

# **An Approach to Control Demand Response Load Shedding Based On Neural Network Load Forecasting**

DISSERTATION/THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE AWARD OF THE DEGREE  
OF

MASTER OF TECHNOLOGY  
IN  
ELECTRICAL ENGINEERING

Submitted by:

**SAMRAWIT TAREKEGN**

**2K17/C&I/19**

Under the supervision of

**Prof. A.K BHATTACHARYA**



**DEPARTMENT OF ELECTRICAL ENGINEERING**

**DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

2019

# **DEPARTMENT OF ELECTRICAL ENGINEERING**

## **DELHI TECHNOLOGICAL UNIVERSITY**

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

### **CERTIFICATE**

I, Samrawit Tarekegn Shiferawu, Roll No. 2k17/C&I/19 student of M. Tech. (Control and Instrumentation Engineering ), hereby declare that the dissertation/project titled “An Approach to Control Demand Response Load Shedding Based On Neural Network Load Forecasting” under the supervision of Prof. A.K Bhattacharya of Electrical Engineering Department Delhi Technological University in partial fulfilment of the requirement for the award of the degree of Master of Technology has not been submitted elsewhere for the award of any Degree.

Place: Delhi

**SAMRAWIT TAREKEGN**

Date: 24.06.2014

**Prof. A.K BHATTACHARYA**

Academic Designation

## ACKNOWLEDGEMENT

First my gratitude goes to my wonderful God who has done great things to me. Next with great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped me in making this thesis a great success. I am grateful to Prof. BHATTACHARYA, my advisor for his moral and technical support throughout the period of execution of the project.

## Candidate's Declaration

This paper is wholeheartedly dedicated to my dear Father MR **Tarekegn Shiferaw** and my dear Mother **Dagnaneshi Achame**, who have been my source of inspiration and gave us strength when I thought of giving up, who continually provide their moral, spiritual, emotional and financial support.

To my dear friend **Bright Mahere** who shared his words of advice and encouragement to finish this study.

## ABSTRACT

This paper describes a Neural Network application to reduce the computational complexities in a smart grid environment by predicting total demand and price for the next operating day. Short term load and price forecasting significantly reduce shortage risks and facilitates costs saving, hence improving the reliability and efficiency of Electrical Service Providers as well as efficiency for consumers.

Investigation of the development of electricity price and demand forecasting, with the emergence of demand response programs is required to design this system,. Short Term Load/Price Forecasting (STL/PF) is done for an electricity market that offers Demand Response (DR) Programs. For forecasting, we built a robust prediction model using Long Short Time Memory (LSTM) to ensure the generality of the results. The Australian National Electricity Market (ANEM), specifically Queensland State, is used as a subject case study.

The objective of the model is forecast total electrical load and price and used it to maximize the utility of the consumer subject to a minimum daily energy usage and minimum energy cost based on maximum and minimum hourly demand levels. Unknown price and electric load is forecast through recurrent neural network with a confidence interval. A simple bidirectional communication device between the power supplier and the consumer enables the functionality of the proposed model. The demand response affects the electricity price while the price affects the demands. Through demand response, the consumer can use electricity efficiently while the supplier can provide power at a lower cost.

# Contents

CERTIFICATE .....	i
ACKNOWLEDGEMENT .....	ii
Candidate's Declaration.....	iii
ABSTRACT .....	iv
CHAPTER 1 INTRODUCTION .....	- 1 -
1.1 General.....	- 1 -
1.2 Background of Project.....	- 2 -
1.3 Objective of Project .....	- 3 -
1.3.1 Specific objective .....	- 3 -
1.4 Problem statement.....	- 4 -
1.5 Scope of Project .....	- 5 -
1.6 Project Methodology .....	- 6 -
The methodology followed:.....	- 6 -
CHAPTER 2 LITERATURE REVIEWS.....	- 7 -
2.1 INTRODUCTION .....	- 7 -
2.2 Up to year 2020 load forecasting using neural nets.....	- 7 -
2.3 Comparison of Very Short-Term Load Forecasting Techniques .....	- 7 -
2.4 Long Term Probabilistic Load Forecasting and Normalization with Hourly Information.....	- 9 -
2.5 Practical Experiences with an Adaptive Neural Network Short-Term Load Forecasting System..	- 10 -
CHAPTER 3 BUILDING RNN FOR SHORT TERM LOAD FORECASTING .....	- 11 -
3.1 Introduction.....	- 11 -
3.2 The Overall System .....	- 11 -
3.3 Short-Term Load Forecasting .....	- 11 -
3.3.1 Benefits of STLF.....	- 12 -
3.4 Part one RNN intuition .....	- 13 -
What is Recurrent Neural Network? .....	- 13 -
Backpropagation through Time.....	- 14 -
3.4.1 Long-Short Term Memory .....	- 15 -
The Activation function.....	- 15 -
3.4.2 How neural network do learns .....	- 17 -
3.4.3 Neural network training.....	- 18 -
3.4.4 Neural Network Architectures for Electricity Load Forecasting .....	- 18 -
3.5 Part two: Data pre-processing.....	- 20 -

3.5.1 Load Data.....	- 22 -
3.5.2 Future Scaling .....	- 22 -
3.5.3 The number of time steps .....	- 23 -
Reshaping.....	- 23 -
3.6 Part three: Define Model .....	- 25 -
3.6.1 Initializing the RNN.....	- 26 -
3.6.2 Adding the LSTM layers and dropout regularization .....	- 26 -
3.6.3 Compile Model.....	- 28 -
3.6.4 Fitting the RNN to the Training.....	- 28 -
<b>Epochs</b> .....	- 29 -
<b>batch_size</b> .....	- 29 -
3.7 Part four Making the prediction and visualizing the result.....	- 29 -
What is the new variable called input contains: .....	- 30 -
3.8 Evaluate Model.....	- 30 -
Tie It All Together and Make Predictions.....	- 31 -
CHAPTER 4 LOAD SHEDDING SYSTEM DESIGN .....	- 34 -
4.1 INTRODUCTION .....	- 34 -
4.2 Load shedding algorithm.....	- 34 -
4.2.1 Control of shiftable loads .....	- 35 -
4.3 HDMS Algorithm .....	- 35 -
4.4.1 Categorization of Loads/Appliances for Scheduling .....	- 36 -
4.5 Domestic Shiftable Loads Operation with HDMS .....	- 38 -
4.5.1 The overall system design .....	- 38 -
• <b>Relay</b> .....	- 39 -
• <b>Opto-coupler</b> .....	- 39 -
4.6 Software section .....	- 40 -
4.6.1 Programming methods.....	- 41 -
CHAPTER 5 DISCUSSION AND CONCLUSION .....	- 42 -
5.1 Discussion.....	- 42 -
5.2 LSTM model evaluation.....	- 42 -
5.3 SHORT TERM LOAD AND PRICE FORECASTING RESULT .....	- 43 -
5.4 Load shedding simulation result.....	- 45 -
5.5 CONCLUSIONS .....	- 48 -
5.5.1 Model Improvement .....	- 48 -
5.5.2 Future scope .....	- 49 -
References .....	- 50 -

Appendix.....	- 52 -
Appendix A: <b>Time-variant electricity pricing</b> .....	- 52 -
Appendix B.....	- 53 -
Appendix C.....	- 53 -
Figure 1 Recurrent VS Feed Forward.....	- 14 -
Figure 2 sigmoid activation function.....	- 16 -
Figure 3 rectifier activation function.....	- 17 -
Figure 4 Neural Network Training.....	- 18 -
Figure 5 framework of NN prediction.....	- 19 -
Figure 6 sample training dataset.....	- 20 -
Figure 7 scatter plot of hourly load and temperature.....	- 21 -
Figure 8 Real Total Demand (MW).....	- 22 -
Figure 9 3D structure of X_train.....	- 24 -
Figure 10 RNN Training.....	- 31 -
Figure 11 imported data in python.....	- 32 -
Figure 12 Normalized output of X_train.....	- 33 -
Figure 13 Types of Loads.....	- 37 -
Figure 14 block diagram of load shedding.....	- 38 -
Figure 15 Relay Circuit.....	- 39 -
Figure 16 Optocoupler.....	- 40 -
Figure 17 flow chart for load shedding.....	- 40 -
Figure 18 model training with 100 epoch.....	- 43 -
Figure 19 real total demand and predicted results.....	- 44 -
Figure 20 load forecasting results.....	- 45 -
Figure 21 Load Shedding for home appliance.....	- 47 -
Figure 22 Time Variant Electricity Pricing.....	- 52 -
Figure 23 Understanding the energy Charge.....	- 53 -
Figure 24 Training Dataset Sample.....	- 56 -
Table 1 Load Priorities.....	- 37 -
Table 3 load shedding simulation result.....	- 46 -



# CHAPTER 1 INTRODUCTION

## 1.1 General

Load Forecasting plays a critical role in the management, scheduling and dispatching operations in power systems, and it facilitates the prediction of energy demand for different time spans. In future electric grids, to achieve a greater control and flexibility than in actual electric grids, a reliable forecasting of load demand could help to avoid dispatch problems given by unexpected loads, and give vital information to make decisions on energy generation and consumption [1].

The ability to predict future behaviours and energy demand is part of the intelligence needed by Smart Grids, where information technology is strongly applied. As a result, load forecasting has been an essential factor which supports economic growth and increase energy efficiency. Three different types of load forecasting are as follows: Short term, which refers to half an hour to one day, medium term for one day to one year and long term for one year and above. Each of these are used depending on the reason for forecasting. Accurate forecasting is critical for implementing necessary planning and management techniques, while inaccurate forecasting may lead to excess or insufficient generation. To have accurate forecasting this paper used a very high dimensionality LSTM layers for short term load and price forecasting

Effective load demand and price forecasting can help improve and properly plan demand response (DR) measures. Demand response alters the normal electrical usage pattern. It can take the form of consumers reducing or shifting their electricity usage based on time based rates or other incentives or it can be direct load control where power companies cycle the air conditioning, water heater, or other loads during peak and off peak demands. We also design a controller called priority based load shifting using Proteus software and C programming that enables consumer reduce their electric usage [1] [2].

## 1.2 Background of Project

Today's world is a world of Digital Electronics. The technological advancement has reached a stage where we can't do anything without the help of sophisticated instruments like computers, phones and wireless mobiles. The distribution plants may become overloaded due to load growth and substation planning and it leads to a complication in the distribution system operation in areas with high load density. An acute power shortage in every corner of the place becomes inevitable, and there is need to cut down the load from one section & supply to other section [2].

The objective of the model is forecast total electrical load and price then use it to maximize the utility of the consumer subject to a minimum daily energy supply level, maximum and minimum hourly demand levels. Unknown price and electric load are forecast through recurrent neural network with a confidence interval.

A five year observation is fed to the neural network to train and test our model.

A simple bidirectional communication device between the power supplier and the consumer enables the achievement of the proposed model. The demand response affects the electricity price while the price affects the demands. Through demand response, the consumer can use electricity efficiently and much cheaply while the supplier can provide power at a lower cost.

This Project is a very good example of an embedded system as all its operations are controlled by the micro controller for load shedding. The theme of our project is to forecast load and price and control the power demand by the purpose of responsive load shedding. We are using ATM80C51 Microcontroller, since this controller has four ports which are more than enough for our project. Technology using microcontrollers are the core of the today's digital circuit design industry. This system uses it for the centralized operation and digital processing. The technology used here is embedded technology which is the future of modern electronics [3] [2].

For developing automatic load shedding technique we have used different equipment's such as microcontroller, power supply(12 volts), relay, transistor, opt coupler, LED, switch etc. Finally we have developed a microcontroller based control system and a technique of load control for fixed load.

## 1.3 Objective of Project

The objective of the model is to forecast total electrical load and price using robust LSTM layers and use it to design a microcontroller based load shedding for peak hours using priority algorithm specified by the customer to minimize peak hour loads to maximize the utility of the consumer subject to a minimum daily energy supply.

### 1.3.1 Specific objective

- To model multidimensional LSTM layers
- To build a model for short term total load and price forecast
- To classify time shift able and power shift able loads
- To determine loads that can be shed
- To enable power all the time for Non time shift able loads
- To reduce the energy wastage
- To provide easy controlling mechanism at home

## 1.4 Problem statement

This paper discusses an algorithm and defines the functionality required for developing a short-term load-forecasting module for demand response applications and applied for load shedding home appliance that can be shifted based on time. special recurrent neural network (RNN) which is called long term short memory (LSTM) algorithms are utilized to provide high forecasting accuracy when handling nonlinear and multivariate problems involving large data sets. The approach is thus suitable for short-term load prediction for non-aggregated sites to optimize the demand response process. Further analysis plots a comparison of actual and forecast loads to determine accuracy of the forecast. The load shedding algorithm makes use of load priorities and considers demand limits provided by the utility [3]. The priority of the shift able power-intensive appliances are specified by the consumer. The considered domestic power-intensive shift able loads are water heater, air conditioner, clothes dryer and electric vehicle. The proposed home demand management system (HDMS) can manage the scheduling of these shift able loads based on the load priorities given by the consumer. The aim is to keep the total household load below the demand limits specified by the utility. In this model, the recurrent neural network is built using python 3.5 software and Proteus simulation tool is developed to adopt the presented algorithm and utilize it in managing the scheduling of the considered shift able loads [2] [4]. The utility has to decide the demand limit levels in domestic sector based on the possible changes in the consumer load profile in response to the energy and power price changes.

## 1.5 Scope of Project

- 1 The robust LSTM layers are build using python 3.5 software in Spider text editor.
2. The 5 year dataset with 10,000 observation that used for training and testing the network is taken from The Australian National Electricity Market (ANEM),
3. The load shedding circuit is basically on the priority based switches and microcontroller based controller. This switches basically determine the sifting algorithm
4. The controller monitors power only for the first prior appliances .the second prior appliance should have to wait until the first prior appliance is off. The activated load is displayed on the LCD.
5. These switches will be connected directly and controlled by a microcontroller. AT89c51 chosen to makes the detector much simpler. The output reading from the sensors will be displayed on the LCD.
6. All of the circuit, simulations and coding are constructed and performed using Proteus 8 and the program is run on Keil uversion 5.

