

An Efficient Short Term Load Demand Forecasting Using a Novel Parallel CNN-BiLSTM Hybrid Neural Network for Bangladesh Perspective

by

Md. Abdur Rahman (170021042)

Al-Amin Hossain (170021044)

Tahmid Jawad (170021154)

A Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING



Department of Electrical and Electronic Engineering
Islamic University of Technology (IUT)
Gazipur, Bangladesh

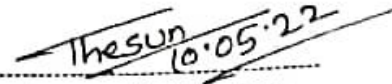
May 2022



CERTIFICATE OF APPROVAL

The thesis titled "An Efficient Short Term Load Demand Forecasting Using a Novel Parallel CNN-BiLSTM Hybrid Neural Network for Bangladesh Perspective" submitted by Md. Abdur Rahman (170021042), Al-Amin Hossain (170021044), and Tahmid Jawad (170021154) has been found as satisfactory and accepted as partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering on 10 May, 2022.

Approved by:



(Signature of the Supervisor)

Md. Thesun Al-Amin
Assistant Professor
Department of Electrical and Electronic
Engineering
Islamic University of Technology

DECLARATION

It is hereby declared that this work presented in this thesis paper was accomplished by the authors under the supervision of Mr. Md. Thesun Al-Amin, Assistant Professor, Department of Electrical and Electronic Engineering (EEE), Islamic University of Technology (IUT). It is further claimed that no portion of the work was submitted anywhere else in order to acquire a degree. Additionally, we explicitly mentioned the references whenever we advised other people's disseminated work.

Authors

Md. Abdur Rahman

Md. Abdur Rahman
ID: 170021042

Al-Amin Hossain

Al- Amin Hossain
ID: 170021044

Tahmid Jawad

Tahmid Jawad
ID: 170021154

Date: 10 May 2022

Table of Contents

DECLARATION	II
LIST OF FIGURES	VII
LIST OF TABLES	IX
LIST OF ACRONYMS	X
ACKNOWLEDGEMENT	XI
ABSTRACT	XII
CHAPTER 1 INTRODUCTION	1
1.1 OVERVIEW OF LOAD FORECASTING	1
1.2 LITERATURE REVIEW:	3
1.2.1 Overview of the STLF Work	3
1.2.2 Selection of Weather Variables	3
1.2.3 Selection of calendar variable:	4
1.2.4 Deep Learning Models for Forecasting.....	4
CHAPTER 2 BACKGROUND THEORY	6
2.1 TIME SERIES	6
2.1.1 Stationary	6
2.1.2 Non-Stationary	7
2.2 TREND	8
2.2.1 Linear trend	9
2.2.2 Non-linear trend	9
2.3 SEASONALITY.....	10
2.4 NOISE	10
2.5 AUTO CORRELATION FUNCTION	11
2.6 PARTIAL AUTOCORRELATION FUNCTION	11
2.7 STATISTICAL METHODS	12
2.7.1 Auto regressive.....	12
2.7.2 Auto regressive moving average	13
2.7.3 Auto Regressive Integrated Moving Average	14
2.8 MACHINE LEARNING	14
2.8.1 Linear Regression.....	15
2.8.1.1 Simple Linear Regression.....	15
2.8.1.2 Ordinary Least Squares	16

2.8.1.3 Gradient Descent	16
2.8.1.4 Regularization.....	16
2.8.2 Decision Trees.....	17
2.8.2.1 Classification trees.....	18
2.8.2.2 Regression trees.....	18
2.8.3 Support Vector Regression.....	18
2.8.4 K-Nearest Neighbors.....	19
2.8.5 Boosting	20
2.8.5.1 XGBoost	21
2.8.5.2 LGBost	21
2.8.6 Bagging	22
2.9 DEEP LEARNING	22
2.9.1 Convolutional Neural Network	23
2.9.2 Recurrent Neural Network	25
2.9.2.1 LSTM :	25
2.9.2.2 GRU.....	27
CHAPTER 3 METHODOLOGY	29
3.1 DATA COLLECTION	29
3.2 DATA PRE-PROCESSING:	29
3.2.1 Missing Value Imputation.....	29
3.2.2 Outliers.....	30
3.3 EXPLORATORY DATA ANALYSIS	31
3.3.1 Weather Data Merge	35
3.4 DATA SCALING.....	36
3.5 DATA WINDOWING	37
3.6 TRAIN TEST VALIDATION	37
3.6.1 Training Set.....	38
3.6.2 Validation Set.....	38
3.6.3 Testing Set.....	38
3.7 CNN MODEL	38
3.8 LSTM MODEL.....	40
3.9 GRU MODEL	41
3.10 PROPOSED MODEL (CNN-BiLSTM).....	42

CHAPTER 4 PERFORMANCE AND RESULT ANALYSIS	44
4.1 EVALUATION METRICS	44
4.1.1 Mean Squared Error (MSE)	44
4.1.2 Root Mean Squared Error	44
4.1.3 R-squared Score	44
4.1.4 Mean absolute percentage error (MAPE).....	45
4.1.5 Accuracy	45
4.2 RESULTS ANALYSIS OF DIFFERENT MODELS:	45
4.2.1 CNN model	45
4.2.2 GRU Model.....	50
4.2.3 LSTM	54
4.2.4 CNN-BiLSTM (Proposed)	58
4.3 OVERALL COMPARISON	69
CHAPTER 5 CONCLUSION & FUTURE WORKS	71
REFERENCES	73

List of Figures

FIGURE 2.1: STATIONARY TIME SERIES	7
FIGURE 2.2: NON-STATIONARY TIME SERIES.....	8
FIGURE 2.3: ACF PLOT	11
FIGURE 2.4: PACF PLOT.....	12
FIGURE 2.5: DECISION TREE.....	17
FIGURE 2.6: SVM STRUCTURE	19
FIGURE 2.7: EXTREME GRADIENT BOOSTING	21
FIGURE 2.8: LSTM CELL	26
FIGURE 2.9: GRU CELL	28
FIGURE 3.1: MISSING VALUES IN DATASET	30
FIGURE 3.2: DAILY AVERAGE DEMAND OF THE YEAR 2015.....	31
FIGURE 3.3: DAILY AVERAGE DEMAND OF THE YEAR 2016.....	31
FIGURE 3.4: DAILY AVERAGE DEMAND OF THE YEAR 2017.....	32
FIGURE 3.5: DAILY AVERAGE DEMAND OF THE YEAR 2018.....	32
FIGURE 3.6: DAILY AVERAGE DEMAND OF THE YEAR 2019.....	33
FIGURE 3.7: DAILY AVERAGE DEMAND OF THE YEAR 2020.....	33
FIGURE 3.8: DAILY AVERAGE DEMAND OF THE YEAR 2021.....	34
FIGURE 3.9: YEARLY AVERAGE DEMAND BAR CHART	34
FIGURE 3.10: TEMPERATURE VS AVERAGE DEMAND (MW) CURVE.....	35
FIGURE 3.11: HEATMAP OF AVERAGE DEMAND OF EACH HOUR OF THE DAY	35
FIGURE 3.12: HEATMAP OF AVERAGE DEMAND OF EACH MONTH OF THE YEAR	36
FIGURE 3.13: DATA WINDOWING.....	37
FIGURE 3.14: ARCHITECTURE OF CNN MODEL	39
FIGURE 3.15: ARCHITECTURE OF LSTM MODEL	40
FIGURE 3.16: ARCHITECTURE OF GRU MODEL	41
FIGURE 3.17: BLOCK DIAGRAM OF PROPOSED MODEL.....	42
FIGURE 3.18: ARCHITECTURE OF PROPOSED MODEL.....	43
FIGURE 4.1: PREDICTION VS ACTUAL PLOT OF 1ST OCTOBER OF CNN MODEL.....	46
FIGURE 4.2: PREDICTION VS ACTUAL PLOT OF 20TH APRIL,2021 OF CNN MODEL	46
FIGURE 4.3: PREDICTION VS ACTUAL PLOT OF 1ST 7-DAYS OF OCTOBER OF CNN MODEL...	47
FIGURE 4.4: REGRESSION PLOT OF CNN MODEL.....	48
FIGURE 4.5: ERROR HISTOGRAM PLOT OF CNN MODEL	49
FIGURE 4.6: PREDICTION VS ACTUAL PLOT OF 1ST SEPTEMBER OF GRU MODEL	50
FIGURE 4.7: PREDICTION VS ACTUAL PLOT OF 20TH APRIL, 2021 OF GRU MODEL	51
FIGURE 4.8: PREDICTION VS ACTUAL PLOT OF 1ST 7-DAYS OF OCTOBER OF GRU MODEL...	51

FIGURE 4.9: REGRESSION PLOT OF GRU MODEL.....	52
FIGURE 4.10: ERROR HISTOGRAM PLOT OF GRU MODEL	53
FIGURE 4.11: PREDICTION VS ACTUAL PLOT OF 1ST SEPTEMBER OF LSTM MODEL	54
FIGURE 4.12: PREDICTION VS ACTUAL PLOT OF 20TH APRIL OF LSTM MODEL.....	55
FIGURE 4.13: PREDICTION VS ACTUAL PLOT OF 1ST 7 DAYS OF OCTOBER OF LSTM MODEL	55
FIGURE 4.14: REGRESSION PLOT OF LSTM MODEL.....	56
FIGURE 4.15: ERROR HISTOGRAM PLOT OF LSTM MODEL	57
FIGURE 4.16: PREDICTION VS ACTUAL PLOT OF 1ST SEPTEMBER OF CNN-BiLSTM MODEL	58
FIGURE 4.17: PREDICTION VS ACTUAL PLOT OF 1ST OCTOBER OF CNN-BiLSTM MODEL ...	59
FIGURE 4.18: PREDICTION VS ACTUAL PLOT OF 20TH APRIL OF CNN-BiLSTM MODEL	59
FIGURE 4.19: PREDICTION VS ACTUAL PLOT OF 7 DAYS OF APRIL OF CNN-BiLSTM MODEL	60
FIGURE 4.20: PREDICTION VS ACTUAL PLOT OF 1ST 7 DAYS OF AUGUST OF CNN-BiLSTM MODEL	61
FIGURE 4.21: PREDICTION VS ACTUAL PLOT OF 1ST 7 DAYS OF SEPTEMBER OF CNN-BiLSTM MODEL	61
FIGURE 4.22: PREDICTION VS ACTUAL PLOT OF 1ST 7 DAYS OF OCTOBER OF CNN-BiLSTM MODEL	62
FIGURE 4.23: PREDICTION VS ACTUAL PLOT OF JULY, 2021 OF CNN-BiLSTM MODEL	62
FIGURE 4.24: PREDICTION VS ACTUAL PLOT OF AUGUST, 2021 OF CNN-BiLSTM MODEL..	63
FIGURE 4.25: PREDICTION VS ACTUAL PLOT OF SEPTEMBER, 2021 OF CNN-BiLSTM MODEL	63
FIGURE 4.26: REGRESSION PLOT OF CNN-BiLSTM MODEL	64
FIGURE 4.27: ERROR HISTOGRAM PLOT OF CNN-BiLSTM MODEL.....	65

List of Tables

TABLE 4.1: EVALUATION METRICS SCORE OF CNN MODEL OF INDIVIDUAL MONTHS.....	45
TABLE 4.2: EVALUATION METRICS SCORE OF GRU MODEL OF INDIVIDUAL MONTHS 2021 .	50
TABLE 4.3: EVALUATION METRICS SCORE OF LSTM MODEL OF INDIVIDUAL MONTHS	54
TABLE 4.4: EVALUATION METRICS SCORE OF PROPOSED MODEL OF INDIVIDUAL MONTHS 2021.....	58
TABLE 4.5: DAILY AVERAGE ACTUAL DEMAND AND PREDICTED DEMAND FOR JULY MONTH OF PROPOSED MODEL.....	66
TABLE 4.6: DAILY AVERAGE ACTUAL DEMAND AND PREDICTED DEMAND FOR AUGUST MONTH OF PROPOSED MODEL	67
TABLE 4.7: DAILY AVERAGE ACTUAL DEMAND AND PREDICTED DEMAND FOR SEPTEMBER MONTH OF PROPOSED MODEL	68
TABLE 4.8: COMPARISON OF EVALUATION METRICS SCORE OF INDIVIDUAL MONTHS OF MODELS.....	69
TABLE 4.9: COMPARISON OF EVALUATION METRICS SCORE ON WHOLE TEST SET OF MODELS	70
TABLE 4.10: COMPARISON AMONG PROPOSED MODEL WITH RECENT EXISTING MODELS....	70

List of Acronyms

ACF	Auto Correlation Function
ANN	Artificial Neural Network
AR	Auto Regressive
ARIMA	Auto Regressive Integrate Moving Average
ARMA	Auto Regressive Moving Average
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
KNN	K-Nearest Neighbours
LGBOOST	Light Gradient Boosting
LSTM	Long Short-Term Memory
LTLF	Long Term Load Forecasting
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Square Error
MTLF	Mid Term Load Forecasting
PACF	Partial Autocorrelation Function
PGCB	Power Grid Company Bangladesh
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
STLF	Short Term Load Forecasting
SVM	Support Vector Machine
SVR	Support Vector Regression
XGBOOST	Extreme Gradient Boosting

Acknowledgement

First and first, we would like to thank ALLAH Almighty, the Most Omnipotent, the Most Merciful, and the Most Beneficial, for blessing us with the opportunity to contribute to this work. We would want to express our heartfelt gratitude to everyone who has helped, assisted, and advised us on several occasions during our undergraduate studies.

We would like to express our heartfelt appreciation to adviser, Md. Thesun Al-Amin, Assistant Professor, Department of Electrical and Electronic Engineering, Islamic University of Technology (IUT), for his important guidance, encouragement, and recommendations during our studies and research. He provided several study prospects and encouraged us to think beyond the boundaries. We appreciate all of his instruction and expertise.

We are also indebted to the Electrical and Electronic Engineering Department professors at Islamic University of Technology (IUT) for their motivation and assistance.

Finally, we want to express our gratitude to our families for their unwavering moral support.

Abstract

STLF (Short Term Load Forecasting) has traditionally become one of the most crucial, delicate, and precise demanding variables in energy systems. An efficient STLF enhances not only the financial feasibility of the system, but also its safety, consistency, and dependability in performance, allowing for the realization of a prospective Smart Electricity System. The state-of-the-art models exhibit significant nonlinearity in load information from available projections, as well as limited applicability in real-world circumstances. However, for real-world application, the energy forecasting area requires better resilience, improved prediction accuracy, and adaptability capacity. The study given in this paper supports the case for a hybrid strategy, in which the complimentary qualities of several cognitive methodologies are merged to provide a superior solution to the STLF problem. The deep learning models for STLF integrated with statistical techniques are presented in this paper for an accurate load forecasting. Temperature, humidity, and day type are all taken into account since they have a substantial effect on the overall performance of an appropriate STLF. The load demand data has been collected from the PGCB database, the weather data has been collected from the rp5 archive and the holidays are considered from the government calendar which excludes the data collection part of our research. The data refinement process has been done where many preprocessing techniques are applied on the raw data. With the proper data analysis and scaling the further process fed into the deep learning models where the training, validation and testing were done. In terms of computing complexity and prediction accuracy, the suggested model outperforms the prior hybrid models. The proposed technique of our methodology against existing power prediction information reveals that it performs better in terms of precision and accuracy. The evaluation has been done considering the Mean Absolute Percentage Error (MAPE) and R-squared score which outperforms the existing literature reviews. When compared to existing baseline models, the suggested technique had the lowest error rate on the Power Grid Company Bangladesh dataset.

Chapter 1

INTRODUCTION

At present one of the most renowned problems in the global world is the increasing amount of electricity consumption. In Bangladesh the growing population tends to affecting the electric power system the most. The proper planning of the generation and distribution ensures the best analysis of the system[1]. For the development of a country, the proper forecasting of the electricity consumption with a good management structure led to beneficial of economics.

In the modern world most of the countries are working on the problem to solve it. Electric load forecasting comes out as a better solution to it where the supervision of the electricity consumption can be done predicting the future load based on the historical data. For the effective and systematic load forecasting the distribution system planning [1] is an important factor to look on. Accurate load forecasting leads to the appropriate power supply and the service of the power system depending on the approval of the resources available.

1.1 Overview of Load forecasting

The load forecasting depends on different parameters or features which includes the data of the time, weather, load shedding, holiday etc. It is very important in today's world ensuring a profitable energy market [2]. The energy world is dealing with rapid change which needs a progressive, effective and decent automation system in order to fulfill the adequate amount of demand required by the customer. The appropriate knowledge of both the requirement of residential and non-residential load is needed. For that the introduction of load forecasting has come out as a substitute solution of the modern issue[3]. Different forecasting techniques incorporate economic data, method of collecting data, and authorized load plans as information.

Different techniques are there for utilizing the load forecasting for different applications. The classification mainly depends on the time period of the users consumed by them and also the effect of various information for example the geographical change, weather data, cultural holiday. For different purposes the different approaches of the load forecasting are applied to make a prediction.

Load forecasting are classified into three types which are depending on the cycle period of the electrical load: Short Term Load Forecasting (STLF), Medium Term Load Forecasting (MTLF) and Long-Term Load Forecasting (LTLT). The time cycle for STLF follows for an hour to a week depending on the operation whereas in MTLF the period for forecasting is more than a week to few months. For an advanced planning management, LTLF is considered which ranges from a year to few years.

Weather-related report including the temperature, humidity, rainfall, dew factor etc. has significant impact on Load forecasting. The weather variable like temperature has the bigger impact on the behavior of the load consumption due to the variations of the weather in different region. With the help of technology, the data of the temperature, humidity, dew factor etc. are available and the use of this data ensures the better performance.

Apart from the weather, the government holidays and cultural off days are counted as the relevant one in the terms of the load forecasting. The load demand is comparatively high on the weekdays compared to the weekend considering the human activity. The holidays put a significant effect on the load performance. The load patterns can also be affected due to the sudden political restrictions or the natural disease. Therefore, the cultural celebration and unwanted restrictions are a great concern in case of load behavior.

In case of a time series one of the prominent problems is the prediction of the future load demand which compromises of both univariate or multivariate features. The recorded data includes some outliers, missing values, redundancy which show seasonal variations and nearly trend but irregularity for corona in the load pattern[4]. For accurate load forecasting the state-of-the-art techniques of machine learning are unavailable to give good performance.

Comparing to machine learning algorithms, the deep learning models help to improve the accuracy in different domain of data science. (CNN-GRU) Deep learning models for example Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) are one of the most important one to predict the load demand. For the forecasting the strategies depending on the deep learning model along with the novel hybrid ones are presented in the related literature. The computational results in case of deep learning model cross the outcomes in case of traditional methods and gives a better output in the STLF which removes the burden of the management people in the power development board of the country.

In our research the scope of the work is narrowed to include only the versions of Short-Term Load Forecasting.

1.2 Literature Review:

In Chapter 1, the importance of the accurate load forecasting being discussed which helps in the planning and management operation. In case of load forecasting applying different techniques of machine learning, many reports have been disclosed in more than a decade. The literature review deals with the review of the work done of STLF and different parameters to implement the STLF.

1.2.1 Overview of the STLF Work

Electrical load is nonlinear, dynamic and diverse in nature where the STLF which has been approached using different machine learning techniques and some hybrid ones too. During the initial period of research the statistical methods were more applicable in case of linear and non linear regression techniques[5], curve fitting techniques, least square approximation techniques, time series analysis [6]etc. In recent years, the Artificial intelligence has a great progress in terms of accuracy. A lot of approaches have been developed which can preprocess the load data in the form of a good pattern of consumption. One of the most popular procedures that has been popular in the recent times for STLF includes the Artificial Neural Network (ANN) [7] Depending on the dataset and the output different hybrid models are also developed.

In case of developing an accurate load forecasting the variables needs to be selected correctly depending on the availability of the right data. In many research papers, the variables need to be processed for better analysis of the forecasting. Different variables for example the effect of temperature and humidity [8] are analyzed in different possible scenarios like the effect of the humidity and wind speed were treated as a linear transformation of temperature which is the better results of the earlier paper [9]. Various weather forecasting methods like the weather forecaster [10], the temperature forecaster [11] are helping to progress the work on the weather forecasting sector.

1.2.2 Selection of Weather Variables

In paper [12] reviewed the electricity demand and weather data from a utility company in Midwest, U.S. Using support vector regression method, the short term multi region load has been predicted. For such proposal the weather variables were taken into account. The recommended technique helped to identify the leading region with the existing weather conditions based on the load which produced the best results of the area. The mathematical results achieved for various area division techniques confirm the proposed multi region forecasting system's efficacy.

In [13] the variables are categorized as weather sensitive and non-weather sensitive variables. In the paper a new developed rule is used for obtaining the short-term load forecast. The created method was tailored for load forecasting for the next day and week. Before formulating the rules, several concerns connected to the dependence of meteorological factors, seasonal fluctuations, and distinct day type characteristics were extensively investigated. The mechanism of the updating rules is also discussed which is a self-learning process. Comparison of different method has been done seeing the accuracy of the load forecasting results using the weather information of both accurate and predicted. The obtained results were of monthly average load forecasting which ranging the error between 2.97% and 10.77% for the week ahead whereas for the day it was the average seasonal errors obtained in the range of 1.03% and 7.53%.

1.2.3 Selection of calendar variable:

Most of the researcher are not fond of the information extracted from the calendar. In most of the paper the effect of holiday is not considered and also the effect of month of the year was not considered [8], [9], [11]

Many papers do, in fact, take into account the influence of the calendar info on the STLF. In the paper [14], the authors reviewed the seasonal variation of everyday electric load. To extract seasonal subsets from the historical record, it follows a standard classification approach using hourly temperature readings. The statistical techniques help to develop a response function fitting to each season. The established functional models are used by the power management with the model library.

1.2.4 Deep Learning Models for Forecasting

Ge Zhang, Xiaoqing Bai, and Yuxuan Wang in [15] implemented a CNN-Seq2Seq model for short-time multi-energy load forecasting. In this paper, CNN is mainly used to extract features of the input dataset. An enhanced CNN with Seq2Seq model is used to extract time-series components of the dataset. As the paper focused more on multi-energy load forecasting, more importance is given to the coupling relationship. An attention mechanism is also placed to gain helpful information by calculating the weight matrix, which in return gives a more accurate prediction.

In a paper by Shengtao He, Canbing Li, Xubin Liu, Xinyu Chen, Mohammad Shahidehpour, Tao Chen, Bin Zhou, and Qiuwei Wu [16] a per-unit curve rotated decoupling (PCRD) method is proposed for day-ahead load forecasting with CNN-TCN framework to overcome the shortcomings of the per-unit curve static decoupling (PCSD) method. CNN is used here to extract the shape feature of the load curve. The temporal features and average loads are extracted by TCN. This method shows higher accuracy and stability in predicting day-ahead load forecasting.

A novel Evolutionary-based CNN Model for intelligent load forecasting is proposed by Seyed Mohammad Jafar Jalali, Sajad Ahmadian, Abbas Khosravi, Miadreza Shafiekhah, Saeid Nahavandi, and Joao P.S. Catalao in [17]. In this paper, a deep neuroevolution algorithm is used to automatically design the CNN structures using a novel modified evolutionary algorithm called enhanced grey wolf optimizer (EGWO). The difference between typical CNN models and the proposed network is that the architecture of CNN and its hyperparameters are optimized by the EGWO algorithm for enhancing its load forecasting accuracy.

A hybrid residual dilated LSTM and exponential smoothing model for midterm electric load forecasting is proposed by Grzegorz Dudek, Paweł Pełka, and Slawek Smyl in [18]. In this research, a novel deep learning model comprised of exponential smoothing (ETS), advanced long short-term memory (LSTM), and ensembling is used to predict a time series for 35 European countries monthly electricity demand successfully. Here the LSTM is equipped with dilated recurrent skip connections and a spatial shortcut path from lower layers which ensured efficient training.

Mehak Khan, Hongzhi Wang, Adnan Riaz, Aya Elfatyany and Sajida Karim proposed a bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification in [19]. In this paper state-of-the-art techniques, bidirectional long short-term memory (BiLSTM), fully convolutional network, and attention mechanism is used to develop a deep learning model. This proposed ABiLSTM-FCN hybrid deep learning architecture is then used to validate the performance on 85 datasets from the University of California Riverside (UCR) univariate time series archive which yielded satisfactory result.

Chapter 2

BACKGROUND THEORY

2.1 Time Series

A time series is a sequence of data points indexed in time order. A time series is most frequently defined as a succession of dates taken at evenly spaced moments in time. It is a series of discrete-time data [20].

Time series are utilized in statistics, signal processing, pattern recognition, weather forecasting, earthquake prediction, control engineering, communications engineering, and, in general, any applied scientific and engineering subject that incorporates temporal data. Data from time series have a natural temporal ordering. This distinguishes time series analysis from cross-sectional investigations, in which the observations are not naturally ordered. Time series analysis differs from spatial data analysis in that the observations are usually related to geographic locations [21].

Methods for evaluating time series data in order to derive relevant statistics and other data features are included in time series analysis. The employment of current and future values estimation based on past observed values is known as time series forecasting. A stochastic model for a time series will represent the fact that data near together in time are more tightly connected than observations further away. Furthermore, time series models frequently make use of time's intrinsic one-way ordering, such that values for a particular period are depicted as originating in some way from past values rather than future values[22]

Time series datasets can be classified as stationary and non-stationary based on trend and seasonality.

2.1.1 Stationary

A stationary process is a stochastic process in which the unconditional joint probability distribution does not change as time passes[23], [24]. This indicates that a stationary process's distribution would appear the same at various points in time. As a result, a stationary process's parameters, such as mean and variance, remain constant across time. If a time series shows stationarity, it is classified as a stationary time series[25]

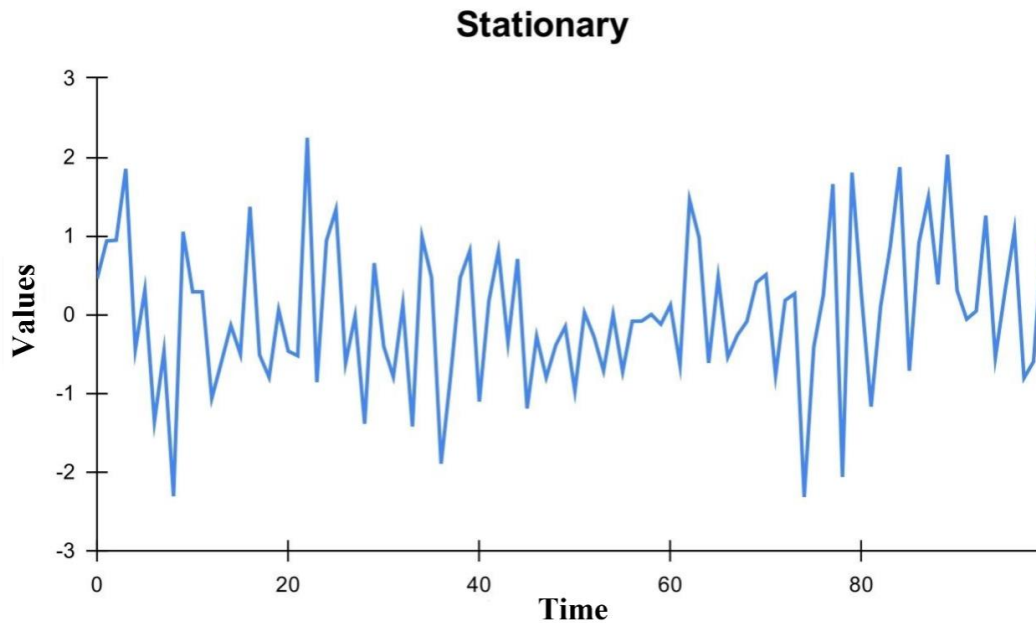


Figure 2.1: Stationary Time Series

A time series whose statistical features, such as mean and variance, stay constant across time is known as stationary time series. This means a stationary time series is one whose statistical features are independent of the time point at which it is examined. The variance of a stationary time series is constant, and it always returns to the long-run mean. Stationary time series do not have trends or seasonality[26]

It is easier to model a time series when it is stationary. Therefore, statistical modeling approaches are based on the assumption that the time series is stationary[25]

2.1.2 Non-Stationary

A non-stationary time series is one in which the statistical characteristics fluctuate with time. A time series containing a trend or seasonality is therefore non-stationary. This is due to the fact that the presence of trend or seasonality affects the mean, variance, and other statistical features at any given period. A non-stationary time series' statistical features are a function of the time at which it is observed[27].

In non-stationary time series, summary statistics such as the mean and variance fluctuate with time, causing a drift in the characteristics that a traditional statistical modeling method may attempt to capture. To prevent this, classical time series analysis and forecasting approaches are focused on making non-stationary time series data stationary by finding and eliminating trends and stationary effects[25].

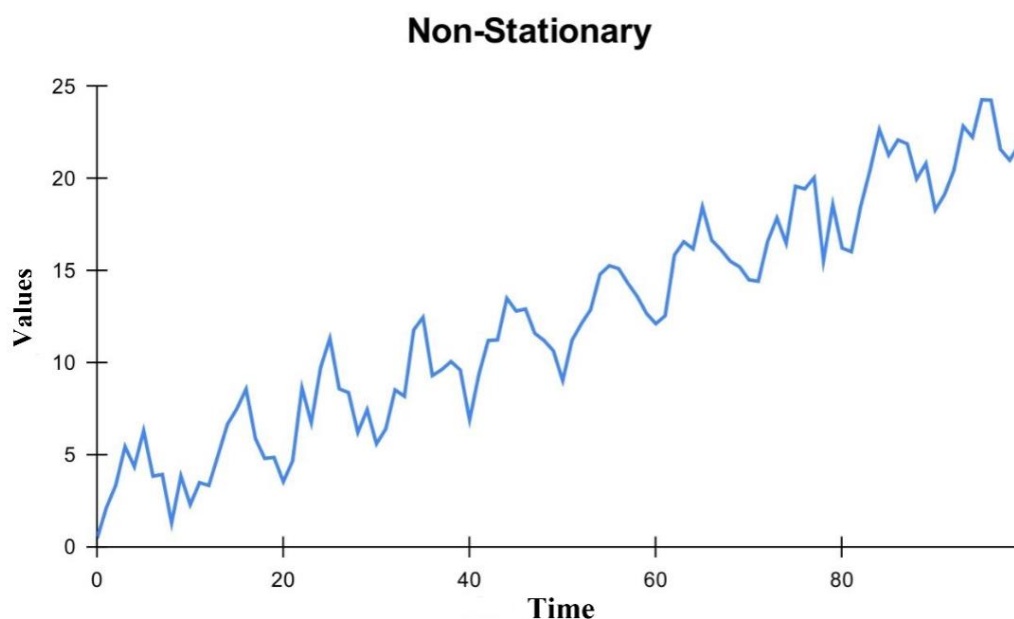


Figure 2.2: Non-Stationary Time Series

One of the most used tests for stationarity is the Augmented Dickey-Fuller test. The Kwiatkowski-Phillips-Schmidt-Shin test is another widely used stationarity test. It is critical to highlight that the meaning of KPSS is diametrically opposed to that of the ADF test, and so these tests cannot be employed interchangeably[26], [28].

