSO+GA) algorithm is 3.074 and which is the smallest $MAPE^p$ compared to other yearly $MAPE^p$: 3.113 with genetic algorithm, 3.439 with particle swarm optimization, and 3.769 with backpropagation.

The minimum and the maximum monthly average $MAPE^p$ with the hybridized algorithm are 2.430 (April) and 6.872 (December) respectively. These values when genetic algorithm use to train the neural network are 2.439 (September) and 7.069 (December). The minimum (2.554) and maximum (8.637) monthly average $MAPE^p$ with particle swarm optimization training algorithm belong to November and December, respectively. The backpropagation training algorithm which gives the highest yearly average $MAPE^p$ has its minimum monthly average $MAPE^p$ in August (2.541) and the maximum monthly average $MAPE^p$ in December (8.490).

Table 5.4 Monthly (2013) average $MAPE^p$ for different training algorithms

	Monthly average $MAPE^p$			
	PSO+GA	GA-NN	PSO-NN	BP-NN
January	2.872	2.927	3.139	3.837
February	2.998	3.038	3.215	3.480
March	3.106	3.104	3.492	4.253
April	2.430	2.494	2.604	3.398
May	2.628	2.639	2.938	3.416
June	3.050	3.106	3.200	3.591
July	2.624	2.617	2.932	3.173
August	2.435	2.454	2.574	2.541
September	2.414	2.439	2.611	2.653
October	2.925	2.929	3.257	3.388
November	2.465	2.471	2.554	2.914
December	6.872	7.069	8.637	8.470
Average	3.074	3.113	3.439	3.769

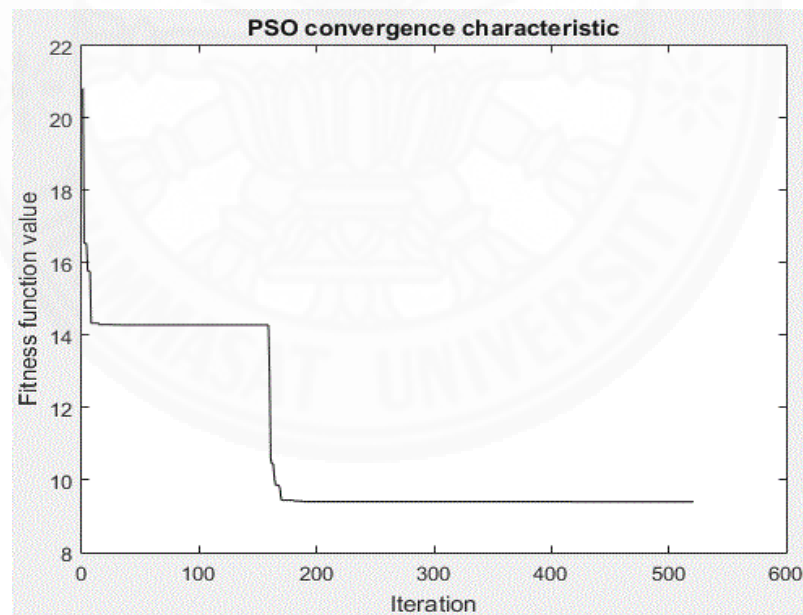


Figure 5.5 Performances of the hybridized training algorithm

The reason that the hybridized training function performs well is, it helps to further reduce the training error (MSE) when particle swarm optimization algorithm traps in local minima. As it uses the breeding functions of genetic algorithm, the hybridized training algorithm has qualities from both genetic algorithm and particle

swarm optimization. Figure 5.5 has the graph of a sample objective function against iterations while it minimizes the objective value using the hybridized algorithm. The fitness or the objective value suddenly drops down to a local minimum (14.2857) and stay there till the functions of genetic algorithm are applied. That helps to reduce the fitness value from 14.2857 to 9.7397 and assume that it has already reached to the global minimum.

Even though the minimum monthly average $MAPE^p$ by each training algorithm comes from different months, the maximum monthly average $MAPE^p$ by all training algorithms belong to December. The reason for this is, although holidays, bridging holidays, and outliers from the sample data set are removed and replaced with the estimated data, still the load consumption in December is much lower compared to the other months. When the neural network is trained with higher consumption values from other months, it forecasts assuming December also has similar consumption values. Therefore, any training algorithm was unable to get good forecasting results for December. An example day (23rd Monday) in December is given by Figure 5.6 to get the idea that forecasted results are much higher compared to the actual consumption ($L_t^p(d)$).

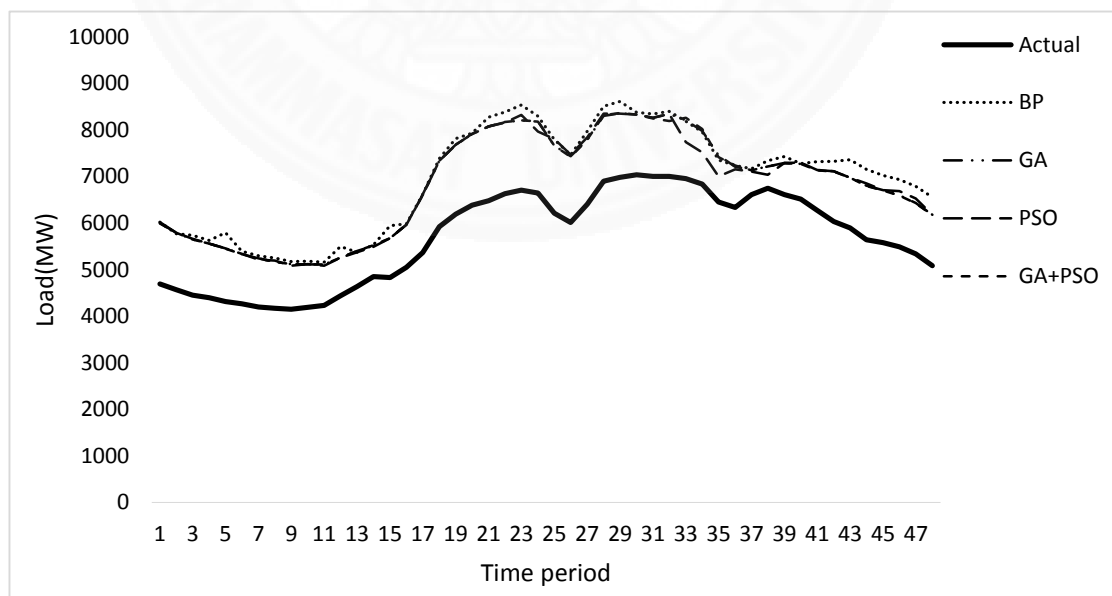


Figure 5.6 Actual vs Forecasted load with different training algorithms for 23rd Monday of December, 2013

Except March, June, and December with the hybridized training algorithm, except February, March, June, and December with genetic algorithm, and except January, February, March, June, October, and December with particle swarm optimization, average monthly $MAPE^p$ of all the other months are less than three. Only August, September, and November give less than three $MAPE^p$ values with backpropagation. Regardless of the training algorithm, average monthly $MAPE^p$ in December are higher than six with all training algorithms. However, average monthly $MAPE^p$ in August, September, and November are less than three with all training algorithm. Figure 5.7 brings forecasting outcomes by all three training algorithms on a day which has good $MAPE^p$ that is selected from one of the above months (25th September, 2013)