## 1.6 Project Methodology

This project mostly focuses on the project development based on the customers' number of load demand and time for shifting. For a given consumer, it establishes a robust model to adjust the hourly load level in response to hourly electricity price. Particularly, the algorithm considers total electric load, price and weather data for the corresponding time period. The model used a five year data set and 20% of the observation is used for model validation and test. The performance of the algorithm is quantitatively assessed using mean absolute per cent error and mean absolute error metrics.

The microcontroller will continuously receive the data from of the push buttons in the digital packet of data. It will process the data and take appropriate action based on the algorithm. The converted data will be displayed by the LCD and the controller takes action. The project methodology shows the steps taken in order to complete the project. The methodology includes the planning and the development of the design.

The methodology followed:

- Literature was reviewed to explore the problem and possible solutions
- Different approaches to solve the problems were reviewed and come up with best solution. LSTM model for short term load and price forecasting and microcontroller based automatic load shedding control system is selected.
- Based on the selected approach read about the Recurrent Neural Network, microcontroller and programming languages.
- Build the neural network model and Designing the automatic control load shedding system
- The designed system is simulated on Proteus 8.0 implemented with available resources and LSTM model programmed in python 3.5 software.
- Finally the proposed system is tested using simulation and test dataset. Future analysis plots a comparison of actual and forecasted loads to determine forecast accuracy.

## CHAPTER 2 LITERATURE REVIEWS

### 2.1 INTRODUCTION

Electricity load forecasting has become an important task for the Demand Side Management (DSM) of power system. The time of use electricity load is one of methods in DSM, which can decrease peak load in summer without building more power plants. Because the importance of electricity load forecasting, many technology is used for predict short-term or long-term prediction, such as linear regression, time-series techniques, and data mining approach, fuzzy theory, and neural network and so on. The neural network model can be used in many areas which also widely used in electricity load forecasting. There are many application researches in the area around the world [2] [4].

### 2.2 Up to year 2020 load forecasting using neural nets

Prediction of peak electric loads in Japan up to year 2020 is discussed using the artificial neural networks (ANNs). In this study, total system load forecast reflecting current and future trends is carried out for nine power companies in Japan. Two ANNs, a three-layered back-propagation and a recurrent neural network, were designed and tested for the purpose. Unlike short-term load forecasting, long-term load forecasting is mainly affected by economic factors rather than weather conditions. This study focuses on economic data that seem to influence long-term electric load demands. Here, 10 factors are selected as inputs for the proposed ANNs: gross national product, gross domestic product, population, and number of households, number of air-conditioners, amount of CO<sub>2</sub> pollution, and index of industrial production, oil price, energy consumption, and electricity price. The data used are: actual yearly, incremental growth rate from the previous year, and both together (actual and incremental growth rate from the previous year). As a result, the demands for 2010 and 2020 are predicted to be 225.779 and 249.617 GW, respectively (preservation of the status). With structure reform, the demands for 2010 and 2020 are predicted to be 219.259 and 244.508 GW [5].

### 2.3 Comparison of Very Short-Term Load Forecasting Techniques

Three practical techniques Fuzzy Logic (FL), Neural Networks (NN), and Auto-regressive model (AR) for very short-term load forecasting have been proposed and discussed in this

paper. Their performances are evaluated through a simulation study. The preliminary study shows that it is feasible to design a simple, satisfactory dynamic forecaster to predict the very short-term load trends on-line. FL and NN can be good candidates for this application [5].

#### Load Forecasting via Fuzzy Logic

The possibility of using fuzzy logic approach in this research is based on the following observations: (I) the very short-term load forecasting problem can be treated as a multiple-input-multiple-output unknown dynamic system. It is known that a fuzzy logic system with centroid defuzzification can identify and approximate any unknown nonlinear dynamic systems on the compact set to arbitrary accuracy (ii) It is known that there is sort of periodic change in weekly load trends and there exist similarities in load trends between weekdays and weekdays, weekends and weekends, months and months, seasons and seasons, and so on. The fuzzy logic systems have been proved to have great capabilities in drawing similarities from a huge of data [6]. Therefore, as long as enough historical input-output data pairs are available, the similarities existing in load trends are able to be identified.

#### Load Forecasting via Neural Networks

The fuzzy-logic-based approach is not the only way to predict the very short-term load change, nor is it necessarily the best way for all purposes. A neural network-based very short-term load forecasting system was also developed in this research. There are many types of neural networks (NN). For example, multilayer perceptron network, self-organizing network, Hopfield's recurrent network, and so on [6]. NN has been applied to time-series prediction and load forecasting for power systems [1]. The previous work in this field used NN to predict the peak load of a day or the hourly load of a day. The performance of such applications is generally acceptable in terms of accuracy. However, there is no applications of NN to the very short-term load forecasting as we did in this research has been reported to our knowledge. Neural network has many applications because of its capability to learn. There are multiple hidden layers in the network. In each hidden layer there are many neurons [2]. The data was pre-processed for presentation to the network in such a manner so as to present the network with past history of load values and load parameters [3] [4]. The time component was presented by transforming the time values using sinusoidal transfer functions in order to avoid the nonlinearity



## Load Forecasting via Auto-regressive Model

If the present load is simply assumed to be a linear combination of the previous load, then an auto-regressive (AR) model of the form  $\hat{L}_t = \sum_{i=1}^m a_i L_{t-i} + \epsilon_t$  can be used to model the load profile. The predicted system load at time  $t$  (in minutes), a random load disturbance,  $a_i$  are unknown coefficients [4] is referred to as an AR model of order  $m$ . The unknown coefficients in [5] can be tuned on-line using the well-known least mean square (LMS) algorithm. The advantages of this algorithm is that it is simple and easy to understand, no off-line training is necessary (no historical data base is needed), and easy to implement.

## 2.4 Long Term Probabilistic Load Forecasting and Normalization with Hourly Information

The classical approach to long term load forecasting is often limited to the use of load and weather information occurring with monthly or annual frequency. This low resolution, infrequent data can sometimes lead to inaccurate forecasts. Load forecasters often have a hard time explaining the errors based on the limited information available through the low resolution data. The increasing usage of smart grid and advanced metering infrastructure (AMI) technologies provides the utility load forecasters with high resolution, layered information to improve the load forecasting process. Three key elements of long term load forecasting are being modernized: predictive modelling, scenario analysis, and weather normalization. Multiple linear regression analysis has been widely used in the forecasting fields, including load forecasting. Detailed coverage on the theory of regression analysis and linear models is provided [13]. Implementation of MLR in SAS is presented [14]. A comprehensive guideline about how to apply MLR models to short term load forecasting is discussed in [1]. In this case study, we start with several MLR models: a classical model for monthly energy forecasting denoted as  $M_1$ , a classical model for monthly peak forecasting denoted as  $M_2$ , Tao's vanilla benchmark denoted as  $M_3$ , and a group of customized short term load forecasting models denoted as  $M_4$ . The models in are derived using Hong's methodology documented [1], where by default, 3 years of data are used for parameter estimation and the year after is used for variable selection. When using year 2010 for variable selection, we denote the resulting variable combination as  $M_{i,j}$ . All of these starting models have the dependent variable Load and an intercept term.

## 2.5 Practical Experiences with an Adaptive Neural Network Short-Term Load Forecasting System

An adaptive neural network based short-term electric load forecasting system is presented. The system is developed and implemented for Florida Power and Light Company (FPL). Practical experiences with the system are discussed. The system accounts for seasonal and daily characteristics, as well as abnormal conditions such as cold fronts, heat waves, holidays and other conditions. It is capable of forecasting load with a lead time of one hour to seven days [7]. The adaptive mechanism is used to train the neural networks when on-line. The results indicate that the load forecasting system presented gives robust and more accurate forecasts and allows greater adaptability to sudden climatic changes compared with statistical methods. The system is portable and can be modified to suit the requirements of other utility companies [8].

**Adaptive Mechanisms** While operating in real time environment, it is imperative that the system should be able to adapt to changing conditions. If the system is unable to adapt to these changes, then the forecasts rendered by the system becomes obsolete. In order to achieve this objective, three adaptation mechanisms have been devised. They are Daily Adaptation (DA), Weekly Adaptation (WA) and Monthly Adaptation (MA). The overall system consists of two modules. One is the system forecast initialization (SFINIT) module and the other is the real time module. The SFINIT mode is used to train various neural network architectures and obtain weights for different years, seasons or months. This process is done off-line. The real time mode utilizes the ANN weights obtained using the SFINIT module and allows the user to interactively obtain a forecast on-line. Forecasts for 1 hour to 7 days can be obtained based on forecast temperature. The system allows the user to make adjustments in the forecasts. This facility gives the load forecaster flexibility to adjust to sudden changes in weather conditions.

## CHAPTER 3 BUILDING RNN FOR SHORT TERM LOAD FORECASTING

### 3.1 Introduction

The demand for electrical energy is increasing. Today we live in a world of Digital Electronics. The technological advancement has gotten to stage where sophisticated instruments like computers, phones, mobiles; wireless etc. are crucial to do anything. Energy is becoming more and more costly. Economic growth increases demand. Production capacities barely keep-up with demand, and new capacities are costly and long-term. The aim of this paper is to forecast the demand using all the information that affects the load to have an effective management of peak demands (load shedding) [7].

Recurrent Neural Networks have been applied successfully for the load shedding application. Neural networks demonstrated their ability to accurately extract and learn the relationships between observed variables, leading to excellent results yielded in short term load forecasting (STLF).

The aim of this project is to allow the end users to live by allowing them to schedule and manage their electric usage by designing a system that helps the customer know the future load and price. The end users select priority of loads and the controller sheds the loads accordingly. In this chapter is a detailed discussion of short term load and price forecasting, load shedding, the hardware and software implementation and the overall system design. Let us see the definition of STLF.

### 3.2 The Overall System

The overall system consists of two modules. One is the system forecast building (STLF) module and the other is the load shedding module. The STLF mode is used to train various neural network architectures and obtain weights for different days of the month. This process is done off-line. The real time mode utilizes the RNN weights obtained using the LSTM model. Forecasts for 30 minutes to days can be obtained based on forecast temperature, total load demand weather condition and electric price.

### 3.3 Short-Term Load Forecasting

An important procedure in real-time energy management method where prediction of the system load over an interval usually from one hour to one week is known as short-term load forecasting (STLF). Prediction of the energy demand allows an optimal allocation of power, keeping the ratio between overall costs and system efficiency low. This process is crucial for

peak shaping and demand response (DR). In fact, an accurate forecasting is an essential factor used to have a fast response predicting demand fluctuations and managing energy storages in an optimal way. Demand Response is used to control the peak demand and give the customer feedback about the status of the electricity grid and market prices allowing the planning and control of electricity consumption [12].

An accurate load forecasting facilitates the ability of the customer to plan so they consume electricity when the electricity costs are lower, and it also allows the operators to manage the grid for better efficiency and lower costs.

For this kind of problem, various determining factors such as weather data or more in general, all the factors influencing the load/consumption pattern should be considered. Meaning for an accurate load forecasting, exogenous variables may be considered. In this paper temperature, dry bulb, dew point, total load and electric price are used. Forecasting effectiveness can be measured by its accuracy (the difference between predicted and real value).

Accurate electricity load forecasting method using neural networks is used after analyse of the problem domain and the most adequate set of attributes are chosen. Market behaviour is predicted using the historical prices, loads and various information for short-term electricity price and load forecasting [12] [13].

### 3.3.1 Benefits of STLF

Traditional power systems have recently been evolving towards Smart Grid (SG), which is characterized by bi-directional flow of electricity, continuous monitoring, and exchange of information between controllers and actuators.

Different optimal strategies have been developed for the efficient management of SGs in these frameworks.

Load forecasts are important because they are involved in many tasks, including market bidding and scheduling of controllable loads and generators. Time spans for such short term applications range from a few minutes to several days. Accurate and reliable short-term load forecasts assist in avoiding the decisions that negatively affect the operation of the grids.

Parameters that are usually taken into account when modelling a national electricity consumption curve are seasonality, calendar events, and weather dependency. This paper focuses on short-term load forecasting and load shedding. We conduct the analysis to determine 1) how to efficiently model all the available information concerning total load, and 2) which information is important to be considered in the forecasting system.

### 3.4 Part one RNN intuition

Designing an RNN model for load forecasting is quite challenging. Appropriate model structures, input variables, training and testing data sets are RNN characteristics. Input variables which directly influence the output are crucial. Four input variables were used in this paper, which are classified as weather variables, price and electric load demand

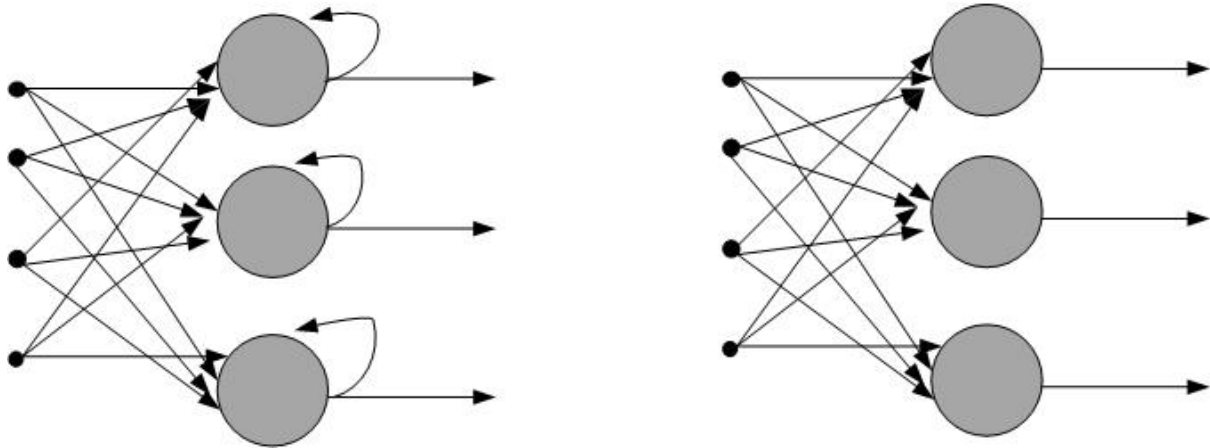
What is Recurrent Neural Network?

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms. This is because they are the only ones with an internal memory. Because of their internal memory, RNN's have the ability to remember important factors of the input they received. This enables them to make precise predictions.

