5-2022

Machine Learning Tools in the Predictive Analysis of ERCOT Load Demand Data

Md Riyad Hossain
The University of Texas Rio Grande Valley

MACHINE LEARNING TOOLS IN THE PREDICTIVE ANALYSIS OF ERCOT LOAD DEMAND DATA

A Thesis

By

MD RIYAD HOSSAIN

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING

Major Subject: Manufacturing Engineering

The University of Texas Rio Grande Valley

May 2022

MACHINE LEARNING TOOLS IN THE PREDICTIVE ANALYSIS OF ERCOT LOAD DEMAND DATA

A Thesis

By

MD RIYAD HOSSAIN

COMMITTEE MEMBERS

Dr. Douglas Timmer
Chair of Committee

Dr. Jianzhi Li
Committee Member

Dr. Hiram Moya
Committee Member

May 2022

ABSTRACT

Hossain, Md Riyad, Machine Learning tools in the predictive analysis of ERCOT load demand data. Master of Science in Engineering (MSE), May, 2022, 90 pp., 5 Tables, 62 Figures, references, 81 titles.

The electric load industry has seen significant transformation over the last few decades, culminating in the establishment and implementation of electricity markets. This transition separates electric generation services into a distinct, more competitive sector of the industry, allowing for the introduction of greater unpredictability into the system. Forecasting power system load has developed into a core research area in power and energy demand engineering in order to maintain a constant balance between electricity supply and demand. The purpose of this thesis dissertation is to reduce power system uncertainty by improving forecasting accuracy through the use of sophisticated machine learning techniques. Additionally, this research provides sophisticated machine learning-based forecasting methodologies for the three forecasting professions from a variety of perspectives, incorporating several advanced deep learning features such as Naïve/default, Hyperparameter Tuning, and Custom Early Stopping. We begin by creating long-term memory (LSTM) and gated recurrent unit (GRU) models for ERCOT demand data, and then compare them to some of the most well-known supervised machine learning models, such as ARIMA and SARIMA, to identify the best set of models for long- and short-term load forecasting. We will also use multiple comparison approaches, such as the radar chart and the Pygal radar chart, to perform a thorough evaluation of each of the deep learning models before settling on the best model.

DEDICATION

This thesis work is dedicated to Razia Sultana, my mother; Mohammad Neamat Ullah, my elder brother; and Farzana Yeasmin, my loving wife. They completely inspired, motivated, and supported me in every way possible to complete this degree. Thank you for your kindness and understanding.

ACKNOWLEDGMENTS

I would like to acknowledge my supervisor, Dr. Douglas Timmer, for his invaluable guidance on the scientific study and for motivating me to complete my research with endless patience and unwavering positivism. Dr. Timmer's unending devotion aided my development as a researcher and as a person. I am indebted to Dr. Jianzhi Li and Dr. Hiram Moya, members of my dissertation committee, for their astute counsel and constructive criticism regarding my graduate study and thesis. I would especially mention Dr. Zhaohui Geng, as his continuous learning mentality and industriousness motivated me a lot. I'd also like to express my gratitude to a few of my friends and fellow community members: Joni, Zisan, Shunondo, Shakil, Rupon, Jabir, Shoriful, Kollol, Dipa, Provashish, Showaib, and Porag for their constant assistance and cooperation. Finally, but not least, I want to say thank you to my caring parents for always being there for me and to my dearest wife for always being supportive during my graduate research.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

1.1 Overview

The power and electrical industries have seen considerable change in recent decades, culminating in the creation and installation of electricity markets. This change separates generation services into a separate, more competitive sector of the economy, allowing sophisticated techniques like smart grids and the integration of high-penetration renewable energy sources to flourish. Modern power systems, on the other hand, face greater unpredictability as a result of these new processes. Because, unlike many other industries, the power industry cannot store large amounts of electricity, it must be created and delivered as quickly as it is consumed.Power system load and renewable energy forecasting have evolved as key research subjects in power and energy engineering in order to achieve a balance between the electricity supply and demand.

The increased penetration of renewable energy and customer engagement in the electricity market, together with the advancement of smart grids, have posed significant problems for energy and load forecasts in power systems. The data received from the power grid, on the other hand, provides new solutions and possibilities for improving forecasting accuracy using advanced data-driven methodologies, such as machine learning methods. The purpose of this

research is to reduce power system uncertainty by improving the predictive performance of Texas ERCOT load demand data through the use of sophisticated machine learning algorithms that outperform existing traditional statistical techniques, while also providing power system administrators with reliable and widely applicable forecasting services.

All parts of the power and energy industry, including production, distribution, transmission, and consumption, employ forecasting. Power system projections are used in a variety of applications, including power supply systems engineering, sales forecasting, congestion control, rate design, and much more. The different activities and operations of the power system follow multiple methods of predictive requirements. Electricity providers must manage energy output more effectively due to growing demand. A sustainable production plan must be used to better understand consumption trends and cut down on electricity use.

Data mining algorithms can utilize this information to learn from past data and estimate future demand. For example, the Electric Reliability Council of Texas (ERCOT) calculates non-rotating reserve requirements using net load forecasting from consumption for the previous three years. Erroneous forecasting can have not only damaging financial implications but also be the reason for equipment failures and system-wide outages. To develop high-quality forecasting, powerful machine learning techniques are required. Machine learning (ML), a branch of AI for identifying patterns from complex data sets, and it has proven to be an effective tool in various fields of research over the past few years [34]. It has already been successful in solving thousands of problems that are not limited to image and speech recognition [35], text generation [36], voice recognition [37], online fraud detection [38], stock market trading [39], and more recently, self-driving cars [40]. ML algorithms are very useful for learning multidimensional

predictions from a set of data. It is also very suitable for determining performance analysis, defect identification, regression/classification study, etc. [8]

Supervised machine learning usually divides a complete dataset into two subdivisions: one is the training set, and the other one is the test set. Some scientists may also consider the validation set as the third option. An impressive amount of research in machine learning has focused on developing a model from a set of previously collected datasets that can later correctly identify or predict new datasets from the same population. A taxonomy of the machine learning algorithms is provided in table 1. Algorithms for machine learning can be initially categorized into four divisions. Supervised machine learning (SL) and unsupervised machine learning (UL) are the two most popular and widely used techniques in recent times.

Machine learning algorithms can spot trends in enormous amounts of data and produce valuable results in a variety of ways. Machine learning is often broken down into three groups based on how much input the learning system can get. These groups are supervised learning, unsupervised learning, and reinforced learning [36].

1. Supervised Learning: A supervised learning algorithm involves predicting a target or result variable using a set of predictors. This suggests that the learning algorithm has a manual that describes the output of the procedure ahead of time [37]. The method generates a function that links the inputs to the intended results. The model is trained until it achieves the desired output and accuracy. Some common ways to learn about things in a supervised way are regression and decision trees. KNN (K Nearest Neighbor) and logistic regression are also common.

2. Unsupervised Learning: In this technique, the aim or outcome variable to be predicted or estimated is unknown. In the training data sets, the algorithm looks for sophisticated processes and patterns [37]. The most common application of unsupervised learning is to classify data. Unsupervised machine learning approaches include k-means clustering, hierarchical clustering, association rules, and others.

3. Reinforcement Learning Algorithms: A machine or algorithm is exposed to an environment in which it learns to make certain judgments through trial and error [38]. The system learns from its past experiences and tries to predict the future based on that knowledge. The reinforcement learning algorithm constructs a model from the input features and output values of training data, and then uses this model to predict the values of new unseen data. Among the numerous types of regression algorithms are linear regression, multivariate regression, regression trees, and lasso regression. In contrast to classification algorithms, regression produces continuous values. Linear regression, multivariate regression, support vector regression, and regression trees are some of the most common ways to do regression. The Markov decision process is an example of reinforcement learning.

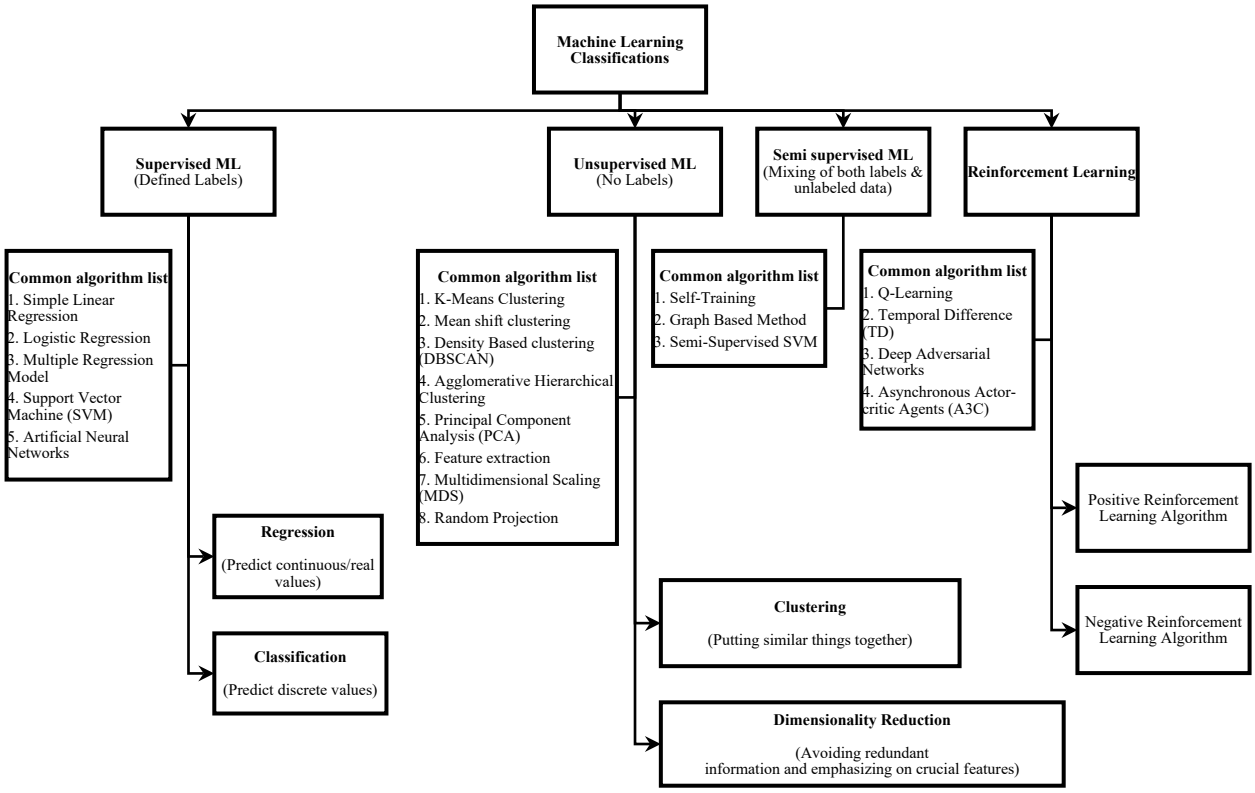
As the taxonomy suggests, the Supervised Learning (SL) algorithm can further be divided into two categories: regression and classification. The SL algorithm deals with the data with inserted labels. In SL, the output of the model type can be identified ahead of time, and this is usually one of the input(s).A simple example of classification is spam filtering, which classifies emails as spam or not spam. Various classification algorithms are used, including trees, random forests, and support vector machines.

The regression problem determines the target variable (output) in continuous values, whereas the classification problem expresses them in discrete values. On the other hand,

unsupervised learning (UL) algorithms process data without labels. It determines the relationships and similarities within the data and clusters them in a zone based on their individual relationships and differences.

Recently, another ML category; Reinforcement Learning (RL), is getting more recognition and popularity because of its effectiveness in decision making. The purpose of using RL is to make decisions in an uncertain environment to maximize the rewards.

Table 1: Machine learning taxonomy with some of the frequently used algorithms



Load forecasting is an important component of power system planning because it allows utilities to estimate future consumption or load demand. Short-term load forecasting assists utilities in determining the resources required, such as fuel to operate generating facilities and

other resources, to ensure uninterrupted and cost-effective generation and distribution of power to customers [12]. Different renewable energy resources, such as wind and solar, are being put on the distribution side of the power system as part of the smart grid construction. The power system's demand will become more unpredictable when these dispersed generation systems are integrated from the consumer side [11]. As a result, load forecasting from all individual users is required. Furthermore, most short-term load forecasting algorithms rely on historical load data and meteorological data as inputs, failing to account for the critical influencing factors for the load pattern.

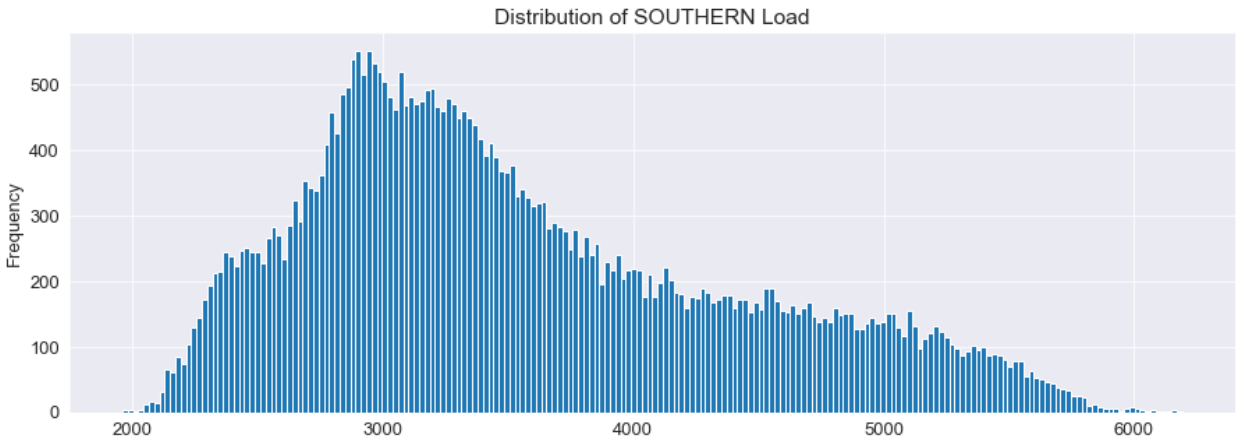


Figure 1: Southern Load distribution of Texas ERCOT dataset

Different essential influencing factors for residential, commercial, and industrial loads lead to accurate forecasting. For an efficient and comprehensive power system, an accurate load forecasting model is required to predict demand and be prepared to supply demand ahead of time. In order to run basic operations like per unit commitment, load flow, fuel allocation, and unit servicing effectively, you need to be able to make a very good guess.For affordable planning of electricity production, fuel purchase scheduling, security assessment, and short-term maintenance scheduling, short-term predictions from a few hours to a few days are required.

Time series data has been widely applied in a range of applications to reduce future uncertainty and avoid losses due to a lack of knowledge. A time series is a chronological or time-oriented sequence of data points (e.g., x_1, x_2,..., x_n) sampled at successive times in time from the variable of interest.

For instance, Figure 2 displays the ERCOT Demand Data for various regions of Texas from 2017–2021. Frequently, the rate variable is selected to be an equally spaced time interval. Time series analysis uses statistical methods to find important relationships and characteristics between points in a time series.

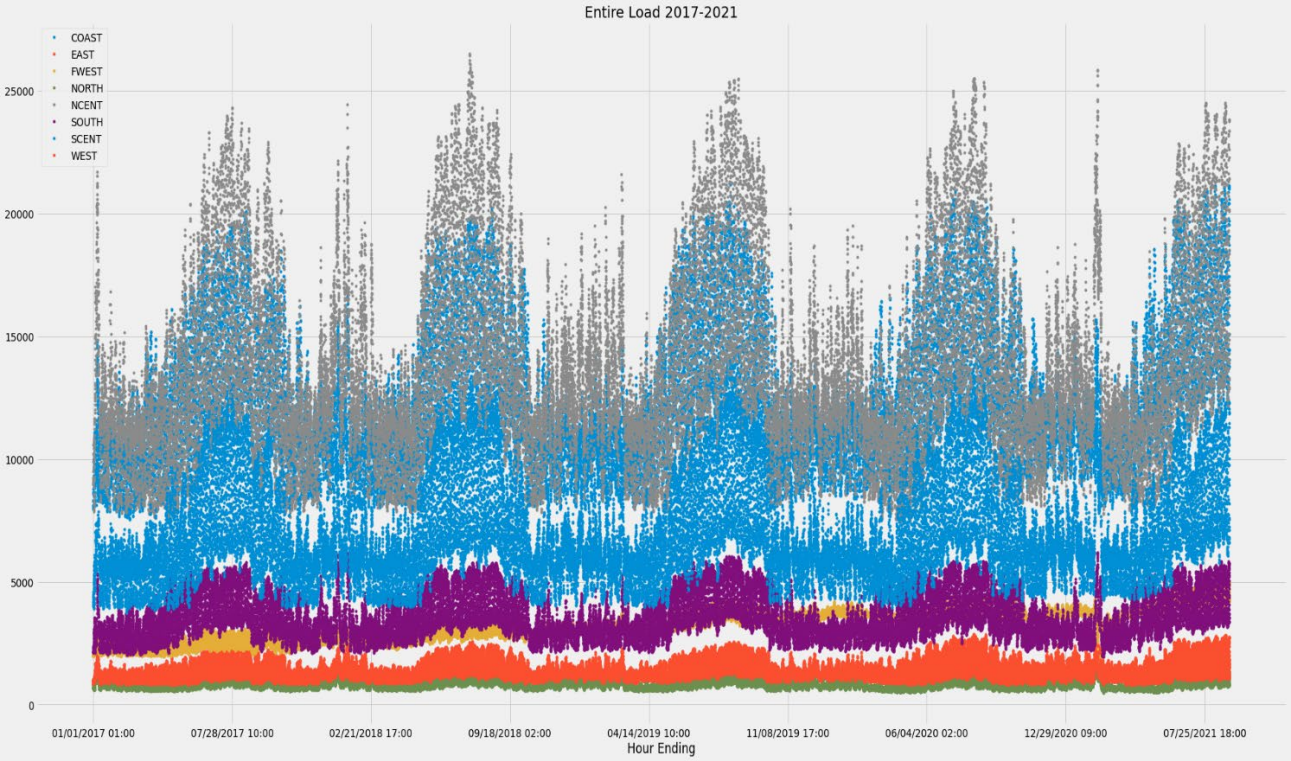


Figure 2: An example of time series. ERCOT Load Data (2017-2021)

Time series forecasting extends these analyses by focusing on the development of a model for forecasting the future based on previously observed time-series data points. Time series forecasting can be beneficial in a variety of areas, including demand forecasting, finance, supply

chain management, and others. For time series forecasting, exponential smoothing methods, Box-Jenkins models, machine learning models, and neural network models have all been developed. Despite the fact that there are numerous time series forecasting models, there is no superior model that outperforms others in all circumstances. As a result, assessing and comparing multiple models' performance is a vital job. The following section will discuss the time series forecasting models that were used in this dissertation. These models incorporate both traditional and machine learning-based methods.

1.2 Time Series Forecasting

Time series modeling has been a hot topic in both academia and business for many years, and the field has expanded significantly in recent years. In statistics, time series analysis is a technique for studying a collection of data points over a specific time period. Time-based models are successfully applied in various fields of science, including medicine, meteorology, and industrial manufacturing. The term "time series" refers to all data series that have been collected in a given period and are generally used in this context. In other words, chronological order is a collection of subsequent measurements made over a period of time. One can do this, for example, daily, weekly, monthly, or annually, depending on preferences.

Time series can be divided into two types: discrete form and continuous form. The discrete form is the most common type of time series. In scientific observation, continuous observations are those that are made on a continuous basis over an extended period of time. Distinct-time series refers to a situation where all of the observations are made at specific points in time [42].

When compared to other types of data collection, time series analyzers collect data at defined intervals over a predetermined time period, rather than infrequently or arbitrarily. In

contrast, this form of research involves much more than simply collecting data over a period of time. Time series data differs from other sorts of data in that it can show how variables change over time, but other types of data cannot. For example, time is an important factor in research because it shows both how data changes and how that changes the conclusions that can be drawn from it.

As a result, a large number of data points are typically required in time series analysis in order to preserve consistency and reliability. It is important to have a large data collection in order to make sure that the represented sample size is large enough to cut through noisy data. Additionally, it ensures that newly discovered currents or patterns are not overlooked, as well as that seasonal variability is taken into consideration. Time series data, on the other hand, can be used to foresee or forecast future data based on historical data. A variety of approaches have been developed for time series prediction challenges. In general, the methodologies can be divided into two categories: statistically based models and machine learning-based models.

Autoregressive models are the statistical procedures that are most extensively utilized (AR). In traditional AR models, a condition is imposed on the time series, requiring it to exhibit properties such as resistance, normalcy, or independence. As a result, they are frequently inadequate for dealing with complex time series or non-linear phenomena. They have a tendency to provide inadequate performance or to generate just an approximate representation of complex systems found in the real world [43] [44].

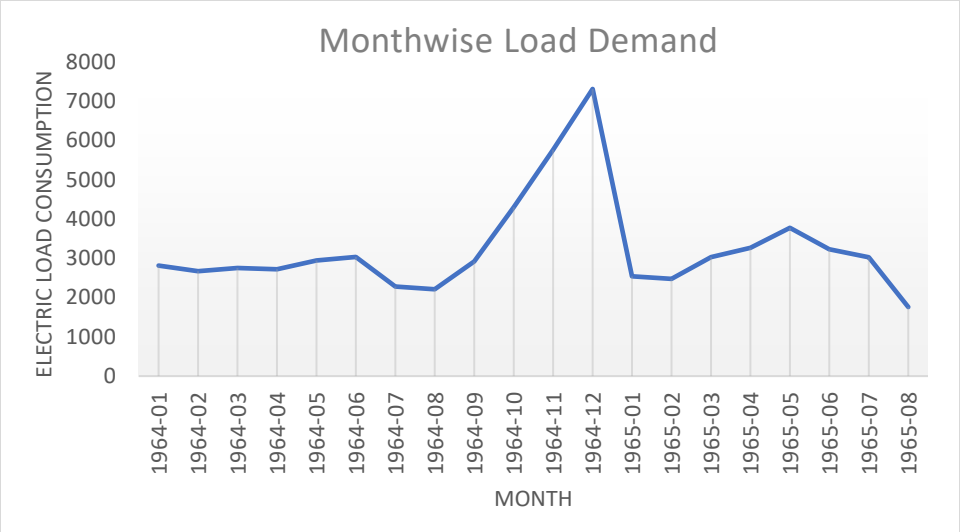


Figure 3: A Schematic representation of simple time series graph

AR models include autoregressive moving average (ARMA) models, autoregressive Integrated Moving Average Model (ARIMA), and seasonal Autoregressive Integrated Motion Models (SARIMA). While using the models, it is important to make the basic assumption that the time series under study is linear and has a certain statistical distribution.

1.3 Problem Statement

The following data is available for forecasting ERCOT load demand in the short and long term for eight distinct regions of Texas, including overall ERCOT load consumption.

Table 2: ERCOT Hourly Load Data Archives

Hour Ending	COAST	EAST	FWEST	NORTH	NCENT	SOUTH	SCENT	WEST	ERCOT
01/01/2017 01:00	8,791.79	896.75	1,997.72	683.62	9,239.15	2,366.63	4,490.78	954.19	**29,420.64**
01/01/2017 02:00	8,569.71	865.93	1,997.78	677.97	9,105.00	2,332.74	4,370.66	951.03	**28,870.81**
01/01/2017 03:00	8,326.43	839.05	1,993.70	672.00	8,988.04	2,237.51	4,210.65	944.36	**28,211.72**
01/01/2017 04:00	8,137.50	822.83	1,995.54	675.27	8,979.15	2,178.10	4,088.71	943.19	**27,820.29**
01/01/2017 05:00	8,011.87	814.02	1,995.25	663.62	9,033.55	2,133.95	4,021.76	954.94	**27,628.96**
01/01/2017 06:00	7,978.10	823.42	1,981.35	669.75	9,195.16	2,122.02	4,045.30	972.56	**27,787.66**
01/01/2017 07:00	8,057.12	844.95	1,983.28	684.49	9,507.35	2,146.56	4,156.67	993.64	**28,374.06**
01/01/2017 08:00	8,125.76	874.58	2,015.31	710.23	9,881.60	2,137.65	4,286.08	1,019.34	**29,050.55**

ERCOT is divided into eight zones in general: COAST, EAST, FWEST, NORTH, NCENT, SOUTH, SCENT, and WEST.

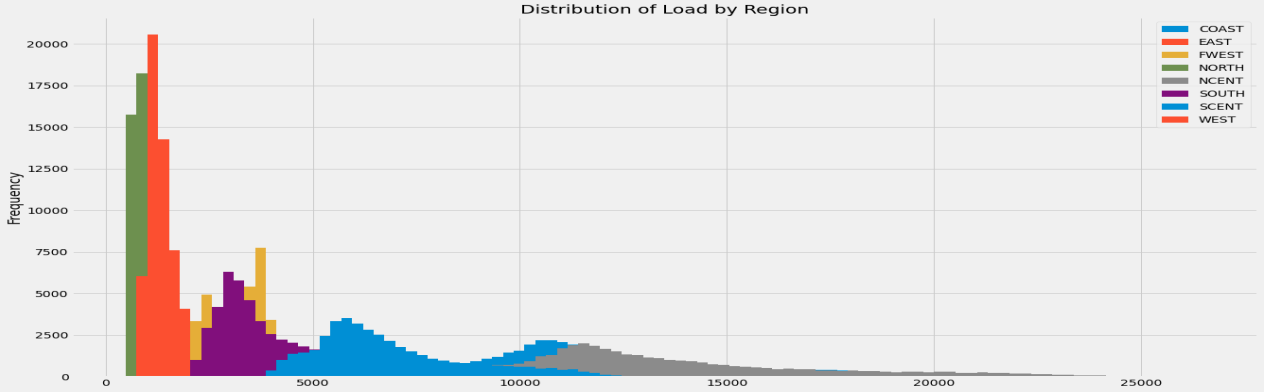


Figure 4: Distribution of Texas ERCOT Load by Region

The hour data is segmented by day, with ERCOT's peak hours being 7:00-22:00 on weekdays and 22:00-7:00 on weekends. Because this work is associated with UTRGV and is located in Texas's South Zone, we chose to train, test, and validate our model for the South Zone.

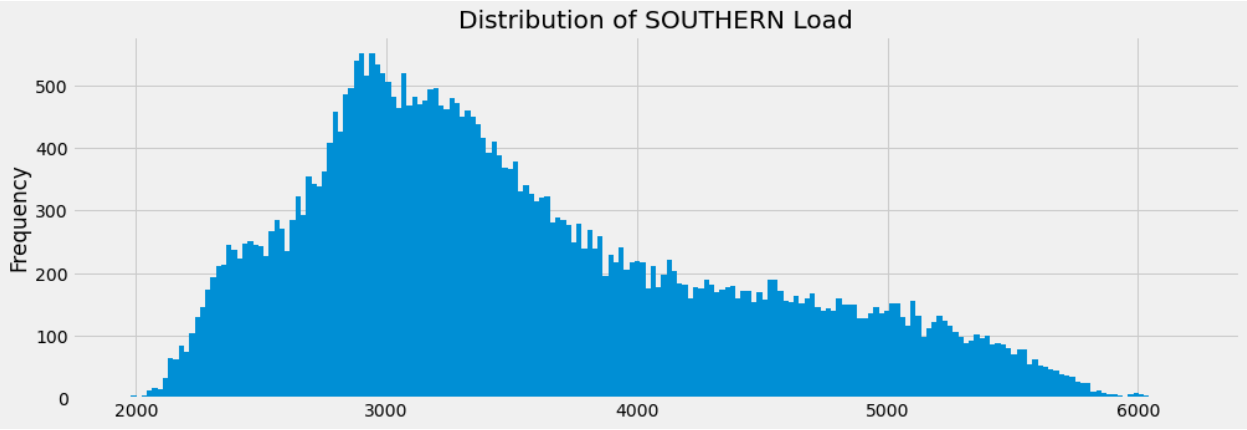


Figure 5: A Schematic representation of Southern Load distribution

Defining and resolving the dynamics of these types of problems is difficult. The difficulty with classic statistical approaches such as autoregressive models (AR) is that their success is frequently predicated on an implicit assumption of the observed process's stationarity or

linearity. These classical models perform well when the time series are linear or near-linear, but they are incapable of capturing the nonlinear patterns inherent in the observed process. That's why Deep learning based Machine Learning knowledge comes handy.

1.4 Research Objectives

To make a sustainable Machine Learning models that can forecast Electricity Load for a long-, Short-, and Medium-term basis.

- Long term forecasts can reduce investment risk
- Medium term forecasts help in planning fuel purchases and scheduling plant maintenance
- Short term forecasts are essential in matching electricity generation and demand for grid reliability

1.5 Organization of the work

The following sections explain the major components of this thesis, which focuses on ERCOT load forecasting using a machine learning approach:

A Literature Review.

Methodologies

Results and Discussion

Conclusion & Future Works

The first chapter provides an overview of time series analysis and prediction, as well as a problem statement, research objectives, and some motivation for its application in the energy and power demand forecasting sectors. The second chapter provides an overview of time series

prediction strategies based on statistical methods, machine learning techniques based on classic supervised learning methods, and deep learning-based neural networks. Chapter three discusses the proposed deep learning-based time series prediction algorithm. The fourth chapter discusses and illustrates the outcomes of our suggested model, as well as a comparison of each model to a radar chart. Chapter five comes to an end with a summary of the main findings and suggestions for more research.

CHAPTER II

LITERATURE BACKGROUND

2.1 Machine Learning & Its Approaches

Machine learning is an algorithm that can learn from data and observations and produce a classification or prediction without having to manually code the program. So, why is machine learning being used now, despite the fact that it was first introduced in the late 1950s? There was a lack of computing and processing capacity in the early years after machine learning algorithms were invented, as well as a lack of storage resources to perform and maintain such intensive computational tasks. Data sets that were relevant to the investigation were also scarce. Machine learning algorithms improved as the number of features in a data set grew. Approaches to forecasting electrical load can be classified into three categories: statistical, artificial intelligence, and hybrid.

