Electricity Load Forecasting using Long Short-Term Memory

Dame Hirpa Olika

A Thesis Submitted to Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2022

Adama, Ethiopia

Electricity Load Forecasting using Long Short-Term Memory

Dame Hirpa Olika

Advisor: Dr. Tilahun Melak

A Thesis Submitted to Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2022

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled "**Electricity Load Forecasting using Long Short-Term Memory**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Dame Hirpa Olika	_____	_____
Name of student	Signature	Date

Recommendation of Advisors/ Supervisors

I/we, the advisor(s) of this thesis, hereby certify that I/we have read the revised version of the thesis entitled "**Electricity Load Forecasting using Long Short-Term Memory**" prepared under my/our guidance by Dame Hirpa Olika submitted in partial fulfillment of the requirements for the degree of Mater's of Science in **Computer Science and Engineering**. Therefore, I/we recommend the submission of revised version of the thesis to the department following the applicable procedures.

Major Advisor/Supervisor	Signature	Date
Co-advisor/Co-supervisor	Signature	Date

Approval Sheet

I/we, the advisors of the thesis entitled "**Electricity Load Forecasting using Long Short-Term Memory**" and developed by Dame Hirpa Olika, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Advisor/Supervisor	Signature	Date
Co-advisor/ Co-supervisor	Signature	Date

Approval of Board of Reviewers

We, the undersigned, members of the Board of Examiners of the thesis by Dame Hirpa have read and evaluated the thesis entitled "**Electricity Load Forecasting using Long Short-Term Memory**" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGMENT

First, I would like to thank Adama Science and Technology University for giving me this post-graduate scholarship opportunity. My deepest gratitude goes to my advisor, Dr. Tilahun Melak, for his kind attitude, helpful comments, and for pointing me in the right direction. I would like to acknowledge Ethiopian Electric Utility employees, especially Mr. Abay Admasu and Mr. Bezakulu Meried, for their cooperation in providing me with the dataset for the study and for being willing to answer all my questions regarding the dataset and the study area of the research.

I am extremely grateful to my parents for always being there for me, encouraging me and supporting me in every way they can. I am also thankful to my sisters for their invaluable support. I would like to express my appreciation to my friends for helping me and pushing me. Last but not least, I would like to acknowledge members of Smart Software SIG and my classmates for creating a supportive learning environment.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF CODE FRAGMENTS

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
ARCH	Autoregressive Conditional Heteroscedastic
CNN	Convolutional Neural Network
DL	Deep Learning
DWT	Discrete Wavelet Transform
EPLF	Electric Power Load Forecasting
FFNN	Feed Forward Neural Network
KW	Kilowatt
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
RBFNN	Radial Basis Function Neural Network
RNN	Recurrent Neural Network
TSF	Time Series Forecasting

ABSTRACT

Nowadays, the demand for electric energy is increasing rapidly while the mismatch between electricity demand, production, and distribution is growing. This leads to continuous power outages, which in turn affect the end customers. On the other hand, a sudden change in electricity flow affects electricity power distribution stations and other electricity distribution infrastructures. Thus, knowing electricity demand at a specific time and the nature of how electricity demand changes beforehand help to properly plan and manage future demands. Electricity load forecasting also help electric energy purchasing, transmission, and distribution planning, for demand-side management operation and maintenance. Many previous studies focused on external factors for electricity load forecasting. Collecting this kind of data requires various types of sensors, which increases the cost of time and resources. But also, in some cases, these types of data may not exist at all. Electricity load forecasting can be done at various levels ranging from generation to end users. Previous research in substation load forecasting used an Artificial Neural Network, which does not capture temporal dependencies between data points unless it is specified explicitly. The dataset we used contains 79,969 records of historical electric power load data measured every 15 minutes in the period from 1/1/2020 to 4/13/2022 from Gofa substation in Addis Ababa. This paper proposes Long Short-Term Memory architecture to forecast substation load demand for Gofa substation. The main objective of this study is to improve performance of electricity load forecasting at substation level using Long Short-Term Memory. For the purpose of comparison, various deep learning models are implemented. Mean Squared Error and Root Mean Squared Error are used to test the accuracy of the models. The result shows that the Long Short-Term Memory model outperforms other deep learning models such as Artificial Neural Network and Recurrent Neural Network, which have been used in previous works.

***Keywords**: Electricity load forecasting, Demand, Substation Load, Long Short-Term Memory*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

Every commercial electric power provider has a number of strategic goals. Giving end users safe and reliable electricity is one of these goals (Almeshaiei & Soltan, 2011).

Forecasting is a technique that uses historical data as input to make accurate predictions about the direction of future trends (Alicia Tuovila, 2022). To manage and plan for future demand, it is important to predict future trends. Demand forecasting enables informed and efficient responses to demand. Hence, forecasting is done to optimize cost and provide necessary input that helps make decisions (Gebreyohans et al., 2018).

Electric Load Forecasting is a crucial step in the organization of the energy sector and the management of electric power systems. Accurate predictions result in significant operational and maintenance cost reductions; increased power supply and delivery system reliability, and wise development decisions (Almeshaiei & Soltan, 2011). Electricity load forecasting is a difficult task because of the nonlinear and nonstationary characteristics of electric loads, which cause electric load signals to be highly unpredictable. (Zor et al., 2017)

Load forecasting is used by electricity market operators for a variety of purposes, including making decisions about unit commitment, reducing spinning reserve capacity, and properly scheduling device maintenance plans in order to ultimately protect the system's security and achieve the highest level of supply reliability while minimizing economic cost.(Setiawan et al., 2009)

Depending on the time frame, which explains how much the prediction is made for, electricity load prediction can be classified into 4 categories. These time frames can be slightly different based on various researches(Kuster et al., 2017). Very short-term load forecasting accounts for one to a few hours, and it is used for electrical power buying and selling contracts, power tie-line operations, and electrical power security systems. Short-term electrical load forecasting covers a

time frame of one day up to several days. It is used for utilizing the existing electrical power in the best possible manner, balancing electricity supply, optimally coordinating generating units, and analyzing a steady state for economic dispatch. Medium-term electrical load forecasting accounts for one to 12 months, and the purpose of this electrical load forecasting is to properly plan for supervising maintenance schedules and tests, power commissioning activities, and electrical power outage times of plants and major equipment. Long-term electrical load forecasting basically covers a period of one up to 20 years or more. This type of electricity load forecasting is used for electrical power system expansion planning, mainly for constructing generation, transmission, and distribution facilities, capital investment return studies, revenue analysis, and other long-term financial planning.(Gebreyohans et al., 2018).

Short-term electric power load forecasting is a very critical issue for proper operation and dispatch of power systems in order to prevent frequent power failures. It is an important prerequisite for economic dispatch of generation units in power plants. The short-term load forecasting technique must be accurate so that it helps users select a more optimal power load scheme that, minimize production costs, and optimize the resources of the power system.

Electricity load forecasting can be done at different levels of electric power distribution, such as substation level or end-user level. The substation loading is highly correlated with the customers served. Forecasting at the substation level supports distribution operations. It is recognized that substation load demand varies with load behavior(Chen et al., 1996) .

Forecasting at the substation level has a variety of advantages. It is a crucial task for system operators in order to run the system dependably and effectively. Electric power companies estimate the load to maintain the right balance between supply and demand. Power companies use forecasting of electric power loads as one of their primary planning tools for routine operations and facility expansion (Venkataramana Veeramsetty, 2020)

Over the years, many different forecasting models have been applied for electricity load predictions, such as multivariate and multiple, SVM, time series, including Autoregressive Integrated Moving Average (ARIMA) and the Autoregressive Moving Average (ARMA). Equally, artificial neural networks (ANN) have become widely used for prediction scenarios. ANN has been used for various tasks such as short-term load forecasting (STLF) in micro grids,

optimization scenarios at building level, and long-term timeframe scenarios to determine annual electricity load of a region, district, or building.

Therefore, in this research, we study electricity load forecasting. Select methods to forecast electricity at substation level. Compare various deep learning models in order to choose the best performing model for the Gofa substation dataset. Finally, we propose an accurate deep learning model to forecast the electricity load of Gofa substation.

1.2. Motivation of the Study

Ethiopia, with a population of over 100 million, has a final energy consumption of around 40,000 GWh. One of the energy sources is electricity, contributing 9000 GWh every year. Electricity demand is expected to rise by a rate of 10–14% per year till 2037. Today, only 27 % of the rural population has access to electricity grid (International Energy Agency, 2021). As population and economy grows the demand to utilize electricity energy increases.

Load prediction models help manage and plan for future electric energy load. They are also used for power market operation, power market design, power systems planning, power systems control, and security of procurement.