2.2 Trend

A trend is a data pattern that depicts the progression of a series to substantially higher or lower values over time. A trend is seen when the slope of a time series increases or decreases. A trend typically lasts for a short period of time before dissipating; it does not reoccur. By raising or lowering the level, a trend causes a time series to become non-stationary. This causes the mean value of the time series to change over time[29].

If a time Series Analysis reveals a general upward pattern, it is referred to be an uptrend; if it reveals a general downward pattern, it is referred to as a downtrend[29]

The Mann-Kendall Test or M-K test can be used to detect if a time series is trending higher or downward monotonically. It is a non-parametric test; therefore, it can be applied to any distribution. As a result, to run this test, the time series does not have to satisfy the normality condition, but the data should be devoid of serial correlation[30]. This test does not work if the dataset had alternative upward and downward trends. So, if the dataset consists of seasonality, instead of the M-K test, the Seasonal Kendall test shortly, SK test is used. SK test is a broader version of the M-K test where for each season, a different M-K test is run and compared separately. This test nullifies the effect of seasonality in trend determination[31].

Different types of differencing methods can be used to remove trends from a time series[25].

2.2.1 Linear trend

A linear trend is characterized by a consistent drop or increase in numbers over time. This information shows a straight line angled diagonally upward or downward on a graph. So, the linear trend might be either up or down. Linear trend is generally easy to deal with. First order differencing is generally enough to remove this type of trend[32].

2.2.2 Non-linear trend

A non-linear trend is characterized by an inconsistent drop or increase in numbers over time. A nonlinear trend is demonstrated by data that increases or decreases by increasing or decreasing amounts at each subsequent time period. Logarithmic transformation can be used to make non-linear time series that increase or decrease by an equal percentage at each consecutive time period linear. Non-linear trend can be classified as exponential trend and damped trend[32].

In the case of exponential trend, plotting the time series results in non-linear curving lines where the data rises or falls at a faster pace than at a constant rate. If the trend is upward, instead of a straight line going diagonally up or down, the graph will display a curved line with the last point in later periods being higher than the initial period. And in case of downward exponential trend, the graph will display curved line where the last point in later times is lower than the earlier time points[33].

In a damped trend, the data points will initially increase or decrease at a step rate. After a certain point, the trend stops rising or falling and almost becomes a straight line parallel to the x-axis[33].

2.3 Seasonality

In time series, seasonality is a phenomenon where the data undergoes regular and predictable changes at regular periods smaller than a year[34]. Any predictable fluctuation or pattern that recurs or repeats over a period of time that does not extend beyond a one-year period is said to be seasonal[35]. Seasonality can recur on a weekly, monthly, or quarterly basis. Seasonality is produced by a variety of causes, including weather, vacation, and holidays, and it consists of periodic, repeating, and typically regular and predictable patterns in time series levels[36].

Seasonal fluctuation is studied for a variety of reasons. The seasonal effect explanation offers a better understanding of the component's influence on a particular time series. After establishing the seasonal pattern, strategies for removing it from the time series can be adopted in order to analyze the influence of other components such as cyclical and irregular fluctuations. This process of removing the seasonal influence is known as de-seasonalizing or seasonal adjustment of data[37].

There are many graphical techniques that can be used to detect seasonality. Run sequence plot, seasonal plot, seasonal subseries plot, multiple box plots, autocorrelation plot, spectral plot are the most used techniques to detect seasonality[38].

Seasonal variance is quantified using an indicator known as a seasonal index. It is an average that may be used to compare an actual observation to what it might be if seasonal fluctuation did not exist. Method of simple averages, ratio to trend method, ratio-to-moving-average method, and link relative method is used to measure seasonal variations of a time series data[34].

2.4 Noise

Noise is defined as random variations in a time series' regular pattern. There are two forms of noise in time series: white noise and red noise[39].

If the variables are independent and identically distributed with a mean of zero, the time series is called white noise. This indicates that all variables have the same variance and that each value in the series has a zero correlation with all other data in the series. The series is known as Gaussian white noise if the variables in it are derived from a Gaussian distribution[40].

Fisher's test is used to check for white noise. Fisher's test determines if a sequence is white noise or not. If the Fisher's test result is less than 0.5, the sequence is deemed noise-free[39].

If a time series has a zero mean, constant variance, and serial correlation in time, it is a red noise sequence. The power spectrum of red noise is skewed toward low frequencies[41].

2.5 Auto Correlation Function

The autocorrelation function (ACF) describes how data points in a time series are connected to each other on average[42]. In other words, it assesses the signal's self-similarity over multiple delay durations. As a result, the ACF is a function of the delay or lag, which defines the time shift into the past used to evaluate the similarity of data points[43].

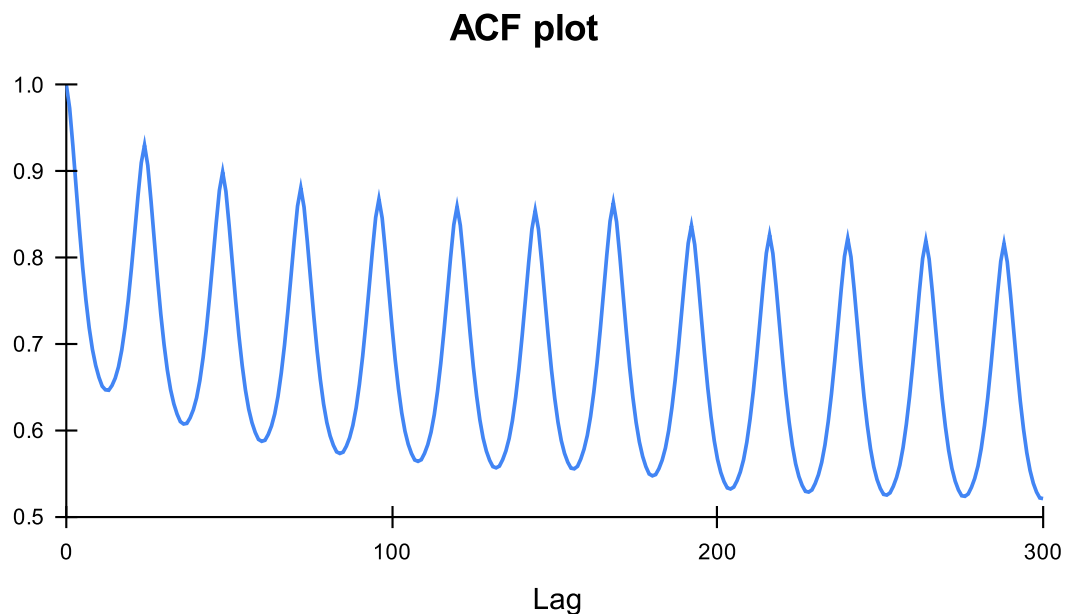


Figure 2.3: ACF Plot

Autocorrelation analysis is a mathematical method for detecting recurring patterns, such as the presence of a periodic signal disguised by noise, or determining the missing fundamental frequency in a signal inferred by its harmonic frequencies[43].

Specific types of autocorrelation processes include unit root processes, trend-stationary processes, autoregressive processes, and moving average processes[43].

2.6 Partial Autocorrelation Function

A partial autocorrelation is a description of the relationship between an observation in a time series and observations at previous time steps that excludes the associations of intervening observations. A plot of the partial autocorrelation of a time series by lag is called the partial autocorrelation function, or the acronym ACF. This plot is sometimes called a partial correlogram or a partial autocorrelation plot[44].

The autocorrelation between an observation and an observation from a previous time step includes both direct and indirect correlations. These indirect correlations are a

linear function of the correlation between data at different time stages. The partial autocorrelation function aims to eliminate these indirect connections. This is the reasoning for partial autocorrelation[44].

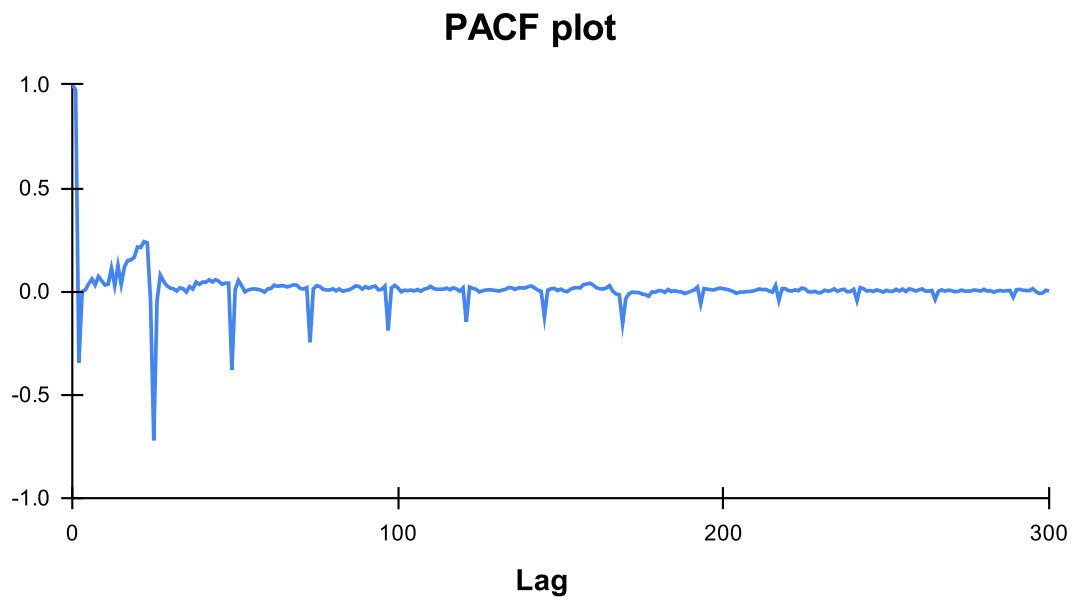


Figure 2.4: PACF Plot

This function is used in data analysis to determine the magnitude of the lag in an autoregressive model. The usage of this function was introduced as part of the Box–Jenkins technique to time series modeling, in which charting the partial autocorrelative functions allowed one to estimate the proper lags p in an AR (p) model or an extended ARIMA (p,d,q) model[45].

2.7 Statistical Methods

2.7.1 Auto regressive

An autoregressive (AR) model is a representation of a type of random process in statistics, econometrics, and signal processing; as such, it is used to explain certain time-varying processes in nature, economics, and so on. The autoregressive model specifies that the output variable is linearly dependent on its own previous values as well as a stochastic term; thus, the model takes the form of a stochastic difference equation[46].

A complete Auto Regressive (AR alone) model is one in which Y_t is solely determined by its own lags.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} + \epsilon_1$$

Here,

α = Intercept Term

Y_{t-1} = Lag1 of the Series

β_1 = Coefficient of Lag1

An AR model forecasts future behavior based on historical data. When there is some connection between values in a time series and the values that before and succeed them, it is utilized for predicting. The procedure is essentially a linear regression of the current series' data against one or more previous values in the same series[47].

2.7.2 Auto regressive moving average

Autoregressive–moving-average (ARMA) models in time series statistical analysis provide a concise description of a stationary stochastic process in terms of two polynomials, one for the autoregression (AR) and the other for the moving average (MA)[48]. The generic ARMA model first described in Peter Whittle's 1951 thesis, Hypothesis testing in time series analysis[49], and it was popularized in George E. P. Box and Gwilym Jenkins' 1970 book[50].

Similarly, a simple Moving Average (MA alone) model is one in which Y_t is determined only by the delayed prediction error.

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \dots + \phi_p \epsilon_{t-p}$$

Here,

α = Intercept Term

ϵ_t = Error of AR models of respective lags

The ARMA model is a technique for analyzing and forecasting future values in a time series. The AR component entails regressing the variable on its own lag values. The MA portion entails modeling the error term as a linear combination of error terms that occur concurrently and at different moments in the past. The model is often known as the ARMA(p,q) model, where p denotes the order of the AR portion and q denotes the order of the MA part. The Box–Jenkins technique can be used to estimate ARMA models. For low-order polynomials (of degree three or less), this technique was beneficial[51]. The ARMA model is just a white noise infinite impulse response filter with some additional interpretation attached to it[48].

2.7.3 Auto Regressive Integrated Moving Average

An autoregressive integrated moving average (ARIMA) model in time series analysis is a generalized version of an ARMA model[52]. In some circumstances, wherein data exhibit signs of non-stationarity in the context of mean, an introductory differencing stage can be performed one or more times to eradicate the mean function's non-stationarity. ARIMA can be used in time series where it has shown non stationarity in the mean but cannot be used if the variance or autocovariance is different[53].

The AR component of ARIMA denotes that the developing variable is regressed within its lag values. The MA component implies that the regression error is a linear combination of error terms whose contents happened simultaneously and at different periods in the past[53]. The I stand for "integrated." It denotes that perhaps the values have already been substituted with the gap across their current and former values. This differencing procedure may have been repeated several times. Each of these characteristics exists to make the model suit the data as closely as feasible.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} + \epsilon_1 + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \dots + \phi_q \epsilon_{t-q}$$

Y_t = Predicted Value

$\beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} + \epsilon_1$ = Linear Combination Lags of Y (Upto p Lags)

$\phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \dots + \phi_q \epsilon_{t-q}$ = Linear Combination of Lagged Forecast Errors (Upto q lags)

Non-seasonal ARIMA models are commonly denoted as ARIMA(p,d,q), where p is the degree of the autoregressive model, d is the order of differencing, and q is the degree of the moving average model. Seasonal ARIMA models are commonly designated as ARIMA(p,d,q)(P, D, Q)m. Here m is the number of seasons, and the uppercase P, D, and Q are the autoregressive, differencing, and moving average components for the seasonal element of the ARIMA model[54].

2.8 Machine learning

Machine learning (ML) is the study of computer systems that can improve themselves automatically based on experience and data[55], [56]. It is a subfield of artificial intelligence that is predicated on the premise that systems can learn from data, spot patterns, and make choices with little or no human interaction[57].

2.8.1 Linear Regression

Linear regression is a statistical method for modeling the connection between a scalar response and one or more explanatory factors[58]. Simple linear regression is used when there is only one explanatory variable; multiple linear regression is used when there is more than one[59]. This phrase differs from multivariate linear regression, predicting several correlated dependent variables rather than a single scalar variable[60].

Relationships are modeled using linear predictor functions whose unknown model parameters are derived from data in linear regression. These are known as linear models[61]. The conditional mean of the answer given the values of the explanatory factors is most usually believed to be an affine function of those values; less typically, the conditional median or another quantile is employed. Like other kinds of regression analysis, Linear regression concentrates on the conditional probability distribution of the answer given the predictor values rather than the joint probability distribution of all of these variables, which is the realm of multivariate analysis.

Linear regression models are frequently fitted using the least squares method. However, they may also be fitted using alternative methods, such as minimizing the "lack of fit" in some other norm, or minimizing a penalized version of the least squares cost function, as in ridge regression and lasso. In contrast, the least squares method may be used to fit models that are not linear. As a result, while the phrases "least squares" and "linear model" are related, they are not synonymous[58].

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n$$

Here, y is the dependent variable,

$b_1, b_2, b_3, \dots, b_n$ are coefficients of the independent variables,

b_0 is the intercept.

Learning a linear regression model entails guessing the values of the coefficients used in the representation based on the given data[62].

There are many techniques to prepare a linear regression model. The most common methods are Simple Linear Regression, Ordinary Least Squares, Gradient Descent, and Regularization.

2.8.1.1 Simple Linear Regression

Statistics may estimate simple linear regression coefficients with a single input. This necessitates calculating statistical features such as means, standard deviations, correlations, and covariance from the data. To traverse and calculate statistics, all of the data must be available. In practice, this strategy is ineffective.

2.8.1.2 Ordinary Least Squares

When there are several inputs, Ordinary Least Squares can be utilized to estimate the coefficient values. The sum of the squared residuals is minimized using the Ordinary Least Squares approach. This implies that, given a regression line across the data, the distance from each data point to the regression line is computed, squared, and the sum of all squared errors is added. Ordinary least squares aim to reduce this amount.

This method analyzes the data as a matrix and uses linear algebra operations to estimate the best coefficient values. It means that all data must be available, and enough memory must be available to accommodate the data and conduct matrix operations. It is rare to use the Ordinary Least Squares approach except as a linear algebra exercise. It is more probable that a technique from a linear algebra library will be called. This approach is incredibly quick to compute[63].

2.8.1.3 Gradient Descent

When one or more inputs may be utilized to optimize the coefficient values by iteratively reducing the model's error on the training data, the procedure is called Gradient Descent, which begins with random values for each coefficient. The total squared errors are computed for each pair of input and output values. As a scale factor, a learning rate is utilized, and the coefficients are updated in the direction of error minimization. The procedure is continued until either a minimum sum squared error is reached or no more improvement is achievable.

When employing this approach, a learning rate (alpha) parameter that specifies the amount of the improvement step to take on each iteration of the operation must be chosen. Gradient descent is frequently taught using a linear regression model since it is simple to grasp. It is beneficial in practice when there is a huge dataset, either in terms of the number of rows or the number of columns, that may not fit into memory[64].

2.8.1.4 Regularization

Regularization methods are additions to the training of the linear model. These strive to minimize both the model's sum of squared errors on the training data (using ordinary least squares) and the model's complexity (like the number or absolute size of the sum of all coefficients in the model).

Lasso Regression and Ridge Regression are two well-known regularization strategies for linear regression.

- **Lasso Regression:** Where the Ordinary Least Squares method is modified to minimize the absolute sum of the coefficients (called L1 regularization)[65].
- **Ridge Regression:** Where Ordinary Least Squares is adjusted to minimize the coefficients' squared absolute sum (called L2 regularization)[66].

These approaches are helpful when there is collinearity in the input values and ordinary least squares will overfit the training data.

2.8.2 Decision Trees

Decision Trees are a kind of Supervised Machine Learning in which data is constantly separated based on a certain parameter. Two entities may explain the tree: decision nodes and leaves. The decisions or consequences are represented by the leaves. And the data is separated at the decision nodes[67]. The objective is to build a model that predicts the value of a target variable using basic decision rules derived from data attributes. A tree is an example of a piecewise constant approximation.

There are two main types of Decision Trees: Classification trees (Yes/No types) and Regression trees (Continuous data types)[67].

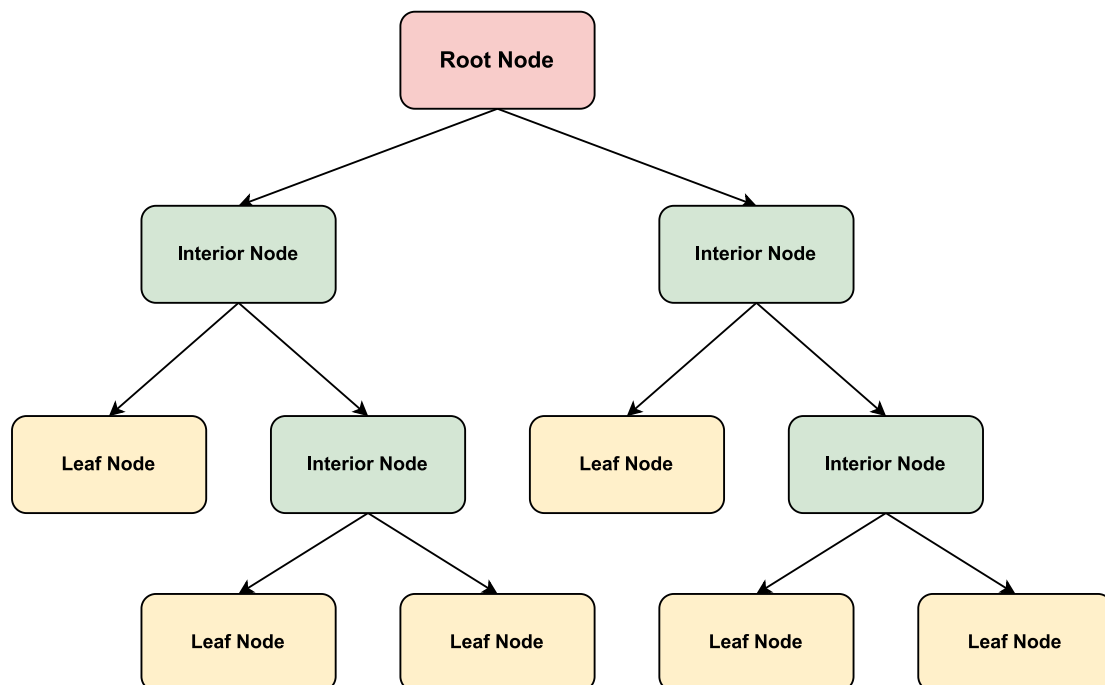


Figure 2.5: Decision Tree

2.8.2.1 Classification trees

Tree models where the target variable can take a discrete set of values are called classification trees.

2.8.2.2 Regression trees

Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

Data preparation is minimal for DT. Other procedures frequently need data standardization, the creation of dummy variables, and the removal of blank values. This module, however, does not allow missing values. It can work with both numerical and categorical data. Other strategies are often limited to examining datasets with a single variable type[68].

Decision tree learners might produce too complicated trees that do not generalize well to new data. This is known as overfitting. To prevent this problem, mechanisms like pruning, limiting the number of samples required at a leaf node, and limiting the tree's depth are required[69].

2.8.3 Support Vector Regression

SVR (Support Vector Regression) is a supervised learning approach for predicting discrete variables. SVR operates on the same premise as support vector machines (SVM). SVR's primary concept is to identify the optimum fit line. The best fit line in SVR is the hyperplane with the most significant number of points[70].

The primary goal of regression-based machine learning algorithms is to anticipate the predictand using a mapping function. This mapping function is represented by providing a collection of characteristics and predictand data known as the training data set. SVR is utilized in a variety of applications, including image processing, remote sensing, and blockchain. It has excellent generalization ability as well as good precision. Furthermore, the computational complexity is independent of the input feature data set[71], [72]

Unlike other Regression models, which seek to minimize the difference between the actual and projected values, the SVR seeks to fit the best line within a specific range. The distance between the hyperplane and the boundary line is the threshold value. The fit time complexity of SVR is more than quadratic with the number of samples, making it difficult to scale to datasets with more than a few tens of thousands of samples[70].

Linear SVR or SGD Regressor is used for big datasets. Linear SVR is quicker than SVR but takes into account the linear kernel. Because the cost function skips sample whose prediction is near their goal, the model built by Support Vector Regression is only dependent on a fraction of the training data[73].

Due to enhanced optimization tactics for a large range of variables, SVR outperforms other algorithms such as Linear Regression, KNN, and Elastic Net in performance prediction. Furthermore, it is adaptable in dealing with geometry, transmission, data generalization, and kernel-specific functionality. This extra functionality improves the model's prediction capability by taking feature quality into account[71], [72].

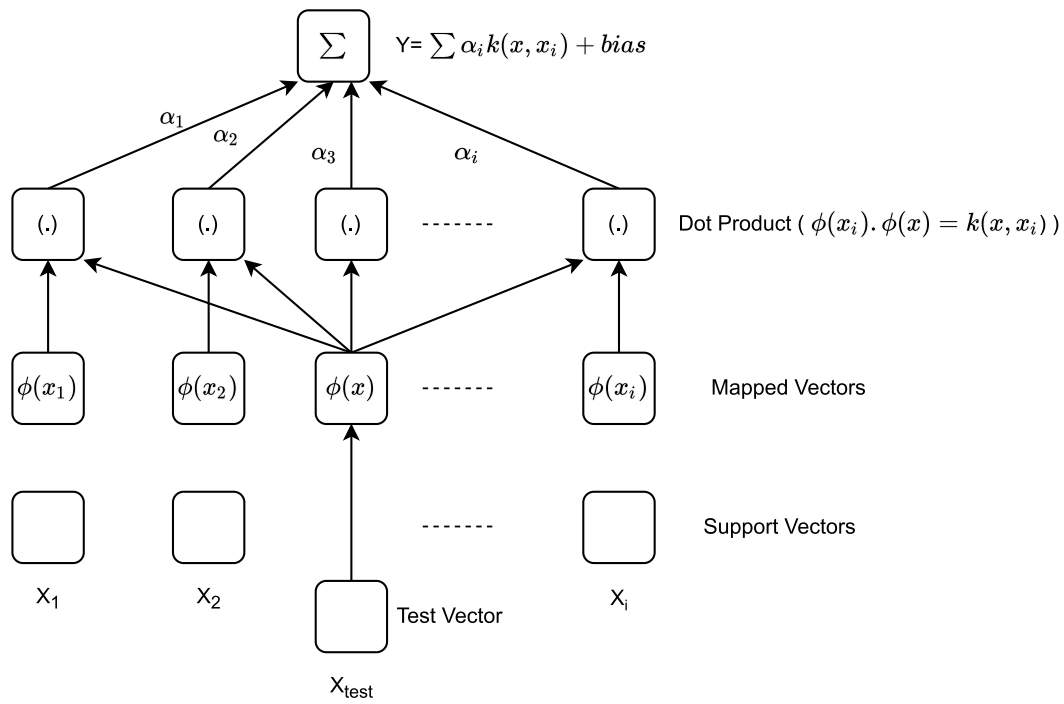


Figure 2.6: SVM Structure

Because the SVR technique is susceptible to interference in the training data, the training samples impact the SVR model's fitting performance. Furthermore, SVR is beneficial in addressing high-dimensional feature regression problems and performs well if the feature metrics are bigger than the sample size.

2.8.4 K-Nearest Neighbors

K-nearest neighbors (KNN) is a supervised learning technique that may be used for regression and classification. By computing the distance between the test data and all of the training points, KNN attempts to predict the proper class for the test data. Then choose the K number of points that are closest to the test data. The KNN method analyzes the likelihood of test data belonging to the classes of 'K' training data, and the class with the highest probability is chosen. The value in the case of regression is the mean of the 'K' chosen training points[74].

The outcome of KNN classification is a class membership. A plurality vote of its neighbors classifies an item, with the object allocated to the class most prevalent among its K nearest neighbors (K is a positive integer, typically small). If $K = 1$, the item is simply assigned to the class of the object's single nearest neighbor[75].

The result of KNN regression is the object's property value. This number is the mean of the values of the K closest neighbors[75].

KNN is a classification method in which the function is only approximated locally and all computation is postponed until the function is evaluated. Because this method depends on distance for classification, normalizing the training data can significantly increase its performance if the features reflect various physical units or arrive at wildly different sizes[76], [77].

An effective strategy for both classification and regression is to apply weights to the contributions of the neighbors, such that the closer neighbors contribute more to the average than the further ones.

The neighbors are chosen from a set of objects whose class or attribute value is known. This may be considered of as the algorithm's training set, but no explicit training is necessary. The KNN technique is unique in that it is sensitive to the local structure of the data[75].

K is a user-defined constant in the classification phase, and an unlabeled vector is categorized by assigning the label that is most common among the K training samples closest to that query point.

Euclidean distance is a popular distance measure for continuous variables. Another measure, such as the overlap metric, can be used for discrete variables, such as text categorization. In the context of gene expression microarray data, for example, KNN has been used as a metric in conjunction with correlation coefficients such as Pearson and Spearman[78]. When the distance metric is learnt using specific methods such as Large Margin Nearest Neighbor analysis, the classification accuracy of KNN may often be greatly improved.

When the class distribution is skewed, the fundamental "majority voting" categorization has a disadvantage. That is, because they are prevalent among the K nearest neighbors, instances of a more frequent class tend to dominate the forecast of the new example. One solution is to weight the categorization, taking into consideration the distance between the test location and each of its K nearest neighbors[79].

2.8.5 Boosting

Boosting is an ensemble modeling strategy that seeks to construct a strong classifier from a collection of weak classifiers[80]. It is accomplished by developing a model in series utilizing weak models[81], [82]. First, a model is constructed using the training data. The second model is then constructed in an attempt to address the faults in the previous model. This approach is repeated until either the whole training data set is properly predicted or the maximum number of models are added[83].

XGBoost and LGBBoost are the most common boosting algorithm.

2.8.5.1 XGBoost

XGBoost stands for Extreme Gradient Boosting. XGBoost is a gradient-boosted decision tree solution optimized for speed and performance[84].

The XGBoost library implements the gradient boosting decision tree algorithm. This algorithm is also known as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines[85].

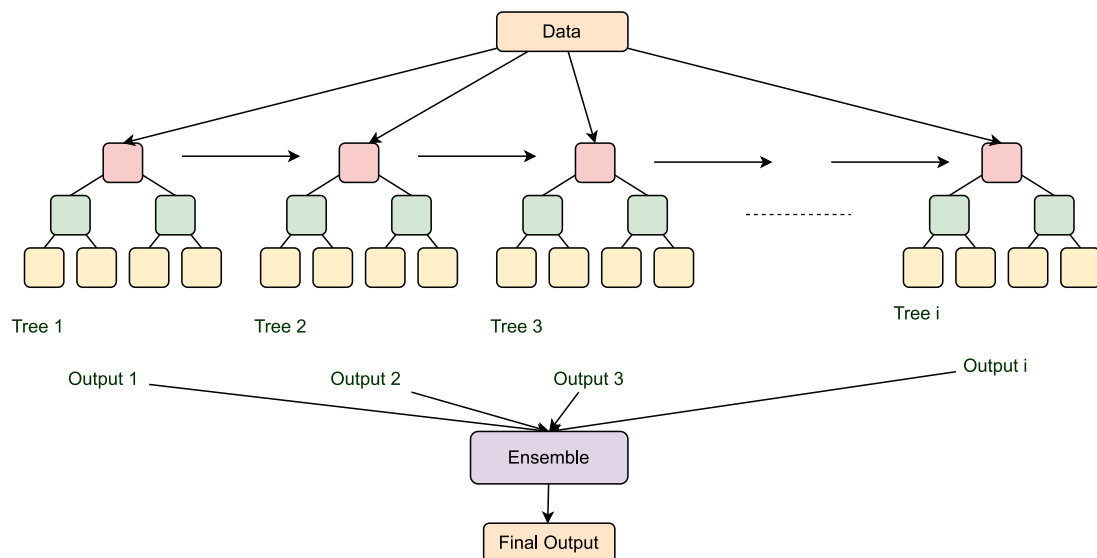


Figure 2.7: Extreme Gradient Boosting

Gradient boosting is a method in which new models are developed that forecast the residuals or mistakes of previous models, which are then combined to form the final prediction. Gradient boosting is so named because it employs a gradient descent approach to minimize loss when adding new models. This method is applicable to both regression and classification predictive modeling issues[85].

2.8.5.2 LGBBoost

LGBBoost(Light Gradient Boosting) is a gradient boosting framework based on decision trees to increase the efficiency of the model and reduce memory usage[86].