Time series models such as auto-regressive (AR), auto-regressive moving average (ARMA), auto-regressive integrated moving average (ARIMA), seasonal ARIMA (SARIMA), linear regression methods, multiple linear regression methods, and exponential smoothing methods are all examples of statistical methods. The accuracy of ARIMA-based methods is dependent on a number of external variables and can be enhanced further by including exogenous variables. They all work well in linear systems, but they don't work well in nonlinear systems. Making an accurate prediction of any observable reality is important. It aids in making better decisions in ambiguous situations. Predicting exact future values, on the other hand, is

extremely difficult. This is because of the uncertainty and nonlinearity that most real-world occurrences entail [1]. The widespread use of renewable energy and a variety of uncertain loads in power grids, as well as the importance of profitability, complicate short-and long-term load forecasting.

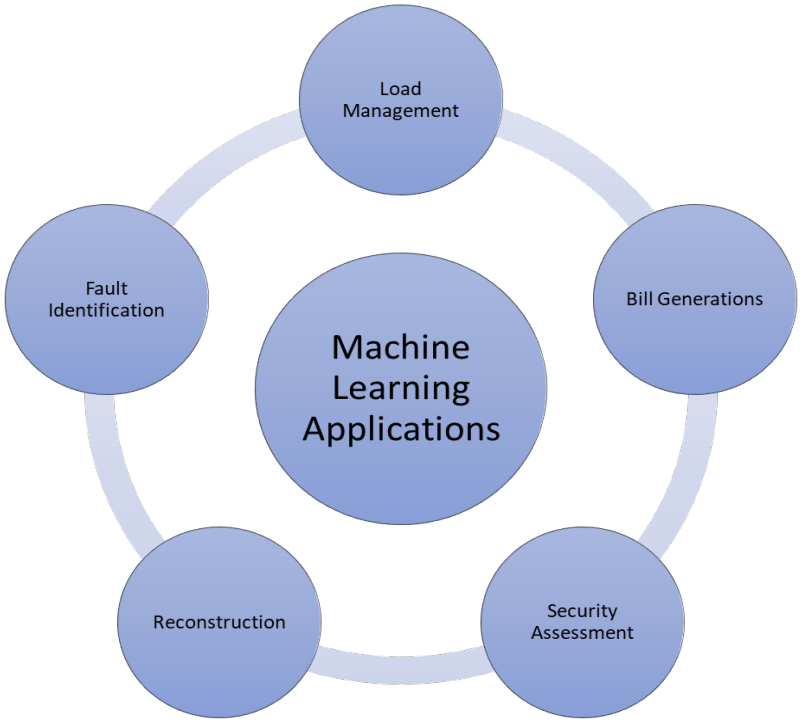


Figure 6: Machine learning applications in Load sector

In other words, one of the primary challenges facing future networks is load management and generation planning using highly uncertain load forecasts and stochastic energy generation. It has been said that neural networks and deep learning methods are important in this field as a result of recent research.

2.2 Electric Load Forecasting Techniques

Load forecasting is a significant subject in the electrical network, with numerous applications. Time series analysis and forecasting can be divided into two categories according to

the research methods used: time series analysis and forecasting. [2] proposes a time series short-term load forecast based on an autoregressive moving average (ARMA) model with non-gaussian process assumptions. Another time-series load forecasting method for power systems is the exponential smoothing model, which is based on the smoothing coefficients [3]. The smoothing coefficients determine the model's accuracy.

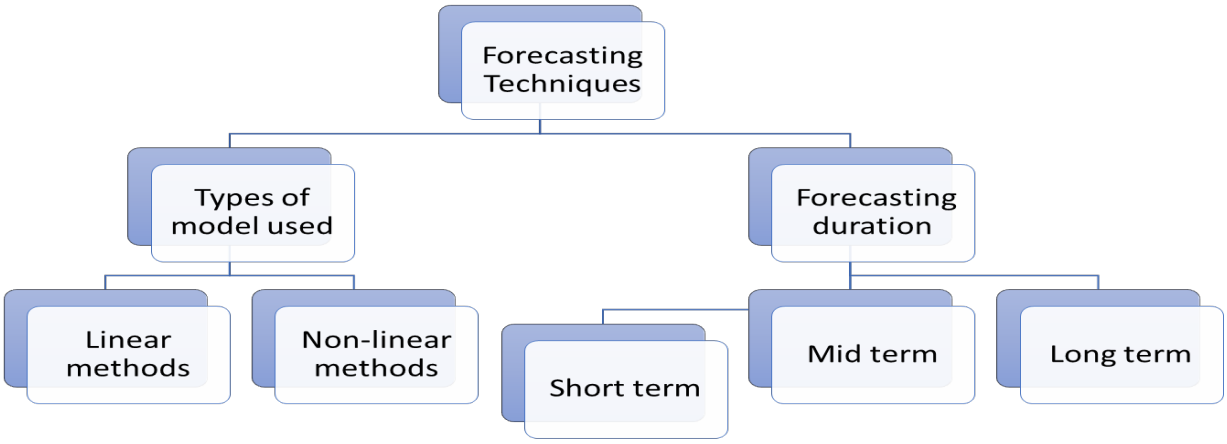


Figure 7: Electrical Load forecasting techniques

The exponential smoothing method was used to find the smoothing coefficient using historical load data and other random data. These time series forecasting methods are mathematical models that do not take into account aspects such as weather, day type, and other variables that have a significant impact on short-term forecasting to calculate energy usage. For short-term load prediction, many machine learning approaches have been applied. Kuhba et al. [4] trained a neural net for load prediction using a multi-layer perceptron with a back-propagation technique. In this method, weather changes, like temperature, humidity, cloudiness, and so on, were thought of as important factors that could affect the projected load.

2.3 Support Vector Machine

The "Support Vector Machine" (SVM) is a supervised machine learning technique that can solve classification and regression problems. It is, however, mostly employed to solve categorization difficulties. The support vector machine algorithm's goal is to find a hyperplane in an N-dimensional space (where N is the number of features) that distinguishes between data points.

One of the most frequently utilized machine learning methods in short-term load forecasting is the support vector machine [8]. The maximum daily load is predicted using the support vector machine (SVM) approach, which takes into account historical load, temperature, and day type (e.g., holiday) [9]. Because SVM has poor processing speed, a forecasting approach using Support Vector Machine with much less training set and faster processing time has been presented in [13]. In [10], a clustering-based SVM for short-term forecasting models was made, and it did better than the SVM alone.

Ma et al. [15] proposed a support vector machine (SVM) technique for predicting building energy usage in China. SVM is built on a foundation of statistical learning theory and mathematical optimization. This approach is used in a variety of domains, including time series forecasting, regression analysis, pattern recognition, and so on [16].

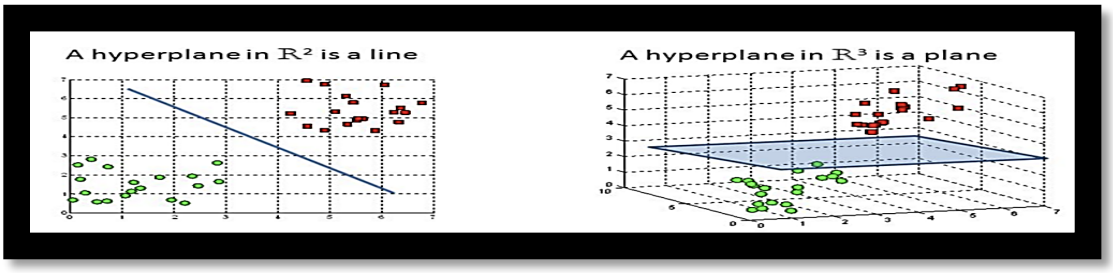


Figure 8: Hyperplanes in 2D and 3D feature space

The hyperplane's location and orientation are influenced by support vectors, which are data points that are closer to the hyperplane. We maximize the classifier's margin by employing these support vectors [17]. The hyperplane's location will be altered by deleting the support vectors.

The optimal number of training datasets is critical for short-term prediction model generalization. As there is no robust method for selecting the number of training datasets to train any data-driven model, Paudel et al. [18] took 12 days of training data, i.e. 1,152 sample datasets, because 12 days of training data outperforms other training data (5–20), as shown in Figure 12. In the Figure, the training datasets are raised from 5 days (480 data samples) to 20 days (1,920 data samples), and it is obvious from their model that the performance of the prediction model is higher with 12 training day datasets (R^2 = 0.96 and RMSE = 21). Furthermore, this 12-day dataset is subdivided into training/learning and validation, which is further subdivided into five-fold cross-validation.

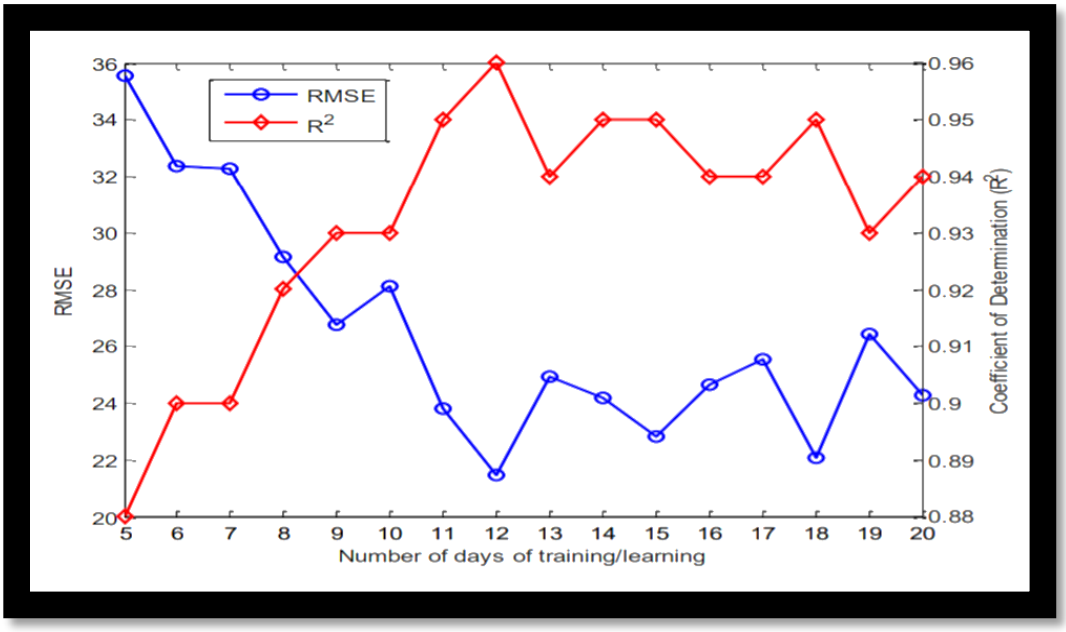


Figure 9: Optimal number of days selection as relevant training days [18]

Ma et al. [15] used multiple parameters, including weather data, such as yearly mean outdoor dry-bulb temperature, relative humidity, and global solar radiation as the inputs to improve the reliability of SVM in building energy consumption prediction.

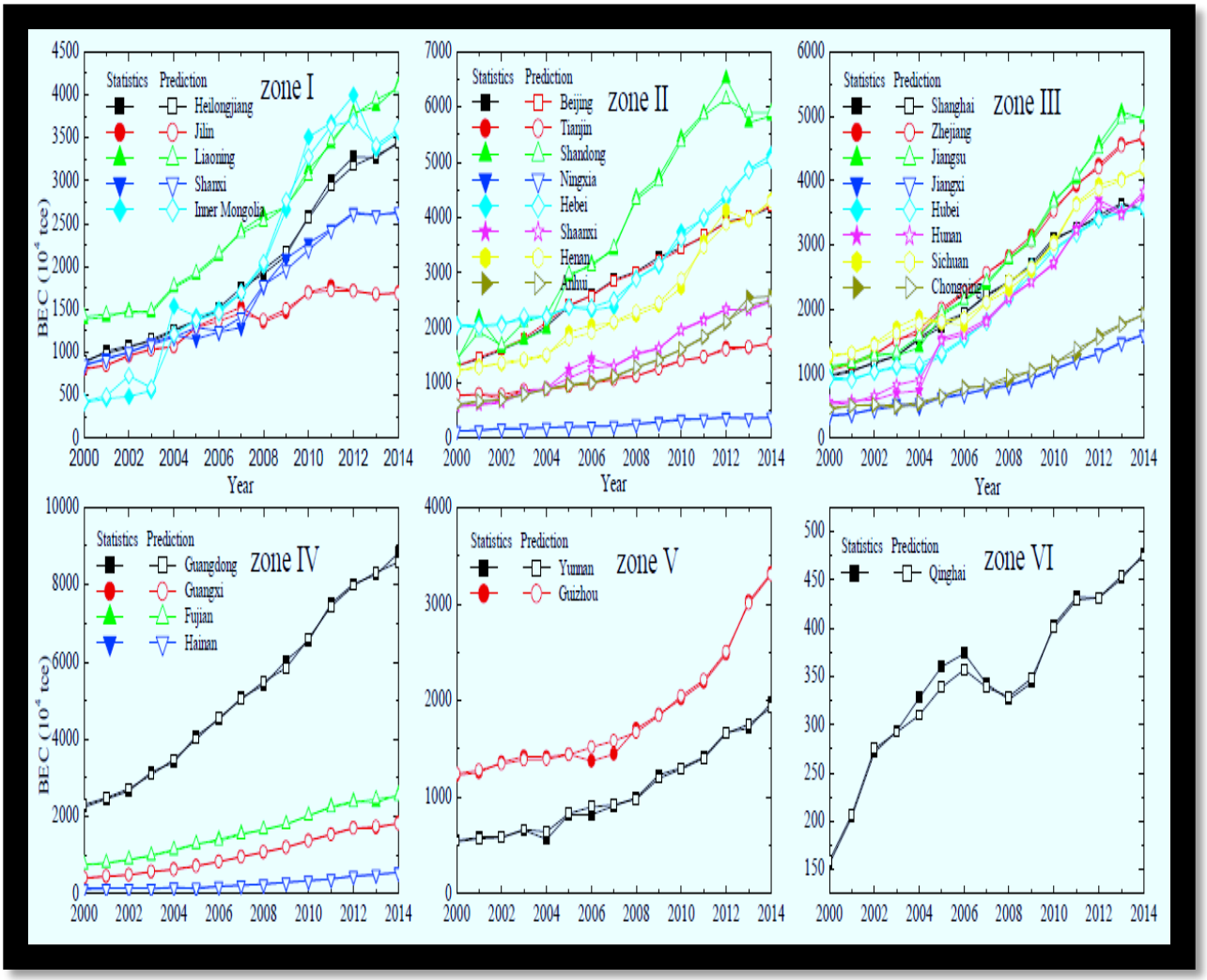


Figure 10: Comparison between the predicted value and statistic data through SVM of the china national building energy whole energy consumption [15].

The results of the statistical error tests reveal that their model can accurately estimate building energy consumption, with an MSE of less than 1E-3 and an R^2 greater than 0.991.

2.4 ARIMA & SARIMA

ARIMA is a common forecasting model that is an abbreviated form of the Auto-Regressing Integrated Moving Average. The word "autoregression" refers to the use of previous values of the same variable in the construction of regression models, i.e., the future forecast is a linear combination of previous values of the variable under consideration. In addition to providing contrasting approaches to the problem, the ARIMA and exponential smoothing models are the two most extensively utilized techniques for time series forecasting today. ARIMA models are different from exponential smoothing models, which try to capture the trend and seasonality of the data. Instead, they try to describe the autocorrelations in the data.

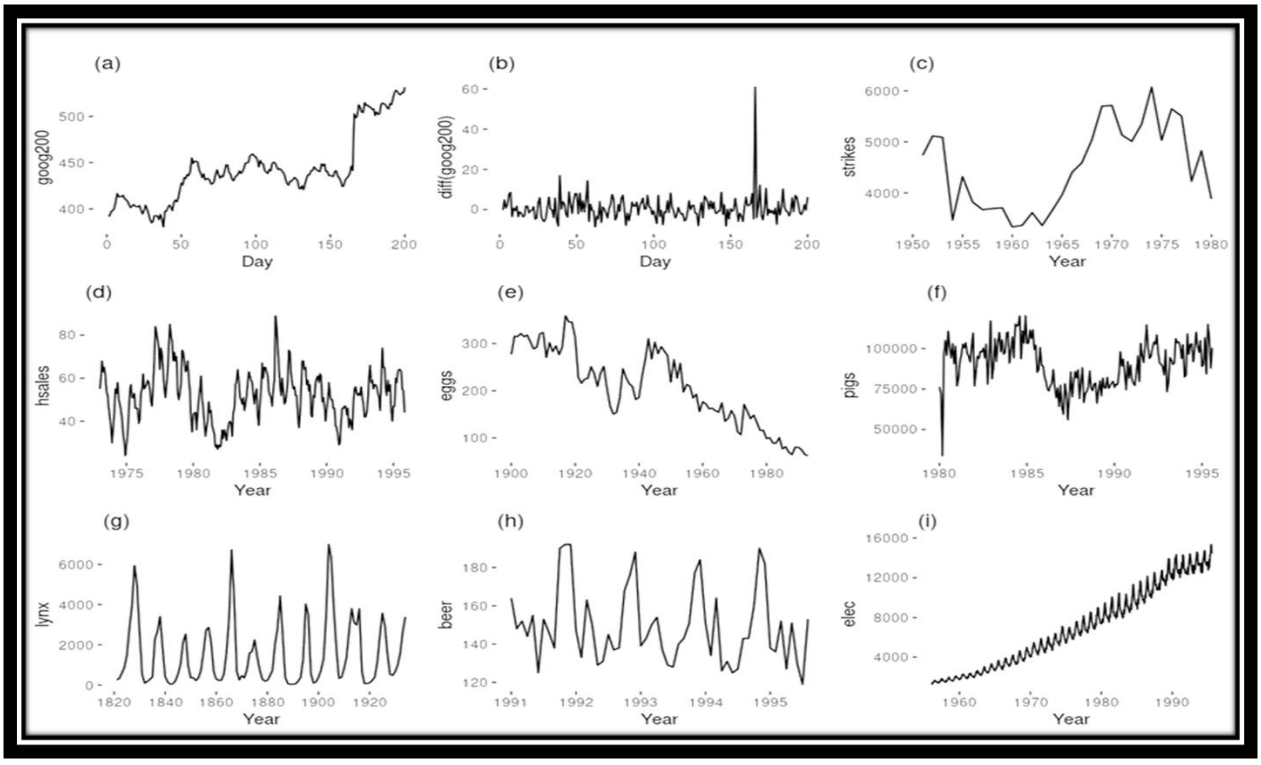


Figure 11: Seasonality vs Stationary modeling (only (b) and (g) are stationary series) [15]

An autoregressive integrated moving average (ARIMA) model and a non-linear autoregressive neural network (NAR) model are presented by Nichiforov et al. [19] for energy

consumption forecasting. ARIMA models include a single dependent variable (Yt), which is a function of previous Y values and the standard errors (Et). ARIMA models may accommodate any continuous result (such as rates or means) as well as huge counts that are not constrained by zero since they assume mistakes are normally distributed. In recent years, generalized linear models have been used to show data that is serially correlated [21]. ARIMA can't be used with small numbers that follow a Poisson distribution.

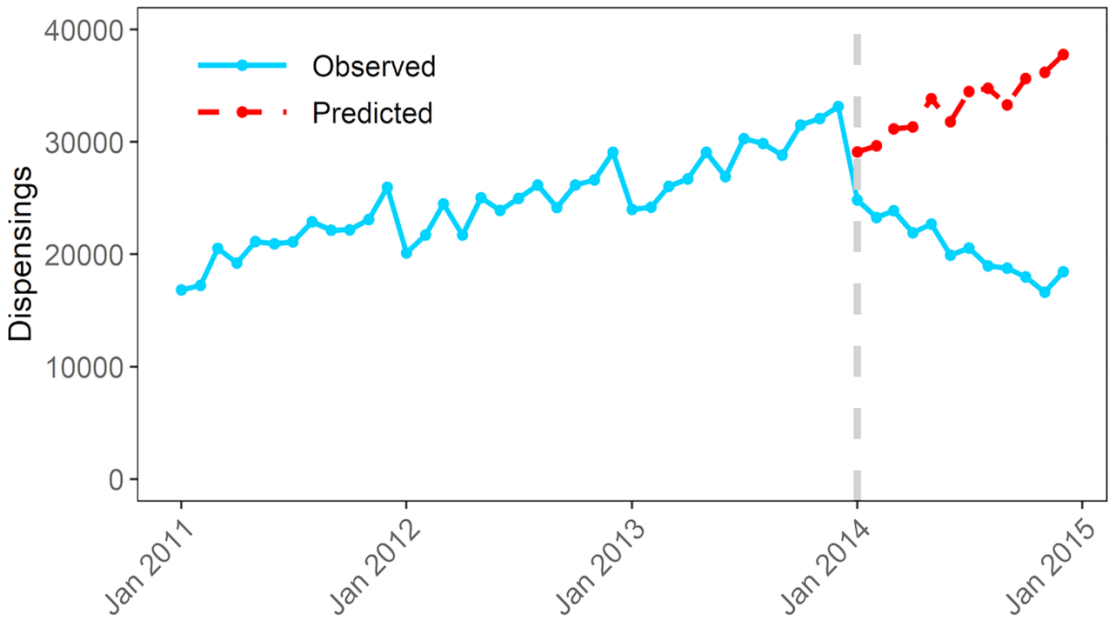


Figure 12: Observed values and predicted values in absence of intervention based on ARIMA model [21]

ARIMA stands for auto-regressive, which means we want to anticipate time series values based on previous periods. I is for integrating, which is an upward or downward trend that we utilize to eliminate. MA stands for moving average, and it informs the mistakes from one period to the next. S—seasonality—is a new concept here.

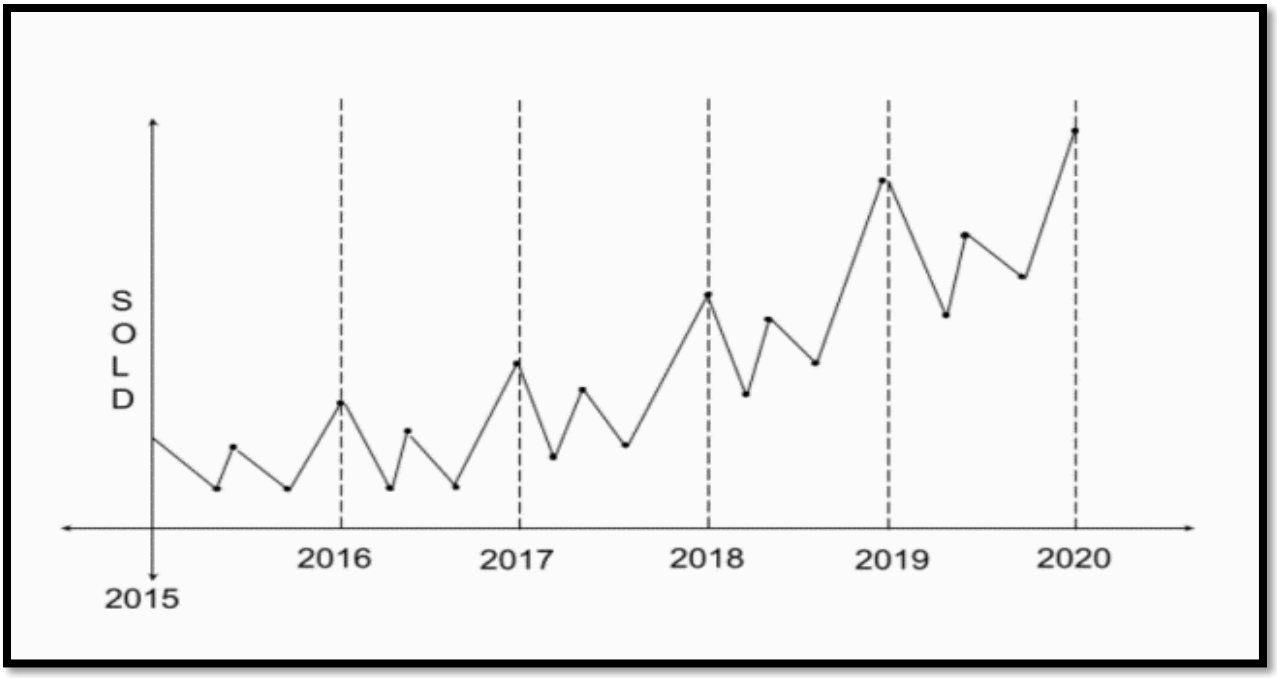


Figure 13: Discovering W-pattern in SARIMA model

How do we know that the seasonal ARIMA (SARIMA) model should be used? The above illustration depicts seasonality. We find a very distinct W-type pattern repeating, indicating the presence of seasonality. m is the seasonal factor in SARIMA (P, Q, and D). It is the number of seasonal time steps in a single season. Each year is divided into four quarters in the graph above. We now have a m value of 4.

2.5 History of Neural Networks

Neural networks are used in machine learning to teach a computer how to accomplish a task by assessing training examples [24]. Typically, samples are pre-labeled by hand. For example, thousands of tagged photographs of vehicles, buildings, paper cups, animals, and other objects can be sent into an object recognition system, which will look for visual patterns associated with specific tags.

A neural network is a densely connected network with hundreds, if not millions, of simple processing nodes that resemble the human brain. Today, the majority of neural networks are "feed-forward," which implies that data flows through them in only one direction and that they are organized into layers of nodes. A single node can receive data from numerous lower-layer nodes and transmit it to several upper-layer nodes. When a neural network receives input indicating whether it is accurate or incorrect, it learns. In order to correct any mistakes, the network will make adjustments in response to the feedback. Consider a soccer player who misses a clear goal opportunity. He'll go back to the dugout and think about what he did wrong. He'll remember what he did wrong the next time he has the same opportunity and adjust accordingly. Neural networks are very flexible and can learn very quickly [25]. They come in a variety of types, which we'll look at next.

2.5.1 Feed Forward Neural Network

This is the most basic and uncomplicated of the neural networks. Data only travels forward in one direction from the input to the output. Along the journey, the sum of the inputs' products and weights is determined. The completed product is routed to the outputs for processing. An et al. [26] combine multi-output FFNN (feedforward neural network) with EMD (empirical mode decomposition)-based signal filtering and seasonal adjustment to overcome the limitations of commonly used multi-step-ahead forecasting approaches, including error amplification and obliviousness to input-output dependency. They have a better model for MFES that predicts more accurately than other models that use the half-hour power demand series of New South

Wales, Australia.

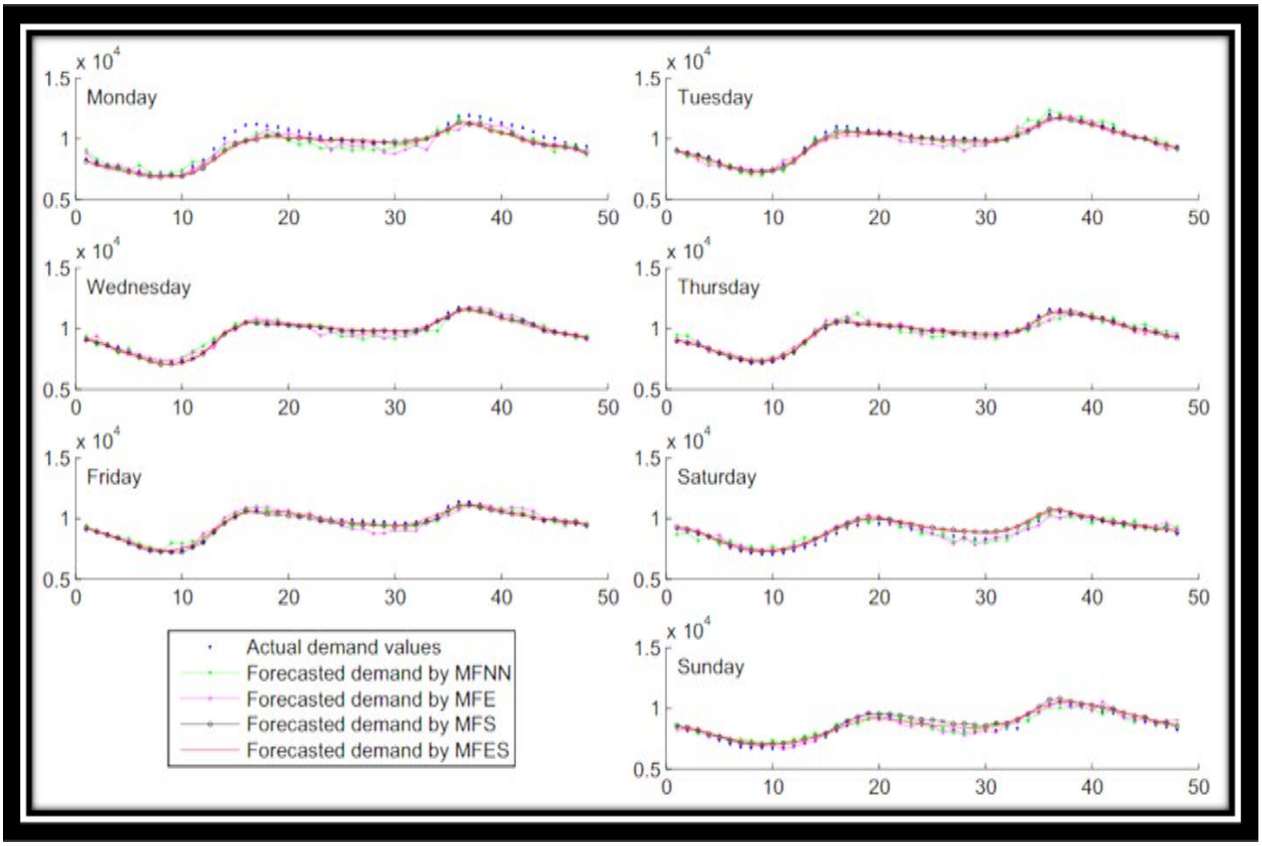


Figure 14: Feed Forward Neural Network to forecast the electricity demand for New South Wales [26]

Bhaskar et al. [27] used a feed-forward neural network (FFNN) to transform projected wind speed into wind energy prediction. For approximating arbitrary nonlinear functions, Zhang et al. [28] proposed this sort of NN as an alternative to the standard FFNN. Due to the local attributes of wavelets and the concept of adapting the wavelet form to the training data set instead of the parameters of the fixed shape basis function, WNNs provide superior generalization features. Function learning [29], nonlinear system identification [31], and time series prediction [32] have all been successful with WNNs. Energy prices were projected using an Adaptive Wavelet Neural Network (AWNN) by Pindoriya et al. [33].