When forecasting in the electricity domain, higher forecasts than the real values lead to misinformation and consequently unnecessary investment decisions for energy investors. Lower forecasts slow new energy investments and, in the long run, cause power outages. Previous researches used either additional variables and/or traditional deep learning models to forecast electricity load at substation level. Collecting additional variables is difficult, sometimes impossible. Traditional DL models used like ANN do not incorporate all the necessary information; hence they are not able to provide the best forecasting accuracy. Thus, building a forecasting model that accurately forecasts future electricity loads by capturing all the available information from the given historical electricity load dataset is needed.

1.3. Statement of the Problem

Electric energy, by itself, cannot be stored in large quantities. Hence, it is extremely important that the amount of electricity necessary to cover the demand is generated as approximately as

possible(Torres et al., 2022). Being aware of the future ahead is important for various decision-making tasks. An incorrect prediction could result in either an overestimation or an underestimation of electricity load. Overestimation could lead to startup of too many units, which results in an excess supply and reserves, increasing costs. An underestimation of the electricity load would result in failures to gather sufficient provisions and higher complementary service costs.(Zor et al., 2017) (Setiawan et al., 2009).

Electricity load forecasting uses historical load data with the assumption that past patterns in the data will continue in the future. Hence, the performance of an electricity load forecasting model is highly dependent on how well the model captures the patterns in the given dataset. So far, only Artificial Neural Network (ANN) has been applied to forecast substation load data. ANN is not a neural network mainly developed for sequence data. ANN does not capture the temporal dependencies in time series data well unless specified directly. Its performance depends on whether the set of inputs and output instances are given correctly. Neural networks like Recurrent Neural Network (RNN) are better options than ANN because they are made to capture the temporal dependency between data points. They work best when the valuable dependencies in the dataset are in the short range or within neighboring data points. However, in order to accurately forecast the power load of a substation, a forecasting model should incorporate all possible dependencies between all data points, regardless of how far apart the data points are. Long Short-Term Memory (LSTM) is a variant of RNN developed to solve the vanishing gradient problem of RNN. LSTM can capture both short-term and long-term dependencies while systematically forgoing unnecessary information.

The nature of electricity load is different for different places. Studying how electricity load changes for specific dataset and assigning necessary hyperparameters while building a new architecture is necessary. It is needed to capture the underlying patterns of the historical time series dataset so that they can be extrapolated to the future effectively. So far, studies made in Ethiopia have used a small dataset (Gebrehiwot Gebreyohans, Nitin Kumar Saxena,Ashwani Kumar, 2018) and used models like ANN. Therefore, this research experiments and finds a deep learning model that can capture trends from substation data effectively by collecting and utilizing enough amounts of electricity load data from Ethiopia to improve the accuracy of prediction.

1.4. Research Questions

This research aims to answer the following questions:

RQ1. Can using historical electricity load dataset without external factors achieve comparable/better performance in forecasting as with external factors?

RQ2. Can Long Short-Term Memory improve the performance of electricity load forecasting at substation level?

RQ3. How to optimize the selected model's hyperparameters to improve the performance of the proposed model?

1.5. Objective of the Study

1.5.1. General Objective

The general objective of the study is to build a model that improves electricity load forecasting performance, at substation level.

1.5.2. Specific Objectives

- Identify deep learning models for electricity load forecasting
- Collect data from Ethiopian Electric Utility
- Perform data analysis and preprocessing on collected historical electricity load dataset
- Train each of the electricity load forecasting models with the dataset
- Apply test set and optimize the performance of the models
- Evaluate and analyze the results and chose the best performing model

1.6. Significance of the Study

This study can be used by Ethiopian Electric Utility to forecast substation load and can also be integrated to the smart grid system by accepting additional data and forecasting in the future. Therefore the significance of the thesis is;

- proposes an electricity load forecasting model using a Long Short-Term Memory approach

- Comparison of the proposed model with other deep learning models such as RNN and ANN
- used Gofa substation's electricity power load database to experiment and evaluate the models

1.7. Scope of the Study

In this research, we studied literature review on electric load prediction, chose deep learning algorithms to train and test on, collected dataset, trained the DL models, chose the better performing electricity load forecasting model for Gofa substation.

1.8. Limitation of the Study

Due to lack of continuous data for a long period of time, we took a portion of the data available with minimum missing data. We could have been able to forecast for a longer horizon if continuous data over a long period of time had existed. Getting data on other variables affecting electricity load at the same frequency as the historical dataset could not be achieved either.

1.9. Thesis Organization

The rest of the thesis is organized in the following way: A literature review of existing papers is presented in chapter two. Chapter three presents the methodology of the research. Next, the proposed model is explained in chapter Four. Chapter five presents the implementation of the LSTM model. In chapter six, results of the experiments and how they answered the research questions are discussed. Finally, in chapter seven, the research is concluded and suggestions are given for future work.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Introduction

Electric power is the rate at which energy is transferred by an electrical circuit. It is a measure of how much energy is consumed over a certain period of time. The SI unit is watt, which equals one joule per second. An electrical substation is part of an electrical generation, transmission, and distribution system. Substations can perform a number of crucial tasks, including converting voltage from high to low or vice versa. Between the generating station and the consumer, electric power flows through several substations at different voltage levels.

2.2. Electricity Load Forecasting

One of the crucial steps in the organization and management of electric utilities is load forecasting. A system operator can efficiently plan a spinning reserve allocation with the use of load forecasting. The forecasts might be short term, medium term or long term, each done for different purposes. Maximizing both short-term and long-term operational system performance is the fundamental challenge facing electric utilities as system operators (Metaxiotis et al., 2003). Estimates of the short term and long term variations of power demand, including changes in load patterns, are crucial for businesses or governmental organizations in the energy sector. Useful data can be gathered to identify vulnerable circumstances beforehand, if applied to the challenge of system security assessment.

2.2.1. Short Term Electricity Load Forecasting (STELF)

The objective of short term load forecasting is to predict future electricity demand based on historical data. Short term load forecast is used for energy transfer scheduling and unit commitment. Short term prediction plays a large role in utility operations as load control measures become more prevalent. In the past, there have been many different STELF techniques applied. Traditionally, STELF approaches employ standard statistical analysis, regression, and smoothing procedures. Peak load models and load shape models are two of the statistical models employed for STELF. The load shape models rely on methods for time series analysis. Also,

time-series statistical analysis techniques such as the autoregressive integrated moving average (ARIMA) have been used to treat forecasts in this field (Lopez-Martin et al., 2021).

2.3. Time Series Forecasting

Time series data is data with one or more variables that is recorded at a specific interval of time. Time series forecasting (TSF) is the process of using historical or past series of data points to predict future data points. TSF offers significant advantages in a wide range of practical problems with a temporal component. Time series forecasting is essential in many fields, including meteorology, energy load, financial indices, retail sales, health monitoring, anomaly detection, and traffic prediction. Time series data has unique characteristics that make analysis challenging because the observations are ordered in time. TSF is an essential field in data mining because of its complexity. The trend of the series, seasonal changes, and correlation between adjacent observed values are some of the components that TSF models must take into account (Pedro Lara-Benitez, 2020). Generally, time series forecasting techniques fall into the two main categories of statistical and computational intelligence methods.

2.3.1. Statistical Models

Prior to the rise of data mining tools, the traditional approaches to time series forecasting were mostly based on statistical models, such as exponential smoothing (ETS) and Box-Jenkins techniques like ARIMA. These models, which have been widely utilized for predicting jobs during the past decades, focus on creating linear functions from recent past observations to offer future forecasts. But when used without taking into account the theoretical conditions of stationary and ergodicity as well as the necessary preprocessing, these statistical methods frequently fail. For instance, the time series in an ARIMA model must be made stationary using a number of differentiating transformations. Another problem is that these models can only be used in circumstances where there is a substantial amount of historical data and an understandable structure. These models frequently fall short in their capacity to adequately extract the underlying features and patterns when data availability for a single series is limited. Furthermore, since these techniques build a model specifically for each time series, it is impossible to apply the knowledge to similar situations. As a result, it would be computationally prohibitive to employ these techniques to handle enormous amounts of data (Pedro Lara-Benitez,

2020). Statistical time series forecasting methods such as ARIMA make the assumption that the time series is stationary. However, the majority of time series data from the real world also includes nonlinear elements. Several nonlinear statistical techniques, such as the autoregressive conditional heteroscedastic (ARCH) model and Generic Autoregressive Conditional Heteroscedastic regression (GARCH), have been developed to address the forecasting of time series with nonlinear patterns. These models come in a wide variety, each of which is best suited to simulating a particular nonlinearity. This makes the process of choosing a suitable model for time series more difficult.

2.3.2. Deep Learning Techniques

Deep neural networks have successfully been applied to address time series forecasting problems, which is a very important topic in data mining. They have proved to be an effective solution given their capacity to automatically learn the temporal dependencies present in time series (Pedro Lara-Benitez, 2020).