As a result, they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more. When compared to other algorithms, they can form a much deeper understanding of a sequence and its context [21]. Unlike other algorithms, Recurrent Neural Networks can produce predictive results in sequential data. Sequential data is basically ordered data, where related values are in a sequence, one after the other. The most popular type of this data is Time series data, which is just a series of data points that are listed in time order [20].

In a RNN, the information cycles through a loop. During decision making, it takes into account the current input and also what it has learned from the inputs it received previously.

Representation of the Recurrent Neural Networks As a result of internal memory, Recurrent Neural Networks are able to accurately remember the previous inputs. It produces which is copied and looked back into the network. The present and the recent past are the 2 inputs of an RNN. This is important because the sequence of data contains crucial information about what is coming next RNN's apply weights to the current and also to the previous input. Furthermore, gradient descent and Back propagation through Time are used to tweak their weights.[8]



*Figure 1 Recurrent VS Feed Forward*

In a Feed-Forward neural network, the information moves straight through the network in one direction, from the input layer, through the hidden layers, to the output layer. As a result, the information does not touch a node twice. Because Feed-Forward Neural Networks don't have memory to store their input and are bad at predicting what will happen next. . In addition, since a feed forward network only considers the current input, it has no notion of order in time. Apart from their training, they simply can't memorize what happened in the past [19].

### Backpropagation through Time

For neural networks, you do Forward-Propagation to get your model output and check how correct the output is, to then analyze the error. You then do Backward-Propagation. This is going back through the network to find the error partial derivatives with respect to the weights, allowing you to adjust the value from the weights. Derivatives are then used by Gradient Descent which is an algorithm that is used to iteratively minimize a given function. It then adjusts the weights up or down accordingly to decrease the error. That is exactly how a Neural Network learns during the training process [10].

Gradient descent is an optimization method used to find the exact combination of weights for a network that will keep the output error at a minimum. To find the lowest point which is where the error is minimal, we descent down the slope of gradient. Minimize the cost function and arrive to the global minimum is done by descending down the gradient. The steepness of the slope (gradient) and learning rate determine the steps [11].

Learning rate: value that speeds up or slowdown how quickly an algorithm learns. It's between 0.0001 and 1.

### 3.4.1 Long-Short Term Memory

Long Short-Term Memory (LSTM) networks are an extension of recurrent neural networks, and it extends their memory. Therefore it is suitable for learning from important experiences that have long time lags in between. The values of an LSTM are used as building units for the layers of a RNN, which is then often called an LSTM network. RNN's can remember their inputs over a long period of time through LSTM's. This is because LSTM's store their information in a memory, similar to the memory of a computer because the LSTM can read, write and delete information from its memory. This memory can be pictured as a gated cell, because it decides whether or not to store the information (e.g. if it opens the gates or not), based on how important it finds the information. The algorithm learns weights and assigns importance of memories accordingly. This simply means that it learns over time, the importance of each bit of information [8] [11].

In an LSTM you have three gates: input, forget and output gate. These gates decide whether or not to let new input in (input gate), delete that information because it is not important (forget gate) or to allow it to affect the output at the current time step (output gate) [12].

#### The Activation function

It receives the output of the summation operation and transforms it into the final output of the node.

Now we will discuss what will happen to the neurons:

**1<sup>st</sup> step:** All of the values from the input side are added up so it takes the added weighted sum of all of the input values [15].

$$\sum_{i=1}^m (W_i X_i) \tag{3.1}$$

Where: m is the number of inputs, W is weight and X input variables.

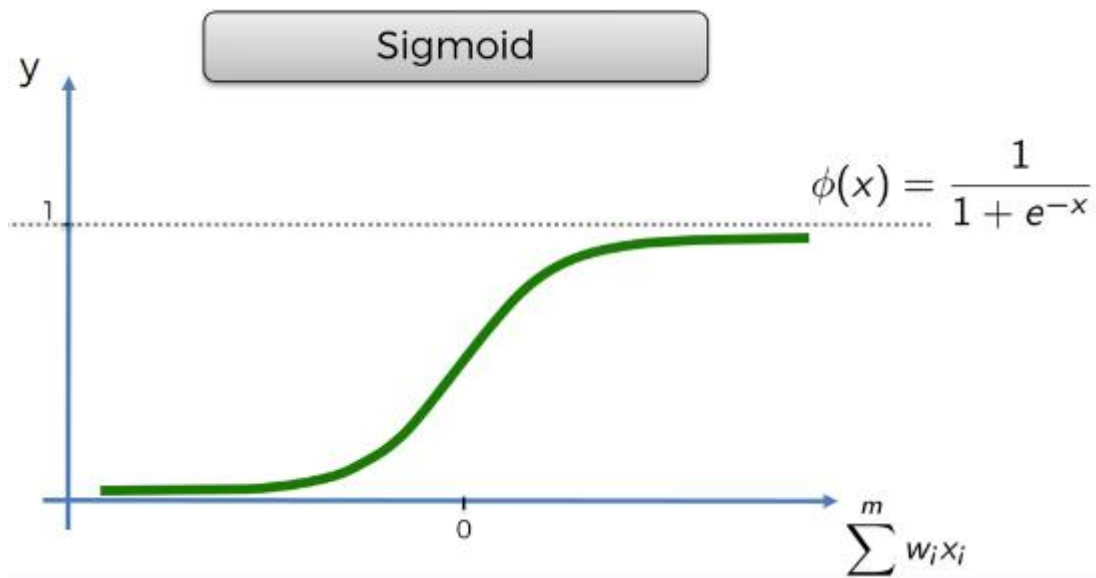
**2<sup>nd</sup> step:** applies an activation function. This is applied to the weighted sum and based on that, the neuron whether it needs to pass on a signal or not [16].

Why an activation function is used.

1. Used to transform the output of a summation operation into the final output of a node.
2. Introducing a non-linearity programs. This essential component enables the network to map input of a network to the output network and successfully train and enable it for the solving of complex problems.

**Activation function used in this paper:**

1. The Sigmoid function: this is a function which is used in the logistic regression .The function gives a smoothed gradual progression. It is very useful in the final layer, in the output layer especially when the aim is to predict probabilities [17].



*Figure 2sigmoid activation function*

2. The rectifier function: this is one of the popular functions. It goes all the way to zero and then from there, it gradually progresses as the input value increases as well [6].



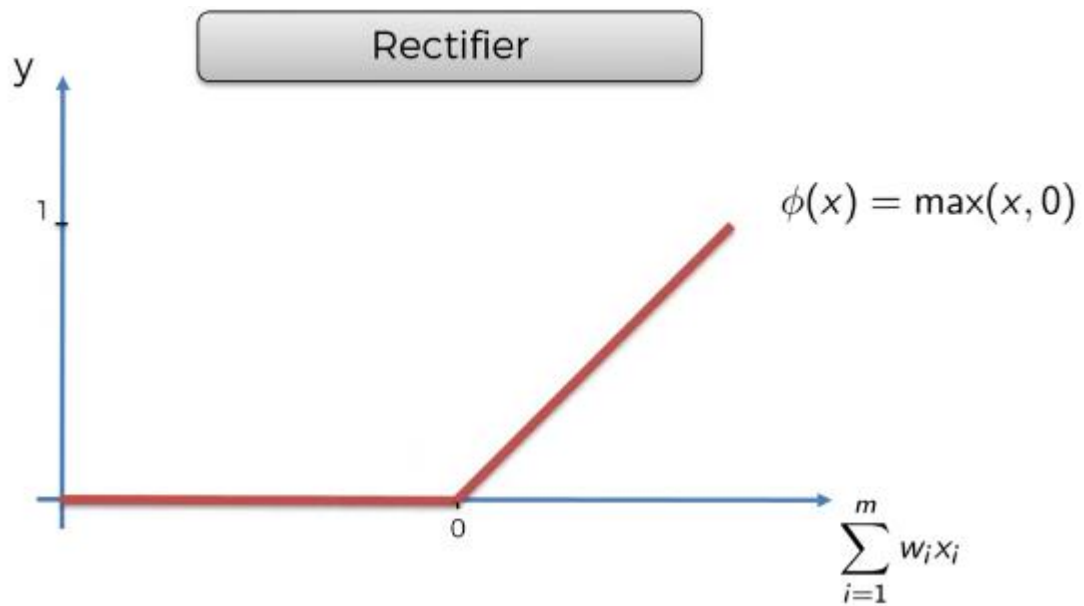


Figure 3 rectifier activation function

### 3.4.2 How neural network do learns

There are two fundamentally different approaches that can be followed. One is hard coded coding where specific rules are coded and wanted outcomes are also specifically coded. We guide it through the whole way and all possible options are accounted for, that the program has to deal with. The other is neural network where a facility is for the program to understand what it is doing on its own. We create this NN where we provide it inputs then we let it figure everything out on its own [18].

We have input values that have been supplied to the perceptron ,basically to our neuron network then the activation function apply here we got the first output. In order to be able to learn, a comparison of the output value and the actual value is needed. If the actual value has a small difference with the output value we should calculate the function known as the cost function which is calculated as [12]:

$$C = 1/2 (\hat{Y} - y) \tag{3.2}$$

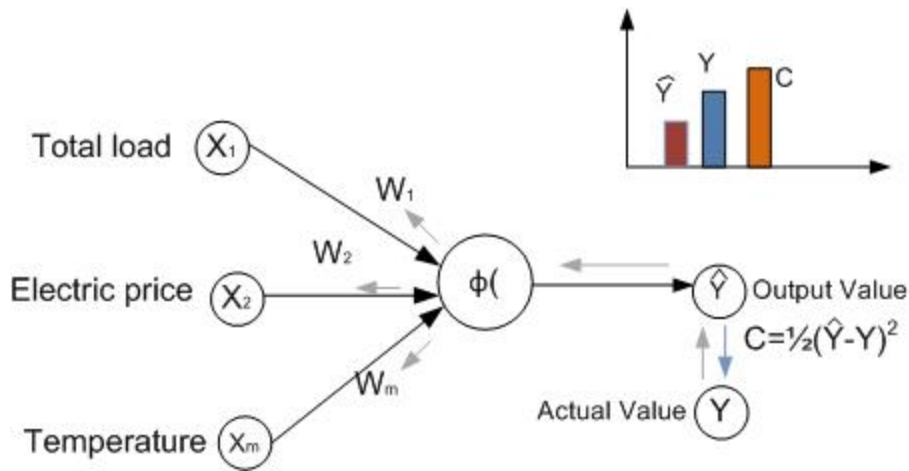


Figure 4 Neural Network Training

### 3.4.3 Neural network training

There are different cost functions that we can use but the above cost function is the most common. The cost function is basically telling us what the error in our prediction is. The goal is to minimize the cost function. The lower the cost function is, the closer the output value to actual value. Once we have compared, the information will put it back into the neural network and the weight gets updated [8]. After several iterations of the cost function, it is closed to zero. As soon as we have found the minimum of the cost function that is the final neural network with adjusted weights. The optimal weights are found for this data set and the network is ready to proceed to the testing phase or to the application phase.

### 3.4.4 Neural Network Architectures for Electricity Load Forecasting

Our electricity load forecasting software uses a combination of time-series techniques, linear regression, data mining approach and neural network algorithms for the prediction of short-term electricity load and price. The main part of the forecasting function is the neural network. The frameworks of the neural network prediction are as follows

## Architecture of electricity load forecasting with neural network

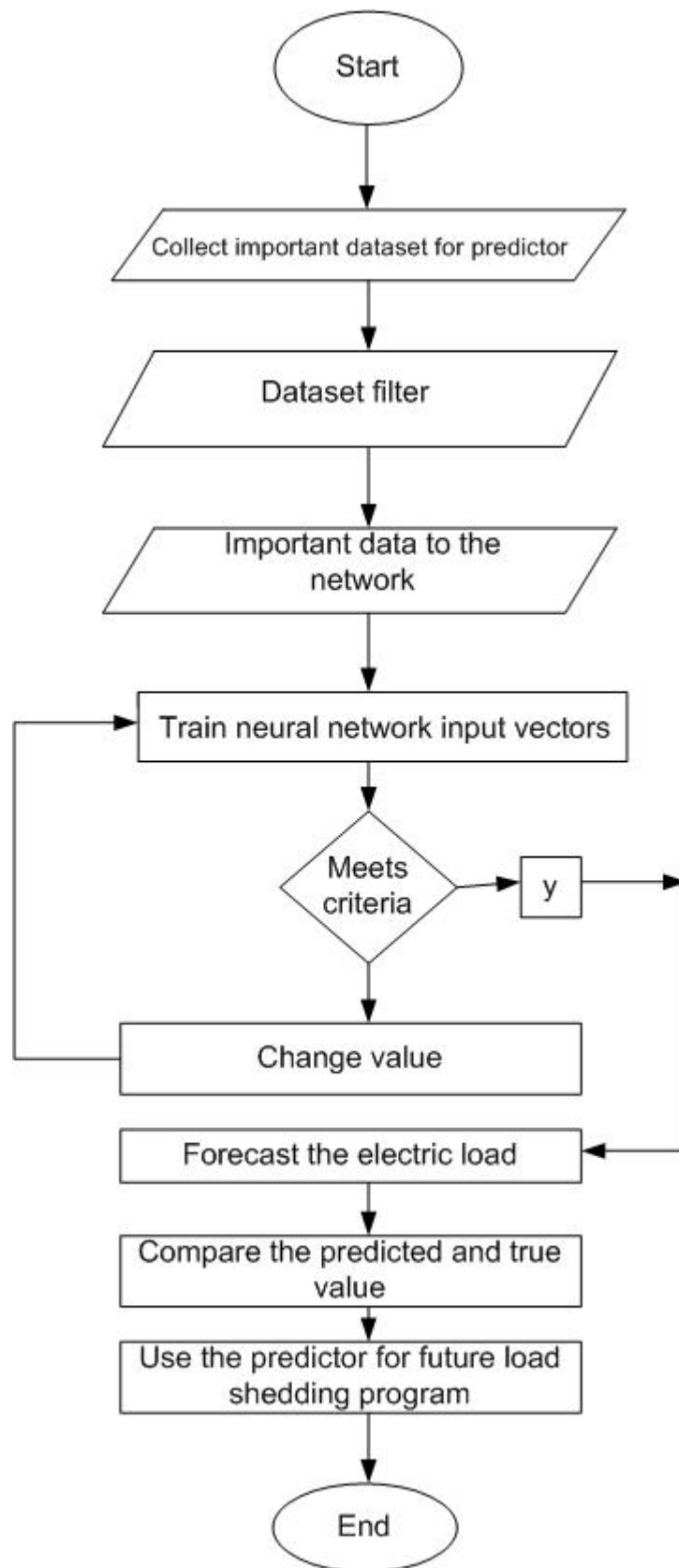


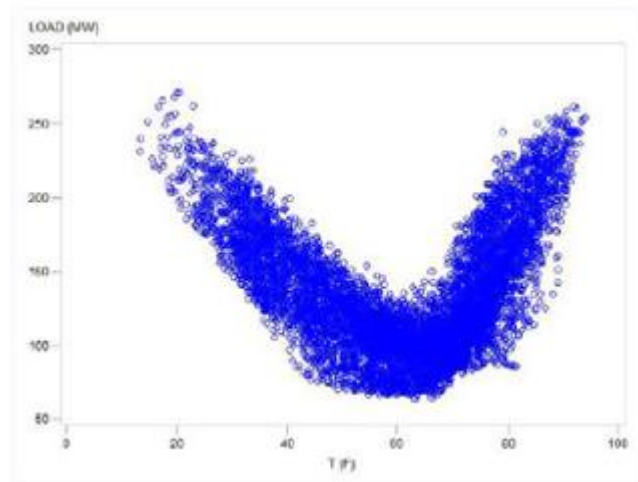
Figure 5 framework of NN prediction

### 3.5 Part two: Data pre-processing

In simple words, pre-processing refers to the alterations done to the data before feeding it to the algorithm. Data Pre-processing is a technique that is used to convert the raw data into a processed usable data set [1]. When data is collected from various, it cannot be used in the analysis in its raw form. For achieving better results from the applied model in Machine Learning projects the format of the data has to be arranged to a proper manner. The training and test datasets are presented in Excel