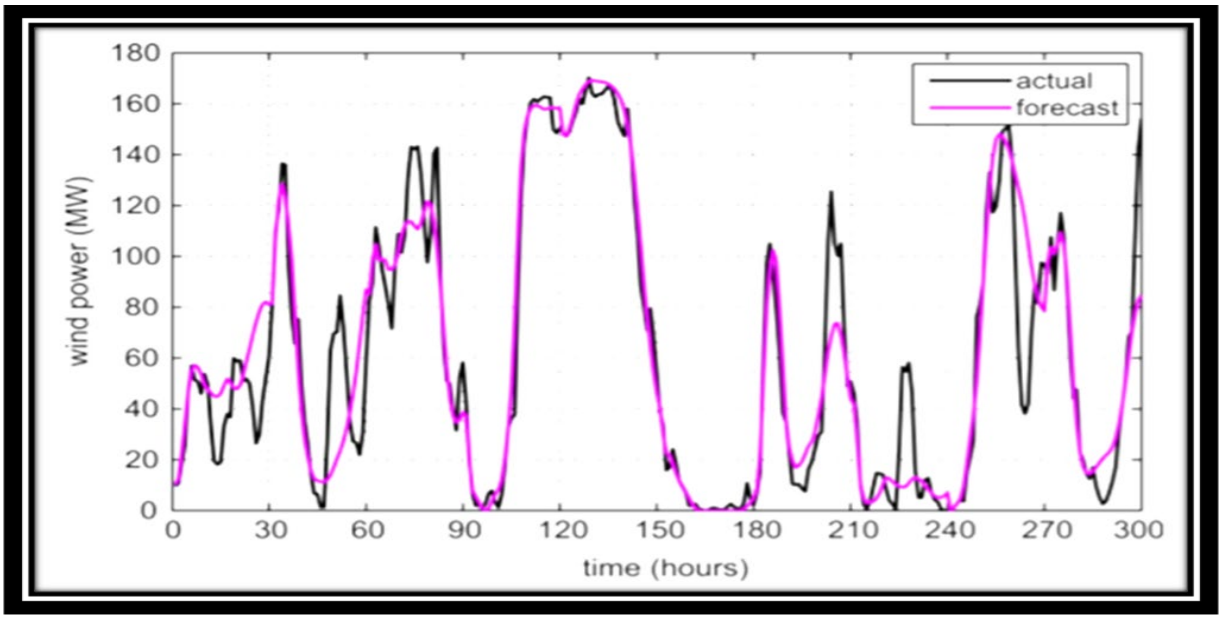


Figure 15: FFNN for the wind power forecast as a function mapper with wind power and wind speed forecasts as inputs [27]

2.5.2 Multilayer Perceptron

A multilayer perceptron (MLP) is a sophisticated artificial feed-forward neural network made up of many perceptrons that work together to solve a problem. The input layers receive signals, the output layers make decisions or predictions based on the input, and an arbitrary number of hidden layers perform true calculations like the true MLP computational engine. Making them extremely versatile, MLPs can approximate work on any continuous function with a single hidden layer.

When solving supervised learning problems, this method is used frequently as you train on a large number of input and output pairs to learn how to represent the correlation (or dependencies) between inputs and outputs. During training, model parameters, such as weights and biases, are adjusted to increase the overall accuracy and robustness of the model. It is used to

alter the weights and biases in proportion to the error, which can be evaluated in a variety of ways, including root mean squared error (RMSE), to improve model accuracy.

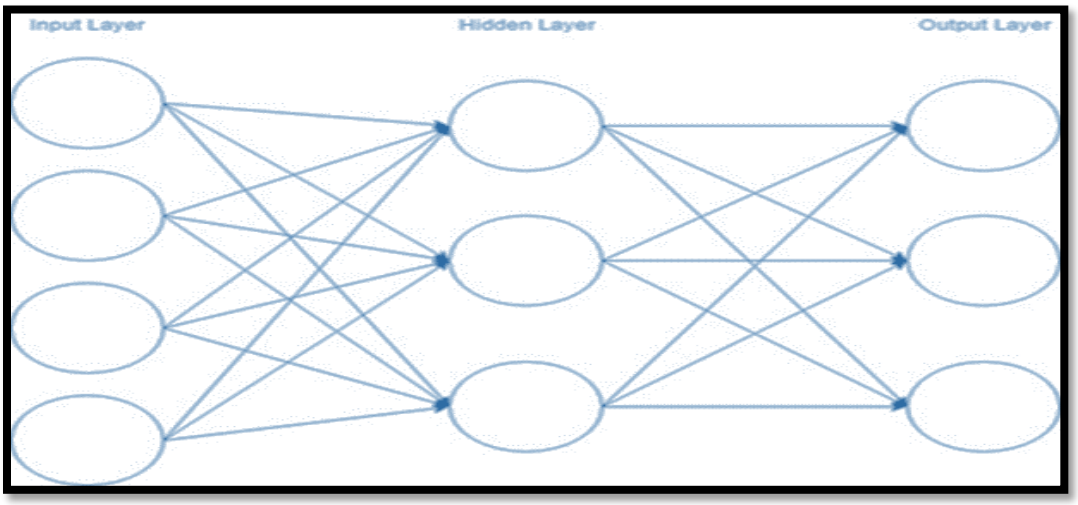


Figure 16: A basic MLP model

The MLP algorithm works as follows:

1. The inputs are pushed forward through the MLP in the same way as in the perceptron by taking the dot product of the input with the weights that exist between the input layer and the hidden layer (WH). The hidden layer receives a value as a result of this dot product. However, we do not handle this value in the same way that we would with a perceptron [22].

2. In each of their calculated layers, MLPs use activation functions. Rectified linear units (ReLU), sigmoid function, and tanh are just some of the activation functions to consider. Any of these trigger functions can be used to send the calculated output to the current layer.

3. Take the dot product with the correct weights and push the calculated output in the hidden layer through the trigger function to the next layer in the MLP.

4. Continue with steps 2-3 until you reach the output layer.

5. The calculations will be used in the output layer for a backpropagation method that corresponds to the MLP trigger function (in the case of training) or a decision based on the output (in the test case) will be made.

Pelka et al. [23] proposed a neural network based on MLPs for evaluating real-world data, including monthly power demand for four European countries: Poland (PL), Germany (DE), Spain (ES), and France (FR) (FR). The data is drawn from the ENTSO-E repository, which is open to the public (www.entsoe.eu). They span the years 1998 to 2014 for Poland and 1991 to 2014 for the other countries. They use previous data to develop forecasting models for 2014.

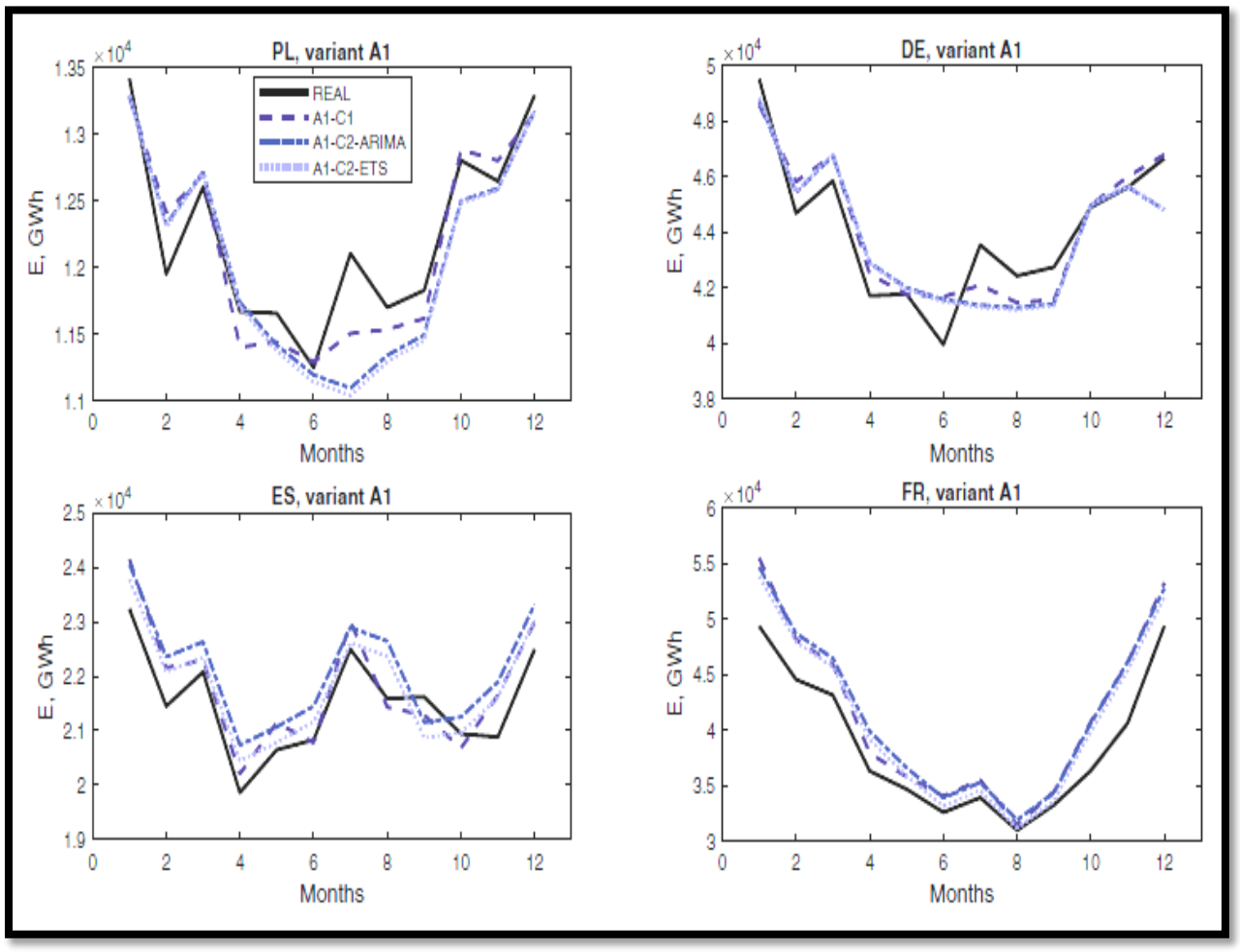


Figure 17: MLP model to Real and forecasted monthly demand prediction for A1 variant [23]

2.5.3 Convolution Neural Network

Convolutional neural networks are comparable to traditional neural networks in that they are made up of neurons with learnable weights and biases. Each neuron receives some inputs, does a dot product, and perhaps responds with a non-linear function. From raw picture pixels on one end to class scores on the other, the complete network continues to represent a single differentiable scoring function. They also keep a loss function (e.g., SVM/Softmax) on the final (fully-connected) layer, as well as all of the tips/tricks we developed for learning regular neural networks. A ConvNet's architecture is similar to the connecting pattern of neurons in the human brain and was inspired by the arrangement of the visual cortex. Individual neurons only respond to stimuli in a narrow section of the visual field known as the receptive field [30]. A group of similar fields will encompass the full visual region if they overlap.

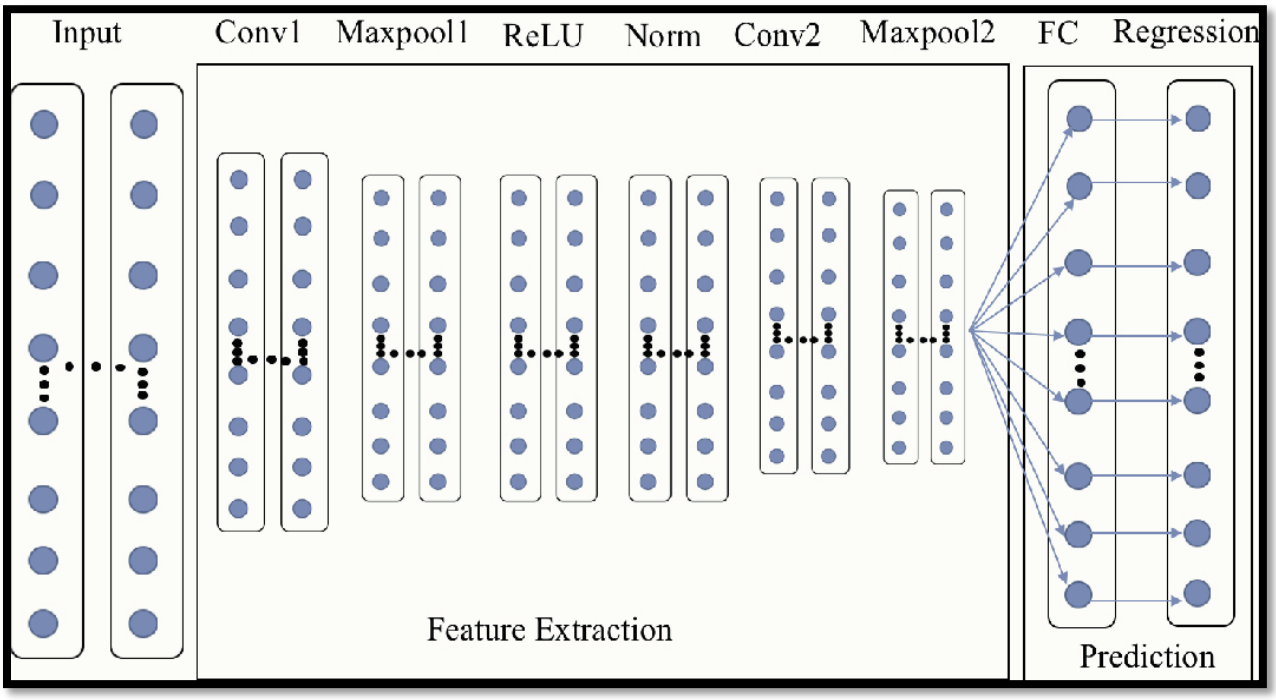


Figure 18: The Regressive Convolution Neural Network to predict Electricity demand forecasting [34]

Zhang et al. [34] suggested an RCNN (Regressive Convolution Neural Network) model with a substantial computation overhead. Then, to estimate power use, they use RCNN to extract features from the data and a Regression Support Vector Machine (SVR) trained with those features. Their RCNN structure only has eight layers (to avoid overfitting with insufficient data), with the input layer being an influential factor. Electricity consumption values are generated by the regression layer. The RCNN extracts characteristics of the influential elements during the training stage and checks if the MSE is convergent. It is possible to forecast the electricity consumption values of test data using the trained RCNN classifier. NA stands for normalization, FC stands for fully connected, and Conv stands for convolution layers.

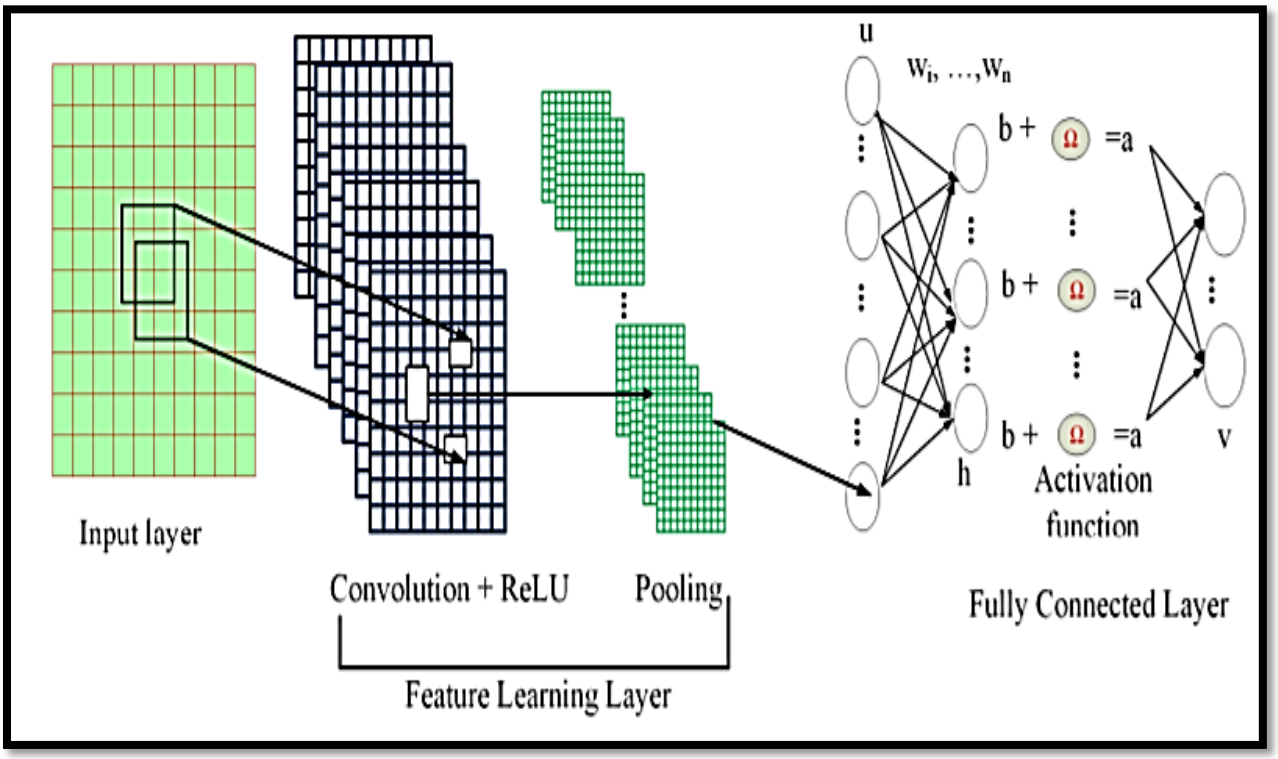


Figure 19: CNN for Real-Time Energy Management of Multi-Microgrid [35]

Samuel et al. [35] used a deep CNN in their research to do a one-step estimate of the aggregated energy cost for short-term applications. In their proposed deep CNN, they apply a

conditional Restrained Boltzmann machine (CRBM) to improve the fully linked layer. Their suggested deep CNN, which consists of three layers, is seen in Figure 9. The input layer is the initial layer, and it accepts data in sequential order. The second layer is the feature learning layer, which extracts features from the input data. By learning features from small squares of input data, convolution maintains the input. By employing small squares of data, a rectified linear unit (ReLU) compensates for the interaction effects between expected and actual data. Nonlinear effects are also taken into account. ReLU is a function that, by convention, returns 0 if it receives any negative input and the same value if it receives any positive input. In addition, the second layer is max-pooling, which has one for each convolution. Each pooling returns the greatest value of the predicted output of the convolution. The third layer is the fully connected layer. Its name refers to the fact that the k-neurons are connected to the max-pooling neurons.

Huang et al. [46] propose a deep learning architecture based on an ensemble of convolutional blocks acting on subsets of the input data. The algorithm is used to anticipate individual residential demands for the day ahead using data from an Irish smart metering electricity customer behavior trial (CBT). The work focuses on attaining a short training period while maintaining a high level of accuracy, with the proposed model getting the best of both with an MAE of 0.3469. Khotanzad et al. [47] describe a straightforward MLP regressor that incorporates a sophisticated feature selection algorithm based on prior consumption and weather data. Farsi et al. [48] propose a Short Term Load Forecasting (STLF) model that combines CNN and LSTM layers. Atef et al. [49] provide a comparison of recurrent neural networks (LSTM) and support vector regression (SVR), demonstrating that the bidirectional LSTM model outperforms both the unidirectional LSTM and SVR models. Numerous studies have used machine learning to improve STLF results in order to overcome the disadvantages of this

statistical method. Jain et al. [63] used SVR to make a model for predicting how much energy a building will use in the future.

2.5.4 Recurrent Neural Network

A recurrent neural network (RNN) is a kind of artificial neural network that operates on sequential or time-series data. These deep learning algorithms are frequently used to solve ordinary or temporal issues like language translation, natural language processing (NLP), speech recognition, and image captioning; they are embedded in popular apps such as Siri, voice search, and Google Translate. Recurrent neural networks, like feedforward and convolutional neural networks (CNNs), learn from training input. They are distinguished by their "memory," which allows them to modify the current input and output based on information from previous inputs. While conventional deep neural networks presume that their inputs and outputs are independent, recurrent neural networks' outputs are dependent on the sequence's prior parts. While future events can also be used to forecast the output of a series, unidirectional recurrent neural networks cannot account for them in their predictions.

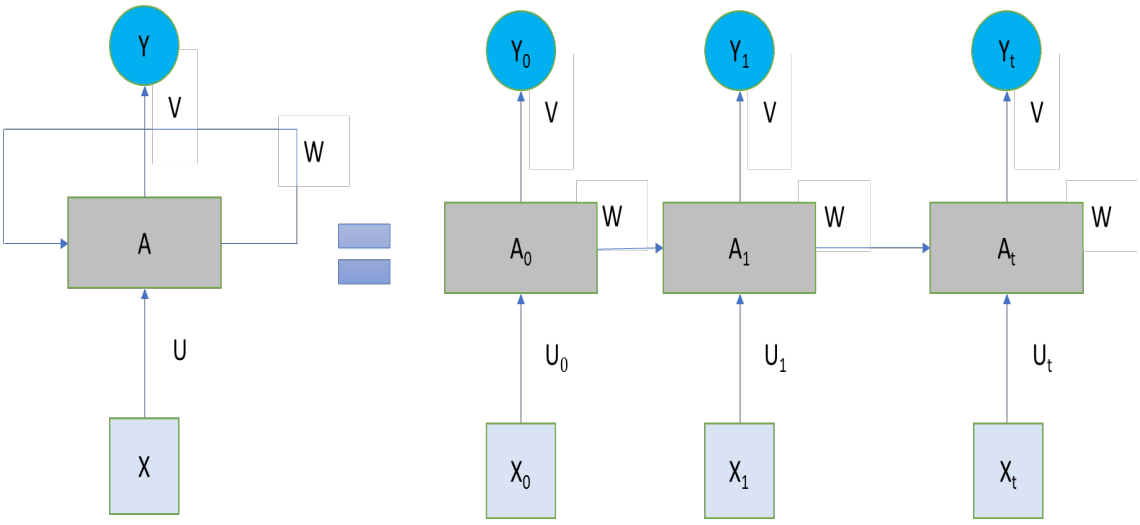


Figure 20: The architecture of a simple RNN unit

In Zhang et al. [69], they came up with a way for RNNs to predict short-term load by taking into account multiple time series (MTS) that had four information sequences (short-term, cycle, long short-term, and cross long short-term).

The fundamental premise of recurrent neural networks is that traditional ANNs are incapable of making a direct connection between previous and subsequent data and correcting errors. Backpropagation can address this shortcoming of the conventional ANN. These networks, dubbed recurrent neural networks (RNN), employ backpropagation to compare the error between the network's input and output until the error falls below a predefined threshold. A common application of such networks is in text prediction software, where the network is trained to predict subsequent words based on previously typed words. Backpropagation is accomplished via network loops. Previous timestamps' output is used as the input for current timestamps. Fig. 3 illustrates the basic structure of an RNN where x_t is the input at timestamp t, and Y_t is the output at timestamp t.

In [55], a deep neural network model was utilized to forecast the weather using large amounts of data. Instead of stacked auto-encoders, a deep belief network was utilized to develop a hybrid model for the joint distribution of the weather predictor variables. Chen et al. also employed a deep belief network model to forecast the Huaihe River Basin's short-term drought index [54]. In this study, the accuracy and efficiency of the deep learning-based neural network model were found to be better than those of the backpropagation neural network, which is what most people know.

2.6 RNN vs Feed Forward Neural Network

The information in a feed-forward neural network only flows in one direction: from the input layer to the output layer, passing through the hidden layers. The data travels in a straight line through the network, never passing through the same node twice.

Feed-forward neural networks have no recollection of the information they receive and are poor predictors of what will happen next. A feed-forward network has no concept of time order because it only analyzes the current input [39]. Except for its training, it has no recollection of what transpired in the past. The information in an RNN cycles via a loop. When it makes a judgment, it takes into account the current input as well as what it has learnt from prior inputs. Another distinctive feature of recurrent networks is that they share parameters across all network layers. Unlike feedforward networks, which have different weights for each node, recurrent neural networks have the same weight parameter shared by each layer of the network [40]. However, these weights are still changed in the backpropagation and gradient descent processes to help the computer learn from its mistakes.

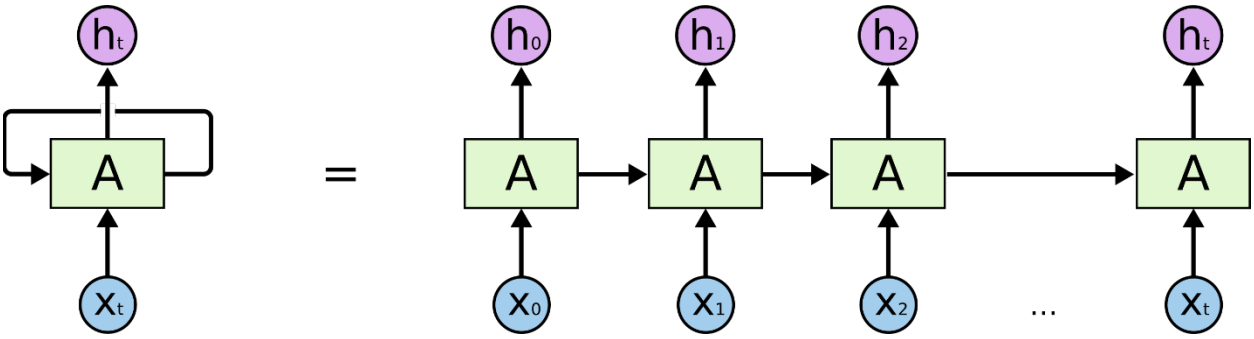


Figure 21 : An unrolled recurrent neural network [57]

Short-term memory is a problem for recurrent neural networks (RNN). Unless a sequence is sufficiently long, they will have difficulty transmitting information from older time steps to

later time steps. RNNs, for example, may not include important information at the start of the process when they process text.

The vanishing gradient problem is a challenge that recurrent neural networks encounter during back propagation. Gradients are values that are used to update the weights of a neural network. When a gradient shrinks as it propagates back through time, this is referred to as the vanishing gradient problem [56]. When a gradient value becomes incredibly small, it is not able to contribute much to learning. LSTMs and GRUs were developed to address the problem of short-term memory storage. It's a process called a gate inside them that can control the flow of information.

2.7 Long Short Term Memory Network (LSTM)

An LSTM behaves in a very similar way to an RNN cell. The LSTM is composed of three sections, each of which performs a distinct purpose. The components are various neural networks that determine which information about the cell's state is allowed to be transmitted. During training, these components might learn which knowledge is important to retain and which information is not. The first portion determines whether the information derived from the previous timestamp should be remembered or whether it is irrelevant and should be discarded entirely [58]. In the second section, the cell attempts to learn new information from the information that has been sent to it. Finally, in the third section, the cell moves the information from the current timestamp to the next timestamp, ending the cycle.

The gates of an LSTM cell are the three components that make up the cell. The first section is referred to as the "forget gate," the second section is referred to as the "input gate," and the final

section is referred to as the "output gate."

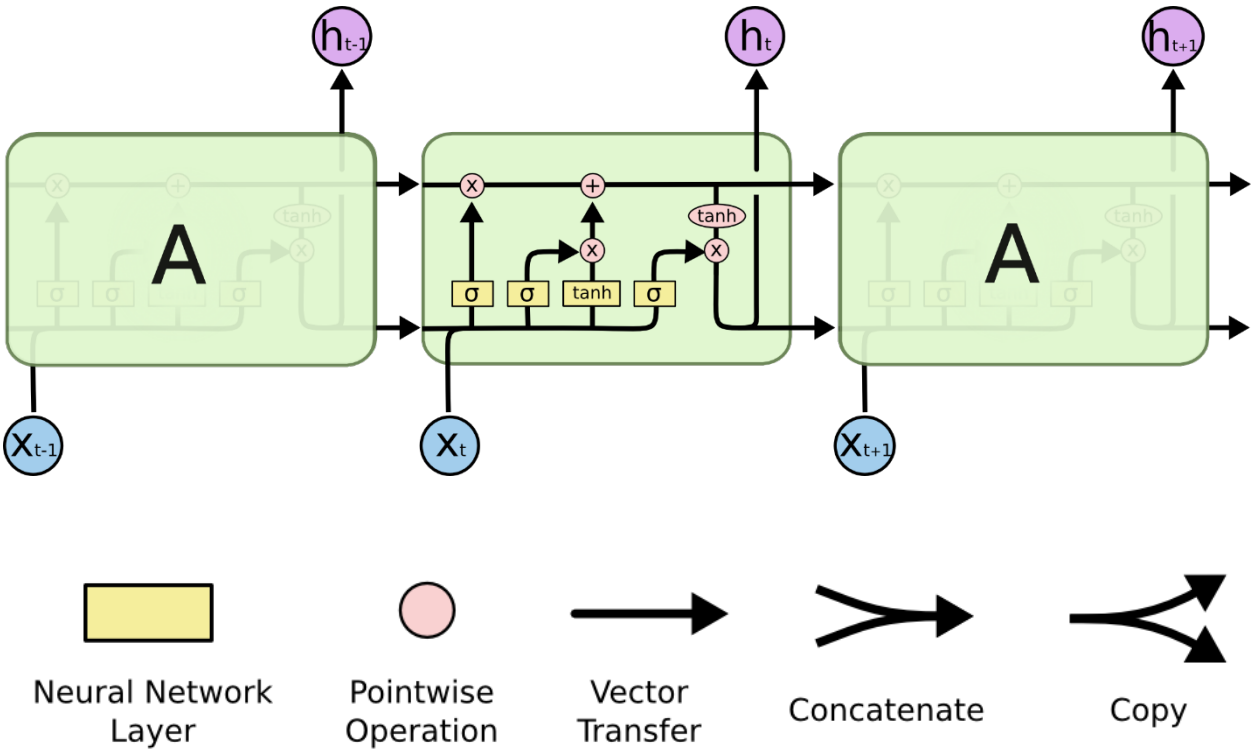


Figure 22: Four interacting layers based LSTM [57]

2.8 Gated Recurrent Unit (GRU)

The GRU is the most recent version of recurrent neural networks, and it functions similarly to the LSTM. Instead of using the cell state, they used the concealed state to transfer information. It also only has two gates: a reset gate and an update gate, both of which are identical. The update gate works similarly to the forget and input gates when used with an LSTM. It determines what data should be kept and what data should be discarded [56]. The reset gate is another gate that determines how much earlier data should be wiped. Each time, there are two input features: the input vector x(t) and the preceding output vector h(t-1). Each gate's output can be derived by performing logical operations on its inputs and performing nonlinear transformations on them. The following table summarizes the relationship between input and output:

$$r(t) = \sigma_g(W_r x(t) + U_r h(t-1) + b_r)$$

$$z(t) = \sigma_g(W_z x(t) + U_z h(t-1) + b_z)$$

$$h(t) = (1 - z(t)) \circ h(t-1) + z(t) \circ \hat{h}(t)$$

$$\hat{h}(t) = \sigma_h(W_h x(t) + U_h(r(t) \circ h(t-1)) + b_h)$$

where z(t) is the vector of the update gate, r(t) is the vector of the reset gate, and W & U are the parameter matrices and vectors, respectively. σ_g denotes the sigmoid function, while $\sigma_h$denotes the hyperbolic tangent [59]. GRUs do somewhat more tensor operations than LSTMs, making them marginally faster to train. When it comes to determining which is superior, there is no clear winner. Typically, researchers and engineers examine both methodologies to determine which is more effective for their specific application.