For several machine learning issues, including time series forecasting, deep learning (DL) algorithms have recently gained popularity. Deep neural networks have demonstrated a tremendous ability to map complicated non-linear feature interactions, in contrast to traditional statistical-based models that can only represent linear correlations in data. Modern neural systems' success is based on their deep architecture, which stacks multiple layers and densely connects many neurons (Pedro Lara-Benitez, 2020).

ANN-MLP

An Artificial neural network (ANN) is a system of hardware and/or software patterned after the operation of neurons in the human brain. Multi-layer Perceptron (MLP), a popular form of ANN with more than two layers, is trained using the backpropagation learning process. The number of hidden layers, the number of neurons in each layer, and the weights between neurons in various layers all affect how well MLP models function(Abbasimehr et al., 2020). The temporal order of a time series cannot be captured by feed-forward neural networks like MLP since they consider each input independently, despite the fact that they have been successfully used in many situations. Performance in time series forecasting is constrained by disregarding the temporal order of input windows, particularly when dealing with instances of various lengths that change

dynamically. As a result, interest in more specialized models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) began to grow.

Recurrent Neural Networks (RNN)

RNN was introduced as a variant of ANN for time-dependent data. Unlike MLPs that ignore the time relationships within the input data, RNNs connect each time step with the previous ones to model the temporal dependency of the data, providing RNN native support for sequence data. RNN's connections between nodes create a cycle that permits signals to travel in diverse directions, unlike feedforward ANNs. One observation is seen by the network at a time, and it can learn about the prior observations and how applicable the current observation is to forecasting. Through this process, the network not only picks up internal patterns between observations of the sequence, but also patterns between input and output. By recording the activations from every time step, RNNs operate as a short-term memory. This qualifies it as a method for handling sequence data. The vanishing and exploding gradient problem is an RNN's vulnerability.

Elman Recurrent Neural Networks (ERNN)

ERNN set out to solve the issue of handling time patterns in data. In order to connect the output of the hidden layer to the input of the hidden layer for the following time step, ERNN converted the feedforward hidden layer to a recurrent layer. The application of ERNN confronts two fundamental issues, notwithstanding the effectiveness of Elman's approximation: the weights may begin to oscillate (exploding gradient problem), or it takes too long to compute long-term patterns (vanishing gradient problem).

Long Short-Term Memory Networks (LSTM)

LSTM was introduced as a solution to ERINN's issues. Without overlooking the short-term trends, LSTMs model temporal dependencies over a wider range of time periods. The hidden layer, also referred to as the LSTM memory cell, distinguishes LSTM networks from ERNN. To avoid irrelevant perturbations from changing the memory units, LSTM cells use a multiplicative input gate to regulate the memory units. Similar to how a multiplicative output shields adjacent cells from present memory disturbances Later, a forget gate was added to the LSTM memory cell

by another researcher. The LSTM can learn to reset memory contents as they become obsolete thanks to this gate.

Convolutional Neural Networks (CNN)

CNN is a family of deep architectures that were originally designed for computer vision tasks. The convolutional operation, a sliding filter that produces feature maps and seeks to capture repeating patterns in different areas of the input, is how the model learns to extract useful features from the raw data. Because the features are extracted independently of where they are in the input, this feature extraction procedure gives CNNs a crucial property known as distortion invariance. CNNs are well suited for handling one-dimensional data, such as time series, because of these features. Sequence data can be compared to a one-dimensional image, from which features can be extracted using a convolutional technique (Pedro Lara-Benitez, 2020).

2.4. Review of Related Works

Since a long time ago, researchers have been working on electricity load forecasting, using both static approaches like ARIMA and deep learning techniques. Below, we discussed some related work on electricity load forecasting.

(Venkataramana Veeramsetty, 2020) employed artificial neural network and machine learning to predict the load at a specific time of day on a power substation. A back propagation algorithm based on ANN Levenberg-Marquardt is employed. Data from the last 3 hours of loading as well as data from the same hour for the previous 4 days are used as input. In the MATLAB environment, the suggested ANN model has been put into practice and tested. The model's performance in terms of error uncertainty and regression was assessed and compared to that of other models.

(Gebreyohans et al., 2018) attempted to address Wolaita Sodo town’s power imbalance problem by implementing a hybrid model to forecast the town’s future electric load. In the paper, they tested 2 hybrid models: multivariable linear regression with an artificial neural network and a hybrid model of multivariable linear regression with an adaptive neuro-fuzzy inference system. They used past electrical load, population growth, and gross domestic product data from nine years of data to forecast the electrical load of the town for the six years ahead. These different

models were compared based on some error performance criteria. The forecasted result has shown that the electrical load consumption of the town would be likely to increase from 13.568 MW for the year of 2017 to 22 MW in 2023. They concluded that out of the two major factors, population has more share than gross domestic product for the increase of electrical power in the town each year.

(Gabreyohannes, 2010) analyzed and forecasted the residential electricity consumption in Ethiopia. For comparison purposes, the application was also extended to standard linear models. In this study, they adopt non-linear models that allow for state dependent or regime switching behavior. They trained both self-exciting threshold autoregressive (SETAR) model and the smooth transition regression (STR) model. Another objective of this study was to utilize STR models for identifying the determinants of changing demand growth, and describing both linear and nonlinear relationships that may exist between electricity demand and these determinants. For SETAR models they considered a series of seasonally unadjusted electricity consumption data points at monthly frequency covering the period July 1984 – June 2002. For STR electricity price (ex-post average price computed as total expenditure on electricity divided by total kilowatt-hours consumed), alternative energy source price (Ex: kerosene price), urban population, household (or private) consumption expenditure as a proxy for household income. Because of the unavailability of monthly data on some of their explanatory variables, the analysis was done on quarterly figures. The SETAR model was found out to be relatively better than the linear autoregressive model in out-of-sample point and interval (density) forecasts. Results from their STR model showed that the residual variance of the fitted STR model was only about 65.7% of that of the linear ARX model. Thus, they concluded that the inclusion of the nonlinear part, which basically accounts for the arrival of extreme price events, leads to improvements in the explanatory abilities of the model for electricity consumption in Ethiopia. They used RMSE, MAD, and MAPE as their evaluation metrics.

(Potapov et al., 2018) applied methodology forecasting using neural networks for building predictive models for LLC “Omsk Energy Retail Company”. The Holt-Winters model, the ARIMA, neural networks, temperature, and wind index were the approaches employed in the study. It took into account how predictive models are built as well as how they have developed since the electricity market was opened. The "Omsk Energy Retail Company" created a neural

network-based forecasting method that accounts for the temperature, wind index, and allocation of frequent day types based on electrical load.

(Widyaning Chandramitasari, Bobby Kurniawan, Shigeru Fujimura, 2018) Forecasted the electricity consumption in the manufacturing company for each 30-minute in the next day to prevent the lack of electricity supply from a power supply company. To carry out the task of forecasting electricity, they suggested a model of deep learning neural network with an approach combining Long Short-Term Memory (LSTM) and Feed Forward Neural Network (FFNN). They used FFNN coupled with additional information represented by one-hot encoding structure to improve forecasting performance and used LSTM to do time series forecasting using historical data of electricity consumption.

(Lin Wang, Shengxiang Lv, Yu-Rong Zeng, 2018) Proposes a model named ESN-DE using an improved echo state network for forecasting electricity energy consumption. The three key parameters of the echo state network are searched for the best values using the differential evolution technique. To verify the applicability and accuracy of the ESN-DE model, two comparison instances and an extended example are employed. They came to the conclusion that the developed ESN-DE, due to its simplicity and stability, is a possible contender for accurate forecasting of electricity energy consumption.

(Khadeejah Adebisi Abdulsalama and Olubayo Moses Babatunde, 2019) Presented an Artificial Neural Network based method for Electrical Energy Demand Forecasting using a case study of Lagos state, Nigeria. They used a curve fitting toolbox to show the dataset is non-linear hence using an artificial neural network is justified.

In the study (Emre Akarslan,Fatih Onur Hocaoglu, 2018) campus area of the Afyon Kocatepe University is considered as a micro grid and the hourly electricity demand of the campus area is forecasted. They took into account loads with various characteristics from various buildings, such as homes, schools, labs, research facilities, hospitals, and public buildings. Artificial neural networks are used to forecast the demand for power. The network takes into account the current season, time, and electricity use to anticipate the need for the following hour.

(Shao et al., 2020) presented a novel hybrid deep model for multiple forecasts by combining Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM) algorithms. In

contrast to the traditional stacked CNN-LSTM model, the suggested hybrid model extracts features from CNN and LSTM simultaneously, resulting in more robust features with less loss of original data. Six statistical components are integrated with the features recovered by CNN and LSTM to provide complete features. Therefore, multi-scale and multi-domain features are combined to form comprehensive features.