It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfill the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks[86].

LGB is quick in model training and uses little memory[87].

2.8.6 Bagging

Bagging is a technique for reducing prediction variance by producing additional data for training from a dataset utilizing combinations with repeats to generate multi-sets of the original data[88].

When the aim is to decrease the variance of a decision tree classifier, bagging is utilized. The goal here is to generate various subsets of data from a training sample selected at random using replacement. Each subset of data is utilized to train their decision trees. As a consequence, we have an ensemble of many models. It is more resilient than a single decision tree classifier to utilize the average of all predictions from different trees[89].

Most commonly used bagging algorithm is Random Forest Algorithm.

Random forest is a Supervised Machine Learning Algorithm commonly used in classification and regression issues. It constructs decision trees from several samples and uses their majority vote for classification and average for regression[90].

One of the most essential characteristics of the Random Forest Algorithm is that it can handle data sets with both continuous and categorical variables, as in regression and classification. It outperforms other algorithms in categorization tasks[90].

2.9 Deep learning

Deep Learning is a branch of machine learning dealing with artificial neural networks that are inspired by the structure and function of the brain[91]. It belongs to a larger family of machine learning approaches that combine artificial neural networks and representation learning[92]. Deep learning can be supervised, semi-supervised, or unsupervised[93].

DL systems assist a computer model in filtering incoming data through layers in order to forecast and categorize information. Deep Learning processes information in the same way that the human brain does. It is utilized in technologies like as driverless automobiles[94]. Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) are two types of DL network architectures.

2.9.1 Convolutional Neural Network

Convolutional neural networks (CNN) are a form of artificial neural network that uses the mathematical operation convolution instead of ordinary matrix multiplication in at least one of its layers[95], [96]. CNN is also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN)[97]. They are employed in image recognition and processing and are especially built to handle pixel data.

CNNs are multilayer perceptrons that have been regularized. Multilayer perceptrons are typically completely connected networks, in which each neuron in one layer is linked to all neurons in the following layer. Because of their "complete connectedness," these networks are prone to data overfitting. Regularization, or preventing overfitting, is commonly accomplished by punishing parameters during training or reducing connectivity. CNNs tackle regularization differently: they use the hierarchical pattern in data to construct patterns of increasing complexity utilizing smaller and simpler patterns imprinted in their filters. As a result, CNNs are at the lowest end of the connectivity and complexity spectrum[98].

When compared to other image classification methods, CNNs require very minimal pre-processing. This implies that, in contrast to traditional methods, the network learns to improve the filters (or kernels) through automatic learning. This freedom from past information and human interference in feature extraction is a significant benefit[98].

An input layer, hidden layers, and an output layer comprise a convolutional neural network. Any intermediary layers in a feed-forward neural network are referred to be hidden because their inputs and outputs are veiled by the activation function and final convolution. The hidden layers of a convolutional neural network include convolutional layers. Typically, this comprises a layer that does a dot product of the convolution kernel and the input matrix of the layer. This is often the Frobenius inner product, and its activation function is typically ReLU. The convolution procedure creates a feature map as the convolution kernel slides along the input matrix for the layer, which then contributes to the input of the following layer. Other layers such as pooling layers, fully linked layers, and normalizing layers follow[98].

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). The picture is abstracted to a feature map, also known as an activation map, after passing through a convolutional layer. Then the shape become (number of inputs) x (feature map height) x (feature map width) x (feature map channels)[98].

Convolutional layers combine input and send the output to the next layer. This is analogous to a neuron's reaction to a specific stimulus in the visual cortex.[9] Each convolutional neuron only processes information for its own receptive field. Although fully linked feedforward neural networks may be used to learn features and categorize data, they are often impracticable for bigger inputs such as high-resolution photos. Due to the vast input size of pictures, where each pixel is an important input characteristic,

it would require a relatively high number of neurons, even in a shallow architecture. Using regularized weights across fewer parameters eliminates the vanishing gradients and expanding gradients difficulties encountered in classic neural networks during backpropagation[99], [100].Furthermore, because spatial interactions between individual features are taken into consideration during convolution and/or pooling, convolutional neural networks are appropriate for data having a grid-like architecture.

Along with standard convolutional layers, convolutional networks may add local or global pooling layers. By merging the outputs of neuron clusters at one layer into a single neuron in the following layer, pooling layers minimize the dimensionality of data[101], [102].Pooling may be divided into two types: maximum and average. The maximum value of each local cluster of neurons in the feature map is used in max pooling, whereas the average value is used in average pooling[103].

Every neuron in one layer is connected to every neuron in the next layer via fully connected layers. It functions similarly to a standard multilayer perceptron neural network (MLP). To identify the photos, the flattened matrix is passed through a fully linked layer[98].

Each neuron in a neural network receives input from a number of sites in the preceding layer. Each neuron in a convolutional layer gets information from just a small portion of the preceding layer, known as the neuron's receptive field. The area is typically square. Of contrast, the receptive field in a completely linked layer is the whole prior layer. As a result, each neuron in each convolutional layer accepts information from a greater region in the input than prior levels. This is due to the convolution being applied repeatedly, which takes into account the value of a pixel as well as its surrounding pixels. When combining the impact of numerous layers, the number of pixels in the receptive field remains constant, but the field becomes increasingly sparsely filled as its dimensions rise[98].

Each neuron in a neural network generates an output value by applying a specified function to the input values received from the preceding layer's receptive field. A vector of weights and a bias decide the function that is applied to the input data. Iteratively modifying these biases and weights is what learning is all about[98].

Filters are vectors of weights and biases that reflect certain aspects of the input. CNNs are distinguished by the fact that several neurons can share the same filter. Because a single bias and a single vector of weights are utilized across all receptive fields that share that filter, the memory footprint is reduced, as opposed to each receptive field having its own bias and vector weighting[104].

2.9.2 Recurrent Neural Network

A recurrent neural network (RNN) is a type of artificial neural network in which connections between nodes create a graph along a temporal sequence. This enables it to display temporal dynamic behavior. RNNs, which are derived from feedforward neural networks, can handle variable length sequences of inputs using their internal state (memory)[105], [106].As a result, they may be used for tasks like unsegmented, linked handwriting recognition or speech recognition[107], [108].Recurrent neural networks are Turing complete in theory and may execute arbitrary algorithms to handle arbitrary sequences of inputs[109].

The phrase "recurrent neural network" refers to a kind of network with an infinite impulse response, whereas "convolutional neural network" refers to a type of network with a limited impulse response. Both types of networks display temporal dynamics[110].An infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled and substituted with a strictly feedforward neural network, whereas a finite impulse recurrent network can be unrolled.

Most commonly used recurrent neural networks are LSTM and GRU.

2.9.2.1 LSTM :

Long short-term memory (LSTM) is a deep learning system that overcomes the vanishing gradient problem[111].LSTM is typically supplemented by recurring gates known as "forget gates." LSTM keeps backpropagated mistakes from disappearing or exploding.

Unlike traditional feedforward neural networks, LSTM has feedback connections. A recurrent neural network of this type may analyze not just single input points (such as photos), but also complete data sequences (such as speech or video)[112].

A cell, an input gate, an output gate, and a forget gate comprise a typical LSTM unit. The cell stores values for arbitrary time intervals, and the three gates control the flow of information into and out of the cell[113], [114].

At its most fundamental, the output of an LSTM at any given time is determined by three factors[111].

1. The current long-term memory of the network (cell state).
2. The output at the previous point in time (previous hidden state).
3. The input data at the current time step.

LSTMs employ a number of 'gates' that regulate how information in a data sequence enters, is stored in, and exits the network. A typical LSTM contains three gates.

1. Forget gate
2. Input gate
3. Output gate

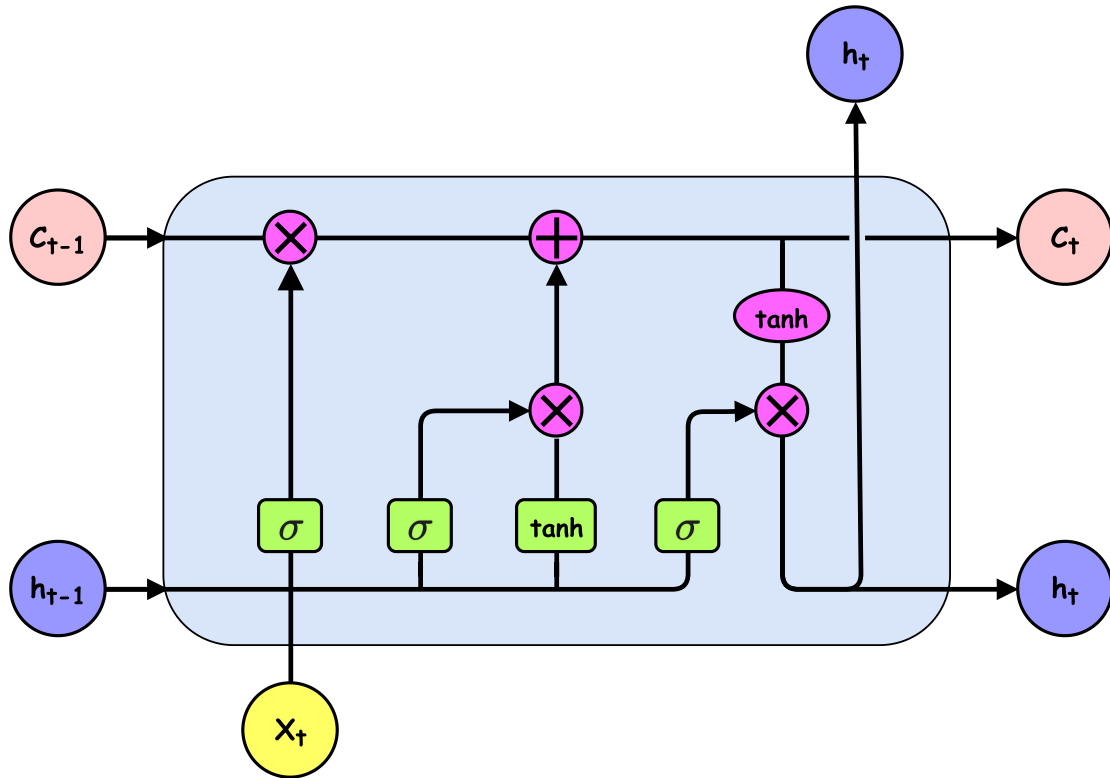


Figure 2.8: LSTM Cell

These gates function as filters and each have their own neural network[111].

The equations for the gates in LSTM are:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

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

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

The equations for the cell state and candidate cell state are :

$$\tilde{c}_t = \tanh(w_c[h_{t-1}, x_t] + b_c)$$

$$c_t = f_t \times c_{t-1} + i_t \times \tilde{c}_t$$

And the equation for final output of LSTM is :

$$h_t = o_t \times \tanh(c_t)$$

Here :

i_t = input gate

f_t = forget gate

o_t = output gate

σ = sigmoid function

w_x = weight for the respective gate(x) neurons

h_{t-1} = output of the previous LSTM block

x_t = input at current time stamp

b_x = biases for the respective gates(x)

c_t = cell state at timestamp(t)

\tilde{c}_t = candidate for cell state at timestamp(t)

Because there might be delays of undetermined duration between critical occurrences in a time series, LSTM networks are well-suited to categorizing, processing, and generating predictions based on time series data. LSTMs were created to address the vanishing gradient problem that can occur when training regular RNNs. LSTM has an advantage over other RNNs in that it is relatively insensitive to gap length[115].

2.9.2.2 GRU

A gated recurrent unit (GRU) is a type of recurrent neural network that uses connections across a series of nodes to accomplish machine learning tasks related to memory and grouping. GRUs help in the modification of neural network input weights to address the vanishing gradient problem, which is a common difficulty with recurrent neural networks[116]. The GRU is similar to an LSTM with a forget gate [117], but it has lower parameters since it misses an output gate[118]. So, it has two gates.

1. Update gate
2. Reset gate

The update gate assists the model in determining how much earlier knowledge (from prior time steps) must be passed on to the future. This is extremely powerful since the model may choose to copy all of the information from the past, therefore eliminating the threat of the fading gradient problem[119].

A reset gate determines how much prior data to discard. The reset gate is similar to the LSTM Forget gate in that it identifies irrelevant data and instructs the model to forget it and proceed without it[119].

The equations for the update gate and reset gate are:

$$r_t = \sigma(x_t \times u_r + h_{t-1} \times w_r)$$

$$u_t = \sigma(x_t \times u_u + h_{t-1} \times w_u)$$

Here :

r_t = reset gate

u_t = update gate

σ = sigmoid function

x_t = input at current time stamp

h_{t-1} = output of the previous GRU block

u_x = weight for the respective gate(x) of input

w_x = weight for the respective gate(x) of output

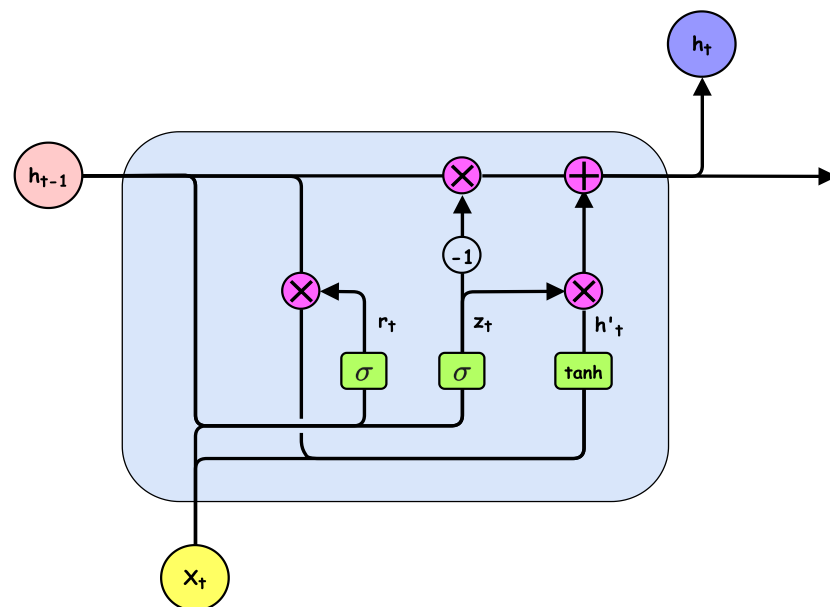


Figure 2.9: GRU Cell

GRU is superior to LSTM since it is easier to alter and does not require memory units. As a result, it is quicker than LSTM and provides results based on performance.

Chapter 3

METHODOLOGY

3.1 Data Collection

The first and foremost objective of the procedure is to collect data. The raw data has been divided into three parts which are load demand data, weather data and date time, holiday data. To acquire better load forecasting output, the parts of the raw data has been merged together since forecasting requires the impact of the weather as well as holiday.

The load demand data has been collected from Power Grid Company Bangladesh from the year 2015 to 2021. This is a univariate data which consists of missing values and outliers. These are later on preprocessed in order to have an accurate forecasting summary. The errors of the dataset are may be of any reasons, one of the reasons might be the data entry problem.

The weather data has been selected from rp5 database. From there specific parameters were chosen which might improve the forecasting. The parameters which are used in case of the dataset are temperature, pressure, relative humidity, mean wind direction and special weather phenomena.

From the literature review we understood the importance of the holiday in case of load forecasting. The holiday has been included in our dataset for the better output. The common festivals like Eid vacation, Puja vacation, Christmas, Pahela Baishak etc. were collected. Other than these fests, government holiday were also significant which are collected from the government calendar of the Bangladesh.

3.2 Data Pre-Processing:

As discussed in the raw data section, there were consists of some missing value and outliers. It needs to be corrected for the best results.

3.2.1 Missing Value Imputation

In the load demand data set, for the specific date there were some missing data which need correction. The imputation of the missing value was done taking the average of all the load demands of that missing day of the whole month. Suppose one of the Sunday of the month April got a missing data. We imputed that missing day by averaging the

demands of the whole month. From the Figure 3.1 the missing values are observed which are significant in number. The blue line denotes the demands in MW unit and the red line denotes the missing value of the dataset from Jul 2015 to Oct 2021.

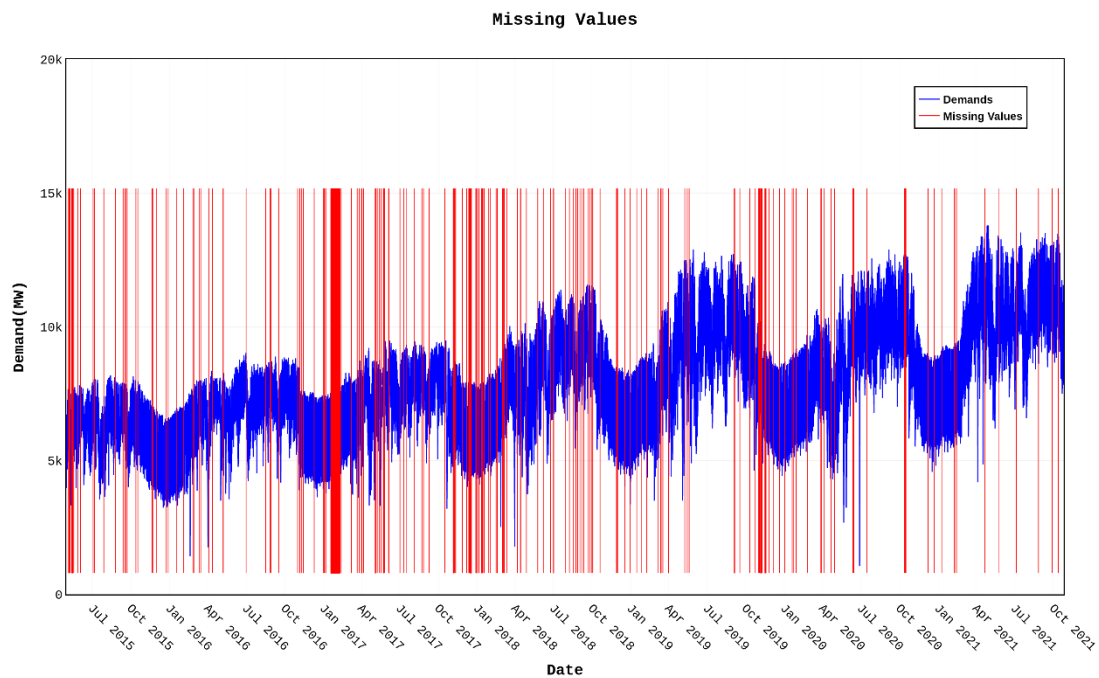


Figure 3.1: Missing Values in Dataset

3.2.2 Outliers

An outlier is a value in a randomized sampling from a population that is significantly different from another values. In other ways, this approach refers to the researcher to determine what is abnormal. An outlier is a data point that is exceptionally high or significantly lower in comparison to the adjacent data point and the rest of the nearby surviving values in a dataset.

In our dataset section of the load demand, as discussed there were some of the outliers. For example: one of the day the load demand in the database showing 80,000 MW but PGCB maximum demand that can be achieved is 14,000 MW. This type of wrong data entry results in formation of many outliers.

3.3 Exploratory Data Analysis

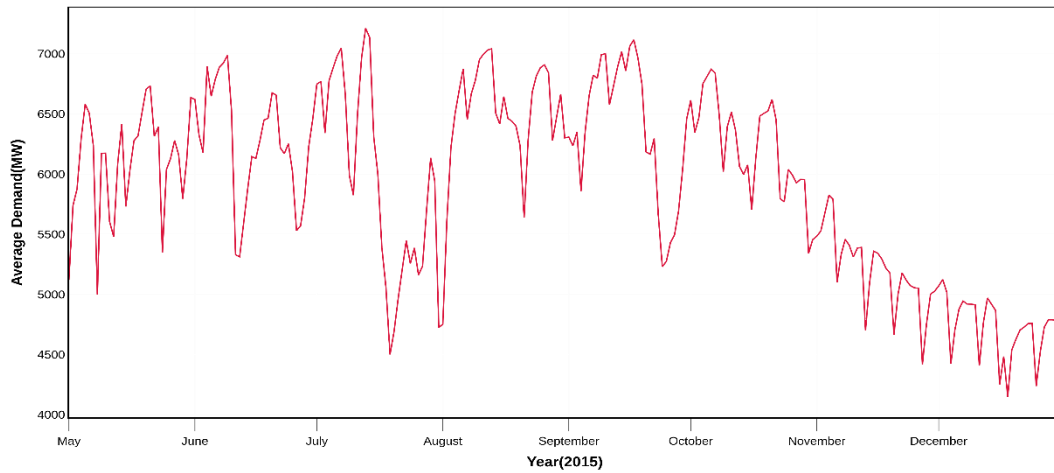


Figure 3.2: Daily Average Demand of the Year 2015

The average demand of the year 2015 is shown where there is low amount of demand in the month of October to December due to weather conditions which is winter in Bangladesh. In the month of May to June the demand is very high.

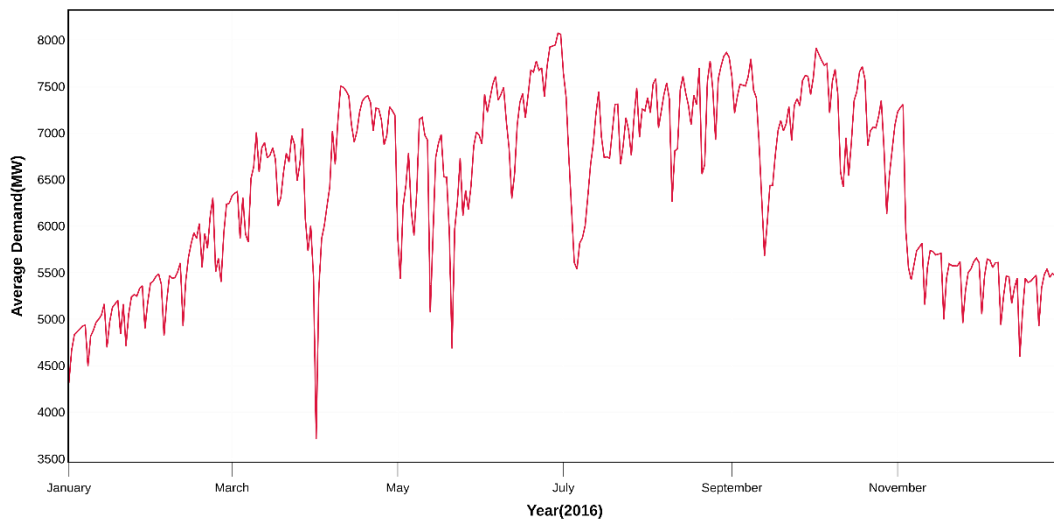


Figure 3.3: Daily Average Demand of the Year 2016

The average demand of the year 2016 is plotted where the demand is low in the month of November to December. In the month of May to June the demand is very high. There is significant change in the month of April which suddenly decreases.

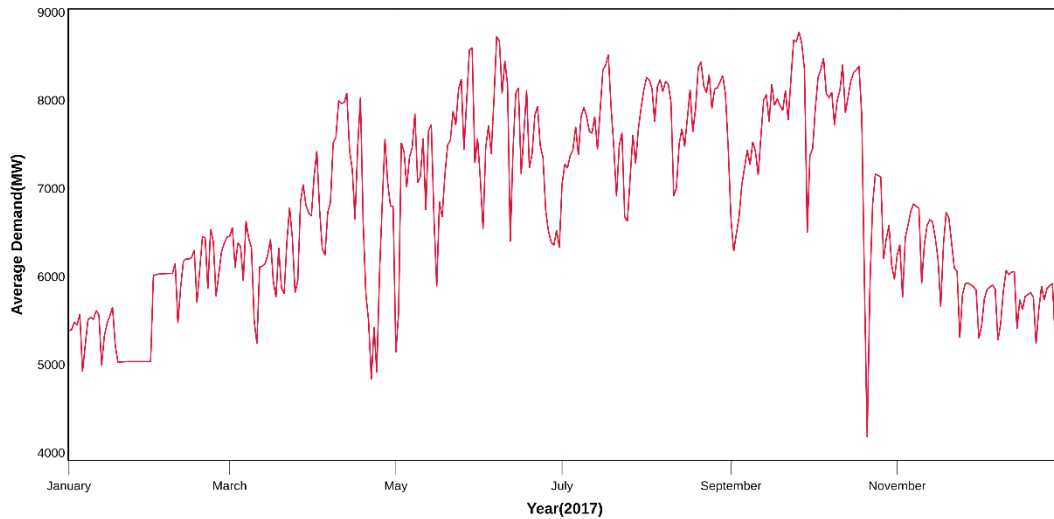


Figure 3.4: Daily Average Demand of the Year 2017

The average demand of the year 2017 is displayed here in which the demand in the late October is very low in MW whereas it is almost really high for the whole June month.

The average demand of the year 2018 is seen in which it shows slowly increasing condition from the beginning of the year and recorded almost 10K MW in the month of September. Later at the end of the year the demand drops in a very rush manner.

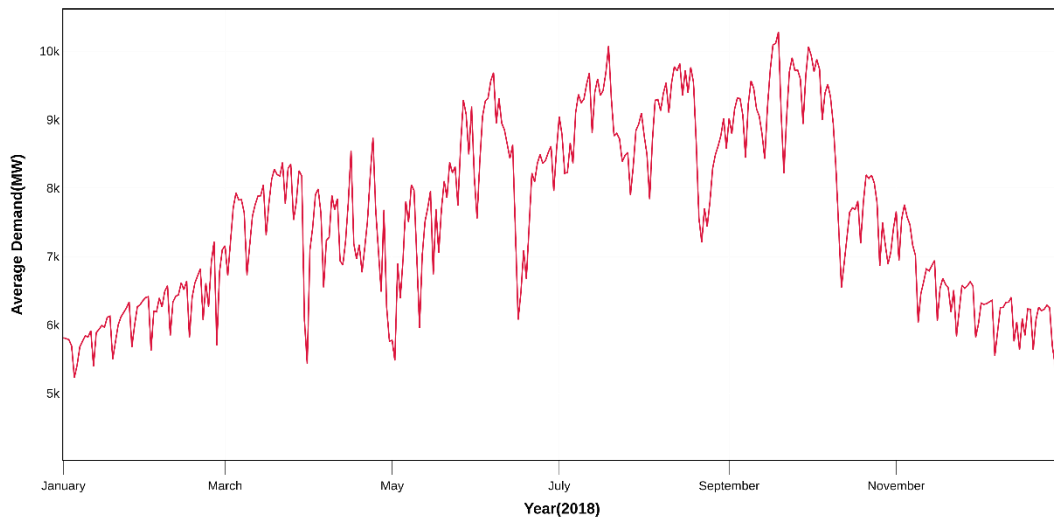


Figure 3.5: Daily Average Demand of the Year 2018

Increasing from the beginning to the drop of demand in the May month describes the average demand of the year 2019 whereas the last part of the year the demand is not that much

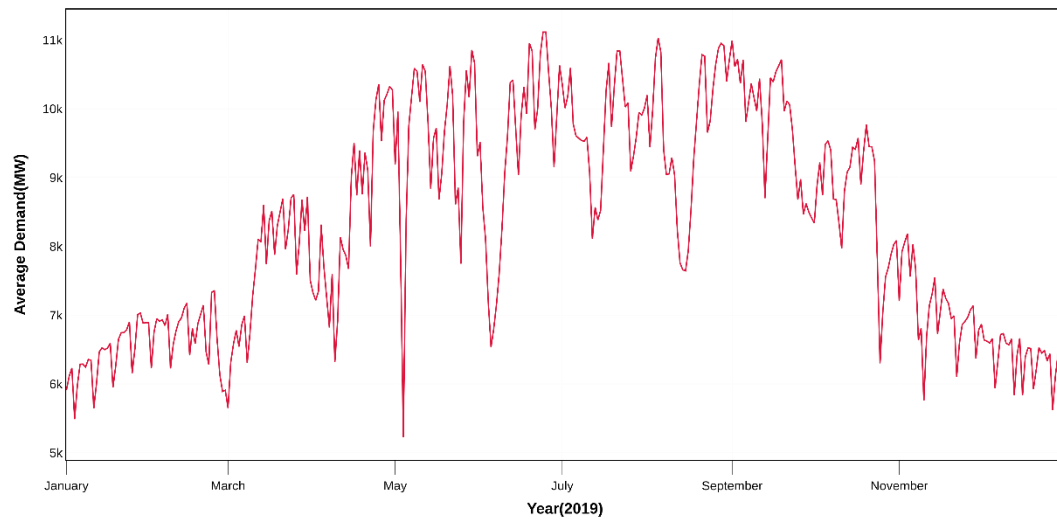


Figure 3.6: Daily Average Demand of the Year 2019

The beginning and the end of the year the average demand remain same for the year 2020. For maximum part of the year due to corona the demand were in same amount.

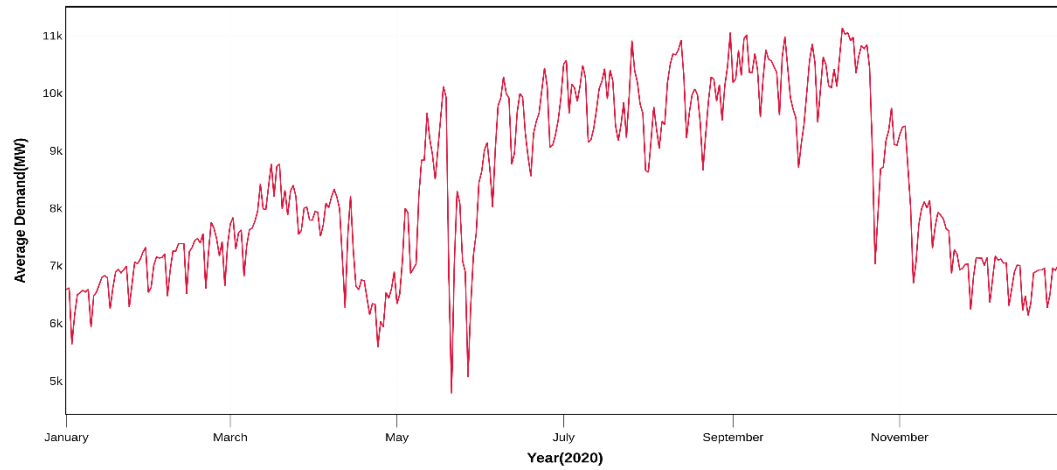


Figure 3.7: Daily Average Demand of the Year 2020

It is the part of the year 2021 which for specific time of the month it remains constant due to the lockdown and it does not deviate much.