1	SETTLEMENTDATE	TOTALDEMAND	RRP	dwpt	dryblb
2	01-01-2018 00:00	6012.23	75.56	28.1	24
3	01-01-2018 00:30	6030.84	81.27	29	24
4	01-01-2018 01:00	5931.49	78.02	29	24
5	01-01-2018 01:30	5875.22	65.02	29	24
6	01-01-2018 02:00	5806.38	64.85	29	23.7
7	01-01-2018 02:30	5731.49	64.23	29	23
8	01-01-2018 03:00	5688.43	63.36	29	24
9	01-01-2018 03:30	5643.23	64.03	29	23
10	01-01-2018 04:00	5594.46	61.29	31	24
11	01-01-2018 04:30	5587.08	60.04	30	23
12	01-01-2018 05:00	5557.7	59.73	30	24
13	01-01-2018 05:30	5594.7	57.74	30	23
14	01-01-2018 06:00	5583.75	52.87	28	24
15	01-01-2018 06:30	5690.27	57.4	28	24
16	01-01-2018 07:00	5717.62	58.74	28	24
17	01-01-2018 07:30	5895.55	59.71	27	24
18	01-01-2018 08:00	5959.12	59.5	27	25
19	01-01-2018 08:30	6144.88	59.73	27	25
20	01-01-2018 09:00	6268.9	60.29	26.9	24.6

*Figure 6 sample training dataset*



*Figure 7 scatter plot of hourly load and temperature*

The scatter shows an asymmetric “U” shape .on the left, the lower the temperatures are, and the higher the loads are, which is primarily due to heating needs. The opposite happens on the right, which is primarily due to space cooling needs.

Data set should be formatted in such a way that allows more than one Machine Learning and Deep Learning algorithms to be executed. The method which gives the best results is to be chosen. In neural network there are 3 type of datasets:[13] [14]

- a. Training dataset: the training dataset is used to adjust the weight of the NN. Usually its 60% of the total observation
- b. Validation dataset : used to minimize over fitting
- c. Testing dataset: used as final test to gauge how accurately the network has been. Trained usually its 20% of the total observation.

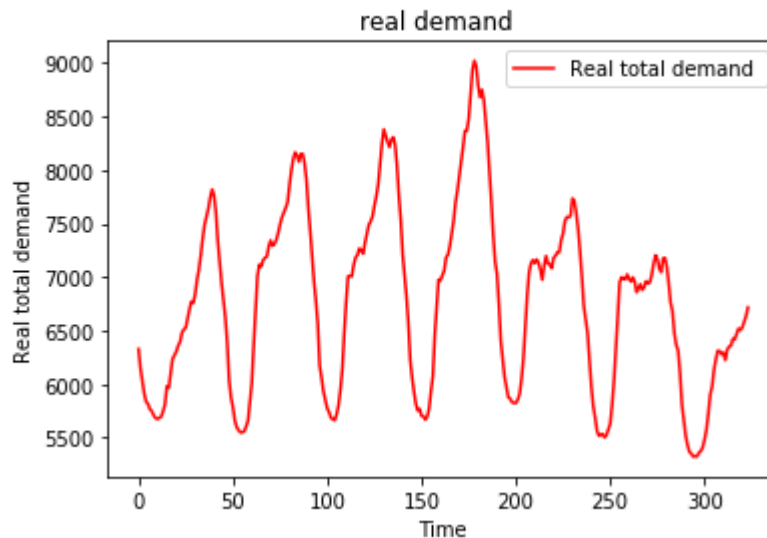
Understanding the various parameters on which electricity demand is depend on is the first step towards developing a machine learning model for load forecasting.The forecasting depends on various daily parameters like previous electricity demand trends, weather, humidity, electricity price, etc.

We performed extensive tests for modelling the selection. The neural is built using a data set which has 8,000 samples used for training and 2,000 samples used for the testing

### 3.5.1 Load Data

We can now load the file directly using the pandas function `pd.read_csv`. There are four input variables and one output variable (the last column). Once loaded we can split the dataset into input variables (X) and the output class variable (Y).

```
In [3]: dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')
...: training_set = dataset_train.iloc[:, 1:2].values
```



*Figure 8 Real Total Demand (MW)*

### 3.5.2 Future Scaling

Feature scaling is an integral step in the data transformation stage of the data preparation process. Feature Scaling is a method used in Machine Learning to standardize independent variables of data sets.

When our data is comprises attributes with varying scales, many machine learning algorithms work better when the data is changed to have the same scales. This is useful for optimization algorithms used in the core of machine learning algorithms such gradient descent. From many ways of future scaling like standardization and normalization, the latter is most useful and common so we used this method here. [22]

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3.3)$$

This avoids the domination of one variable by other variables. From sklearn preprocessing normalization can be imported. In this paper normalization is performed using minmaxscaler class. Data can be rescaled using scikit-learn using the MinMaxScaler class. After rescaling all of the values are in the range between 0 and 1 [5].

### 3.5.3 The number of time steps

The specified number of time steps defines the number of input variables ( $X$ ) used to predict the next time step ( $y$ ). As such, for each time step used in the representation, that many rows must be removed from the beginning of the dataset. This is because there are no prior observations to use as time steps for the first values in the dataset [6].

It is important to have the right number of timestep because a wrong number of timestep could lead to over fitting or nonsense predictions. For this neuron training 60 timestep is used. It means that at each 60 time  $T$ , the RNN is going to look at the 60 observations between 60 days before time  $T$ . Based on the trained it is capturing during these 60 previous timestep it will try to predict the next outcome. So 60 timestep of the past information from which our RNN is going to learn and understand some correlation or some trends. Based on its understanding the neuron predicted the next output.

Therefore two different data structure entities are created. The first entity is  $X_{train}$  which will be the input of the neural network and the second entity will be  $Y_{train}$  which contain the output. For each observation  $X_{train}$  contains the 60 previous variables (total load, price, dry bulb and dew point) and  $Y_{train}$  contains the total day of the next day. This is a special data structure which is needed to do for every time  $T$ .

### Reshaping

The LSTM input layer is specified by the “**input\_shape**” argument on the first hidden layer of the network. The three dimensions of this input are: [20]

- **Samples.** One sequence is one sample. A batch is comprised of one or more samples.
- **Time Steps.** One time step is one point of observation in the sample.
- **Features.** One feature is one observation at a time step.

This means that the input layer expects a 3D array of data when fitting the model and making predictions, even if specific dimensions of the array have a single value, e.g. one sample or one feature. When defining the input layer of your LSTM network, the network will assume that you have 1 or more samples and therefore requires that you specify the number of time steps

and the number of features. This can be done by specifying a tuple to the “*input\_shape*” argument. We can then use the *reshape()* function on the NumPy array to reshape the one-dimensional array into a three-dimensional array with 8,000 sample, 60 time steps, and 4 feature at each time step. The *reshape ()* function when called on an array takes one argument which is a tuple defining the new shape of the array. We cannot pass in any tuple of numbers; the reshape must evenly reorganize the data in the array.

```
# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

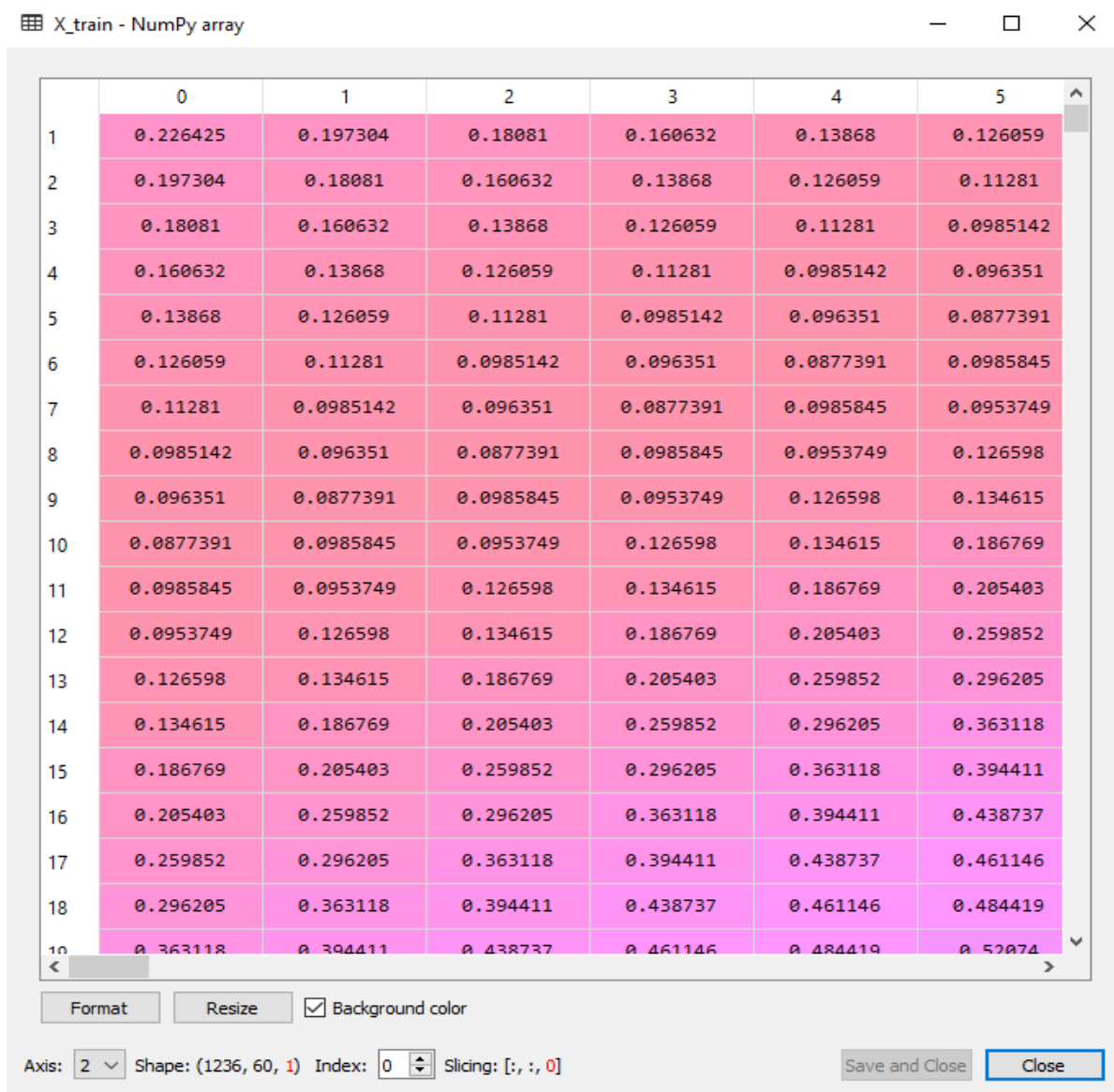


Figure 9 3D structure of X\_train



### 3.6 Part three: Define Model

In this case, we will initialize the network weights to a small random number which is generated from a uniform distribution (**'uniform'**), in this case between 0 and 0.05. This is because that is the default uniform weight initialization in Keras. [15] [16]

Usage of initializers: Initializations define the way to set the initial random weights of Keras layers. Usually it is simply *kernel\_initializer*.

We will use the rectified linear unit (**'Relu'**) activation function on the hidden layers and the sigmoid function in the output layer. The sigmoid and hyperbolic tangent activation functions cannot be used in networks with many layers due to the vanishing gradient problem. The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better. The rectified linear activation is the default activation when developing multilayer Perceptron [16].

$$R(x) = \max(0, x) \tag{3.4}$$

We use a sigmoid on the output layer to ensure our network output is between 0 and 1. The main reason why we use sigmoid function is because it exists between (0, 1). Therefore it is especially used for models where we have to predict the probability as an output. Since the probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

$$\emptyset(x) = \frac{1}{1+e^{-x}} \tag{3.5}$$

To build the RNN model important Keras libraries and packages should be imported. Libraries like Sequential, Dense, LSTM, and Dropout.

Import modules:

- a) Sequential module: required to initialize our neuron network. It can be imported from Keras.model in class called sequential. We can create a Sequential model by passing a list of layer instances to the constructor. The focus of the Keras library is a **model**. The simplest **model** is defined in the **Sequential** class which is a linear stack of Layers [21].

- b) Dense module: required to build the neuron network .it can be imported from Keras layer, import Dense.

Let's build robust model step by step.