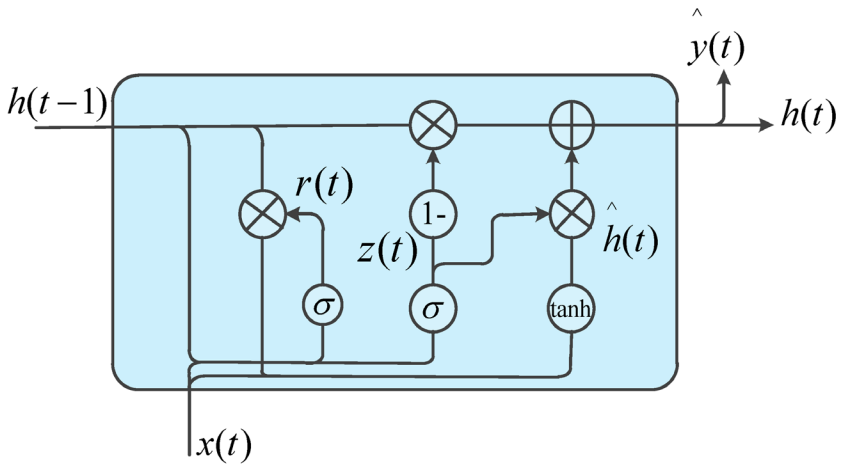


Figure 23: The structure of gated recurrent unit (GRU) network [59]

There is some extensive use of Neural Networks for short-term load forecasting in the literature. To mention a few: In [5], with the deep learning neural network and decision tree strategy short-term forecasting was measured and the final models is formulated as a mixed

regression method to determine a preliminary load forecast and a fuzzy inference system to minimize prediction error [6] [7].

Hernández et al. [45] describe an ANN model that forecasts electric demand 24 hours in advance using previously aggregated usage trends. The patterns are generated using a self-organizing map (SOM) and then clustered using the k-means algorithm. Gerossier et al. [60] developed a forecasting model for hourly household electric load using quantile smoothing spline regression and three input variables: the previous day's hourly load, the previous week's mid-load, and temperature. They estimated the predicted quantile distribution's mean and used it as a single-point forecast. These statistical approaches perform well for simple demand patterns but are somewhat inaccurate for complex demand patterns. Using linear regression analysis, it can be hard to give the right weight to variables that have a nominal or nonlinear correlation with the input variable [61,62].

Energy consumption data was gath ered from multi-family residential buildings on the New York City campus of Columbia University. Their predictions included those examining the effects of time (daily, hourly, 10-minute intervals) and spatial granularity (the entire building, by floor, by unit). The most efficient models were constructed using hourly consumption data at the floor level. Spatial granularity, in particular, was shown to have a significant effect on the predictive power of sensor-based forecasting models, as granular data at the floor and individual unit levels produced more accurate predictions.

Amber et al. [64] used MR and genetic programming (GP) to forecast the daily electric load of an administration building on the London South Bank University's Southwark campus. Although the GP model had a lower total absolute error (TAE) of 6% compared to the MR model's 7%, its training time was longer. Deep learning has since been applied to a variety of

challenges, consistently outperforming the state-of-the-art in a variety of applications, including robotic processing, object recognition, speech and handwriting recognition, picture classification, and natural language processing. Due to its success in these diverse applications, it has been expanded to time series issues, primarily in a semi-supervised fashion to enhance learning performance.

There are a few distinct deep learning algorithms for completing time series forecasting tasks in the literature, which may be found in many publications. For example, [50] employed a Deep Belief Network (DBN) to forecast time series using a DBN model refined by particle swarm optimization. So the new model outperformed other neural network models, like MLPs, SOFNNs, and ARIMAs, which are already used in the world of computer science.

CHAPTER III

METHODOLOGY

3.1 Electric Demand in Texas ERCOT

The dataset studied in the experimental investigation spans from 2017-01-01 H01:00 to 08/31/2021 H24:00. During this time, the consumption was measured every hour, resulting in a time series of 40850 measurements. The data was given by the Electric Reliability Council of Texas and is accessible for public use at [41].

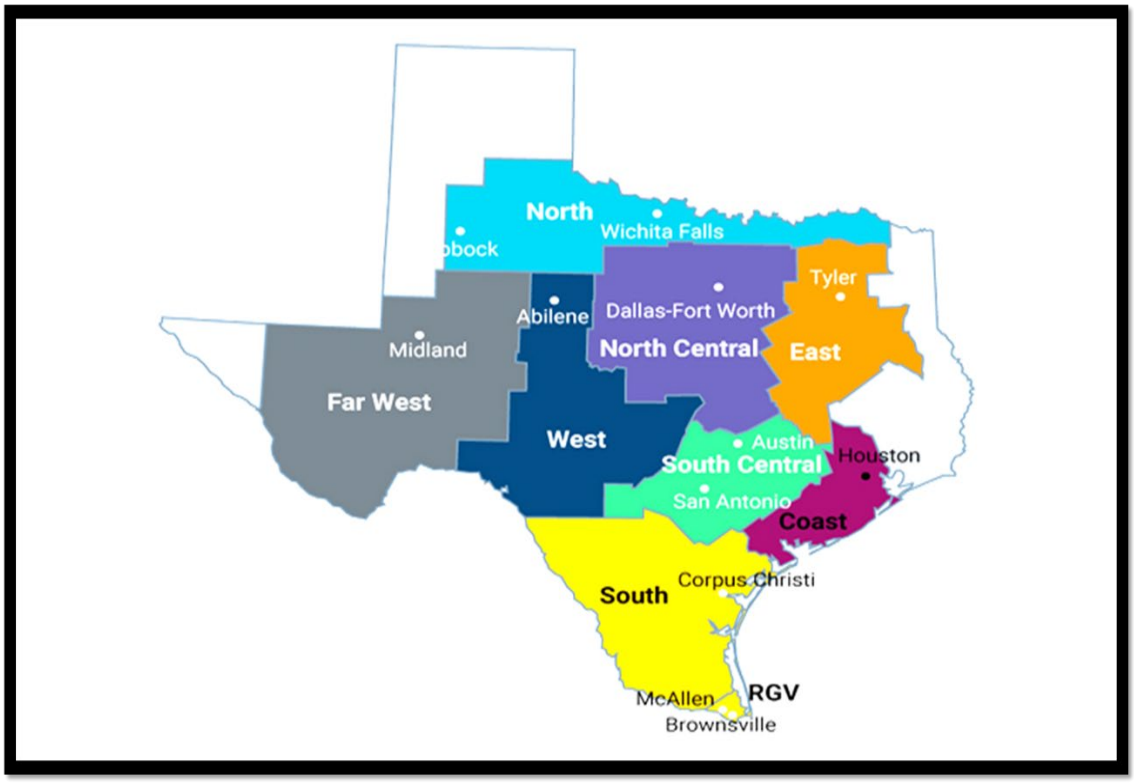


Figure 24: Distribution of Texas ERCOT Load by Region [80]

The dataset was divided into two sets: the training set contains 35000 samples from 2017-01-01 H01:00 to 12/29/2021 H07:00, and the test set has 5895 samples from 12/29/2021 H08:00 to 08/31/2021 H23:00.

3.2 Data Decomposition

Figure 25 provides statistics on electric demand at different scales. As can be seen, the time series exhibits both daily and weekly seasonality. In general, demand is higher during the week than on weekends, and it drops significantly at night on a daily basis.We conducted this in R to discover an interesting trend in the data.

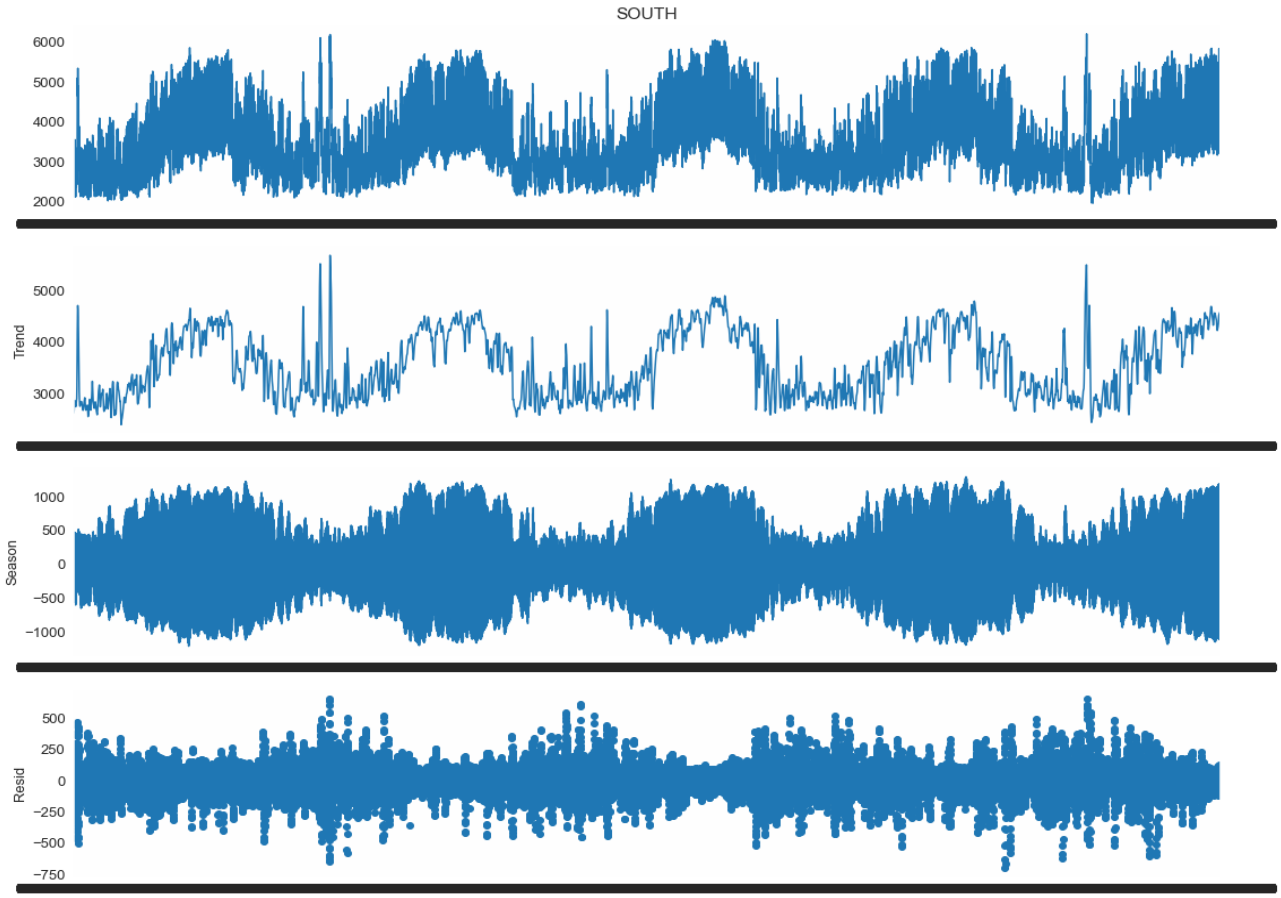


Figure 25: Line graphs showing the electric demand time series data at various scales (daily basis)

Figure 25 depicts line graphs presenting data from the southern demand at various scales. The breakdown is hazy. A cyclical trend is not a positive thing. It will be used in the annual rerun to show how to make a strong trend from the same data set. The period = 24 * 265 will be used.

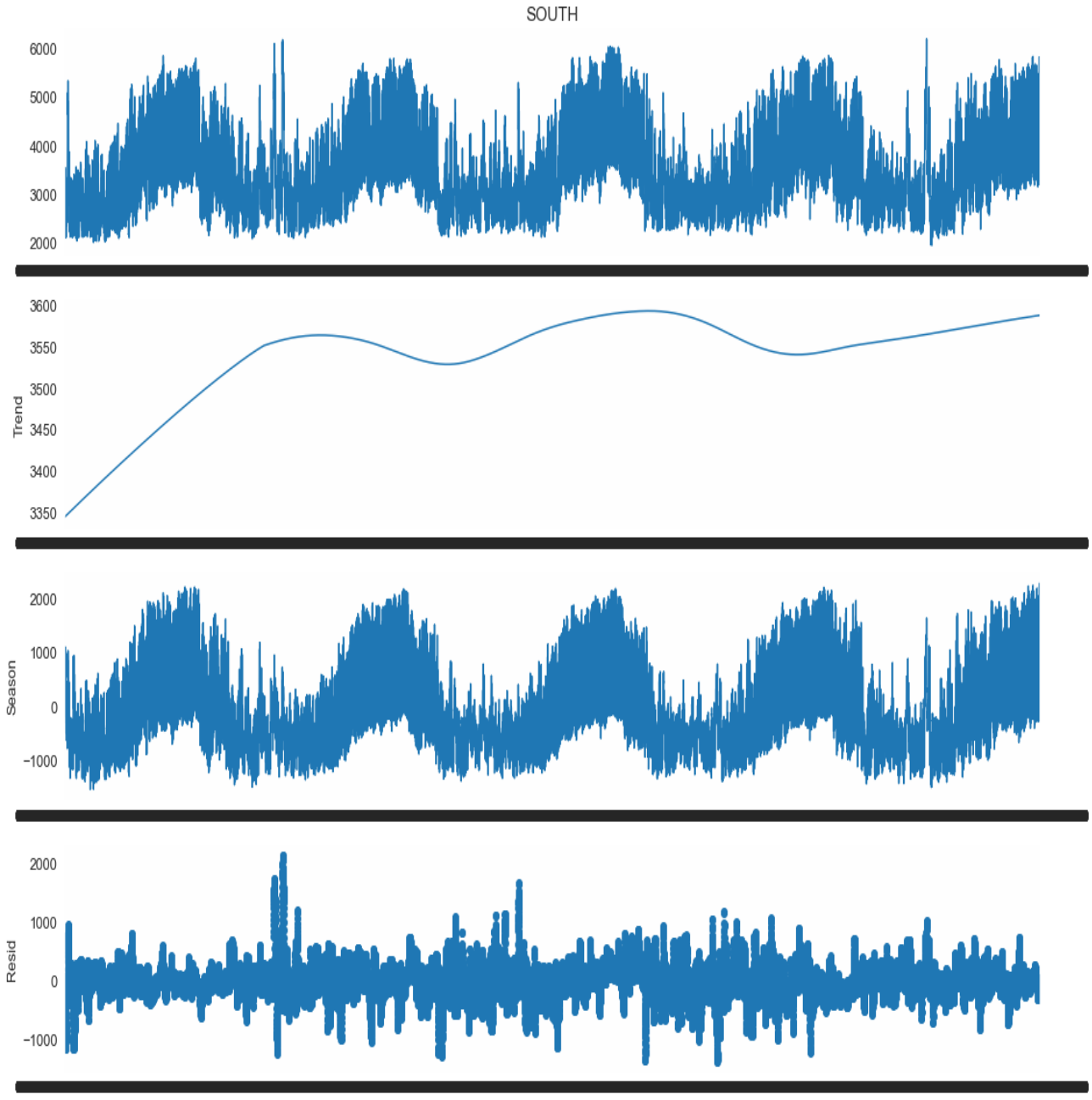


Figure 26: Line graphs showing the electric demand time series data at various scales (Yearly basis)

3.3 ARIMA

The future value of a time series is considered to be a linear function of multiple prior values of the original series and random mistakes in an ARIMA model [19].

The ARIMA Model consists of four steps [20]:

(1) The original sequence's stationary test. If the initial sequence is not stationary, the differential transform is used to modify it to fulfill the stability requirements.

(2) The parameters p and q in the ARIMA model will be determined by calculating the statistics that characterize sequence features (i.e. auto-correlation coefficient and partial auto-correlation coefficient).

(3) The model's unknown parameters are estimated, and the model's logic is examined.

(4) Diagnostic analysis to ensure that the model generated is compatible with the data features observed.

3.4 Neural Networks

The term "neural networks" refers to a subset of machine learning algorithms that are inspired by the biological neural networks found in the human brain. The human brain is divided into three distinct regions: the temporal lobe, the occipital lobe, and the frontal lobe. It was through the combination of these components and neurons that the numerous types of neural networks available today were discovered. In 1958, psychologist Frank Rosenblatt invented the first artificial neural network, the Preceptron. Its objective was to mimic the way the human brain processes information and train it to recognize objects. Preceptrons, like neurons, are modeled after the human brain. The perceptron algorithm is a supervised learning approach for

binary classifiers in machine learning. A binary classifier is a function that determines whether the numbers contained in an input vector are members of a particular class. The concept was to perform complex tasks with a group of simple mathematical neurons. Afterwards, the general feed-forward process was done, which involved layering a group of preceptrons as an input layer as well as hidden layers and an output layer over each other.

Recent years have seen an explosion of research into neural networks. This is largely due to recent breakthroughs in deep neural networks. These improvements have had a significant impact on computer vision, natural language processing, and speech recognition. Prior to the advent of deep learning, the discipline was dominated by single layer neural networks using hand-designed features. In terms of learning and accuracy, deep neural networks have been demonstrated to outperform single-layer neural networks. The availability of low-cost storage and massive data sets contributes to the proliferation of neural network research.

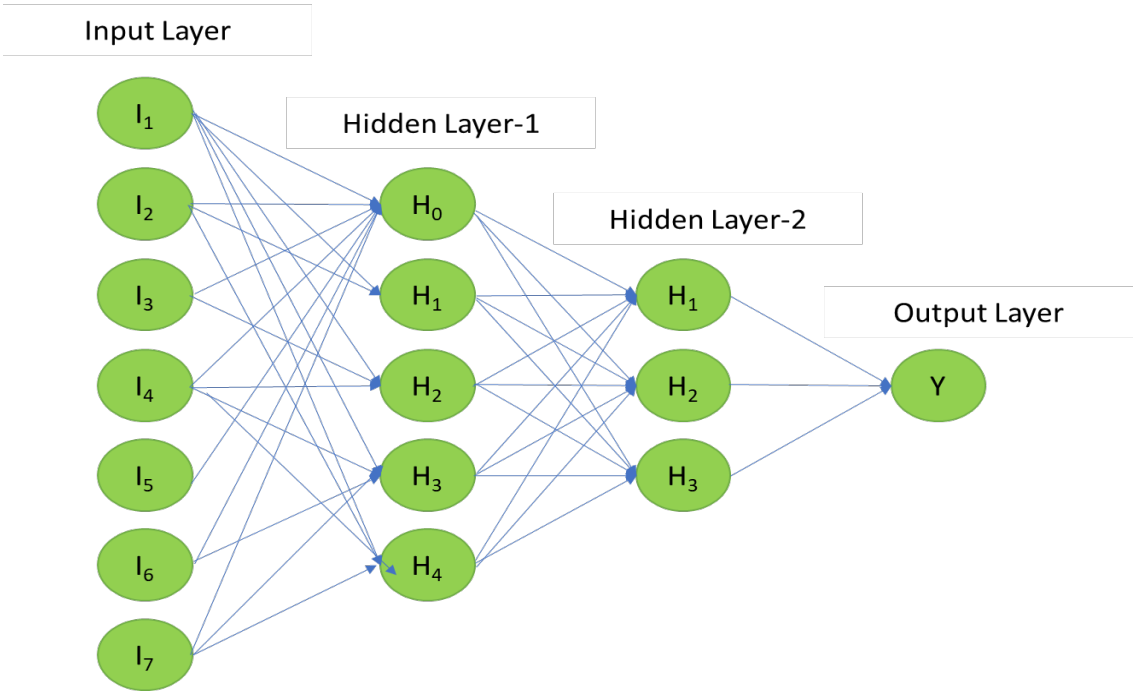


Figure 27: A simple Neural Network Architecture with 2 hidden layers

The more data is fed into a neural network, the more it can be modified to account for and generalize that data point. By increasing our resources for data collection, we may enhance the intelligence of our systems. This will be evident in the next few years as smart grids are implemented, which will collect data from the user's electric meter.

The Neural Network is composed of three layers: The initial data of the neural network is stored in the input layer. Hidden layers are layers that sit between the input and output layers and are where all computation occurs. In the output layer, you generate a result for the given inputs.The weights are initially set to zero for a simple preceptron before being passed through the layers. The unit step activation function then produces an output for each training sample. With the goal of minimizing errors, the weights are then updated based on the output values. In a layer, all of the weights are updated at the same time.

The capacity to run larger models with increased processing power has significantly improved the results we can achieve. With more neurons in the model, the fit can get more difficult. The size of modern neural networks is still an order of magnitude lower than the size of the human brain. While modern artificial neurons don't look like the neurons in the brain, this comparison shows how well they can do math.

3.4.1 Backpropagation

Although backpropagation was introduced in the 1970s, its importance was not fully appreciated until 1986, when David Rumelhart, Geoffrey Hinton, and Ronald Williams [70] published a seminal paper. This article describes several neural networks in which backpropagation performs significantly better than previous learning methods, allowing neural nets to solve previously unsolvable problems. Backpropagation is now the algorithm of choice

for neural network learning. The basic learning approach of backpropagation is to start with untrained networks, present a training pattern to the input layers, pass the signals through the net, and determine the output at the output layer. When the network outputs match the desired outputs, the error function is minimized, which is some scalar function of the weights. As a result, the weights are adjusted in order to reduce the error.

3.4.2 Activation Functions

A neuron in a neural network computes the weighted sum of input and adds a bias before passing into the activation function, which then decides whether to send that value to the next neuron or not. The activation function maps the resulting values between 0 and 1 or -1 to 1, depending on the function used.The Activation Functions can be basically divided into 2 types: linear Activation Functions and non-linear Activation Functions. The output of a line or linear function is not restricted to a specific range. As a result, it isn't very good at dealing with the complexity or different parameters of the typical data that neural networks get. Some of the common non-linear activation functions are: (1) Sigmoid or Logistic Activation Function (2) Tanh or hyperbolic tangent Activation Function (3) ReLU (Rectified Linear Unit) Activation Function etc.

3.4.3 Sigmoid Functions

The sigmoid function is a continuous, monotonically increasing function with a "S"-shaped curve. It possesses a number of intriguing properties, making it an obvious choice for nodes in artificial neural networks [71]. The domain of the sigmoid function is defined as the set of all real numbers, R, and it is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

We use the sigmoid function because it exists between two points (0 and 1). As a result, it is particularly useful in models where the probability must be predicted as an output. Because the probability of anything only exists between 0 and 1, sigmoid is the best option.

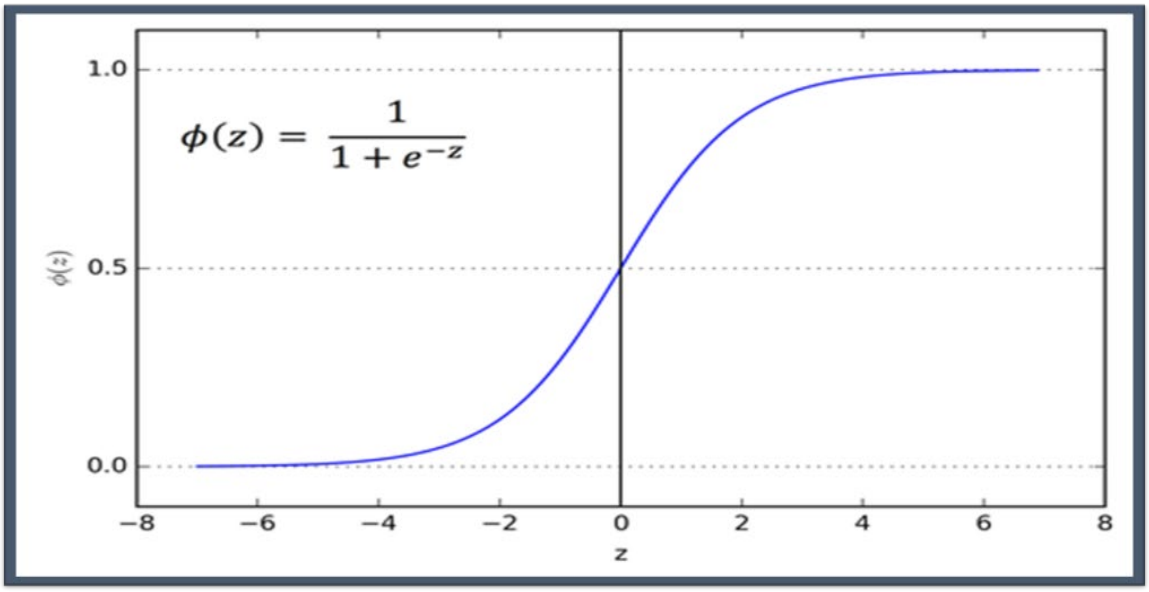


Figure 28: Sigmoid or Logistic Activation Function [77]

3.4.4 Tanh Activation Function

The Tanh (also "tanh" and "TanH") function is another name for the hyperbolic tangent activation function. It is visually similar to the sigmoid activation function and even has the same S-shape. It accepts any real value as an argument and returns a value between -1 and 1. The greater the input (the more positive), the closer the output is to 1.0; the smaller the input (the more negative), the closer the output is to -1.0 [72].

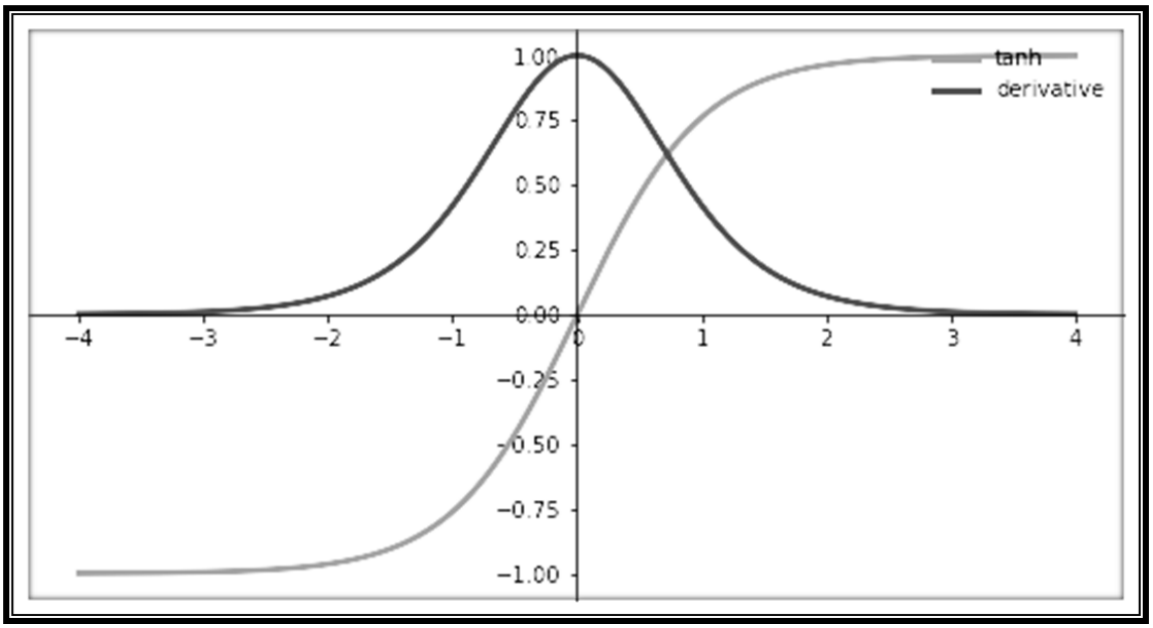


Figure 29: Tanh Activation Function [76]

3.4.5 ReLU Activation Function

In deep learning models, the Rectified Linear Unit is the most commonly used activation function. If the function receives any negative input, it returns 0. However, if the function receives any positive value, x, it returns that value.

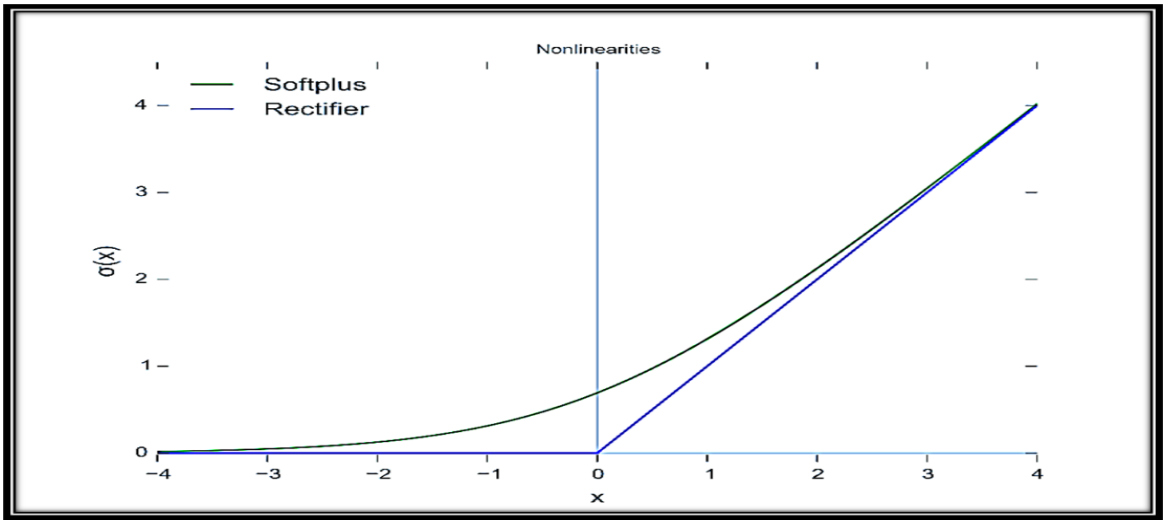


Figure 30: ReLU Activation Function [77]

As you can see from the figure 30, the ReLU is half rectified. When z is less than zero, f(z) is zero, and when z is greater than or equal to zero, f(z) is equal to z.