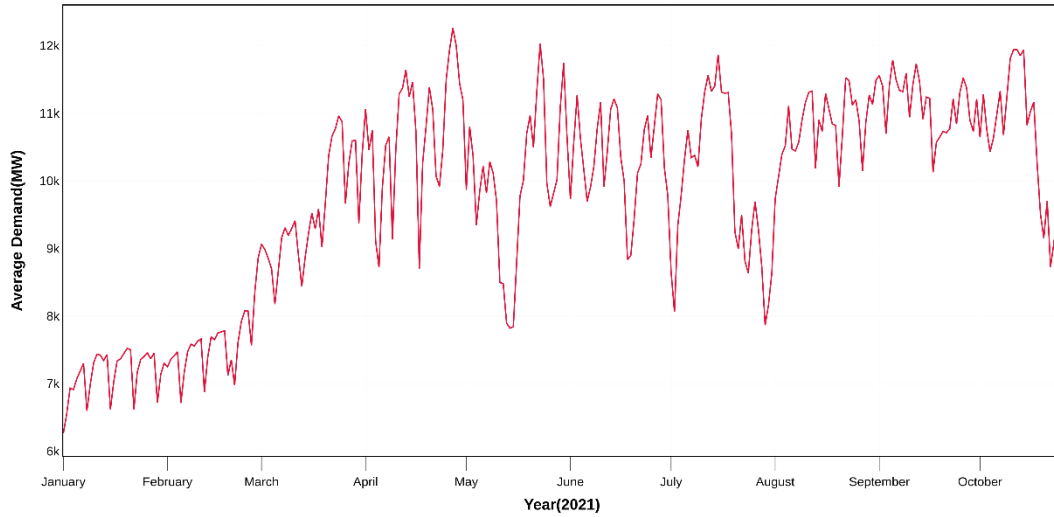


Figure 3.8: Daily Average Demand of the Year 2021

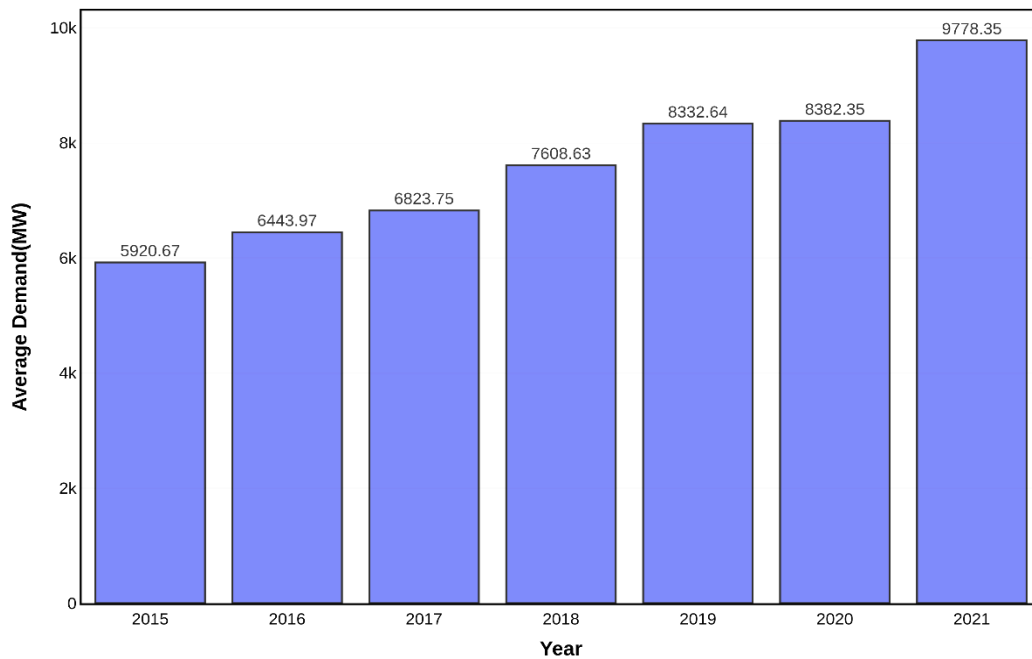


Figure 3.9: Yearly Average Demand Bar Chart

The average demand in MW for each of the year is displayed in Figure 3.9. Starting from the year 2015 the gradual increase of demand every year has been shown. There is an exception in the year of 2020 since the change is not that much visible to our eyes as it supposed to be. The change is really negligible and the reason behind it is the Corona disease outbreak which leads to the full lockdown of the whole country. Due to sudden opening after the lockdown and in some area partial lockdown in the year 2021 the demand again increases at a very high rate.

3.3.1 Weather Data Merge

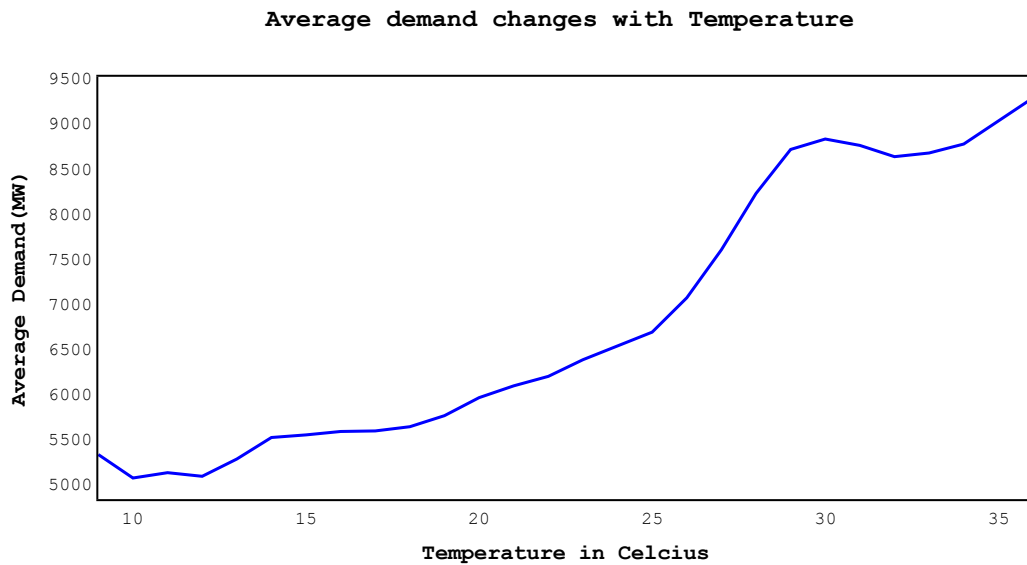


Figure 3.10: Temperature Vs Average Demand (MW) Curve

In winter we need less electrical appliances which leads us to the fact that the lowest the temperature the lowest the demand is. As observed from the Figure 3.10 when the temperature is at around 10-15 degree Celsius the load demand is very low, but with the increase of the temperature the average demands also increase at a high rate. The temperature change is seen significant at the temperature of 27-30 degree Celsius where the load demand is very high which is around 8.5k-9k MW.

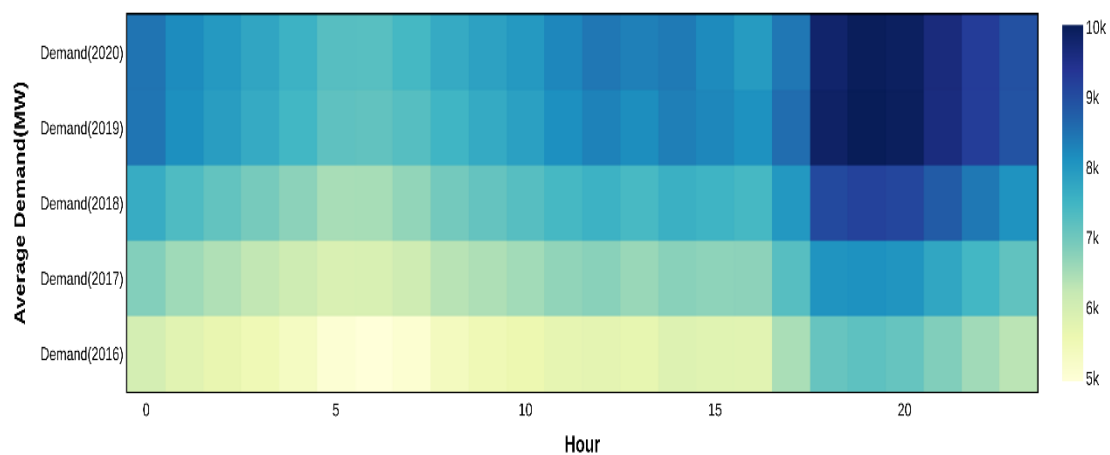


Figure 3.11: Heatmap of Average Demand of Each Hour of the Day

The heatmap in Figure 3.11 of the average demand with respect to hour which shows that the maximum demand in every year is around at the evening to the night of the day. The demand in the daytime is not that much observed.

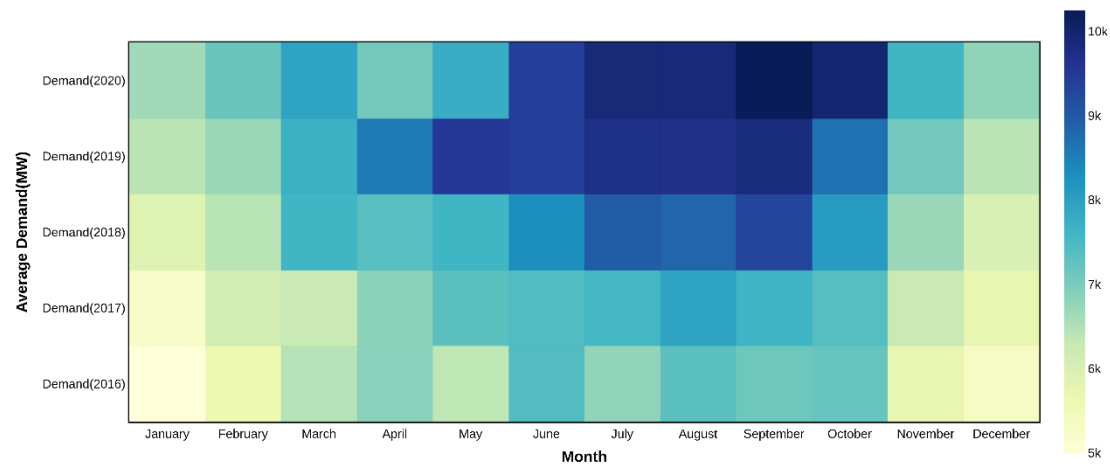


Figure 3.12: Heatmap of Average Demand of Each Month of the Year

With respect to the month the heatmap shown in Figure 3.12 is observed which shows the highest demand in the time of June to September as this month are in the summer time of our country which results in high load demands. During the winter period like November to February the demand is not that much as the people use less electrical devices.

3.4 Data Scaling

The data has been scaled using standard scaling. The scaling feature columns to range about the mean 0 and the standard deviation is 1. The scaling of data in deep learning will help to make a model learn and better understanding the problem. Due to this, the training is faster and also help to prevent the optimization from getting stuck in local optima.

$$X_{\text{Scaled}} = \frac{X_i - \text{mean}(X)}{\text{std}(X)}$$

Here

X_{Scaled} = Scaled Value of X

X_i = i^{th} value of X

$\text{mean}(X)$ = Mean value of X

$\text{std}(X)$ = Standard Deviation of X

3.5 Data windowing

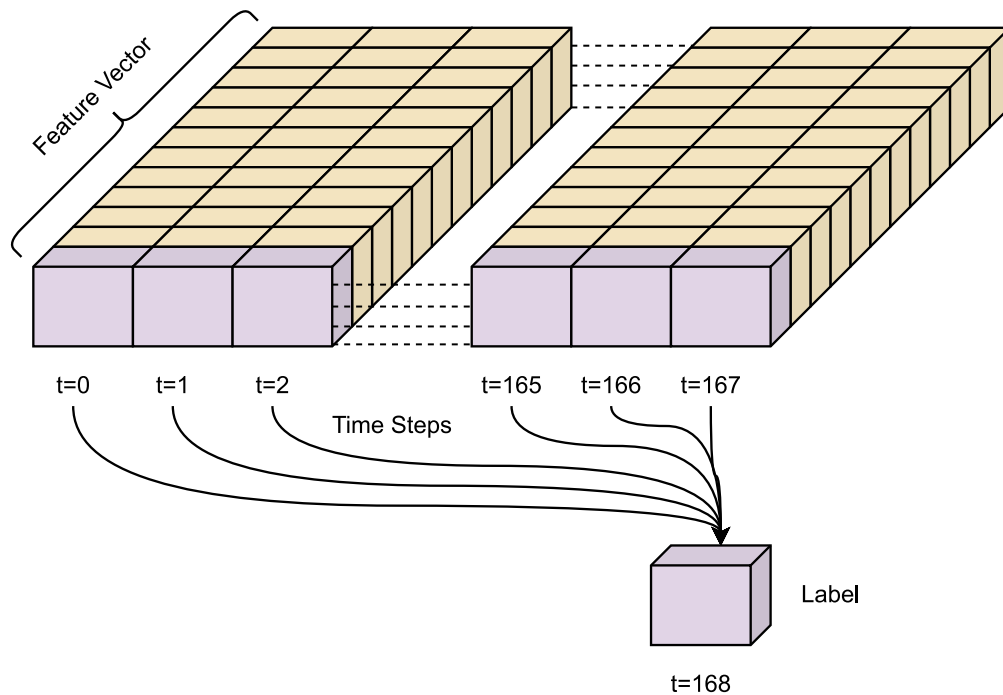


Figure 3.13: Data Windowing

The data has been framed before putting it into the convolutional layer. To forecast the demand for future time steps, current and previous 168 data samples with their associated attributes are used as 2D input. In the figure, the time steps are taken as input, with 12 features in each time step, and the output label is shown.

3.6 Train test validation

A model should not over-learn from training data and then perform poorly in production. There is a way for determining how effectively a model generalizes. To prevent overfitting and successfully assess a model, segregate the input data into training, validation, and testing subsets.

The Train-Valid-Test split is a strategy for evaluating the performance of a model, whether it is for classification or regression. There must be a specified dataset that has been divided into three subgroups.

Training Set	85% (1 st May,2015 to 28 th October,2020)
Validation Set	7% (29 th October,2020 to 12 th April,2021)
Testing Set	8% (13 th April,2021 to 25 th October,2021)

3.6.1 Training Set

The data sample used to fit the model, which is the actual subset of the dataset utilized to train the model. The model watches, learns, and optimizes its parameters based on this data. In our approached model it is considered from the time of May 1,2015 to October 28,2020 which is about 85% of the datasets.

3.6.2 Validation Set

The validation set is a different collection of data from the training set that is used to validate the performance of our model during training. This validation method provides data that allows us to fine-tune the model's input variables and settings. Our proposed model got a validation set of about 7% which starts from the October 29,2020 to the month of April 12,2021.

3.6.3 Testing Set

The sample of data used to offer an unbiased evaluation of a final model fit on the training dataset. It is only utilized once the model has been thoroughly trained using the training and validation sets. As a result, the test set is used to simulate the sort of circumstance that would be experienced after the model is deployed for real-time use. The testing set for the model we proposed is 8% starting from day 13 of April, 2021. It ends in 25th October,2021 which helps to give us the predicted results.

3.7 CNN Model

In Figure 3.14, data windowing leads to give the merged data a shape of samples,168,12 where 168 is the time steps and 12 is the corresponding feature vector of each time step which then follows the path to the convolutional layer with a filter size of 64. The max-pooling layer mainly down sampled the feature set which then goes to the flatten layer which is used to make the feature set one dimensional. Dense layers are used for forecasting the output. First a hidden dense layer of size 8 is used with ReLU activation function. It has been used which helps not to activate all the neurons at the same time. As few of the neurons are triggered, thus in case of computational efficiency it is very good than the other activation functions. In case of output dense layer of size 1 with the activation function named linear to extract the predicted results.

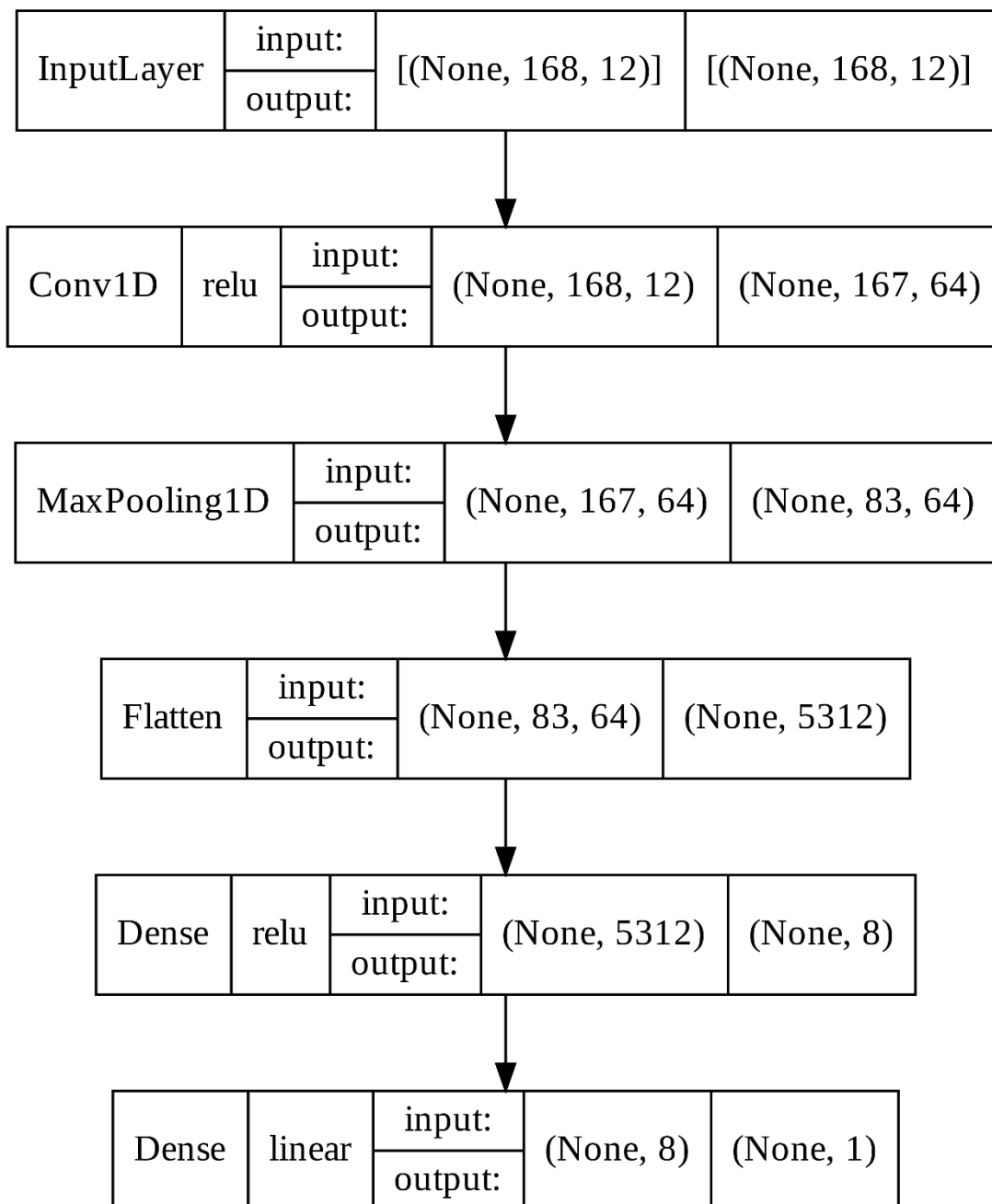


Figure 3.14: Architecture of CNN Model

3.8 LSTM Model

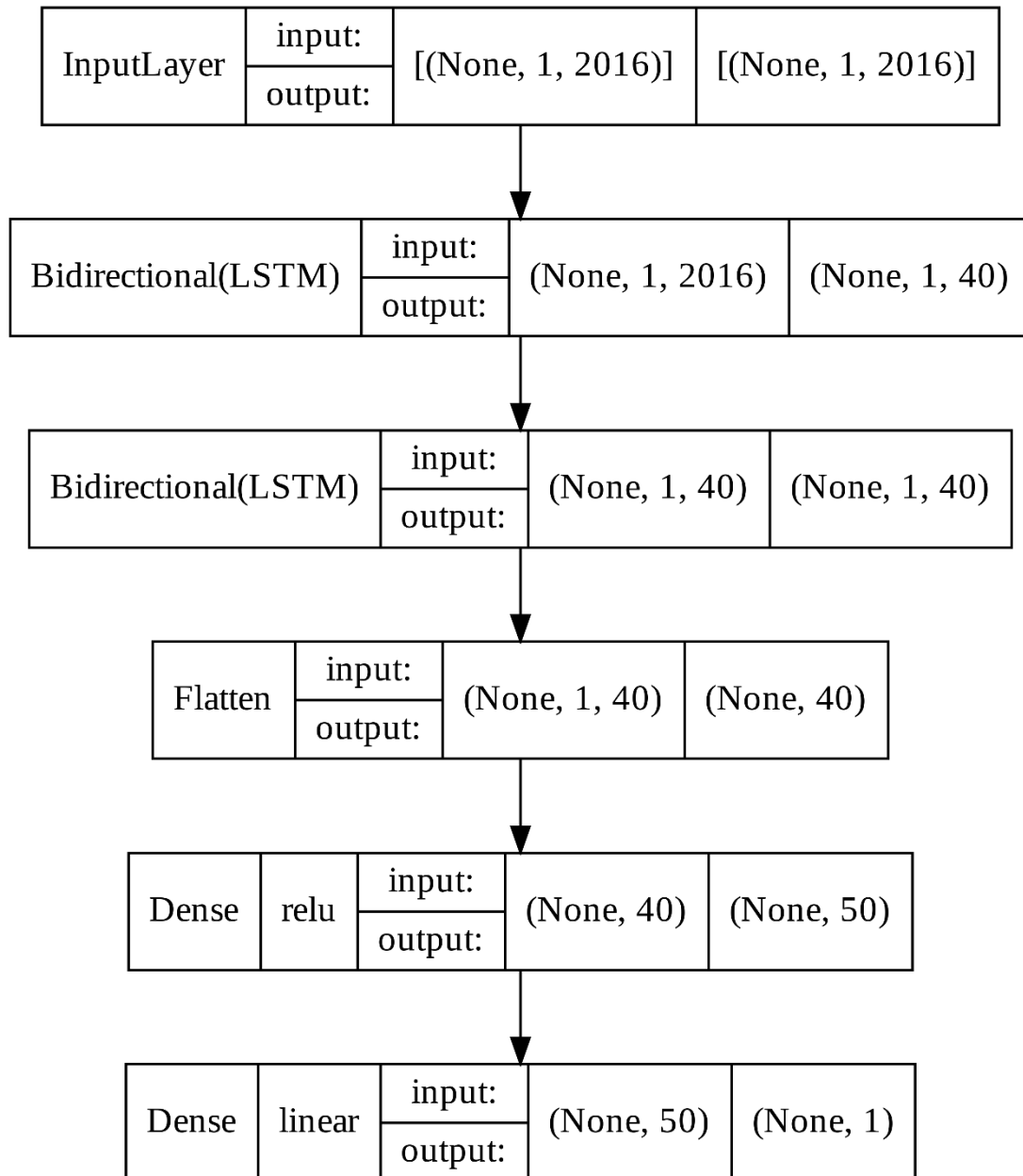


Figure 3.15: Architecture of LSTM Model

From Figure 3.15, the combined data which is shape of samples,168,12 is reshaped into samples,1,2016 where 1 is the time steps and 2016 is feature set to feed into the first Bi-LSTM layer. First LSTM layer has 20 units with Tanh activation function which leads to second Bi-LSTM layer of units 20. The flatten layer is utilized to reduce the feature set to one dimension. Forecasting output is done using dense layers. A hidden dense layer of size 50 is utilized first, followed by the ReLU activation function. It has

been used to avoid activating all of the neurons at once. Because just a small number of neurons are activated, it has a higher computational efficiency than the other activation functions. In the case of an output dense layer of size 1 with the activation function linear, the expected results are extracted.

3.9 GRU Model

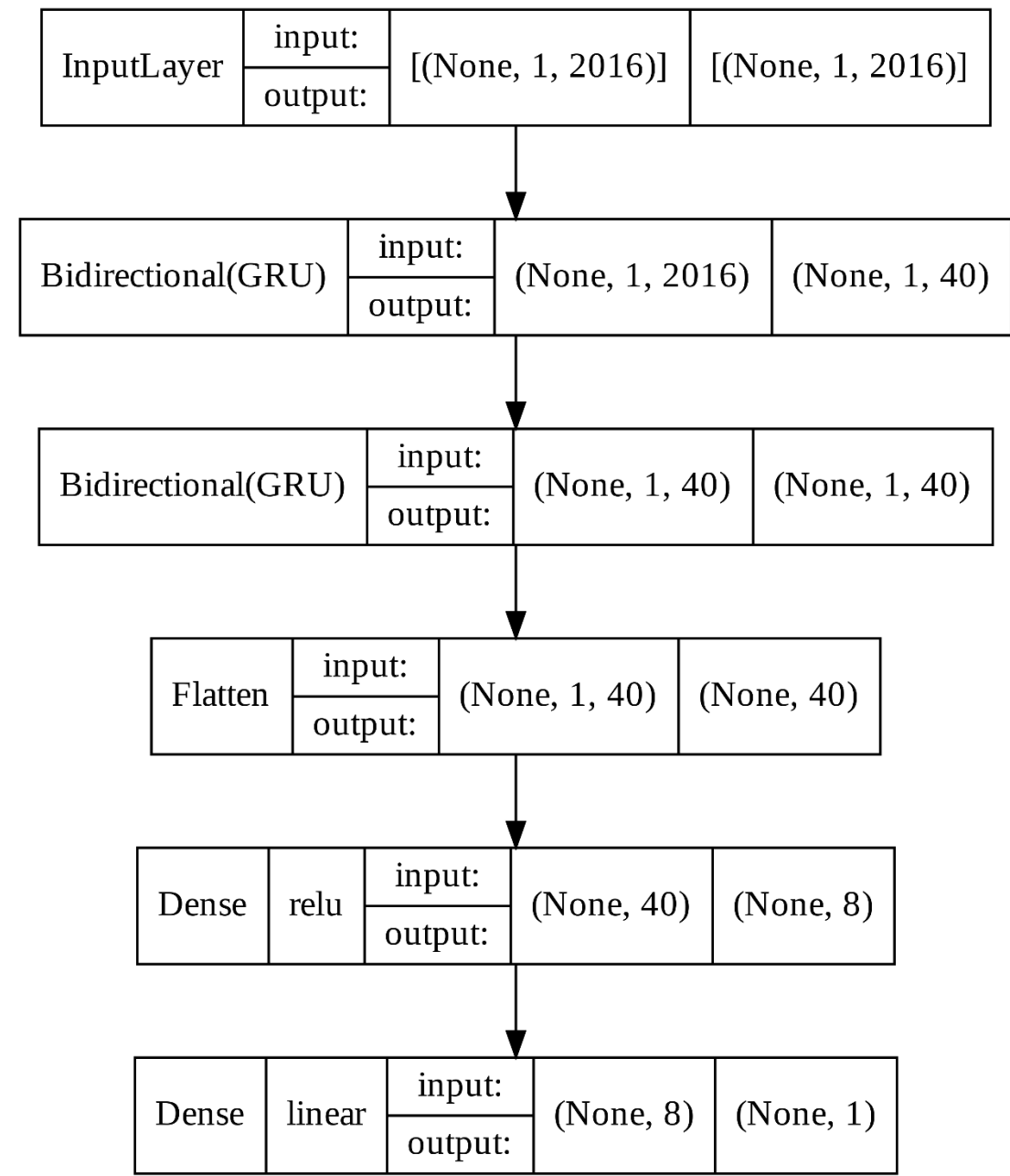


Figure 3.16: Architecture of GRU Model

The aggregated data shown in Figure 3.16, which is in the shape of samples,168,12, is reshaped into samples,1,2016, where 1 represents the time steps and 2016 represents

the feature set to input into the first Bi-GRU layer. The first GRU layer comprises 20 units with Tanh activation, which leads to the second Bi-GRU layer of 20 units. The flatten layer is used to condense the feature set to a single dimension. Dense layers are used to forecast production. The ReLU activation function is used initially, followed by a hidden dense layer of size 8. It has been used to prevent stimulating all neurons simultaneously. Because it only activates a limited number of neurons, it has a better computational efficiency than the other activation functions. The predicted results are retrieved for an output dense layer of size 1 with a linear activation function.

3.10 Proposed Model (CNN-BiLSTM)

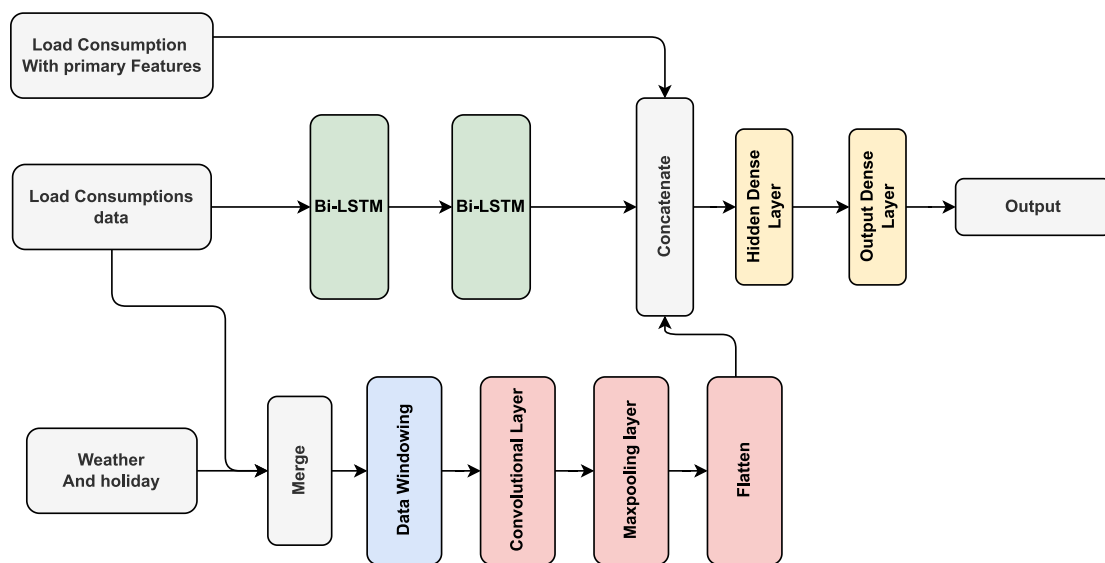


Figure 3.17: Block Diagram of Proposed Model

Figure 3.17 is a graphical representation of our whole suggested model. Only the load demand data, with suitable data framing and time steps, is gathered and fed into the bidirectional LSTM layers first. Long short-term memory (LSTM) layers are utilized to extract temporal information from the load demand sequence. It is capable of learning the sequence trend and how the data evolves over time. Bidirectional learning is used to extract more hidden trend and temporal patterns in data by learning the sequence from both ends forward and backward. The load demand and weather data are then combined. And the data is sent into the convolutional layer using correct data windowing. Among the major characteristics, this CNN or convolutional layer extracts spatial information and hidden patterns. The retrieved LSTM and CNN layers are then concatenated with the main features. With thick layers, these feature sets are then utilized to estimate future load needs.