Table 2.1 Summary of related works

S/N	Title	Author	Objective	Gap	Algorithm
1	Electric power load forecasting on a 33_11 KV substation using artificial neural networks	(Venkataramana Veeramsetty, 2020)	To forecast the load at a particular hour of the day on electric power substation	The model is unable to capture the temporal order of a time series; rather, a static set of timestamps are assumed to have a relation	ANN LevenbergMarquardt based back propagation

2	Long-Term Electrical Load Forecasting of Wolaita Sodo Town, Ethiopia Using Hybrid Model Approaches	(Gebrehiwot Gebreyohans, Nitin Kumar Saxena,Ashwani Kumar, 2018)	To forecast long-term electrical load for Wolaita Sodo Town	Very small data(9 years/9 timestamps) , thus, difficult to generalize from such small data	Hybrid model of multiple linear regression (MLR) and with artificial neural network (ANN) and hybrid model of multiple linear regression (MLR) with Adaptive Neuro-fuzzy inference system (ANFIS)
3	A nonlinear approach to modeling the residential electricity consumption in Ethiopia	(Gabreyohannes, 2009)	To model, analyze and forecast the residential electricity consumption in Ethiopia	Model predicts for only residential electricity consumption	- SETAR - STR

4	Short-Term Forecast of Electricity Load for LLC "Omsk Energy Retail Company" Using Neural Network	(Viktor Potapov,Rustam Khamitov,Vladimir Makarov,Aleksandr Gritsay,Igor Chervenchuk,Dmitry Tyunkov, 2018)	Applying methodology forecasting using neural networks for building predictive models for LLC "Omsk Energy Retail Company".	Compared with traditional/statistical models	Neural network, Holt-Winters model, ARIMA
5	Building Deep Neural Network Model for Short Term Electricity Consumption Forecasting	(Widyaning Chandramitasari, Bobby Kurniawan, Shigeru Fujimura, 2018)	To forecast the electricity consumption in the manufacturing company for each 30-minutes in the next day.	LSTM covers for loss of neighboring information	Long Short-Term Memory (LSTM) and Feed Forward Neural Network (FFNN)

6	Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm	(Lin Wang, Huanling Hu, Xue-Yi Ai, Hua Liu, 2018)	To propose an effective and stable model named ESN-DE using an improved echo state network for forecasting electricity energy consumption	Does not capture long term dependency	ESN-DE, an improved echo state network
7	Electrical energy demand forecasting model using artificial neural network: A case study of Lagos State, Nigeria	(Khadeejah Adebisi Abdulsalama and Olubayo Moses Babatunde, 2019)	To forecast electrical energy demand for Lagos, Nigeria	Small dataset	ANN

8	Electricity Demand Forecasting of a Micro Grid Using ANN	(Emre Akarslan,Fatih Onur Hocaoglu, 2018)	To forecast the hourly electricity demand of Afyon Kocatepe University	Didn't consider dynamic temporal dependency	ANN
9	Accurate Deep Model for Electricity Consumption Forecasting Using Multi-Channel and Multi-Scale Feature Fusion CNN–LSTM	(Xiaorui Shao , Chang-Soo Kim * and Palash Sontakke, 2020)	To make multi-scale and multi-domain forecast	Does not capture long term dependency	Multi-Channel and Multi-Scale Feature Fusion CNN–LSTM

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. Overview

The objective of this research is to improve the performance of electricity load forecasting at substation level using deep learning model to forecast 15-minute, an hour, and day-ahead electricity load. In this chapter, procedures and tools used to accomplish the research objective are explained. The study area, method of collecting the dataset, preprocessing dataset procedures, model selection, and evaluation are explained. Tool selection is also outlined.

3.2. Dataset Collection Method

The dataset used for this research was collected from the Ethiopian Electric Utility company. It is an electric load measured every 15 minutes over three years at the Gofa substation in Addis Ababa city. The dataset was retrieved from a system that automatically stores electricity loads, from various destinations, sent over a network. However, this dataset contains 31% missing values, where most of them are adjacent to each other. This amount of missing value affects the accuracy of a model to be developed. Hence, the subset of the dataset that has an acceptable amount of missing values but also a larger number of data points is filtered for the purpose of this research. The dataset used for this research starts from 1/1/2020 12:00:00 am to 4/13/2022 12:00:00 pm and contains 79,969 data points.

3.3. Data Processing

One of the main factors for developing an accurate deep learning model is the quality of the dataset. Most of the time, collected datasets do not exist in a format suitable for models. Data preprocessing is the art of transforming a collected dataset into a format and quality a deep learning model can accept and give a better result. This procedure entails selecting appropriate data, filling gaps in the dataset, dividing it into training, validation, and test sets, and converting

a series of data points into a supervised set of input and output values. The final preprocessed data is fed into the models. In this section, each data preprocessing procedure is explained.

3.3.1. Dataset Selection

The original dataset contains a missing value percentage that is more than tolerable. Hence, part of the dataset is selected for better accuracy of the models.

3.3.2. Handling Missing Value

Data often contains missing values. Missing data, also known as missing mechanism, might occur for several reasons. There are three main types of missing data:

- Missing Completely At Random (MCAR): $\mathrm{p}(\mathrm{missing})$ is unrelated to all variables, observed and unobserved

$$p(missing|complete\ data) \neq p(missing)$$

Ex: sensor recording failure

- Missing At Random (MAR): $p(missing)$ is related to observed variables [observed data] only

$$p(complete\ data) = p(missing|observed\ data)$$

Ex: skipped answers in a survey can be predicted

- Missing Not At Random (MNAR): $p(missing)$ is related to the unobserved missing variable or missing data

$$p(missing|complete\ data) \neq p(missing|observed\ data)$$

There are three ways to deal with missing data. These options include replacing them, dropping the missing data points, or leaving them as-is. Each data point in a time series has information on the temporal dependencies between it and other data points; the order of the data points is important. Thus, we are unable to delete the missing data. As a result, before predicting, time series forecasting requires that missing data be substituted with appropriate values. Imputation is the process of substituting missing values for other observable or expected values. Several

approaches have been developed to address missing values in accordance with their missing mechanism. The most common methods are:

Imputation: Using observation from the entire dataset to fill in the value

Interpolation: Using neighboring data points to estimate the value

Deletion: Removing time series period

Not all statistical methods are appropriate for TS data. It is useful to first clarify the type of time series data, whether the time series has no trends and no seasonality, no seasonality but trend, no trend but seasonality, or has both trend and seasonality.

3.3.3. Normalization

Normalization is done to improve the convergence of deep neural networks. Normalization is used to scale each data point in the time series data between 0 and 1. Having a bounded range in the dataset can suppress the effect of outliers. One of the methods to normalize is by using the MinMaxScaler.

MinMaxScaler uses the following equation to convert a given value to scale of 0 to 1.

$$X_n = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

Equation 3.1 Normalization, MinMaxScaler

Where, X_n is normalized value of a variable X

X_i is current value of variable X

X_{min} is minimum value of variable X and,

X_{min} is the maximum value of variable X

3.3.4. Preparing Training Instance

Deep learning models require a training dataset in the form of input-output instances. There are numerous methods that can be used for this. The recursive approach, which makes predictions in one step and uses the outcome as the final input for the following prediction, the direct approach creates a single model for each time step. Employing just one model, the multi-output technique produces the entire forecasting horizon vector. The error in the predictions is not accumulated by many outputs. Since it employs a single model for every prediction, it is also faster than the direct technique in terms of calculation time. The time series are converted into training instances for the models using the moving window technique. A fixed-size window is slid using this procedure, producing an input-output instance at each location. The input to the deep learning models is a fixed-length window (past history), and the output is a dense layer with as many neurons as the predicting horizon specified by the challenge. The past-history parameter that needs to be chosen determines the length of the input (Pedro Lara-Benitez, 2020)

$$s_{input} = [s_1, s_2, \dots, s_{N-L}]^T$$
$$= s_n(t_1)\ s_n(t_2)\ \dots . s_n(t_2)\ s_n(t_3)\ \dots . \dots\ s_n(t_1)\ \dots\ s_n(t_1)\ \dots .\qquad s_n(t_L)\ s_n(t_{L+1})\ \dots\ s_n(t_1)$$

$$s_{output} = [o_1, o_2, \dots, o_{N-L}]^T = s_n(t_{L+1})\ s_n(t_{L+2})\ \dots\ s_n(t_N)$$

3.3.5. Splitting the Dataset

In this step, the instances are divided into three separate parts, including the training, validation, and test sets. The training set is inputted to the deep learning model to obtain a forecasting model. The validation set is also fed into the model to help us optimize the parameters of the model, as it gives insight on how the model is performing on data it has not been exposed to. The test set is used for evaluating the predictive power of the built model in terms of performance metrics.