3.4.6 Loss Function

Starting with an untrained network, setting a training pattern on the input layer, passing the signals through the net, and determining the output at the output layer is the basic approach to learning. These outputs are compared to the target values, and any discrepancy is considered an error. To reduce the measurement error, the weights are then adjusted. A loss function must be defined to calculate the model error in neural networks. Mean squared error, mean absolute error, mean absolute percentage error, root mean squared error, and other loss functions are commonly used loss functions in neural network models.

(1) **Mean Squared Error:** An MSE is a measure of how close a fitted line is to data points. For every data point, we take the distance vertically from the point to the corresponding y value on the curve fit (the error) and square the value.

$$mse = \frac{\sum_{i=1}^{n} (y_i - \lambda(x_i))^2}{n}$$

(2) **Mean Absolute Error:** MAE is the difference between the predicted value and the "true" value. For example, if a scale states 90 KW but you know your true load is 89 KW, then the scale has an absolute error of 90 KW – 89 KW = 1 KW.

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

(3) **Mean Absolute Percentage Error:** MAPE is a metric that measures how accurate a forecasting system is. It can be calculated as the average absolute percent error for each time period minus the actual values divided by the actual values.

$$M = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

(4) **Root Mean Squared Error:** RMSE/RMSD is the square root of the mean of the square of all of the errors. The use of RMSE is very common, and it is considered an excellent general-purpose error metric for numerical predictions.

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$$

3.4.7 Optimizers

Optimizers are algorithms or methods for reducing losses by altering the characteristics of your neural network, such as weights and learning rate. Gradient Descent, Stochastic Gradient Descent, Mini-Batch Gradient Descent, Nesterov Accelerated Gradient, Adagrad, AdaDelta, and Adam are some of the optimization algorithms that can be used. The general but not obvious rules of neural networks are [73]:

- ✓ Adam is the best optimizer for training the neural network in less time and more efficiently.
- ✓ Use optimizers with dynamic learning rates for sparse data.
- ✓ If you want to use a gradient descent algorithm, min-batch gradient descent is the best option.

3.5 Recurrent Neural Network

Recurrent neural networks (RNNs) are distinguished by their topological connections, which allow them to handle sequential input while maintaining a recurrent hidden state. Many machine learning applications, such as video action identification [77], picture analysis and classification [78], machine deterioration prognosis, and remaining usable life prediction, have demonstrated promising results using RNN. RNN is well suited to learning variable-length features from sequence data. In the diagram, x is the model's primary input, o is the output matrix, xt and xt+1 are the inputs for the next sequential hidden state, and U, W, and V are the weight matrices that are identical. The benefit of this structure is that the model output is tied not just to the present time input, but also to an earlier arbitrary moment's input.

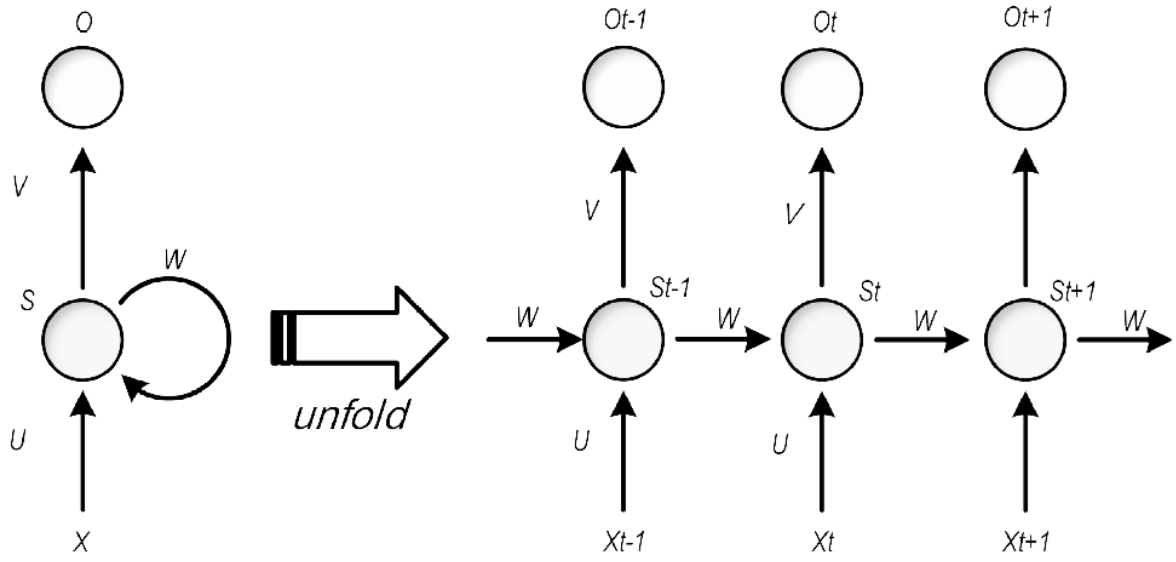


Figure 31: The schematic diagram of recurrent neural network model [77]

The normal RNN is incapable of retaining long-term memory information due to the tendency of gradients to vanish or erupt. Training the RNN model using a gradient-based optimization method (for example, the backpropagation algorithm) presents difficulties. RNN

variations have been designed to overcome this issue. Among these, long- and short-term memory, as well as closed recurrent drive, receive considerable attention.

3.6 Long short-term memory

Long-Short Term Memory (LSTM) is a sort of recurrent neural network that uses a unique structure called a memory cell and gate unit to overcome the basic RNN's long-term dependence. LSTM has been demonstrated in applications involving sequential data, including video analysis, traffic forecasting, stock price prediction, and remaining usable life prediction. LSTM can be thought of as an RNN extension with three gates added to keep track of state values over an arbitrary period of time.

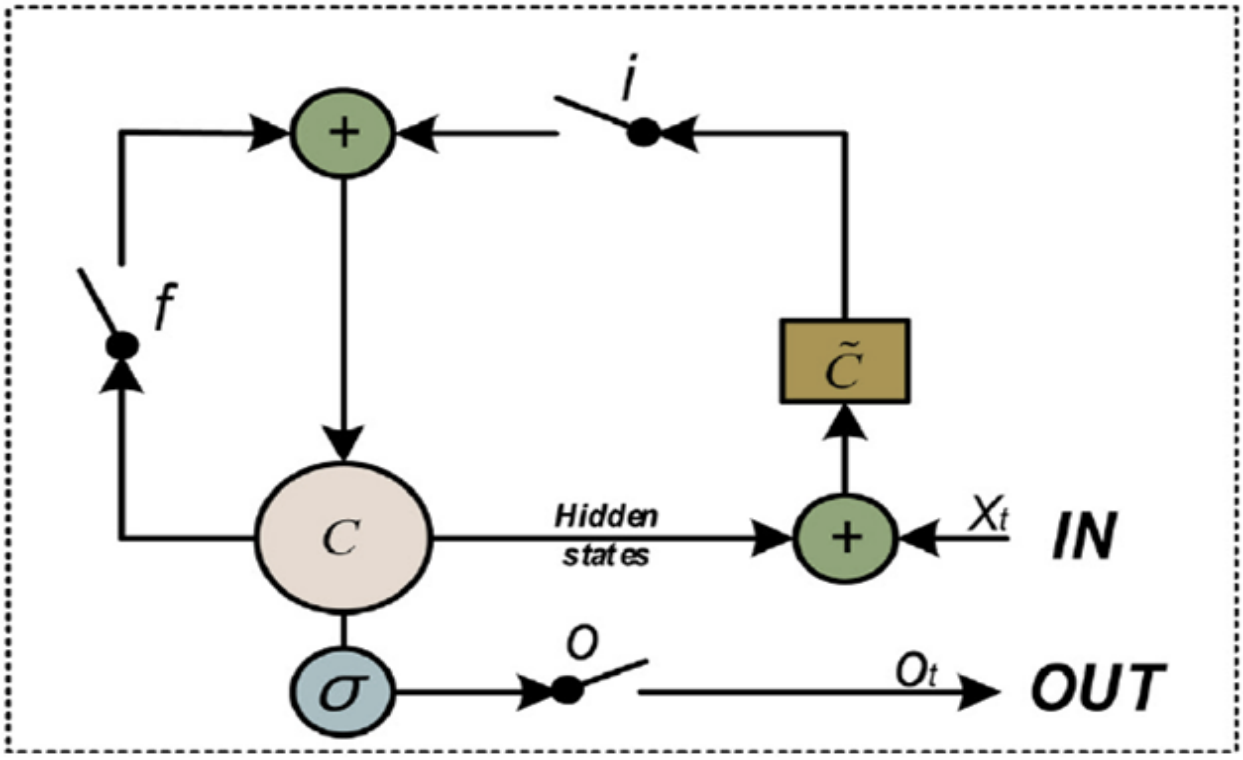


Figure 32: The schematic diagrams of LSTM [79]

The critical aspect of LSTM is controlling the long-term state c. Three control gates are used in this case to implement the LSTM. The first gate controls whether the long-term state c is saved indefinitely; the second gate controls the input of the immediate state to the long-term state c; and the third gate controls whether the long-term state c is the output of the current LSTM.

3.6.1 Forward calculation of LSTM

The gate is essentially a stack of fully linked layers with a vector as its input and a real vector between 0 and 1 as its output. In this example, if W is the gate's weight vector and b is an offset term, the gate can be shown this way:

$$g(\mathrm{a}) = \sigma(W\mathrm{a} + b)$$

The gate multiplies the element's output vector by the vector we desire to control. Because the gate output is a real vector between 0 and 1, when the gate output is 0, multiplying any vector by itself produces a 0 vector, which is equivalent to nothing passing; when the gate output is 1, multiplying any vector by itself produces a 1 vector. Multiplication continues indefinitely, which is equivalent to passing. Due to the fact that x (the sigmoid function) has a value range of (0, 1), the gate is in a half-open/half-closed state.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad (2)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad (5)$$

$$h_t = o_t * tanh(C_t). \quad (6)$$

Two gates in the LSTM regulate the contents of the unit state c. The forget gate controls how much of the unit state C_{t-1} is retained at the current moment C_t from a previous time; the input gate controls how much of the network's input X_t is saved in the unit state C_t at the current instant. An output gate is used to adjust the quantity of unit state output to the LSTM's current output value.

3.7 Gated Recurrent Unit

Numerous modifications have been developed to handle the vanishing-exploding gradient problem that typically happens during the operation of a basic recurrent neural network. One of the most well-known variants is the Long Short Term Memory Network (LSTM). A lesser known but equally strong variant is the Gated Recurrent Unit Network (GRU). Gated recurrent unit (GRU) can be regarded as an updated version of LSTM with a simple structure and has gained popularity in many applications [79].

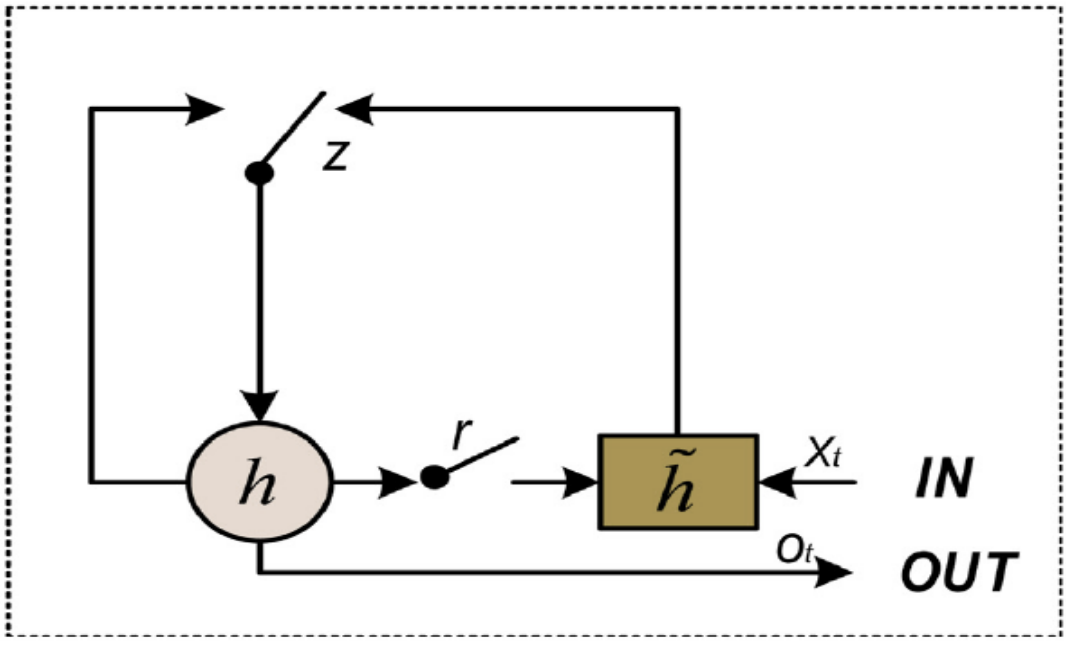


Figure 33: The schematic diagrams of GRU [79]

GRU merges the input gate and the forgotten gate into the update gate while the memory units and hidden units are combined. The gating units in GRU modulate the information flow inside the units without having to separate memory cells.

The mathematic formulation of GRU is described as:

$$z_t = \sigma(W^z x_t + V^z h_{t-1} + b_z) \quad (7)$$

$$r_t = \sigma(W^r x_t + V^r h_{t-1} + b_r) \quad (8)$$

$$\tilde{h}_t = \tanh(W^c x_t + V^c (r_t \otimes h_{t-1})) \quad (9)$$

$$\tilde{h}_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \tilde{h}_t \quad (10)$$

However, because GRU does not have any mechanism to control the degree of its state being exposed, it exposes the whole state each time.

In comparison to LSTM, it has only three gates and does not maintain track of the cell's internal state. The information stored in the internal cell state of an LSTM recurrent unit is incorporated into the hidden state of the gated recurrent unit. This pooled information is sent on to the next gated recurrent unit. A GRU's multiple gates are as follows:

- ✓ **Update Gate** specifies the amount of historical data that must be transferred into the future. It is equivalent to the output gate of an LSTM recurrent unit.
- ✓ **Reset Gate** specifies the extent to which prior information is to be discarded. Similar to an LSTM unit with the Input Gate and the Forget Gate together, this is how it works:
- ✓ **Current Memory Gate** is a feature that is sometimes overlooked in discussions of gated recurrent unit networks. It is a sub-component of the reset gate, much like the input modulation gate is a sub-component of the input gate, and it is used to introduce

non-linearity into the input while maintaining its zero-mean value. Adding it as a subcomponent to the reset gate is another way to keep old data from having an effect on the data that comes in the future.

3.8 Hyperparameter Tuning Approach

Finding the best machine learning parameters necessitates hyper-parameter adjustment. It takes a long time to find the appropriate hyper-parameters, especially when the objective functions are difficult to establish or there are a lot of parameters to adjust. In contrast to traditional machine learning methods, neural networks require more tuning hyper-parameters since they must handle many parameters simultaneously, and the model's accuracy can range from 25% to 90% depending on the fine tuning. Grid search, random forest, Bayesian optimization, and other strategies for tweaking hyper-parameters in deep learning algorithms are among the most effective.

Every method has its own set of benefits and drawbacks. Grid search, for example, has been found to be a good way to change hyper-parameters, even though it has some drawbacks, like testing too many permutations and not being good at changing multiple parameters at the same time [74]. Parameters that cannot be altered during the machine learning training process are known as hyper-parameters. How quickly and precisely a model can be taught depends on the stochastic gradient descent learning rate, batch size, and optimizer.

They can also influence the number of hidden layers and the activation function in a model. HPO dates back to the early 1990s, and as machine learning becomes more prevalent, the technology is increasingly being used for neural networks. HPO is the last step in the model creation process and the first step in neural network training. Given the importance of hyper-

parameters in terms of training accuracy and speed, they must be configured with experience before the training process begins. HPO automatically optimizes the hyperparameters of a machine learning model, removing humans from the machine learning system's loop. It takes a lot of computer power to run the HPO, especially when a lot of hyper-parameters are optimized at the same time.

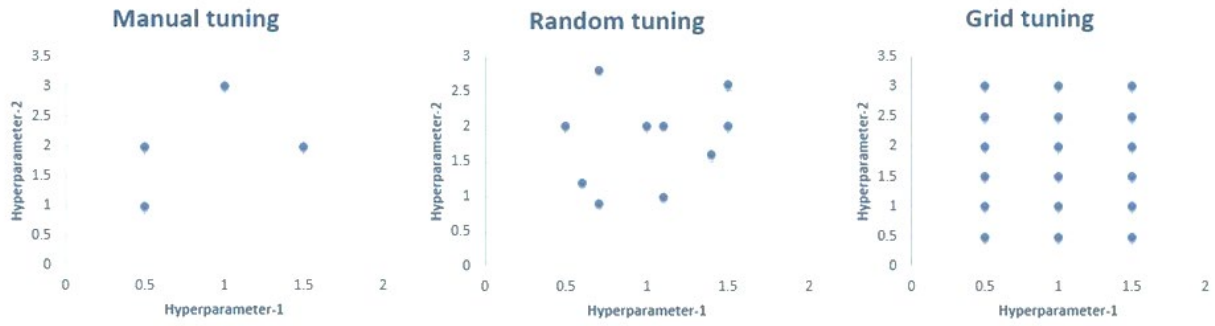


Figure 34: (a) Manual tuning (b) Random tuning (c) Grid tuning approach [From left to Right] [74]

3.8.1 Grid Search

Grid search is a method of exhaustively searching a manually defined portion of the target algorithm's hyperparameter space. Doing a grid search, for example, to perform tests or processes on a variety of conditions is a traditional way of discovering the optimal. If there are three components, for example, a 15 × 15× 15 would imply completing 3375 trials under various situations. When you need to find anything quickly, grid search is the way to go [74]. (1) The model's total number of parameters is small, say M <10. Because the grid is M-dimensional, the number of test solutions is proportional to L^M, where L is the number of test solutions along each of the grid's dimensions. (2) The solution falls inside a certain range of values, which can be used to determine the grid's boundaries. (3) The direct problem d = g (m) can be solved rapidly

enough that LM can be solved from it in a reasonable amount of time. (4) Because the error function E (m) is uniform on the scale of the grid spacing, m, the minimum is not lost due to the coarse grid spacing.

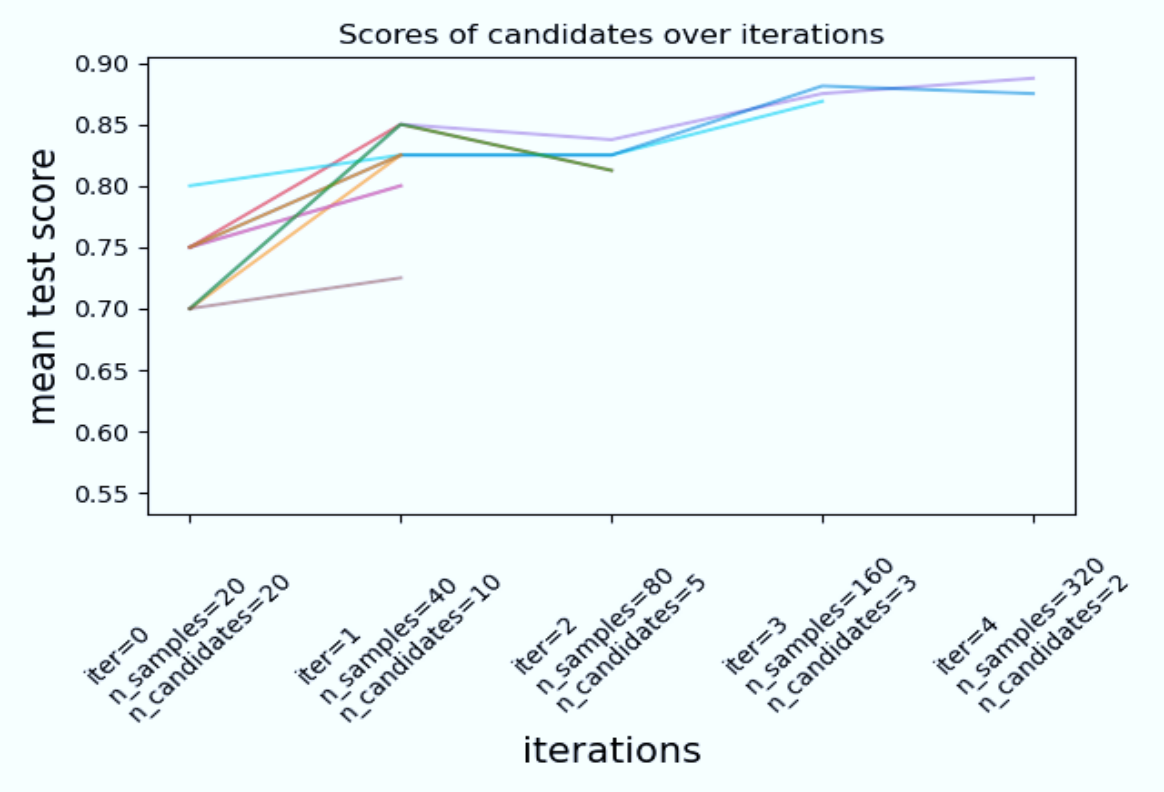


Figure 35: Grid Search Technique [75]

Only a small percentage of candidates "survive" until the final iteration, as shown in the figure above. These are the candidates who have consistently placed among the top scorers in all editions. Each time, more resources, such as the quantity of samples, are allocated to each candidate.

3.8.2 Random Search

A basic enhancement to grid search is random search. It began with a randomized search for hyper-parameters from certain distributions with approximate parameter values. This process

continues until the predetermined budget is depleted, or until the desired level of precision is achieved. These are the most basic stochastic optimization methods, and they're great for challenges like tiny search areas and fast-running simulations. Prior to the probability distribution function, RS finds a value for each hyperparameter. The cost measure is estimated by both the GS and the RS using the obtained hyperparameter sets [75]. RS has been found to be more effective than grid search in many situations, even though it is simple.

Because of various advantages, a random search has been found to produce superior results: First, based on the distribution of the search space, the budget can be determined independently. Random search strategies, on the other hand, can be more effective when the many hyper-parameters aren't spread out evenly across the screen.

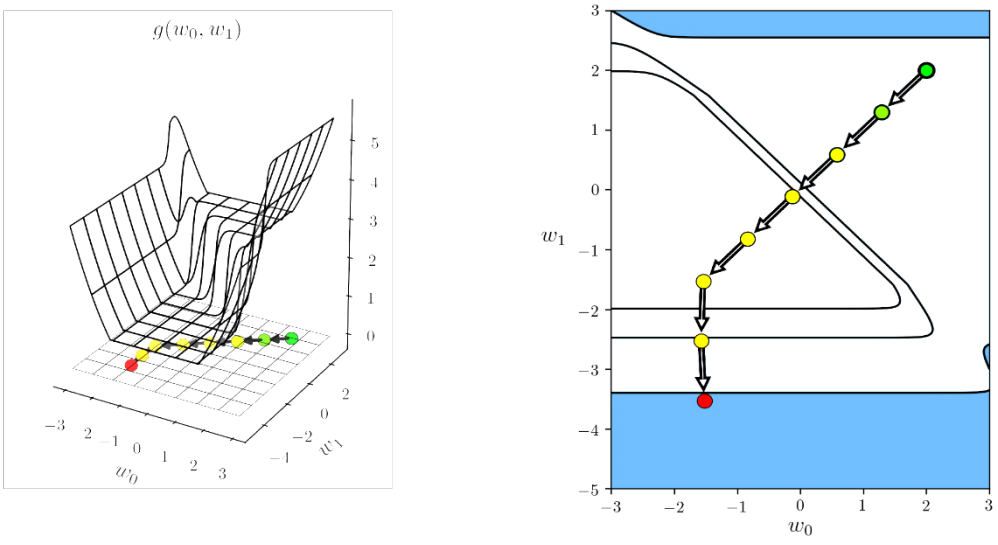


Figure 36: Minimizing a function with many local minima using random search [76]

In this example, Authors [76] shows what one may need to do in order to find the global minimum of a function using (normalized) random local search. They initialize two runs - at w_0=4.5 and w_0=−1.5 . For both runs they use a steplength of α=0.1 fixed for all 10 iterations.

As we can see from the result depending on where we initialize we may end up near a local or global minimum - here resulting from the first and second initialization respectively.

As a result of advancements in IoT and hardware [65,66], DNN has been used in a variety of research fields recently, including forecasting electric load. Ryu et al. [67] developed and compared two DNN-based electric load forecasting models without pre-training using RBM and ReLU. They confirmed that models constructed with ReLU were easier to learn and performed better than models constructed with SNN. Kuo and Huang [68] developed a novel model for forecasting electric load using 1D-CNN and a pooling layer. They came up with a new way to figure out how much electricity will be used in the future by comparing traditional machine learning methods with MLP.

CHAPTER IV

RESULTS AND DISCUSSION

The Electric Reliability Council of Texas (ERCOT) is a self-contained system operator that is responsible for approximately 90% of the state's electric load. ERCOT has made considerable investments in renewable energy, notably wind energy, and continues to be the leading wind generator in the United States. ERCOT's website contains sufficient market and grid information that can be easily viewed and downloaded [80]. If you require data that is not available on the website, you may contact ERCOT via the information request form [81]. ERCOT is willing to assist and reacts promptly.

This thesis was forecasted by using ARIMA, SARIMA, Default LSTM, Custom Stopped LSTM, LSTM tuned, GRU, and GRU tuned algorithms, and then the result was compared to see which model performed the best in practice. The next subsections will detail the outcomes and comparisons.

4.1 ARIMA & SARIMA

Due to the fact that the Texas ERCOT dataset is updated every hour, a shift of 1 will be used in ARIMA and SARIMA modeling to account for this.

Table 3: Load differences & Seasonal differences for the ARIMA & SARIMA Model

Hour Ending	**South**	**Load First Difference**	**Seaonal First Difference**
01/01/2017 01:00	2366.632745	NaN	NaN

01/01/2017 02:00	2332.744630	-33.888115	-33.888115
01/01/2017 03:00	2237.506202	-95.238428	-95.238428
01/01/2017 04:00	2178.102265	-59.403937	-59.403937
01/01/2017 05:00	2133.953870	-44.148395	-44.148395

The absolute value of AIC can be interpreted in a number of ways. (It is possible that various software packages will produce radically different AICs when given the same data for the same model.) The difference in AIC between various models applied to the same data can be interpreted as a difference in AIC between different models applied to different data.

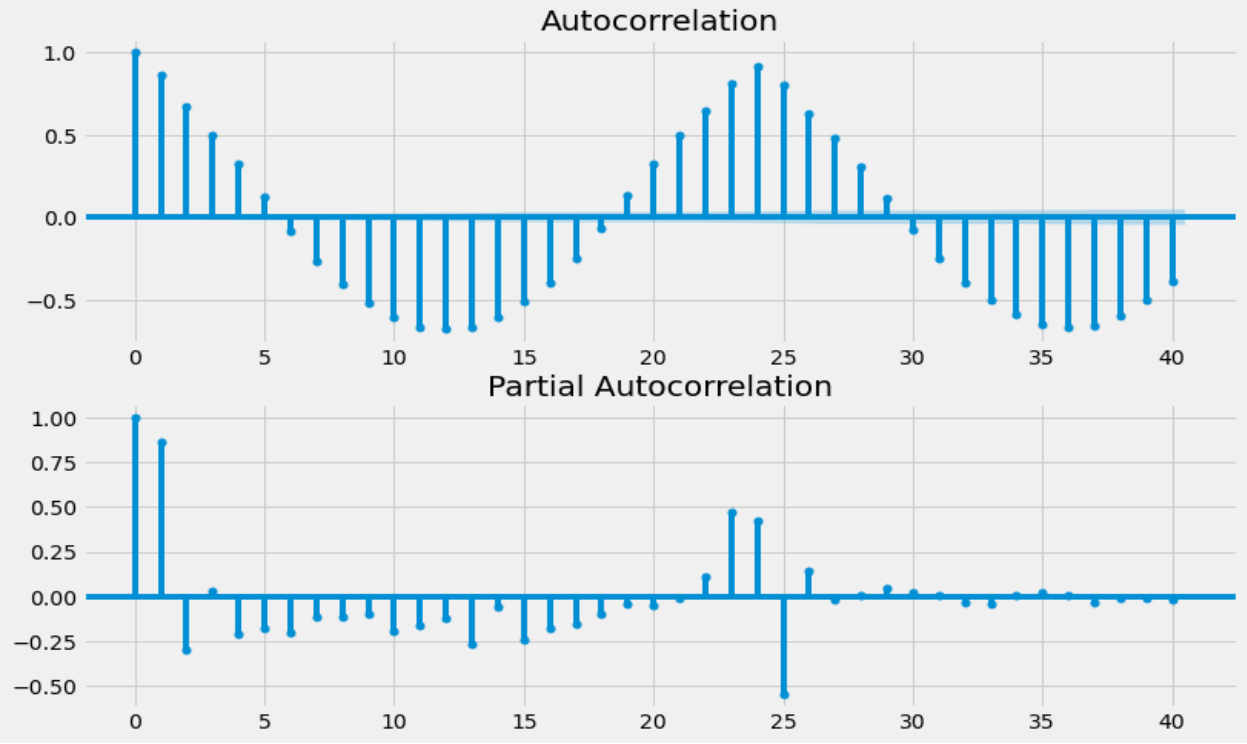


Figure 37: Autocorrelation & Partial Autocorrelation of ARIMA model

In the graphs above, each spike which is above the dashed area considers to be statistically significant. Whereas, Generally, a difference of 2 indicates that both models are essentially equal in quality; a difference of 5 indicates that the model with the lower AIC is slightly better; and a difference of 10 or more indicates that the model with the lower AIC is significantly better.

Table 4: ARIMAX Results:

ARIMAX Results			
Dep. Variable:	SOUTH	**No. Observations:**	40895
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-237061.806
Date:	Sat, 06 Nov 2021	**AIC**	474129.613
Time:	22:00:17	**BIC**	474155.469
Sample:	0	**HQIC**	474137.789
	- 40895		
Covariance Type:	opg		

There are different AIC values for different models. The model with a lower AIC value than the other is thought to be better than the other because it is less complicated and still fits the data better than the other model. A density plot can also be drawn from the residuals.