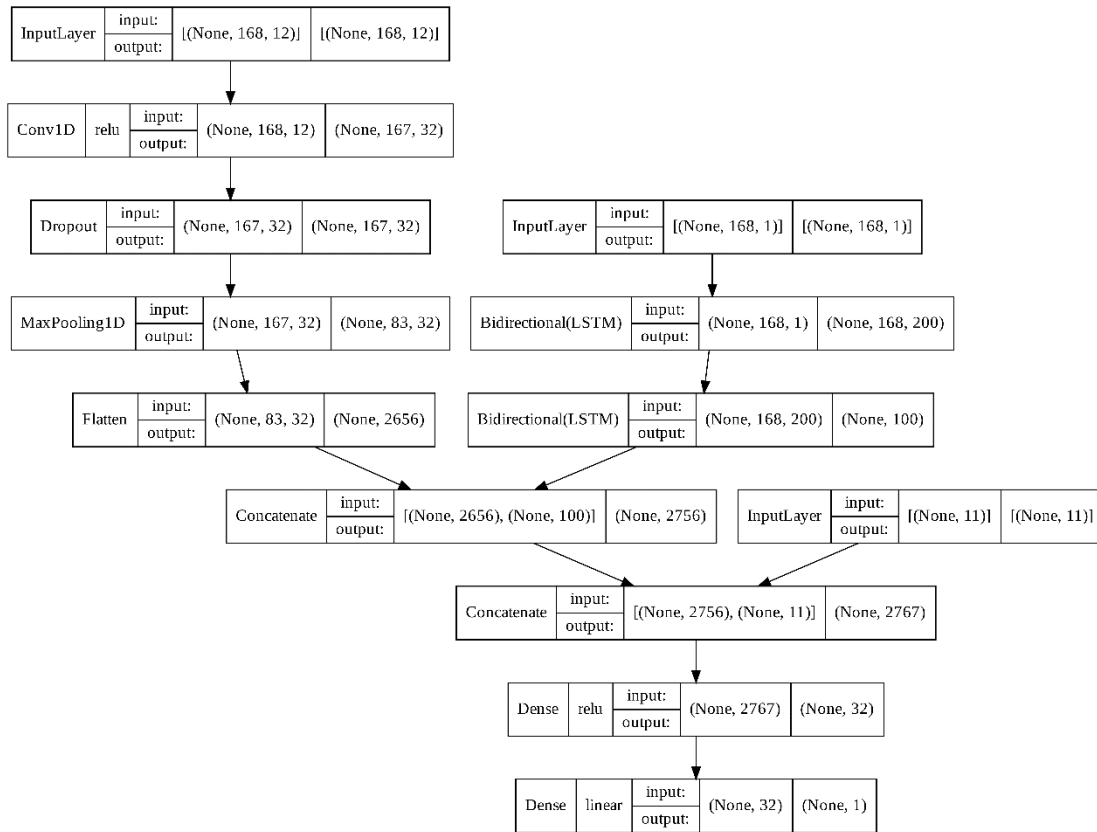


Figure 3.18: Architecture of Proposed Model

In Figure 3.18 the combined data, after passing through data windowing, yields the shape with 168,12, where 168 is the number of time steps and the 12 represents the characteristics of each time step. This data is then sent into the convolutional layer, which has a filter size of 32 and a kernel size of 2. Then, to reduce overfitting issues, a dropout layer is applied. The feature set is then down sampled using the max pooling layer, and the flatten layer is used to make it one dimensional so that it may be concatenated with the output features of the lstm layers. The first lstm layer has 100 units. It outputs 200 characteristics when bidirectional. The last characteristics are extracted using another LSTM layer. After concatenating the data, the outputs are predicted using two dense layers, one of which is a hidden layer with a size of 32.

Chapter 4

Performance and Result Analysis

4.1 Evaluation metrics

To evaluate the models' performance five different evaluation metrics are used. Then the results of different models are compared and analyzed based on these metrics. These five evaluation metrics are mean squared error (MSE), root mean squared error (RMSE), R-squared score (R2-Score), mean absolute percentage error (MAPE), and accuracy.

4.1.1 Mean Squared Error (MSE)

MSE is a very simple and most used evaluation metric. It means the average squared difference between actual and predicted values. It is calculated by the sum of squares of errors in prediction divided by the total number of data points.

$$MSE = \frac{1}{N} \sum_{i=1}^N (A_i - P_i)^2$$

In the above equation A_i is the i^{th} actual value, P_i is the i^{th} predicted value and N is the total number of data points.

4.1.2 Root Mean Squared Error

RMSE is the square root of MSE. It is another most commonly used evaluation metric to evaluate the predicted results. RMSE scores are in the same unit of actual data which make the interpretation easier.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (A_i - P_i)^2}$$

In the above equation A_i is the i^{th} actual value, P_i is the i^{th} predicted value and N is the total number of data points.

4.1.3 R-squared Score

R-squared score is a statistical measurement of the closeness of the data points to fitted regression line. It indicates the proportion of explained variability of dependent variable

by the independent variables. It is calculated by the explained variation divided by the total variation.

$$R - squared = \frac{SSR}{SST}$$

Here SSR means sum of squares due to regression and SST means the total sum of squares.

4.1.4 Mean absolute percentage error (MAPE)

It is a very simple evaluation metric and easy to interpret. It means the percentage of deviation of predicted results from actual values. It is calculated by averaging the absolute forecasted errors.

$$MAPE = \frac{1}{N} \sum_{i=1}^N \frac{(A_i - P_i)}{A_i}$$

Here A_i is the i^{th} actual value, P_i is the i^{th} predicted value and N is the total number of data points.

4.1.5 Accuracy

It explains how correctly the model can predict the values. It can be calculated by simply subtract the MAPE from 100%.

$$Accuracy = 100\% - MAPE(\%)$$

4.2 Results analysis of different models:

4.2.1 CNN model

The test data is used to evaluate the CNN model. The whole test set is split based on the month. Then the split test set is used to forecast the hourly energy demand.

Table 4.1: Evaluation Metrics Score of CNN Model of individual Months

<i>Month Name</i>	<i>RMSE</i>	<i>MSE</i>	<i>R-Squared</i>	<i>MAPE (%)</i>	<i>Accuracy (%)</i>
July	285.92	81751.98	94.06	2.31	97.69
August	253.07	64048.03	95.01	1.83	98.17
September	278.97	77829.75	93.74	1.86	98.14
October	290.24	84244.28	95.77	2.18	97.82

In Table 4.1 the evaluated results are shown. Here, the best RMSE and MSE score is in August month although the best R-squared score is got for October month. Lowest MAPE is 1.83% and maximum accuracy is 98.17 both for August month.

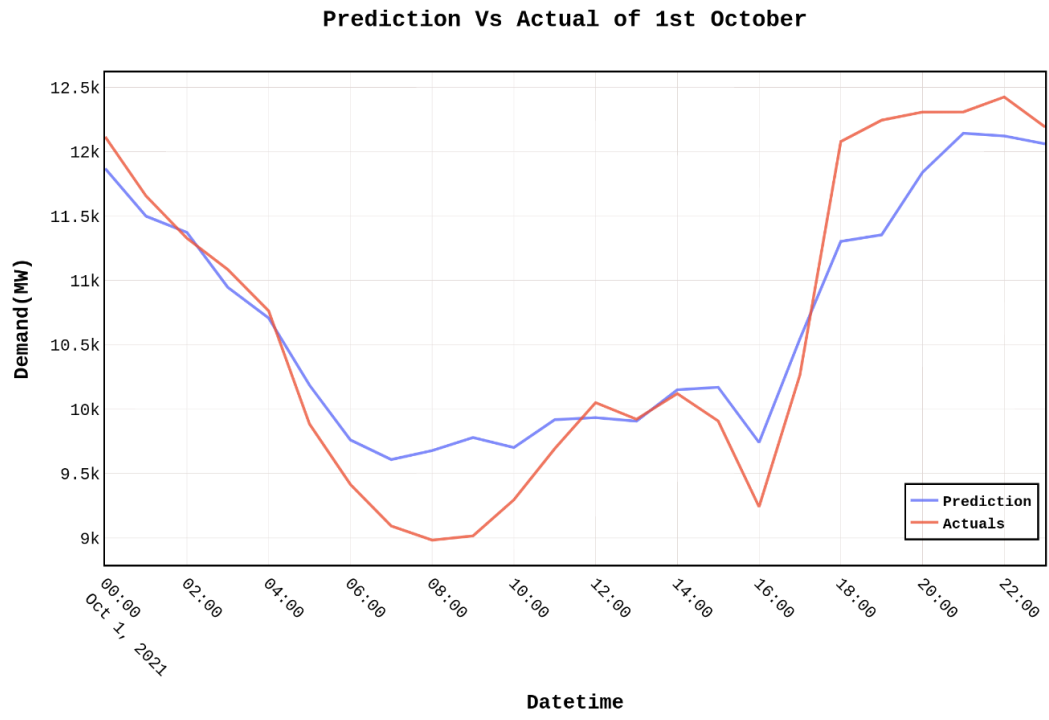


Figure 4.1: Prediction Vs Actual plot of 1st October of CNN Model

In Figure 4.1, the actual demands and the predicted demands are shown for 1st October, 2021. From the figure it is visible that prediction is not accurate from 6.00 am to 11.00 am and 6.00 pm to 11.00 pm.

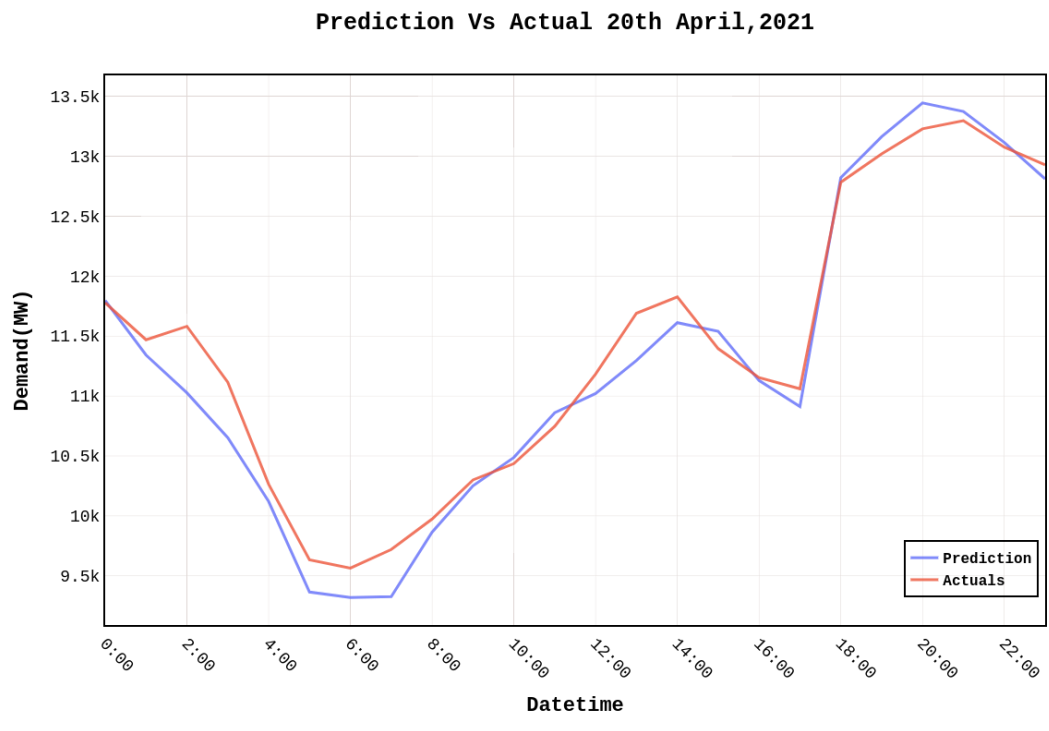


Figure 4.2: Prediction Vs Actual plot of 20th April,2021 of CNN Model

In Figure 4.2, another day is shown to further visualization of predicted results. But for this day the prediction is almost accurate compare to previous Figure 4.1

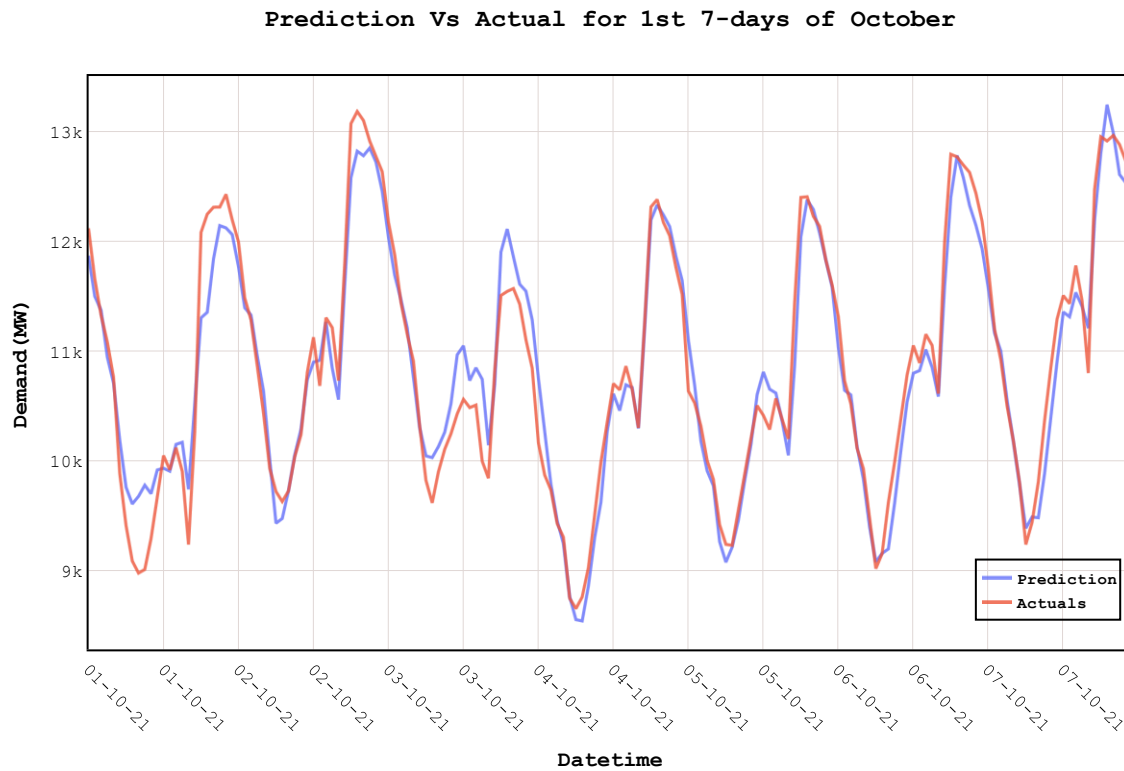


Figure 4.3: Prediction Vs Actual plot of 1st 7-days of October of CNN Model

In figure 4.3, 1st 7- days of October month with actual demands and predicted results are shown. From here it is clearly visible that the model struggles to predict maximum and minimum points of demand. Although almost everywhere it predicted almost accurately.

Regression Plot of CNN Model

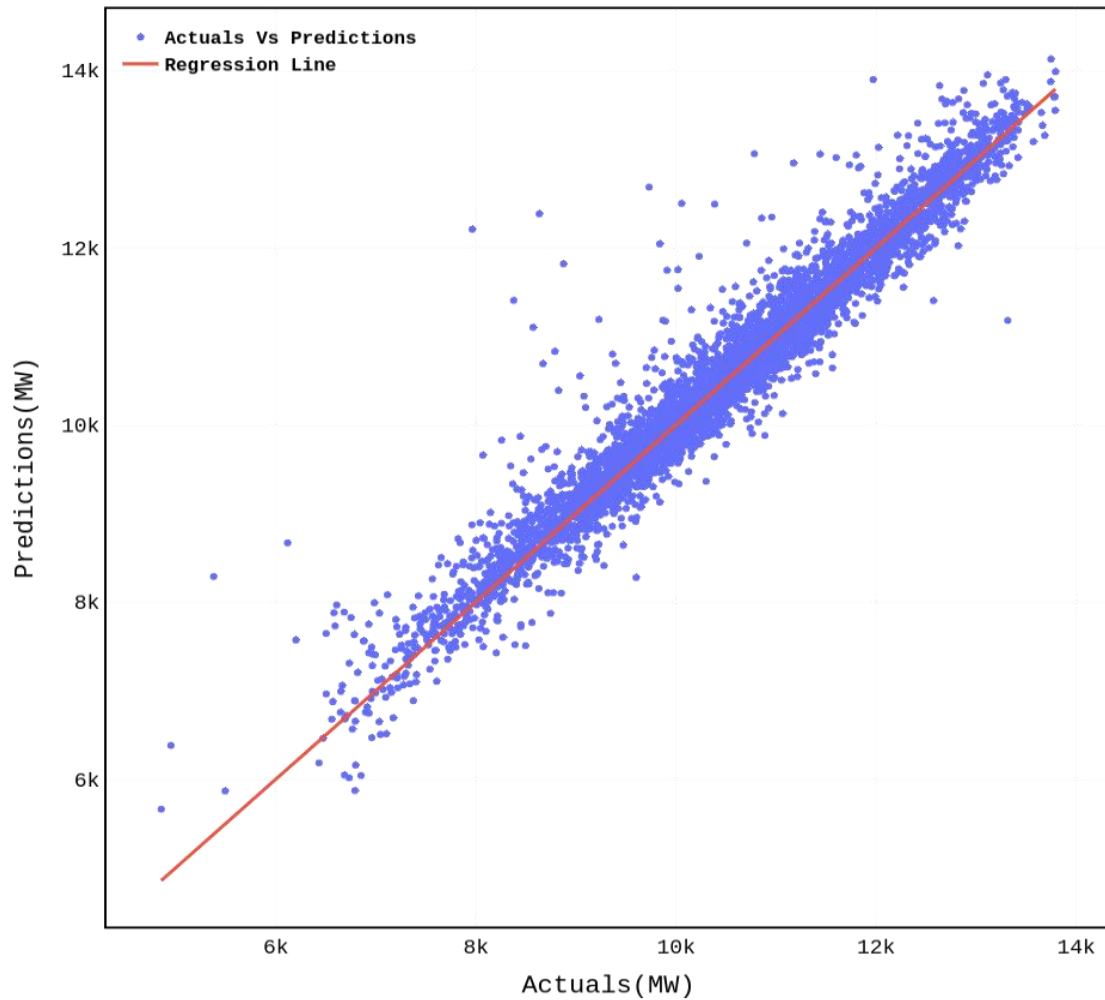


Figure 4.4: Regression Plot of CNN Model

The Figure 4.4, is for showing the regression plot of CNN model for the entire test set. The X-axis presents the actual demands (MW) and Y-axis presents the corresponding predicted demands. From this figure it is visible that almost every point is on the regression line or close to regression line. Very few points are deviated from the regression line.

Error Histogram Plot of CNN Model

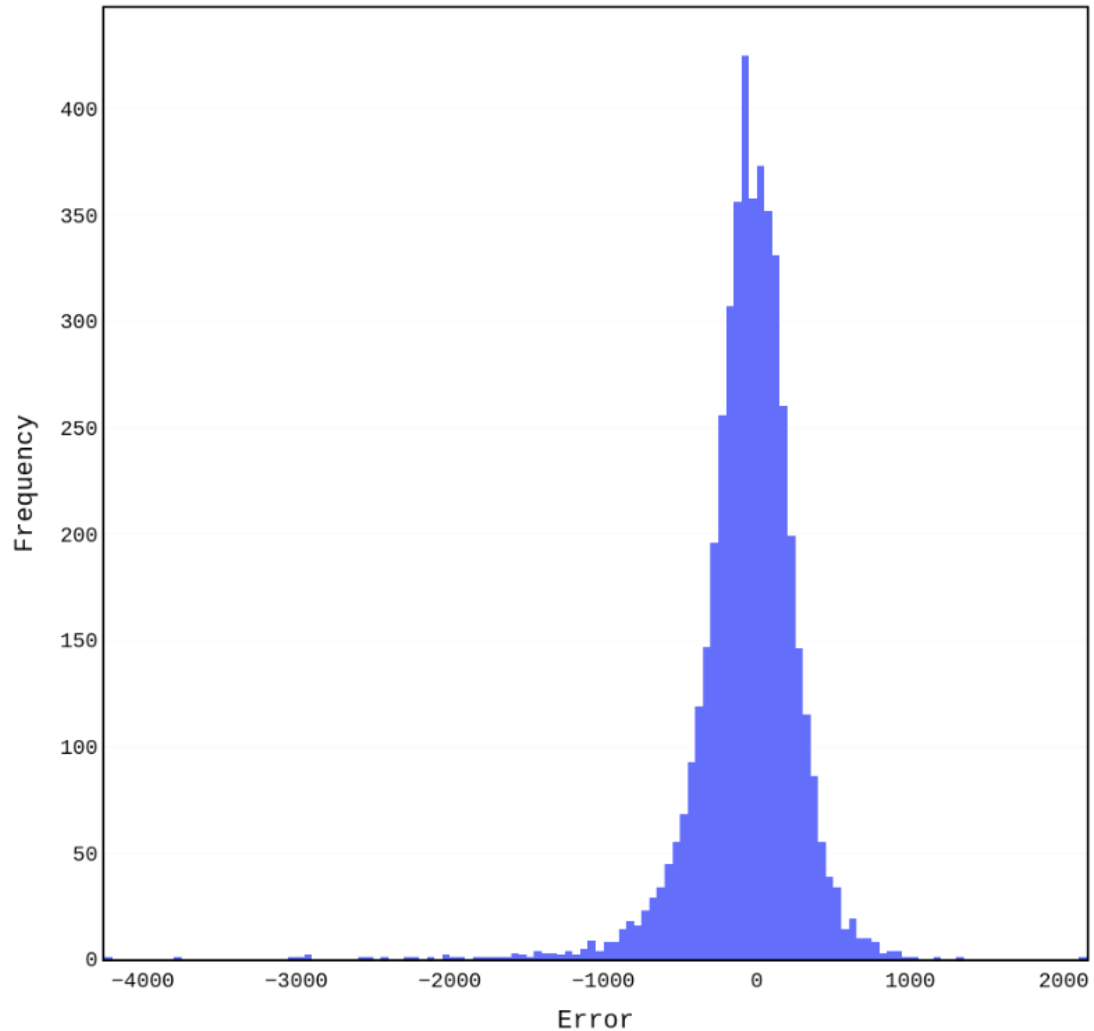


Figure 4.5: Error Histogram Plot of CNN Model

In Figure 4.5 the error histogram plot of CNN model is shown. It is visible that there is significant amount of data points are present with error greater than 300MW or more which is not desirable.

So, from the figures and results it can be derived that although CNN model can predict close to actual values but it is not robust in terms of high and low demands.

4.2.2 GRU Model

GRU is a special kind of RNN with two gates. It incorporates the memory to remember the prior inputs to get the current output. The split test set is used to forecast the hourly energy demand and evaluation metric is used to evaluate the prediction.

Table 4.2: Evaluation Metrics Score of GRU Model of individual Months 2021

<i>Month Name</i>	<i>RMSE</i>	<i>MSE</i>	<i>R-Squared</i>	<i>MAPE (%)</i>	<i>Accuracy (%)</i>
July	199.44	39779.83	98.43	1.61	98.39
August	195.86	38364.53	97.01	1.43	98.57
September	213.71	45671.70	96.33	1.44	98.55
October	212.35	45094.74	97.73	1.52	98.48

In Table 4.2, it shows that the GRU model can predict the demand more accurately than CNN model with lower RMSE score and higher R-squared score and accuracy.

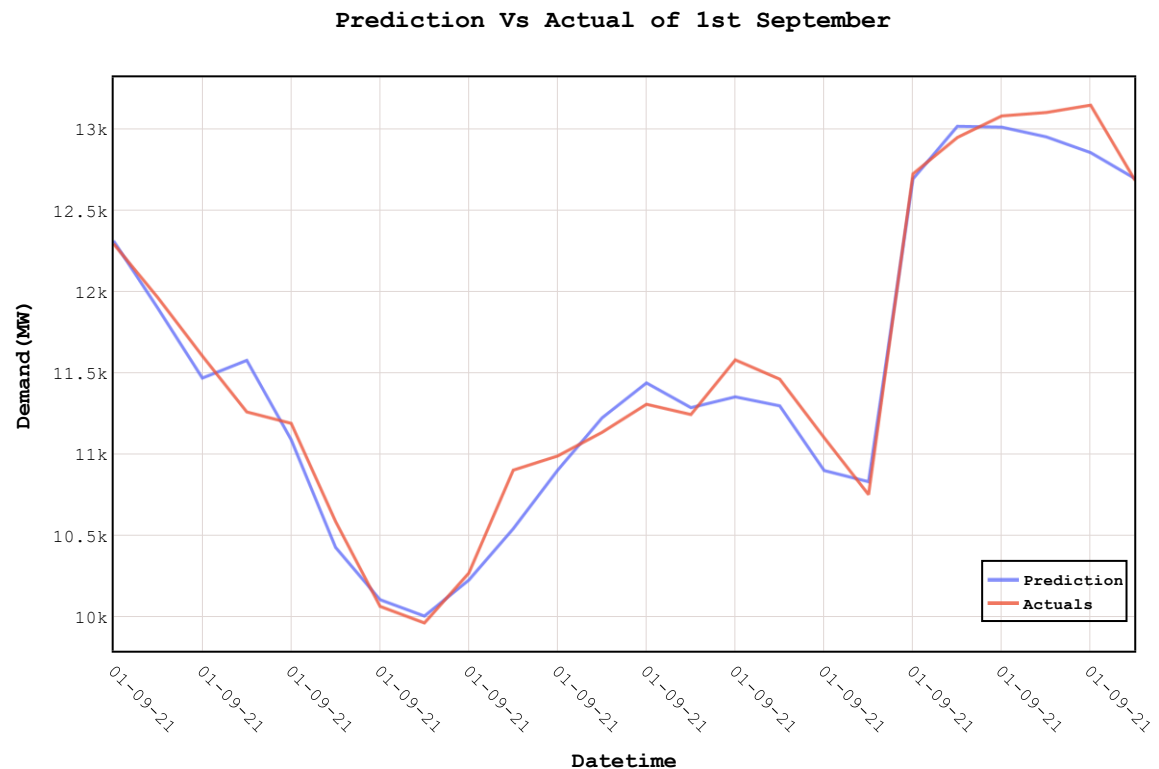


Figure 4.6: Prediction Vs Actual plot of 1st September of GRU Model

In Figure 4.6 the prediction vs actual plot of GRU model for 1st September is shown and it is clearly visible that the model predicts the actual demands almost accurately. For further evaluation, prediction vs actual plot of another day is shown in Figure 4.7

Prediction Vs Actual of 20th April, 2021 (GRU)

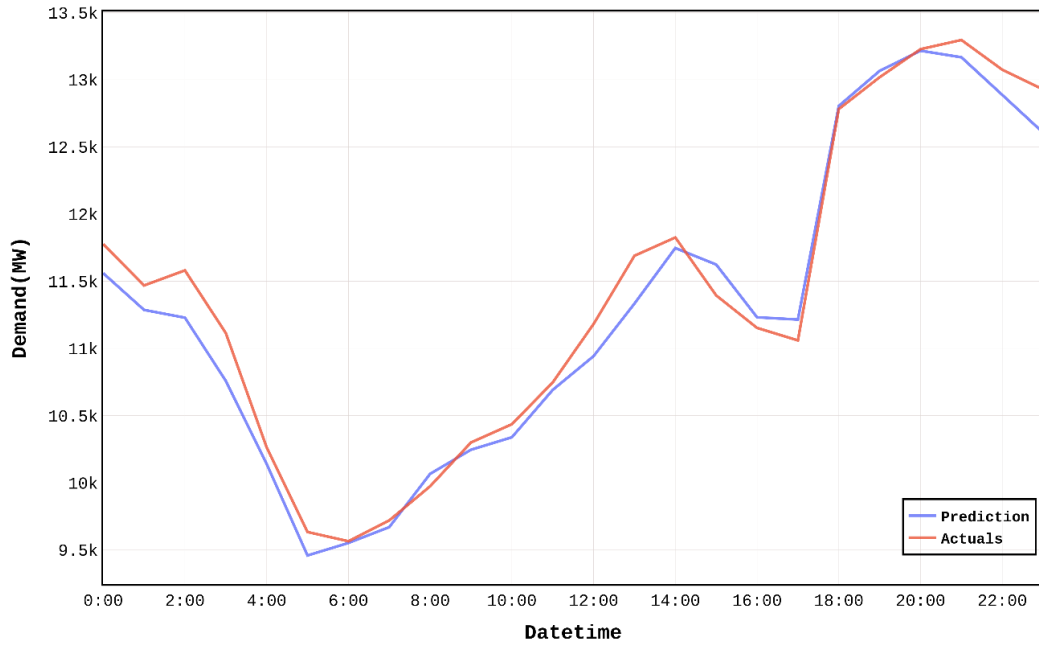


Figure 4.7: Prediction Vs Actual plot of 20th April, 2021 of GRU Model

In this Figure 4.7 prediction of 20th April ,2021 is shown and it can be seen that the prediction of this day is also accurate.