```
In [4]: from keras.models import Sequential
...: from keras.layers import Dense
...: from keras.layers import LSTM
...: from keras.layers import Dropout
Using TensorFlow backend.
```

### 3.6.1 Initializing the RNN

There are two way of initializing a deep learning model we used a sequence of layers in this project [9].

- i. Defining the sequence of layers
- ii. Defining the graph

Regressor is an object of a sequential class which represents exactly the sequence of the layers that can predict the continuous time .Defined as a sequence of layers. Regression analysis is a type of predictive modelling technique which analyses the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and for finding the causal\_effect\_relationship between the variables. As mentioned before, this regression analysis estimates the relationship between two or more variables

```
# Initialising the RNN
regressor = Sequential()
```

- It indicates the significant relationships between dependent variable and independent variable.
- It indicates the strength of impact of multiple independent variables on a dependent variable.

### 3.6.2 Adding the LSTM layers and dropout regularization

For LSTM layers the first argument is the number of *units*, which is the number of LSTM cells or units we want to have in this LSTM layer. We choose the relevant number.

The second argument is *return\_sequences* and set is equal to true since we are building a stacked LSTM which have several LSTM layers. When another LSTM layer is needed to add, it should set to True. If it is false another layers cannot be added.

The third argument is *input\_shape* that is the shape of the input Containing X\_train that created.

The model should have high dimensionality by adding multiple LSTM layers. But it is also possible to increase even more dimensionality by including a large number of neurons in each of the LSTM layers since capturing the trend of 5 years data is complex. The model need to have high dimensionality and therefore large number of neurons in each of the multiple LSTM layers is required.

The number of neurons we choose for this first LSTM layer is 50 and for the next LSTM layer, 50 neurons will give a model with high dimensionality. Choosing too small number of neurons in each of the LSTM layers it wouldn't be able to capture very well. In this model four LSTM layers are added to build a robust LSTM structure [13] [17].

```
# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))
```

What is Dropout?

Dropout is a technique whereby randomly selected neurons are ignored during the training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally suspended on the forward pass and any updates on weight are not applied to the neuron on the backward pass. This makes the network becomes less sensitive to the specific weights of neurons. Resulting in a network that is capable of better generalization and less likely to over fit the training data. [21]

Dropout is easily carried out by randomly selecting nodes to be dropped-out with a given probability (e.g. 20%) each weight update cycle. This is how it is implemented in Keras. Dropout is only used during the model training and is not used for evaluation of the skill of the model.

### 3.6.3 Compile Model

Whenever a neural network finishes passing a batch through the network and results prediction generation, it must decide how to use the difference between the obtained results and the values it knows to be true for weight adjustment on the nodes so that the network steps towards a correct solution. The Optimizing algorithm is the algorithm that determines that step. Now that the model is defined, compilation is next. Model compilation uses the efficient numerical libraries under the covers (the so-called backend) like Theano or TensorFlow. The backend automatically finds the best way to represent the network for training and making predictions to run on the hardware, such as CPU or GPU [9] [13].

When compiling, we must specification of additional properties is required when training the network. Remember training a network is finding the best set of weights to make correct predictions for this problem. The loss function must be specified to evaluate a set of weights, the optimizer used to search through different weights for the network and any optional metrics we may want to collect and report during training. We must specify the loss function that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation. In this case, we will use Regression Loss Functions called mean square error loss (MSE). MSE, is calculated as the average of the squared forecast error values. Squaring the forecast error values forces those to be positive. We will also use the efficient gradient descent algorithm **Adam**. Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data [13].

```
# Compiling the RNN
regressor.compile(optimizer='adam', loss="mean_squared_error")
```

### 3.6.4 Fitting the RNN to the Training

In this stage everything is ready, the training data set can be fit on to the model that was built before. Our model can be trained or fit on our loaded data by calling the **fit ()** function on the

model. To fit the training data set fit class is used with the following arguments. X\_train and Y\_train: the training and the test dataset respectively

**Epochs:** The number of epochs is a hyper parameter which defines the number times that the learning algorithm will work through the entire training dataset. One epoch means that each sample in the training dataset has had a chance to update the internal model parameters. An epoch is comprised of one or more batches. In this model 100 iteration is preferred to obtain accurate predictions [10].

**batch\_size:** The batch size is a hyper parameter that defines the number of samples to work through before updating the internal model parameters. The model batch size is 32, this means in every 32 observation the weights updated [10].

```
# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

### 3.7 Part four Making the prediction and visualizing the result

1. The model trained to be able to predict the total load demand and price at time T plus one based on the 60 previous total load demand. Therefore to predict each total load at each day of 28, January 2018, we will need the 60 previous days before the actual day.
2. To get at each day of January 2018, the 60 previous total load demand of the 60 previous day, we will need both the training set and the test set because some of the 60 days that will be from the training set and also some total load demand of the test because some of them come from January 2018. Therefore Concatenation of the training set and the test set will be able to get the 60 produced inputs for each day of January 2018.
3. The concatenation which will be to concatenate the original Dataframe. From this original concatenation, we will get the input of each prediction that is the 60 produced total load demand at each time T, this is then what will be scaled. These inputs that we will scale to get the predictions and scaling the input by not changing the actual test value. This led us to the most relevant results.

#### Concatenation the original Dataframe using the *concat* function

```
# Getting the predicted total Load demand of 2018
dataset_total = pd.concat((dataset_train['TOTALDEMAND'], dataset_test['TOTALDEMAND']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
```

What is the new variable called input contains:

We want to predict the total load demand of January 2018. The input we are going to use for the first day of 2018 which is January 28, the lower bounds of the range of input needs will be the first day of January 2018 minus 60. The upper bound will be the previous total load demand just before the last total demand that we predict. Therefore this will be the last total load of our dataset total. Finally, predict total demand estimated from regressor using predict method. The most important thing not to forget it to transform the predicted values in to the normal scale since the model used Normalization to suppress dominance of one variable to other.

### 3.8 Evaluate Model

Our neural network has been trained on the entire dataset and the performance of the network can be evaluated on the same dataset.

This will only give an idea of how well the dataset has been modeled (e.g. train accuracy), but no idea of how well the algorithm might perform on new data.

## Tie It All Together and Make Predictions

```
# Recurrent Neural Network

# Part 1 - Data Preprocessing

# Importing the Libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the training set

dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')
training_set = dataset_train.iloc[:, 1:4].values

# Feature Scaling

from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)

# Creating a data structure with 60 timesteps and 1 output
X_train = []
y_train = []
for i in range(60, 1258):
    X_train.append(training_set_scaled[i-60:i, 0:3])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)

# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 4))
```

*Figure 10 RNN Training*

In the below image all the variable outputs are shown .these are Imported data that help for the prediction.

Name	Type	Size	Value
X_test	float64	(324, 60, 1)	[[[0.79400221] [0.79328407]
X_train	float64	(1236, 60, 1)	[[[0.22097028] [0.22642522]
dataset_test	DataFrame	(324, 5)	Column names: SETTLEMENTDATE, TOTALDEMAND, RRP, dwpt, dryblb
dataset_total	Series	(1620,)	Series object of pandas.core.series module
dataset_train	DataFrame	(1296, 5)	Column names: SETTLEMENTDATE, TOTALDEMAND, RRP, dwpt, dryblb
i	int	1	383
inputs	float64	(384, 1)	[[0.79400221] [0.79328407]
predicted_stock_price	float32	(324, 1)	[[6353.6387] [6187.2905]
predicted_total_demand	float32	(324, 1)	[[6353.6387] [6187.2905]
real_total_demand	float64	(324, 1)	[[6327.72] [6147.49]
training_set	float64	(1296, 1)	[[6012.23] [6030.84]
training_set_scaled	float64	(1296, 1)	[[0.22097028] [0.22642522]
y_train	float64	(1236,)	[0.17358475 0.24273433 0.31258739 ... 0.53269883 0.49485724 0.41643046 ...

File explorer
Variable explorer
Help

Figure 11 imported data in python



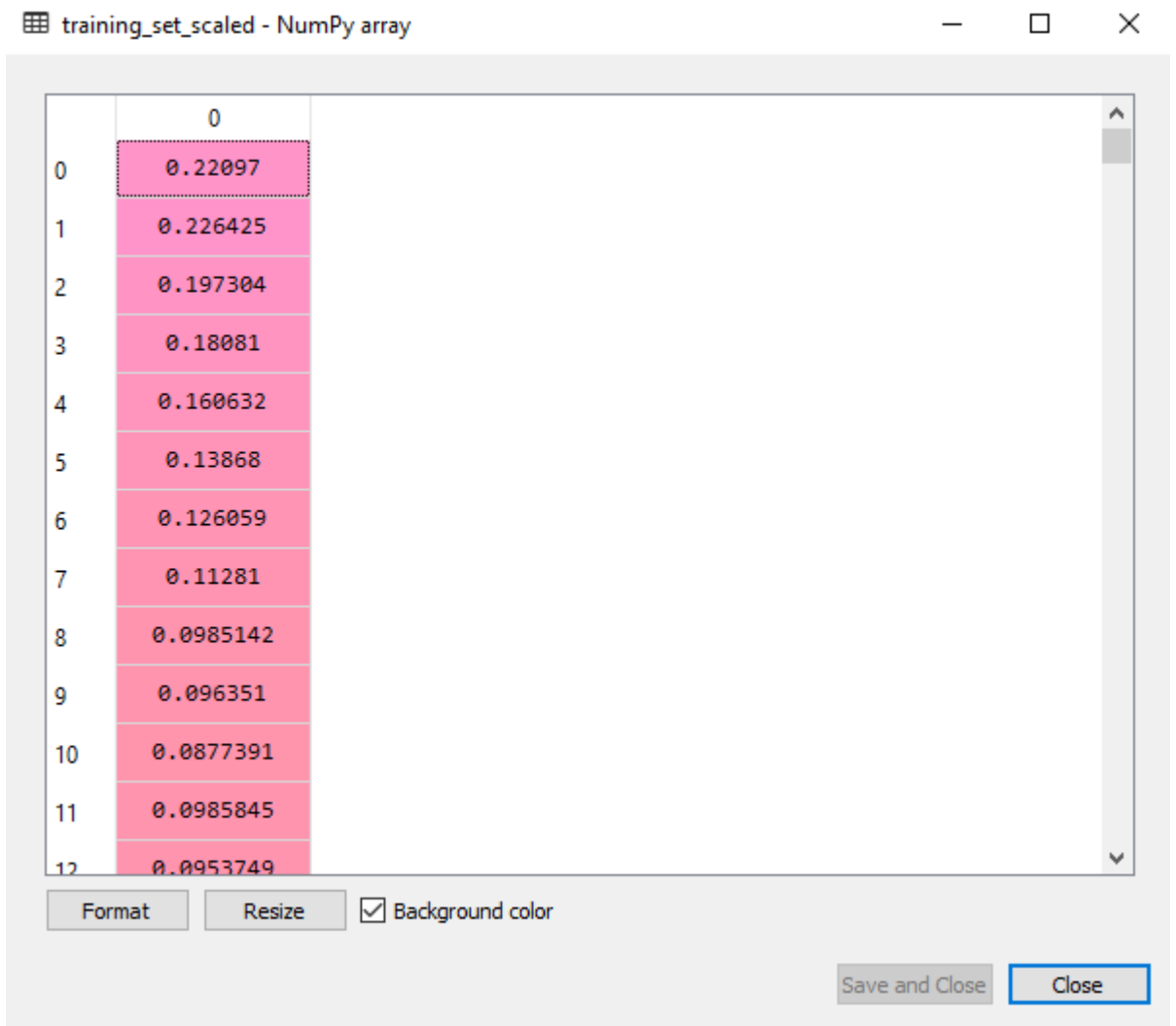


Figure 12 Normalized output of  $X_{train}$

## CHAPTER 4 LOAD SHEDDING SYSTEM DESIGN

### 4.1 INTRODUCTION

Load shedding is the automatic reduction of power consumption according to pre-set parameters. The cost of electricity is based on two main components – the amount used (consumption) and the intensity of its use (demand). For a certain period of time, usually a month, the demand is determined as the highest measured intensity. Load shedding is a mechanism to help manage electricity use in order to reduce this peak demand, significantly reducing monthly utility costs. Unnecessary peak periods – such as 11:00 AM to Noon on weekdays that account for 30% higher average charges than other hours can be prevented using an intelligent load shedding control [3].

#### Benefits of load shedding

- Load shedding reduces peak demand, thereby reducing electricity costs. This is typically transparent to managers
- System operation can be automatic, or programmed to alert the building manager with a recommended action to temporary shed electrical use
- Many electricity utilities offer financial incentives to reduce demand, particularly during the winter and the peak of summer
- Businesses with growing electrical needs can mitigate costly electrical upgrades, by re-distributing process-related electrical loads to remain within their current service amperage limit

### 4.2 Load shedding algorithm

It is an algorithm defining how several loads will be switched-off (and switched on again) automatically to maintain the power demand below a defined level [4].

#### Main reasons for Load shedding

1. Voluntary load-shedding in order to avoid exceeding a self-estimated maximum use of Power (in kW).

End-User wishing to make savings and/or develop green conscience can decide to limit and control his own maximum power level using a load-shedding system, even though there is no

risk of the main breaker tripping. Choosing not to use more than 6kW instantaneous power, for instance, to make savings by reducing his overall consumption.

## 2. Adaptation to Dynamic Time of use contracts from utility

Peak time management has become a major concern for Utilities, facing high demand during short periods, causing them to either buy kWh from other vendors at premium prices, or use production means (fuel, gas turbines) generating high CO<sub>2</sub> emissions. This has been triggering the use of dynamic Time of Use pricing (i.e. tariffs per kWh changing at unpredictable times). Because of this, a user may need to reduce their consumption by cutting off appliances during these high tariff periods then switch them on when the tariffs have normalized [5].

### 4.2.1 Control of shiftable loads

There are some electricity loads that users do not need to use urgently and can be shifted to times when the tariffs are low. In those cases smart controller's find the best moments to supply shiftable loads according to context data (e.g., load definitions, predicted consumptions, prices for the next day and some additional constraints. In this project appliances like water heaters, air conditioners, clothes dryers, and electric vehicles are considered as shiftable loads and used in Home Demand Management Systems (HDMS) algorithm. HDMS needs to focus on reducing the total electricity bills and minimize the peak demand on utility and related CO<sub>2</sub> emissions [10].

## 4.3 HDMS Algorithm

This section mainly focuses on the presentation of the HDMS algorithm for reducing the household power demand.