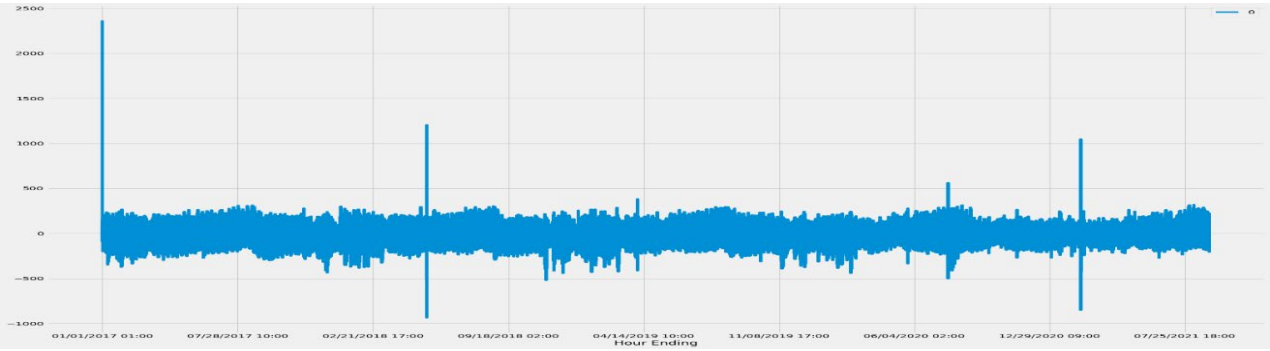


Figure 38: Residual's plot of ARIMA model

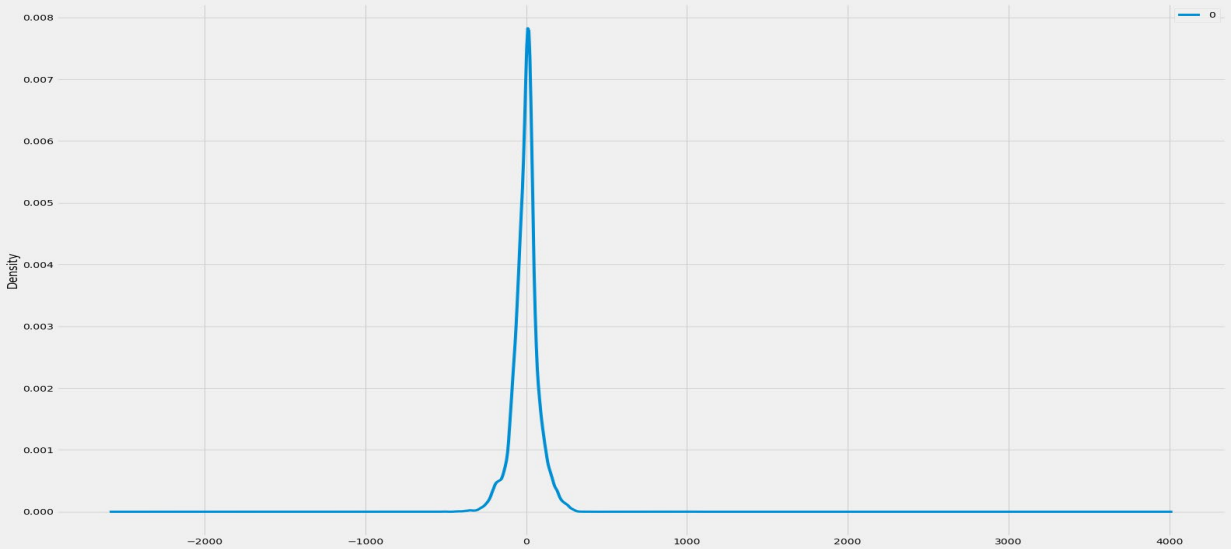


Figure 39: Summary stats of residuals

As observed above, the mean is not exactly zero, which means there is some bias in the data.

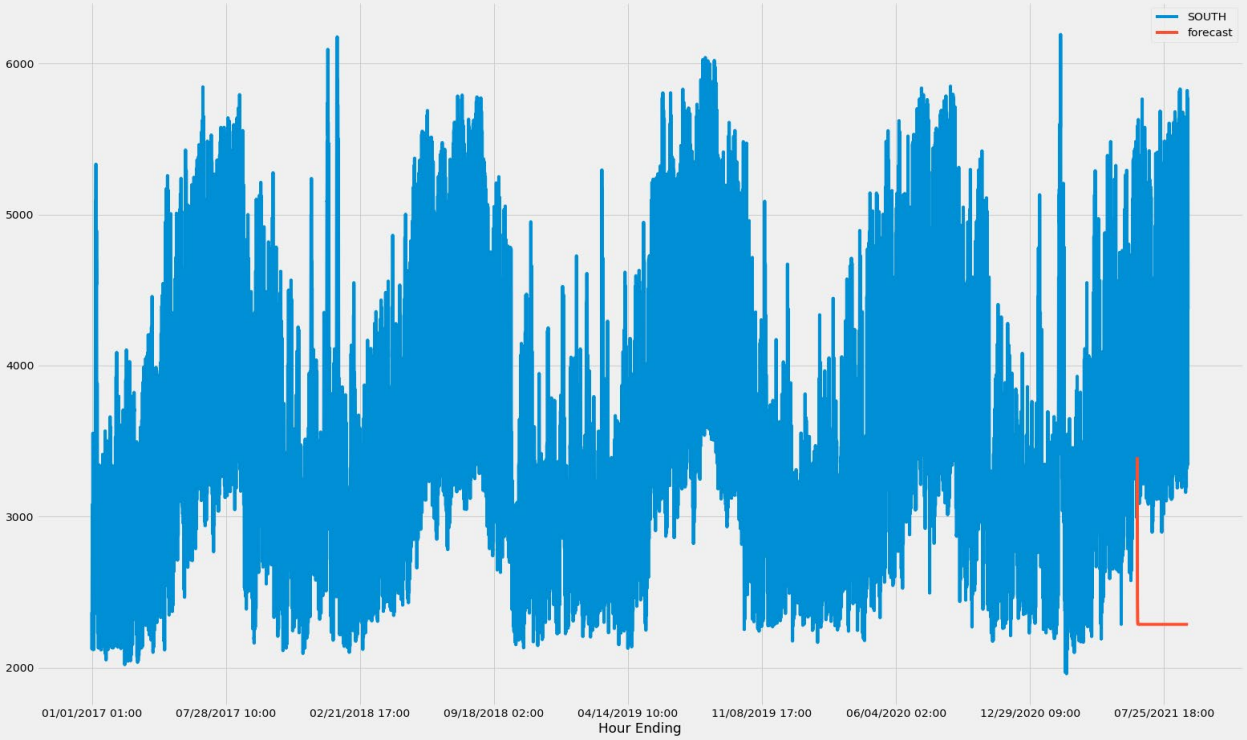


Figure 40: Forecasting next 6 months ERCOT Load through ARIMA model

Table 5: SARIMAX Results:

SARIMAX Results			
Dep. Variable:	SOUTH	**No. Observations:**	40895
Model:	SARIMAX(1, 1, 1)x(1, 1, 1, 24)	**Log Likelihood**	-208352.440
Date:	Sat, 06 Nov 2021	**AIC**	416714.879
Time:	22:03:24	**BIC**	416757.970
Sample:	0	**HQIC**	416728.505
	- 40895		

When looking at the forecast graphs made by ARIMA and SARIMA, it can be seen that SARIMA produces better results effectively over a specific period than ARIMA because of the presence of seasonality in the dataset.

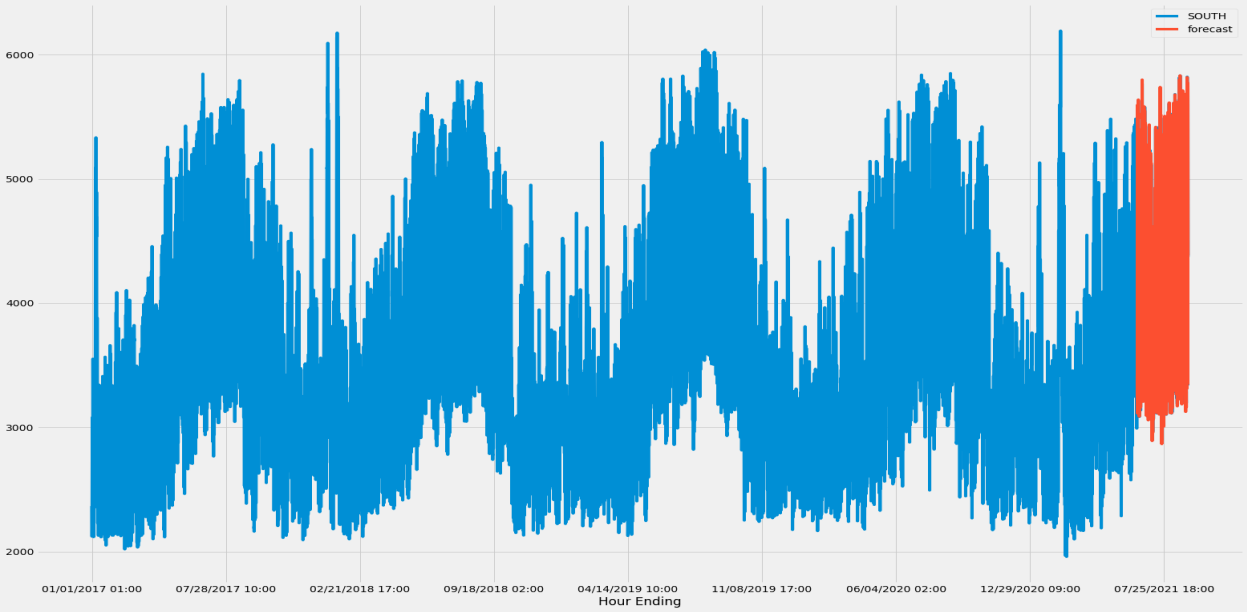


Figure 41: Forecasting next 6 months ERCOT Load through SARIMA model

4.2 Long-Short Term Memory Network

The objective of this section was to assess the suitability of the different Long Short-Term Memory Network models for load forecasting. In this section, the performance metrics of the various models used in this study will be discussed in great detail.

All metrics have values greater than zero and no upper limit, with the exception of R^2, which has an upper limit of 1 and no lower limit. The smaller the value of the R^2 metric, where the relationship is inverse, the better the result. The R^2 value represents the amount of variance that the model can explain. A value of 1 indicates that the model fits the real data perfectly; a value of zero indicates that the dummy prediction always uses the mean value; and a negative value indicates that the prediction is worse than always using the mean. The remaining metrics (MSE, MAE, MAPE, and RMSE) are always positive error metrics, with zero being the best result. The metric MSE was valued most in this work is because it shows how much of a difference there is between the error and the real values.

4.2.1 Naïve Long Short-Term Model

The best results for the Naive LSTM models were achieved when using only recurring networks, while the inclusion of convolutional layers deteriorates the forecast. Naïve LSTM models (Figures 42 and Figure 43) present some of the best results for very short-term forecasts and the worst for average and long-term forecasts. These models have difficulty of converging with a best result can be achieved after 38 epochs.

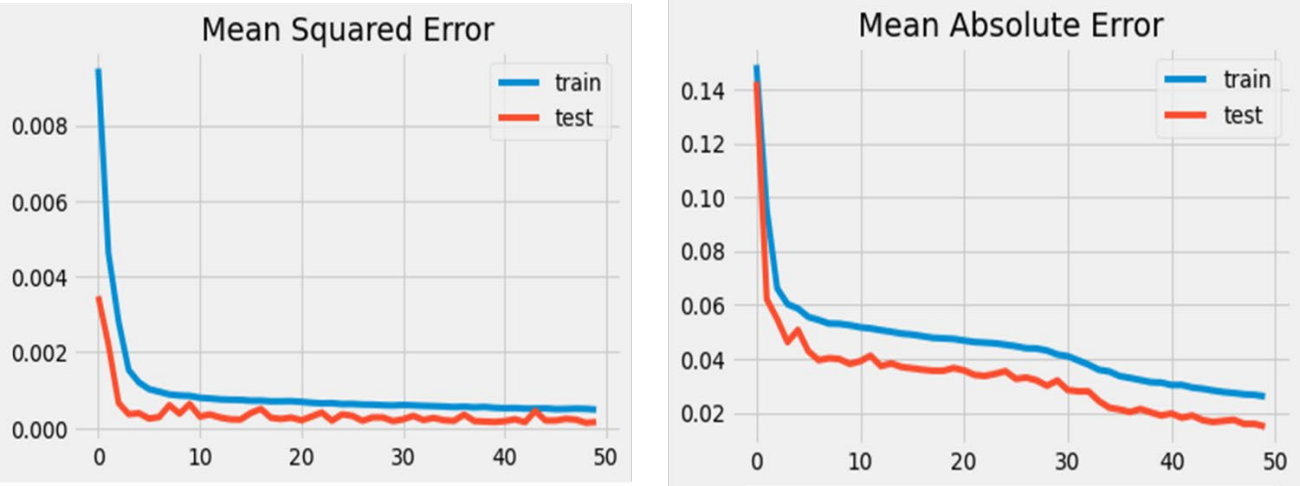


Figure 42: MSE & MAE training history for Naïve LSTM model (Epochs=50, validation split=0.20, batch size=512, verbose=2)

Additionally, it may be worthwhile to examine the evolution of the loss function during training & testing. The evolution of the Naive LSTM method is shown in Figure 43, using the mean absolute error, and R^2 values.

Figure 43: MAPE & R^2 training history for Naïve LSTM model

In general, it is observed that the majority of models converged smoothly after 30–40 epochs.

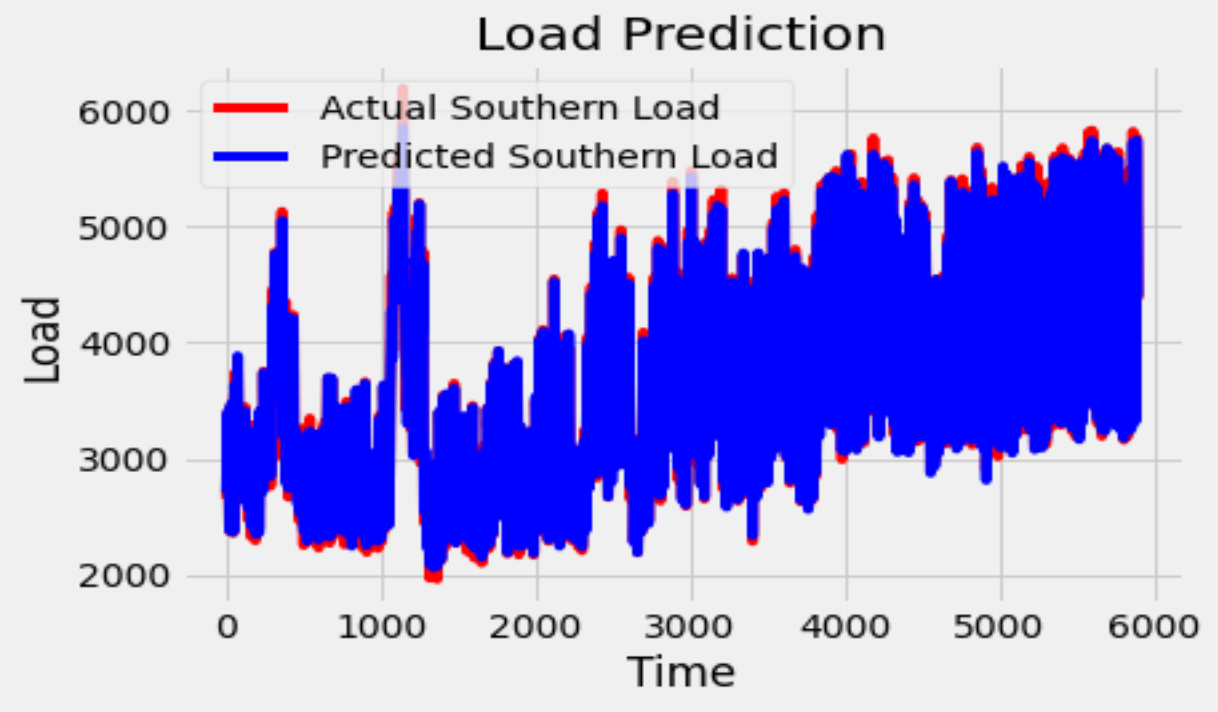


Figure 44: Actual Load vs Predicted Load for Naïve LSTM model

To demonstrate the forecast's accuracy, Figure 44 compares actual load values to forecasted load values as the forecast time horizon is extended. The blue color represents the ground truth or actual values, while the red color represents forecasted values. The diagram depicts time windows of 1000 hours taken at random points throughout the test set. Until the forecast begins to drift, the forecast nearly perfectly replicates the real signal. The drift point varies according to sample size, with some signals being followed almost perfectly all of the time and others drifting apart during initial forecasts. The forecast signal is a smoothed representation of the actual signal in each case. The root mean squared error of Naïve LSTM model was found 90.824.

4.2.2 Default Early Long Short-Term Model

Figure 45 demonstrates forecast accuracy by comparing actual load values to forecasted load values. We can see the huge drift between actual load and forecasted load in the figure because our model has to stop predicting because of the default early stopping features in the model. Furthermore, through this method, it is seen that by including a custom early-stop feature in the model, the overall forecasting results can be improved.

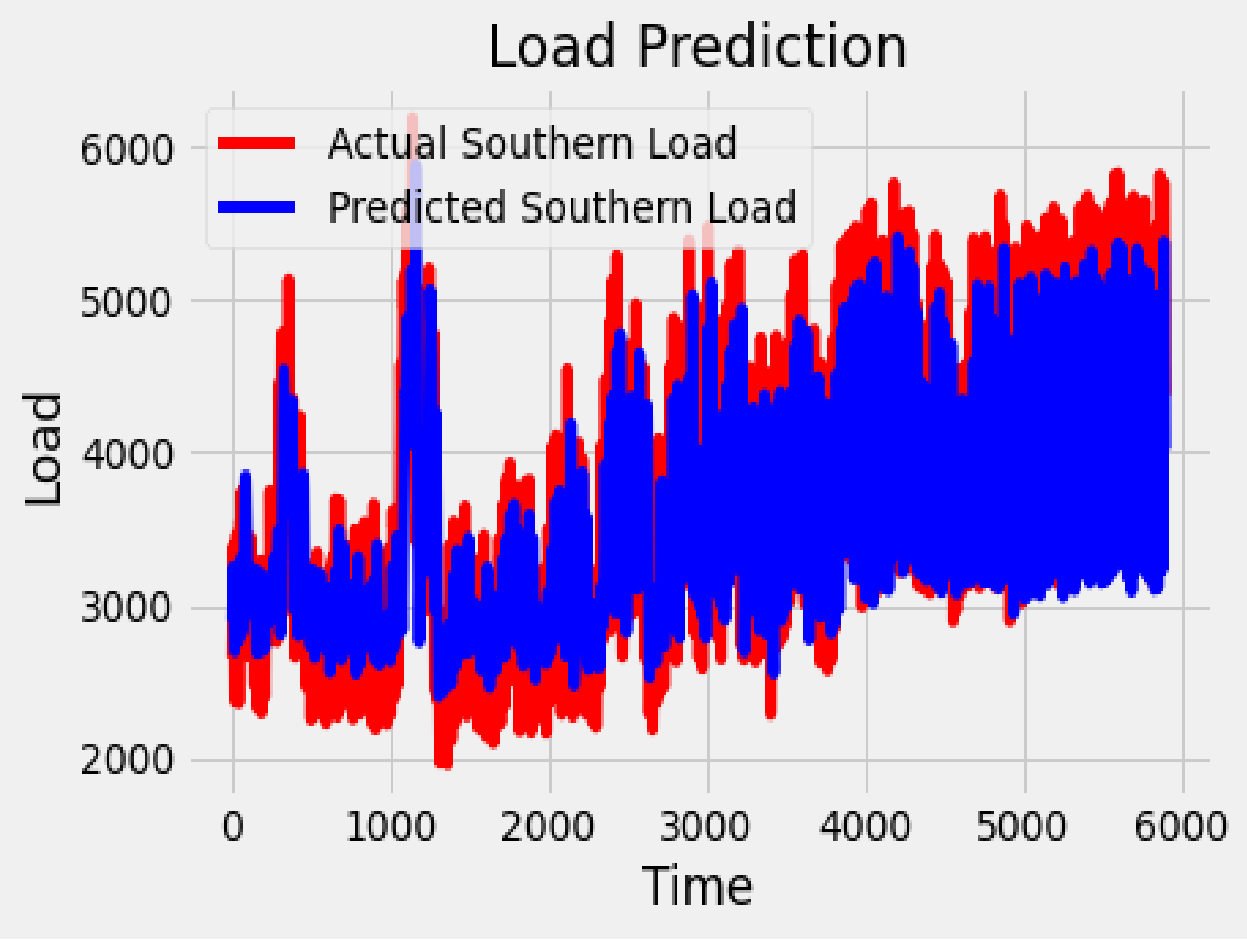


Figure 45: Actual Load vs Predicted Load for Default early stopped LSTM model

The root mean squared error of the LSTM with the default early stopping feature model was found to be 434.161.

4.2.3 Custom Early Stopped Long Short-Term Model

The best results for the LSTM models with stopping features can be achieved by customizing our model to some extent. With the custom early stopping features, our LSTM models (Figures 46 and 47) present some of the best results for very short-term forecasts and for certain long-term forecasts. These models converged, with the best result being achieved after 43 epochs.

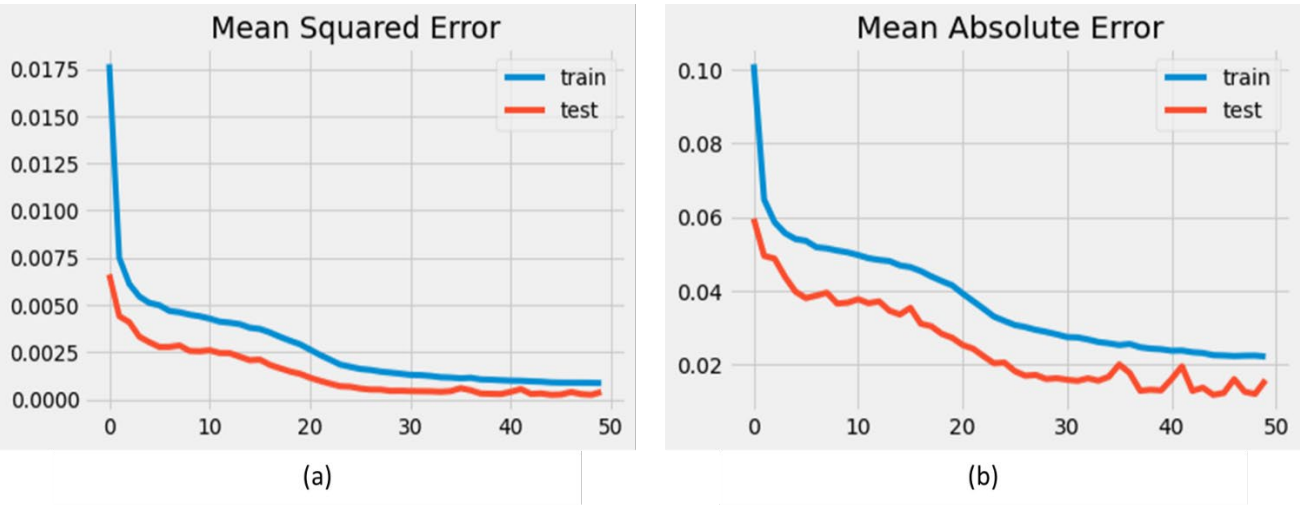


Figure 46: MSE & MAE training history for Custom Early Stopped LSTM

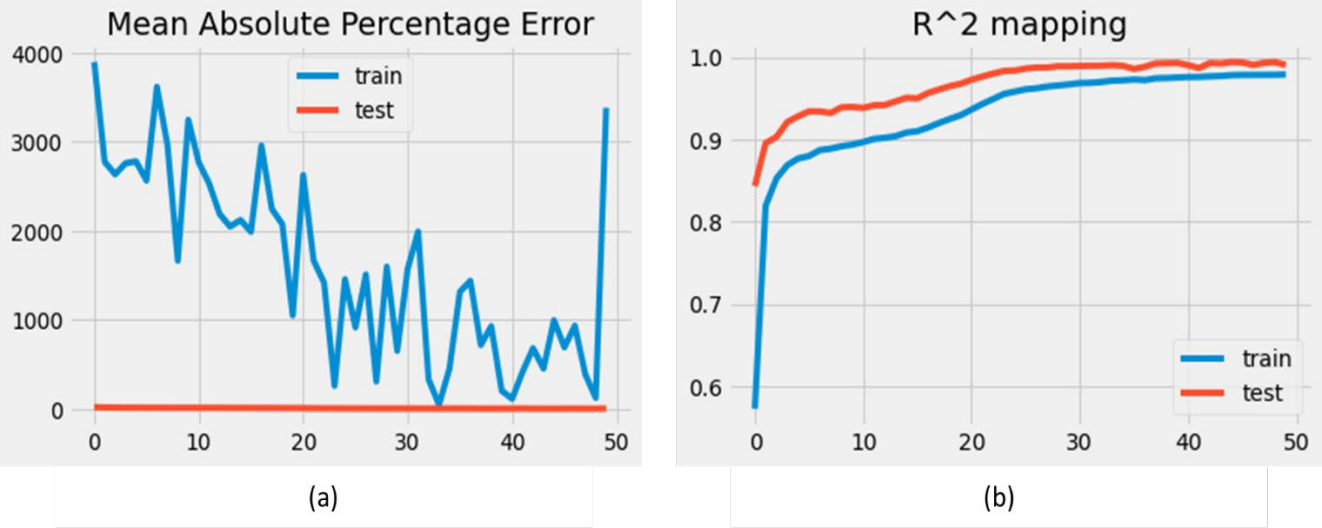


Figure 47: MAPE & R^2 training history for Custom Early stopped LSTM model

For the Custom Early Stopped LSTM model, Figure 48 compares actual load values to forecasted load values. The ground truth or actual values are represented by the blue color in the graphic, whereas forecasted values are represented by the red color. The graphic depicts the same as1000-hour time windows taken at random intervals throughout the test set. The forecast virtually exactly mirrors the real signal until it starts to deviate.

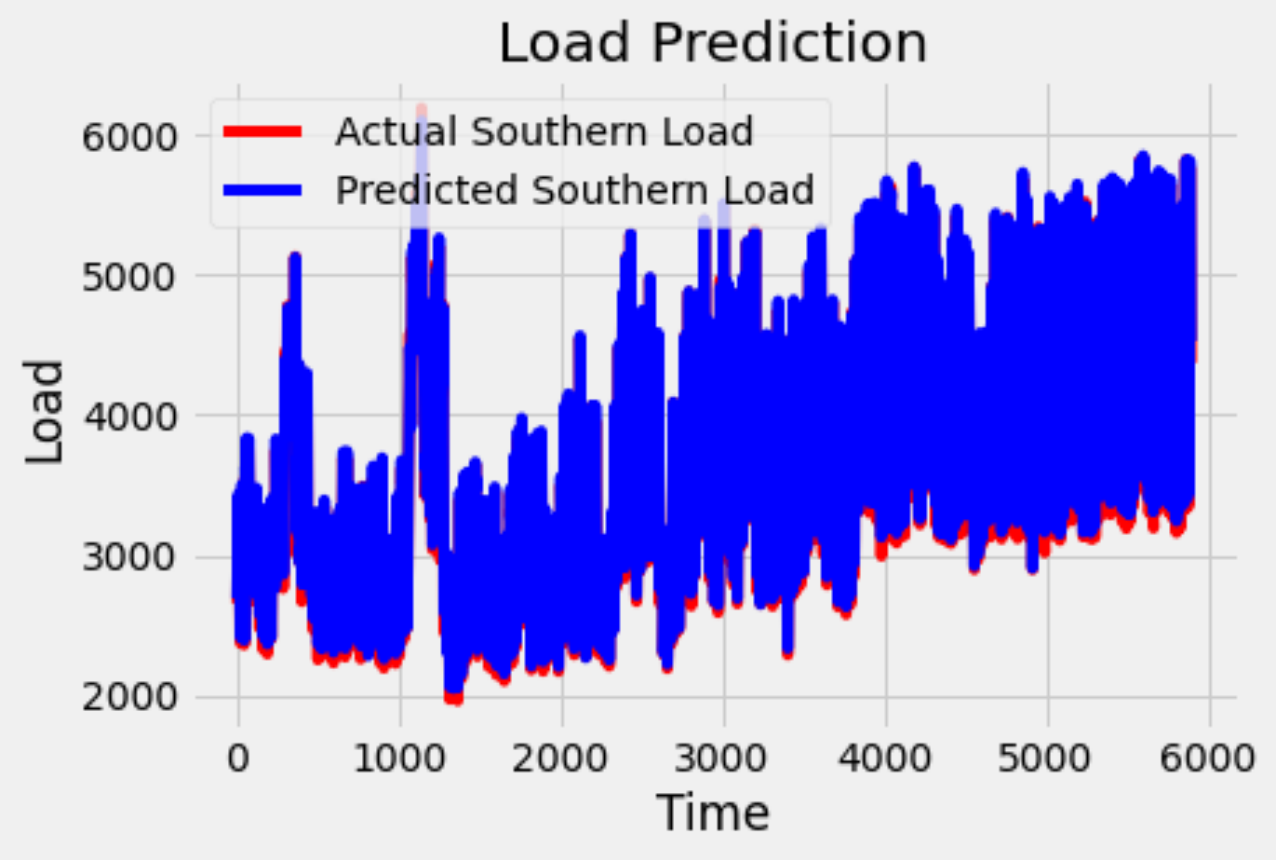


Figure 48: Actual Load vs Predicted Load for Custom early stopped LSTM model

The drift point varies depending on sample size, with some signals being practically exactly tracked all of the time and others slipping apart during early forecasts. In each scenario, the prediction signal is a smoothed version of the actual signal. The root mean squared error of the custom early stopping LSTM model was found to be 83.299.

4.2.4 Tuned Long Short-Term Model

Tuning the LSTM model yields the greatest results out of all of the LSTM models. The models that are depicted in Figures 49 and 50 provide some of the best results for both extremely short-term and long-term forecasting, thanks to the tailored hyperparameter features of the LSTM model.