Prediction Vs Actual for 1st 7-days of October

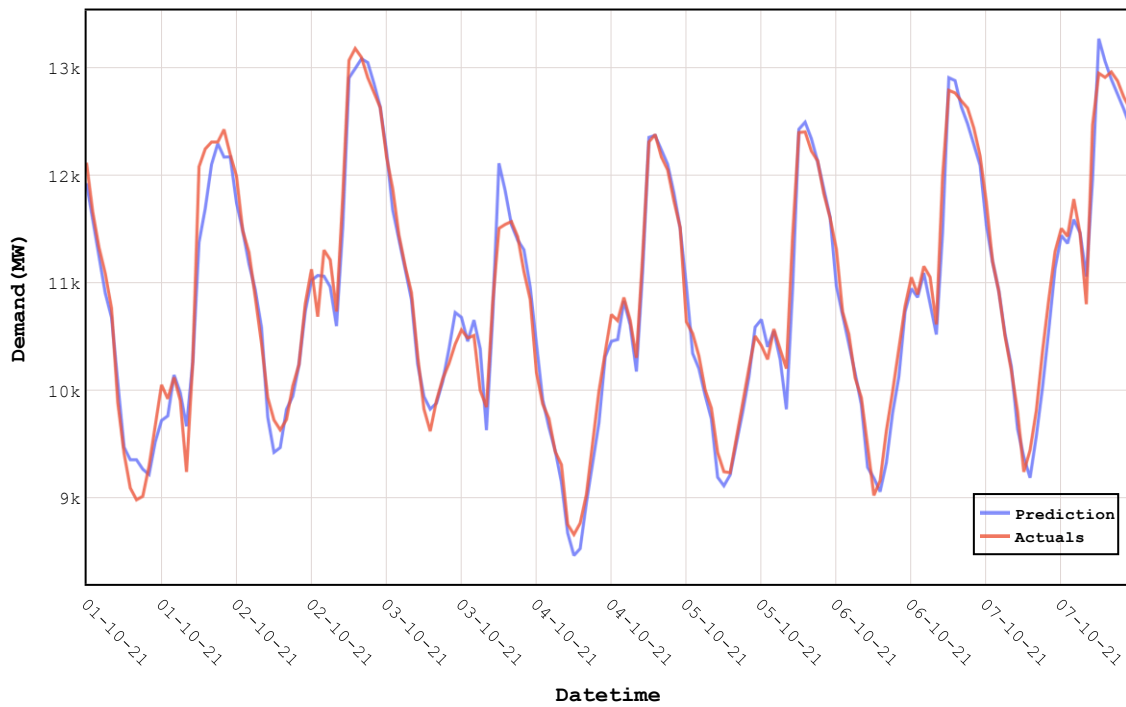


Figure 4.8: Prediction Vs Actual plot of 1st 7-days of October of GRU Model

In Figure 4.8, similar plot is plotted but for 1st 7 days of October,2021. This GRU model can predict almost accurately the actual values although some mismatch is present in peak values.

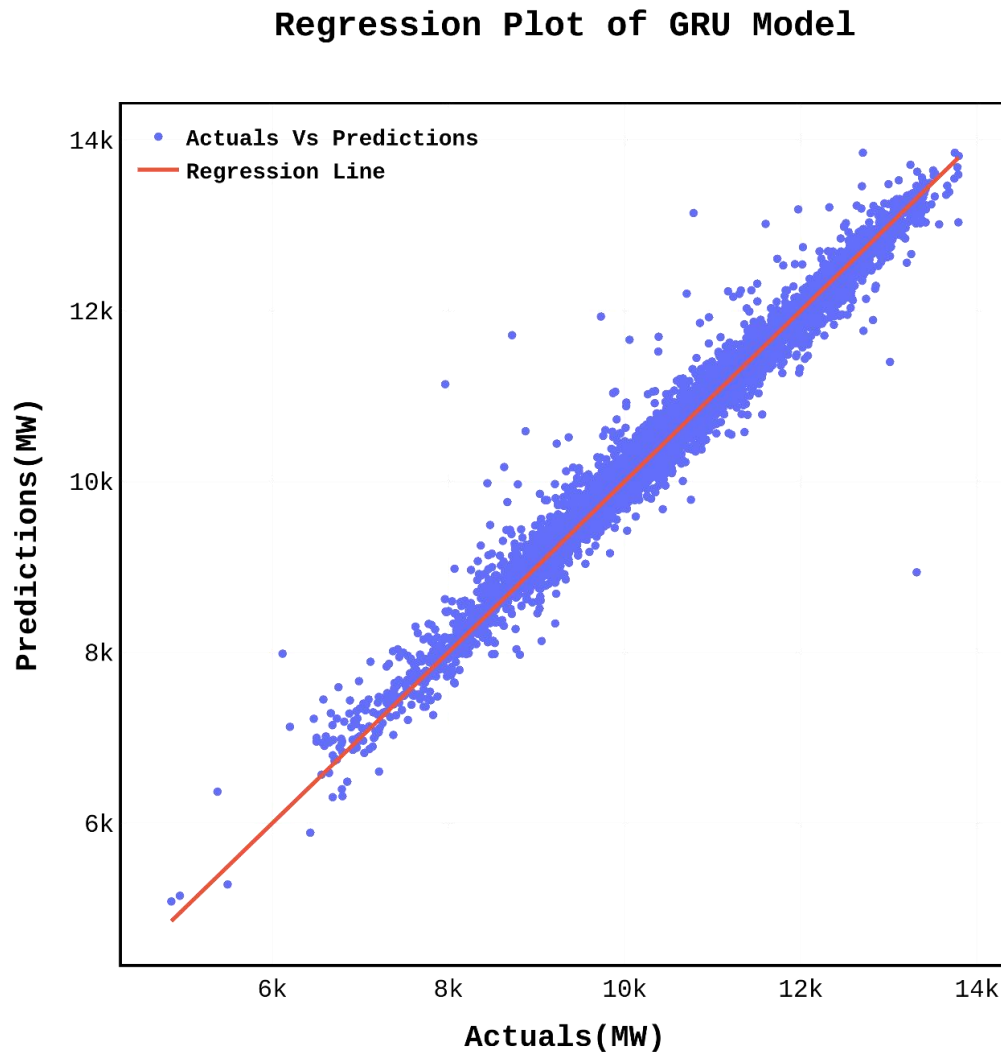


Figure 4.9: Regression Plot of GRU Model

In Figure 4.9, the regression plot of actual and predicted results is shown. From this figure it shows that almost all the values are close to or on the regression line which means accurate prediction of these points. Some deviation is also present which represents the poor prediction for these particular points.

Error Histogram Plot of GRU Model

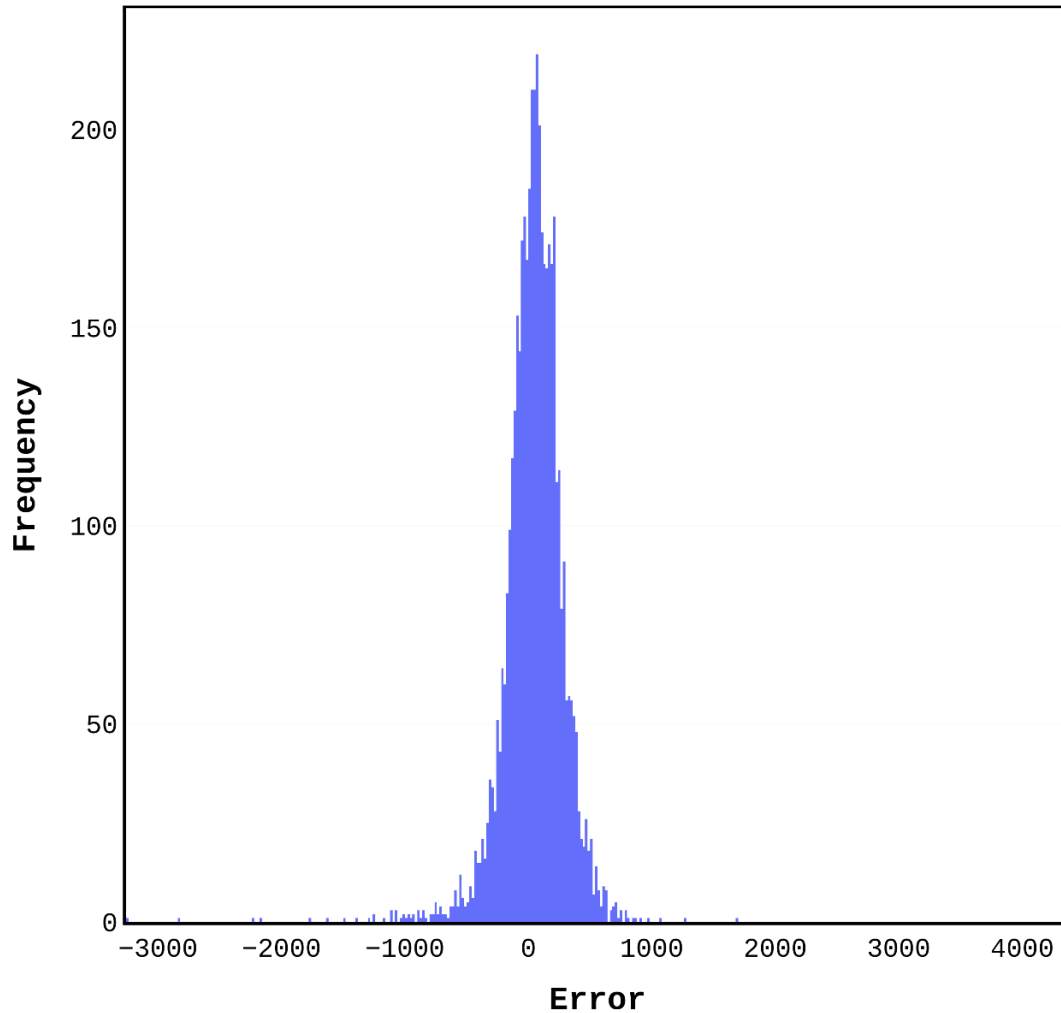


Figure 4.10: Error Histogram Plot of GRU Model

Like CNN model the error histogram plot of GRU model is shown in figure 4.10. The frequency of zero error or close to zero error is higher, although there is some error present with the error higher than 500MW.

4.2.3 LSTM

After CNN and GRU model evaluation, similarly the LSTM model is also evaluated on the test set for individual months.

Table 4.3: Evaluation Metrics Score of LSTM Model of individual Months

<i>Month Name</i>	<i>RMSE</i>	<i>MSE</i>	<i>R-Squared</i>	<i>MAPE (%)</i>	<i>Accuracy (%)</i>
July	191.35	36616.63	98.55	1.52	98.48
August	187.94	35323.21	97.25	1.34	98.66
September	201.68	40678.38	96.73	1.34	98.66
October	203.21	41295.55	97.92	1.44	98.56

In Table 4.3, the results are shown. Comparing with CNN and GRU model, LSTM performs much better in terms of every evaluation metric. As LSTM can remember the long-term sequence and how the time series changes over time, it predicts the output almost close to actual demand.

Prediction Vs Actual of 1st September (LSTM)

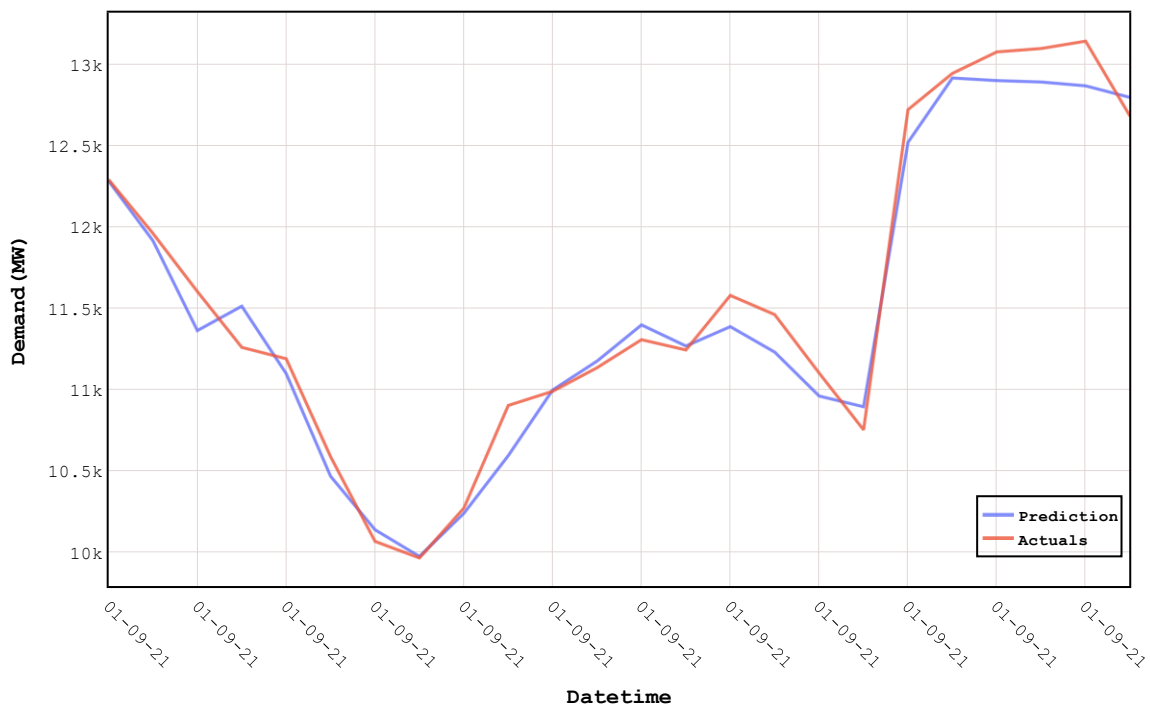


Figure 4.11: Prediction Vs Actual plot of 1st September of LSTM Model

Prediction Vs Actual of 20th April,2021 (LSTM)

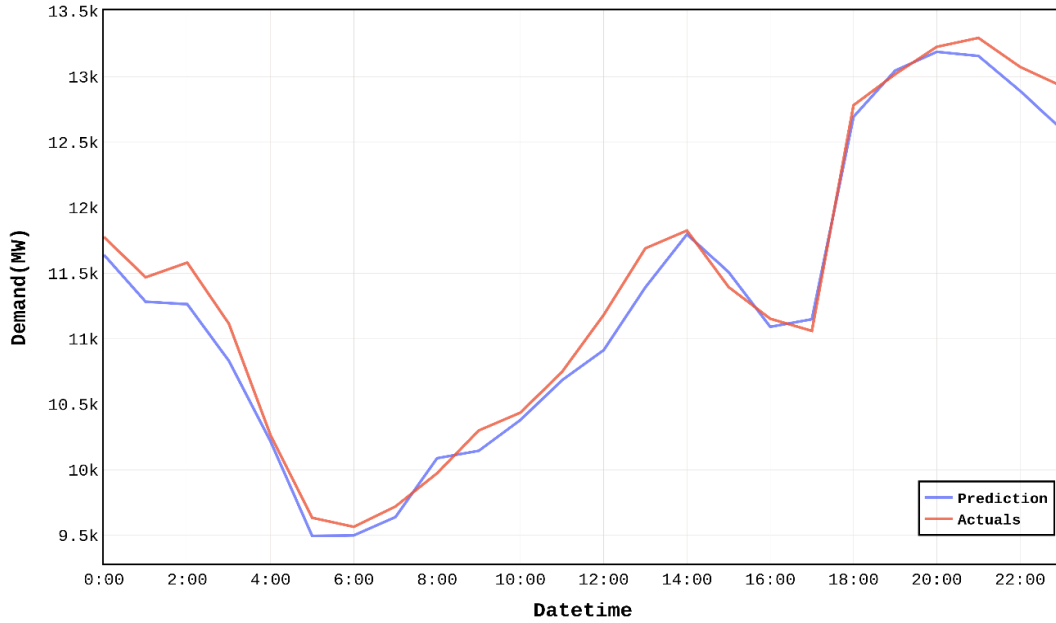


Figure 4.12: Prediction Vs Actual plot of 20th April of LSTM Model

In Figure 4.11 and Figure 4.12 similar plot is shown for two different days. In Figure 4.11 Actual vs prediction plot of 1st September, 2021 is shown where in Figure 4.12 for the day 20th April,2021 is shown. In both the figure it is clearly visible that the LSTM model predicts very well both the normal and the peak values.

Prediction Vs Actual of 1st 7-Days of October (LSTM)

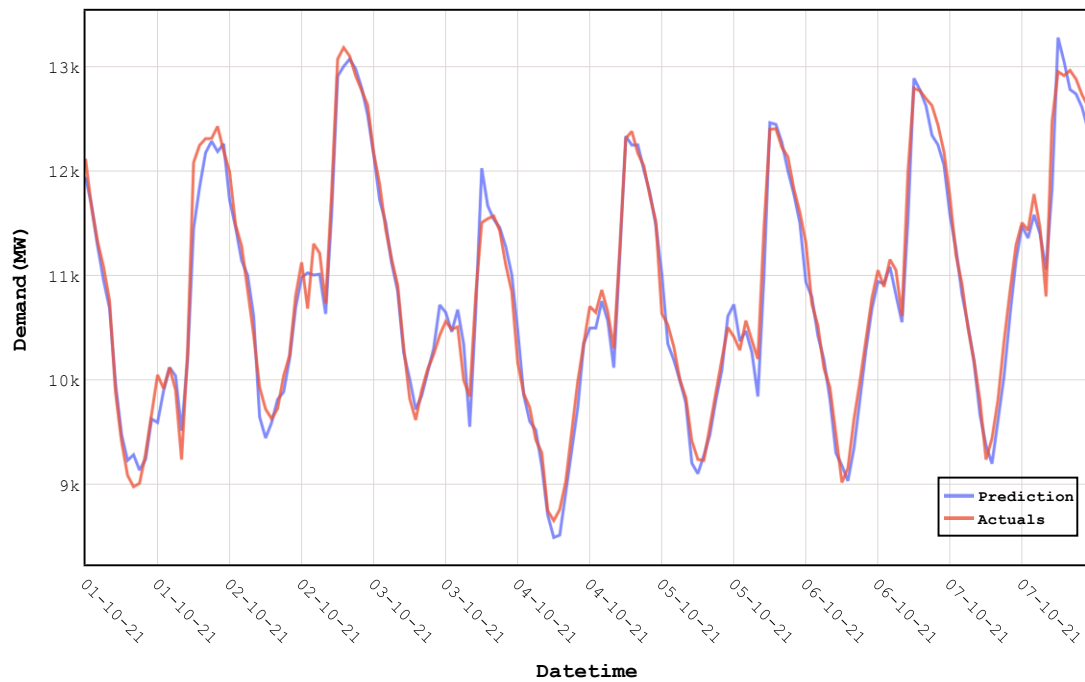


Figure 4.13: Prediction Vs Actual plot of 1st 7 days of October of LSTM Model

For further visualization in Figure 4.13 the same type of plot is plotted but for 1st 7- days of October month, 2021. Although very few mismatch is present in the plot but the overall prediction of this LSTM model is almost accurate.

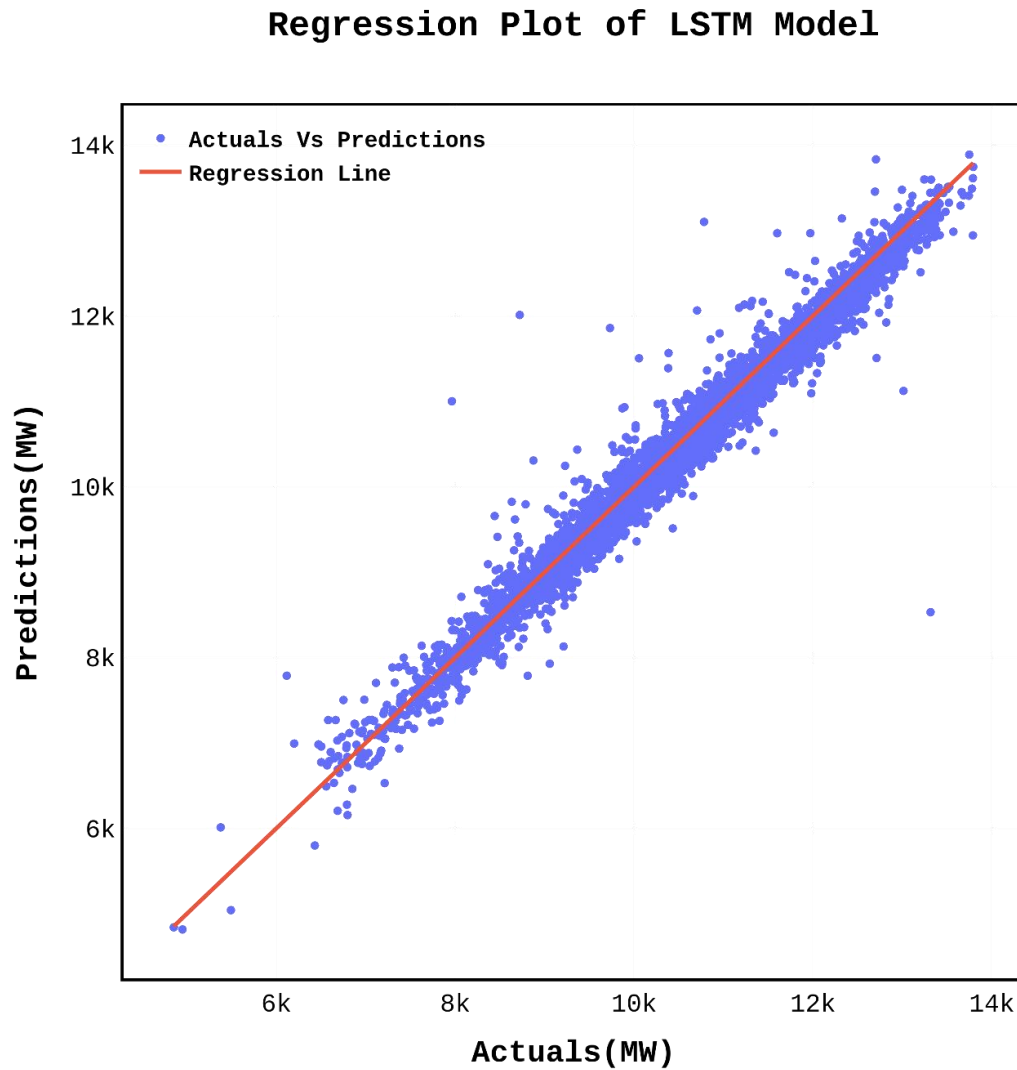


Figure 4.14: Regression Plot of LSTM Model

Error Histogram Plot of LSTM Model

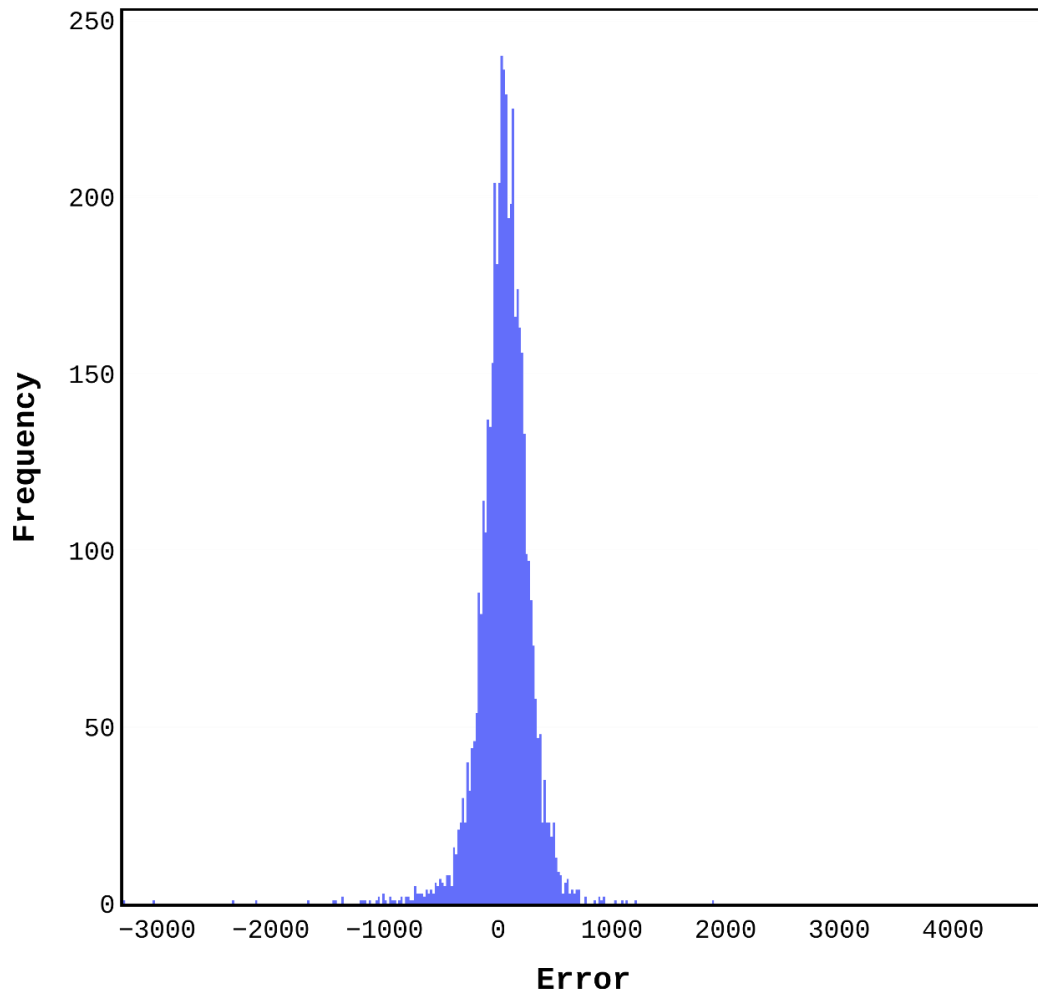


Figure 4.15: Error Histogram Plot of LSTM Model

Like CNN and GRU model, in figure 4.14 and figure 4.15 the regression plot and error histogram plot are shown respectively. In regression plot almost all the points are on or close to regression line, where in error histogram plot the frequency of higher error is very low compare to CNN and GRU model.

4.2.4 CNN-BiLSTM (Proposed)

LSTM model can learn the long-term changes and sequence very well for our data and CNN model can extract the spatial features and can detect the hidden pattern and dependency of the demand output on the other features like weather, holiday, datetime. So combinedly this parallel CNN-BiLSTM model is presented in this work and evaluated on the test set similarly with mentioned five evaluation metric.

Table 4.4: Evaluation Metrics Score of Proposed Model of individual Months 2021

<i>Month Name</i>	<i>RMSE</i>	<i>MSE</i>	<i>R-Squared</i>	<i>MAPE (%)</i>	<i>Accuracy (%)</i>
July	176.43	31130.23	98.77	1.39	98.61
August	160.26	25685.29	98.00	1.13	98.87
September	177.67	31566.63	97.46	1.15	98.85
October	186.83	34908.63	98.24	1.28	98.72

In Table 4.4 the results is shown for individual months of test set. The lowest RMSE is 160.26 MW and lowest MAPE is 1.13% both for August month. In this table the obtained results indicate that the model is very much accurate comparing the traditional CNN, GRU and LSTM model in terms of every evaluation metric.

In Figure 4.16,4.17 and 4.18, the actual vs prediction plot of three different days is shown. In fig prediction of 1st September,2021, in fig 1st October ,2021 and in fig 20th April ,2021 is depicted. In every figure it can be seen that the prediction is very much close to actual demands. This model also can predict the maximum and minimum demands very well comparing the other models.