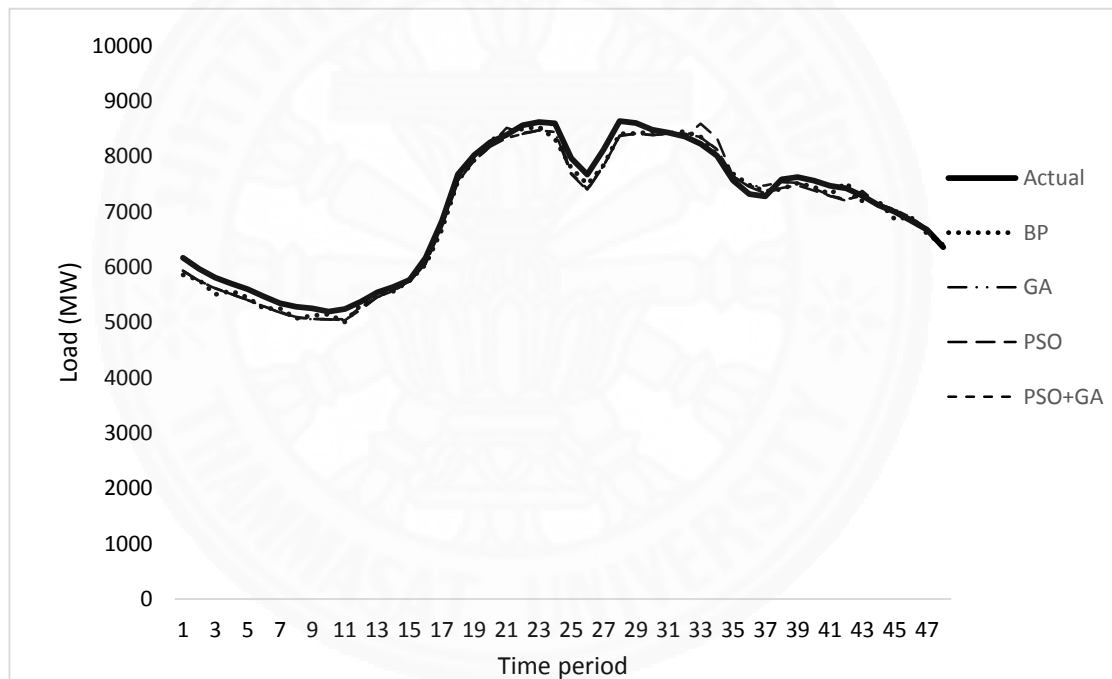


Figure 5.7 Actual vs Forecasted load with different training algorithms for 25th Wednesday of September, 2013

Table 5.5 and Table 5.6 summarize the average monthly and yearly $MAPE^p$ by different training algorithms for different categories that can be categorized based on the electricity consumption patterns on those days: Weekdays, Mondays, Weekends, Holidays, and Bridging holidays. Even though genetic algorithm performs almost same as the hybridized training algorithm, the best yearly average $MAPE^p$ for all the

categories come with the hybridized training algorithm. Nevertheless, the maximum yearly average $MAPE^p$ for all the categories are belong to backpropagation training algorithm. However, the minimum yearly average $MAPE^p$ for Weekdays, Mondays, Weekends, Holidays, and Bridging holidays are 2.826, 3.150, 3.222, 3.555, and 3.777 respectively.

Table 5.5 Yearly average $MAPE^p$ for different categories of days by different training algorithms

	Yearly average $MAPE^p$			
	PSO+GA	GA	PSO	BP
Weekdays	2.826	2.856	3.145	3.317
Mondays	3.150	3.176	3.322	3.666
Weekends	3.222	3.254	3.491	4.047
Holidays	3.555	3.678	4.554	5.262
Bridging Holidays	3.777	3.837	4.407	4.795

Table 5.6 Monthly average $MAPE^p$ for different categories of days by different training algorithms

	Monthly average $MAPE^p$			
	PSO+GA	GA	PSO	BP
January				
Weekdays	2.037	2.058	2.249	2.966
Mondays	2.592	2.652	2.860	3.773
Weekends	4.916	5.082	5.305	5.399
Holidays	5.605	5.667	5.432	6.051
Bridging Holidays	1.929	1.941	2.314	3.394

	Monthly average $MAPE^p$			
	PSO+GA	GA	PSO	BP
February				
Weekdays	2.544	2.551	2.763	2.697
Mondays	2.643	2.631	2.798	3.473
Weekends	4.108	4.236	4.378	5.092
Holidays				
Bridging Holidays				

	Monthly average $MAPE^p$			
	PSO+GA	GA	PSO	BP
March				
Weekdays	2.816	2.824	3.093	3.345

Mondays	2.596	2.445	2.586	2.503
Weekends	3.610	3.623	4.340	6.077
Holidays				
Bridging Holidays				

April	PSO+GA	GA	PSO	BP
Weekdays	2.264	2.259	2.602	3.111
Mondays	5.123	5.330	5.245	5.513
Weekends	1.898	1.934	2.003	2.858
Holidays	1.947	2.107	2.152	3.349
Bridging Holidays	1.707	1.689	1.608	2.780

May	PSO+GA	GA	PSO	BP
Weekdays	2.838	2.876	3.182	3.702
Mondays	2.081	2.077	2.133	2.581
Weekends	2.416	2.461	2.952	3.407
Holidays	2.949	2.862	2.996	3.643
Bridging Holidays	1.837	1.810	2.129	2.185

June	PSO+GA	GA	PSO	BP
Weekdays	3.268	3.366	3.450	3.893
Mondays	2.613	2.655	2.867	3.097
Weekends	2.878	2.870	2.934	3.305
Holidays				
Bridging Holidays				

July	PSO+GA	GA	PSO	BP
Weekdays	2.699	2.706	3.146	3.222
Mondays	1.890	1.813	1.765	1.900
Weekends	2.691	2.684	2.860	3.559
Holidays	2.783	2.751	3.040	3.092
Bridging Holidays				

August	PSO+GA	GA	PSO	BP
Weekdays	2.760	2.774	2.935	2.819
Mondays	1.421	1.530	1.884	2.000
Weekends	2.086	2.059	2.054	2.117
Holidays	2.772	3.002	2.817	2.988

Bridging Holidays				
-------------------	--	--	--	--

September	PSO+GA	GA	PSO	BP
Weekdays	2.430	2.494	2.572	2.400
Mondays	1.933	1.942	2.129	2.352
Weekends	2.653	2.616	2.948	3.270
Holidays				
Bridging Holidays				

October	PSO+GA	GA	PSO	BP
Weekdays	2.813	2.819	3.252	3.294
Mondays	3.032	3.041	3.162	3.771
Weekends	3.153	3.145	3.283	3.491
Holidays	2.707	2.747	3.499	2.747
Bridging Holidays				

November	PSO+GA	GA	PSO	BP
Weekdays	2.204	2.207	2.317	2.531
Mondays	1.497	1.476	1.563	1.787
Weekends	3.388	3.411	3.443	4.139
Holidays				
Bridging Holidays				

December	PSO+GA	GA	PSO	BP
Weekdays	6.373	6.541	8.183	7.581
Mondays	9.176	9.323	10.708	9.948
Weekends	4.811	4.956	5.381	5.631
Holidays	5.781	6.105	9.332	10.102
Bridging Holidays	8.898	9.128	10.860	9.743

While using the hybridized training algorithm, the minimum (2.037) and the maximum (6.373) weekdays' $MAPE^p$ come from January and December, respectively. These values are equal to 2.058 (January) and 6.541 (December) when the neural network uses genetic algorithm for training. The minimum and the maximum weekdays' $MAPE^p$ with particle swarm optimization are belong to January (2.249) and December (8.183) while these values with backpropagation are in September (2.400) and December (7.581). Regardless of the training algorithm, the maximum weekdays'

$MAPE^p$ always recorded in December. However, except with backpropagation training algorithm, the minimum weekdays' $MAPE^p$ by all training algorithms are belong to January.

