DEEP LEARNING FOR SHORT-TERM ELECTRICITY LOAD FORECASTING

BY

PYAE PYAE PHYO

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 2017

Ref. code: 25605822043898AVW

DEEP LEARNING FOR SHORT-TERM ELECTRICITY LOAD FORECASTING

BY

PYAE PYAE PHYO

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 2017

Ref. code: 25605822043898AVW

DEEP LEARNING FOR SHORT-TERM ELECTRICITY LOAD FORECASTING

A Thesis Presented

By

PYAE PYAE PHYO

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 — (Assoc. Prof. Dr. Pornthipa Ongkunaruk)

Committee Member and Chairperson of Examination Committee — (Asst. Prof. Dr. Somsak Kittipiyakul)

NOVEMBER 2017

Abstract

DEEP LEARNING FOR SHORT-TERM ELECTRICITY LOAD FORECASTING

by

PYAE PYAE PHYO

Bachelor of Engineering (Electrical Power), Technological University (Mandalay), Myanmar, 2013
Master of Science (Engineering and Technology), Sirindhorn International Institute of Technology, Thammasat University, 2017

Forecasting of the daily load demand is a complex problem as it is to solve nonlinearity with influenced external factors. Deep learning, machine learning and artificial intelligence techniques have been successfully employed in electric consumption load predictions, financial market predictions, and reliability predictions. In this paper, we propose to use deep neural network (DNN) and recurrent neural network (RNN) with long short-term memory (LSTM) for short-term load forecasting (STLF) to overcome nonlinearity problems and to achieve higher forecasting accuracy.

The historical data has been collected every 30 minutes for 24 hours from the Electricity Generating Authority of Thailand (EGAT). The proposed techniques are tested with cleaned data from 2012 to 2017 where holidays, bridging holidays, and outliers are replaced. The forecasting accuracy is measured by mean absolute percentage error (MAPE).

In this research, we propose two DNN forecasting structures, i.e., 1-period and 48-periods structures are tested and compared by using the proposed model. The results show that 1-period forecasting structure give more accurate than 48-periods forecasting structure.

Ref. code: 25605822043898AVW

Moreover, there are two different structures of training dataset including everyday training dataset and same day training dataset. The outcomes of deep neural network (DNN) are compared with artificial neural network (ANN) and support vector machines (SVM) with everyday training dataset. The empirical results reveal that the proposed DNN model outperforms ANN model and SVM model.

Moreover, the DNN model trained with same day training datasets provides better performance than DNN trained with everyday training dataset for weekends, bridging holidays, and Monday. In addition, the DNN using same day training datasets provides higher accuracies for December and January. Consequently, the DNN model provides better forecasting accuracy in power industrial management.

In addition, we train both DNN and RNN with LSTM using new collected data from 2013 to 2017. The empirical results show that RNN outperforms DNN to overcome time series data problems by keeping long data information at every time steps. From all mentioned above outcomes, deep learning models based on neural network provide reliable and robustness accuracy for STLF in power system management.

Keywords: Short-Term Electricity Load Forecasting, Deep Learning, Deep Neural Network, Recurrent Neural Network, Long Short-Term Memory

Ref. code: 25605822043898AVW

Acknowledgements

I would like to express my profound gratitude to my supervisor Assoc. Prof. Dr. Chawalit Jeenanunta, School of Management Technology, Sirindhorn International Institute of Technology, Thammasat University, Thailand, for giving me freedom and encouragement to pursue the research that led to this dissertation. He supervised the research and gave invaluable guidance from the starting to finalizing stage of the research. His extensive knowledge, unfailing optimism, and easy nature have made this thesis an exciting and enjoyable experience, which definitely will shape my future. I am also thankful to him for his constant encouragement, kind advice and support throughout my work.

Also, I am grateful to every useful comment and suggestions from my Co-Advisor Assist. Prof. Dr. Somsak Kittipiyakul and Assoc. Prof. Dr. Pornthipa Ongkunaruk, External committee member and Chairperson of Examination Committee. Their advice always been helpful to my works. Afterward, I am highly indebted to SIIT, Thammasat University for their constant supervision and their support by providing me an EFS scholarship and golden opportunity to pursue my master degree.

Finally, a special thanks to my elder sisters 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 seniors, my friends, my research group 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.

Ref. code: 25605822043898AVW

Table of Contents

Ref. code: 25605822043898AVW

Ref. code: 25605822043898AVW

Ref. code: 25605822043898AVW

List of Tables

Ref. code: 25605822043898AVW

List of Figures

Ref. code: 25605822043898AVW

Ref. code: 25605822043898AVW

Chapter 1
Introduction

1.1 Electricity Load Forecasting

Electricity plays an important role in our daily lives and one of the most popular driving factors of a country's economy. Load forecasting is indispensable section of designing, planning and operation of electric utilities, moreover, it is necessary to allocate considerable amount of electric energy to increase the economy significantly by electric power manufacturers. The electricity load forecasting that is a way of estimating what future electric load will be for a given perspective based on the available system information. Energy usage of load demand is one of major factor affecting on Thailand's economics evaluated by Electricity Generating Authority Thailand (EGAT). The total amount of 2.1 billion THB was spent on the energy usage and there is an increase amount comparing with 2013 and 2014. This increment of energy consumption is caused by increasing in population, economics, transportation, and human facilities in both Bangkok region and metropolitan region.

This load forecasting is generally classified according to time interval into three main categories. Moreover, there are two types of forecasting that are the subjective and objective forecasting methods. The first one is referred to methods that are used to measure either individual or group opinion. The better known subjective forecasting methods involve:

- Scales force composites.
- Customer survey.
- Jury of executive opinion.
- The Delphi method.

Another one is the objective forecasting methods that may be classified on the basic of past history data into two groups as time series methods and regression. Regression models often integrate the previous history of other series; however, time series forecasting methods apply only for the past historical data of the series to be predicted. The latter one has the advantage of easily being combined into a computer program for updating and automatic forecasting. A casual model forecasts the

Ref. code: 25605822043898AVW

dependent variable based on the evolution of one or more other independent variables. The final goal of using time series forecasting is to find predictable and repeatable patterns including increasing or decreasing linear trend, curvilinear trend and seasonal fluctuations in historical past data. Nowadays, electricity load forecasting has developed into the majority research field in power engineering. It is basic requirement of power generation, commercial appropriation between plants, scheduling preservation, and networking with interconnected utilities.

1.2 Types of Load Forecasting

In this section, types of load forecasting are illustrated in Table 1.1 in details. The input load flow study or emergency analysis in the daily operation like energy transfer scheduling, and demand-side management is needed to consider for short term load forecasting. In order to correlate predicted advancement in demand, the medium and long-term forecasting are applied for enlargement of capacity of generation, transmission and distribution. Load forecasting is classified as long-term; medium-term; and short-term forecasting according to time periods. Long-term load forecasting (LTLF) is more than one year and medium-term load forecasting (MTLF) ranges from one month to one year. Short-term load forecasting (STLF) can be regarded as minutes; hour; and day or week ahead predictions.

Table 1.1: Types of Load Forecasting

	Time Intervals	**Forecasted Values**	**Accuracy**	**Operation**	**Planning**
STLF	24h–1 week	Load curves	Fixed Load Curves	Economic Load Dispatch	Unit Commitment
MTLF	1 week-1 year	Load curves	Capacity>>Error	Unit Commitment	Reserve Planning
LTLF	>1year	Energy needed	Fixed Energy	Power System Planning	Future Capacity Expansion

It has big challenge with the upward trend in the electricity market prices since forecasting accuracy is the basic requirement in energy field. Moreover, forecasting performance is important thing to balance between energy supply and consumption.

Ref. code: 25605822043898AVW

Therefore, electrical producers and authority expect to reduce the generating energy cost. As mentioned above, STLF has been emerged more and more important role in Energy Management System (EMS) with power markets development over the past decades.

1.3 Factors Affecting on Electricity Load Forecasting

Similarly, there are multiple influenced factors on the performance of load which can be analyzed as time, day, temperature, weather, random and economic factors. Among them, one of the most popular factors is temperature associated with meteorological situations for short term load forecasting. Selection of suitable variables for load forecasting is related to generalize mathematical model. The accuracy of such model depends on the quality of input information. Deterministic and stochastic are two different categories of variable. On the other hand, in general, the electric consumption load is based on human activities. Similarly, human activities are also depended on population and their economic status. Therefore, the variables are more or less interconnected to each other. For instance, as a result of changing the consumer's comfort feeling for heaters (water, room) and air conditioner and so on, it causes the changes of weather conditions.

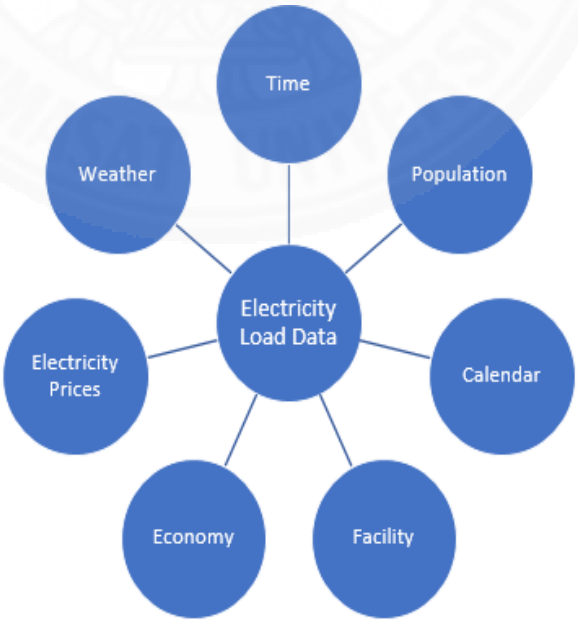

Figure 1.1: Different Factor Affecting on Load Forecasting

Ref. code: 25605822043898AVW

In Figure 1.1, it is clear that there are many influencing factors affected on the electricity load demand. Firstly, time is one of the most popular factors that can be separated into midnight, morning, evening, night, lunch time and so on when we consider for one day. So, we can forecast next day electricity demand because we have different past data for one day. Similarly, we can consider seasonally, yearly, monthly, weekly and daily etc. Next, weather is also one important thing affected on load, for example, we can diverge temperature, cloud cover or sunshine, humidity and so on. Moreover, calendar can also be considered like seasonal variation, daily variation, weekly cyclic and holidays, hence we can get different data and predict different results. Likewise, the remaining things such as population, human facilities, economic for business and electricity prices are influenced on forecasting electricity load demand.

1.4 Motivation

First of all, it is difficult for predictors to choose the right technique to solve both linear and nonlinear process. Nowadays, neural network algorithms are the most popular in AI-based models for nonlinear time series problems. However, if there are multiple hidden layers, neural networks do not work very well because of back propagation method. It takes time too long, sometimes it gets poor local minimum and slow convergence due to randomly initialization. To overcome the complex machine learning problems, deep architectures are the best choice for STLF. Moreover, the historical data have been collected every 30 minutes for 24 hours, so that it looks like sequential data. Recurrent neural network is a great tool for modelling time series. Therefore, we use deep learning models such as deep neural network and recurrent neural network to predict daily load.

1.5 Problem Statement

Electricity load forecasting techniques are very prominent parts to execute the good accurate forecast results for generation and transmission planning of the utility and its related economic parts. The generating amount of electricity should be stabilized with the utilizing one by customers since there is no way of storing electrical energy and

Ref. code: 25605822043898AVW

producing it immediately. If the expected energy and generating energy are not balanced, it can cause unsatisfied conditions for the country's economy. Therefore, electricity manufacturers must produce balanced electricity between generation, transmission, distribution and consumer services. In order to overcome problems, electricity load forecasting can help and reduce the extra cost over generating and end-user. The proposed deep learning methods are used in order to improve the forecasting accuracy performance.

1.6 Objective

The main objective of this research is to upgrade the forecasting accuracy performance by using different deep learning methods. Nowadays, artificial intelligence (AI) is everywhere based on knowledge around the world and machine learning (ML) which can learn from the data without explicitly programmed. However, it has many weakness points, i.e., back-propagation learning algorithms. To overcome learning and time series problems, deep learning methods were introduced.

Moreover, this research work is conducted with a short-term load forecasting with external factors like temperature, day of week, month of year and calendar dates. Considering external factors and original data have to be included to enhance the accuracy. Furthermore, the objective of this research is to get the minimum error of 1.9 percent for testing data by taking into account all these facts. Mean absolute percentage error (MAPE) is considered to calculate the forecasting error which is difference between the forecasted results and the actual load data from the Electricity Generating Authority Thailand (EGAT).

1.7 Significance of Study

To conquer the above-mentioned problems, there are various forecasting methods to have less than one percentage error in today's world. After forecasting by using the historical past data, we can calculate the forecast error to check how many the forecast demand deviates from the actual data. This is one of essential roles in load forecasting. There are three ways which are commonly used to measure the forecast errors.

Ref. code: 25605822043898AVW

The first way is mean absolute deviation (*MAD*) that indicates how many units the forecasting is deviate from the actual demand. The following equation shows how to calculate the MAD_d for day *d*.

$$MAD_d = \frac{1}{48}\sum_{t=1}^{48}\left|\frac{L_t(d) - F_t(d)}{L_t(d)}\right|$$

The second one is mean square error (*MSE*) which is similar to variance of random sample. MSE is computed without using absolute error terms. The equations for MSE_d for day *d* is given below.

$$MSE_d = \frac{1}{48}\sum_{t=1}^{48}(L_t(d) - F_t(d))^2$$

The last one is mean absolute percentage error (*MAPE*) that indicates how many units the forecasting is deviate from the actual demand. The last way is more popular than the others. Therefore, the $MAPE_d$ equation for each forecasting day *d* can be seen below because it is also needed to use in this paper.

$$MAPE_d = \frac{1}{48}\sum_{t=1}^{48}\left|\frac{L_t(d) - F_t(d)}{L_t(d)}\right| \times 100\%$$

However, this MAPE still could lead to extra costs, or could not afford the necessary amount of electricity for the consumers under the problem statement when the companies produce the large amount of electricity. In addition to this, over forecasts could cause extra costs for the utility companies. Therefore, precise accuracy of forecasting techniques is very prominent than by choosing single available technique. So, in this research, it is proposed to use different methods of deep learning to improve the forecasting accuracy performance from problem statement.

Ref. code: 25605822043898AVW

Chapter 2
Literature Review

2.1 Time Series Components

There are many techniques to forecast accuracy for short-term interval. Typically, these can be divided into three main categories in literature techniques like similar day approach, simulation models and time series models. Time series component can be separated into four parts: trend, seasonal, cyclic and random. In trend component, it can be persistent and it indicates upward or downward trends. It can change due to technologies, population, age, culture, etc and it can occur generally within several years duration.

In seasonal component, it is ordinary model of up and down variations and can cause due to climate, routines, etc. Normally, it can occur within a single year, for example, one-week period consists seven days. Likewise, for one month we can consider weekly and daily basis, however, weekly period is varied 4 to 4.5 weeks and daily also vary 28 to 31 days. Similarly, one year can be considered as four quarters, 12 months and 52 weeks, etc. So, we can consider it as different season because we know each day has different demand patterns.

Next, the cyclical patterns are persisting up or down motions affected by business cycle, governmental, and commercial factors during several years. And then random component patterns are unsystematic and 'residual' alternations due to random alteration or unexpected events. There is no repeating and it can occur during short duration period. Finally, there have been many techniques that can be seen from Figure 2.1.

2.2 Classifications of forecasting models

Forecasting techniques can be classified into two groups, i.e., traditional statistic models and artificial intelligence (AI) based models for STLF. Traditional statistical models include regression analysis, moving average, exponential smoothing, and stochastic time series models and so on. Machine learning, data mining, artificial

Ref. code: 25605822043898AVW

neural networks, genetic algorithms, fuzzy time series and expert systems are based on AI-based models. Figure 2.1 indicates various models of load forecasting.

Neural networks algorithms are the most popular in AI-based models for nonlinear time series problems. However, if there are multiple hidden layers, neural networks do not work very well because of back propagation method. It takes time too long, sometimes it gets poor local minimum and slow convergence due to randomly initialization. To overcome the complex machine learning problems, deep architectures are the best choice for STLF. Therefore, we used deep learning models to predict daily load for one-month period.