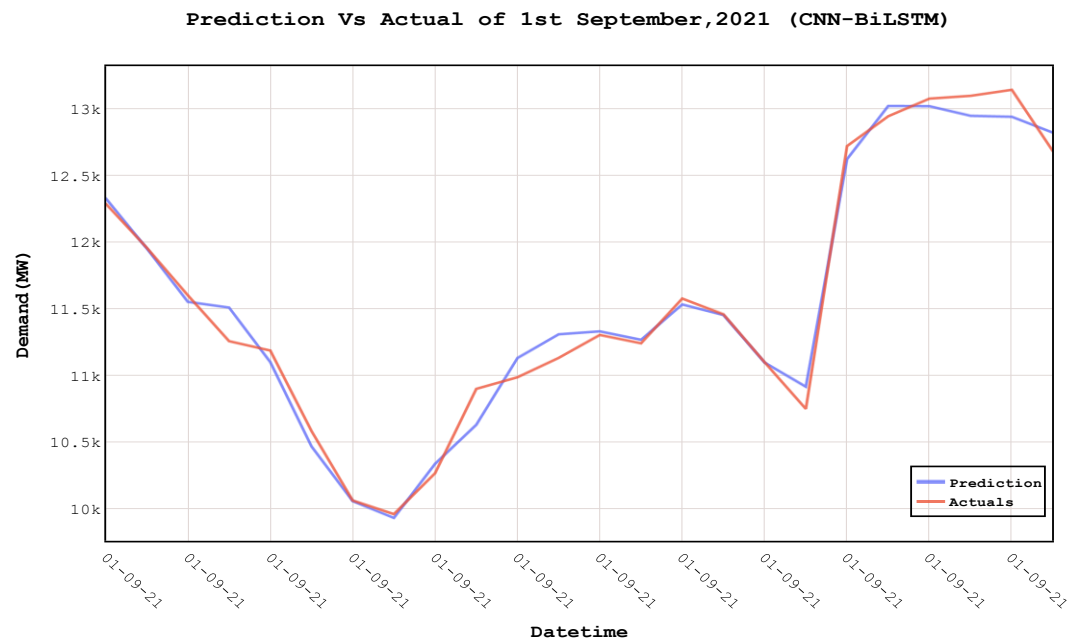


Figure 4.16: Prediction Vs Actual plot of 1st September of CNN-BiLSTM Model

Prediction Vs Actual of 1st October,2021 (CNN-BiLSTM)

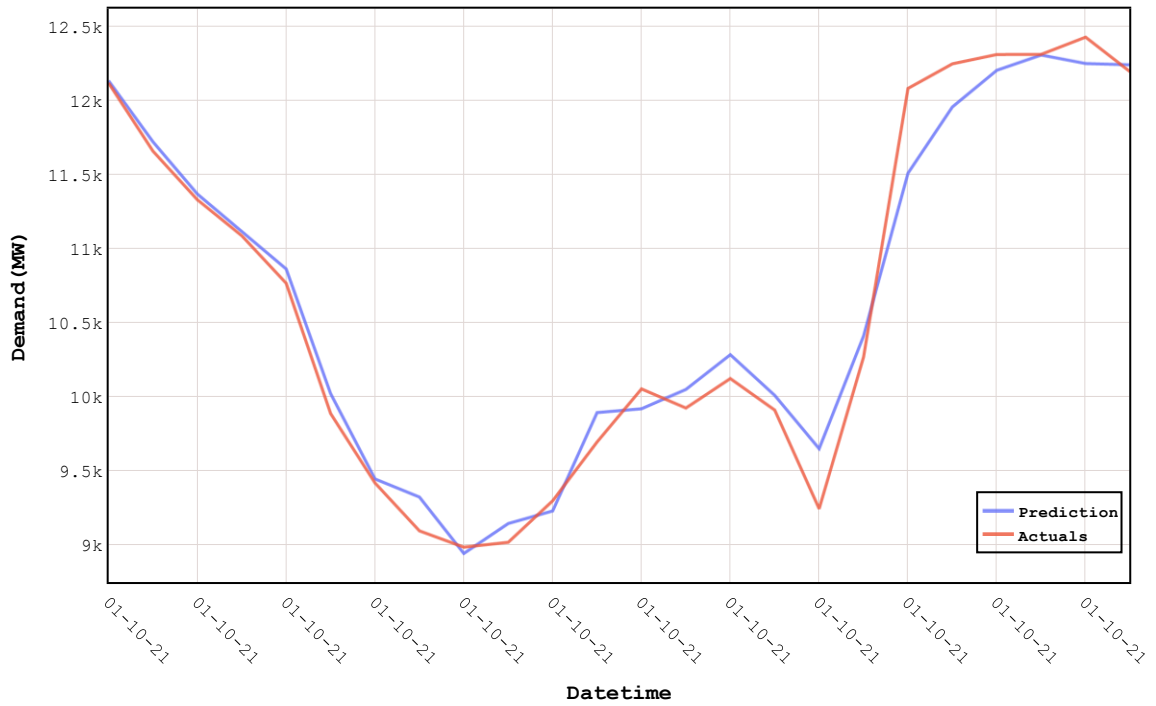


Figure 4.17: Prediction Vs Actual plot of 1st October of CNN-BiLSTM Model

Prediction Vs Actual of 20th April,2021 (CNN-biLSTM)

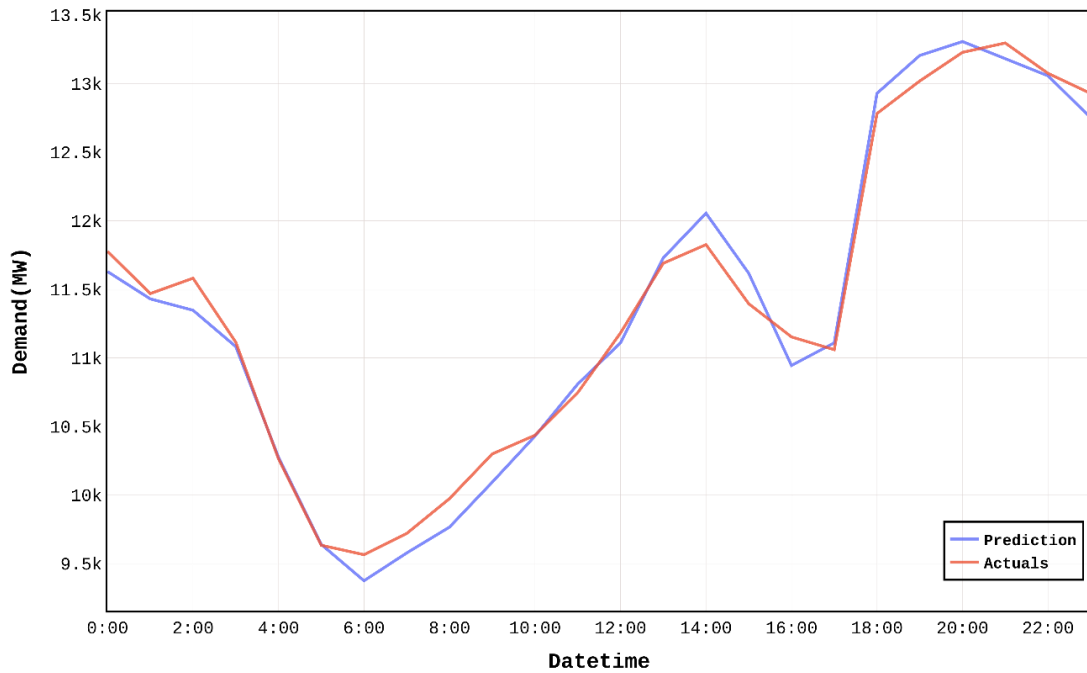


Figure 4.18: Prediction Vs Actual plot of 20th April of CNN-BiLSTM Model

For further evaluation, similar kind of plot but for 7-days is plotted in Figure 4.19

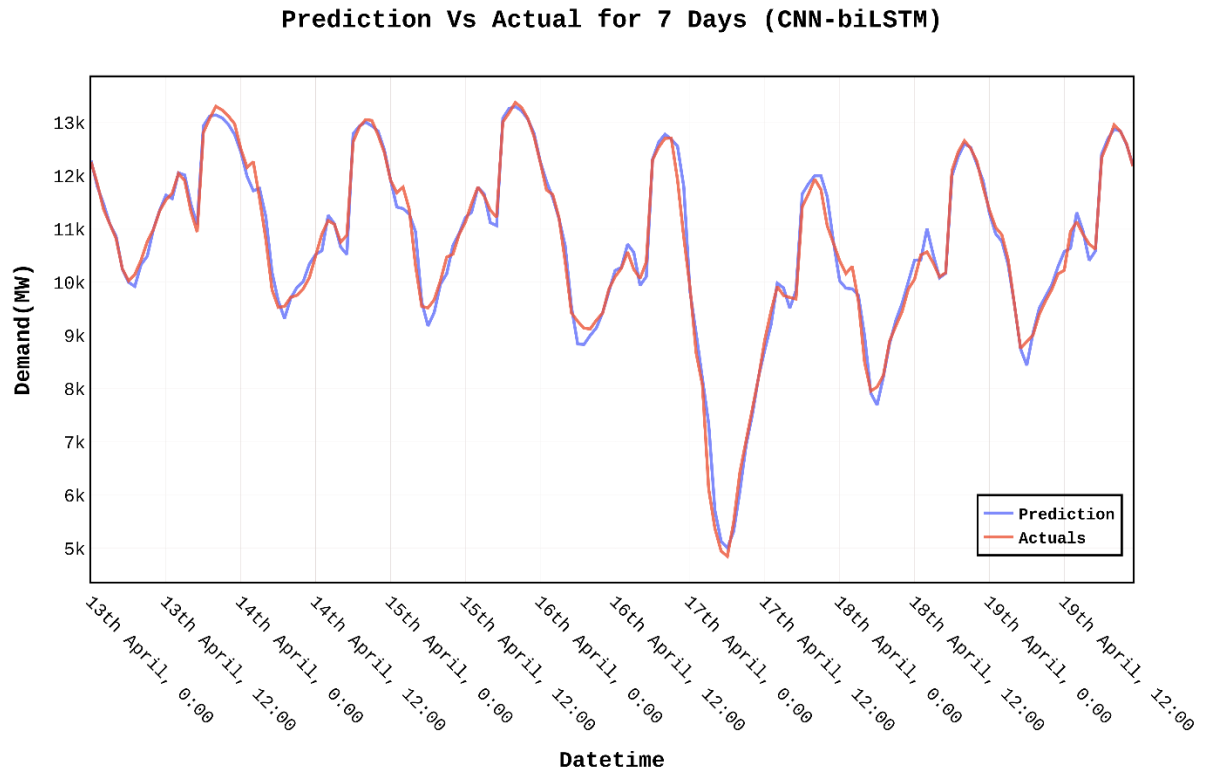


Figure 4.19: Prediction Vs Actual plot of 7 days of April of CNN-BiLSTM Model

In Figure 4.19 the predicted results along with actual results are plotted from 13th April,2021 to 19th April,2021. In figure 4.20, 4.21 and 4.22, same plot for 1st 7 days of August, September and October months of 2021 is shown respectively.

Prediction Vs Actual of 1st 7-Days of August,2021 (CNN-BiLSTM)

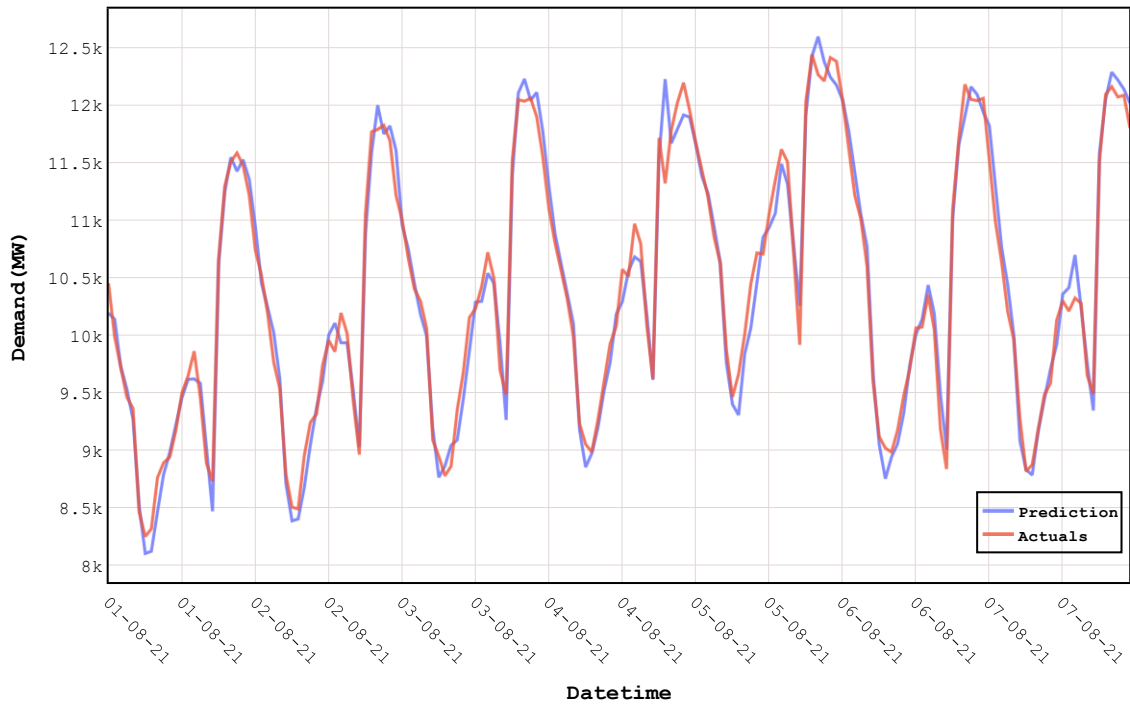


Figure 4.20: Prediction Vs Actual plot of 1st 7 days of August of CNN-BiLSTM Model

Prediction Vs Actual of 1st 7-Days of September,2021 (CNN-BiLSTM)

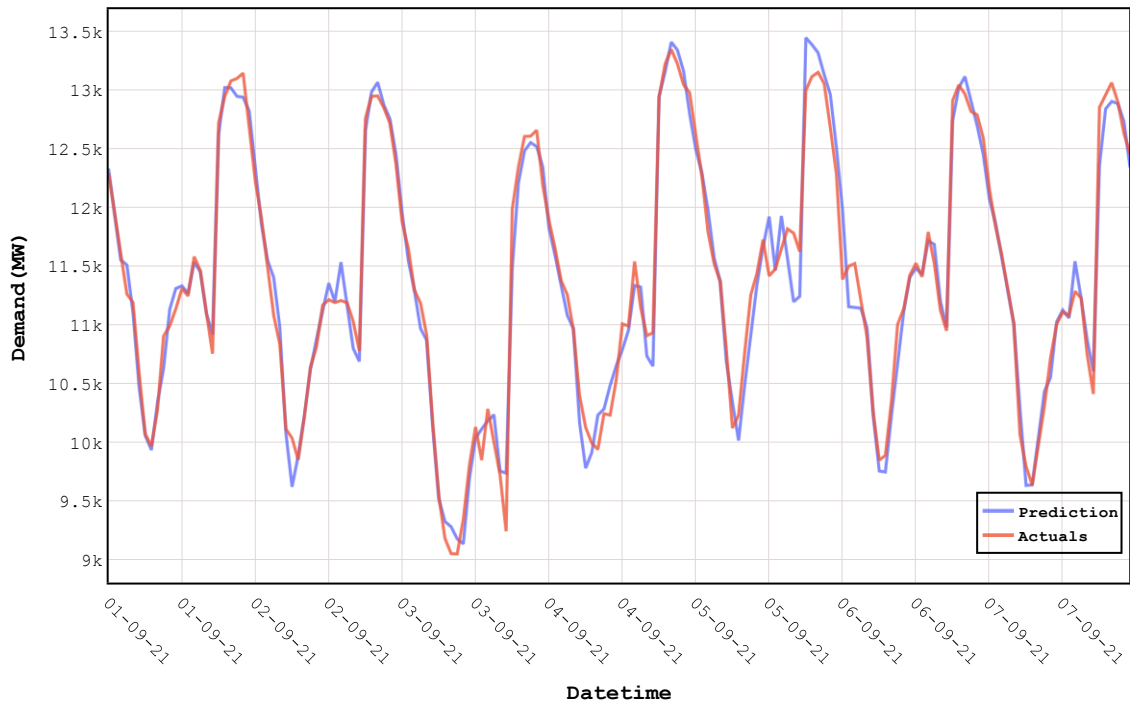


Figure 4.21: Prediction Vs Actual plot of 1st 7 days of September of CNN-BiLSTM Model

Prediction Vs Actual of 1st 7-Days of October,2021 (CNN-BiLSTM)

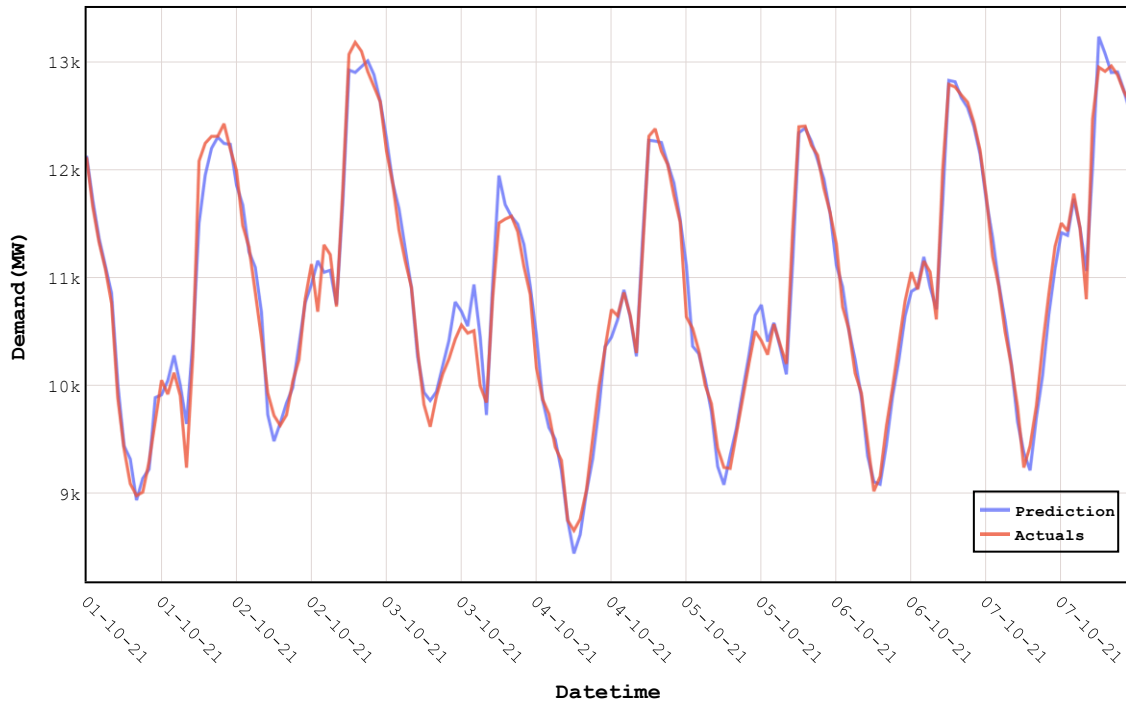


Figure 4.22: Prediction Vs Actual plot of 1st 7 days of October of CNN-BiLSTM Model

Although for April, August, October months the model predicted very well and accurate, for September month it struggles in some data points.

Prediction Vs Actual of July,2021 (CNN-BiLSTM)

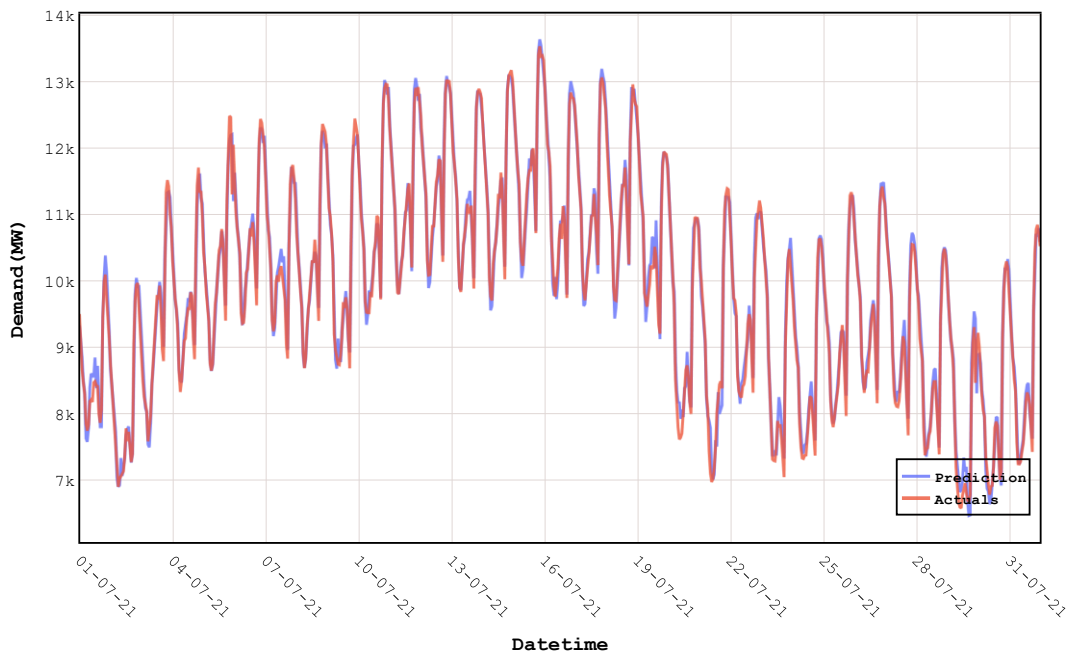


Figure 4.23: Prediction Vs Actual Plot of July, 2021 of CNN-BiLSTM Model

Prediction Vs Actual of August,2021 (CNN-BiLSTM)

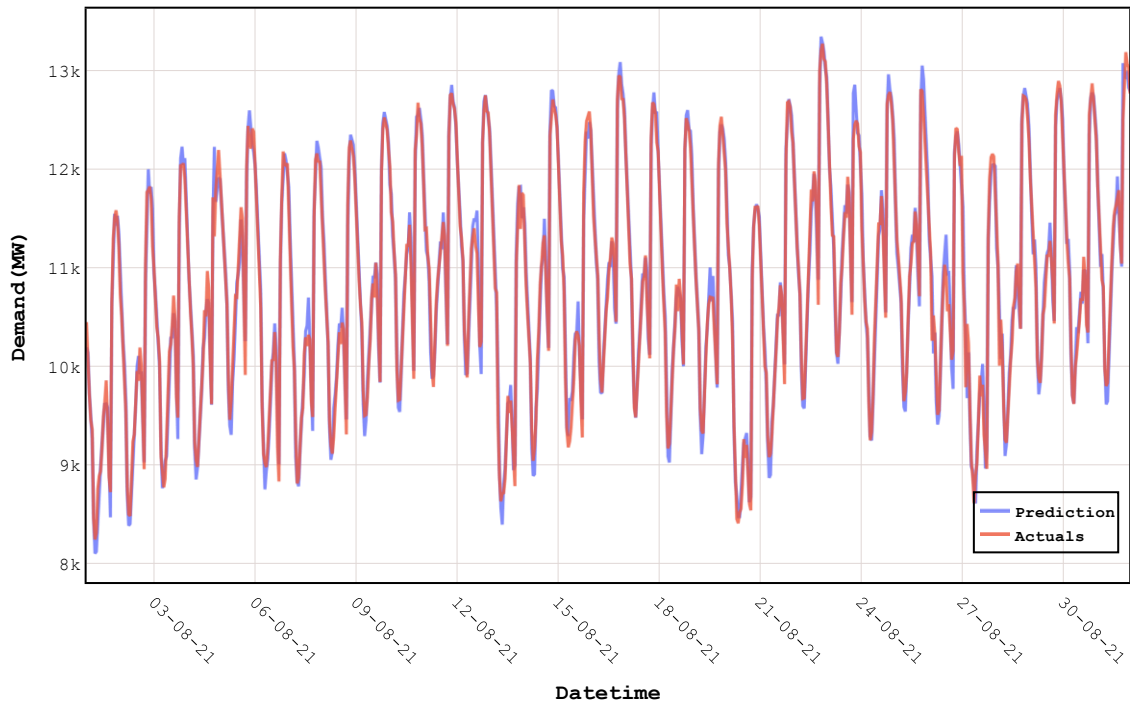


Figure 4.24: Prediction Vs Actual Plot of August, 2021 of CNN-BiLSTM Model

Prediction Vs Actual of September,2021 (CNN-BiLSTM)

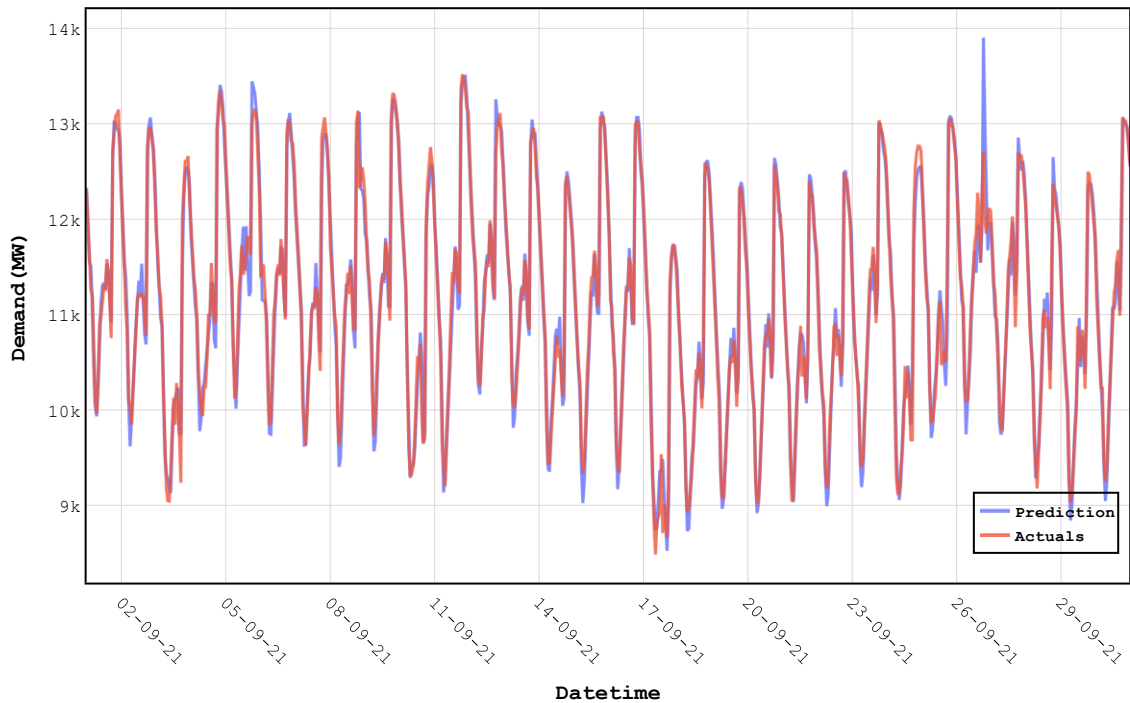


Figure 4.25: Prediction Vs Actual Plot of September, 2021 of CNN-BiLSTM Model

In Figure 4.23, 4.24 and 4.25, the similar plots plotted but for the whole month to further visualization of prediction. The model predicted the peak values and normal values very accurately except 26th September,2021.

Regression Plot of CNN-biLSTM Model

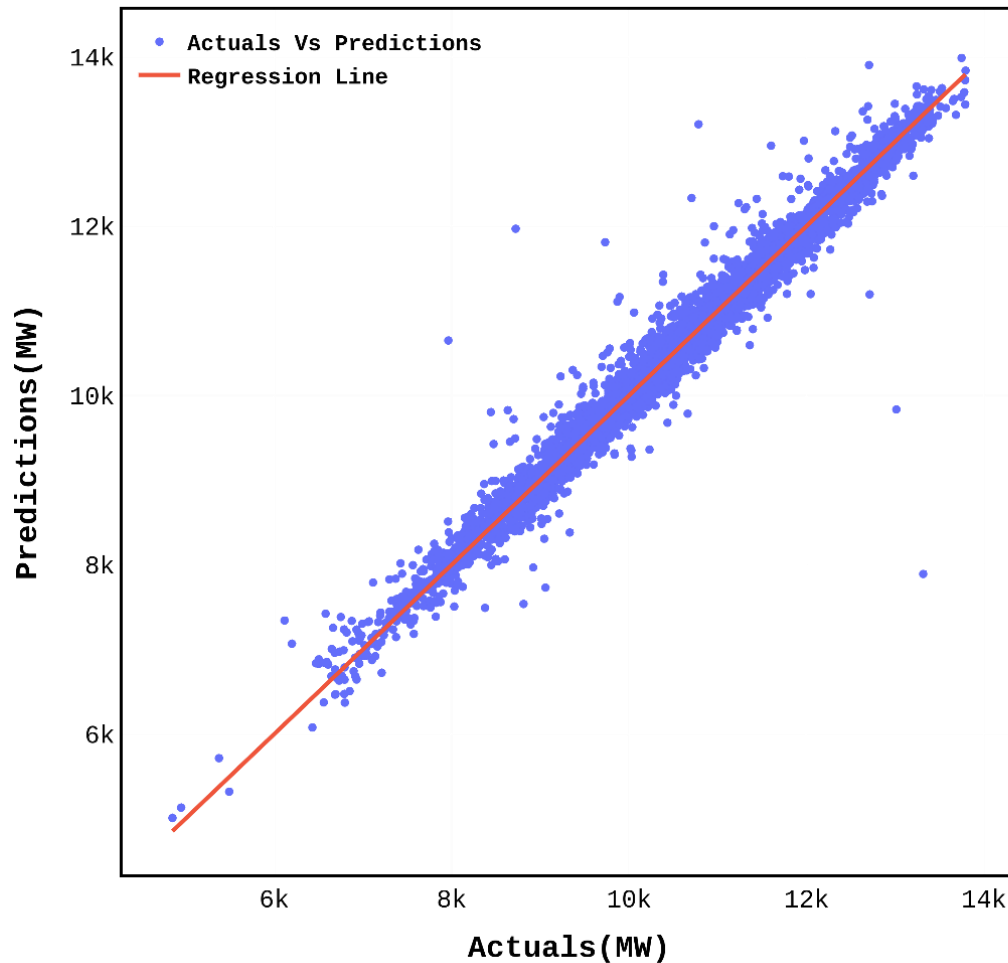


Figure 4.26: Regression Plot of CNN-BiLSTM Model

In Figure 4.26 the actual vs prediction regression plot of CNN-BiLSTM is shown. Very few points are deviated from the regression line. Almost all points are close to or on the regression line which indicates that this model performs comparatively better than the traditional CNN, LSTM and GRU model.

Error Histogram Plot of CNN-biLSTM Model

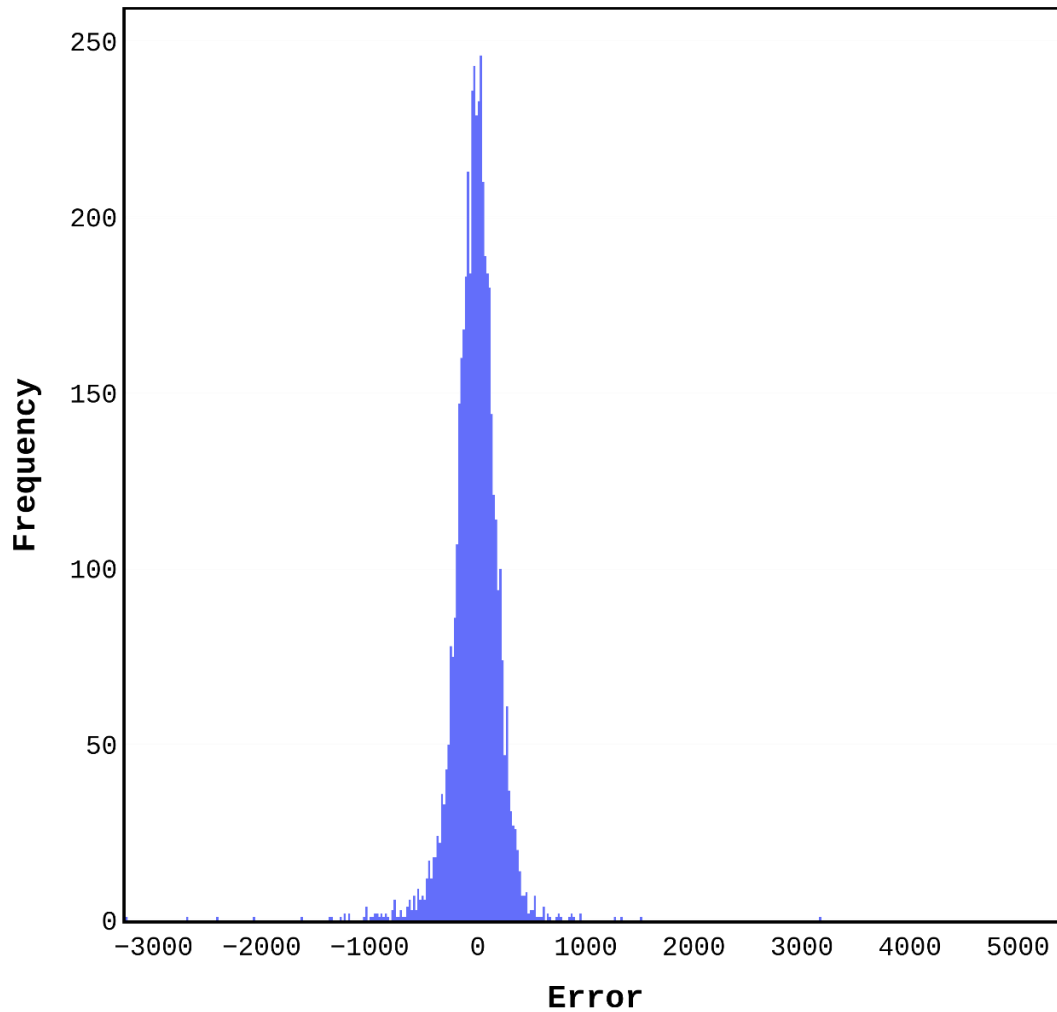


Figure 4.27: Error Histogram Plot of CNN-BiLSTM Model

Figure 4.27 shows the error histogram for CNN-BiLSTM. In this figure it is visible that the frequency of points with error greater 500MW is very less comparing to the frequency of zero error or close to zero error.

In Table 4.5,4.6 and 4.7 the daily average Actual demand and corresponding predicted demands by CNN-BiLSTM model is shown for July, August and September month respectively.