Figure 49: MSE & MAE training history for Tuned LSTM model

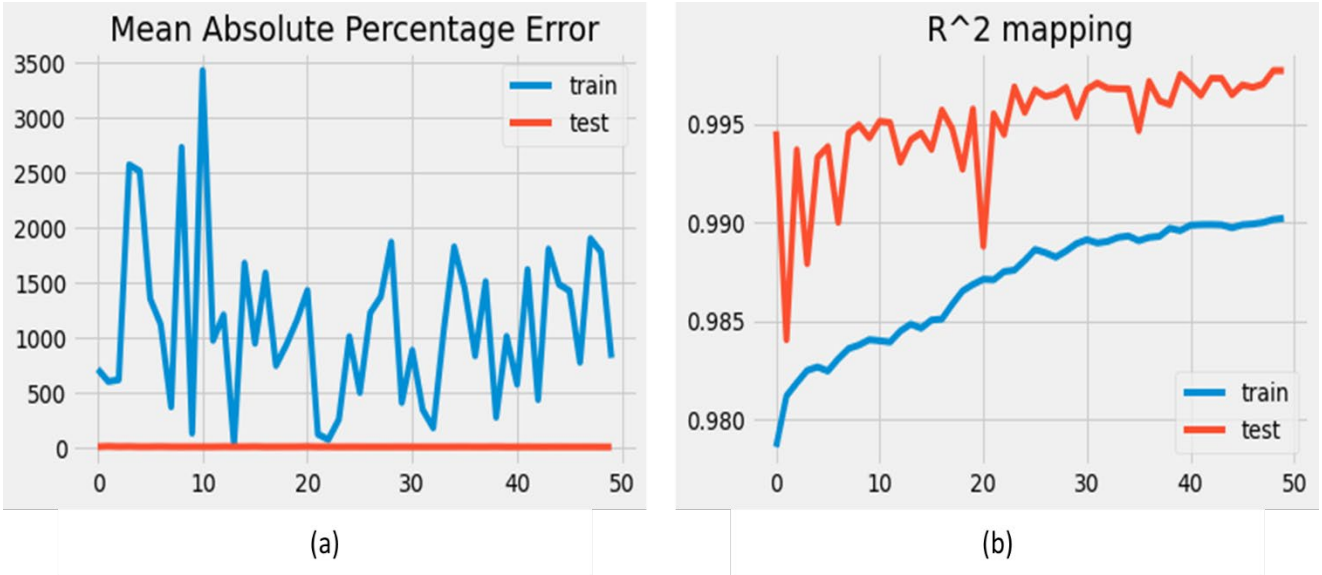


Figure 50: MAPE & R^2 training history for Tuned LSTM model

Figure 51 shows the difference between actual and anticipated load values for the tuned LSTM model. The blue color in the figure represents actual values, while the red color represents anticipated values. As with the other two models, this one use 1000-hour time periods gathered at random locations across the test set. For as long as it takes for the forecast to end, the signal almost closely matches the forecast.

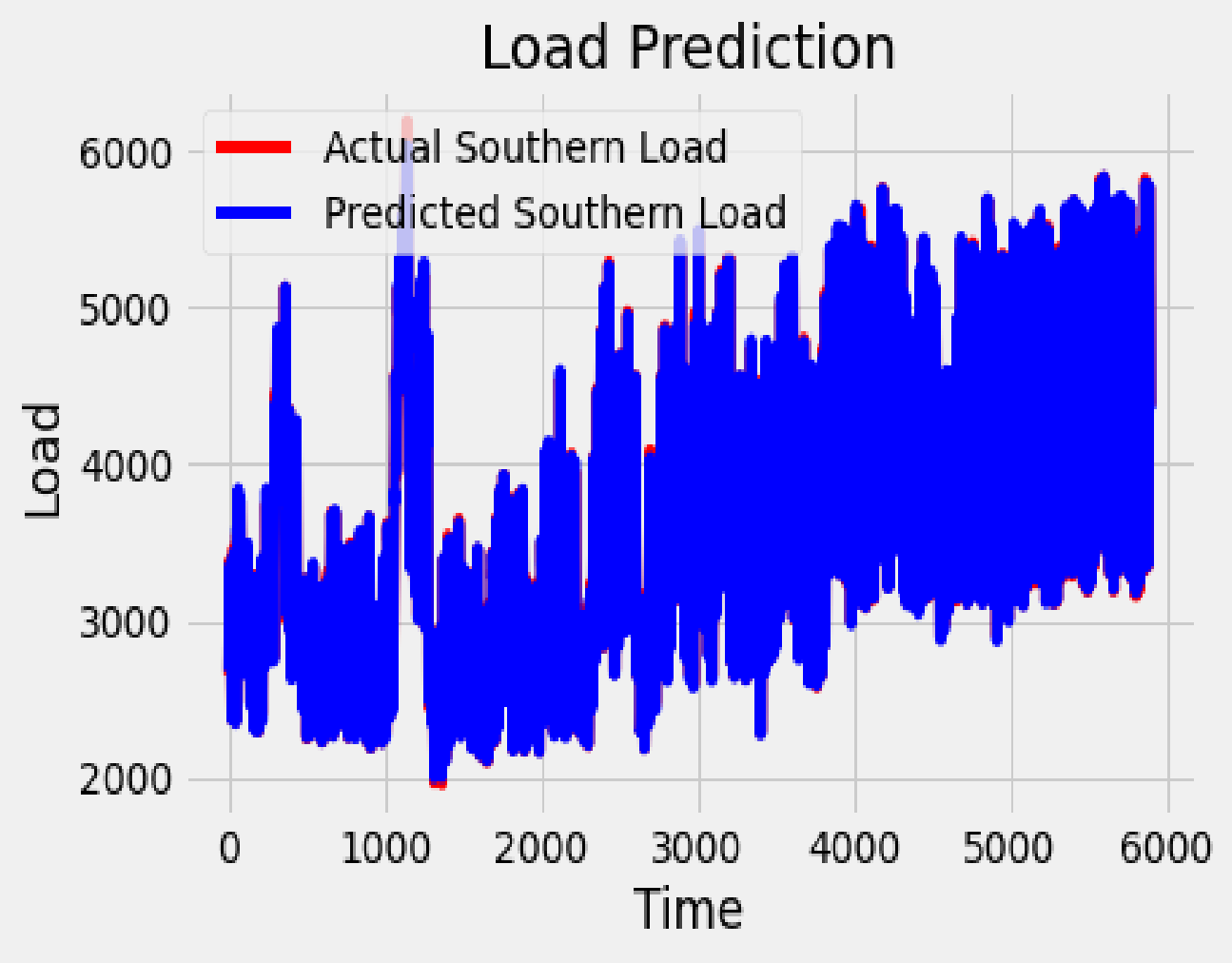


Figure 51: Actual Load vs Predicted Load for Tuned LSTM model

The root mean squared error of the tuned LSTM model was found to be 52.869, which is lower than the root mean squared error of all preceding models.

4.3 Gated Recurrent Unit

4.3.1 Naive Gated Recurrent Unit

The hyperparameter tuning strategy can be used to produce the best results for gated recurrent unit models. Except for the tuned LSTM model, GRU models (Figures 52 and 53) can outperform the other two LSTM models (Naive and Custom Early Stopped) using default GRU approaches.

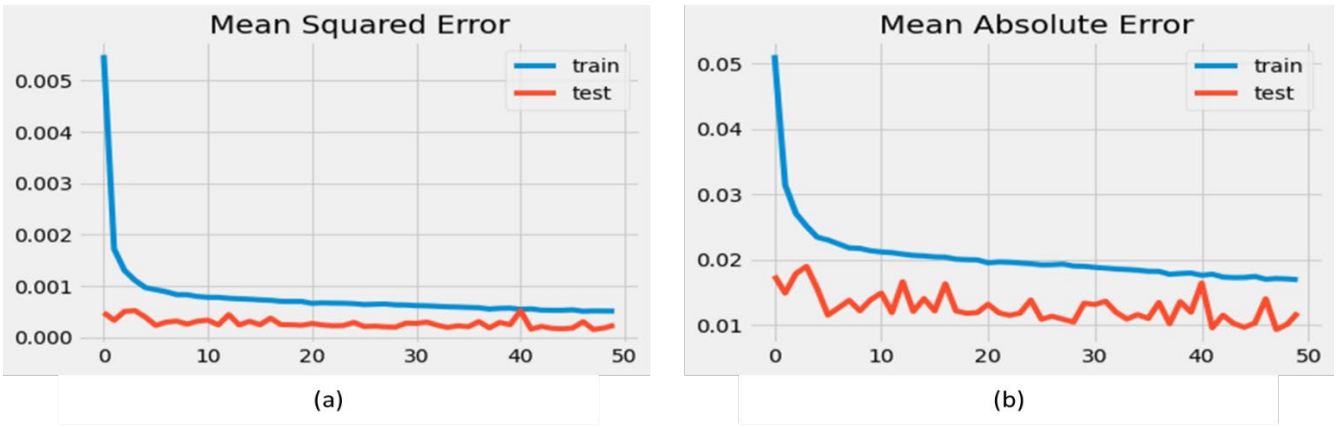


Figure 52: MSE & MAE training history for Naive GRU model

Figure 53 MAPE & R^2 training history for Naive GRU model

This model can also be used to predict some of the best results for both short-term and long-term forecasting. With the default GRU technique, the best result was reached after 40 epochs.

The disparity between the actual and forecasted load numbers for the Naive GRU model is depicted in Figure 54. This model does better than the Naive LSTM and the custom stopped LSTM models, but it doesn't do as well as the tuned LSTM model shown in the previous illustration.

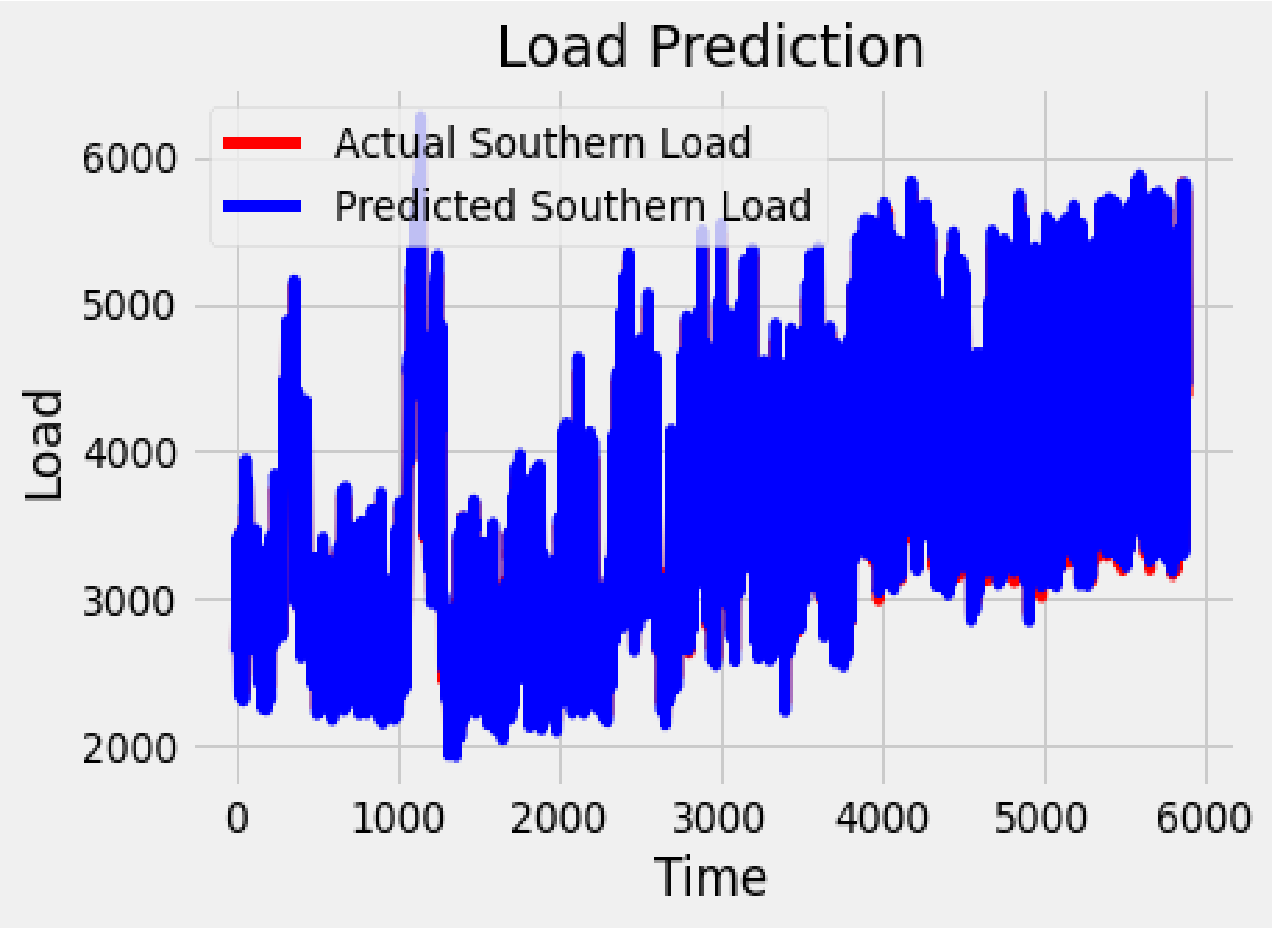


Figure 54: Actual Load vs Predicted Load for Naïve GRU

The root mean squared error of Naïve GRU model is 67.768.

4.3.2 Tuned Gated Recurrent Unit

Tuning the GRU model produces the best results of any model. The GRU model's outcomes shown in Figures 55 and 50 provide some of the best results for forecasting extremely short and extremely long time periods, owing to the GRU model's adjusted hyperparameter features.

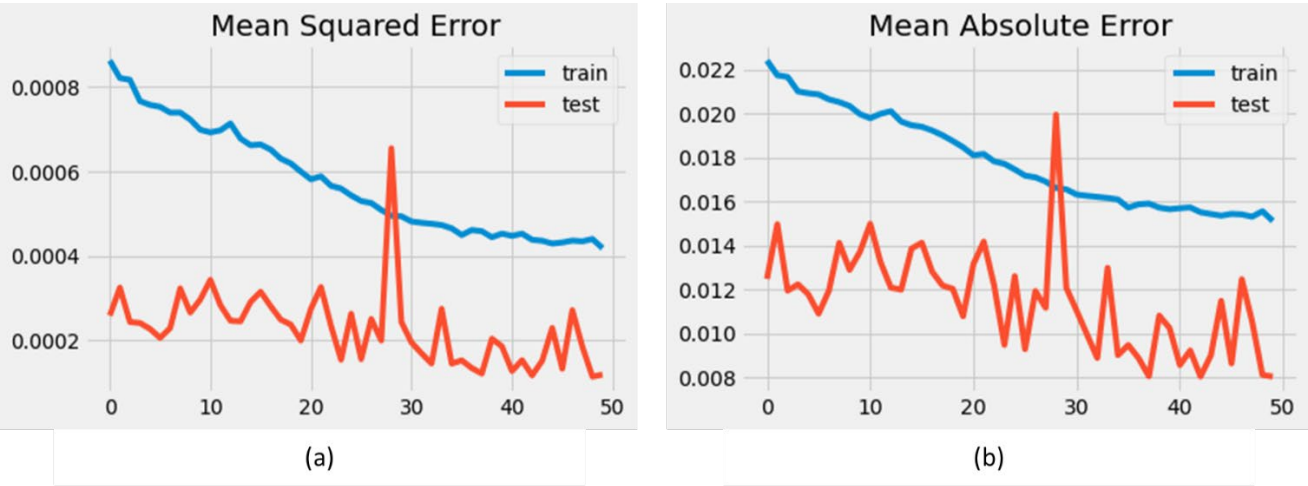


Figure 55: MSE & MAE training history for Naive GRU model Tuned GRU model

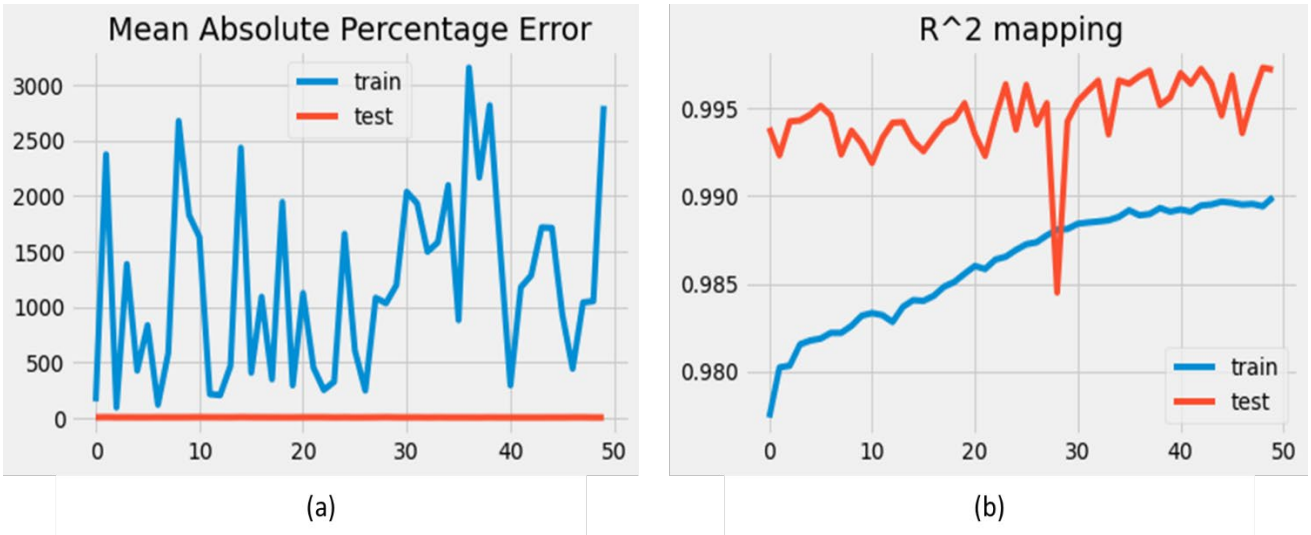


Figure 56: MAPE & R^2 training history for Naive GRU model Tuned GRU model

The discrepancy between actual and projected load values for the tuned GRU model is shown in Figure 57. Actual values are represented by the blue color in the graph, whereas forecasted values are represented by the red color.

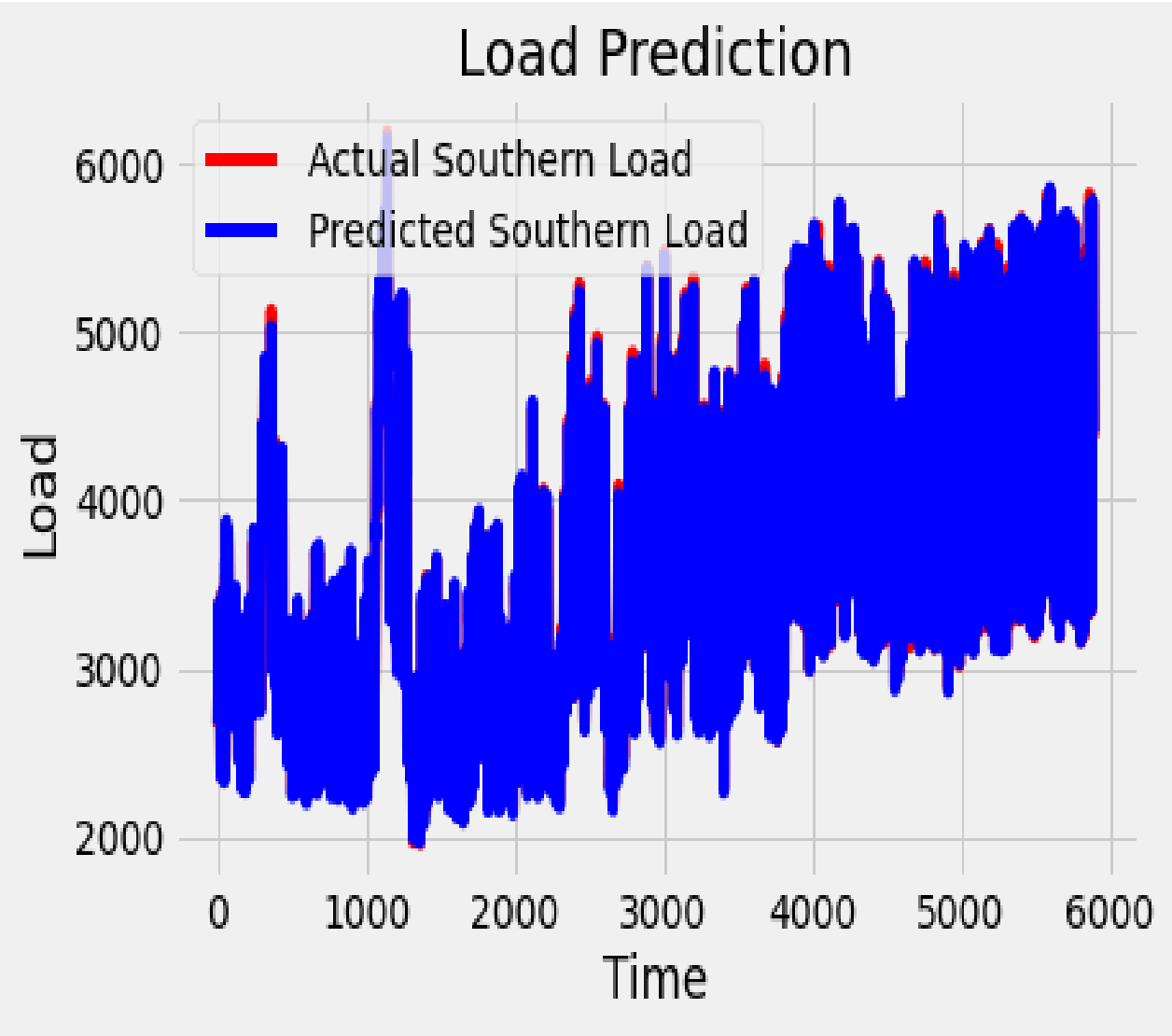


Figure 57: Actual Load vs Projected Load for Tuned GRU

When comparing the tuned GRU model to other models, the tuned GRU model gives the best result as it can be seen that there are fewer significant differences between the colors of the

actual load and the colors of the load in this model. The root mean squared error for the tuned GRU model is found to be 49.758.

4.4 Root Mean Squared Error Comparison

An illustration of a Root Mean Squared error bar chart generated by Minitab is displayed in Figure 58, and it demonstrates that the GRU model has the lowest RMSE of the models, at 49.76, while the Nave LSTM model has the greatest RMSE, at a substantially higher value of 90.82.

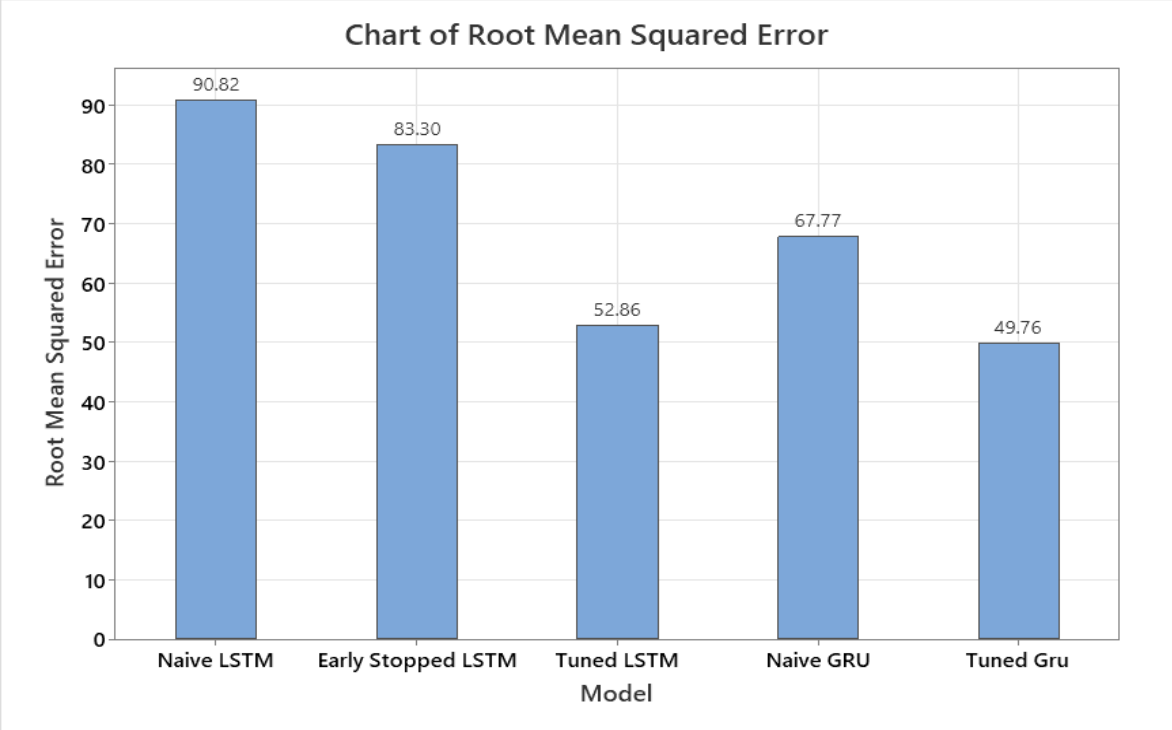


Figure 58: Chart comparison of RMSE for various DL model

4.5 MSE comparison

The MSE comparison graph in Figure 59 shows the MSE for each epoch for the two best models: (1) tuned GRU and (2) tuned LSTM. As time spent on training the model increased, the

MSE value went down even more for both models, but tuned GRU outperformed tuned LSTM most of the time.

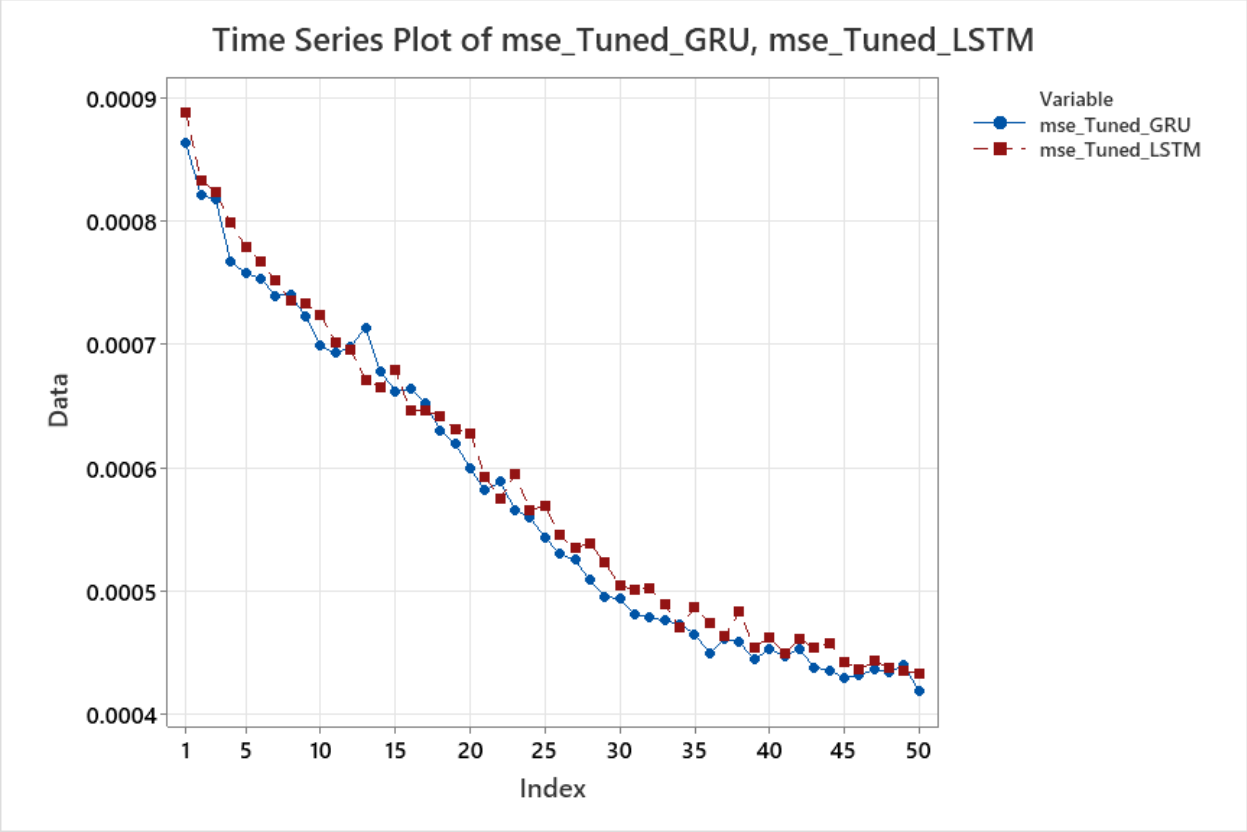


Figure 59: MSE comparison of Tuned GRU vs Tuned LSTM

4.6 Radar Chart for Model Comparison

Radar chart is a visual representation of numerous quantitative data points. Because radar charts make it easy to observe which variables in a dataset are performing well or poorly, they are useful for determining how well a group performed on a test.

Each variable has an axis that begins at the center. All axes are radially oriented, with equal spacing between them while preserving the same scale. Grid lines that link the axes are

frequently used as guides. Each variable's value is shown on its own axis, and the values of all variables in a dataset are linked to form a polygon.

Figure 60 depicts a radar chart using six models and the six error values generated for each model using our models. To find out the comparison, six error functions were utilized: the loss function MSE, the mean absolute error, the mean absolute percentage error, the validation loss, the RMS values, and the validation R^2 values.