3.4. Parameter Setting

When the model's high-level parameters are selected and set up optimally, the majority of deep learning models can perform well. Depending on the type of model being used, different configuration parameters must be set. Some deep learning configuration parameters include the

number of hidden layers, the number of units in each hidden layer, the number of epochs, the loss function, the learning rate, the batch size, dropout, the type of activation functions, and Lag.

3.5. Model Training

3.5.1. Artificial Neural Network (ANN)

An artificial neuron network (neural network) is a computational model that mimics the way nerve cells work in the human brain. Artificial neural networks (ANNs) employ learning algorithms that enable them to autonomously adjust—or, in a sense, learn—as they are presented with fresh data. As a result, they are an excellent tool for modeling non-linear statistical data. Deep learning ANNs help the larger field of artificial intelligence (AI) technology and play a significant part in machine learning (ML).ANN (adjusts its computational parameters) as information "flows" through its node layers, based on that input and output (IGI Global, 2022). ANN can be used for temporal data as long as the right sets of input and output are given.

3.5.2. Recurrent Neural Networks (RNN)

RNN, differ from other varieties of artificial neural networks in that they use feedback loops to process a sequence of data that informs the final output, are effective in use cases where context is important for predicting an outcome. As a result of these feedback loops, information can endure. This effect is frequently referred to as memory.

The majority of RNN use cases are associated with language models, where predicting the next letter in a word or the next word in a phrase depends on the information that comes before it. It is also useful in time series forecasting, where there is temporal dependency between data points.

3.5.3. Long Short-Term Memory (LSTM)

LSTM is a variation of recurrent neural network. LSTM was created to address the vanishing gradient problem and identify long-term dependencies in a sequence without overlooking the short-term trends; LSTMs may model temporal dependencies over a wider range of time periods. A hidden layer in LSTM networks is referred to as an LSTM memory cell. To avoid irrelevant perturbations from changing the memory units, LSTM cells use a multiplicative input gate to regulate the memory units. Similar to how a multiplicative output shields adjacent cells from present memory disturbances later, the LSTM memory cell receives a forget gate. With the use

of this gate, LSTM may learn to erase outdated data from memory (Pedro Lara-Benitez, 2020). As more parameters may be learned due to the LSTM cell, long-term memory is added in a way that is even more efficient. This makes it the most effective Recurrent Neural Network for predicting, particularly when your data show a longer-term trend.

3.6. Evaluation

Time series forecasting performance measurements provide a summary of the forecast model's skill and capability in making the forecasts. There are a variety of metrics available to measure time series forecasting capability.

RMSE

Root mean squared error (RMSE) is an absolute error measure that squares the deviations to keep the positive and negative deviations from canceling one another out. When comparing models, this measure makes significant errors appear more prominent. To choose the optimal model, RMSE is computed using denormalized input and forecasted output data. The equation for calculating RMSE:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(Y_t - \widehat{Y_t})^2}{n}}$$

Equation 3.2 Root Mean Squared Error

Where Y_t is the actual value of a data point for a given time period t

$\widehat{Y_t}$ is the predicted value of a point for a given time period t

n is the total number of data points

MAPE

Mean absolute percentage error (MAPE) measures the accuracy of the forecasting method. It indicates, on average, how accurate the anticipated quantities were in relation to the actual

amounts by averaging the absolute percentage errors of each entry in a dataset(Indeed Editorial Team, 2021).

The equation for calculating MAPE:

$$MAPE = \frac{1}{n} x \sum_{t=1}^{n} \frac{|Y_t - \widehat{Y_t}|}{Y_t} x100$$

Equation 3.3 Mean Absolute Percentage Error

Where Y_t is the actual value of a data point for a given time period *t*

$\widehat{Y_t}$ is the predicted value of a point for a given time period *t*

n is the total number of data points.

MAE

The Mean Absolute Error (MAE) measures the average differences between predicted values and actual values. The formula for the mean absolute error is:

$$MAE = \frac{1}{n} x \sum_{t=1}^{n} |Y_t - \widehat{Y_t}|$$

Equation 3.4 Mean Absolute Error

Where Y_t is the actual value of a data point for a given time period *t*

$\widehat{Y_t}$ is the predicted value of a point for a given time period *t*

n is the total number of data points.

3.7. Development Tools

- **TensorFlow:** TensorFlow is an end-to-end open source platform for deep learning applications. It has a wide range of adaptable tools and libraries(tensorflow, 2022). Due to its spontaneous high-level APIs like Keras, TensorFlow can develop neural networks faster and

easier. Models can be easily trained and deployed in the browser, the cloud, or on-device. It is an easy and flexible architecture to take new ideas from concept to code or to state-of-the-art models.

- **Keras:** Keras is a high-level deep learning API developed by Francois Chollet from Google for implementing neural networks. It is used to make the implementation of neural networks simple and was developed in Python. Additionally, It supports various backend neural network computation(simplilearn, 2021). Its minimalistic and extensible features make Keras a preferable choice for deep learning implementation.
- **Anaconda**: Anaconda is an open-source Python distribution for scientific computing that simplifies package management and deployment. The distribution comes with packages for data science. More than 1500 packages, a virtual environment manager, and conda packages are included in Anaconda. It seeks to make deployment and package management simpler. Anaconda offers the tools required to quickly collect data (from files, databases), manage environments, deploy projects with a single click, share them, collaborate on them, and recreate them. Users may run programs, manage conda packages, and more using Anaconda's desktop graphical user interface (GUI), Anaconda Navigator. Jupiter Notebook, Spyder, and Visual Studio Code are among the programs that are accessible by default in the navigator.
- **Jupyter notebook**: Jupyter Notebook is an open source web application for creating and sharing computational documents. It provides an efficient, document-focused interface. It includes text, equations, graphics, and live code (docker hub, 2022). Its interactive behavior makes it a suitable tool for the application of deep learning models.
- **Google colab**: Google Colab is a Jupyter notebook on the cloud with the main machine learning and deep learning packages preinstalled. It is an easy-to-configure interactive environment that comes with access to GPUs(Chng, 2022).
- **Python**: Python is an open-source, interactive, interpreted, and object-oriented programming language. Classes, dynamic typing, very high level dynamic data types, exceptions, modules, and exception handling are all included. Python is the most widely used programming language for data science and machine learning(docker hub, 2022).
- **Pandas**: Pandas is an open source data analysis and manipulation tool, built on top of the python programming language(pandas, 2022). Pandas is a fast and well-organized dataframe object for data management with unified reading and writing of data in different formats

(CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format). It offers flexible data reshaping and turning, huge dataset indexing and sub setting, data combining or conversion, and split-apply-combine operations on datasets. When manipulating time series data, its date range generation, frequency conversion, date shifting, and lagging functionality are helpful.

- **Plotly**: Plotly is an Open Source Graphing Library for Python. The interactive, graph-publishing-ready graphs are created by Plotly's Python graphing package. It can be applied to interactive dataset visualization.

- **Microsoft Visio:** Microsoft Visio is professional diagramming software used to draw flowcharts diagrams and schematics. It is an efficient tool to draw methodology procedures and model architectures.

CHAPTER FOUR

4. PROPOSED LONG SHORT-TERM MEMORY MODEL

4.1. Overview

In this research we used electricity power load historical data of Gofa substation, which is found in Addis Ababa, Ethiopia measured for over 2 years. In this chapter, the dataset used and how the proposed Long-Short Term Memory Model is developed is explained. Various deep learning models are tested in order to identify which model works best with time series data, in particular on our dataset. Since LSTM has a capability of learning the relationship between neighboring data points as well as dependability between data points that are far away, it is chosen to be used in this thesis.