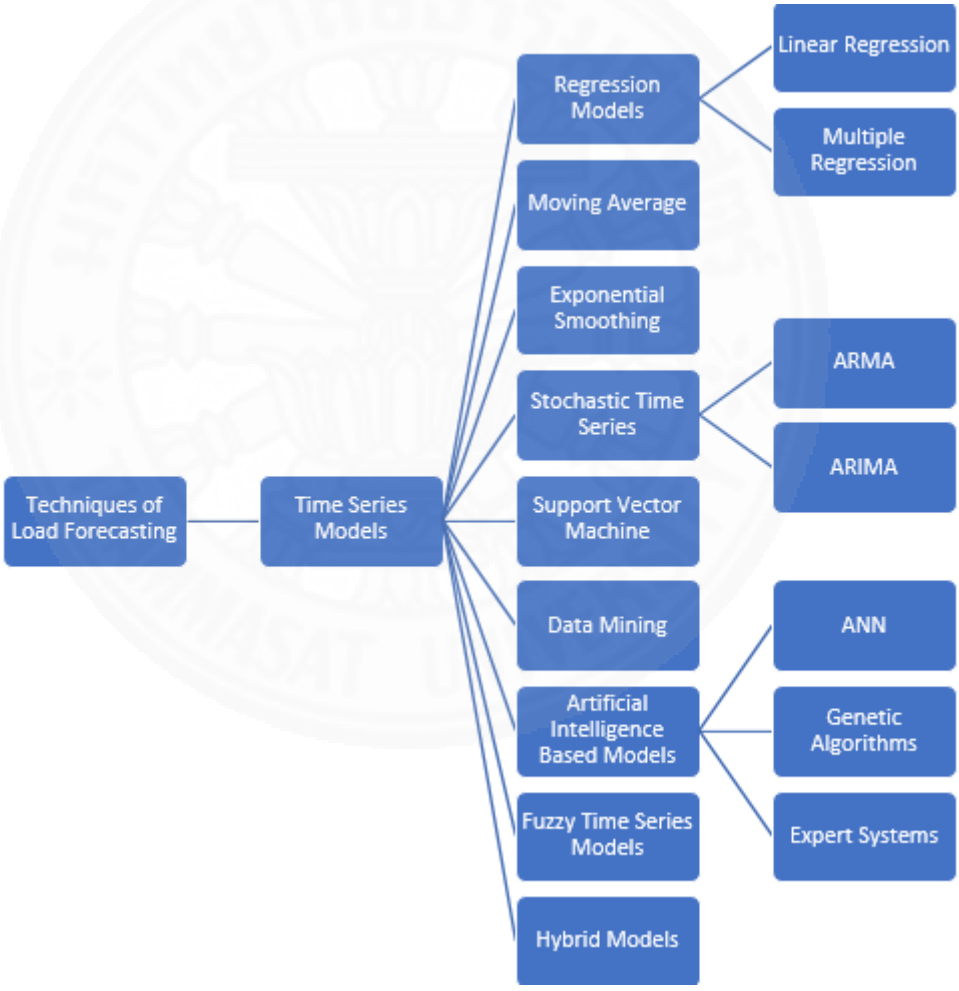

Figure 2.1: Different Load Forecasting Techniques

Ref. code: 25605822043898AVW

2.3 Related Works and Models

Warren McCulloch and Walter Pitts prior approached artificial neural network (ANN) since 1943 (Mcculloch & Pitts, 1990). Many researchers have shown that ANN is an excellent tool to apply many areas including medicine, business, communications, and industrial process control. ANN is the most popular computational model that is influenced by the structure and functional aspects of biological neural networks. A neural network composes of an interconnected group of artificial neurons, and it computes the targets by processing information using a connection. During the learning process, an ANN is an adaptive system that can change its structure depending on external information flowing through the network. Modern neural networks are usually used to model complex relationships between inputs and outputs to find patterns in data.

In the late 1980's and early 1990's, ANN has been applied to forecast load demand in the electric power system (Bala, Yadav, Hooda, & Registrar, 2014; Hippert, Pedreira, & Souza, 2001; Park, El-Sharkawi, Marks, Atlas, & Damborg, 1991). Jeenanunta and Abeyrathna adjusted the parameters of ANN to enhance the forecasting accuracy using a transfer function of ANN (Jeenanunta & Abeyrathna, 2016). A feed-forward neural network with the Levenberg-Marquardt algorithm performs well to get a better forecasting accuracy for electricity load demand (Rodrigues, Cardeira, & Calado, 2014). ANN relies on many things such as the numbers of invisible layers, backpropagation algorithms and the selected input variables to improve accuracy requirements for STLF (Rui & El-Keib, 1995). Even ANN has become very popular among machine leaning techniques, however, there are drawbacks to train the model and obtain better forecasting results due to the weakness of the backpropagation algorithm (Krunic, Krcmar, & Rajakovic, 2000).

Deep learning architectures become a vital role in the fields of pattern recognition, classification and complex prediction to solve the drawbacks of machine learning. The feedforward multilayer perceptron which is one of deep learning models was introduced for supervised learning algorithms. A deep belief network was approached to forecast load demand with the Macedonia hourly electricity consumption data (Dedinec, Filiposka, Dedinec, & Kocarev, 2016). Moreover, El-

Ref. code: 25605822043898AVW

sharkh presented multilayer perceptron, radial basis and RNN with a parallel structure ANN which gave better results compared to usual time series methods (El-sharkh, 2012). Rashid et. al approached a recurrent neural network (RNN) with the internal feedback structure for electricity load prediction and showed their results are reliable and robust (Rashid, Huang, Kechadi, & Gleeson, 2006).

Deep neural network (DNN) is commonly based on ANN with multiple hidden layers and the approximation error can be reduced by adding hidden layers between the input and output layers. Deep architectures are relevant to detect higher level representation and capture higher level abstractions. Quan et al. applied an ensemble deep belief network with one artificial dataset and three regression datasets time series and regression predictions (Qiu, Zhang, Ren, Suganthan, & Amaratunga, 2014). Moreover, a nonlinear auto-regressive RNN provided smoothly forecasted results, contrast with previous studies for hourly predictions of high resolution wave power (Hatalis, Pradhan, Kishore, Blum, & Lamadrid, 2014).

The feedforward multilayer perceptron which is one of deep neural network models was introduced for supervised learning algorithms (Liu et al., 2017). A deep belief network was also applied to forecast load demand with the hourly electricity consumption data in Macedonia (Dedinec et al., 2016). Moreover, El-sharkh presented multilayer perceptron, radial basis and recurrent neural network (RNN) with a parallel structure ANN which gave better results compared to usual time series methods (El-sharkh, 2012). Rashid et. al proposed a RNN with the internal feedback structure for electricity load prediction and showed their results are reliable and robust (Rashid et al., 2006).

Kelo and Dudul used a novel hybrid of wavelet and Elman network as a recurrent neural network to increase one-day ahead prediction accuracy in all seasons (Kelo & Dudul, 2012). A hybrid quantized elman recurrent neural network was proposed for hourly load predictions and provided an acceptable accuracy comparing with multilayer feedforward neural network (Li, Li, Xiong, Chai, & Zhang, 2014). Moreover, Siddarmeshwara conducted elman recurrent neural network which outperforms weather sensitive models and nonweather sensitive model (Siddarameshwara, 2010). A novel recurrent neural network (RNN) was approached

Ref. code: 25605822043898AVW

using extreme learning machine training method and obtained results were compared with traditional machine learning and linear regression model (Ertugrul, 2016). RNN could handle any type of activation functions in both forward and feedback loops and provide better performance than several computational methods.

Kermanshahi proposed recurrent neural network and three layer feed forward back propagation to forecast one-year ahead and provided reasonable results for long-term load forecasting (Kermanshahi, 1998). RNN executed better performance than time series models and computational intelligence methods (J. T. Connor, Atlas, & Martin, 1991). Marvuglia and Messineo showed the important correlation between electricity load demand and application of electronic devices for one-hour short-term forecasting (Marvuglia & Messineo, 2012). Recurrent wavelet network with a new orthogonal least square initialization method was applied to short-term special days over complex load forecasting problems (Baniamerian, Asadi, & Yavari, 2009). Training RNN on filtered data gave better accuracies than on unfiltered data (J. Connor, Martin, & Atlas, 1994). Vermaak and Botha applied the recurrent networks which provides higher dynamic load performance than normal networks. The more closely approximation model chose, the better expected results execute (Vermaak & Botha, 1998).

Support vector machines was introduced for STLF and it outperformed autoregressive model by comparing outcomes based on root-mean-square errors (Mohandes, 2002). Moreover, Mohandes also studied that the improvement of performance of SVM depends on increasing the training dataset. Chen et. al presented that SVM with conceptual time series like seasonal could enhance the forecasting performance and temperature factor could not influence on mid-term load forecasting from other experiments (Chen, Chang, & Lin, 2004). SVM has the ability to accurately forecast time series data. Moreover, SVR outperformed other nonlinear models to solve nonlinear, non-stationary and not defined a-priori problems (Sapankevych & Sankar, 2009). A recurrent support vector machines combined with genetic algorithms (RVSVMG) which determines parameters of SVM was applied to forecast regional electricity load (Pai & Hong, 2005). The performance using a hybrid model obtained higher forecasting accuracies than using the regression model, SVM

Ref. code: 25605822043898AVW

model and ANN model (Fard & Akbari-Zadeh, 2014). Sometimes researchers applied many hybrid models with SVR to improve dynamic high-performance accuracy for STLF (Ao, Wang, & Zhang, 2017).

Ref. code: 25605822043898AVW

Chapter 3
Methodology

3.1 Deep Learning (DL)

Deep learning (DL) has received a great success in the last couple of years. Many researchers have produced state-of-the-art results successfully in the fields of image classification, market predictions, automatic speech and face recognitions, natural language processing, and bioinformatics. There are various deep learning architectures such as long short-term memory (Hochreiter & Schmidhuber, 1997), recursive neural network (Urban et al., 2016), convolutional neural network (Urban et al., 2016), and deep belief network (Hinton, Osindero, & Teh, 2006). Most of them admit computational methods that are comprised of numerous hidden layers to learn representations of data with numerous abstraction levels. They also can detect complex structure in large data sets by using back propagation process to solve the drawbacks of machine learning.

Many researchers presented that the ability of traditional machine-learning methods was restricted to process raw data. These methods are required to design manually the feature extractors converting from raw data by human engineering. However, deep learning models are representation or feature learning models that can detect automatically multiple levels of features for detection or classification. Moreover, it needs very little engineering by hand, so it can increase the amount of available data and computation. The current progress of new learning architectures and algorithms advance in deep neural networks.

The main core of DL is a class of neural network models which have an input layer, an arbitrary number of hidden layers and an output layer. All layers are composed of neurons or neural units by sharing some similarities with the behavior of the neurons present in the human brain. For our cases, a neuron serves as a nonlinear function of the weighted sum of its inputs. Therefore, the neuron is really basic part of any DL model.

Ref. code: 25605822043898AVW

3.1.1 The Single Neuron Model

To describe neural networks, firstly a single neuron model is shown in Figure 3.1. A single neuron is a function which maps an input vector $\{x_1, x_2, \ldots, x_t\}$ with the weight vector $\{w_1, w_2, \ldots, w_t\}$ to a scalar output y passing through a nonlinear function f.

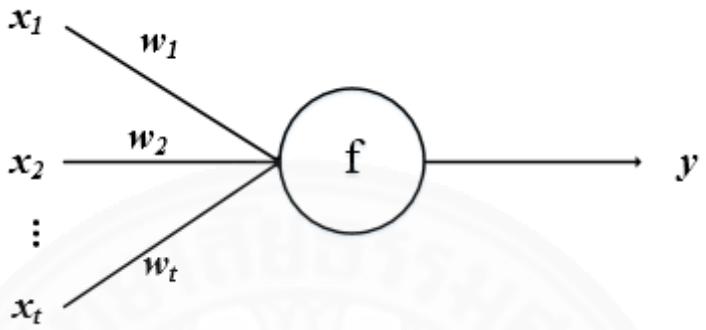

Figure 3.1: A Single Neuron Model

The link function f takes weighted sum of input x and executes y. The function provides the nonlinearity between input and output which is known as an activation function. In this paper, we describe three types of activation function which are the most commonly used for neural networks.

1. The sigmoid function: $f(x) = \dfrac{1}{1+e^{-x}}$
2. The hyperbolic tangent, or tanh function: $f(x) = \tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
3. The rectified linear activation function: $f(x) = \max(0, x)$

In this study, the proposed model uses the rectified linear activation function. This activation function is different from sigmoid and tanh because it is not bounded or continuously differentiable. It is piece-wise linear and saturates at exactly 0 whenever the input x is less than 0. Figure 3.2 represents three different activation functions.

Ref. code: 25605822043898AVW

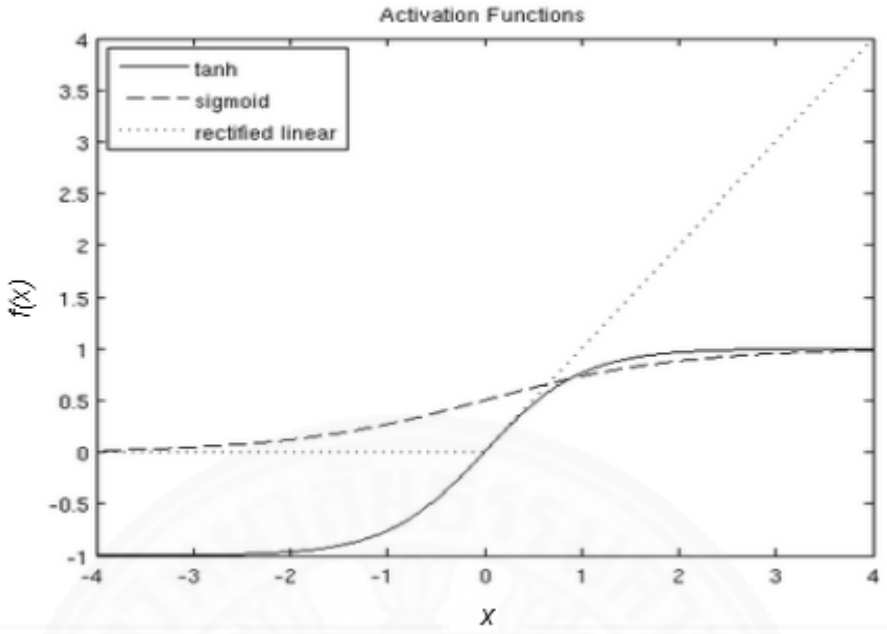

Figure 3.2: Activation Functions

3.1.2 Neural Network Model

Neural network model is used as simple feed-forward neural network trained by a back-propagation algorithm. In the feed-forward process, the information is only moving to the forward direction from the input nodes *x* passing through the hidden nodes to the output nodes *y* without any cycling or looping in the network. The model uses simple sigmoid activation function (*f*) to produce output values (*y*). In Figure 3.3, suppose that w_{kj} is the weight connection between input layer and hidden layer, and w_{ij} is the weight connection between hidden layer and output layer.

Ref. code: 25605822043898AVW

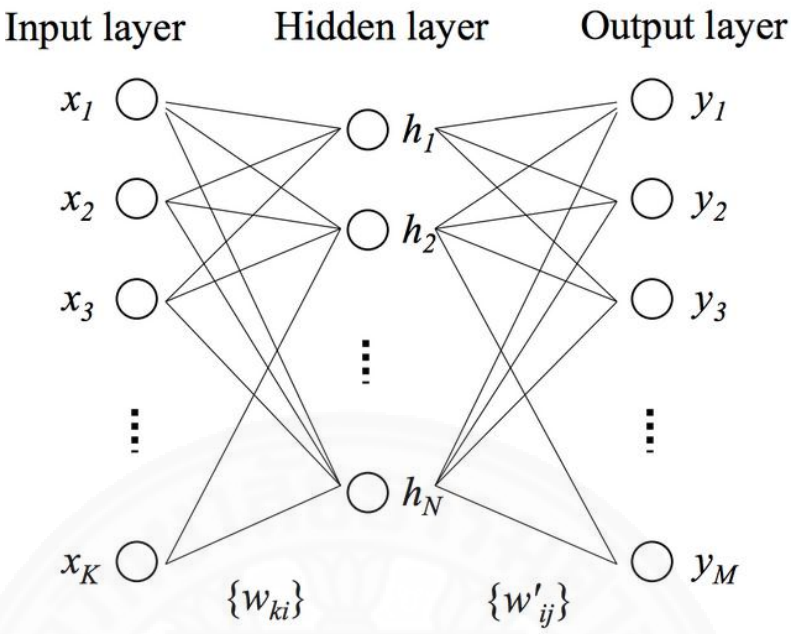

Figure 3.3: Neural network structure

The output equation for the hidden layer is: $h_i = f(\sum_{k=1}^{K} w_{ki} x_k)$

The output equation for the output layer is: $y_j = f(\sum_{i=1}^{N} w'_{ij} h_i)$

The objective function is to minimize the error is written as: $E = \frac{1}{2}\sum_{j=1}^{M}(y_j - t_j)^2$

Afterwards, the network updates weights (*w*) using back propagation algorithm (Rui & El-Keib, 1995).

3.1.3 Back Propagation Algorithm

In back propagation algorithm, the predicted output values are compared with the target values to calculate the value of some predefined error-function. By feeding the error back through the network, the algorithm adjusts the weights of each connection to minimize error by some amount depending on specified learning rate. We start with the final output error ($y_j - t_j$) for the output neuron *j* and this error gets propagated backwards throughout the network in order to update the weights. To update weight value, we compute gradient descent using chain rule. The following equations are how to compute gradient and update weight.

Ref. code: 25605822043898AVW

The equation for the gradient: $\frac{\partial E}{\partial w_{ki}} = \sum_{j=1}^{M}[(y_j - t_j).y_j(1-y_j).w_{ij}^{'}].h_i(1-h_i).x_k$

The update weight equation: $w_{ki}^{new} = w_{ki}^{old} - \eta.\frac{\partial E}{\partial w_{ki}}$

This training back propagation process is repeated until the performance of the network is good enough and the network converges to a small error after repeating many training cycles.

3.2 Deep Neural Network (DNN)

In this study, deep learning is approached to train the model consisting of a large number of processing layers. Deep neural network is a class of neural network model that has an input layer, an output layer, and an arbitrary number of hidden layers. In Figure 3.4, the input *x* layer and the output *y* layer are referred by the bottom and the top layers respectively. The layers between *x* and *y* represents the hidden layers (*h*) for the network which perform as a black box.