The main aim of this work is to present an algorithm for keeping the household energy consumption below demand limits specified by switching off appliances during peak hours based on the users' priorities. The schedulable loads, which are considered in this work, are AC unit, water heater, clothes dryer and electric vehicles [8].

Any non-shiftable load will not be scheduled and the system will not affect then in any way. Assumption is that demand limit levels given by the utility company, will be received by the HDMS through smart meters installed in the houses. The HEMS can manage the scheduling of

the power-intensive appliances, which have a significant impact on households' hourly and total demand.

Estabilising the status of the household appliances is the first step in HDMS. This means that, data received from each appliance are compared with the set points or requirements by the controller. If it is not required to switch on any appliance, then no action is performed. If any appliance has to be switched on and if there is no demand limit level imposed by the utility, the HDMS decides to turn on the relevant appliance and issues a control signal. If there is a request for operation from schedulable load and if the total household load is below the demand limit level imposed by the utility, the HDMS issues the control signal for turning on that appliance [8].

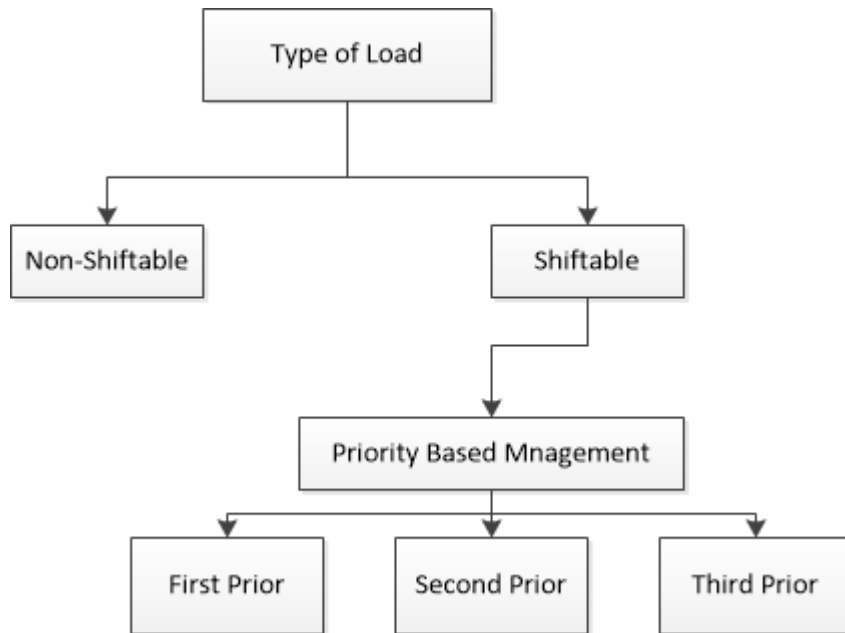
The HEMS will force the schedulable loads to shift their operating timings if the total household load is greater than the demand limit level at any given time. The appliances are switched off starting with the lowest priority appliance until the total household load is lower than the demand limit imposed by the utility or specified by the user. If any power-intensive shiftable appliance has to be turned on and if this will result in a power consumption greater than the demand limit level, the priority of that appliance will be checked with the appliances which are in on status and the those with a higher priority are kept on or switched on while those with a lower priority are turned off. The processes of checking the priority and forcing the delay of operating times of loads with a lower priority is done until a situation where the maximum peak is not exceeded is reached. The operation of shiftable loads considering total household demand and load priorities are observed while making operation schedules through HDMS under utility specified demand limits [9].

#### 4.4.1 Categorization of Loads/Appliances for Scheduling

Appliances have been differentiated as power-shiftable and time-shiftable appliances based on their power flexibility and time flexibility. A more detailed load classification is done where loads are categorized on the basis of their physical properties, job types and their sizes [5].

- Baseline loads - appliances that need to be served immediately upon request, e.g. light bulbs,
- Burst loads - appliances that can withstand delay at start but cannot be interrupted until they finish, e.g. washing machine and

- Regular loads - appliances that need to be served continuously but can cope with short interruptions, e.g. refrigerator.



*Figure 13 Types of Loads*

The appliances' time flexibility which is based on their tolerance to postponing their loads is considered. The appliances/loads of customers are categorized into two types only, non-shiftable loads which are those than need to be used at the times specified by the customer or those that need to be used throughout the day and shiftable which are those that can be postponed without much effect and these loads are called primary and secondary loads respectively. [8]

Time shiftable diagram

Shiftable Loads	Priority
Water heater	1
AC unit	2
Cloth dryer	3
Electric Vehicle	4

*Table 1 Load Priorities*

## 4.5 Domestic Shiftable Loads Operation with HDMS

The time period from 2 p.m. - 7 p.m. is considered as on-peak period where the peak demand of the electricity on the grid is really high. As a result, utility expects domestic consumers to decrease their consumption during this period. Therefore, a demand level per household is set during this time to limit the consumption and this is when time shiftable loads are switched on or off accordingly. The peak demand as well as the cost of electricity during the night time is low, so in this case, no demand limit is imposed after 11 p.m. until 7 a.m. and as many loads as possible should be shifted to these times [15].

### 4.5.1 The overall system design

The hardware part of the system designed by using different electrical and electronic components. The main components are micro controller, sensors, virtual terminal, driver and LCD. Each has their own specification.

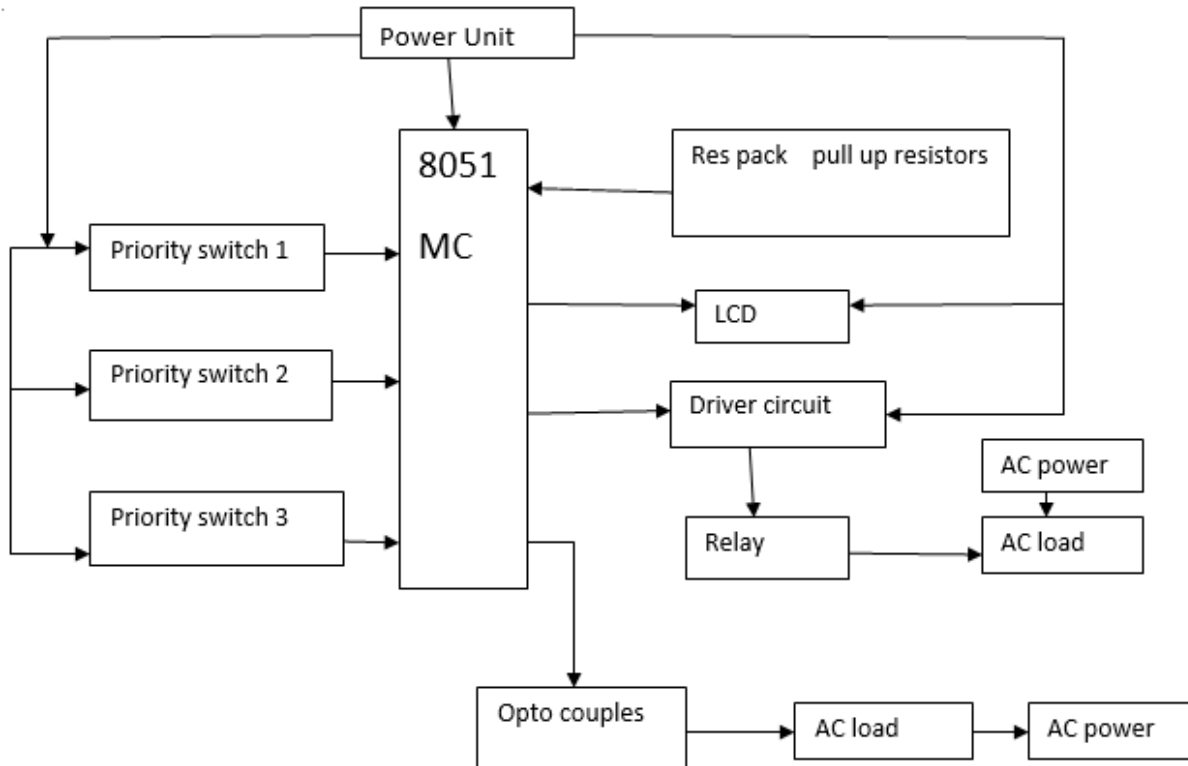
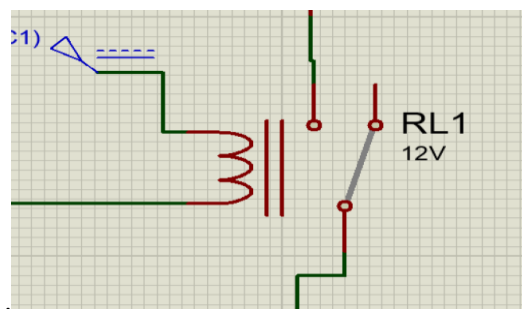


Figure 14 block diagram of load shedding

- **Relay**

A relay is an electromagnetic switch operated by a relatively small electric current that can turn on or off a much larger electric current. The heart of a relay is an electromagnet (a coil of wire that becomes a temporary magnet when electricity flows through it. As the name suggests, many sensors are incredibly sensitive pieces of electronic equipment and produce only small electric currents. But often we need them to drive bigger pieces of apparatus that use bigger currents. Relays bridge the gap, making it possible for small currents to activate larger ones. That means relays can work either as switches (turning things on and off) or as amplifiers (converting small currents into larger ones) [13]



*Figure 15 Relay Circuit*

- **Opto-coupler**

The basic design of an opt coupler, also known as an **Opto-isolator**, consists of an LED that produces infra-red light and a semiconductor photo-sensitive device that is used to detect the emitted infra-red beam. Both the LED and photo-se device are enclosed in a light-tight body or package with metal legs for the electrical connections as shown [19].

An optocoupler or opto-isolator consists of a light emitter, the LED and a light sensitive receiver which can be a single photo-diode, photo-transistor, photo-resistor, photo-SCR, or a photo-TRIAC with the basic operation of an optocoupler being very simple to understand [20].

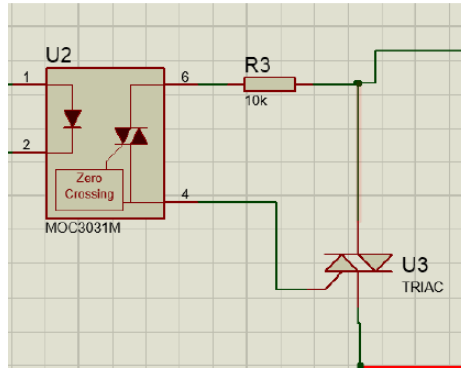


Figure 16 Optocoupler

#### 4.6 Software section

The first step here is to select an appropriate algorithm that solves the problem. Various algorithms should be considered and a comparison in terms of code size, speed, difficulty, and ease of maintenance should be done [21]. After selection of the basic algorithm, the overall problem should be broken down into smaller problems.

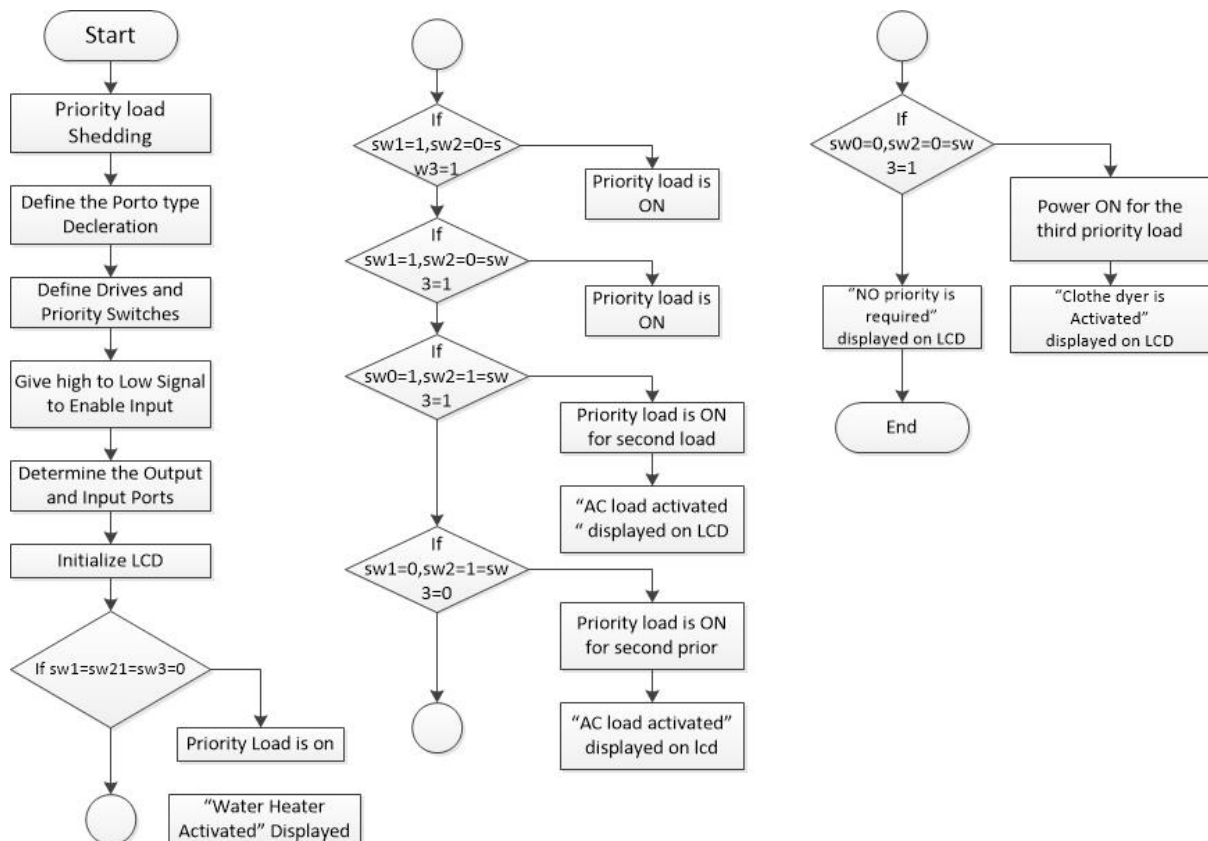


Figure 17flow chart for load shedding



#### 4.6.1 Programming methods

The program has to achieve the specified objectives list in chapter one. The aim of the project is to provide the customer with a load shedding mechanism that they can control. Depending on the customer's specification the all loads given the priority. In this project three switches specifies the priority level [21].