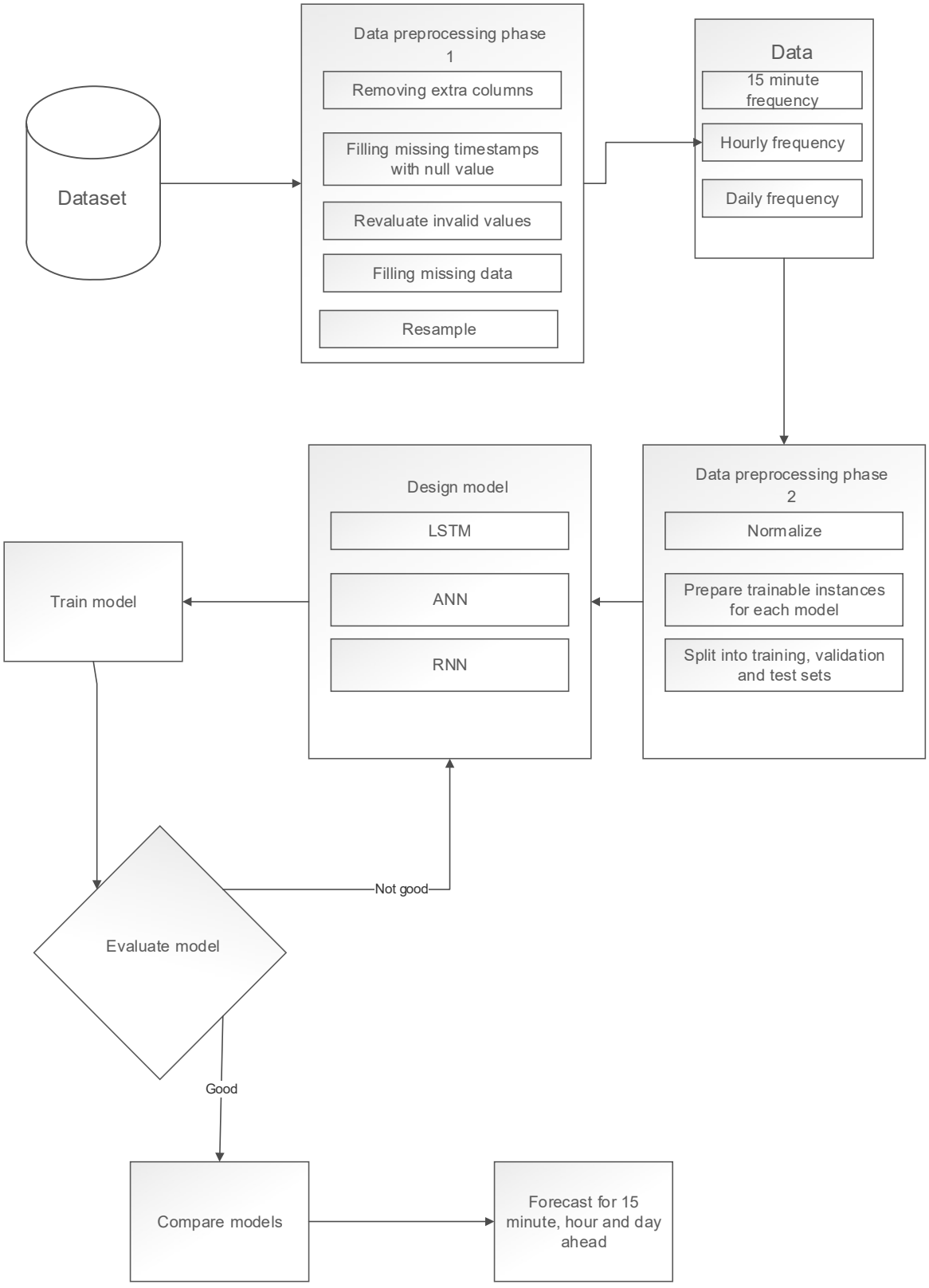

Figure 4.1 Prediction Model architecture

4.2. LSTM Architecture

LSTM is a variation of recurrent neural network which have the ability to remember past data in its memory. This network does processing, classification, and time series-based forecasting very efficiently. For resolving the long-term reliance issue, Long Short-Term Memory is developed as an upgrading version of recurrent neural network (RNN). The LSTM structure is arranged in a chain, just as the RNN. However, LSTM uses four interacting layers as the repeating module as opposed to a single tanh layer(Hasan-Al-Shaikh et al., 2019). The structure of the LSTM's hidden layers is more intricate. Each hidden layer of the LSTM contains gate and a memory cell. Input gate I forget gate f, output gate o, and self-connected memory cells c make up the majority of a memory block. The activations' entry into the memory cell is managed by the input gate. The output gate gets information of what cell activations to output to the succeeding network. The forget gate assists in clearing the network's memory cells and erasing previous input data. To further enable long-term access and storage of the information by the memory cells, multiplicative gates are applied with caution. The vanishing gradient issue can be successfully mitigated by such a structure(Jian Zheng et al., 2017). Because of this, LSTM is an architecture that works well for issues involving long-term dependencies.

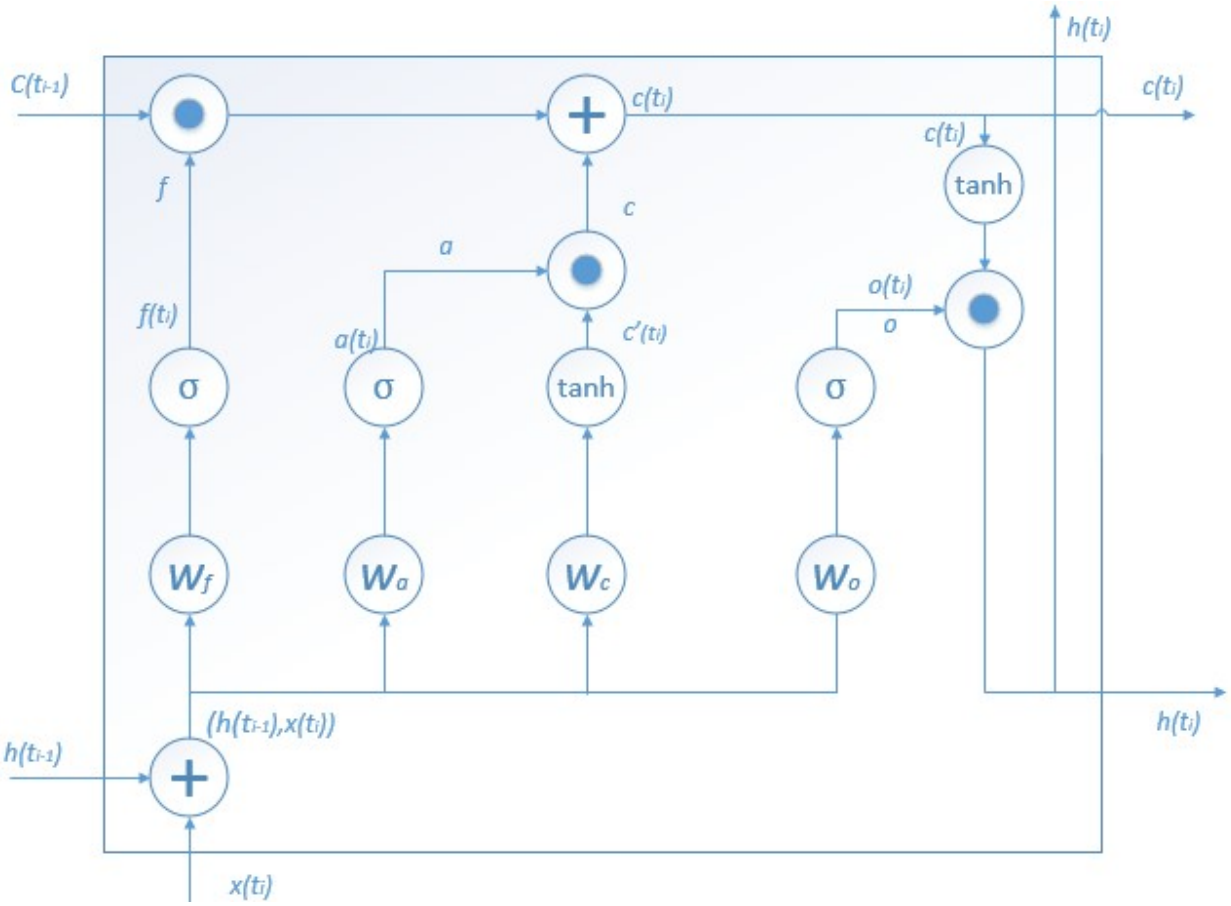

Figure 4.2 LSTM architecture (Pedro Lara-Benitez, 2020)

The operation of LSTM is as follows:

The forget gate $f(t_i)$ uses $x(t_i)$ and $h(t_{i-1})$ as input to compute the information to be preserved in $c(t_{i-1})$ using a sigmoid activation.

The input gate $a(t_i)$ takes $x(t_i)$ and $h(t_{i-1})$ to compute the value of $c(t_i)$.

The output gate $o(t_i)$ performs regulation on the output of LSTM cell by considering $c(t_i)$ and applying both sigmoid and tanh layers. Mathematically, the forward learning of the LSTM is as follows:

$$a(t_i) = \sigma(w_a x(t_i) + w_{ha} h(t_{i-1}) + b_a)$$

Equation 4.1 LSTM eq1

$$f(t_i) = \sigma(w_f x(t_i) + w_{hf} h(t_{i-1}) + b_f)$$

Equation 4.2 LSTM eq2

$$c(t_i) = f_t \, X \, c(t_{i-1}) + a_t \, X \, tanh \, (w_c x(t_i) + w_{hc}(h(t_{i-1}) + b_c))$$

Equation 4.3 LSTM eq3

$$o(t_i) = \sigma(w_o x(t_i) + w_{ho} h(t_{i-1}) + b_o)$$

Equation 4.4 LSTM eq4

$$h(t_i) = o(t_i) \, X \, tanh(c(t_i))$$

Equation 4.5 LSTM eq5

Where:

$x(t_i)$: The input value

$h(t_i)$ and $h(t_{i-1})$: The output value at time points t_i and t_{i-1}

$c(t_i)$ and $c(t_{i-1})$: Cell state at time points t_i and t_{i-1}

$\vec{W} = \{w_a, w_f, w_c, w_o\}$: Weight matrices of input gate, forgate gate, internal state and output gate

$\vec{a} = \{a(t_i), f(t_i), c(t_i), o(t_i)\}$: Output results for input gate, forgate gate, internal state and output gate

σ and tanh are activation functions, and X indicates point-wise multiplication. Overall, the LSTM learns using the following steps: Compute the LSTM output using equations 4.1 to 4.5 (forward learning). Then, Compute the error between the resulting data and input data of each layer. The error is reversely propagated to input gate, cell, and forget gate. Based on the error term, the weight of each gate is updated using an optimization algorithm. The above four-step processes are repeated for a given number of iterations and the optimal values of weights and biases are obtained.