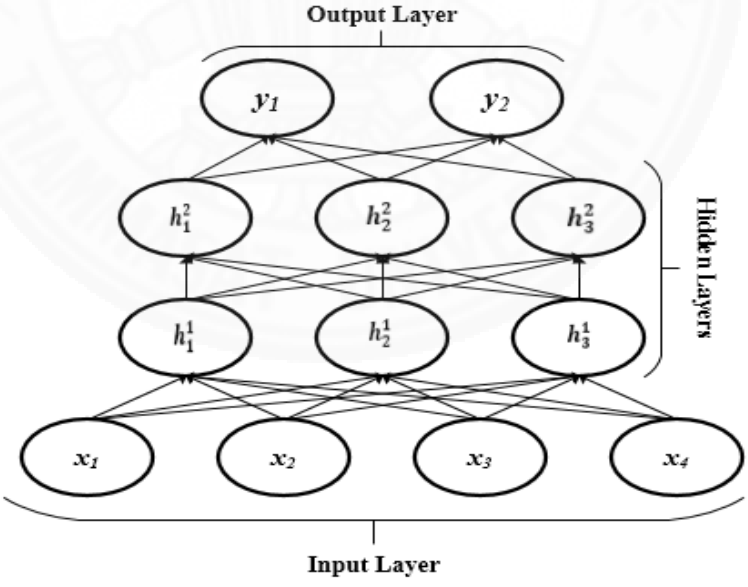

Figure 3.4: Deep Neural Network Structure

The DNN model trains a feed-forward network to execute the corresponding output values throughout all invisible layers and neuron nodes in the

Ref. code: 25605822043898AVW

forward propagation. The following equation indicates that a nonlinear function f takes weighted sum of input x and returns y.

$$y = f(\sum_{i=1}^{k} w_{ij}^{l} x_i) = f(w^t x)$$

where,

y = scalar output,

x_i = the ith input,

w_{ij}^{l} = the weight between node i in layer (l-1) and node j in layer l,

f = activation function.

The main core is to minimize an error term for the output layer after producing the corresponding output values. Therefore, the network compares predicted output values with the actual existing values. Next, the proposed model optimizes the minimum error using stochastic gradient descent (SGD) algorithm before updating weights (Bottou, 2010).

3.2.1 Stochastic Gradient Descend (SGD)

The objective of using SGD is to overcome speed convergence obstacles and getting stuck at a local minimum. At first, the training data are shuffled at each iteration of training network during SGD process. Next, the entire parameters are updated using only one sample or a few training samples. To reach a global minimum, SGD updates parameters frequently in the direction of the gradient of the loss function at every iteration. Unlike ordinary gradient descent, SGD selects a single data point instead of entire dataset to compute the gradient at each iteration. The following equation describes how to update weights in the SGD process.

$$w^{new} = w^{old} - \eta \nabla_w J(w^{old}; x^{(k)}, y^{(k)})$$

where,

w^{new} = updated weight value,

w^{old} = old weight value,

J = gradient value,

η = learning rate,

($x^{(k)}$, $y^{(k)}$) = a pair of training sample at k iteration.

Ref. code: 25605822043898AVW

There are many advantages to applying SGD over ordinary gradient descent. In general, most gradient optimization methods converge effectively in terms of using the full training set.

1. SGD can converge much faster than ordinary gradient descent methods because of less memory intensive on behalf of using one data point at a time (Bottou, 2010).
2. SGD has the ability to get a meaningful update without iterating over the entire dataset to overcome redundancy into datasets.
3. If the loss function is convex, using SGD has a guarantee to find a global minimum. SGD can obtain better performance solutions for big learning models and large data sets.

3.3 Recurrent Neural Network (RNN)

Recurrent neural network (RNN) is a neural network model proposed in the 80's for modelling time series. In a traditional neural network, all inputs and outputs are independent of each other. Therefore, neural network cannot handle for time series prediction (e.g., of financial series), time series production (e.g., motor control in non-Markovian environments) and time series classification or labeling (e.g., rhythm detection in music and speech). RNN is a great tool for modelling sequential data that is dataset are depended on other points.

The structure of the network is similar to feed-forward neural network, but it allows a recurrent hidden state whose activation at each time is dependent on that of the previous time (cycle). RNN is called recurrent because it performs the same task for every part of a sequence, with the output being depended on the previous computations. RNNs have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps.

Ref. code: 25605822043898AVW

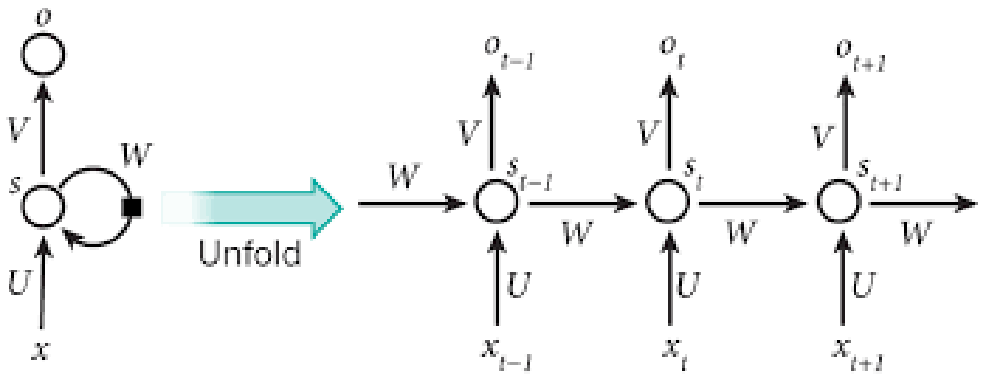

Figure 3.5: Recurrent Neural Network Structure

Figure 3.5 indicates a RNN being unrolled (or unfolded) into a full network. For example, if the sequence is 5-time steps, the network would be unrolled into a 5-layer neural network, one layer for each time steps. The formulas in a RNN are as follows:

$$s_t = f(ux_t + ws_{t-1})$$

$$o_t = \tanh(vs_t)$$

According to the above equation, x_t is the input at time step *t*, s_t is the hidden state at time step and o_t is the output at time step t. The recurrent model only has single layer, but it keeps a state to remember analysis. This state "recurs" back into the net with each input. The hidden layers and the output depend from previous states of the hidden layers. First of all, input data flow into the model's single layer. Afterwards, data process like traditional net, but net also receives the state along with input. For the first state point, net use initial state which is depended on type of data. After processing, data is output with a new state that represents most recent state point. Then this new state is fed back into the net with next data and so on. The net repeats these steps until all data is processed. It can produce a different output for same input depending on a current state since the state changes at every step.

Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (*U*, *V*, and *W* above equation) across all steps. Training a RNN is similar to training a traditional Neural Network. But for

Ref. code: 25605822043898AVW

back-propagation algorithms, RNNs use the Back-propagation Through Time (BPTT) because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. Consequently, vanilla RNNs trained with BPTT have difficulties learning long-term dependencies due to vanishing/exploding gradient problem. To overcome long-term dependencies problems, the solution is to use gating methods such as long short-term memory (LSTM) and gated recurrent unit (GRU).

3.4 Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) was first proposed by Hochreiter & Schmidhuber (1997) and since then has been modified by many researchers (Hochreiter & Schmidhuber, 1997). The LSTM architecture consists of a set of recurrently connected subnets, known as memory blocks. Each memory block consists of: memory cell, input gate, forget gate and output gate. Unlike the traditional recurrent unit which overwrites its content each time step, the LSTM unit is able to decide whether to keep the existing memory via the introduced gates. LSTMs avoid explicitly the long-term dependency problem. They remember information for long periods of time, not something they struggle to learn. All RNNs have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four interacting in a very special way.

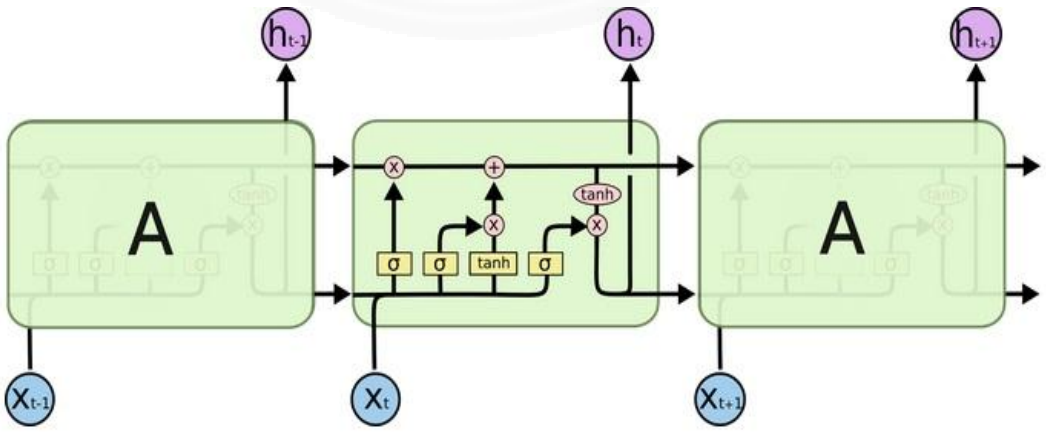

Figure 3.6: Long Short-Term Memory Structure

Ref. code: 25605822043898AVW

In Figure 3.6, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles denote pointwise operations, like vector addition. The yellow boxes are learned neural network layers. Lines merging denotes concatenation, while a line forking denotes its content being copied and the copies going to different locations. The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. It maintains information flowing along through it unchanged. The LSTMs are capable to remove or add information to the cell state by carefully regulating gates. Gates tend to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each element should be let through. A value of zero means "forget everything," while a value of one means "keep everything". An LSTM has three gates to protect and control the cell state.

The first step in LSTM is to decide what information need to throw away from the cell state by a sigmoid layer called the "forget gate layer." It looks at previous hidden layer and input, and outputs a number between 0 and 1 for each number in the cell state. The next step is to decide what new information going to be stored in the cell state by combining two parts to create an update to the state. The first one is that a sigmoid layer called the "input gate layer" decides which values need to update. The second one is that a *tanh* layer creates a vector of new candidate values that could be added to the state. Next, multiplying old state by forgetting the things and adding new candidate's values in order to update the old cell state into new cell state. Finally, the net executes the output which will be based on our cell state, but will be a filtered version. First, a sigmoid layer executes outputs using the cell state. Then, we put the cell state through tanh and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

3.5 Support Vector Machine

Support vector machine (SVM) is one of the computational or mathematical model to solve complex problems between inputs and outputs (Mohandes, 2002). The SVM learning method can be used for regression, classification and other tasks. It also can

Ref. code: 25605822043898AVW

learn a fast algorithm and provide good results for many tasks. In the training process, a SVM uses linear or quadratic and even asymmetric loss functions. A support vector machine usually sets up a hyperplane or set of hyperplanes in a high- or infinite-dimensional space. The hyperplane might have the largest distance to the nearest training data points of functional margin in order to achieve good separation. The sets of hyperplanes separate nonlinearly in a finite dimensional space.

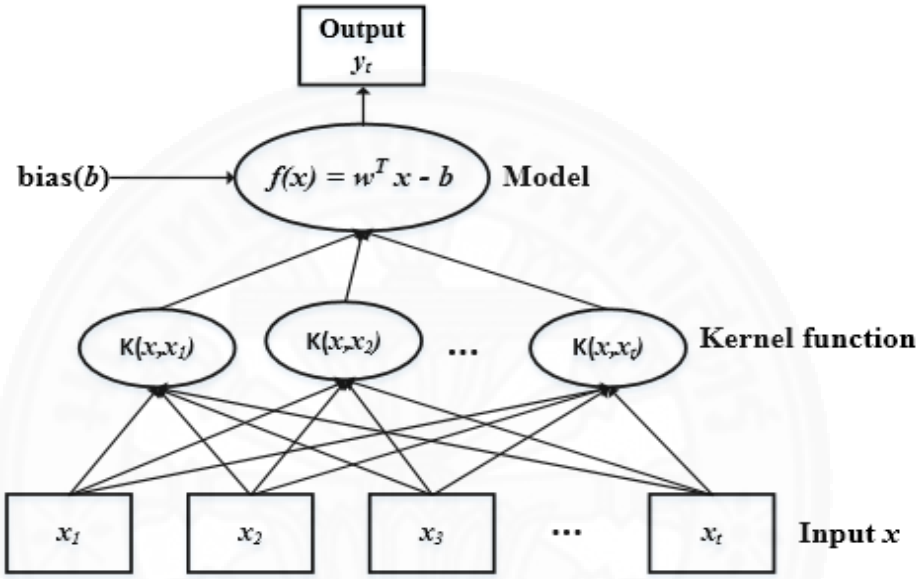

Figure 3.7: Support Vector Machine Structure

In order to solve this problem, the original finite-dimensional space approaches to map into a higher-dimensional space by making discrimination easier in that space. Therefore, in this study, we use two essential factors for the implementation of SVM. The first thing is the kernel model to get a large amount of solution space. Another one is the quadratic function to achieve SVM parameters in the entire training process. In the Figure 3.7, input *x* represents the input of training data and output *y* refers to a corresponding output value. The objective concept of SVM is to map the input data into a high dimensional space from a non-linear mapping and conduct a linear regression.

Ref. code: 25605822043898AVW

Chapter 4
Design of Experiment

4.1 Data Collection and Cleansing

The collected 30 minutes load data from 1st March 2009 to 31st December 2013 are from the Electricity Generating Authority of Thailand (EGAT). This data has been collected from five different regions: Central, Bangkok and Metropolitan, South, North and North-East in Thailand. There is a low electricity demand with only one peak load curve at night in the three regions: South, North, and North-East. On the other hand, there is a high electricity demand with three peak load curves in two regions: Bangkok and Metropolitan, and Central. In this research, only Bangkok and Metropolitan areas are considered because of a large load demand value and high variations.

4.1.1 Holidays and Bridging Holidays Replacement

The original historical data must be cleaned because there are many holidays, missing values, and outliers. If these outliers are included in the training data, the accuracy performance of load predictions would be lower. We categorize load patterns into five patterns as Monday, weekday, weekend, holiday and bridging holiday as shown in Figure 4.1. For bridging holiday load pattern, for instance, if Thursday and Saturday are holidays, we consider Friday as a bridging holiday. To calculate average load for each category, for example, since there are eight days for weekends, we take the average load in each period from all weekends in January 2013, etc. It can be seen clearly that holiday and bridging holiday load patterns are totally different from regular any other load patterns. Consequently, we apply a weighted moving average method to replace the holidays and bridging holidays. Average load of N previous weeks of the day (d), at time period, t, is used to replace holidays and bridging holidays. Weighted moving average method is shown as below equation,

$$L_t(d) = w_1 \times L_t(d-7) + w_2 \times L_t(d-14)$$

The holiday's data and bridging holiday's data are replaced by selecting two days from recent previous two weeks. In the weighted moving average equation, we

Ref. code: 25605822043898AVW

give the weight w_1 as 0.7 for the first previous week and the weight w_2 as 0.3 for the previous two weeks.

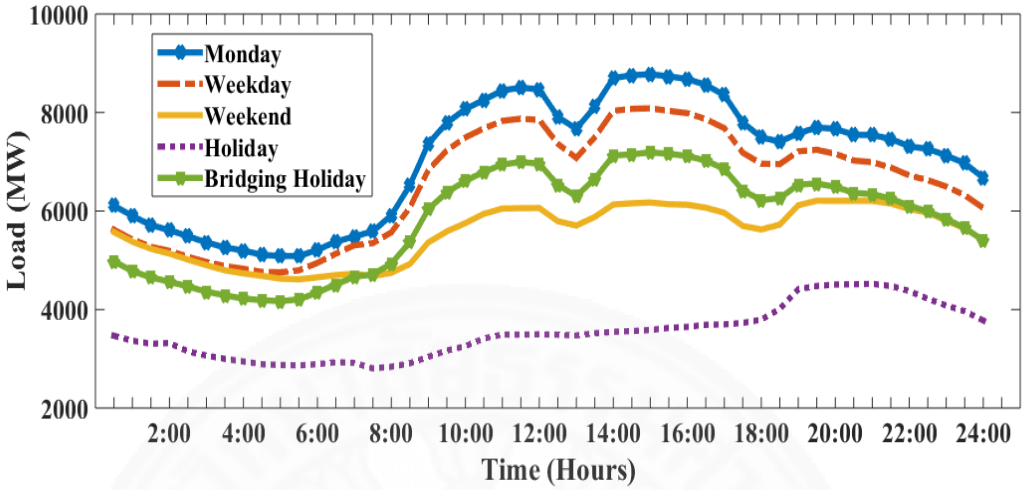

Figure 4.1: Load patterns for five categories

4.1.2 Outliers Detection and Replacement

Furthermore, we detect outliers by using time-window based filtering band as there is a similar pattern on the same time period and the same day of week. We arrange the dataset consisting of the same weekday and same time period to construct the filtering band of each weekday and each time period. We construct time-window based filtering band by using four weeks moving average and standard deviation using same time period and same day of week. After that, all of the data outside the filtering band are regarded as outliers and replaced by two weeks moving average. To regard the outliers, we use the following equation.

$$L_t^{'}(d) = \begin{cases} w_1 L_t(d-7), + w_2 L_t(d-14), & if \;\; L_t(d) \notin B_t(d) \\ L_t(d), & Otherwise \end{cases}$$

The time window for time *t*, of day d for *p* weeks is given as:

$$V_t(d) = L_t(d^{'}), L_t(d^{'}-7), ..., L_t(d^{'}-7\times p),$$

where d' refers the last 7 days in the selected data set and the variable *p* represents different days of the weeks which have different number of weeks within the sample data set.

The k-period filtering band is written as:

Ref. code: 25605822043898AVW

$$B_t(d) = \frac{[\sum_{i=1}^{K} L_t(d - 7 \times i)]}{K} \pm N \times SD(V_t(d)),$$

where N denotes the width of the k-period filtering band. In this research, the variable N is set to be 1.6 which is optimum value.

After detecting outliers using the k-period time window filtering band, they are replaced by moving average method. The detected outliers are replaced by using the average load data from similar time periods from similar days of previous two weeks.

Moving average method for replacing outliers:

$$L_t(d) = \frac{\sum_{i=1}^{N} L_t(d - 7 \times i)}{N}$$

After cleansing the original data, they are ready to train the model. The data arrangement of using cleaned data for all models and training algorithms are discussed in the following sections.

Ref. code: 25605822043898AVW

4.2 Deep Neural Network

4.2.1 Data Arrangement

There are two types of data set, i.e., training and testing data sets. Each data also includes a pair of input and output. The data from 1st May 2012 to 31st May 2013 is selected as one-year training data set to train the model and forecast each day in 2013. Therefore, there are 388 pairs of training data set to forecast one day. There are 48-time periods in one day. Consequently, the network has to be trained 388*48 times to predict each day in 2013.