Most of the time, the monthly average $MAPE^p$ for Mondays are different compared to the weekday's $MAPE^p$. They can be higher or lower than weekdays' $MAPE^p$ in that month with that specific training algorithm. However, the minimum and the maximum monthly average $MAPE^p$, which given by the hybridized training algorithm, are belong to August (1.421) and December (9.176), respectively. The minimum monthly average Mondays' $MAPE^p$ by other three individual training algorithms are in November and those are equal to 1.476, 1.563, and 1.787. December takes the highest monthly average Mondays' $MAPE^p$ for the individual training algorithms and those values are 9.323, 10.708, and 9.948, respectively.

All the Saturdays and Sundays of each month are categorized as weekends. This is a different category as they have different demand variation than the normal weekdays. The hybridized algorithm gives its minimum monthly average weekends' $MAPE^p$ in April (1.898) and the maximum monthly average weekday's $MAPE^p$ in January (4.916). These values with genetic algorithm own by April (1.934) and January (5.082). The minimum and the maximum monthly average weekends' $MAPE^p$ are in April (2.003) and December (5.381) when the neural network uses particle swarm optimization to adjust its weights and bias. However, backpropagation has its minimum monthly average weekends' $MAPE^p$ in August (2.117) while the maximum belongs to March (6.077). Therefore, all the training algorithms except backpropagation give their minimum monthly average weekends' $MAPE^p$ in April while sharing the maximum by different months.

Holidays and Bridging holidays appear only in several months. While no holidays are recorded in February, March, June, September, and November, Bridging holidays are recorded only in January, April, May, and December. The hybridized algorithm gives its minimum monthly average $MAPE^p$ for Holidays and Bridging

holidays in April (1.947 and 1.707) while the maximums are in December (5.781 and 8.898). Genetic algorithm also has its minimum and maximum monthly average $MAPE^p$ for Holidays and Bridging holidays in the same months: minimums are in April (2.107 and 1.689) and maximums are in December (6.105 and 9.128). When, also the above months are common for particle swarm optimization, backpropagation has its minimum monthly average $MAPE^p$ for Holidays in October (2.747) and for Bridging holidays in May (2.185). However, still the maximum monthly average $MAPE^p$ for Holidays and Bridging holidays by backpropagation are recorded in December (10.102 and 9.743).

Most of the time, minimum monthly average of each category of each month comes with the hybridize algorithm. These minimum average $MAPE^p$ are somewhat close to the minimum $MAPE^p$ by genetic algorithm. However, minimum averages $MAPE^p$ by particle swarm optimization and backpropagation are still higher compared to the minimum average by hybridized and genetic algorithm at most of the occasions.

Chapter 6

Conclusions and Recommendations

6.1 Case 1: Study of Neural Network's Parameters

There are two main objectives under this experiment. The first one is to find relationships of the number of inputs in training data set and the number of hidden layers with the number of neurons when NNs are used in electricity forecasting. The other one is to find a good transfer function to use in neurons while training and testing NNs for electricity forecasting. MAPEs are higher with small numbers of neurons in hidden layers. But it is suddenly reduced while the neurons are increased until certain numbers which equal to the number of inputs to the network. When neurons increase even after that point, most of the time, MAPEs are gradually decreased. But in some cases, MAPEs are increased while increasing of neurons. Therefore, we can conclude that, to get an accurate forecast result using NNs, the number of neurons in hidden layers should be at least equal to the number of inputs to the network. Additionally, the number of layers do not make any difference in the performance of NNs.

Tan-Sigmoid Transfer Function (tansig) and Satlin Transfer Function (satlin) have lowest MAPEs which are 2.3879 and 2.4029 respectively. Computational times are also good with that Tan-Sigmoid and Satlin Transfer Functions and equal to 4.22 and 4.42 minutes respectively. Therefore, we can conclude that, to have accurate and quick results with NNs, Tan-Sigmoid and Satlin Transfer Functions are better to use in hidden layer neurons. However, this experiment is done only with two hidden layers. But the same experiment can be extended to check the accuracy variations with 3 or more hidden layers. As well as, other than these commonly used transfer functions, there are many transfer functions available to use in neurons. Therefore, this experiment may be extended to check the performance, and computational time of more different transfer functions with electricity data.

6.2 Case 2: Effect of the Amount of Training Data and Number of Inputs

The objective of this experiment is to find how the electricity load forecast accuracy varies with different amounts of data and inputs to train a Neural Network. The results show that MAPEs decrease while increasing the number of inputs to the network. Therefore, to have an accurate forecast series, the number of inputs to the Neural Network have to be increased. While increasing the amount of data to train the network, MAPE values decrease. Therefore, an increase of historical data to train the network has a positive impact on forecast accuracy since it reduces MAPE values. But it has to be done carefully, by thinking about demand variations due to seasonal effects. Therefore, final conclusion is, for an accurate forecast series, training data can be increased and have to be selected from recent months.

Slightly less accurate results have been given by the NN while it uses 10 months of data to train the network. The same experiment can be done by replacing the same year different seasonal data by the same seasonal data of the previous year. Finally, testing of this experiment is done by using 2013 November data. This same experiment can be used to test other seasonal months.

6.3 Case 3: Performances of Different Training Algorithms

The objective of the experiment was to identify the best way of training ANN for getting good forecasting outcomes by minimizing the training error during the training process. Due to the limitations of using the backpropagation training algorithm, where it can trap in local minima, two stochastic optimization techniques (Genetic Algorithm and Particle Swarm Optimization) are suggested for further reducing the training error. Another training algorithm is created using the properties of both Genetic Algorithm and Particle Swarm Optimization techniques. The concept was initiated with the bad training performances of Particle Swarm Optimization technique where it cannot get out of the local minima in case if they trap in one. When they stick in a local minimum, crossover and mutation functions in Genetic Algorithm change the positions

of each particle in the solution space and help to get out from the local minimum. These theoretical concepts have been proved with the arranged experiments for Short Term Load Forecasting with Thailand's electricity consumption data. According to the daily forecasting results for 2013 with the suggested neural network, the best training algorithm to get the good forecasting outcome is the hybridized training algorithm which use the properties of both genetic algorithm and particle swarm optimization. Genetic algorithm also gives competitive results compared to the hybridized training algorithm. However, Particle Swarm Optimization and backpropagation training algorithms are fairly good compared to the other two training algorithms.