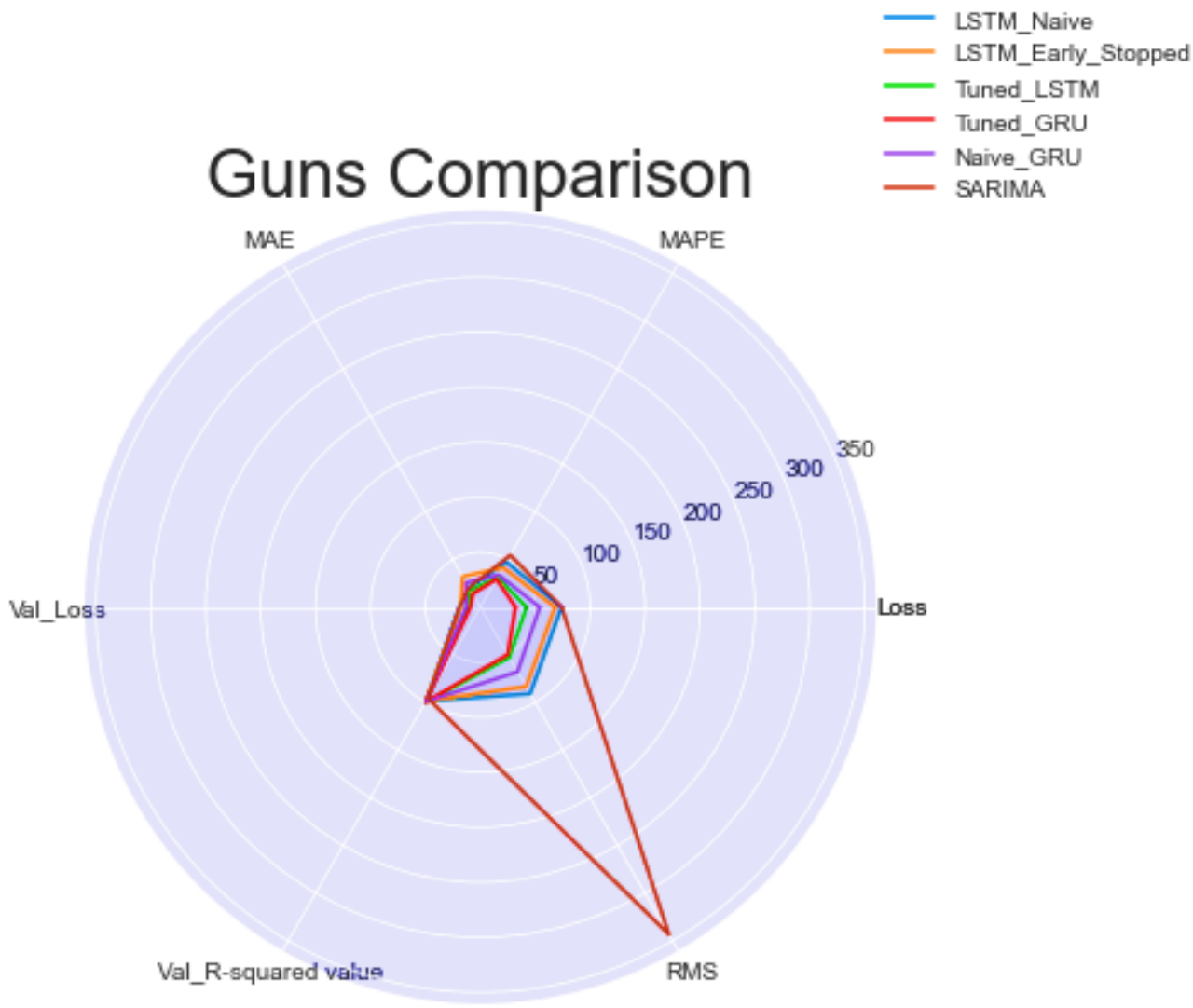


Figure 60: Radar chart for error comparison

Whichever area is covered, the model with the least area will be the best model, as, except for the R^2 values, our loss function indicates that the lower the number, the better the model performs.

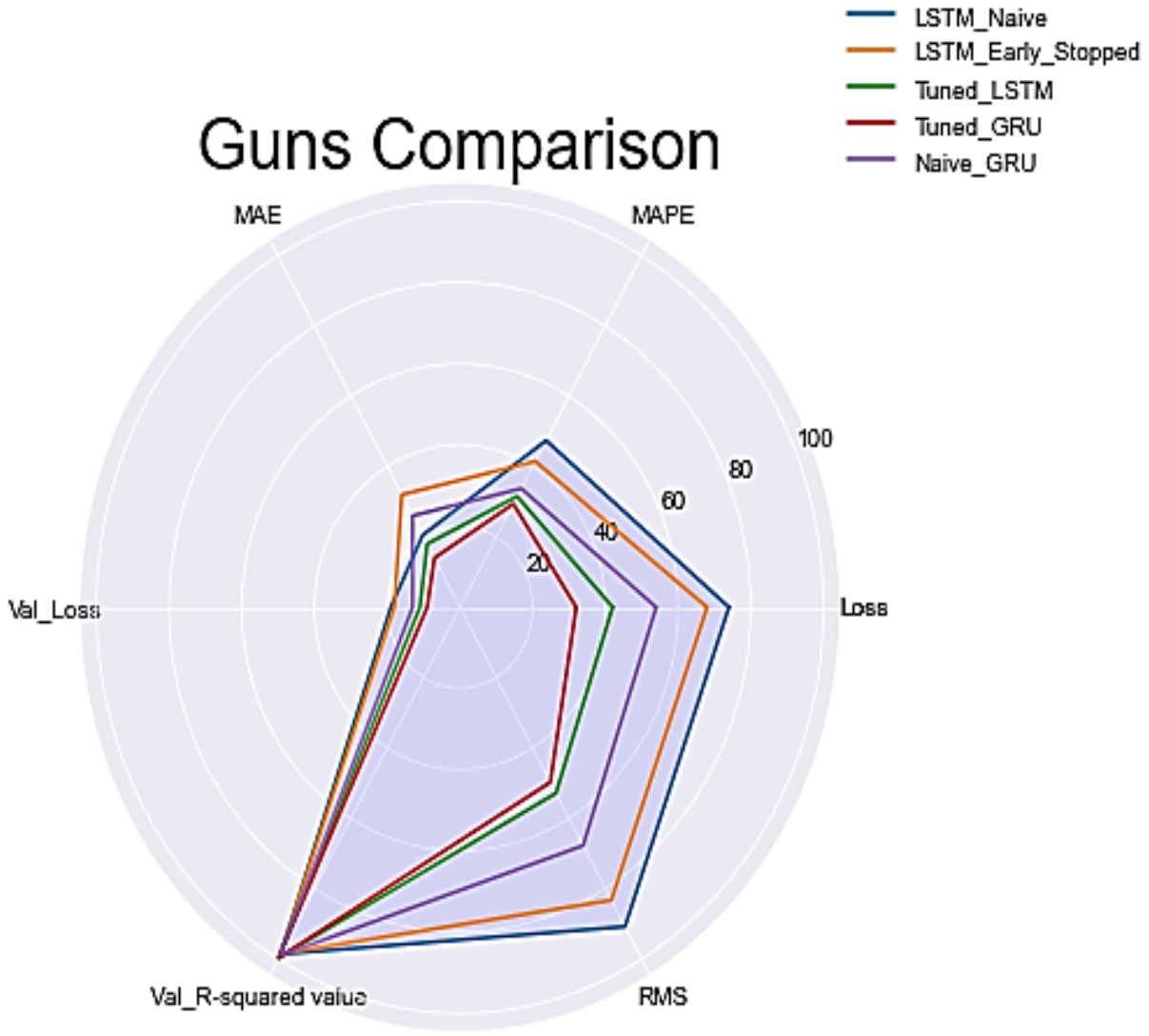


Figure 61: Final Radar chart for error comparison

Figure 61 is an extended version of previous radar chart to aid in comprehension of the area. As shown in Figure 60, the SARIMA model covered the biggest area, while tuned GRU covered the least area, followed by LSTM tuned and other models.

4.7 Pygal Radar Chart

Pygal is a Python package that is mostly used to create graphs and charts in Scalar Vector Graphics. SVG is a vector-based visual formatted in XML that is editable in any text editor. Pygal can generate simple-to-understand graphs with a few lines of code.

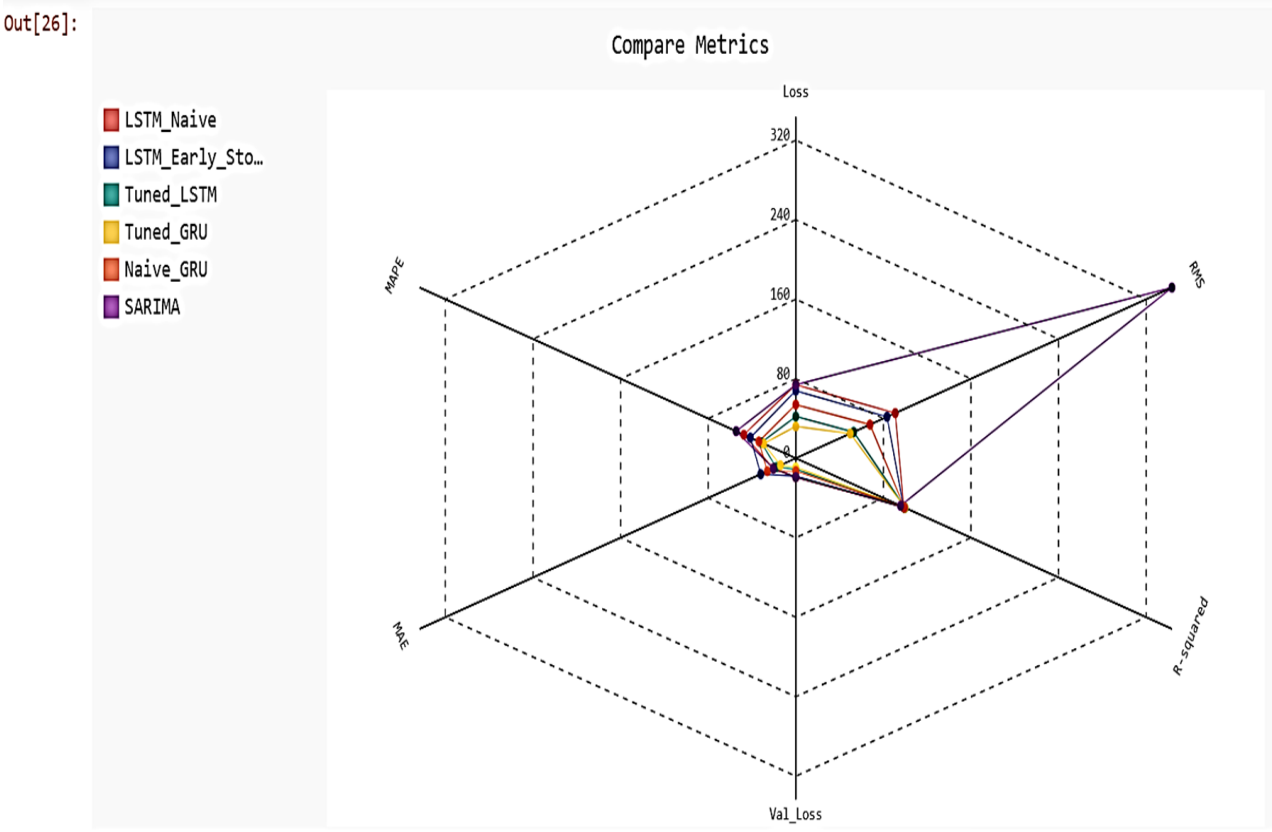


Figure 62: Pygal Radar chart for error comparison

Figure 62 is a Pygal Radar chart that shows how many areas each model error covers. The tuned GRU, which is shown in orange, covers the least area and is our best model, while the tuned LSTM model, which covers slightly less area, is found to be the second best model.

CHAPTER V

CONCLUSIONS & FUTURE WORKS

This work is primarily performed through the analysis of five deep learning models for forecasting hourly ERCOT load in Texas: Naive LSTM, Early Stopped LSTM, Tuned LSTM, Naive GRU, and Tuned GRU. It turns out that a GRU model with the right hyperparameters is better than other deep learning-based neural networks, like the well-known LSTM network, at predicting the next hour's load.

Contrary to recent academic findings that LSTMs outperform other deep learning techniques for forecasting short-term loads, with this work, it is safe to conclude that tuned GRU models outperform all other DL techniques. As a result, this work's final recommendation will be to employ tuned GRU for hourly, short-term, and long-term load forecasting.

Researchers could merge multiple distinct deep learning model methodologies, such as convolutional neural networks and recurrent neural networks, to develop a superior model as a future avenue of research. Integrating weather data into the problems in order to develop a more sustainable model will also be a future objective of this work, as this work was unable to incorporate weather data into the model during this study due to a lack of reliable, publicly available hourly weather data for Southern Texas. Another future study will be examining a more effective strategy for modifying the problem's hyperparameters. For this work, random search tuning techniques were employed to tune hyperparameters.

REFERENCES

[1] Panigrahi, Sibarama; Behera, H.S. (2020). A study on leading machine learning techniques for high order fuzzy time series forecasting. Engineering Applications of Artificial Intelligence, 87(), 103245–. doi:10.1016/j.engappai.2019.103245

[2] S.-J., Huang and K.-R. Shih, “Short-term load forecasting via ARMA model identification including non-Guassian process considerations,” IEEE Transactions on Power Systems, vol. 18, no. 2, pp. 673–679, May 2003

[3] P. R. Ji, D. Xiong, P. Wang, and J. Chen, “A study on exponential smoothing model for load forecasting,” in Proceedings of 2012 Power and Energy Engineering Conference (APPEEC), Shanghai, China, 2012, pp. 1– 4.

[4] Dr. Hassan Kuhba, Hassan A. Hassan Al-Tamemi, "Power System Short-Term Load Forecasting Using Artificial Neural Networks", International Journal of Engineering 47 Development and Research (IJEDR), ISSN:2321-9939, Volume.4, Issue 2, pp.78-87, April 2016

[5] Samuel, Arthur (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. doi:10.1147

[6] R.-H. Liang, C.-C. Cheng, "Combined regression-fuzzy approach for short-term load forecasting", Proc. Inst. Elect. Eng.-Gen. Transm. Dist., vol. 147, no. 4, pp. 261-266, July 2000.

[7] Yan Hong Chen, Wei-Chiang Hong, Wen Shen and Ning Ning Huan, “Electric Load Forecasting Based on a Least Squares Support Vector Machine with Fuzzy Time Series and Global Harmony Search Algorithm”, Published: 26 January 2016

[8] Mohamed Abuella , Badrul Chowdhury, “Solar Power Forecasting Using Support Vector Regression”, American Society for Engineering Management International Annual Conference, 2016.

[9] B. E. Turkay, D. Demren, "Electrical load forecasting using support vector machines", Proceedings of 7th International Conference on Electrical and Electronics Engineering (ELECO), pp. 49-53, 2011.

[10] A. Jain, B. Satish, "Clustering based short term load forecasting using support vector machines", Proc. IEEE Bucharest PowerTech, pp. 1-8, 2009.

[11] P. Zhang, X. Wu, X. Wang, S. Bi, "Short-term load forecasting based on big data technologies", CSEE J. Power Energy Syst., vol. 1, no. 3, pp. 59-67, Sep. 2015.

[12] Steven Mill, “Electric load forecasting: advantages and challenges”

[13] NIU Dong-xiao, GU Zhi-hong, XING Mian, WANG Hui-qing (School of Business Administration, North China Electric Power University, Baoding 071003,Hebei Province,China);Study on Forecasting Approach to Short-term Load of SVM Based on Data Mining[J];Proceedings of the CSEE;2006-18

[14] O. A. Alsayegh, "Short-term load forecasting using seasonal artificial neural networks", International Journal of Power and Energy Systems, vol. 23, no. 3, pp. 137- 142, 2003.

[15] Ma, Zhitong; Ye, Cantao; Li, Huashan; Ma, Weibin (2018). Applying support vector machines to predict building energy consumption in China. Energy Procedia, 152(), 780–786. doi:10.1016/j.egypro.2018.09.245

[16] Vinagre, E., Pinto, T., Ramos, S., Vale, Z., & Corchado, J. M. (2016). Electrical energy consumption forecast using support vector machines. 2016 27th International Workshop on Database and Expert Systems Applications (DEXA). https://doi.org/10.1109/dexa.2016.046

[17] Gandhi, R. (2018, July 5). Support Vector Machine - introduction to machine learning algorithms. Medium. Retrieved October 12, 2021, from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[18] Subodh Paudel, Phuong H. Nguyen, Wil L. Kling, Mohamed Elmitri, Bruno Lacarrière, et al.. Support Vector Machine in Prediction of Building Energy Demand Using Pseudo Dynamic Approach. Proceedings of ECOS 2015-The 28th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems, Jun 2015, Pau, France. ffhal-01178147f

[19] Nichiforov, Cristina; Stamatescu, Iulia; Fagarasan, Ioana; Stamatescu, Grigore (2017). [IEEE 2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE) - Galati (2017.10.20-2017.10.22)] 2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE) - Energy consumption forecasting using ARIMA and neural network models. , (), 1–4. doi:10.1109/ISEEE.2017.8170657

[21] Schaffer, A.L., Dobbins, T.A. & Pearson, SA. Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: a guide for evaluating large-scale health interventions. *BMC Med Res Methodol* **21,** 58 (2021). https://doi.org/10.1186/s12874-021-01235-8

[22] DeepAI. (2019, May 17). Multilayer Perceptron. DeepAI. Retrieved October 14, 2021, from https://deepai.org/machine-learning-glossary-and-terms/multilayer-perceptron.

[23] Pełka, P., & Dudek, G. (2019). Pattern-based forecasting monthly electricity demand using Multilayer Perceptron. Artificial Intelligence and Soft Computing, 663–672. https://doi.org/10.1007/978-3-030-20912-4_60

[24] Larry Hardesty | MIT News Office. (n.d.). *Explained: Neural networks*. MIT News Massachusetts Institute of Technology. Retrieved October 14, 2021, from https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[25] Davis, T. (2019, December 12). *A beginners guide to Neural Networks*. Medium. Retrieved October 14, 2021, from https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13.

[26] An, N., Zhao, W., Wang, J., Shang, D., & Zhao, E. (2013). Using multi-output feedforward neural network with empirical mode decomposition based signal filtering for electricity demand forecasting. *Energy*, *49*, 279–288. https://doi.org/10.1016/j.energy.2012.10.035

[27] Bhaskar, K., & Singh, S. N. (2012). AWNN-assisted wind power forecasting using feed-forward neural network. IEEE Transactions on Sustainable Energy, 3(2), 306–315. https://doi.org/10.1109/tste.2011.2182215

[28] Q. Zhang and A. Benveniste, “Wavelet networks,” IEEE Trans. Neural Netw., vol. 3, no. 6, pp. 889–898, Nov. 1992.

[29] J. Zhang, G. G. Walter, Y. Miao, and W. N. W. Lee, “Wavelet neural networks for function learning,” IEEE Trans. Signal Process., vol. 43, no. 6, pp. 1485–1497, Jun. 1995.

[30] Saha, S. (2018, December 17). *A comprehensive guide to Convolutional Neural Networks - the eli5 way*. Medium. Retrieved October 14, 2021, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[31] S. A. Billings and H. L. Wei, “A new class of wavelet networks for nonlinear system identification,” IEEE Trans. Neural Netw., vol. 16, no. 4, pp. 862–874, Jul. 2005.

[32] S. S. Iyenger, E. C. Cho, and V. V. Phoha, Foundation of Wavelet Networks and Applications. Boca Raton, FL: Chapman & Hall, 2002.

[33] N. M. Pindoriya, S. N. Singh, and S. K. Singh, “An adaptive wavelet neural network-based energy price forecasting in electricity markets,” IEEE Trans. Power Syst., vol. 23, no. 3, pp. 1423–1432, Aug. 2008

[34] Zhang, Y., & Li, Q. (2019). A regressive convolution neural network and support vector regression model for electricity consumption forecasting. *Lecture Notes in Networks and Systems*, 33–45. https://doi.org/10.1007/978-3-030-12385-7_4

[35] Samuel, O., Javaid, N., Khalid, A., Khan, W. Z., Aalsalem, M. Y., Afzal, M. K., & Kim, B.-S. (2020). Towards real-time energy management of Multi-Microgrid using a deep convolution neural network and cooperative game approach. IEEE Access, 8, 161377–161395. https://doi.org/10.1109/access.2020.3021613

[36] Classification of Machine Learning Algorithms, Issue 03, Volume 3 (March 2016) www.ijirae.com

[37] R. Sathya, Annamma Abraham, "Comparison of supervised and unsupervised learning algorithms for pattern classification", Int J Adv Res Artificial Intell, vol. 2, no. 2, pp. 34-38, 2013.

[38] S. Sutton, G. Barto, "Reinforcement Learning: An Introduction" in, Cambridge MA USA MIT Press, 1998.

[39] Donges, N. (2021, July 29). A guide to RNN: Understanding recurrent neural networks and LSTM Networks. Built In. Retrieved October 16, 2021, from https://builtin.com/data-science/recurrent-neural-networks-and-lstm.

[40] Cloud Education, I. B. M. (2020, September 14). Recurrent Neural Networks. IBM. Retrieved October 16, 2021, from https://www.ibm.com/cloud/learn/recurrent-neural-networks.

[41] *Hourly Load Data Archives*. Electric Reliability Council of Texas (ERCOT). (n.d.). Retrieved November 11, 2021, from http://www.ercot.com/gridinfo/load/load_hist.

[42] Vladimir Vapnik. The nature of statistical learning theory. Springer science & business media, 2013.

[43] Itamar Arel, Derek C Rose, and Thomas P Karnowski. Deep machine learning-a new frontier in artificial intelligence research [research frontier]. IEEE Computational Intelligence Magazine, 5(4):13–18, 2010.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[45] Hernández, L.; Baladrón, C.; Aguiar, J.M.; Carro, B.; Sánchez-Esguevillas, A.; Lloret, J. Artificial neural networks for short-term load forecasting in microgrids environment. Energy 2014, 75, 252–264.

[46] Huang, Y.;Wang, N.; Gao,W.; Guo, X.; Huang, C.; Hao, T.; Zhan, J. LoadCNN: A Low Training Cost Deep Learning Model for Day-Ahead Individual Residential Load Forecasting. arXiv 2019, arXiv:1908.00298.

[47] Khotanzad, A.; Afkhami-Rohani, R.; Maratukulam, D. ANNSTLF-Artificial Neural Network Short-Term Load Forecastergeneration three. IEEE Trans. Power Syst. 1998, 13, 1413–1422.

[48] Farsi, B.; Amayri, M.; Bouguila, N.; Eicker, U. On Short-Term Load Forecasting Using Machine Learning Techniques and a Novel Parallel Deep LSTM-CNN Approach. IEEE Access 2021, 9, 31191–31212.

[49] Atef, S.; Eltawil, A.B. Assessment of stacked unidirectional and bidirectional long short-term memory networks for electricity load forecasting. Electr. Power Syst. Res. 2020, 187, 106489.

[50] Takashi Kuremoto, Shinsuke Kimura, Kunikazu Kobayashi, and Masanao Obayashi. Time series forecasting using a deep belief network with restricted boltzmann machines. Neurocomputing, 137:47–56, 2014.

[51] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In Proceedings of the 24th international conference on Machine learning, pages 791–798. ACM, 2007.

[52] Pablo Romeu, Francisco Zamora-Mart´ınez, Paloma Botella-Rocamora, and Juan Pardo. Time-series forecasting of indoor temperature using pre-trained deep neural networks. In International Conference on Artificial Neural Networks, pages 451–458. Springer, 2013.

[53] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow

prediction with big data: a deep learning approach. IEEE Transactions on Intelligent

Transportation Systems, 16(2):865–873, 2015.

[54] Junfei Chen, Qiongji Jin, and Jing Chao. Design of deep belief networks for short-term prediction of drought index using data in the huaihe river basin. Mathematical Problems in

Engineering, 2012, 2012.

[55] Aditya Grover, Ashish Kapoor, and Eric Horvitz. A deep hybrid model for weather forecasting. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 379–386. ACM, 2015.

[56] Phi, M. (2020, June 28). Illustrated guide to LSTM's and GRU's: A step by step explanation. Medium. Retrieved November 23, 2021, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[57] Olah, C. (2015, August 15). Understanding LSTM networks. Understanding LSTM Networks -- colah's blog. Retrieved November 23, 2021, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[58] Saxena, S. (2021, March 18). LSTM: Introduction to LSTM: Long short term memor. Analytics Vidhya. Retrieved November 23, 2021, from https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/.

[59] Wang, Y., Liao, W., & Chang, Y. (2018). Gated recurrent unit network-based short-term photovoltaic forecasting. Energies, 11(8), 2163. https://doi.org/10.3390/en11082163

[60] A. Gerossier , R. Girard , G. Kariniotakis , A. Michiorri , A. Gerossier , R. Girard , et al. , Probabilistic day-ahead forecasting of household electricity demand, in: 24th Int Conf Electr Distrib, 2017, Jun 2017, p. 0625 .

[61] M.Q. Raza, A. Khosravi, A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings, Renew. Sustain. Energy Rev. 50 (2015) 1352–1372, doi: 10.1016/j.rser.2015.04.065 .

[62] X. Ke, A. Jiang, N. Lu, Load profile analysis and short-term building load fore- cast for a university campus, in: 2016 IEEE Power Energy Soc Gen Meet, 2016, pp. 1–5, doi: 10.1109/PESGM.2016.7742034 .

[63] R.K. Jain, K.M. Smith, P.J. Culligan, J.E. Taylor, Forecasting energy consump- tion of multi-family residential buildings using support vector regression: investigating the impact of temporal and spatial monitoring granularity on performance accuracy, Appl. Energy 123 (2014) 168–178, doi: 10.1016/j. apenergy.2014.02.057 .

[64] K.P. Amber, M.W. Aslam, S.K. Hussain, Electricity consumption forecasting models for administration buildings of the UK higher education sector, Energy Build. 90 (2015) 127–136, doi: 10.1016/j.enbuild.2015.01.008 .

[65] H. Shi, M. Xu, R. Li, Deep learning for household load forecasting –a novel pooling deep RNN, IEEE Trans. Smart Grid 3053 (5) (2017) 1–1, doi: 10.1109/ TSG.2017.2686012 .

[66] W. Kong, Z.Y. Dong, Y. Jia, D.J. Hill, Y. Xu, Y. Zhang, Short-term residential load forecasting based on LSTM recurrent neural network, IEEE Trans. Smart Grid 3053 (1) (2017) 1–11, doi: 10.1109/TSG.2017.2753802 .

[67] S. Ryu, J. Noh, H. Kim, Deep neural network based demand side short term load forecasting, Energies 10 (2017) 1–20, doi: 10.3390/en10 010 0 03 .

[68] P.-H. Kuo, C.-J. Huang, A high precision artificial neural networks model for short-term energy load forecasting, Energies 11 (1) (2018) 213, doi: 10.3390/ en11010213 .

[69] B. Zhang, J.-L. Wu, P.-C. Chang, A multiple time series-based recurrent neural network for short-term load forecasting, Soft Comput. 22 (12) (2017) 4099–4112, doi: 10.10 07/s0 050 0-017- 2624- 5 .

[70] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

[71] Sharma, S. (2021, July 4). Activation functions in neural networks. Medium. Retrieved April 7, 2022, from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[72] Brownlee, J. (2021, January 21). How to choose an activation function for deep learning. Machine Learning Mastery. Retrieved April 7, 2022, from https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning

[73] Doshi, S. (2020, August 3). Various optimization algorithms for training neural network. Medium. Retrieved April 7, 2022, from https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

[74] RIYAD HOSSAIN, DR. DOUGLAS TIMMER, Md. Machine Learning Model Optimization with Hyper Parameter Tuning Approach. Global Journal of Computer Science and Technology, [S.l.], sep. 2021. ISSN 0975-4172.

[75] 3.2. tuning the hyper-parameters of an estimator. scikit. (n.d.). Retrieved April 12, 2022, from https://scikit-learn.org/stable/modules/grid_search.html

[76] 5.4 random search - kenndanielso.github.io. (n.d.). Retrieved April 12, 2022, from https://kenndanielso.github.io/mlrefined/blog_posts/5_Zero_order_methods/5_4_Random_search.html

[77] Liu, Xueping; Li, Yibo; Wang, Qingjun (2018). Multi-View Hierarchical Bidirectional Recurrent Neural Network for Depth Video Sequence Based Action Recognition. International Journal of Pattern Recognition and Artificial Intelligence, (), S0218001418500337–. doi:10.1142/S0218001418500337

[78] Qin, C., Schlemper, J., Caballero, J., Price, A. N., Hajnal, J. V., & Rueckert, D. (2019). Convolutional Recurrent Neural Networks for Dynamic MR Image Reconstruction. IEEE Transactions on Medical Imaging, 38(1), 280-290. doi:10.1109/tmi.2018.2863670

[79] Wang, Jinjiang; Yan, Jianxing; Li, Chen; Gao, Robert X.; Zhao, Rui (2019). Deep heterogeneous GRU model for predictive analytics in smart manufacturing: Application to tool wear prediction. Computers in Industry, 111(), 1–14. doi:10.1016/j.compind.2019.06.001

[80] ERCOT. Electric Reliability Council of Texas. (n.d.). Retrieved April 17, 2022, from https://www.ercot.com/

[81] Request for Records, Data, or Documents. INFO request. (n.d.). Retrieved April 17, 2022, from https://www.ercot.com/about/contact/inforequest

BIOGRAPHICAL SKETCH

Md Riyad Hossain received his B.Sc. in Industrial and Production Engineering from Khulna University of Engineering & Technology, Bangladesh in 2014. He then began his professional career as a Manufacturing & Quality Engineer at Bangladesh Honda Private Limited, before moving to the same position at DIRD Polytex Limited, Bangladesh's largest synthetic polymer manufacturer. After just over 3.5 years in the industry, he became the Head of the Production & Quality team at DIRD. He was also an integral part of the DIRD's research and development team during his tenure. Mr. Hossain began his master's degree in January 2020 as a Presidential Graduate Research Assistant in the University of Texas Rio Grande Valley's Department of Manufacturing Engineering. He worked as a data scientist and machine learning researcher with Dr. Timmer from January 2020 to December 2021. Following that, he worked as a research assistant for the UTRGV Additive Manufacturing Team at the National Institute of Standards and Technology (NIST) project, where he worked to develop an expert system for additive manufacturing and 3D printing parts. Additionally, he served as a teaching assistant in the department of Manufacturing Engineering from January 2021 to December 2021. In May 2022, he received a Master of Science in Manufacturing Engineering from the University of Texas Rio Grande Valley. His current research interests include the use of machine learning tools and techniques in the fields of additive manufacturing and advanced machining. He can be reached by email at riyad35@gmail.com.