Table 4.1: Example of data arrangement of training and testing pair for 1st May 2013 in 1-period forecasted model

			Input					**Target**
Training Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*DoW*	*MoY*	$F_t(d)$
	1	01/05/12 (Tue)	07/05/12 (Mon)	07/05/12 (Mon)	08/05/12 (Tue)	1	5	08/05/12 (Tue)
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	388	23/05/13 (Fri)	29/5/13 (Thurs)	29/5/13 (Thurs)	30/05/13 (Fri)	4	5	30/05/13 (Fri)
			Input					**Output**
Testing Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*DoW*	*MoY*	$F_t(d)$
	1	24/04/13 (Wed)	30/04/13 (Tue)	30/04/13 (Tue)	01/05/13 (Wed)	2	5	01/05/13 (Wed)

The simplified forecasting equation for the model is;

$$F_t(d) = a_1 L_t(d-1) + a_2 L_t(d-7) + a_3 T_t(d-1) + a_4 T_t(d) + DoW + MoY$$

where,

$F_t(d)$ = Forecasted load at period *t*,

$L_t(d-1)$ =Yesterday load at period *t*,

$L_t(d-7)$ = Previous week same day load at period *t*,

$T_t(d-1)$ = Yesterday's temperature at period *t*,

$T_t(d)$ = Forecasted day's temperature at period *t*,

t =1, 2, 3, …, 48 periods, and $a_1, \ldots, a_4$ = coefficients of load and temperature.

Ref. code: 25605822043898AVW

DoW is the day of week, for instance, we put input as 1 for Monday, 2 for Tuesday and so on. Likewise, *MoY* is the month of year, for example, input is 4 for April, 5 for May, etc. Therefore, this paper uses six inputs training data set to test load demand on the networks. Table 4.1 is given as the example of the data arrangement for training data set and testing data set pair for 1st May 2013. Similarly, Fig. 3 illustrates the 48-periods forecasting structure, where there are 194 inputs to forecast for 48 forecasted outputs for one day (because the data is collected every 30 minutes for one day).

4.2.2 Study of Different Forecasting Structures for DNN

We propose two forecasting structures of deep learning to predict the load demand. These are 1-period forecasting structures and 48-periods forecasting structure. In the 1-period forecasting structure, there are six inputs to forecast one period in each model whereas in the 48-period forecasting structure, there are 194 inputs to forecast for 48 forecasted outputs for one day. Thus, there are 48 deep learning models to forecast one day for 1-period forecasting structure. However, there is only one deep learning model to forecast one day for 48-period forecasting structure. Figure 4.2 show the 1-period forecasting structure of the proposed deep learning model. The inputs are yesterday load, previous week same day load, yesterday temperature, forecasted day temperature, the day of week, and the month of year. Similarly, Figure 4.3 illustrates the 48-periods forecasting structure, where there are 194 inputs to forecast for 48 forecasted outputs for one day (because the data is collected every 30 minutes for one day).

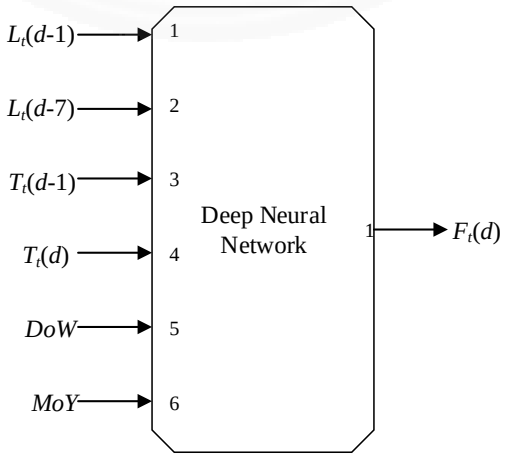

Figure 4.2: 1-Period Forecasting Structure

Ref. code: 25605822043898AVW

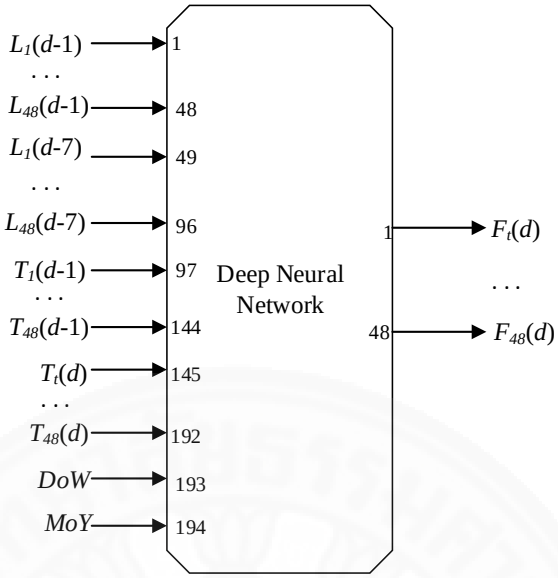

Figure 4.3: 48-Periods Forecasting Structure

Both proposed forecasting structures are tested with 100, 200, and 300 hidden layers. After forecasting by using the proposed model, we need to calculate the error to check how many the load demand deviates from the actual data. Normally, most researchers use mean absolute percentage error (MAPE) to compute the forecasting accuracy. However, in this study, MAPE' is used to measure the error between the cleaned load and the forecasted load. The following equation is utilized to get MAPE' for monthly load predictions.

For the mean absolute percentage error,

$$MAPE' = \frac{1}{t}\sum_{t=1}^{48}\left|\frac{L_t'(d) - F_t(d)}{L_t'(d)}\right| \times 100\%,$$

where,

L_t' = the cleaned load at period t,

F_t = the forecasted load at period t,

t = the time period for one day (t = 1, 2, …, 48).

Ref. code: 25605822043898AVW

4.3 Study of Different Forecasting Models

4.3.1 Data Arrangement for ANN, SVM, and DNN_1

The data is separated into two datasets, i.e., training and testing datasets. Both datasets are arranged as a pair of input and target to train and test the models. In this paper, we select one-year training dataset from 7th May 2012 to 30th May 2013 to train the models. Thus, there are 388 days in our training dataset. As a result, all models have to be trained with 388 datasets to test on one-day forecast.

In DNN, ANN, and SVM models, there are six input variables as yesterday load, previous week same day load, yesterday temperature, forecasted day temperature, day of week (*DoW*) and month of year (*MoY*). The basic forecasting equation is given by;

$$F_t(d) = b_1 L_t(d-1) + b_2 L_t(d-7) + b_3 T_t(d-1) + b_4 T_t(d) + DoW + MoY$$

where,

$F_t(d)$ = Forecasted load at period t for day d,

$L_t(d\text{-}1)$ = Load at period t for day d-1 (yesterday load),

$L_t(d\text{-}7)$ = Load at period t for d-7 (previous week same day),

$T_t(d\text{-}1)$ = Temperature at period t for d-1 (yesterday temperature),

$T_t(d)$ = Forecasted day temperature at period t for day d,

DoW = Day of Week (1, 2, …, 7),

MoY = Month of Year (1, 2, …, 12),

t =1, 2, 3, …, 48 periods,

$b_1, \ldots, b_5$ = coefficients of load and temperature.

According to the above equation, all models take six inputs to forecast the next day load demand. Table 4.2 shows that there are 388 pairs of training dataset from May 1 2012 to May 30 2013 to forecast June 1 2013. This data arrangement is noted as everyday training dataset. After completing the training, each model will be tested by one testing dataset. In this case, the target is to forecast June 1 2013. Once we complete the testing dataset, the models are trained with new rolling 388 pairs of dataset and they are tested to the next forecasting day, June 2 2013.

Ref. code: 25605822043898AVW

Table 4.2: Everyday training data arrangement for testing target 1st June 2013

			Input					Target
Training Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*DoW*	*MoY*	$F_t(d)$
	1	01/05/12 (Tue)	07/05/12 (Mon)	07/05/12 (Mon)	08/05/12 (Tue)	1	5	08/05/12 (Tue)
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	388	23/05/13 (Fri)	29/5/13 (Thurs)	29/5/13 (Thurs)	30/05/13 (Fri)	4	5	30/05/13 (Fri)
			Input					**Output**
Testing Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*DoW*	*MoY*	$F_t(d)$
	1	25/05/13 (Sat)	31/05/13 (Fri)	31/05/13 (Fri)	01/06/13 (Sat)	5	6	01/06/13 (Sat)

4.3.2 Data Arrangement for DNN_2

In addition, we propose another data arrangement with five input variables. This data arrangement is called the same day training dataset and it is used with DNN. For this DNN model, we selected five input variables as yesterday load, previous week same day load, yesterday temperature, forecasted day temperature, and month of year (*MoY*). The basic forecasting equation is written as;

$$F_t(d) = b_1 L_t(d-1) + b_2 L_t(d-7) + b_3 T_t(d-1) + b_4 T_t(d) + MoY$$

where,

$F_t(d)$ = Forecasted load at period t for day d,

$L_t(d\text{-}1)$ = Load at period t for day d-1 (yesterday load),

$L_t(d\text{-}7)$ = Load at period t for d-7 (previous week same day),

$T_t(d\text{-}1)$ = Temperature at period t for d-1 (yesterday temperature),

$T_t(d)$ = Forecasted day temperature at period t for day d,

MoY = Month of Year (1, 2, …, 12),

t =1, 2, 3, …, 48 periods,

$b_1, \ldots, b_5$ = coefficients of load and temperature.

This second training dataset is arranged to use the same day for the target as shown in Table 4.3. In this table, all of the target is arranged to be only on Saturday.

Ref. code: 25605822043898AVW

The training dataset consists of 52 datasets. It also includes Friday load as a yesterday input when the model predicts Saturday load as shown in Table 4.3.

Table 4.3: Same day training data arrangement for testing target 12th Jan 2013

		Input					**Target**
Training Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*MoY*	$F_t(d)$
	1	07/01/12 (Sat)	13/01/12 (Fri)	13/01/12 (Fri)	14/01/12 (Sat)	1	14/01/12 (Sat)
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	52	22/12/12 (Sat)	28/12/12 (Fri)	28/12/12 (Fri)	29/12/12 (Sat)	12	29/12/12 (Sat)
		Input					**Output**
Testing Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*MoY*	$F_t(d)$
	1	05/01/13 (Sat)	11/01/13 (Fri)	11/01/13 (Fri)	12/01/13 (Sat)	1	12/01/13 (Sat)

4.4 Study of Different Deep Learning Models

In this case, we use the 30 minutes new load data from Dec 2013 to July 2017 which has also been collected by the Electricity Generating Authority of Thailand (EGAT). The data is also separated into two datasets, i.e., training and testing datasets. Both datasets are arranged as a pair of input and target to train and test the models. In this paper, we select training dataset from December 2013 to July 2016 to train the models. As a result, all models have to be trained using two and half year's datasets to test on one-day forecast.

We propose deep neural network (DNN) and recurrent neural network (RNN) with long short-term memory (LSTM). In both models, there are five input variables as yesterday load, previous week same day load, yesterday temperature, forecasted day temperature, and month of year (*MoY*). For training and testing datasets, we apply training dataset from 2013 December to July 2016 and testing dataset from August 2016 to July 2017. Same day training datasets is used as the data structures and then we train the models using same day dataset to test same day. The training data arrangement for both models is show in Table 4.4.

Ref. code: 25605822043898AVW

Table 4.4: Same day training data arrangement for testing target 6th Aug 2017

		Input					**Target**
Training Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*MoY*	$F_t(d)$
	1	23/12/13 (Sat)	03/01/14 (Fri)	03/01/14 (Fri)	04/01/14 (Sat)	1	04/01/14 (Sat)
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	52	23/07/16 (Sat)	29/07/16 (Fri)	29/07/16 (Fri)	30/07/16 (Sat)	7	30/07/16 (Sat)
		Input					**Output**
Testing Dataset	No.	$L_t(d\text{-}7)$	$L_t(d\text{-}1)$	$T_t(d\text{-}1)$	$T_t(d)$	*MoY*	$F_t(d)$
	1	30/07/16 (Sat)	05/08/16 (Fri)	05/08/16 (Fri)	06/08/16 (Sat)	8	06/08/16 (Sat)

4.4.1 Training process in DNN model

In DNN training process, we separated the data into training and testing datasets. In here, we use the 48-period forecasting structure which has 193 inputs to forecast for 48 forecasted outputs for one day. There is only one deep learning model to forecast one day for the 48-period forecasting structure as shown in Figure 4.4. We trained the model passing through 100 hidden layers with rectified linear unit activation function. After performing the entire process, we train the networks to forecasting daily load.

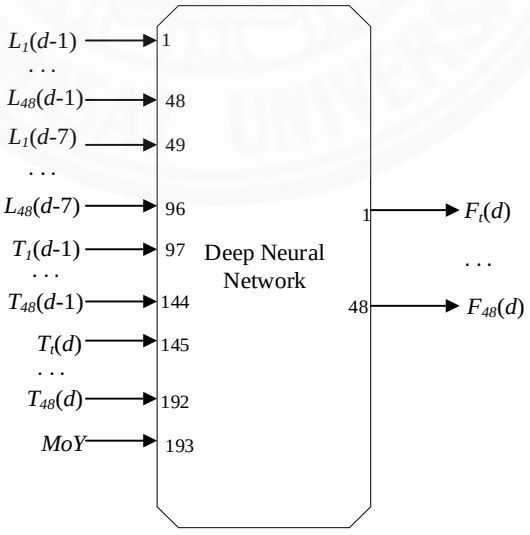

Figure 4.4: 48-periods DNN structure

Ref. code: 25605822043898AVW

4.4.2 Training process in RNN with LSTM

In RNN training process, we use all datasets from December 2013 to July 2017. At first, we split the data into training (70%) and testing (30%) individually. When activation functions, specifically the sigmoid or tanh are used in the training process, LSTMs are sensitive to the scale of the input data. Therefore, we rescale the data to the range of 0-to-1, also called normalizing. We can easily normalize the dataset using the *MinMaxScaler* preprocessing. In RNN with LSTM model, we need to look back to trace the memory of time series predictions. In this case, we set up 48 as look back at each time step to forecast load at each period. Figure 4.5 illustrates how to import the input in the RNN with LSTM model at each time step. After setting up the entire parameters, the model is ready to forecast load profile.

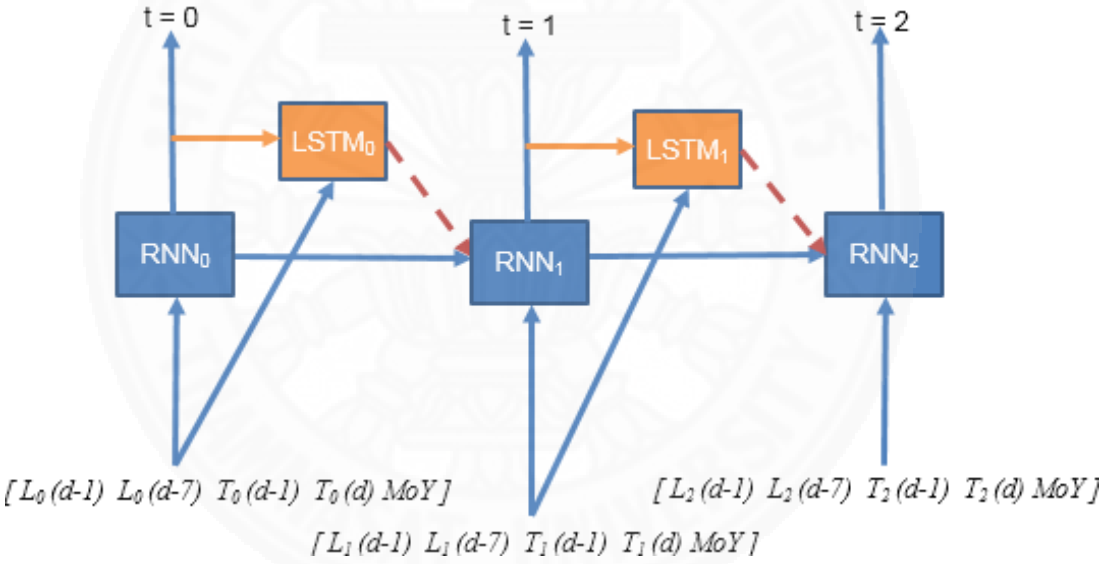

Figure 4.5: Load input data for RNN at each time step

Ref. code: 25605822043898AVW

Chapter 5
Results and Discussion

5.1 Mean Absolute Percentage Error (MAPE')

The main objective of all models is to minimize forecasting errors. In this research, we use cleaned data instead of original data to compare with forecasted values for the errors. Mean absolute percentage error (MAPE') is the accuracy measurement between the cleaned load and the forecasted load.

Mean absolute percentage error (MAPE'),

$$MAPE' = \frac{1}{t}\sum_{t=1}^{48}\left|\frac{L_t'(d) - F_t(d)}{L_t'(d)}\right| \times 100\%,$$

where,

$F_t(d)$ = Forecasted load at period t for day d,

$L_t'(d)$ = Cleaned load at period t for day d,

t = 1, 2, 3, …, 48 periods.

5.2 Result for deep neural network

5.2.1 Case 1: Deep neural network

In this study, the proposed structures of deep learning are used with cleaned data instead of original data to train and test the network. Afterwards, the proposed models are used to forecast load for each day in 2013. The summarized results for each month is shown in Table 5.1. There are six categories of MAPE' including weekdays (WD), weekends (WK), holidays (H), bridging holidays (BH), Monday (Mon) and total average (TA) for both the structures.

It can be seen that MAPE' results for December are the highest for different categories because it has the lowest electricity consumption compared to other months. In addition, total averages for 1-period forecasting structure are better than those for 48-periods forecasting structure. Both the structures obtained higher MAPE' for weekends compared to weekdays while the MAPE' for holidays in both the

Ref. code: 25605822043898AVW

structures are similar to weekdays. Monday average results are the worst results among the six categories.