Regardless of the training algorithm, the hardest month to forecast is December. Due to the low electricity consumption in December, all the forecasting outcomes are higher than the actual electricity consumption. Other than December, March and June also hard to forecast according to the monthly average $MAPE^p$ of those months.

Days of the sample set are categorized into four sections based on the patterns of the consumption curves. Out of them, forecasting weekdays except Mondays are quite easier compared to forecasting Mondays, weekends, Holidays, and Bridging holidays.

Therefore, the research can be further extended to get good forecasting outcomes for other categories: Monday forecasting, weekend forecasting, holiday forecasting, and bridging holiday forecasting. Also, some strategies have to be used for forecasting the demands in December. These could be achieved by creating new training algorithms, new neural network arrangements, and/or with new data arrangement which help to capture the seasonal variations of data.

References

1. Akole, M., & Tyagi, B. (2009, December). Artificial neural network based short term load forecasting for restructured power system. *In Power Systems, 2009. ICPS'09. International Conference on (pp. 1-7). IEEE.*
2. Bagnasco, A., Fresi, F., Saviozzi, M., Silvestro, F., & Vinci, A. (2015). Electrical consumption forecasting in hospital facilities: An application case. *Energy and Buildings, 103*, 261-270.
3. Bhurtun, C., Jahmeerbacus, I., & Jeewoath, C. (2011, August). Short term load forecasting in Mauritius using Neural Network. *In Industrial and Commercial Use of Energy (ICUE), 2011 Proceedings of the 8th Conference on the (pp. 184-191). IEEE.*
4. Brockwell, P. J., & Davis, R. A. (2016). Introduction to time series and forecasting. *springer.*
5. Caiqing, Z., Ming, L., & Mingyang, T. (2008, December). BP neural network optimized with PSO algorithm for daily load forecasting. *In Information Management, Innovation Management and Industrial Engineering, 2008. ICIII'08. International Conference on (Vol. 3, pp. 82-85). IEEE.*
6. Castelli, M., Vanneschi, L., & De Felice, M. (2015). Forecasting short-term electricity consumption using a semantics-based genetic programming framework: the South Italy case. *Energy Economics, 47*, 37-41.
7. Che, J. (2014). A novel hybrid model for bi-objective short-term electric load forecasting. *International Journal of Electrical Power & Energy Systems, 61*, 259-266.
8. Chiang, C. C., Ho, M. C., & Chen, J. A. (2006). A hybrid approach of neural networks and grey modeling for adaptive electricity load forecasting. *Neural Computing & Applications, 15*(3-4), 328-338.
9. Chitsaz, H., Shaker, H., Zareipour, H., Wood, D., & Amjady, N. (2015). Short-term electricity load forecasting of buildings in microgrids. *Energy and Buildings, 99*, 50-60.
10. Clements, A. E., Hurn, A. S., & Li, Z. (2016). Forecasting day-ahead electricity

load using a multiple equation time series approach. *European Journal of Operational Research*, 251(2), 522-530.

11. De Felice, M., Alessandri, A., & Ruti, P. M. (2013). Electricity demand forecasting over Italy: Potential benefits using numerical weather prediction models. *Electric Power Systems Research*, 104, 71-79.
12. Ding, N., Bésanger, Y., & Wurtz, F. (2015). Next-day MV/LV substation load forecaster using time series method. *Electric Power Systems Research*, 119, 345-354.
13. Dudek, G. (2015). Pattern similarity-based methods for short-term load forecasting—Part 1: *Principles*. *Applied Soft Computing*, 37, 277-287.
14. Energy Policy and Planning Office, M. o. (2012). *THAILAND POWER DEVELOPMENT PLAN*. Bangkok, Thailand.
15. Friedrich, L., & Afshari, A. (2015). Short-term forecasting of the Abu Dhabi electricity load using multiple weather variables. *Energy Procedia*, 75, 3014-3026.
16. Gajowniczek, K., & Ząbkowski, T. (2014). Short term electricity forecasting using individual smart meter data. *Procedia Computer Science*, 35, 589-597.
17. Ghareeb, W. T., & El Saadany, E. F. (2013, August). A hybrid genetic radial basis function network with fuzzy corrector for short term load forecasting. *In Electrical Power & Energy Conference (EPEC), 2013 IEEE (pp. 1-5)*. IEEE.
18. Haifeng, L., & Gengyin, L. (2004, April). An adaptive BP-network approach to short term load forecasting. *In Electric Utility Deregulation, Restructuring and Power Technologies, 2004. (DRPT 2004). Proceedings of the 2004 IEEE International Conference on (Vol. 2, pp. 505-509)*. IEEE.
19. Haque, A. U., Mandal, P., Meng, J., & Pineda, R. L. (2012). Performance evaluation of different optimization algorithms for power demand forecasting applications in a smart grid environment. *Procedia Computer Science*, 12, 320-325.
20. Hippert, H. S., & Taylor, J. W. (2010). An evaluation of Bayesian techniques

for controlling model complexity and selecting inputs in a neural network for short-term load forecasting. *Neural networks*, 23(3), 386-395.

21. Hippert, H. S., Pedreira, C. E., & Souza, R. C. (2001). Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on power systems*, 16(1), 44-55.
22. Honghui, Z., & Yongqiang, L. (2012, March). Application of an Adaptive Network-Based Fuzzy Inference System Using Genetic Algorithm for Short Term Load Forecasting. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on (Vol. 2, pp. 314-317)*. IEEE.
23. Hu, Z., Bao, Y., Xiong, T., & Chiong, R. (2015). Hybrid filter–wrapper feature selection for short-term load forecasting. *Engineering Applications of Artificial Intelligence*, 40, 17-27.
24. Ismail, Z., Jamaluddin, F., & Jamaludin, F. (2008). Time series regression model for forecasting Malaysian electricity load demand. *Asian Journal of Mathematics and Statistics*, 1(3), 139-149.
25. Ismail, Z., Yahya, A., & Mahpol, K. A. (2009). Forecasting peak load electricity demand using statistics and rule based approach. *American Journal of Applied Sciences*, 6(8), 1618.
26. Jaramillo-Morán, M. A., González-Romera, E., & Carmona-Fernández, D. (2013). Monthly electric demand forecasting with neural filters. *International Journal of Electrical Power & Energy Systems*, 49, 253-263.
27. Kandananond, K. (2011). Forecasting electricity demand in Thailand with an artificial neural network approach. *Energies*, 4(8), 1246-1257.
28. Khwaja, A. S., Naeem, M., Anpalagan, A., Venetsanopoulos, A., & Venkatesh, B. (2015). Improved short-term load forecasting using bagged neural networks. *Electric Power Systems Research*, 125, 109-115.
29. Li, H. Z., Guo, S., Li, C. J., & Sun, J. Q. (2013). A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. *Knowledge-Based Systems*, 37, 378-387.
30. Li, S., Goel, L., & Wang, P. (2016). An ensemble approach for short-term load

forecasting by extreme learning machine. *Applied Energy*, 170, 22-29.