Table 4.5: Daily Average Actual Demand and Predicted Demand for July Month of Proposed Model

<i>Date</i>	<i>Actual (MW)</i>	<i>Prediction (MW)</i>
2021-07-01	8648.083	8750.132
2021-07-02	8060.916	8096.951
2021-07-03	9337.208	9330.509
2021-07-04	9789.0	9834.208
2021-07-05	10321.458	10283.859
2021-07-06	10753.041	10809.784
2021-07-07	10341.833	10431.541
2021-07-08	10376.0	10351.935
2021-07-09	10203.625	10206.123
2021-07-10	10918.958	10938.808
2021-07-11	11314.083	11342.807
2021-07-12	11561.083	11552.818
2021-07-13	11322.166	11356.273
2021-07-14	11402.375	11385.324
2021-07-15	11864.416	11822.275
2021-07-16	11311.416	11315.380
2021-07-17	11294.041	11326.553
2021-07-18	11304.708	11317.413
2021-07-19	10685.541	10784.804
2021-07-20	9245.0	9361.765
2021-07-21	8994.541	8959.491
2021-07-22	9493.25	9546.815
2021-07-23	8808.166	8917.386
2021-07-24	8635.458	8736.140
2021-07-25	9273.791	9305.622
2021-07-26	9691.291	9742.861
2021-07-27	9277.375	9411.510
2021-07-28	8736.25	8817.208
2021-07-29	7862.458	7986.746
2021-07-30	8158.666	8136.191
2021-07-31	8641.375	8691.208

Table 4.6: Daily Average Actual Demand and Predicted Demand for August Month of Proposed Model

<i>Date</i>	<i>Actuals (MW)</i>	<i>Predictions (MW)</i>
2021-08-01	9732.416	9686.183
2021-08-02	10063.375	10065.311
2021-08-03	10392.25	10372.296
2021-08-04	10516.416	10492.828
2021-08-05	11107.083	11041.142
2021-08-06	10467.916	10479.713
2021-08-07	10441.083	10511.491
2021-08-08	10570.541	10617.773
2021-08-09	10899.625	10919.054
2021-08-10	11160.5	11185.028
2021-08-11	11308.208	11290.584
2021-08-12	11326.791	11340.877
2021-08-13	10183.708	10230.737
2021-08-14	10899.208	10904.335
2021-08-15	10731.0	10779.203
2021-08-16	11292.166	11274.806
2021-08-17	11074.208	11117.987
2021-08-18	10839.625	10860.004
2021-08-19	10819.375	10836.495
2021-08-20	9914.125	9967.918
2021-08-21	10668.583	10671.526
2021-08-22	11526.625	11483.477
2021-08-23	11479.541	11540.967
2021-08-24	11117.541	11132.010
2021-08-25	11199.291	11215.917
2021-08-26	10881.229	10894.381
2021-08-27	10147.083	10171.540
2021-08-28	10864.083	10868.605
2021-08-29	11269.791	11289.237
2021-08-30	11129.5	11135.521
2021-08-31	11487.708	11442.934

Table 4.7: Daily Average Actual Demand and Predicted Demand for September Month of Proposed Model

<i>Date</i>	<i>Actuals (MW)</i>	<i>Predictions (MW)</i>
2021-09-01	11555.291	11554.035
2021-09-02	11396.583	11418.608
2021-09-03	10693.916	10690.266
2021-09-04	11409.458	11371.810
2021-09-05	11786.5	11801.815
2021-09-06	11493.125	11458.211
2021-09-07	11336.916	11329.601
2021-09-08	11319.125	11294.513
2021-09-09	11589.041	11536.126
2021-09-10	10938.625	10937.887
2021-09-11	11417.125	11371.881
2021-09-12	11731.5	11765.802
2021-09-13	11465.625	11461.367
2021-09-14	10911.0	10945.407
2021-09-15	11237.375	11140.828
2021-09-16	11220.0	11221.174
2021-09-17	10125.083	10195.183
2021-09-18	10567.041	10556.639
2021-09-19	10646.208	10703.383
2021-09-20	10730.291	10740.526
2021-09-21	10710.541	10755.167
2021-09-22	10772.916	10773.835
2021-09-23	11214.333	11156.575
2021-09-24	10837.416	10831.381
2021-09-25	11309.166	11342.319
2021-09-26	11521.083	11529.054
2021-09-27	11373.583	11365.962
2021-09-28	10894.458	10960.380
2021-09-29	10730.583	10720.985
2021-09-30	11209.0	11143.850

4.3 Overall Comparison

Comparing all the implemented model in table 4.8, the proposed CNN-BiLSTM model performs best in terms of every evaluation metrics and for every individual month. And 2nd best model is LSTM. Because it learns the long-term sequence very well for this particular Energy Demand Dataset.

Table 4.8: Comparison of Evaluation Metrics Score of Individual Months of Models

<i>Month Name</i>	<i>Metrics</i>	<i>Model Name</i>			
		<i>CNN</i>	<i>GRU</i>	<i>LSTM</i>	<i>CNN-BiLSTM</i>
July	RMSE	285.92	199.44	191.35	176.43
	MSE	81751.98	39779.83	36616.63	31130.23
	R-Squared	94.03	98.43	98.55	98.77
	MAPE	2.31	1.61	1.52	1.39
	Accuracy	97.69	98.39	98.48	98.61
August	RMSE	253.07	195.86	187.94	160.26
	MSE	64048.03	38364.53	35323.21	25685.29
	R-Squared	95.01	97.01	97.25	98.00
	MAPE	1.83	1.43	1.34	1.13
	Accuracy	98.17	98.57	98.66	98.87
September	RMSE	278.97	213.71	201.68	177.67
	MSE	77829.75	45671.70	40678.38	31566.63
	R-Squared	93.74	96.33	96.73	97.46
	MAPE	1.86	1.44	1.34	1.15
	Accuracy	98.14	98.56	98.66	98.85
October	RMSE	290.24	212.35	203.21	186.83
	MSE	84244.28	45094.74	41295.55	34908.63
	R-Squared	95.77	97.73	97.92	98.24
	MAPE	2.18	1.52	1.44	1.28
	Accuracy	97.82	98.48	98.56	98.72

In Table 4.9 the models are evaluated on the total test set. And the proposed model with the lowest RMSE, MSE, MAPE and highest accuracy and R-squared score, performed best comparing to these traditional deep learning model.

Table 4.9: Comparison of Evaluation Metrics Score on Whole Test Set of Models

<i>Model name</i>	<i>RMSE</i>	<i>MSE</i>	<i>R²</i>	<i>MAPE</i>	<i>Accuracy</i>
CNN	343.5489056	118025.8506	94.33963054	2.267586688	97.73241331
GRU	257.9533592	66538.59026	96.80304338	1.738415817	98.27158418
LSTM	250.2367135	62618.41282	96.99690068	1.639398134	98.36060186
CNN-BiLSTM	239.3840430	57304.72991	97.36540077	1.51270646	98.48729353

Table 4.10: Comparison among Proposed Model with Recent Existing Models

<i>Paper</i>	<i>RMSE (MW)</i>	<i>MAPE (%)</i>	<i>Model</i>
Proposed	239.38	1.51	CNN-BiLSTM
(Massaoudi et al., 2021) [120]	278.68	2.01	XGB-LGBM-MLP
(Sajjad et al., 2020) [4]	301.56	2.73	CNN-GRU
(Rafi, Nahid-Al-Masood, Deeba and Hossain, 2021)[121]	323.74	3.41	CNN-LSTM

In Table 4.10 the proposed approach is compared with the current and recent models found in the literatures. And for the Energy Demand forecasting in Bangladesh, our proposed model performs best.

Chapter 5

Conclusion & Future Works

The steady expansion in data analysis in all areas, combined with improved computational capability to use a huge amount of data and a lot of data analysis, ensures that the fundamental issue of understanding and using one's data will remain critical across a large number of power stations for power system failures. Although simple statistical approaches and machine learning for data analysis have been used for a long time, the use of sophisticated techniques for intelligent data analysis has only lately been significant. Different models for STLF using time series approaches and artificial intelligence techniques have been presented in the literature during the last few decades.

In this work, we proposed a hybrid parallel CNN-BiLSTM model to predict the load demand for efficient power generation and management. The proposed model is tested on PGCB Power Demand Data. For accurate prediction of load demand, weather data and date time related data like holiday, hours, day of week are merged as features. Due to the non-linearity in the input data, first data has been scaled by applying standard scalar then fed the normalized data for further training processes.

Next, we investigated several traditional deep learning models and optimally developed a hybrid model in which we combined CNN with Bidirectional LSTM in parallel. First, we extracted spatial features through CNN and in parallel multi-layered Bidirectional LSTM to extract temporal features corresponding to the input time series data. The projected model work well as compared to alternative baseline models, indicating the real-world implementation of our proposed model.

In future the model can be further improved by incorporating attention layers and parameter optimization. And model can be applied on several more test set to tune and implement more robustly. The dataset is confined to only the Bangladesh dataset, whereas it can be expanded to other datasets and compared the results. The work can be proposed on other datasets and will execute the improvement. The fuzzy logic concept can be added to have more accuracy in terms of forecasting. In case of both residential and commercial load, the model can be used to have an idea about the Short-Term Load Forecasting. In future the evolution of the model can be tested on the Middle Term Load Forecasting and as well as the Long-Term Load Forecasting.

PAPER NAME

**An Efficient Short Term Load Demand Fo
recasting Using a Novel Parallel CNN-BiL
STM Hybrid Neural Netw**

WORD COUNT

19835 Words

CHARACTER COUNT

115736 Characters

PAGE COUNT

90 Pages

FILE SIZE

6.9MB

SUBMISSION DATE

May 17, 2022 9:58 AM GMT+6

REPORT DATE

May 17, 2022 10:05 AM GMT+6**● 20% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 11% Internet database
- 6% Publications database
- Crossref database
- Crossref Posted Content database
- 18% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Cited material
- Small Matches (Less than 10 words)
- Manually excluded text blocks

References

- [1] “Shodhganga@INFLIBNET: Short Term Load Forecasting for Smart Power Systems.” <http://shodhganga.inflibnet.ac.in:8080/jspui/handle/10603/222275> (accessed May 09, 2022).
- [2] A. S. Khwaja, A. Anpalagan, M. Naeem, and B. Venkatesh, “Joint bagged-boosted artificial neural networks: Using ensemble machine learning to improve short-term electricity load forecasting,” *Electric Power Systems Research*, vol. 179, p. 106080, Feb. 2020, doi: 10.1016/J.EPSR.2019.106080.
- [3] B. Farsi, M. Amayri, N. Bouguila, and U. Eicker, “On short-term load forecasting using machine learning techniques and a novel parallel deep LSTM-CNN approach,” *IEEE Access*, vol. 9, pp. 31191–31212, 2021, doi: 10.1109/ACCESS.2021.3060290.
- [4] M. Sajjad *et al.*, “A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting,” *IEEE Access*, vol. 8, pp. 143759–143768, 2020, doi: 10.1109/ACCESS.2020.3009537.
- [5] A. D. Papalexopoulos and T. C. Hesterberg, “A regression-based approach to short-term system load forecasting,” *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1535–1547, 1990, doi: 10.1109/59.99410.
- [6] M. T. Hagan and S. M. Behr, “The Time Series Approach to Short Term Load Forecasting,” *IEEE Transactions on Power Systems*, vol. 2, no. 3, pp. 785–791, 1987, doi: 10.1109/TPWRS.1987.4335210.
- [7] H. S. Hippert, C. E. Pedreira, and R. C. Souza, “Neural networks for short-term load forecasting: A review and evaluation,” *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, Feb. 2001, doi: 10.1109/59.910780.
- [8] A. Khotanzad, R. Afkhami-Rohani, T. L. Lu, A. Abaye, M. Davis, and D. J. Maratukulam, “ANNSTLF - A neural-network-based electric load forecasting system,” *IEEE Transactions on Neural Networks*, vol. 8, no. 4, pp. 835–846, 1997, doi: 10.1109/72.595881.
- [9] A. Khotanzad and R. Afkhami-Rohani, “ANNSTLF - Artificial neural network short-term load forecaster - generation three,” *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1413–1422, 1998, doi: 10.1109/59.736285.
- [10] H. S. Hippert and C. E. Pedreira, “Estimating temperature profiles for short-term load forecasting: Neural networks compared to linear models,” *IEE Proceedings: Generation, Transmission and Distribution*, vol. 151, no. 4, pp. 543–547, Jul. 2004, doi: 10.1049/IP-GTD:20040491.
- [11] A. Khotanzad, M. H. Davis, and Alireza Abaye, “An artificial neural network hourly temperature forecaster with applications in load forecasting,” *IEEE Transactions on Power Systems*, vol. 11, no. 2, pp. 870–876, 1996, doi: 10.1109/59.496168.
- [12] S. Fan, K. Methaprayoon, and W. J. Lee, “Multiregion load forecasting for system with large geographical area,” *IEEE Transactions on Industry Applications*, vol. 45, no. 4, pp. 1452–1459, 2009, doi: 10.1109/TIA.2009.2023569.
- [13] S. Rahman, “Formulation and Analysis of a Rule-Based Short-Term Load Forecasting Algorithm,” *Proceedings of the IEEE*, vol. 78, no. 5, pp. 805–816, 1990, doi: 10.1109/5.53400.
- [14] N. F. Hubele and C. S. Cheng, “Identification of seasonal short-term load forecasting models using statistical decision functions,” *IEEE Transactions on Power Systems*, vol. 5, no. 1, pp. 40–45, 1990, doi: 10.1109/59.49084.
- [15] G. Zhang, X. Bai, and Y. Wang, “Short-time multi-energy load forecasting method based on CNN-Seq2Seq model with attention mechanism,” *Machine Learning with Applications*, vol. 5, p. 100064, Sep. 2021, doi: 10.1016/J.MLWA.2021.100064.
- [16] S. He *et al.*, “A per-unit curve rotated decoupling method for CNN-TCN based day-ahead load forecasting,” *IET Generation, Transmission & Distribution*, vol. 15, no. 19, pp. 2773–2786, Oct. 2021, doi: 10.1049/GTD2.12214.
- [17] S. M. J. Jalali, S. Ahmadian, A. Khosravi, M. Shafie-khah, S. Nahavandi, and J. P. S. Catalao, “A Novel Evolutionary-based Deep Convolutional Neural Network Model for Intelligent Load Forecasting,” *IEEE Transactions on Industrial Informatics*, 2021, doi: 10.1109/TII.2021.3065718.
- [18] G. Dudek, P. Pelka, and S. Smyl, “A Hybrid Residual Dilated LSTM and Exponential Smoothing Model for Midterm Electric Load Forecasting,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021, doi: 10.1109/TNNLS.2020.3046629.

- [19] M. Khan, H. Wang, A. Riaz, A. Elfatyany, and S. Karim, “Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification,” *Journal of Supercomputing*, vol. 77, no. 7, pp. 7021–7045, Jul. 2021, doi: 10.1007/S11227-020-03560-Z/FIGURES/6.
- [20] “time series | statistics | Britannica.” <https://www.britannica.com/topic/time-series> (accessed May 10, 2022).
- [21] “Time series - Wikipedia.” https://en.wikipedia.org/wiki/Time_series (accessed May 10, 2022).
- [22] “What Is Time Series Forecasting?” <https://machinelearningmastery.com/time-series-forecasting/> (accessed May 10, 2022).
- [23] P. A. Gagniuc, “Markov chains : from theory to implementation and experimentation”.
- [24] “Stationary process - Wikipedia.” https://en.wikipedia.org/wiki/Stationary_process (accessed May 10, 2022).
- [25] “How to Remove Trends and Seasonality with a Difference Transform in Python.” <https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/> (accessed May 10, 2022).
- [26] “Stationarity in Time Series Analysis Explained using Python.” <https://blog.quantinsti.com/stationarity/> (accessed May 10, 2022).
- [27] “How to Check if Time Series Data is Stationary with Python.” <https://machinelearningmastery.com/time-series-data-stationary-python/> (accessed May 10, 2022).
- [28] “Augmented Dickey–Fuller test - Wikipedia.” https://en.wikipedia.org/wiki/Augmented_Dickey%E2%80%93Fuller_test (accessed May 10, 2022).
- [29] “What is a trend in time series? - GeeksforGeeks.” <https://www.geeksforgeeks.org/what-is-a-trend-in-time-series/> (accessed May 10, 2022).
- [30] “Mann Kendall Trend Test: Definition, Running the Test - Statistics How To.” <https://www.statisticshowto.com/mann-kendall-trend-test/> (accessed May 10, 2022).
- [31] “Seasonal Kendall Test - Statistics How To.” <https://www.statisticshowto.com/seasonal-kendall-test/> (accessed May 10, 2022).
- [32] “Time Series and Forecasting Time Series • A time series is a sequence of measurements over time, usually obtained at equally spaced intervals-Daily-Monthly-Quarterly-Yearly”.
- [33] “What Are Data Trends and Patterns, and How Do They Impact Business Decisions? - DATAVERSITY.” <https://www.dataversity.net/data-trends-patterns-impact-business-decisions/> (accessed May 10, 2022).
- [34] “Seasonality - Wikipedia.” <https://en.wikipedia.org/wiki/Seasonality> (accessed May 10, 2022).
- [35] “Seasonality Definition.” <https://www.investopedia.com/terms/s/seasonality.asp> (accessed May 10, 2022).
- [36] “seasonality - Barrons Dictionary - AllBusiness.com.” https://www.allbusiness.com/barrons_dictionary/dictionary-seasonality-4946957-1.html (accessed May 10, 2022).
- [37] “How to Identify and Remove Seasonality from Time Series Data with Python.” <https://machinelearningmastery.com/time-series-seasonality-with-python/> (accessed May 10, 2022).
- [38] “Chapter 2 Time series graphics | Forecasting: Principles and Practice (2nd ed).” <https://otexts.com/fpp2/graphics.html> (accessed May 10, 2022).
- [39] “Noise in time series data. There are two types of noise in time... | by Gajanan Kothawade | Medium.” <https://medium.com/@kothawadegs/noise-in-time-series-data-63c5450e10f9> (accessed May 10, 2022).
- [40] “White Noise Time Series with Python.” <https://machinelearningmastery.com/white-noise-time-series-python/> (accessed May 10, 2022).
- [41] “Lecture 11: White and red noise”.
- [42] M. Vilela *et al.*, “Fluctuation Analysis of Activity Biosensor Images for the Study of Information Flow in Signaling Pathways,” *Methods in Enzymology*, vol. 519, pp. 253–276, Jan. 2013, doi: 10.1016/B978-0-12-405539-1.00009-9.
- [43] “Autocorrelation - Wikipedia.” <https://en.wikipedia.org/wiki/Autocorrelation> (accessed May 10, 2022).
- [44] “A Gentle Introduction to Autocorrelation and Partial Autocorrelation.” <https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/> (accessed May 10, 2022).

- [45] “Partial autocorrelation function - Wikipedia.” https://en.wikipedia.org/wiki/Partial_autocorrelation_function (accessed May 10, 2022).
- [46] “Autoregressive model - Wikipedia.” https://en.wikipedia.org/wiki/Autoregressive_model (accessed May 10, 2022).
- [47] “Autoregressive Model: Definition & The AR Process - Statistics How To.” <https://www.statisticshowto.com/autoregressive-model/> (accessed May 10, 2022).
- [48] “Autoregressive–moving-average model - Wikipedia.” https://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model (accessed May 10, 2022).
- [49] J. Gurland and P. Whittle, “Hypothesis Testing in Time Series Analysis.,” *J Am Stat Assoc*, vol. 49, no. 265, p. 197, Mar. 1954, doi: 10.2307/2281054.
- [50] P. Whittle, *Hypothesis testing in time series analysis*. Uppsala: Almqvist & Wiksells boktr., 1951.
- [51] “P The Statistical Theory of Linear Systems”, Accessed: May 10, 2022. [Online]. Available: <https://epubs.siam.org/terms-privacy>
- [52] “Autoregressive integrated moving average - Wikipedia.” https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average (accessed May 10, 2022).
- [53] D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin, “Testing the null hypothesis of stationarity against the alternative of a unit root. How sure are we that economic time series have a unit root?,” *Journal of Econometrics*, vol. 54, no. 1–3, pp. 159–178, 1992, doi: 10.1016/0304-4076(92)90104-Y.
- [54] “8.9 Seasonal ARIMA models | Forecasting: Principles and Practice (2nd ed).” <https://otexts.com/fpp2/seasonal-arma.html> (accessed May 10, 2022).
- [55] “Machine learning - Wikipedia.” https://en.wikipedia.org/wiki/Machine_learning (accessed May 10, 2022).
- [56] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
- [57] “Machine Learning: What it is and why it matters | SAS.” https://www.sas.com/en_us/insights/analytics/machine-learning.html (accessed May 10, 2022).
- [58] “Linear regression - Wikipedia.” https://en.wikipedia.org/wiki/Linear_regression (accessed May 10, 2022).
- [59] D. A. Freedman, “Statistical models: Theory and practice,” *Statistical Models: Theory and Practice*, pp. 1–442, Jan. 2009, doi: 10.1017/CBO9780511815867.
- [60] “Methods of Multivariate Analysis - Alvin C. Rencher, William F. Christensen - Google Books.” https://books.google.com.bd/books?id=0g-PAuKub3QC&pg=PA19&redir_esc=y#v=onepage&q&f=false (accessed May 10, 2022).
- [61] H. L. Seal, “Studies in the History of Probability and Statistics. XV The historical development of the Gauss linear model,” *Biometrika*, vol. 54, no. 1–2, pp. 1–24, Jun. 1967, doi: 10.1093/BIOMET/54.1-2.1.
- [62] “Linear Regression for Machine Learning.” <https://machinelearningmastery.com/linear-regression-for-machine-learning/> (accessed May 10, 2022).
- [63] “Ordinary least squares - Wikipedia.” https://en.wikipedia.org/wiki/Ordinary_least_squares (accessed May 10, 2022).
- [64] “Gradient descent - Wikipedia.” https://en.wikipedia.org/wiki/Gradient_descent (accessed May 10, 2022).
- [65] “Lasso (statistics) - Wikipedia.” [https://en.wikipedia.org/wiki/Lasso_\(statistics\)](https://en.wikipedia.org/wiki/Lasso_(statistics)) (accessed May 10, 2022).
- [66] “Tikhonov regularization - Wikipedia.” https://en.wikipedia.org/wiki/Tikhonov_regularization (accessed May 10, 2022).
- [67] “| Xoriant.” <https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html> (accessed May 10, 2022).
- [68] “1.10. Decision Trees — scikit-learn 1.0.2 documentation.” <https://scikit-learn.org/stable/modules/tree.html> (accessed May 10, 2022).
- [69] “Decision Tree in Machine Learning | by Prince Yadav | Towards Data Science.” <https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96> (accessed May 10, 2022).
- [70] “Unlocking the True Power of Support Vector Regression | by Ashwin Raj | Towards Data Science.” <https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0> (accessed May 10, 2022).

- [71] “Support Vector Regression (SVR) | Analytics Vidhya.” <https://medium.com/analytics-vidhya/support-vector-regression-svr-model-a-regression-based-machine-learning-approach-f4641670c5bb> (accessed May 10, 2022).
- [72] A. Singh, V. Kotiyal, S. Sharma, J. Nagar, and C. C. Lee, “A Machine Learning Approach to Predict the Average Localization Error with Applications to Wireless Sensor Networks,” *IEEE Access*, vol. 8, pp. 208253–208263, 2020, doi: 10.1109/ACCESS.2020.3038645.
- [73] “sklearn.svm.SVR — scikit-learn 1.0.2 documentation.” <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html> (accessed May 10, 2022).
- [74] “K-Nearest Neighbor. A complete explanation of K-NN | by Antony Christopher | The Startup | Medium.” <https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4> (accessed May 10, 2022).
- [75] “k-nearest neighbors algorithm - Wikipedia.” https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm (accessed May 10, 2022).
- [76] S. M. Pirayonesi and T. E. El-Diraby, “Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems,” *Journal of Transportation Engineering, Part B: Pavements*, vol. 146, no. 2, p. 04020022, Apr. 2020, doi: 10.1061/JPEODX.0000175.
- [77] Trevor. Hastie, Robert. Tibshirani, and J. H. (Jerome H.) Friedman, *The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations*. New York: Springer, 2001.
- [78] P. A. Jaskowiak, P. A. Jaskowiak, and R. J. G. B. Campello, “Comparing Correlation Coefficients as Dissimilarity Measures for Cancer Classification in Gene Expression Data,” *BRAZILIAN SYMPOSIUM ON BIOINFORMATICS (BSB)*, pp. 1--8, 2011, Accessed: May 10, 2022. [Online]. Available: <https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.208.993>
- [79] D. Coomans and D. L. Massart, “Alternative k-nearest neighbour rules in supervised pattern recognition : Part 1. k-Nearest neighbour classification by using alternative voting rules,” *Analytica Chimica Acta*, vol. 136, no. C, pp. 15–27, Jan. 1982, doi: 10.1016/S0003-2670(01)95359-0.
- [80] “Wayback Machine.” <https://web.archive.org/web/20150119081741/http://oz.berkeley.edu/~breiman/arcall96.pdf> (accessed May 10, 2022).
- [81] “Boosting (machine learning) - Wikipedia.” [https://en.wikipedia.org/wiki/Boosting_\(machine_learning\)](https://en.wikipedia.org/wiki/Boosting_(machine_learning)) (accessed May 10, 2022).
- [82] Z. Zhou, “Boosting 25 Years,” *Else*, 2020.
- [83] “Boosting in Machine Learning | Boosting and AdaBoost - GeeksforGeeks.” <https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/> (accessed May 10, 2022).
- [84] “XGBoost - Wikipedia.” <https://en.wikipedia.org/wiki/XGBoost> (accessed May 10, 2022).
- [85] “A Gentle Introduction to XGBoost for Applied Machine Learning.” <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/> (accessed May 10, 2022).
- [86] “LightGBM (Light Gradient Boosting Machine) - GeeksforGeeks.” <https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/> (accessed May 10, 2022).
- [87] “LGB, the winning Gradient Boosting model ★ Code A Star.” <https://www.codeastar.com/lgb-winning-gradient-boosting-model/> (accessed May 10, 2022).
- [88] “A Primer to Ensemble Learning – Bagging and Boosting .” <https://analyticsindiamag.com/primer-ensemble-learning-bagging-boosting/> (accessed May 10, 2022).
- [89] “Bagging and Random Forest Ensemble Algorithms for Machine Learning.” <https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/> (accessed May 10, 2022).
- [90] “Random Forest | Introduction to Random Forest Algorithm.” <https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/> (accessed May 10, 2022).
- [91] “What is Deep Learning?” <https://machinelearningmastery.com/what-is-deep-learning/> (accessed May 10, 2022).
- [92] “Deep learning - Wikipedia.” https://en.wikipedia.org/wiki/Deep_learning (accessed May 10, 2022).

- [93] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature* 2015 521:7553, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
- [94] “What Is Deep Learning? | How It Works, Techniques & Applications - MATLAB & Simulink.” <https://www.mathworks.com/discovery/deep-learning.html> (accessed May 10, 2022).
- [95] “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science.” <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (accessed May 10, 2022).
- [96] M. v. Valueva, N. N. Nagornov, P. A. Lyakhov, G. v. Valuev, and N. I. Chervyakov, “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,” *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, Nov. 2020, doi: 10.1016/J.MATCOM.2020.04.031.
- [97] J. Tanida, K. Itoh, W. Zhang, and Y. Ichioka, “Parallel distributed processing model with local space-invariant interconnections and its optical architecture,” *Applied Optics, Vol. 29, Issue 32, pp. 4790-4797*, vol. 29, no. 32, pp. 4790–4797, Nov. 1990, doi: 10.1364/AO.29.004790.
- [98] “Convolutional neural network - Wikipedia.” https://en.wikipedia.org/wiki/Convolutional_neural_network (accessed May 10, 2022).
- [99] R. Venkatesan and B. Li, “Convolutional Neural Networks in Visual Computing : A Concise Guide,” *Convolutional Neural Networks in Visual Computing*, Oct. 2017, doi: 10.4324/9781315154282.
- [100] R. Venkatesan and B. Li, “Convolutional neural networks in visual computing : a concise guide,” p. 168.
- [101] D. C. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. “Urgen Schmidhuber, “Flexible, High Performance Convolutional Neural Networks for Image Classification”.
- [102] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”.
- [103] D. Cireşan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3642–3649, 2012, doi: 10.1109/CVPR.2012.6248110.
- [104] “MNIST Demos on Yann LeCun’s website.” <http://yann.lecun.com/exdb/lenet/> (accessed May 10, 2022).
- [105] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” *Heliyon*, vol. 4, no. 11, p. e00938, Nov. 2018, doi: 10.1016/J.HELIYON.2018.E00938/ATTACHMENT/7F6968D1-2173-4A69-BC61-62AD41135D5C/MMC2.
- [106] A. Tealab, “Time series forecasting using artificial neural networks methodologies: A systematic review,” *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 334–340, Dec. 2018, doi: 10.1016/J.FCIJ.2018.10.003.
- [107] X. Li and X. Wu, “Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition,” Oct. 2014, doi: 10.48550/arxiv.1410.4281.
- [108] H. H. Sak, A. Senior, and B. Google, “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling”.
- [109] “Turing Machines are Recurrent Neural Networks.” <http://users.ics.aalto.fi/tho/stes/step96/hyotyniemi1/> (accessed May 10, 2022).
- [110] A. Milos Miljanovic, “Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction.”
- [111] “LSTM Networks | A Detailed Explanation | Towards Data Science.” <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9> (accessed May 10, 2022).
- [112] “Long short-term memory - Wikipedia.” https://en.wikipedia.org/wiki/Long_short-term_memory (accessed May 10, 2022).
- [113] “LSTM can Solve Hard Long Time Lag Problems.” <https://papers.nips.cc/paper/1996/hash/a4d2f0d23dcc84ce983ff9157f8b7f88-Abstract.html> (accessed May 10, 2022).
- [114] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to Forget: Continual Prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000, doi: 10.1162/089976600300015015.
- [115] Ovidiu. Calin, “Deep learning architectures : a mathematical approach,” p. 760.

- [116] “What is a Gated Recurrent Unit (GRU)? - Definition from Techopedia.”
<https://www.techopedia.com/definition/33283/gated-recurrent-unit-gru> (accessed May 10, 2022).
- [117] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *IEE Conference Publication*, vol. 2, no. 470, pp. 850–855, 1999, doi: 10.1049/CP:19991218.
- [118] “Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano – WildML.”
<https://web.archive.org/web/20211110112626/http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/> (accessed May 10, 2022).
- [119] “Understanding GRU Networks In 2021.” <https://www.jigsawacademy.com/blogs/data-science/gru/> (accessed May 10, 2022).
- [120] M. Massaoudi, S. S. Refaat, I. Chihi, M. Trabelsi, F. S. Oueslati, and H. Abu-Rub, “A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting,” *Energy*, vol. 214, p. 118874, Jan. 2021, doi: 10.1016/J.ENERGY.2020.118874.
- [121] S. H. Rafi, N. Al-Masood, S. R. Deeba, and E. Hossain, “A short-term load forecasting method using integrated CNN and LSTM network,” *IEEE Access*, vol. 9, pp. 32436–32448, 2021, doi: 10.1109/ACCESS.2021.3060654.