Similarly, the proposed model is used to predict load using different hidden layers (HLs) size. Most of the MAPE' results are similar for three different layers' size. There is a higher MAPE' for 100HLs with 1-period forecasting structure, but there is a higher MAPE' for 300HLs with 48-periods forecasting structure for holidays. MAPE' results for weekends are also not as good as that for weekdays and bridging holidays. If we increase more HLs size for 1-period forecasting structure, we get better performance for predictions. However, errors were steadily rising when we enhance HLs for 48-periods forecasting structure.

The proposed model is also used to predict load using different hidden layers (HLs) size. Based on the experimental results, we found that when the number of HLs is increased, the MAPE' is slightly increased. Therefore, prediction result does not only depend on the number of HLs and the number of neurons. It may also depend on other factors, e.g., activation functions, learning algorithms and selection of input variables. Hence, a fine-tuning of parameters might be required.

Table 5.1: Monthly MAPE' for 48-period and 1-period Forecasting Structures with 100HLs, 200HLs & 300HLs

#HLs	**Month**	**48-periods forecasting structure**						**1-period forecasting structure**					
		WD	*WK*	*H*	*BH*	*Mon*	*TA*	*WD*	*WK*	*H*	*BH*	*Mon*	*TA*
100	Jan.	3.4046	7.6559	2.0878	-	6.6207	4.8742	2.5667	3.2849	2.0566	-	3.7512	2.8884
	Feb.	2.8766	7.2087	5.3490	2.5459	7.3890	4.6743	2.1164	3.8291	2.9940	3.2889	4.8777	2.9748
	Mar	3.2091	5.9577	-	-	5.6893	4.4158	2.2313	2.7866	-	-	3.4279	2.5648
	Apr	3.2936	5.2634	4.4016	3.0580	5.8884	4.1321	2.8369	2.3475	2.6515	1.8318	5.1315	2.9080
	May	3.2268	5.5091	4.7213	5.0233	4.3549	4.2514	3.0134	3.4442	3.0981	3.5437	2.3554	3.1715
	Jun	2.8833	4.5394	-	2.8860	5.0434	3.7235	2.9571	2.7828	-	1.4880	3.4678	2.9181
	Jul	2.9826	6.1876	5.8875	2.5320	9.7082	4.7271	2.4768	3.7718	3.2900	3.5407	3.4963	3.0227
	Aug	2.7007	4.5059	2.7260	1.4332	9.2843	3.8219	2.1366	2.7742	1.4314	1.6161	2.7074	2.3374
	Sep	2.3499	5.1202	-	-	4.6338	3.5616	2.1508	2.7561	-	-	3.4975	2.5569
	Oct	3.2412	6.1803	1.9085	-	5.6843	4.3137	2.8674	3.6634	1.8215	-	4.6251	3.2659
	Nov	2.3451	6.1803	-	-	4.8273	3.8266	2.1873	3.9929	-	-	2.7575	2.8050
	Dec	8.0776	9.7391	12.411	8.5933	11.724	8.6787	6.9367	5.9039	5.6978	5.4356	10.833	6.3317
	Mean	3.3826	6.1706	4.9365	3.7245	6.7373	4.5834	2.8731	3.4448	2.8801	2.9635	4.2440	3.1454

Ref. code: 25605822043898AVW

200	Jan	3.7491	7.1804	2.1358	-	5.9980	4.8727	2.6233	3.1996	1.7952	-	3.9239	2.9131
	Feb	2.9756	6.6282	4.5213	2.4189	6.8570	4.4704	2.1415	3.6985	2.8276	2.8359	4.7308	2.9131
	Mar	3.2107	6.5518	-	-	5.6017	4.5970	2.2461	2.9000	-	-	3.3071	2.5939
	Apr	3.4454	6.2311	5.1460	3.5836	7.0235	4.6689	2.7402	2.1308	2.6770	1.9999	5.2347	2.8487
	May	3.2276	6.3494	4.8462	5.1828	4.7301	4.4944	2.9730	3.2199	3.8502	3.7499	1.8781	3.0738
	Jun	3.0377	5.6426	-	3.3601	5.6171	4.2607	2.9538	2.6971	-	1.9829	3.6635	2.9305
	Jul	3.1736	6.4512	6.2645	2.1982	11.528	5.0956	2.5178	3.8771	3.3731	3.3216	3.5525	3.0775
	Aug	3.0900	5.3117	4.0872	1.5096	11.659	4.5455	2.2152	2.7278	2.3941	1.2157	2.4619	2.3614
	Sep	2.4998	5.6176	-	-	5.3337	3.9075	2.1074	2.8976	-	-	3.2885	2.5413
	Oct	3.3963	6.6318	1.8778	-	5.7863	4.4907	2.6981	3.5903	1.5530	-	5.6010	3.2660
	Nov	2.4171	6.3946	-	-	5.7401	4.0534	2.1187	4.1187	-	-	2.9464	2.8291
	Dec	8.9863	10.049	11.455	10.466	9.9781	9.7567	7.3922	6.1908	5.3639	5.9837	12.315	6.6454
	Mean	3.6008	6.5867	5.0418	4.1028	7.1544	4.9345	2.8939	3.4374	2.9793	3.0128	4.4086	3.1662
300	Jan	3.6838	7.7800	1.7098	-	7.0086	5.1062	2.7053	3.4380	1.8879	-	4.0077	3.0361
	Feb	2.6381	7.1167	5.1293	2.8411	7.2801	4.5113	2.0326	3.7895	3.2200	3.0906	4.3729	2.8655
	Mar	2.8471	6.6579	-	-	5.4917	4.4176	2.1141	2.6546	-	-	3.1495	2.4221
	Apr	3.3300	6.1679	5.0988	3.1159	6.9532	4.5709	2.8173	2.5050	2.5826	2.1579	5.2071	2.9275
	May	3.3930	6.2335	4.9829	5.1020	4.6231	4.5510	3.0383	3.4274	3.0684	3.7555	2.5751	3.2086
	Jun	2.9992	5.6281	-	3.3727	5.6589	4.2426	2.8842	2.7659	-	1.3553	3.5827	2.8869
	Jul	3.1472	6.4986	5.5572	2.6229	11.583	5.0738	2.4809	3.8384	3.4998	4.0629	3.3095	3.0610
	Aug	3.1023	5.4598	3.8757	1.6219	11.390	4.5660	2.0827	2.6054	2.9488	1.8038	2.3133	2.2757
	Sep	2.6947	5.7028	-	-	5.3618	4.0416	2.0859	2.7904	-	-	3.2590	2.4928
	Oct	3.5489	6.9080	1.8215	-	5.8501	4.6569	2.6659	3.6310	1.9633	-	5.5500	3.2644
	Nov	2.6904	6.7970	-	-	5.7460	4.3298	2.1532	3.9735	-	-	3.0532	2.8193
	Dec	9.0272	11.025	13.398	9.8725	10.349	10.253	7.3330	6.8512	4.9387	5.7524	11.741	6.7312
	Mean	3.5918	6.8314	5.1967	4.0784	7.2746	5.0268	2.8661	3.5225	3.0137	3.1400	4.3434	3.1660

Ref. code: 25605822043898AVW

5.2.2 Case 2: Study of Different Forecasting Models

In this research, we use DNN model with two different data structures to predict daily load demand. The first one is noted as DNN_1 which is tested by using everyday training dataset. The second one is referred as DNN_2 which applies the same day training dataset to train. All models including ANN and SVM use the cleaned data to train and test. In addition, ANN and SVM are using everyday training dataset. Table 5.2 is the summarized monthly MAPE' outcomes using four different forecasting models for each month in 2013.

Table 5.2: Monthly MAPE' in 2013 for ANN, SVM, DNN_1, and DNN_2.

Month	ANN	SVM	DNN1	DNN2	Number of holidays
Jan	6.2233	6.8384	4.9290	4.1814	1
Feb	4.8546	4.7640	3.5650	4.9687	1
Mar	4.0913	4.6069	3.0386	3.7373	-
Apr	5.2078	5.8053	4.3674	3.4356	7
May	4.6311	5.3445	4.0751	4.0455	5
Jun	3.6040	4.6400	3.2080	3.8653	-
Jul	4.8488	5.8225	4.4771	4.1947	3
Aug	3.2853	3.8207	2.4050	3.7734	1
Sep	3.1977	3.9151	2.5414	4.0620	-
Oct	3.9449	4.3180	3.1084	4.4549	1
Nov	3.5938	4.1131	2.7328	4.5204	-
Dec	11.9294	13.1251	12.3886	6.8291	4
Total Average	4.9510	5.5928	4.2364	4.3390	

According to the Table 5.2, it is clear that the performance usage of DNN_1 is better than the ANN and SVM models as the results produced less MAPE'. The MAPE' of December is significantly higher than the rest of the month due to higher fluctuation in load. The forecasted result is particularly different from the actual load results due to the Christmas Season and the unexpected amount of tourist presence in Thailand. Figure 5.1 shows the monthly average load for October, November, and December. The average loads in December are lower than other two months because of the lowest average temperature. The variation still continues in the month of January with unexpected tourists and New Year Celebrations.

Ref. code: 25605822043898AVW

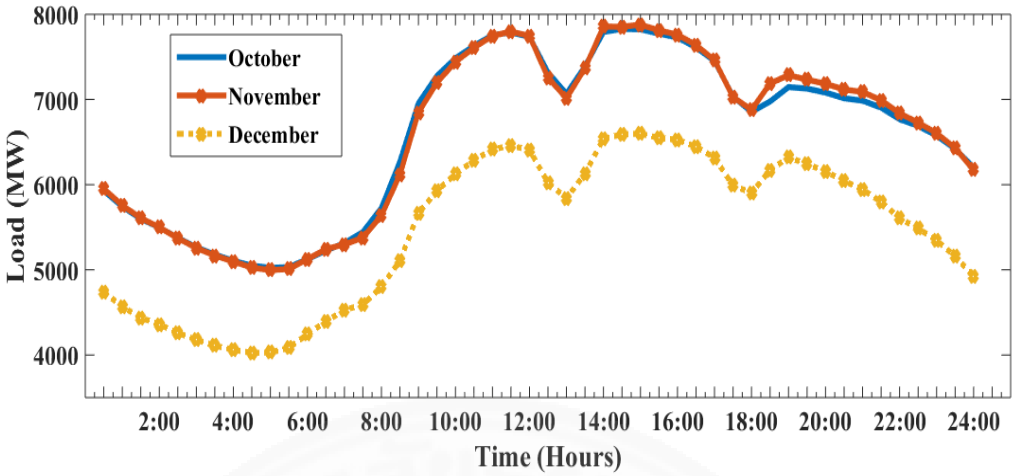

Figure 5.1: Comparison of average load curves for October, November, and December, 2013

Electricity consumptions are high from April to July due to high temperature. However, during the month of June, there are no holidays which results in similar pattern in the load. Fluctuations in the load are low during the months from August to November resulting in better forecasted results, thus low MAPE'. Comparing between two data arrangements, DNN_2 provides better accuracy than DNN_1 for months which there are many holidays; April, May, July and December. Consequently, using the same day training dataset is good for predicting loads in months which have many holidays whereas everyday training dataset gives better performance for other months. Moreover, MAPE' for DNN_1 has almost twice percent error of DNN_2 in December.

Table 5.3: Comparing MAPE' in 2013 for day type category of ANN, SVM, DNN_1 and DNN_2

	ANN	SVM	DNN_1	DNN_2
Weekdays	4.1471	4.3409	3.3206	5.2315
Weekends	4.8538	5.7487	4.2273	3.1795
Monday	5.6348	7.4658	4.5151	3.7454
Holidays	10.2910	11.6927	10.6844	3.7034
Bridging holidays	6.2243	6.4526	5.9337	3.1571
Total average	4.9649	5.6185	4.2544	4.3378

Furthermore, we summarize the MAPE' into six categories based on the load patterns including weekdays, weekends, Monday, holidays, bridging holidays, and total average to compare the results. According to Table 5.3, ANN, SVR and DNN_1

Ref. code: 25605822043898AVW

get highest errors for holidays and bridging holidays because these load patterns are different from normal weekday load. The results of MAPE' on Monday also have lower accuracy since we use Sunday as an input to forecast Monday. However, the load on Sunday are normally smooth and lower than the load on Monday. In addition to this, the results of MAPE' on weekends are generally worse than weekdays for ANN, SVM and DNN_1. This is also due to the same explanation that we use Friday as an input to forecast Saturday. By using the same day training dataset, it improves the forecasting accuracy for Monday, weekends, and bridging holidays.

5.2.3 Case 3: Effect of Different Training Datasets

The deep networks were trained by using different training datasets for 48-periods forecasting structure. In fig.9, horizontal axis represents date in May, while vertical axis denotes MAPE results by using different training datasets as same day and every day. Same day training datasets mean that only one-year Sunday datasets were used to predict Sunday in May. For example, if there are 51 Sunday in one year, we used 51pairs of training inputs and targets to predict 4 Sunday in May. If there are 55 Monday in one year, so we used 55 pairs for training and so on. On the other hand, everyday training datasets mean that the whole one year was used as training inputs and targets to predict all day in May. Results can be seen clearly in Table 5.4. From Table 5.4, we categorized as five groups: weekdays, weekends, holiday, Monday and total average. MAPE' results using same day data are significantly higher than using everyday data. As a result, forecasting using everyday training datasets can provide better performance for predictions.

Ref. code: 25605822043898AVW

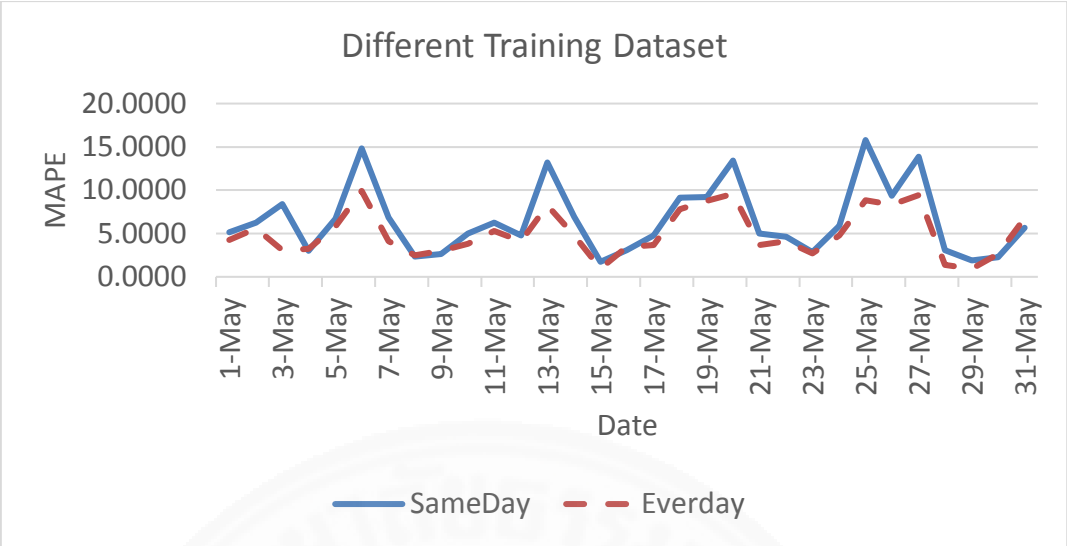

Figure 5.2: MAPE' Curves by using Same Day and Everyday Training Datasets for May

Table 5.4: MAPE' for Different Training Datasets

	Same Day Training Dataset	Everyday Training Dataset
Weekdays	4.3064	3.3650
Weekends	8.2199	6.6333
Holidays	9.1495	6.5979
Monday	13.6502	9.4789
Total Average	6.5741	5.0189

Moreover, we predicted May by using different monthly training datasets. Figure 5.3 indicates MAPE results for five categories which are already mentioned above by using different monthly training datasets. In here, the nets were trained by different training datasets such as two months, four months, six months, eight months and ten months respectively. As a result, performance might be depended on training dataset. According to Figure 5.3, results are good for holidays and Monday. According to total average, results are getting gradually higher by increasing training months. Therefore, performance results might be depended on training datasets for forecasting.

Ref. code: 25605822043898AVW

Figure 5.3: MAPE' Curves by using different monthly training datasets

Likewise, we predicted separate time period by using one-year separate time training datasets in May. We predicted load demand at 12:00AM, 11:00AM, 13:00PM and 17:00PM by using separate periods training datasets at 12:00AM, 11:00AM, 13:00PM and 17:00PM respectively. Prediction curves for those periods are shown in following figures individually. All figures indicate that prediction curves are almost nearly as target curves except at 12:00AM. Therefore, our forecasting performance might also be depended on time periods because time is one of external factors affecting on load demand.