31. Li, Y., & Liu, K. (2010, January). Core vector regression with particle swarm optimization algorithm in short term load forecasting. *In Computer Modeling and Simulation, 2010. ICCMS'10. Second International Conference on (Vol. 2, pp. 325-329). IEEE.*
32. Lin, C. T., Chou, L. D., Chen, Y. M., & Tseng, L. M. (2014). A hybrid economic indices based short-term load forecasting system. *International Journal of Electrical Power & Energy Systems*, 54, 293-305.
33. Liu, N., Tang, Q., Zhang, J., Fan, W., & Liu, J. (2014). A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids. *Applied Energy*, 129, 336-345.
34. Lou, C. W., & Dong, M. C. (2015). A novel random fuzzy neural networks for tackling uncertainties of electric load forecasting. *International Journal of Electrical Power & Energy Systems*, 73, 34-44.
35. Ma, L., Zhou, S., & Lin, M. (2008, December). Support vector machine optimized with genetic algorithm for short-term load forecasting. *In Knowledge Acquisition and Modeling, 2008. KAM'08. International Symposium on (pp. 654-657). IEEE.*
36. Mishra, S., & Patra, S. K. (2008, December). Short term load forecasting using a neural network trained by a hybrid artificial immune system. *In Industrial and Information Systems, 2008. ICIS 2008. IEEE Region 10 and the Third international Conference on (pp. 1-5). IEEE.*
37. Mishra, S., & Patra, S. K. (2008, July). Short term load forecasting using neural network trained with genetic algorithm & particle swarm optimization. *In Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on (pp. 606-611). IEEE.*
38. Mishra, S., & Patra, S. K. (2009, July). Short term load forecasting using a robust novel Wilcoxon Neural Network. *In Nonlinear Dynamics and Synchronization, 2009. INDS'09. 2nd International Workshop on (pp. 1-7). IEEE.*

39. Nadtoka, I. I., & Balasim, M. A. Z. (2015). Mathematical Modelling and Short-term Forecasting of Electricity Consumption of the Power System, with Due Account of Air Temperature and Natural Illumination, Based on Support Vector Machine and Particle Swarm. *Procedia Engineering*, 129, 657-663.
40. Nahmias, S., & Olsen, T. L. (2015). Production and operations analysis. *Waveland Press*.
41. Ramakrishna, R., Boiroju, N. K., & Reddy, M. K. (2011). • FORECASTING DAILY ELECTRICITY LOAD USING NEURAL NETWORKS. *International Journal of Mathematical Archive (IJMA) ISSN 2229-5046*, 2(8).
42. Reddy, S. S., & Momoh, J. A. (2014, September). Short term electrical load forecasting using back propagation neural networks. *In North American Power Symposium (NAPS), 2014 (pp. 1-6). IEEE*.
43. Ringwood, J. V., Bofelli, D., & Murray, F. T. (2001). Forecasting electricity demand on short, medium and long time scales using neural networks. *Journal of Intelligent and Robotic Systems*, 31(1-3), 129-147.
44. Singh, N. K., Singh, A. K., & Tripathy, M. (2014, September). Short Term Load Forecasting using genetically optimized Radial Basis Function Neural Network. *In Power Engineering Conference (AUPEC), 2014 Australasian Universities (pp. 1-5). IEEE*.
45. Sinha, N., Lai, L. L., Ghosh, P. K., & Ma, Y. (2007, November). Wavelet-GA-ANN based hybrid model for accurate prediction of short-term load forecast. *In Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on (pp. 1-8). IEEE*.
46. Son, H., & Kim, C. (2015). Forecasting Short-term Electricity Demand in Residential Sector Based on Support Vector Regression and Fuzzy-rough Feature Selection with Particle Swarm Optimization. *Procedia Engineering*, 118, 1162-1168.
47. Srinivasan, D., Liew, A. C., & Chen, J. S. P. (1991, July). Short term forecasting using neural network approach. *In Neural Networks to Power Systems, 1991., Proceedings of the First International Forum on Applications of (pp. 12-16). IEEE*.
48. Sun, C., & Gong, D. (2006). Support vector machines with PSO algorithm for

- short-term load forecasting. *In Networking, Sensing and Control, 2006. ICNSC'06. Proceedings of the 2006 IEEE International Conference on* (pp. 676-680). IEEE.
49. Taylor, J. W. (2010). Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, 204(1), 139-152.
 50. Taylor, J. W., & Buizza, R. (2002). Neural network load forecasting with weather ensemble predictions. *IEEE Transactions on Power Systems*, 17(3), 626-632.
 51. Verbesselt, J., Hyndman, R., Newnham, G., & Culvenor, D. (2010). Detecting trend and seasonal changes in satellite image time series. *Remote sensing of Environment*, 114(1), 106-115.
 52. Vilar, J. M., Cao, R., & Aneiros, G. (2012). Forecasting next-day electricity demand and price using nonparametric functional methods. *International Journal of Electrical Power & Energy Systems*, 39(1), 48-55.
 53. Wang, J. M., & Wang, L. P. (2008). A new method for short-term electricity load forecasting. *Transactions of the Institute of Measurement and Control*, 30(3-4), 331-344.
 54. Wang, J., & Zhou, Y. (2008, June). Short—term electricity load forecasting based on SAPSO-ANN algorithm. *In Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on* (pp. 97-102). IEEE.
 55. Wang, J., Zhou, Y., & Chen, X. (2007, August). Electricity load forecasting based on support vector machines and simulated annealing particle swarm optimization algorithm. *In Automation and Logistics, 2007 IEEE International Conference on* (pp. 2836-2841). IEEE.
 56. Wang, Y., Gu, D., Xu, J., & Li, J. (2009, June). Back propagation neural network for short-term electricity load forecasting with weather features. *In Computational Intelligence and Natural Computing, 2009. CINC'09. International Conference on* (Vol. 1, pp. 58-61). IEEE.
 57. WONGLA, T. (2013). Thailand Power Development Plan (2010-2030). Bangkok, Thailand: *Energy Policy and Planning Office, Ministry of Energy*.

58. Worawit, T., & Wanchai, C. (2002, October). Substation short term load forecasting using neural network with genetic algorithm. *In TENCON'02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering* (Vol. 3, pp. 1787-1790). IEEE.
59. Xiao, L., Wang, J., Hou, R., & Wu, J. (2015). A combined model based on data pre-analysis and weight coefficients optimization for electrical load forecasting. *Energy*, 82, 524-549.
60. Xuan, W., Jiake, L., Chaofu, W., & Deti, X. (2008, October). A hybrid particle swarm optimization neural network approach for short term load forecasting. *In 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*.
61. Yalcinoz, T., & Eminoglu, U. (2005). Short term and medium term power distribution load forecasting by neural networks. *Energy Conversion and Management*, 46(9), 1393-1405.
62. Yue, H., Dan, L., Liqun, G., & Hongyuan, W. (2009, June). A short-term load forecasting approach based on support vector machine with adaptive particle swarm optimization algorithm. *In Control and Decision Conference, 2009. CCDC'09. Chinese* (pp. 1448-1453). IEEE.
63. Zhai, M. Y. (2015). A new method for short-term load forecasting based on fractal interpretation and wavelet analysis. *International Journal of Electrical Power & Energy Systems*, 69, 241-245.
64. Zhang, S., Lian, J., Zhao, Z., Xu, H., & Liu, J. (2008, June). Grouping model application on artificial neural networks for short-term load forecasting. *In Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on* (pp. 6203-6206). IEEE.
65. Zhu, S., Wang, J., Zhao, W., & Wang, J. (2011). A seasonal hybrid procedure for electricity demand forecasting in China. *Applied Energy*, 88(11), 3807-3815.