- The first switch leveled with the highest priority. If this switch is on, only the load associated with this switches is on. All the other switches are forced to stay off even if some loads request power. Other loads can be activated only when the first switch which is the highest priority switch is now off.
- The second switch leveled as the second prior load specifies .The load corresponding to this load is switched on only if the highest prior switch is off. This switch is given preference over all other switches except for the first switch. As a result, the third priority switch would only be on if the higher priorities are off.
- The third prior specifies only activated when the above higher prior's are off.

## CHAPTER 5 DISCUSSION AND CONCLUSION

### 5.1 Discussion

A smart grid is an electricity grid that comprises of advanced power, communications, control, and computing technologies. Under the smart grid paradigm, a more flexible scheme replaces the traditional power grid. Customers can react to electricity prices by altering consumption patterns through Demand Response (DR) Programs. The aim is to achieve a balance between electrical energy supply and demand, and a more reliable and cheaper power system operation by encouraging consumers to change their consumption from peak times, to times of lower consumption which spreads consumption evenly throughout the day. For successful deregulated electricity market operation, and for successful DR programs, future demand and price values are essential. This makes electricity price and demand forecasting a vital task for an efficient smart grid operation. This load forecasting program uses linear regression, time-series techniques and neural network algorithm to predict short- term electricity load and price. The neural network is the main part of the forecasting function [22].

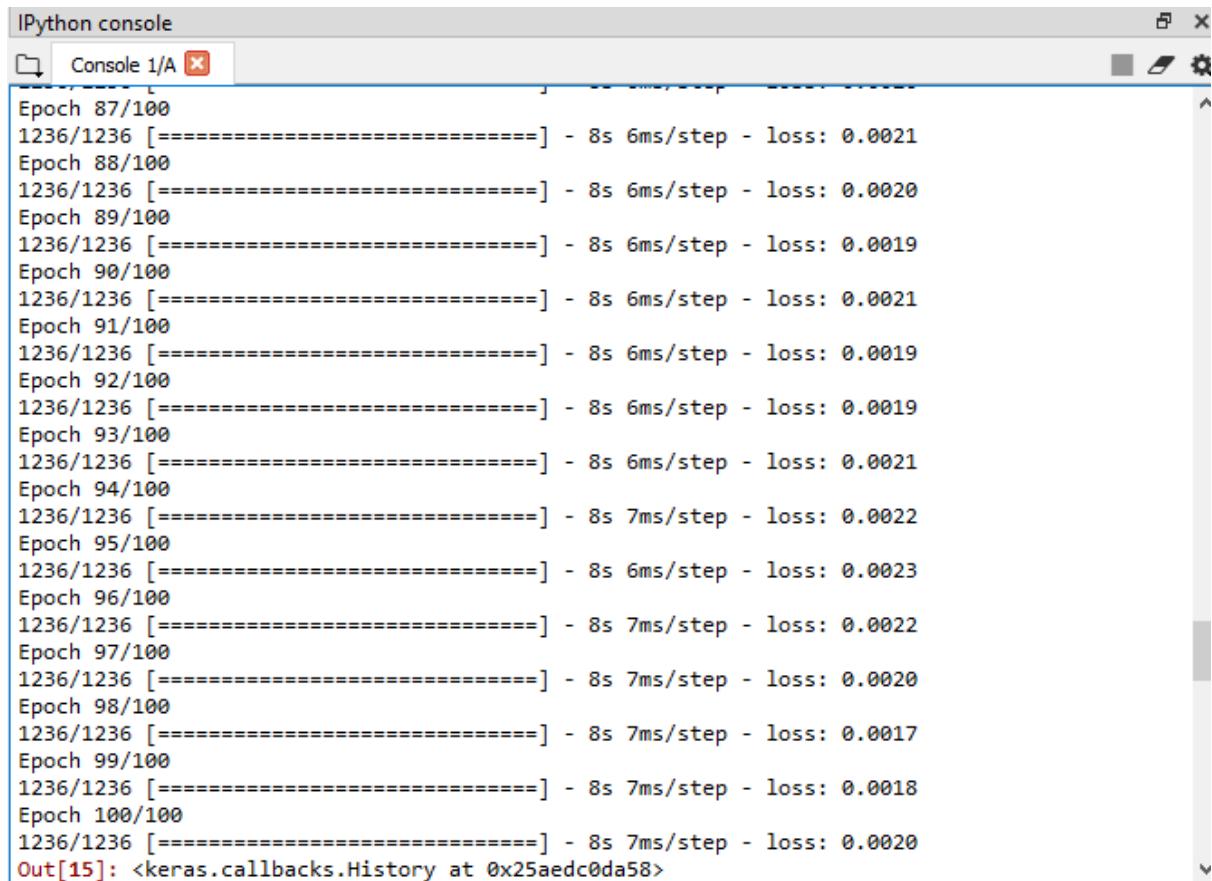
In this paper, reducing peak demand through HDMs programs can eliminate the operation of high cost generating units. For provision of the HMDs program the customer should have to specify time shiftable loads. The program does not provide the shifting algorithms for Non time shiftable loads like electric lamps and refrigerators. As stated in the previous section the electric pricing is based on consumption and demand. At peak hours, the increase is incredible. Therefore, based on the time shiftable loads given by the customer, the algorithm gives a specific priority to all appliances and provides power only for the necessary appliances instead of all.

In this paper we propose a mechanism for load scheduling which is a smart option for load shedding. We propose a home based mechanism. Using the priority level we switch on or off these home appliances.

### 5.2 LSTM model evaluation

We have trained our neural network based on the entire dataset and we can evaluate the performance of the network on the test dataset.

This gives an idea of how well the data set model is (e.g. train accuracy), but not show how well the model will handle other pieces of data or new data. The image below shows how the model is build and it shows the loss of the model for each Epoch.



```
IPython console
Console 1/A
Epoch 87/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0021
Epoch 88/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0020
Epoch 89/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0019
Epoch 90/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0021
Epoch 91/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0019
Epoch 92/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0019
Epoch 93/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0021
Epoch 94/100
1236/1236 [=====] - 8s 7ms/step - loss: 0.0022
Epoch 95/100
1236/1236 [=====] - 8s 6ms/step - loss: 0.0023
Epoch 96/100
1236/1236 [=====] - 8s 7ms/step - loss: 0.0022
Epoch 97/100
1236/1236 [=====] - 8s 7ms/step - loss: 0.0020
Epoch 98/100
1236/1236 [=====] - 8s 7ms/step - loss: 0.0017
Epoch 99/100
1236/1236 [=====] - 8s 7ms/step - loss: 0.0018
Epoch 100/100
1236/1236 [=====] - 8s 7ms/step - loss: 0.0020
Out[15]: <keras.callbacks.History at 0x25aedc0da58>
```

Figure 18 model training with 100 epoch

### 5.3 SHORT TERM LOAD AND PRICE FORECASTING RESULT

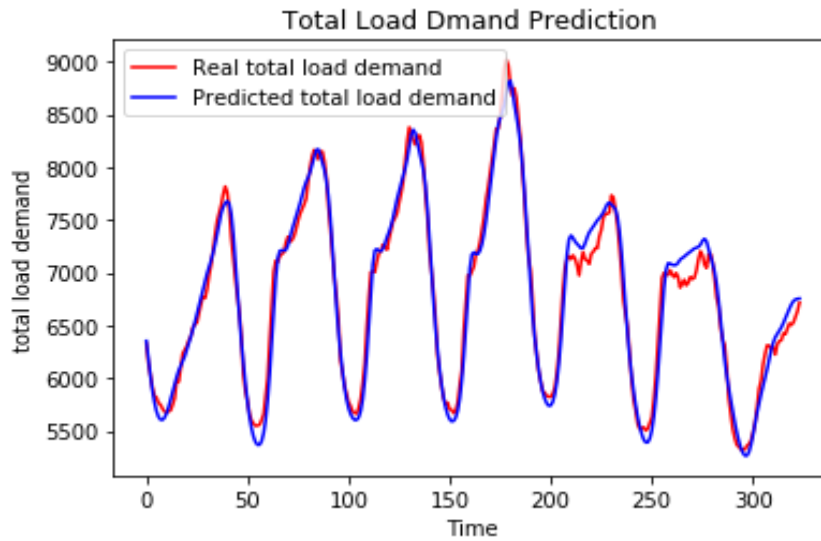
5 years observation data will be used to train the LSTM model we will implement. Once the model is trained we have to predict the total load demand for the next defined demand. In this paper, the next 48 values will be predicted. A comparison of our prediction to the actual result that is the real total demand is made to examine the accuracy of the model. This is shown below:

## One day ahead load forecasting Prediction after 27\01\2018 23:30



*Figure 19 real total demand and predicted results*

The graph below shows plots of the real and predicted total demand. The two graphs are mostly aligned, except at the very peak and strike points which is expected because in the previous data, the neuron built does not include this as an expected fluctuation of load. This shows how the model is powerful and robust.



*Figure 20 load forecasting results*

#### 5.4 Load shedding simulation result

Simulated results: These were obtained using the circuit simulated using Proteus. As decided by the algorithm the system receives power request from all the appliances especially at the peak hours, but the system only provides power for priority appliances. The overall result is shown in a table below.

Priority Switch one	Priority Switch two	Priority Switch three	Controller Action
High	High	High	The highest prior load only get activated
	High	Low	
	Low	High	
	Low	Low	
Low	High	High	The second prior load only activated
		Low	
Low	Low	High	The third prior load activated
		Low	No priority is requested

*Table 2 load shedding simulation result*

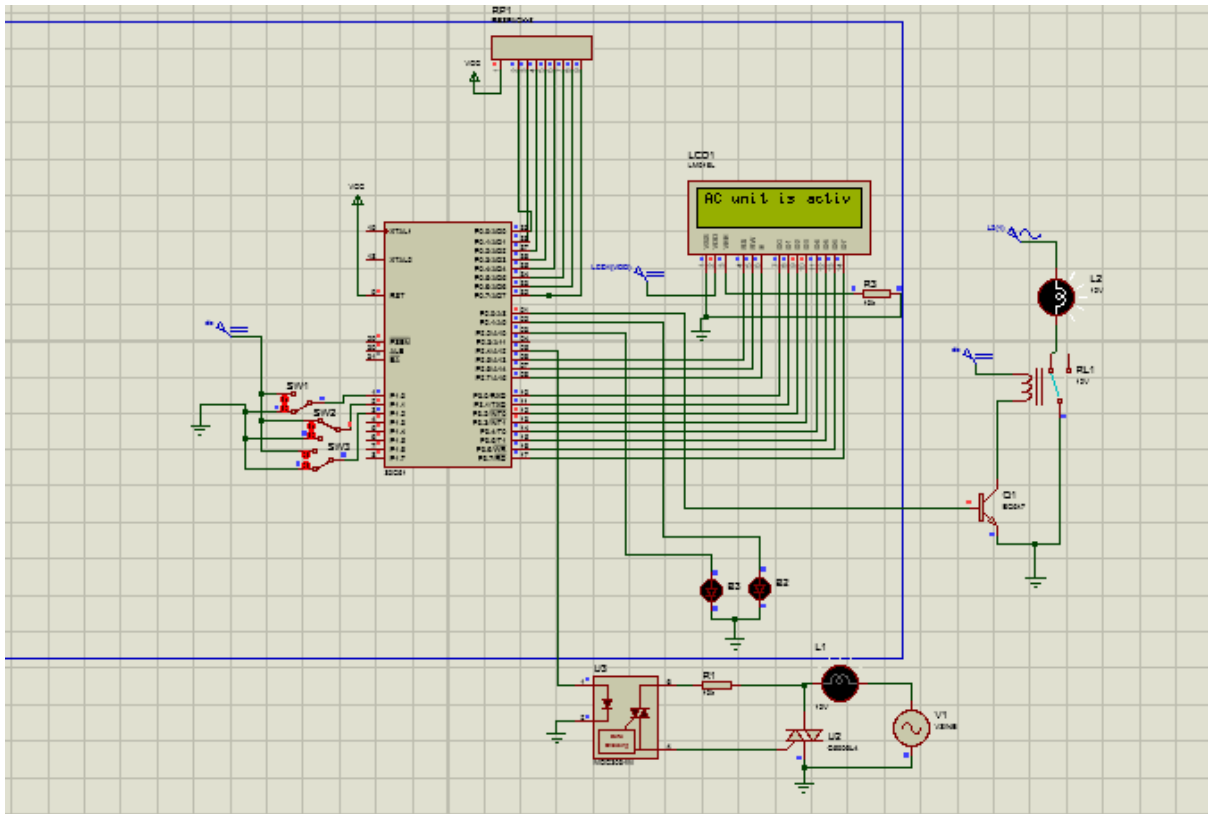


Figure 21 Load Shedding for home appliance

## 5.5 CONCLUSIONS

In this paper, the effect of DR on the predictability of demand and price time series, under smart grid environment, was investigated. The Australian Queensland state was taken as a subject case study. Price and demand profiles were generated every 30 minutes to ensure the reported DR effect is present in the data. LSTM model in neuron network were used to generalize the results. This paper show that that electricity is positively affected by DR and it produces significantly lower forecasting errors and this is independent of what forecasting tool was used.

Building a robust load and price forecasting model in neuron network and designing an automatic load shedding algorithm based on customers demand and priority request was the main objective of the project. The detector has been designed that uses AT89C51 microcontroller and priority specific switches.

As seen in the previous chapters, the RNN we built was a regressor. Indeed, we were dealing with Regression because the aim was to predict a continuous outcome (the total electric load). For Regression, evaluation of a model performance is through a metric called RMSE (Root Mean Squared Error). It is calculated as the root of the mean of the squared differences between the predictions and the real values [22].

### 5.5.1 Model Improvement

1. Getting more training data: we trained our model on the past 5 years of the total electric demand but it would be even better to train it on the past 10 years.
2. Increasing the number of timesteps: the model remembered the stock prices from the 60 previous total demand days to predict the total demand of the next day. That's because we chose a number of 60 timesteps (3 months). You could try to increase the number of timesteps, by choosing for example 120 timesteps (6 months).
3. Adding some other indicators:
4. Adding more LSTM layers: we built a RNN with four LSTM layers but you could try with even more.
5. Adding more neurons in the LSTM layers: we highlighted the fact that we needed a high number of neurons in the LSTM layers to respond better to the complexity of the problem and we chose to include 50 neurons in each of our 4 LSTM layers. You could try an architecture with even more neurons in each of the 4 (or more) LSTM layers.