Figure 5.4: Prediction Curve for 12:00AM in May 2013

Ref. code: 25605822043898AVW

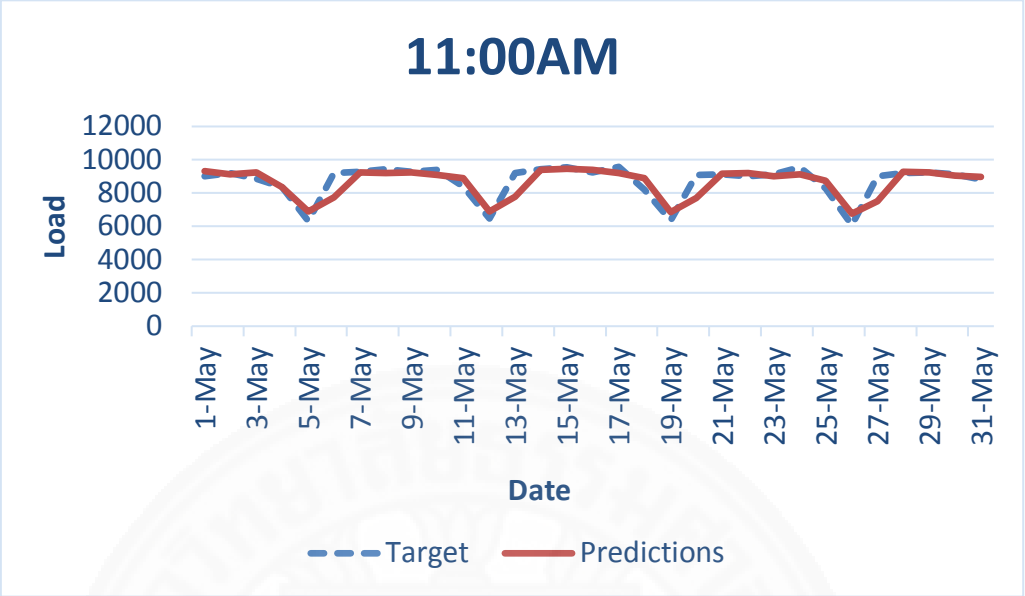

Figure 5.5: Prediction Curve for 11:00AM in May 2013

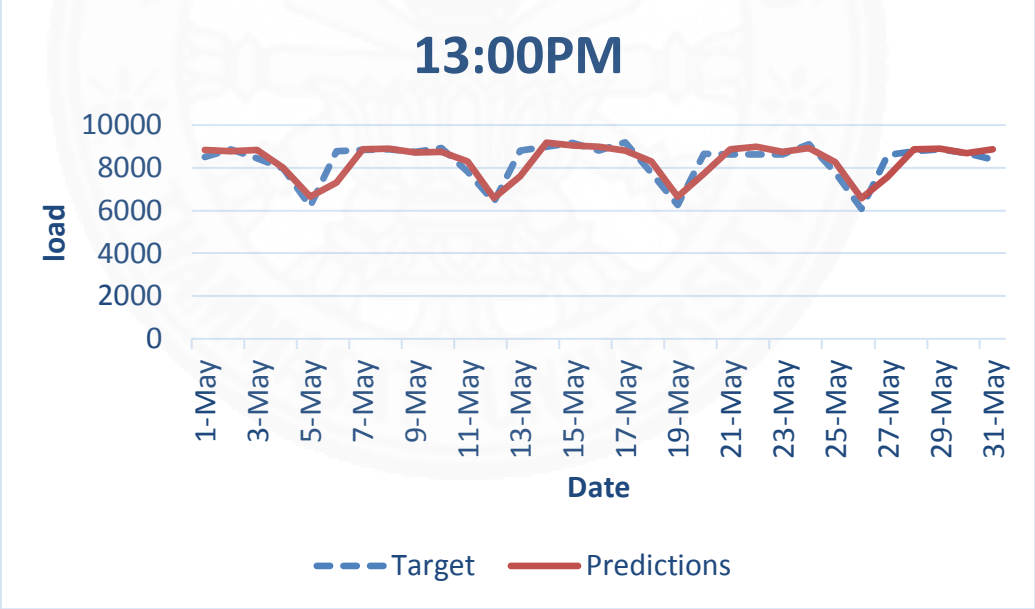

Figure 5.6: Prediction Curve for 13:00PM in May 2013

Ref. code: 25605822043898AVW

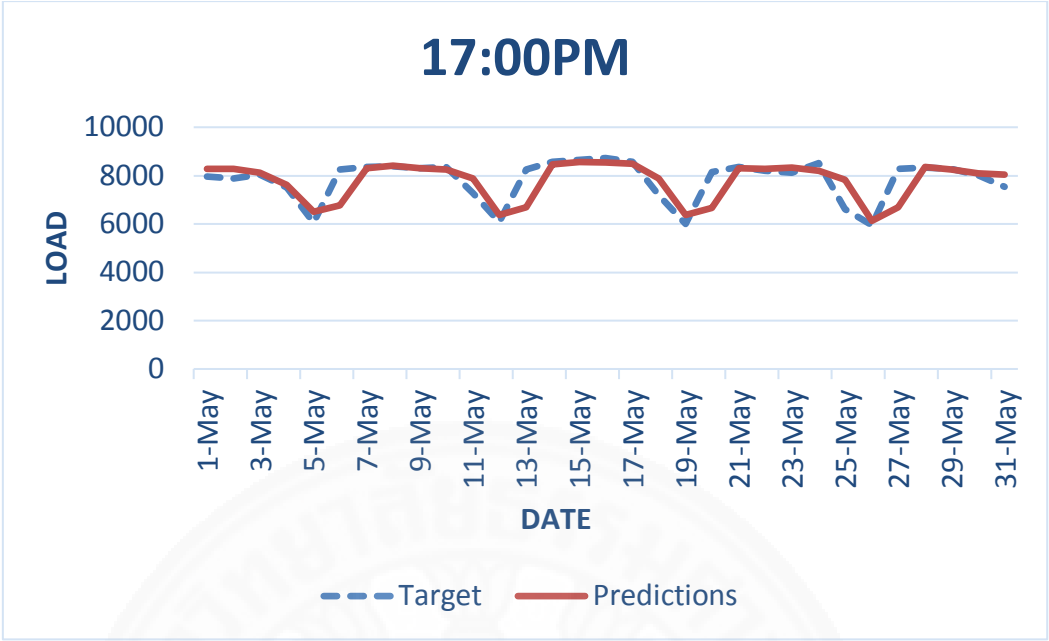

Figure 5.7: Prediction Curve for 17:00PM in May 2013

5.2.4 Case 3: Effect of Different Activation Functions

In our proposed model, we can also change many parameters during the training process. There are many activation functions to learn algorithms in the training process. These are sigmoid, tanh, rectified linear unit and drop out functions, etc. Some functions are good for regression, while some are good for classification. The networks can train by using hyper parameters and we used Rapidminer tool for deep neural network. In this tool, we can change parameters during training that it is shown in Figure 5.8.

Ref. code: 25605822043898AVW

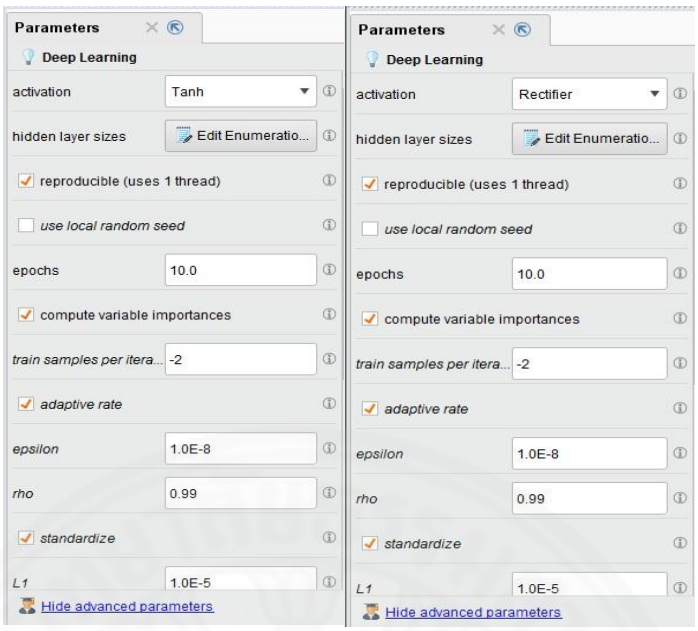

Figure 5.8: Available Parameters during Training Process

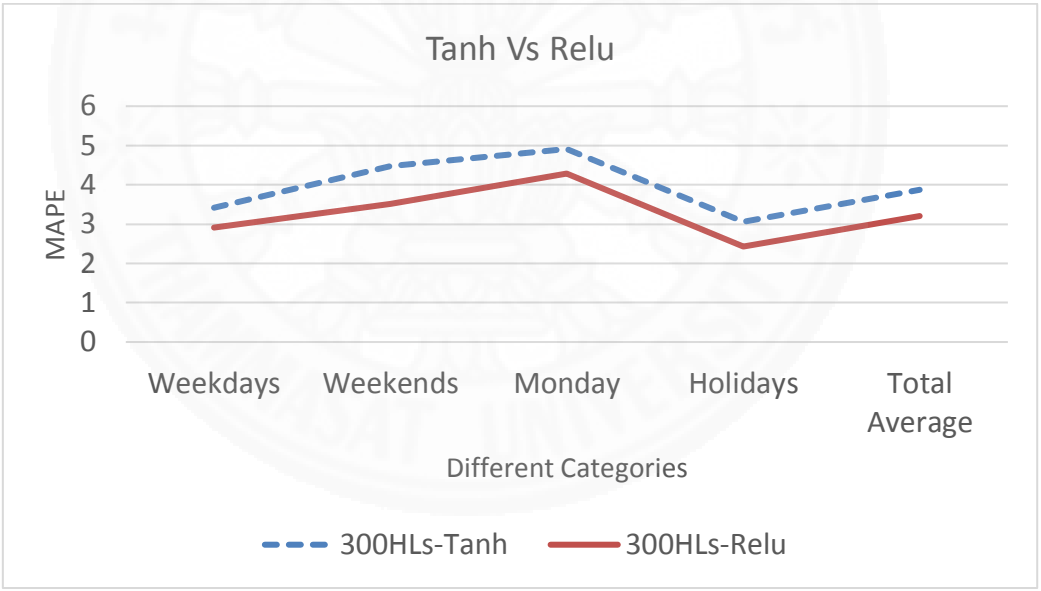

Figure 5.9: Prediction Curve using Different Activation Function

Ref. code: 25605822043898AVW

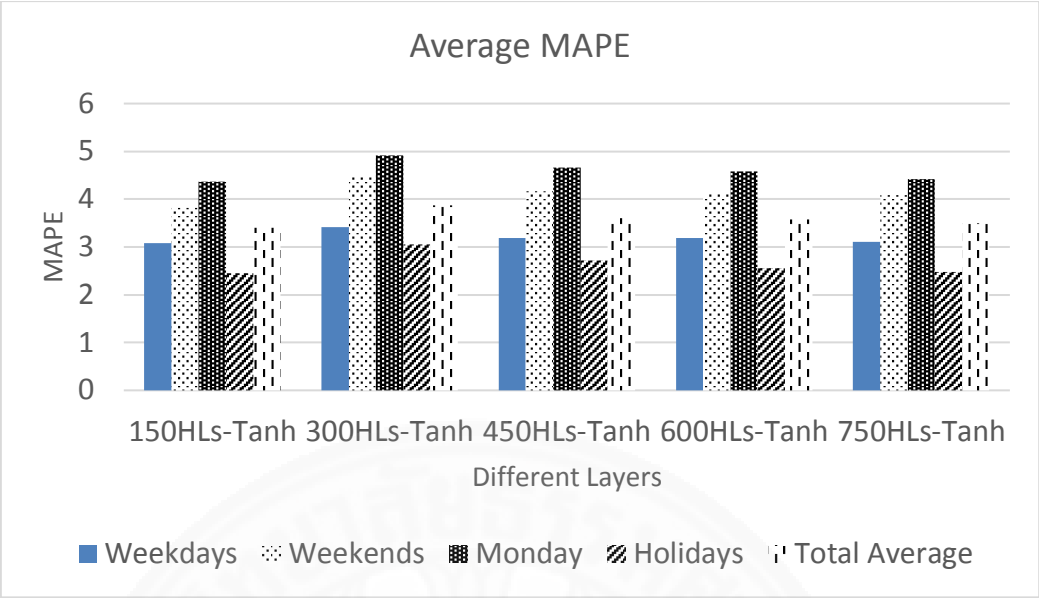

Figure 5.10: Prediction Curve for Different Hidden Layers using Tanh Activation Function

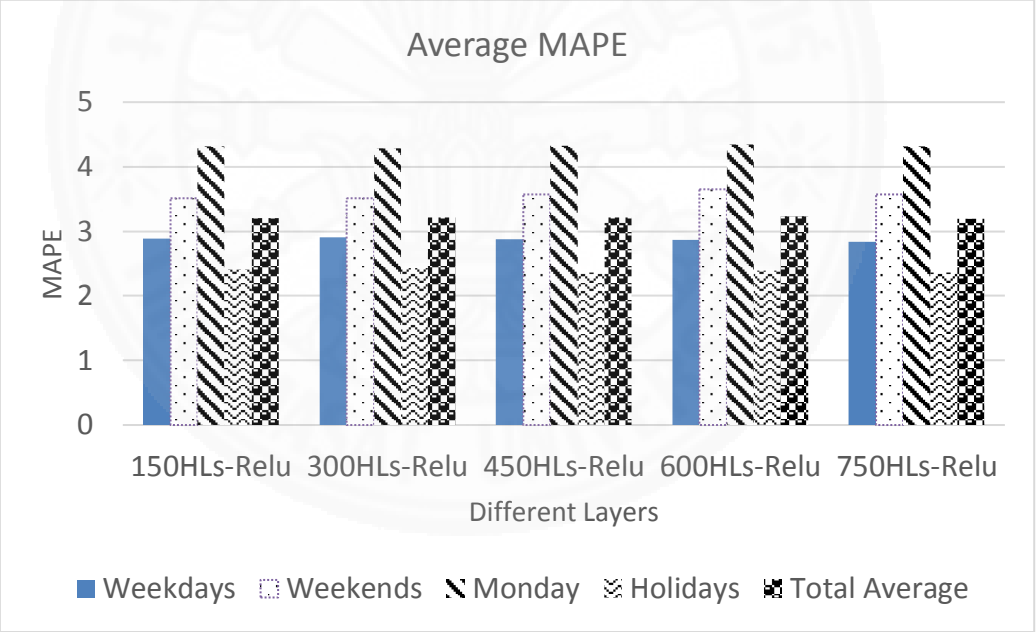

Figure 5.11: Prediction Curve for Different Hidden Layers using Relu Activation Function

Ref. code: 25605822043898AVW

5.3 Results for Recurrent Neural Network

In this study, the second proposed model of deep learning is recurrent neural network with long short-term memory. It is a great tool for modeling time series. Our original data have also been collected every 30minutes for 24hours. Therefore, RNNs seem to provide better performance for time series sequential data. During RNN training process, networks are used with cleaned data instead of original data to train and test the network. Afterwards, the proposed models are used to forecast load for each day in May 2013. The summarized results for each day are shown in the following figures individually. From all figures, horizontal axis and vertical axis represents time periods in minutes and load demand in MW. Time periods are depended on how many days in weekdays and weekends in May. For instance, there are four Monday, five Wednesday and four Sunday in May and so on. Prediction curves are divided into each day in weekdays and weekends.

At first, Figure 5.12 shows prediction curve for Monday. It can be seen clearly that Monday load predictions are totally different from actual load demands. The next Figure 5.13 is prediction curve for Tuesday which can predict nearly as actual load. Next, predicted loads for Wednesday and Thursday are almost similar as actual load that are shown in Figure 5.14 and Figure 5.16 individually, otherwise Friday prediction curve is a little bit different from original. Moreover, Figure 5.17 and Figure 5.18 presents weekend's prediction curves in May. Prediction curves in both Saturday and Sunday provides significantly overestimate of actual demand. As an overall result, load demands seem significantly dependent of different day of training and testing datasets.