Appendix A

MATLAB code: Data Cleaning

Detecting and Replacing Outliers

```
clear all;
clc

%Loading Raw Data into the program
filename = '2012 - 2013 Raw data.xlsx';
sheet = 1;
X = xlsread(filename,sheet);
D = X;

%period "j" = 10
j = 10;
N = 0.6;
m = 0;
for t = 673:size(D);
    %Define the boundaries to detect the outliers
    UpperBound(t) = mean(D(t-j+1:t)) + N * std(D(t-j+1:t));
    LowerBound(t) = mean(D(t-j+1:t)) - N * std(D(t-j+1:t));
    %Identifying the outliers
    if UpperBound(t) < D(t) || D(t) < LowerBound(t);
        %Replace outliers with 2-period MA
        D(t) = (D(t-336) + D(t-672))/2;
        m = m + 1;
    else
        D(t) = D(t);
    end
end
end

%Plot the outcomes
figure
num = 1;
plot(UpperBound1(:,num),'g')
hold on
plot(LowerBound1(:,num),'g')
hold on
plot(X(:,num),'r')
hold on
plot(D1(:,num),'b')
```

Appendix B

MATLAB code: Training ANN with Backpropagation

Creating the ANN, Feeding Data, Recording Outputs

```
clear all;
clc;

%Loading cleaned data into the program
p = xlsread('Set.xlsx',7)';
t = xlsread('Target.xlsx',7)';
IN = xlsread('In.xlsx',7)';

for ii = 1:52;

for jjj = 1:48;

tic;

    %Run the program for separate time periods
    inputs = p([jjj jjj+48 jjj+96 jjj+144],ii:ii+50);
    targets = t([jjj,ii:ii+50]);
    in = IN([jjj jjj+48 jjj+96 jjj+144],ii);

%Set the ANN parameters
net = feedforwardnet([4], 'trainscg');
net.layers{1}.transferFcn = 'tansig';
net.performFcn = 'mse';
net.trainParam.epochs = 1000;
net.divideFcn = '';
net.trainParam.showWindow = false;

%Training the ANN with created network and selected data
[net, tr] = train(net, inputs, targets);

%Testing the performance
out = sim(net, in);

%Recording the outputs
A{ii,jjj} = out;
time{ii,jjj} = toc;

    jjj = jjj + 1
end
    ii = ii + 1
end
```

Call the Outputs and Calculating the Errors

```
OUT = cell2mat(A);
Sheet = 7;
filename = 'Results_In4_H(1,2)_Out1.xlsx';
xlRange = ['A', num2str(1)];
xlswrite(filename, OUT, Sheet, xlRange);

TIME = cell2mat(time);
Sheet = 7;
filename = 'Results_In4_H(1,2)_Out1.xlsx';
xlRange = ['A', num2str(55)];
xlswrite(filename, TIME, Sheet, xlRange);

Sheet = 7;
filename = 'Compare_In4_H(1,2)_Out1.xlsx';
s1 = ['A', num2str(1)];
s2 = ['AV', num2str(52)];
xlRange = [s1, ':', s2];
C = xlsread(filename, Sheet, xlRange);

Sheet = 7;
filename = 'Results_In4_H(1,2)_Out1.xlsx';
s1 = ['A', num2str(1)];
s2 = ['AV', num2str(52)];
xlRange = [s1, ':', s2];
out = xlsread(filename, Sheet, xlRange);

for i = 1:52;
    R = out(i, :);
    S = C(i, :);

    for j = 1:48
        e(j) = R(j) - S(j);
        j = j + 1;
    end
    e;

    sum = 0;
    for m = 1:48
        sum = sum + abs(e(m)/S(m));
        m = m + 1;
    end

    MAPE = (1/48)*(sum)*100

    Sheet = 7;
    filename = 'Compare_In4_H(1,2)_Out1.xlsx';
    s1 = ['B', num2str(i+54)];
    xlswrite(filename, MAPE, Sheet, s1);

    i = i + 1

end
```

Appendix C

MATLAB code: Training ANN with GA

Creating the Objective Function

```
function mse_calc = mse_test(x, net, inputs, targets)

% x has all the variables (weights and bias)
net = setwb(net, x');

%set weights and biases in their specific places
y = net(inputs);

%Calculating the error
e = targets - y;

%Calculating the Mean Squared Error
mse_calc = mse(e);

end
```

Creating the ANN, Feeding Data, Recording Output

```
clear all;
clc;

%Loading cleaned data into the program
p = xlsread('Set.xlsx',7)';
t = xlsread('Target.xlsx',7)';
IN = xlsread('In.xlsx',7)';

for ii = 1:52;

for jjj = 1:48;

tic;

[I N ] = size(inputs)
[O N ] = size(targets)

H = 1;
Nw = (I+1)*H+(H+1)*O;

%Create the ANN
net = feedforwardnet(H);
net = configure(net, inputs, targets);

%Call the Objective function
h = @(x) mse_test(x, net, inputs, targets);
```

```

%Set the parameters for GA
ga_opts = gaoptimset('TolFun',2e-2,'display', 'iter',
'Generations', 1000, 'PopulationSize', 100, 'MutationFcn',
@mutationgaussian, 'CrossoverFcn', @crossoversinglepoint,
'UseParallel', true);

%Optimize weights and bias with GA
[x_ga_opt, err_ga] = ga(h, Nw, [], [], [], [], [], [], [], ga_opts);

%Put the weights back in the ANN
net = setwb(net, x_ga_opt');

%Testing outputs
out = net(in)

%Recording the outputs
A{ii,jjj} = out;
time{ii,jjj} = toc;

    jjj = jjj + 1
end
    ii = ii + 1
end

```

Call the Outputs and Calculating the Errors

```

OUT = cell2mat(A);
Sheet = 7;
filename = 'Results_In4_H(1,2)_Out1.xlsx';
xlRange = ['A', num2str(1)];
xlswrite(filename, OUT, Sheet, xlRange);

TIME = cell2mat(time);
Sheet = 7;
filename = 'Results_In4_H(1,2)_Out1.xlsx';
xlRange = ['A', num2str(55)];
xlswrite(filename, TIME, Sheet, xlRange);

    Sheet = 7;
    filename = 'Compare_In4_H(1,2)_Out1.xlsx';
    s1=['A', num2str(1)];
    s2=['AV', num2str(52)];
    xlRange = [s1, ':', s2];
    C = xlsread(filename, Sheet, xlRange);