LSTM forward learning is the process of computing the LSTM output using the aforementioned equations. Then calculate the error between each layer's output data and input data. The input gate, cell, and forget gate all experience reverse propagation of the error. An optimization technique is used to change each gate's weight based on the error term. The aforementioned procedures are repeated for a certain number of iterations in order to acquire the ideal weights and biases(Greff et al., 2017).

4.3. Implemented LSTM Model

4.3.1. Input Data

LSTM learns from a set of supervised instances given to it as an input. A single input of the implemented LSTM contains one electricity load value from one lag and one current load value as an output. These instances are then transformed into a Numpy array with three shapes, which are number of instances, number of time steps and number of features. In our case the input has one feature that is electricity load; one time step and the no of instances vary depending on the forecasting horizon we experimented with.

4.3.2. Hidden Layers

In developing an LSTM model, one of the key decisions to make is choosing the number of hidden layers and number of units in each layer. These values vary depending on the problem were working on. The neurons at each layer perform an activation function, add the bias, and compute the weighted sum of the inputs and the weights. In this study, we used two LSTM layers as hidden layers. Each of the LSTM layers contains 32 neurons.

4.3.3. Dense Layer

Dense layer sometimes referred to as fully connected layer is a deeply connected layer. The LSTM's network's latter stages involve the usage of a dense layer. This layer modifies the dimensionality of the output from the earlier LSTM layers by conducting matrix vector multiplication. The LSTM model can more clearly define the relationship between the values it is working on by utilizing the dense layer. In the implementation, the dense layer is connected to the 2 hidden LSTM layers. It functions as the output layer, and it contains one unit.

4.4. Dataset Collection

The dataset used for this research is collected from Ethiopian electric utility. The dataset consists of historical electricity loads recorded with a meter every 15 minutes of Gofa, Addis Ababa substation and sent to Ethiopian Electric Utility management system over network. The sample dataset collected from Ethiopian Electric Utility for Gofa substation is displayed in Appendix A.

4.5. Preprocessing

The dataset collected is checked for invalid values and missing values. The original collected dataset spans from 3/21/2019 to 4/21/2022. It contains 31% missing data.

4.5.1. Selection

First, the section of the dataset with longest time span, but smaller missing values is chosen. The chosen subsection spanned from 1/1/2020 12:00:00 am to 4/13/2022 12:00:00 pm and contains 79,969 data points. Since demand value must not exceed the maximum transformer capacity, all values larger than 30000 are treated. Also, any value less than zero are treated as zero.

4.5.2. Filling Missing Value

Autocorrelation and lag-plot analysis are made to identify how to replace the missing values. Using forward or backward fill was not effective; hence most of the values are consecutive. Rather, since the dataset exhibited a strong daily seasonality, each missing value is replaced with the 96^{th} back data which is a load value at the same time of the day before. Sample dataset after preprocessing is displayed in Table 4.1.The proposed model forecasts with three time horizons: 15 min, hour and day ahead. Hence, the data is resampled hourly and daily. The plot of the dataset with 15 minute and daily frequency is displayed in Figure 4.3 and Figure 4.4 respectively.

Table 4.1 Sample preprocessed dataset

Datetime	D_KW
1/1/2020 0:00	3120
1/1/2020 0:15	3228
1/1/2020 0:30	3104
1/1/2020 0:45	2708
1/1/2020 1:00	2664
1/1/2020 1:15	2588
1/1/2020 1:30	2604
1/1/2020 1:45	2540
1/1/2020 2:00	2608
1/1/2020 2:15	2596
1/1/2020 2:30	2568
1/1/2020 2:45	2508

Datetime	D_KW
1/1/2020 3:00	2608
1/1/2020 3:15	3120
……	……..
……	……..
4/12/202222:15	10280
4/12/2022 22:30	9240
4/12/2022 22:45	8240
4/12/2022 23:00	8000
4/12/2022 23:15	7280
4/12/2022 23:30	6760
4/12/2022 23:45	6200
4/13/2022 0:00	5400

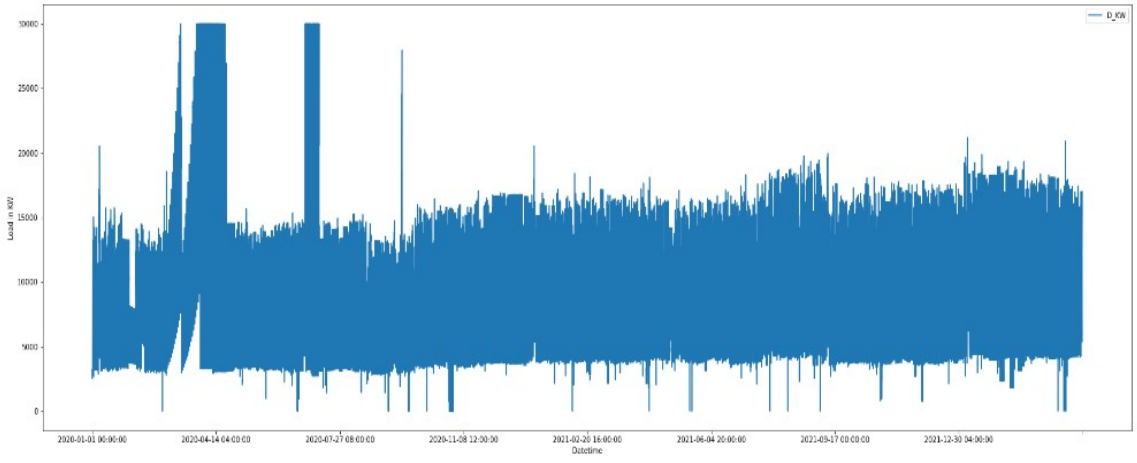

Figure 4.3 Preprocessed 15 min Load dataset plot

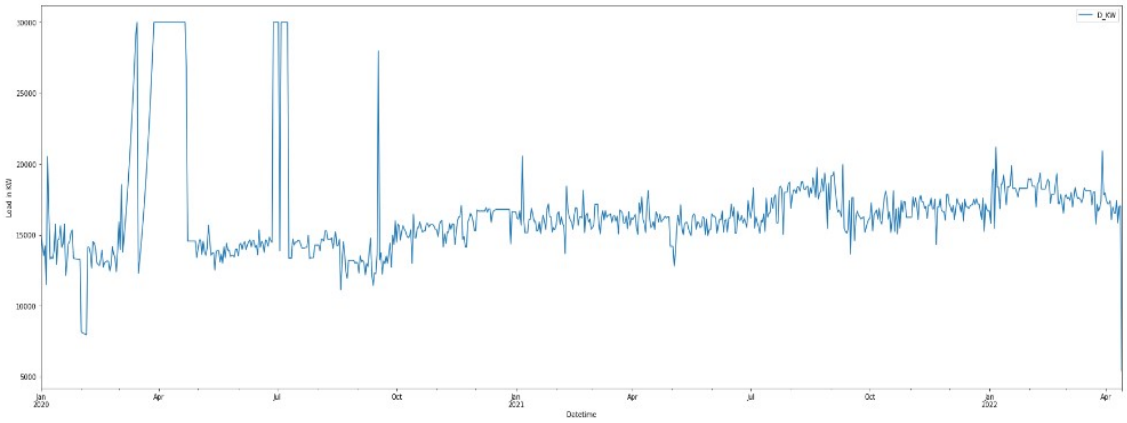

Figure 4.4 Preprocessed daily Load dataset plot

4.6. Parameter Setting

Most of the deep learning models can perform well given that the hyperparameters of the model is chosen and configured in the best possible way. The high parameters to be configured are different depending on what kind of model were using. Some of the parameters of deep learning include:

4.6.1. Number of Hidden Layers

Neural networks are built from at least three layers, input layer, hidden layer and output layer. The number of hidden layers can be one or more. Deciding how many hidden layers to use and how many units each layer should contain is an important factor when it comes to the accuracy of a given model. If we increase the number of hidden layers then the neural network complexity increases. Hence, for this study, the LSTM model is set to have 2 hidden layers.