Ref. code: 25605822043898AVW

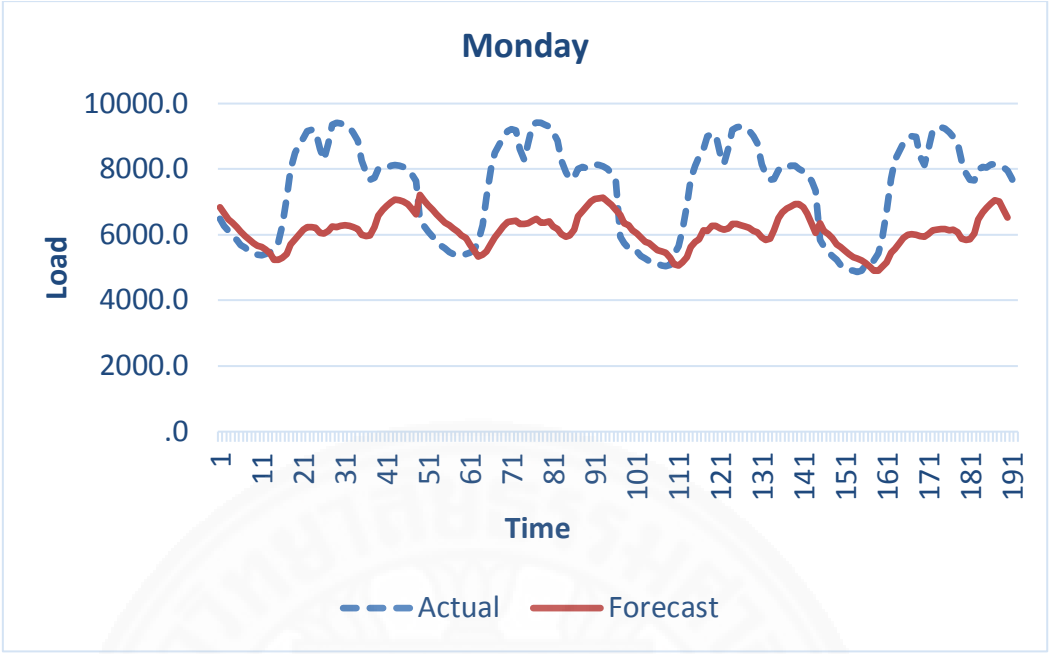

Figure 5.12: Prediction Curve for Monday in May 2013

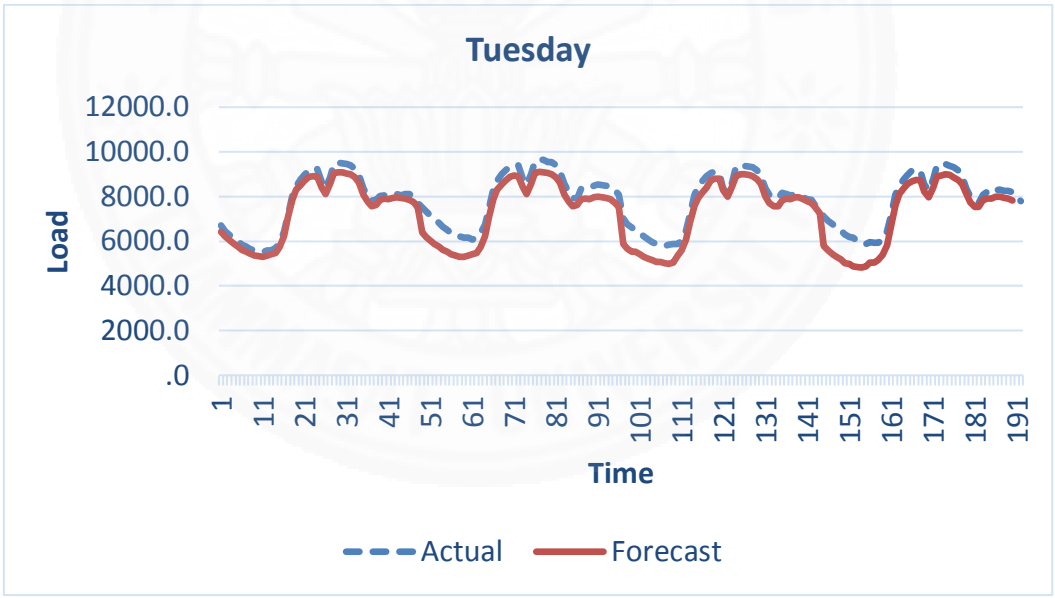

Figure 5.13: Prediction Curve for Tuesday in May 2013

Ref. code: 25605822043898AVW

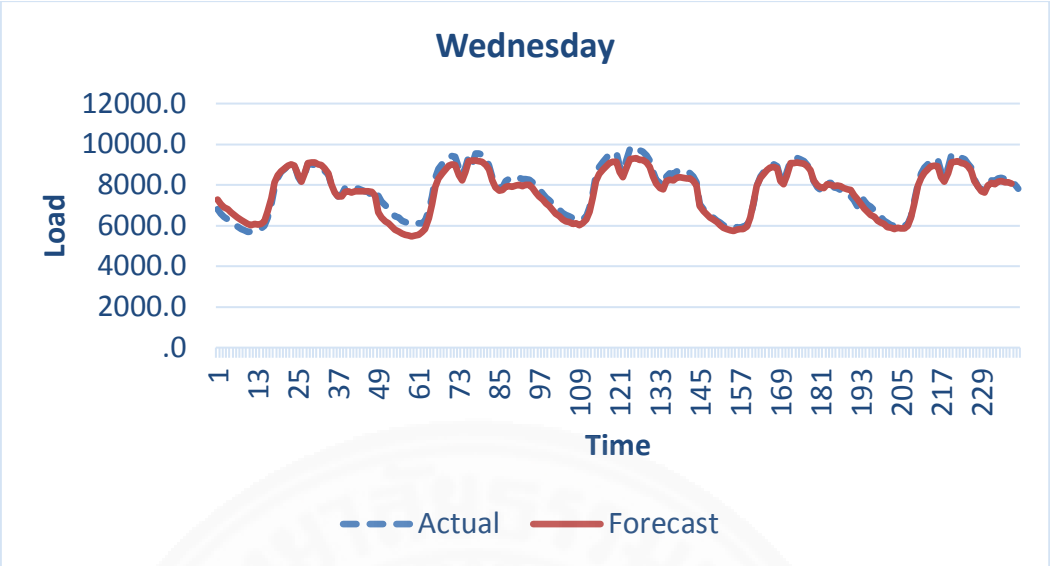

Figure 5.14: Prediction Curve for Wednesday in May 2013

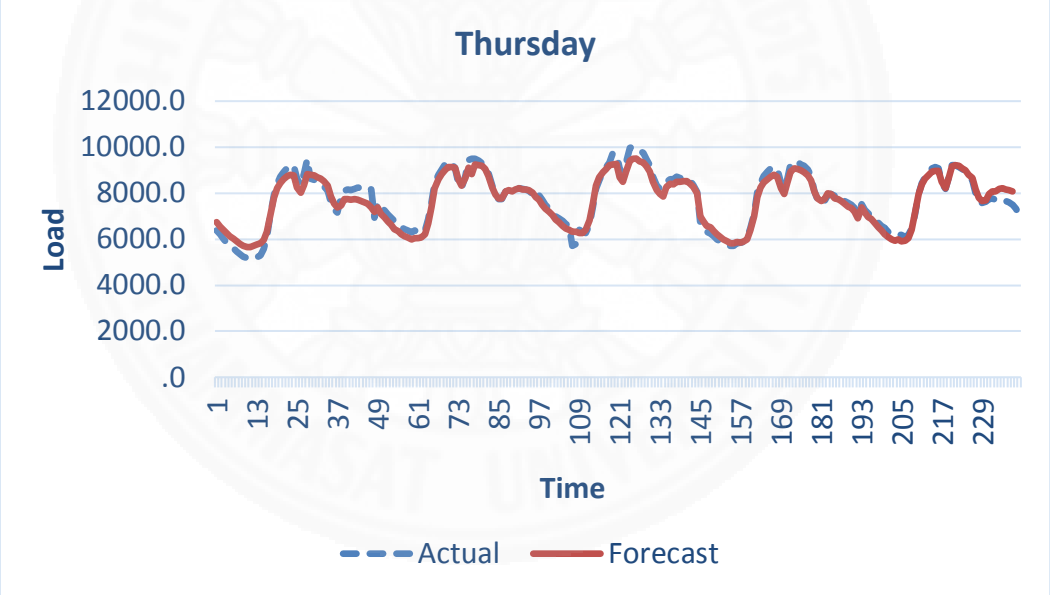

Figure 5.15: Prediction Curve for Thursday in May 2013

Ref. code: 25605822043898AVW

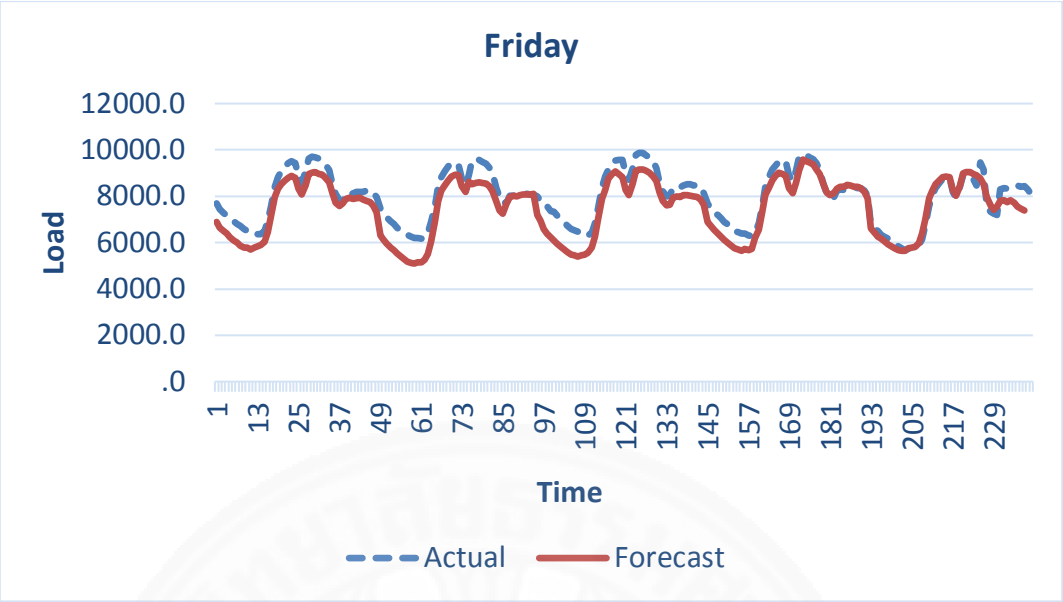

Figure 5.16: Prediction Curve for Friday in May 2013

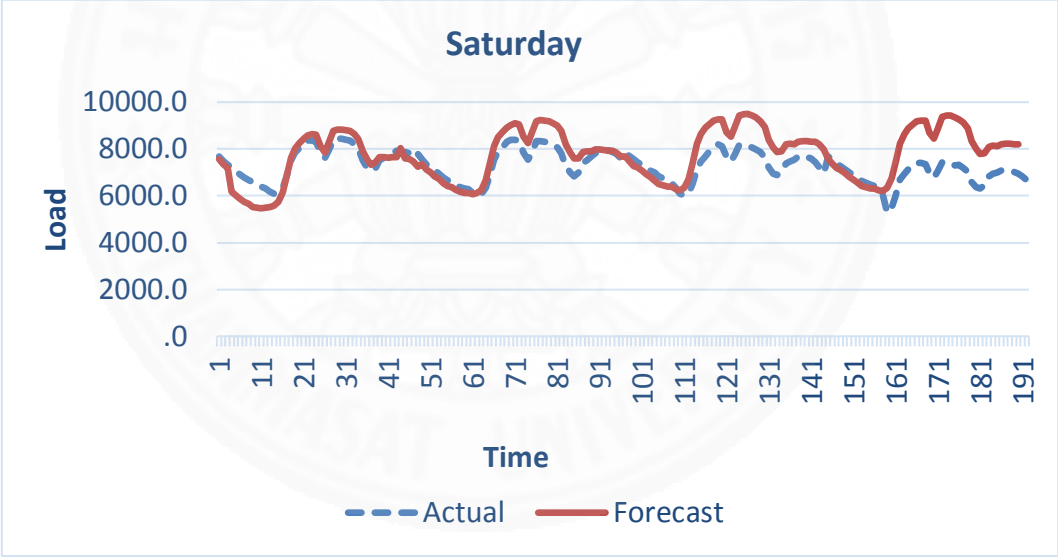

Figure 5.17: Prediction Curve for Saturday in May 2013

Ref. code: 25605822043898AVW

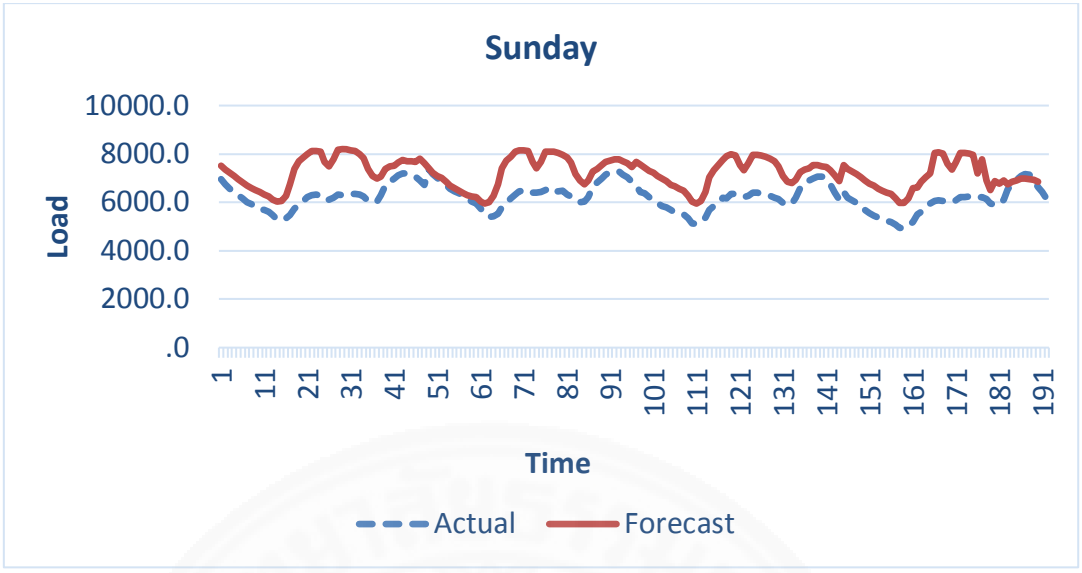

Figure 5.18: Prediction Curve for Sunday in May 2013

Similarly, we also predicted separate time period by using one-year separate time training datasets in May. We predicted load demand at 12:00AM, 11:00AM, 13:00PM and 17:00PM by using separate periods training datasets at 12:00AM, 11:00AM, 13:00PM and 17:00PM respectively. Prediction curves for those periods are shown in following figures individually. All figures indicate that prediction curves are almost nearly as target curves except at 12:00AM. Therefore, our forecasting performance might also be depended on time periods because time is one of external factors affecting on load demand.