### 5.5.2 Future scope

In future work, as an expansion of the proposed methodology, we would like to further explore the following directions:

1. Bi-directional module which can transfer the information between load forecasting models to load shedding system.
2. Include adaptable mechanism in the neural network architecture for accurate load forecasting.
3. Define a training dataset for holidays and weekends

## References

- [1] P. Gupta and Y. Yamada, Adaptive short-term forecasting of hourly loads using weather information , 1972, Vols. vol. PAS-91,pp.2085-2094, london: IEEE, 1998.
- [2] A. A. d, A Specification of Neural Network e Load Forecasting Problem on Control Systems Technology,, Vols. v01.2, n0.2, london: IEEE Trans, 2000.
- [3] K.L. Ho, Y.Y. Hsu, and C.C. Yang,, "Short Term Load Forecasting Using A Multilayer Neural Network with an Adaptive Learning Algorithm," , vol. vo1.7, IEEE Trans. on Power System, 2008, pp. pp.141-149..
- [4] M. e. al, "Fast Training of Neural Networks for Remote Sensing," Remote Sensing Reviews, . Operating Guides, Guide I: System Control., vol. voil.9, North American Electric Reliability Council, 2005.
- [5] K.Y. Lee and J.H. Park,, , "Short-Term Load Forecasting Using An Artificial Neural Network " vo1.7, no.1, 1992, pp.124-132., vol. vol.7, IEEE Trans. on Power Systems, 2010, p. pp.124.
- [6] S. Rahman and R. Bhatnagar,, " An expert system based algorithm for short load forecast," , vol. vol.3, IEEE Tr. Power Sya, May, 19 2012, pp. pp.392-399.
- [7] J. Chen, W. Li, A. Lau, J. Cao, and K. Wang,, ""Automated Load Curve Data Cleansing in Power Systems," Smart Grid,," vol. vol.1, pp. pp.213-221, 2010.
- [8] S. Fan, K. Methaprayoon, and W.-J. Lee,, ""Multiregion Load Forecasting for System With Large Geographical Area," , IEEE Transactions on Industry Applications, vol. vol. 45, pp. pp. 1452-1459, 2009.
- [9] M. T. Hagan and S. M. Behr, ""The Time Series Approach to Short Term Load Forecasting," , vol. 2, pp. 785-791, 1987.," vol. vol.2, no. IEEE Transactions on Power Systems, pp. pp. 785-7941, 1987.
- [10] "T. Haida and S. Muto, "Regression based peak load forecasting using a transformation technique," IEEE Transactions on Power Systems, vol. 9, pp. 1788-1794, 1994.," IEEE Transactions on Power Systems, vol. vol.9, pp. pp.1788-1794, 1994.
- [11] T. Hong, P. Wang, A. Pahwa, M. Gui, and S. M. Hsiang, " "Cost of temperature history data uncertainties in short term electric load forecasting," , in presented at 2010 IEEE 11th International Conference on Probabilistic, 2010.
- [12] "John Berdy, "Load shedding," in General Electric Company Electric utility Engineering , N.Y, 2005, pp. 120-123.," pp. pp.120-123, 2005.
- [13] "S.shao,M.Pipattanasomporn,S.Rahma, "DEvelopment of physical demand response enabled residential load models may 2014, p. vols28.," in IEEE TRAN, vol. vol.2, pp. pp.6-8, May 2014.
- [14] G. Graditi, M.L. Di Silvestre, R. Gallea, and E. Heuristi RivaSanseverino,, "Electric -based shiftable loads optimal management in smart micro-grid," in IEEE Trans, pp. pp.271-280, 2015.

- [15] Kinjyo, Y.; Miyagi, M.; Senjyu, T., "Decentralized Controllable Loads Control in Small Power , and , japan , November 2012, pp. pp. 1–6.," in *International Conference on Renewable Energy Research*, NOV.2012.
- [16] S. Xing, " "Microgrid emergency control based on the stratified controllable load shedding,"" in *Proceedings of the International Conference on Sustainable Power Generation*, China , 8-9,Sep 2012.
- [17] Chu, C.-M.; Jong, T.-L., ""A Novel DirectAir-Conditioning Load Control Method.," , England , 2008, pp. 1356–1363.," pp. pp.1356-1363, 2008.
- [18] Huang, K.-Y.; Huang, Y.-C., ""Integrating Direct LoadControl with Interruptible Load Management ,,"" pp. pp.123-144, 2015.
- [19] Salehfar, H.; Patton, A.D., ""A Production Costing Methodologyfor Evaluation of Direct Load Control,"" *IEEE Trans. Power Sys*, pp. pp.21-26, 2001.
- [20] Tao Hong, Pu Wang, and H. Lee Willis, A Naive Multiple Linear Regression Benchmark for Short Term Load Forecasting, Detroit: IEEE, Jul 24-29, 2011.
- [21] T. Hong, "Short Term Electric Load Forecasting", North Carolina State University: IEE, 2010.
- [22] S.Vemuri, W. Huang, and D. Nelson, On-line AI- gorithm for Forecasting Hourly Loads for an Electric Utility vol., , Aug., 1981, Vols. vol. PAS- 10, nework : IEEE, Aug., 1981.

## Appendix

### Appendix A: Time-variant electricity pricing

Many utilities have begun to realize that this type of pricing is hardly efficient. Pricing electricity in a way that reflects its true cost can help utilities reduce overall costs and pass these lower prices onto customers. For example, utilities can charge customers different rates at different times of the day or throughout the month – this is known as ‘time-variant electricity pricing’ [12]

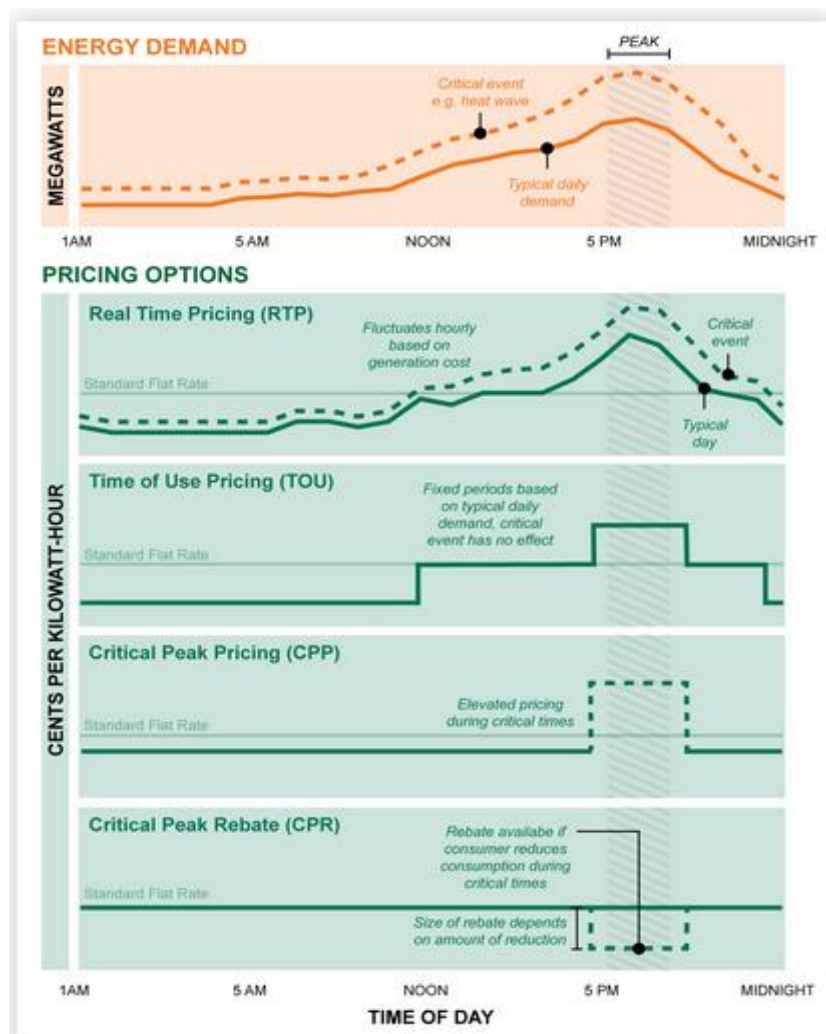


Figure 22 Time Variant Electricity Pricing

## Appendix B

Pricing allows customers to have greater control over their electricity bill. By reducing electricity use during times when it's more expensive to produce, they can take advantage of cheaper electricity being offered at other times. Furthermore, environmentally-conscious customers can reduce their carbon emissions by timing their electricity use. Essentially, time-variant pricing empowers electricity customers by bringing them into the market and allowing them to affect it with their behavior: if we would all shift away from periods of high demand, electricity prices would fall for everyone[13] [14] [17]

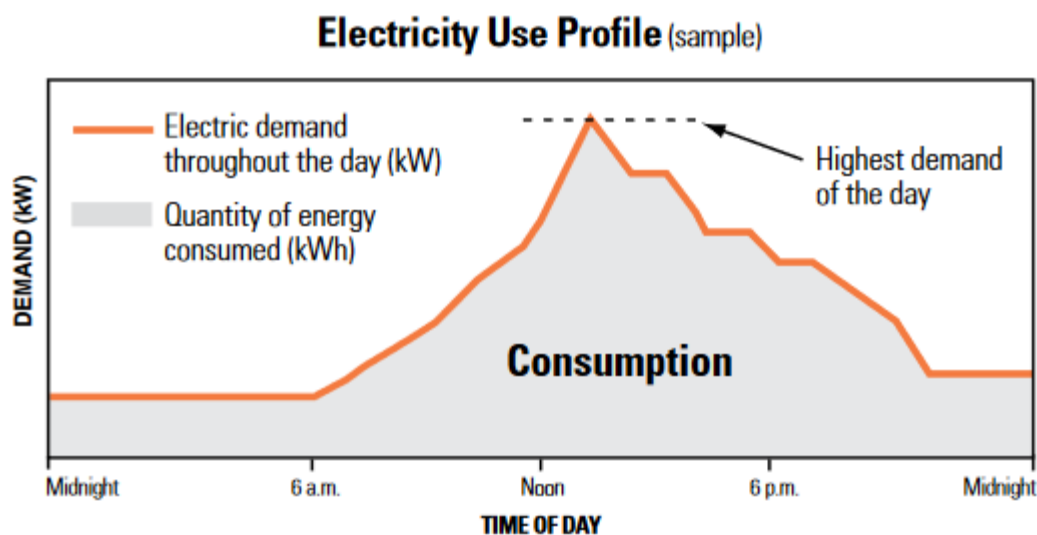


Figure 23 Understanding the energy Charge

## Appendix C

Pseudo code for the domestic load scheduling algorithm

*Initialize  $x=1$*

*If request of time shiftable appliance A is turn on*

*If total household power demand is greater than the demand limit level*

*While total household demand is less than the demand limit level*

*If the request time is peak hour off*

*SWITCH x*

*Case x=1:*

*If electric vehicle status is ON*

*If priority of electric vehicle is less than the priority of the appliance A*

*Set status of electric vehicle to OFF*

*Set x=x+1*

*Else*

*Check the priority of electric vehicle*

*Set x=x+1*

*Case x= 2*

*If clothes dryer status is ON*

*If priority of the clothes dryer is less than the priority of the appliance A*

*Set status of clothes dryer to OFF*

*Set x=x+1*

*Else*

*Check the priority of clothes dryer*

*Set x=x+1*

*Case x=3*

*If AC Unit status is ON*

*If Priority of AC Unit is greater than the priority of appliance A*

*Set status of AC Unit to OFF*

*Set x=x+1*

*Else if*

*Check the priority of AC units.*

*Set x=x+1*

*End if*

*Case x=4*

*If Water Heater status ON*

*If Priority of Water Heater is greater than the priority of appliance A*

*Set status of AC Unit to OFF*

*Set x=x+1*

*Else if*

*Check the priority of water heater status*

*Set  $x=x+1$*

*If the request time is peak hour*

*Delay all shiftable appliances to peak hour off to reduce total cost*

*End while*

*Load Priority:*

*Water heater -1*

*AC units-2*

*Clothes dryer -3*

Appendix C

## **Data pre-processing**

In simple words, pre-processing refers to the alterations done to the data before feeding it to the algorithm. Data Pre-processing is a technique that is used to convert the raw data into a processed usable data set. When data is collected from various, it can not be used in the analysis in its raw form. For achieving better results from the applied model in Machine Learning projects the format of the data has to be arranged to a proper manner.

1	SETTLEMENTDATE	TOTALDEMAND	RRP	dwpt	dryblb
2	01-01-2018 00:00	6012.23	75.56	28.1	24
3	01-01-2018 00:30	6030.84	81.27	29	24
4	01-01-2018 01:00	5931.49	78.02	29	24
5	01-01-2018 01:30	5875.22	65.02	29	24
6	01-01-2018 02:00	5806.38	64.85	29	23.7
7	01-01-2018 02:30	5731.49	64.23	29	23
8	01-01-2018 03:00	5688.43	63.36	29	24
9	01-01-2018 03:30	5643.23	64.03	29	23
10	01-01-2018 04:00	5594.46	61.29	31	24
11	01-01-2018 04:30	5587.08	60.04	30	23
12	01-01-2018 05:00	5557.7	59.73	30	24
13	01-01-2018 05:30	5594.7	57.74	30	23
14	01-01-2018 06:00	5583.75	52.87	28	24
15	01-01-2018 06:30	5690.27	57.4	28	24
16	01-01-2018 07:00	5717.62	58.74	28	24
17	01-01-2018 07:30	5895.55	59.71	27	24
18	01-01-2018 08:00	5959.12	59.5	27	25
19	01-01-2018 08:30	6144.88	59.73	27	25
20	01-01-2018 09:00	6268.9	60.29	26.9	24.6
21	01-01-2018 09:30	6497.18	64.89	27	25
22	01-01-2018 10:00	6603.94	65.11	26	25
23	01-01-2018 10:30	6755.16	66.47	27	25

*Figure 24 Training Dataset Sample*