```

```

Sheet = 7;
filename = 'Results_In4_H(1,2)_Out1.xlsx';
s1=['A',num2str(1)];
s2=['AV',num2str(52)];
xlRange = [s1, ':', s2];
out = xlsread(filename,Sheet,xlRange);

for i = 1:52;
R = out(i,:);
S = C(i,:);

for j = 1:48
e(j) = R(j) - S(j);
j = j + 1;
end
e;

sum = 0;
for m = 1:48
sum = sum + abs(e(m)/S(m));
m = m + 1;
end

MAPE = (1/48)*(sum)*100

Sheet = 7;
filename = 'Compare_In4_H(1,2)_Out1.xlsx';
s1=['B',num2str(i+54)];
xlswrite(filename,MAPE,Sheet,s1);

i = i + 1

end

```

Appendix D

MATLAB code: Training ANN with PSO

Creating the Objective Function

```
function mse_calc = mse_test(x, net, inputs, targets)

% x has all the variables (weights and bias)
net = setwb(net, x');

%set weights and biases in their specific places
y = net(inputs);

%Calculating the error
e = targets - y;

%Calculating the Mean Squared Error
mse_calc = mse(e);

end
```

Creating the ANN

```
clear all;
clc;

%Loading cleaned data into the program
p = xlsread('Set.xlsx',7)';
t = xlsread('Target.xlsx',7)';
IN = xlsread('In.xlsx',7)';

for ii = 1:52;

for jjj = 1:48;

tic;

[I N ] = size(inputs)
[O N ] = size(targets)

H = 1;
Nw = (I+1)*H+(H+1)*O;

%Create the ANN
net = feedforwardnet(H);
net = configure(net, inputs, targets);
```


Starting PSO

```
rng default
% pso parameters values
m=Nw; % number of variables
n=100; % population size
wmax=0.9; % inertia weight
wmin=0.4; % inertia weight
c1=2; % acceleration factor
c2=2; % acceleration factor
LB= zeros(1, m); %lower bounds of variables
for ll = 1:m;
    LB(ll) = -1000;
end
UB= zeros(1, m); %upper bounds of variables
for jj = 1:m;
    UB(jj) = 1000;
end
% pso main program
maxite=100; % set maximum number of iteration
maxrun=1; % set maximum number of runs need to be
for run=1:maxrun
    run
    % pso initialization-----
-start
    for i=1:n
        for j=1:m
            x0(i,j)=round(LB(j)+rand()*(UB(j)-LB(j)));
        end
    end
    x=x0; % initial population
    v=0.1*x0; % initial velocity
    for i=1:n
        f0(i,1)=mse_test(x0(i,:),net, inputs, targets);
    end
    [fmin0,index0]=min(f0);
    pbest=x0; % initial pbest
    gbest=x0(index0,:); % initial gbest
    % pso initialization-----
---end

    % pso algorithm-----
-start
    ite=1;
    tolerance=1;
    while ite<=maxite && tolerance>10^-4

        w=0.5; % update inertial weight
        % pso velocity updates
        for i=1:n
            for j=1:m
                v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))...
                    +c2*rand()*(gbest(1,j)-x(i,j));
            end
        end
    end
end
```

```

% pso position update
% pso position update
for i=1:n
for j=1:m
x(i,j)=x(i,j)+v(i,j);
end
end
% handling boundary violations
for i=1:n
for j=1:m
if x(i,j)<LB(j)
x(i,j)=LB(j);
elseif x(i,j)>UB(j)
x(i,j)=UB(j);
end
end
end
% evaluating fitness
for i=1:n
f(i,1)=mse_test(x(i,:),net, inputs, targets);
end
% updating pbest and fitness
for i=1:n
if f(i,1)<f0(i,1)
pbest(i,:)=x(i,:);
f0(i,1)=f(i,1);
end
end
[fmin,index]=min(f0); % finding out the best particle
ffmin(ite,run)=fmin; % storing best fitness
ffite(run)=ite; % storing iteration count
% updating gbest and best fitness
if fmin<fmin0
gbest=pbest(index,:);
fmin0=fmin;
end
% calculating tolerance
if ite>100;
tolerance=abs(ffmin(ite-100,run)-fmin0);
end
% displaying iterative results
if ite==1
disp(sprintf('Iteration Best particle Objective fun'));
end
disp(sprintf('%8g %8g %8.4f',ite,index,fmin0));
ite=ite+1;
end
% pso algorithm-----
---end
gbest;
fvalue=mse_test(gbest,net, inputs, targets);
rgbest(run,:)=gbest;
disp(sprintf('-----'));
end

```

```

% pso main program-----
-----end
disp(sprintf('\n'));
disp(sprintf('*****
*****'));
disp(sprintf('Final Results-----'));
[bestfun,bestrun]=min(fff)
best_variables=rgbest(bestrun,:);
disp(sprintf('*****
*****'));
toc
% PSO convergence characteristic
plot(ffmin(1:ffite(bestrun),bestrun),'-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
#####

%-----
#####3
net = setwb(net, best_variables');
out = net(in);

A{ii,jjj} = out;
time{ii,jjj} = toc;

        ii = ii + 1
end      jjj = jjj + 1
end

```

Call the Outputs and Calculating the Errors

```

OUT = cell2mat(A);
Sheet = 1;
filename = 'Results.xlsx';
xlRange = ['A',num2str(1)];
xlswrite(filename,OUT,Sheet,xlRange);

TIME = cell2mat(time);
Sheet = 1;
filename = 'Results.xlsx';
xlRange = ['A',num2str(10)];
xlswrite(filename,TIME,Sheet,xlRange);

        Sheet = 1
        filename = 'Compare.xlsx';
        s1=['A',num2str(1)];
        s2=['AV',num2str(4)];
        xlRange = [s1, ':', s2];
        C = xlsread(filename,Sheet,xlRange);

```

```

Sheet = 1
    filename = 'Results.xlsx';
    s1=['A',num2str(1)];
    s2=['AV',num2str(4)];
    xlRange = [s1, ':', s2];
    out = xlsread(filename,Sheet,xlRange);

for i = 1:4;
    R = out(i,:);
    S = C(i,:);

    for j = 1:48
        e(j) = R(j) - S(j);
        j = j + 1;
    end
    e;

    sum = 0;
    for m = 1:48
        sum = sum + abs(e(m)/S(m));
        m = m + 1;
    end

    MAPE = (1/48)*(sum)*100

    Sheet = 1
    filename = 'Compare.xlsx';
    s1=['B',num2str(i+6)];
    xlswrite(filename,MAPE,Sheet,s1);

    i = i + 1

end

```

Appendix E

MATLAB code: Training ANN with PSO+GA

Creating the Objective Function

```
function mse_calc = mse_test(x, net, inputs, targets)

% x has all the variables (weights and bias)
net = setwb(net, x');

%set weights and biases in their specific places
y = net(inputs);

%Calculating the error
e = targets - y;

%Calculating the Mean Squared Error
mse_calc = mse(e);

end
```

Creating the ANN

```
clear all;
clc;

%Loading cleaned data into the program
p = xlsread('Set.xlsx',7)';
t = xlsread('Target.xlsx',7)';
IN = xlsread('In.xlsx',7)';

for ii = 1:52;

for jjj = 1:48;

tic;

[I N ] = size(inputs)
[O N ] = size(targets)

H = 1;
Nw = (I+1)*H+(H+1)*O;

%Create the ANN
net = feedforwardnet(H);
net = configure(net, inputs, targets);
```