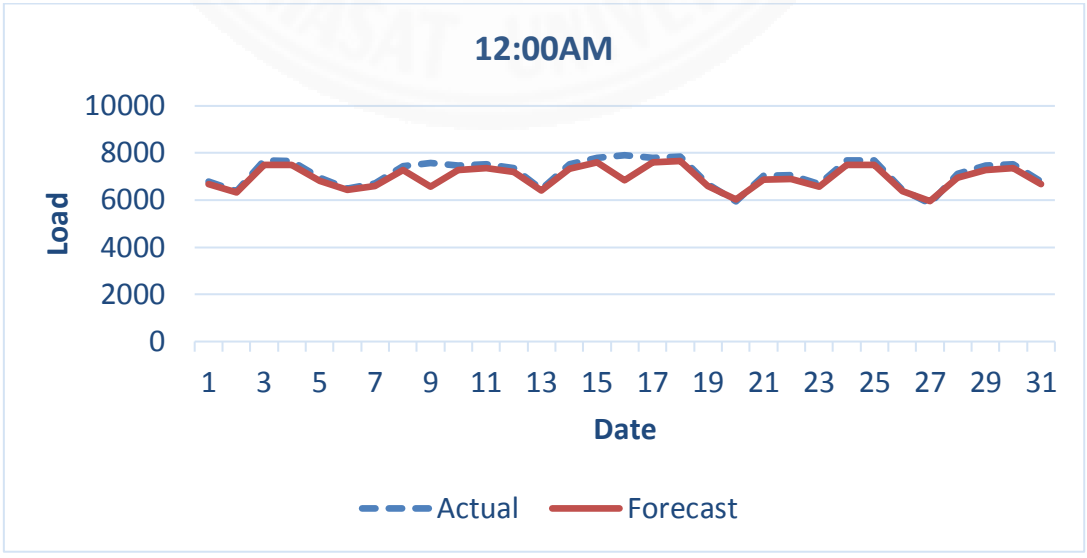

Figure 5.19: Prediction Curve for 12:00AM in May 2013

Ref. code: 25605822043898AVW

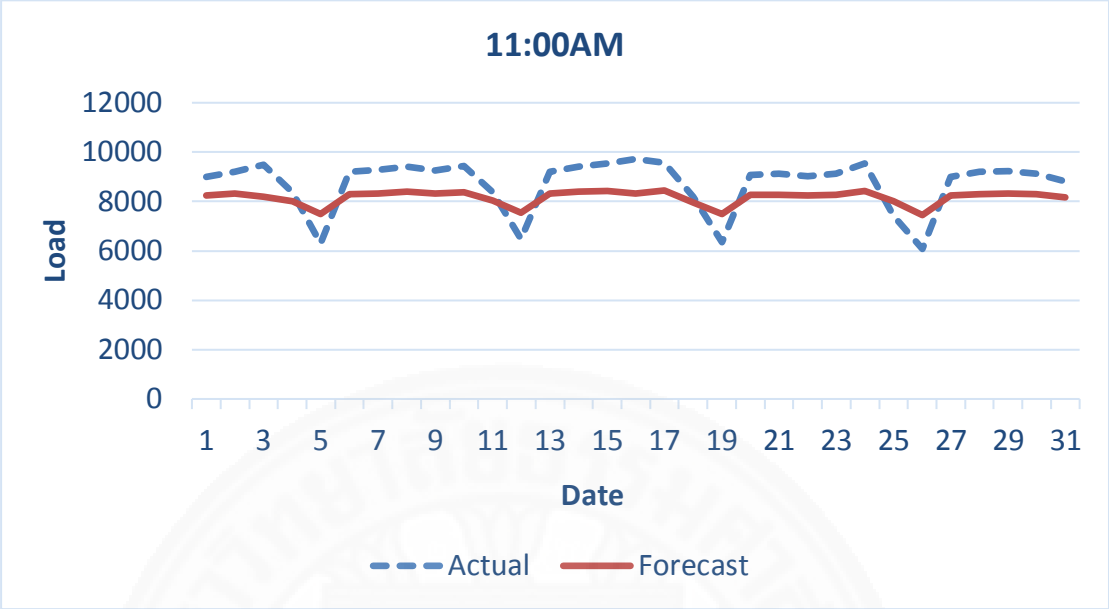

Figure 5.20: Prediction Curve for 11:00AM in May 2013

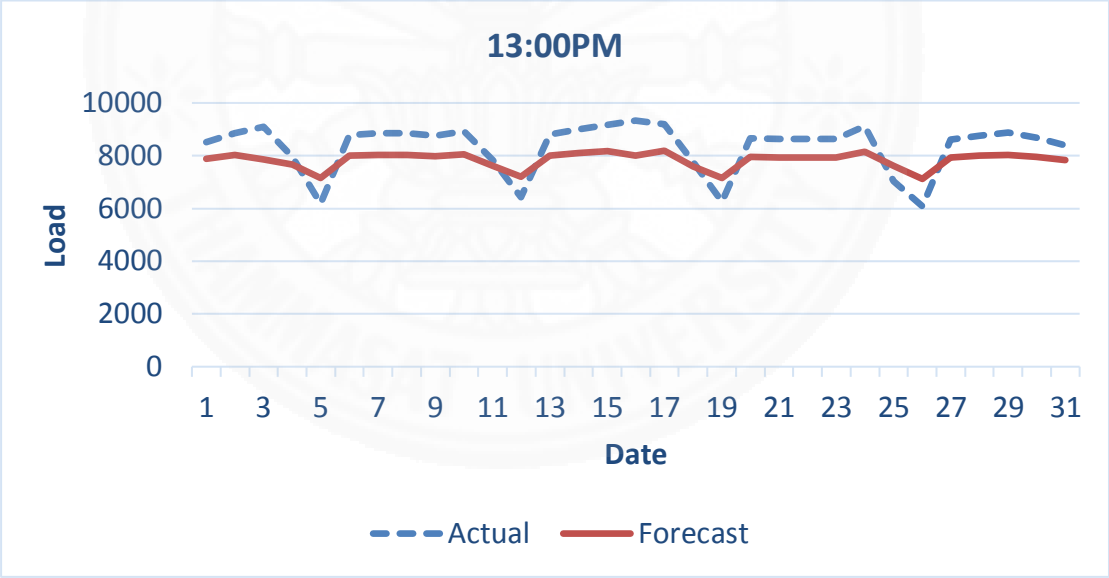

Figure 5.21: Prediction Curve for 13:00PM in May 2013

Ref. code: 25605822043898AVW

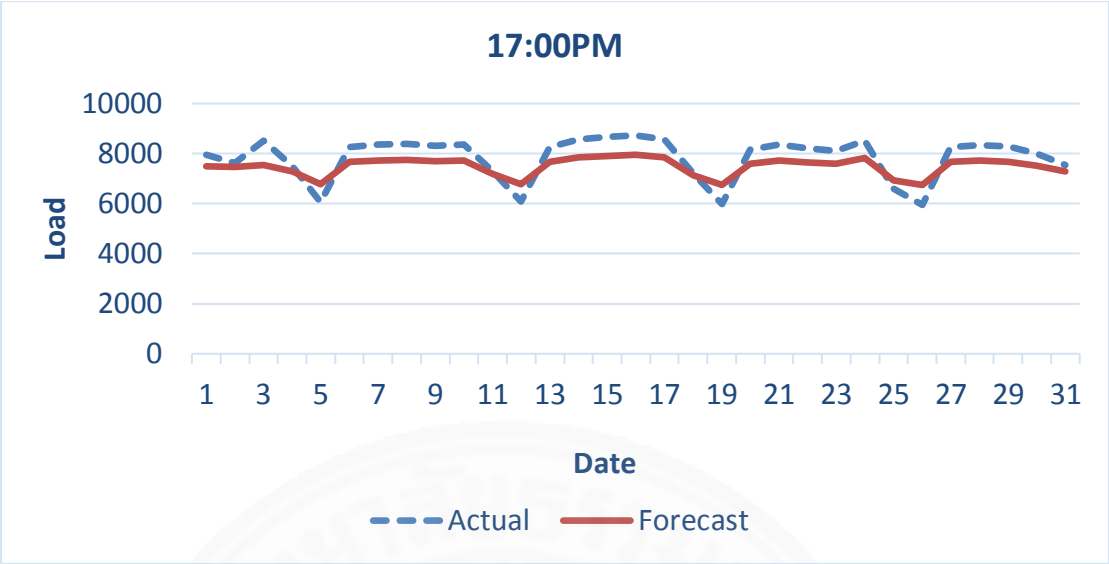

Figure 5.22: Prediction Curve for 12:00AM in May 2013

5.4 Study of Different Deep Learning Models

When we train both DNN and RNN with LSTM on new load and temperature data, both models provide sufficient performance for all months. According to the following result table, RNN with LSTM outperforms DNN because the data collected shows similarity with time series data collected for every 30 minutes and is shown in the Table 5.5 below. Though there are similar values for some months from both the models, the significant difference during certain months ensure the selection of RNN with LSTM is much better than the DNN model. Months like December and January, though showed errors with respect to the tourism influence, this proposed method clearly shows that the errors are reduced and the values are well presented. However, the month of April, shows similarity with results from DNN model along with the presence of seven holidays.

Ref. code: 25605822043898AVW

Table 5.5: Average Monthly MAPE' for DNN and RNN with LSTM

Months	DNN	RNN with LSTM
Aug, 2016	6.7081	5.5765
Sep, 2016	5.5765	4.6046
Oct, 2016	5.4661	5.1345
Nov, 2016	5.6945	5.2714
Dec, 2016	7.3852	5.0978
Jan, 2017	7.2926	5.1484
Feb, 2017	6.4061	6.1342
Mar, 2017	6.8167	5.1921
Apr, 2017	5.7507	5.7545
May, 2017	7.3764	5.9844
Jun, 2017	7.8781	5.8278
Jul, 2017	10.3636	9.1466

Ref. code: 25605822043898AVW

Chapter 6
Conclusion

In this research, we proposed the short-term electricity load forecasting using the deep neural network model. The original data from 2009 to 2013 is obtained from Electricity Generating Authority of Thailand (EGAT). The cleansed load data; temperature; day of week; and month of year are used as inputs. One-year training data sets are used to predict each daily load demand in 2013. We proposed two forecasting structures of deep neural network including 1-period forecasting structure and 48-periods forecasting structure with 100, 200, and 300 hidden layers and hidden nodes. The results show that deep neural network with 1-period forecasting structure provided higher accuracy than that with 48-periods forecasting structure. Finally, the proposed model provides more accurate load for time series predictions.

Furthermore, we utilize three powerful forecasting techniques: deep neural network (DNN), artificial neural network (ANN) and support vector machine (SVM) to overcome nonlinear problems in STLF. All techniques are trained and tested using cleaned load data; temperature; day of week; and month of year to predict daily forecasts in 2013. The outcomes of DNN are compared with ANN and SVM with everyday training dataset. The empirical results reveal that the proposed DNN model outperforms ANN model and SVM model.

There are two structures of training datasets, i.e., everyday training dataset and same day training dataset. All three models are trained with everyday training dataset that are used to predict each daily load demand for 2013. Moreover, the DNN model is also trained with the same day training dataset to compare the performance. The proposed DNN model with everyday training dataset obtains better forecasting performance compared to ANN model and SVM model for every month in 2013 except December. Furthermore, DNN model also performs better for weekdays, weekends, Monday, and bridging holidays. This empirical result shows that DNN provides a promising model for electricity load forecasting in electric power industry.

Additionally, we propose the DNN model using the same day training dataset to predict daily load. For this second training dataset, we use five inputs to train the model. After using same day training dataset, the DNN model improves MAPE'

Ref. code: 25605822043898AVW

results for bridging holiday, Monday, and weekends. Moreover, it is also give higher accuracy for January and December where they normally have lowest accuracy comparing to other months of the year. It also provides better accuracy for months with many holidays. The forecast of December gets highest MAPE' results since it has the lowest load demand because of the lowest temperature.

Finally, we also propose RNN with LSTM to test on new data from December 2013 to July 2017. After that, the outcomes of RNN with LSTM are compared to those of DNN. At that moment, RNN with LSTM outperforms DNN to solve time series problems for short-term load forecasting. For future research, different time series forecasting models or different arrangement of dataset may be applied to better forecasting performance.

Ref. code: 25605822043898AVW

References

Ao, L., Wang, Y., & Zhang, Q. (2017). New Zealand Journal of Agricultural Application of a hybrid model on short - term load forecasting based on support vector machines (SVM), *8233*(April 2015), 37–41. https://doi.org/10.1080/00288230709510324

Bala, A., Yadav, N. K., Hooda, N., & Registrar, D. (2014). Implementation of Artificial Neural Network for Short Term Load Forecasting. *Current Trends in Technology and Science*, *3*(4), 247–251.

Baniamerian, A., Asadi, M., & Yavari, E. (2009). Recurrent wavelet network with new initialization and its application on short-term load forecasting. *EMS 2009 - UKSim 3rd European Modelling Symposium on Computer Modelling and Simulation*, 379–383. https://doi.org/10.1109/EMS.2009.41

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT'2010*, 177–186. https://doi.org/10.1007/978-3-7908-2604-3_16

Chen, B.-J., Chang, M.-W. M.-W., & Lin, C.-J. C.-J. (2004). Load Forecasting Using Support Vector Machines: A Study on EUNITE Competition 2001. *IEEE Transactions on Power Systems*, *19*(4), 1821–1830. https://doi.org/10.1109/TPWRS.2004.835679

Connor, J., Martin, R., & Atlas, L. (1994). Recurrent Neural Networks and Robust Time Series Prediction. *Neural Networks, IEEE …*, *5*(2), 240–254. https://doi.org/10.1109/72.279188

Connor, J. T., Atlas, L. E., & Martin, D. (1991). Recurrent Neural Networks and Load Forecasting.

Dedinec, A., Filiposka, S., Dedinec, A., & Kocarev, L. (2016). Deep belief network based electricity load forecasting: An analysis of Macedonian case. *Energy*, *115*, 1688–1700. https://doi.org/10.1016/j.energy.2016.07.090

El-sharkh, M. Y. (2012). Forecasting Electricity Demand Using Dynamic Artificial Neural Network Model, 1691–1694.

Ertugrul, Ö. F. (2016). Forecasting electricity load by a novel recurrent extreme learning machines approach. *International Journal of Electrical Power and Energy Systems*, *78*, 429–435. https://doi.org/10.1016/j.ijepes.2015.12.006

Fard, A. K., & Akbari-Zadeh, M.-R. (2014). A hybrid method based on wavelet, ANN and ARIMA model for short-term load forecasting. *Journal of Experimental & Theoretical Artificial Intelligence*. Taylor & Francis. https://doi.org/10.1080/0952813X.2013.813976

Hatalis, K., Pradhan, P., Kishore, S., Blum, R. S., & Lamadrid, A. J. (2014). Multi-step forecasting of wave power using a nonlinear recurrent neural network. *PES General Meeting | Conference & Exposition, 2014 IEEE*, 1–5. https://doi.org/10.1109/PESGM.2014.6939370

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, *18*(7), 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527

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*,

Ref. code: 25605822043898AVW

16(1), 44–55. https://doi.org/10.1109/59.910780

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jeenanunta, C., & Abeyrathna, K. D. (2016). The Study of Artificial Neural Network Parameters for Electricity Forecasting, 1–7.

Kelo, S., & Dudul, S. (2012). A wavelet Elman neural network for short-term electrical load prediction under the influence of temperature. *International Journal of Electrical Power and Energy Systems*, *43*(1), 1063–1071. https://doi.org/10.1016/j.ijepes.2012.06.009

Kermanshahi, B. (1998). Recurrent neural network for forecasting next 10 years loads of nine Japanese utilities. *Neurocomputing*, *23*(1–3), 125–133. https://doi.org/10.1016/S0925-2312(98)00073-3

Krunic, M. S. S., Krcmar, I., & Rajakovic, N. (2000). An improved neural network application for short-term load forecasting in power systems. *Electric Power Components and Systems*, *28*(8), 703–721. https://doi.org/10.1080/07313560050082703

Li, P., Li, Y., Xiong, Q., Chai, Y., & Zhang, Y. (2014). Application of a hybrid quantized Elman neural network in short-term load forecasting. *International Journal of Electrical Power and Energy Systems*, *55*, 749–759. https://doi.org/10.1016/j.ijepes.2013.10.020

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, *234*(November 2016), 11–26. https://doi.org/10.1016/j.neucom.2016.12.038

Marvuglia, A., & Messineo, A. (2012). Using recurrent artificial neural networks to forecast household electricity consumption. *Energy Procedia*, *14*, 45–55. https://doi.org/10.1016/j.egypro.2011.12.887

Mcculloch, W. S., & Pitts, W. (1990). A logical calculus nervous activity. *Bulletin of Mathematical Biology*, *52*(l), 99–115. https://doi.org/10.1007/BF02478259

Mohandes, M. (2002). Support vector machines for short-term electrical load forecasting. *International Journal of Energy Research*, *26*(4), 335–345. https://doi.org/10.1002/er.787

Pai, P. F., & Hong, W. C. (2005). Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms. *Electric Power Systems Research*, *74*(3), 417–425. https://doi.org/10.1016/j.epsr.2005.01.006

Park, D. C., El-Sharkawi, M. a., Marks, R. J., Atlas, L. E., & Damborg, M. J. (1991). Electric load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, *6*(2), 442–449. https://doi.org/10.1109/59.76685

Qiu, X., Zhang, L., Ren, Y., Suganthan, P., & Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIEL 2014: 2014 IEEE Symposium on Computational Intelligence in Ensemble Learning, Proceedings*, (December). https://doi.org/10.1109/CIEL.2014.7015739

Rashid, T., Huang, B. Q., Kechadi, M. T., & Gleeson, B. (2006). Auto-regressive recurrent neural network approach for electricity load forecasting. *International Journal of Computational Intelligence*, *3*(1), 36–44.

Rodrigues, F., Cardeira, C., & Calado, J. M. F. (2014). The Daily and Hourly Energy Consumption and Load Forecasting Using Artificial Neural Network Method: A

Ref. code: 25605822043898AVW

Case Study Using a Set of 93 Households in Portugal. *Energy Procedia*, *62*, 220–229. https://doi.org/10.1016/j.egypro.2014.12.383

Rui, Y., & El-Keib, a. a. (1995). A Review of ANN-based Short-Term Load Forecasting Models The BP network structures. *System Theory, 1995., Proceedings of the Twenty-Seventh Southeastern Symposium*, 78–82. https://doi.org/10.1109/SSST.1995.390613

Sapankevych, N., & Sankar, R. (2009). Time series prediction using support vector machines: A survey. *IEEE Computational Intelligence Magazine*, *4*(2), 24–38. https://doi.org/10.1109/MCI.2009.932254

Siddarameshwara, N. (2010). Electricity Short term Load Forecasting using Elman Recurrent Neural Network. https://doi.org/10.1109/ARTCom.2010.44

Urban, G., Geras, K. J., Kahou, S. E., Aslan, O., Wang, S., Caruana, R., … Richardson, M. (2016). Do Deep Convolutional Nets Really Need to be Deep and Convolutional? https://doi.org/10.1038/nature14539

Vermaak, J., & Botha, E. C. (1998). Recurrent neural networks for short-term load forecasting. *IEEE Transactions on Power Systems*, *13*(1), 126–132. https://doi.org/10.1109/59.651623

Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep sparse rectifier neural networks. *In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315-323).

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *In Advances in neural information processing systems* (pp. 153-160).

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. *In Advances in neural information processing systems* (pp. 396-404).

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb), 1137-1155.

Ref. code: 25605822043898AVW

Appendix

Ref. code: 25605822043898AVW

Appendix A

Python code: RNN with LSTM

```
import numpy
import matplotlib.pyplot as plt
import pandas
import math
import pandas as pd

from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error


# convert an array of values into a dataset matrix

def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back-1):
        a = dataset[i:(i + look_back), :]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 4])
    return numpy.array(dataX), numpy.array(dataY)

# fix random seed for reproducibility
numpy.random.seed(7)


# load the dataset
dataframe = pandas.read_csv('TrainSun.csv', engine='python')
dataset = dataframe.values

#*********************************************

train_size = int(len(dataset) * 0.72)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
print('train',train)
print('test',test)


# reshape into X=t and Y=t+1
look_back = 48
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)

print('trainX',trainX)
print('trainY',trainY)
print('testX',testX)
print('testY',testY)

#*********************************************

# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)

# split into train and test sets
train_size = int(len(dataset) * 0.72)
test_size = len(dataset) - train_size
```


Ref. code: 25605822043898AVW

```
train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]

# reshape into X=t and Y=t+1
look_back = 48
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)

# reshape input to be  [samples, time steps, features]
trainX = numpy.reshape(trainX, (trainX.shape[0], look_back, 5))
testX = numpy.reshape(testX, (testX.shape[0],look_back, 5))

# create and fit the LSTM network
model = Sequential()
model.add(LSTM(4, input_shape=(look_back,5)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history= model.fit(trainX, trainY, epochs=100, batch_size=32)

# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)

# Get something which has as many features as dataset
trainPredict_extended = numpy.zeros((len(trainPredict),5))
# Put the predictions there
trainPredict_extended[:,4] = trainPredict[:,0]
# Inverse transform it and select the 5rd column.
trainPredict = scaler.inverse_transform(trainPredict_extended) [:,4]
print('trainPredict',trainPredict)
# Get something which has as many features as dataset
testPredict_extended = numpy.zeros((len(testPredict),5))
# Put the predictions there
testPredict_extended[:,4] = testPredict[:,0]
# Inverse transform it and select the 5rd column.
testPredict = scaler.inverse_transform(testPredict_extended)[:,4]
print('testPredict',testPredict)

trainY_extended = numpy.zeros((len(trainY),5))
trainY_extended[:,4]=trainY
trainY=scaler.inverse_transform(trainY_extended)[:,4]

testY_extended = numpy.zeros((len(testY),5))
testY_extended[:,4]=testY
testY=scaler.inverse_transform(testY_extended)[:,4]

# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY, trainPredict))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY, testPredict))
print('Test Score: %.2f RMSE' % (testScore))
print('train_size',train_size)
print('test_size',test_size)

# shift train predictions for plotting
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict)+look_back, 4] = trainPredict

# shift test predictions for plotting
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
```


Ref. code: 25605822043898AVW

```
testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, 4] =
testPredict

## convert your array into a dataframe
df = pd.DataFrame (trainPredict)
dfTest = pd.DataFrame (testPredict)

## save to xlsx file
filepath = 'trainPredictSunday.xlsx'
df.to_excel(filepath, index=False)
testFile = 'testPredictSunday.xlsx'
dfTest.to_excel(testFile, index=False)

#plot
serie,=plt.plot(scaler.inverse_transform(dataset)[:,4])
prediction_training,=plt.plot(trainPredictPlot[:,4],linestyle='--')
prediction_test,=plt.plot(testPredictPlot[:,4],linestyle='--')
plt.title('Electricity Load Forecasting')
plt.ylabel('Load')
plt.xlabel('Date')
plt.legend([serie,prediction_training,prediction_test],['serie','Training','
test'], loc='upper right')
plt.show()
```


Ref. code: 25605822043898AVW