Starting PSO

```
% pso parameters values
m = Nw; % number of variables
n = 100; % population size
c1=2; % acceleration factor 1
c2=2; % acceleration factor 2
maxite=1000; % set maximum number of iteration

LB= zeros(1, m); %lower bounds of variables/geans of each particle
for ll = 1:m;
    LB(ll) = -1000;
end
UB= zeros(1, m); %upper bounds of variables/geans of each particle
for jj = 1:m;
    UB(jj) = 1000;
end

%Start PSO
%Start with a random population
for i=1:n
    for j=1:m
        R = [LB(j) UB(j)];
        x0(i,j)=rand()*range(R)+min(R);
    end
end

x=x0; % initial population
v=0.1*x0; % initial velocity
%Set the inputs to the objective function and calculate the
fitness of
%each particle. x0(i,:) = all geans (columns) of each particle
(rows)
for i=1:n
    f0(i,1)=mse_test(x0(i,:),net, inputs, targets);
end
plotmax = max(f0)
plot(f0, '+')
axis([0 100 0 plotmax])
drawnow

[fmin0,index0]=min(f0); %find the minimum from the fitness set,
"fmin0" and the position of it, "index0"
pbest=x0; % initial pbest/the best of each particle is still same
at the begining
gbest=x0(index0,:); % initial gbest/ the best of the initial
particle set

GenCount = 0;
ite=1; %just a number to start the loop; number of iterations
stall = 0; %just a number to start the loop; the number of stall
generations
while stall<200 && ite<=maxite %runs until conditions are not
true
```

```

w=0.5; %inertia weight
%velocity of each particle is updated
for i=1:n
for j=1:m
v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j)) ...
+c2*rand()*(gbest(1,j)-x(i,j));
end
end

%position of each particle is updated
for i=1:n
for j=1:m
x(i,j)=x(i,j)+v(i,j);
end
end

% handling boundary violations: if the position is beyond the
boundary,
% make the position equal to the boundry value
for i=1:n
for j=1:m
if x(i,j)<LB(j)
x(i,j)=LB(j);
elseif x(i,j)>UB(j)
x(i,j)=UB(j);
end
end
end

% evaluating fitness of the new particle set
for i=1:n
f(i,1)=mse_test(x(i,:),net, inputs, targets);
end

% updating pbest and fitness based on the new fitness values
for i=1:n
if f(i,1)<f0(i,1)
pbest(i,:)=x(i,:);
f0(i,1)=f(i,1);
end
end
plot(f0, '+')
axis([0 100 0 plotmax])
drawnow

[fmin,index]=min(f0); % %find the minimum from the new fitness
set, "fmin" and the position of it, "index"
ffmin(ite)=fmin; % storing best fitness at each iteration

% updating gbest and best fitness
if fmin<fmin0
gbest=pbest(index,:);
fmin0=fmin;
end

```

Checking the Stall Generations

```
%Stall generations
if ite>2;
    if ((ffmin(ite-1) - ffmin(ite))/ffmin(ite-1))<0.02;
        stall = stall + 1;
    else
        stall = 0;
    end
end
```

Applying Crossover and Mutation Functions

```
%-----Start GA Operations -----
-----

if GenCount < 2 && stall > 150;
disp(['Start GA operation ', num2str(GenCount+1), ' at '
num2str(ite-1), ' iterations']);

    %Half mutation and Half crossover
    %m = number of variables
    %n = population size

    %start producing Mutation child. "M" is the number of mutation
    child in the population

    M = fix(n/2);
    for d = 1:M
        x(d,:) = x(randperm(100,1),:) + (1-(-1)).*rand() + -(1);
    end

    %start producing Crossover child. "C" is the number of crossover
    child in the population

    vv = fix(m/2);
    for q = M+1:n
        s = randperm(100,2);
        x(q,:) = horzcat(x(s(1),1:vv) , x(s(2),vv+1:m));
    end

%-----should have this part after GA childs-----
-----

    % handling boundary violations: if the position is beyond the
    boundry,

for i=1:n
    for j=1:m
        if x(i,j)<LB(j)
            x(i,j)=LB(j);
        elseif x(i,j)>UB(j)
```



```

x(i,j)=UB(j);
end
end
end

% evaluating fitness of the new particle set
for i=1:n
f(i,1)=mse_test(x(i,:),net, inputs, targets);
end
plot(f, '+')
axis([0 100 0 plotmax])
drawnow
pause(1)

% updating pbest and fitness based on the new fitness values
for i=1:n
if f(i,1)<f0(i,1)
    pbest(i,:)=x(i,:);
    f0(i,1)=f(i,1);
end
end

[fmin,index]=min(f0); % %find the minimum from the new fitness
set, "fmin" and the position of it, "index"
ffmin(ite)=fmin; % storing best fitness at each iteration

% updating gbest and best fitness
if fmin<fmin0
gbest=pbest(index,:);
fmin0=fmin;
end

    GenCount = GenCount + 1;
    stall = 0;
end
%-----End of GA part-----
-----

```

Starting PSO Back

```

% displaying iterative results
if ite==1
disp(sprintf('Iteration Best particle Objective fun'));
end
disp(sprintf('%8g %8g %8.4f',ite,index,fmin0));
ite=ite+1;
end

%get the final result with best variables
gbest;
fvalue = mse_test(gbest,net, inputs, targets);
disp(sprintf('-----'));

```

```

%disply the final result
disp(sprintf('\n'));
disp(sprintf('*****
*****'));
disp(sprintf('Final Results-----'));
Best_value = fvalue
Best_variables = gbest
disp(sprintf('*****
*****'));
toc
% plot the result at each epoch
plot(ffmin, '-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
#####

%-----
#####3
net = setwb(net, Best_variables');
out = net(in);

A{ii,jjj} = out;
time{ii,jjj} = toc;

    ii = ii + 1
end
    jjj = jjj + 1
end

```

Call the Outputs and Calculating the Errors

```

OUT = cell2mat(A);
Sheet = 1;
filename = 'Results.xlsx';
xlRange = ['A', num2str(1)];
xlswrite(filename, OUT, Sheet, xlRange);

TIME = cell2mat(time);
Sheet = 1;
filename = 'Results.xlsx';
xlRange = ['A', num2str(10)];
xlswrite(filename, TIME, Sheet, xlRange);

    Sheet = 1
    filename = 'Compare.xlsx';
    s1 = ['A', num2str(1)];
    s2 = ['AV', num2str(4)];

```

```

xlRange = [s1, ':', s2];
C = xlsread(filename, Sheet, xlRange);

Sheet = 1
filename = 'Results.xlsx';
s1=['A',num2str(1)];
s2=['AV',num2str(4)];
xlRange = [s1, ':', s2];
out = xlsread(filename, Sheet, xlRange);

for i = 1:4;
R = out(i, :);
S = C(i, :);

for j = 1:48
e(j) = R(j) - S(j);
j = j + 1;
end
e;

sum = 0;
for m = 1:48
sum = sum + abs(e(m)/S(m));
m = m + 1;
end

MAPE = (1/48)*(sum)*100

Sheet = 1
filename = 'Compare.xlsx';
s1=['B',num2str(i+6)];
xlswrite(filename,MAPE, Sheet, s1);

i = i + 1

end

```