4.6.2. Number of Units

Selecting the optimal number of neurons for each layer in the LSTM network is not a straightforward task. If the number of neurons is very small, the LSTM will not be able to memorize all necessary information to perform prediction optimally. Also, if the number of neurons is very high, the LSTM will overfit on the training instances and will not demonstrate suitable generalization to accurately forecast the test set. In the LSTM model, setting the number of units to both of the layers to 32 worked the best.

4.6.3. Number of Epochs

A single iteration over all training instances is referred to as one training epoch. Using too few numbers of epochs can cause a model not to capture enough information from the training dataset hence leading to underfitting and using too many number of epochs can cause the model to over learn from the training dataset hence couldn't generalize to a new dataset. Therefore, finding a suitable epoch number is vital in achieving a model with high performance. The proposed model performed its best after running 20 rounds; hence the selected epoch is 20.

4.6.4. Loss Function

Selecting the right loss function for a deep learning problem is crucial. Loss function is an object that measures how often a model makes an incorrect prediction. Loss is the prediction error of Neural Net. And the method to calculate the loss is called Loss Function. The Loss is used to calculate the gradients. And gradients are used to update the weights of the Neural Net. In this study, MSE (Mean Squared Error) is used to measure loss during model training. MSE is used for regression tasks. It is calculated by taking the mean of squared differences between actual and predicted values.

$$MSE = \frac{\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2}{n}$$

Equation 4.6 Mean Squared Error

4.6.5. Learning Rate

Learning rate is an important hyperparameter which adjusts the extent of change to the model weights. Choosing the best learning rate is essential in obtaining a model with high performance. After experimenting with various learning rates, 0.1 is found to fit the dataset well.

4.6.6. Batch Size

Batch size is the number of trainable instances used to train a model before updating its trainable model variables, the weights and biases. In every single training step, a batch of samples is propagated through the model and then backward propagated to calculate gradients for every sample. Batch size has a critical impact on the convergence of the training process as well as on the resulting accuracy of the trained model. Typically, there is an optimal value or range of values for batch size for every neural network and dataset. Different neural networks and

different datasets may have different optimal batch sizes. Two of the main potential consequences of using small or large batch sizes are generalization and convergence speed.

- Generalization: Generalization is a model's ability to perform well on samples outside the training set. Large batch sizes may cause bad generalization.

- Convergence speed: Small batch sizes may lead to slow convergence of the learning algorithm. The variable updates applied in every step that was calculated using a batch of samples will determine the starting point for the next batch of samples. Training samples are randomly drawn from the training set every step, and therefore the resulting gradients are noisy estimates based on partial data. The fewer samples we use in a single batch, the noisier and less accurate the gradient estimates will be. That is, the smaller the batch, the bigger impact a single sample has on the applied variable updates. In other words, smaller batch sizes may make the learning process noisier and fluctuating, essentially extending the time it takes the algorithm to converge.

Hence, we have to choose a batch size that will be neither too small nor too large. We need to find one that would be optimal for the specific neural network and dataset we are using. The LSTM model worked best with batch number = 20.

4.6.7. Lag

Lag is how far back a model has to look in the past to predict the present or the present and future values. Exploring the nature of the dataset to identify in which lag a strong dependency exists between the past and future data points is important for accuracy of a model to be trained. In figure 4.5, it can be seen adjacent values in our dataset have strong correlation.

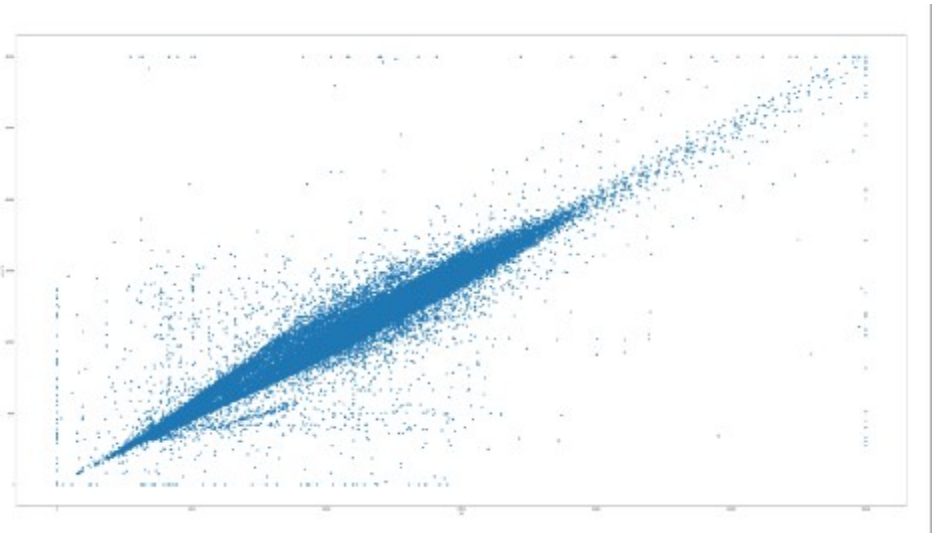

Figure 4.5 Lag plot, lag=1

4.6.8. Optimizer

Optimizers are algorithms or methods used to minimize an error function (loss function) or to maximize the efficiency of production. Optimizers are mathematical functions which are dependent on the model's learnable parameters, i.e. Weights & Biases. It helps to know how to change weights and learning rate of neural networks to reduce the losses. Optimizers are very crucial to increase the accuracy of a model.

Adam: It is an optimizer that calculates learning rate for each parameter that is shown by its developers to work well in practice and to compute favorably against other adaptive learning algorithms(Bikal Basnet, 2016). Adam optimization algorithm is an extension of stochastic gradient descent (optimization algorithm) to update network weights during training. Adam optimizer updates the learning rate for each network weight individually. The Adam optimizers inherit the features of both Adagrad and RMS prop algorithms. In Adam, instead of adapting learning rates based upon the first moment (mean) as in RMS Prop, it also uses the second moment of the gradients. It means, the uncentered variance by the second moment of the gradients(we don't subtract the mean). The adam optimizer has several benefits, due to which it is used widely. The algorithm is straightforward to implement, has faster running time, low memory requirements, and requires less tuning than any other optimization algorithm.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left[\frac{\delta L}{\delta \omega_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2)\left[\frac{\delta L}{\delta \omega_t}\right]^2$$

Equation 4.7 Adam optimizer

Equation 4. 1 represents the working of Adam optimizer. Here, B1 and B2 represent the decay rate of the average of the gradients (Ayush Gupta, 2022). The fully connected long short-term memory model proposed is displayed in Figure 4.6.

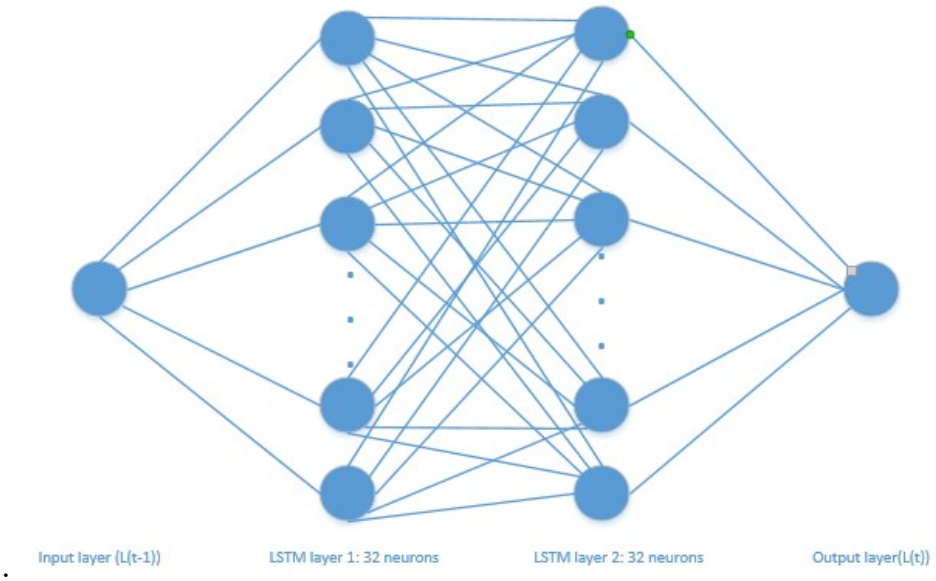

Figure 4.6 Proposed LSTM model

CHAPTER FIVE

5. IMPLEMENTATION

5.1. Overview

In this chapter, data preprocessing steps, implementation details of the proposed model, including hyperparameters setting, prediction and evaluation, are provided. While the appendix contained illustrations of detailed codes, significant code segments are presented in this chapter.

5.2. Working Environment

To implement the model, we used both local computer and Google Colab. The hardware and software specifications of the local computer used are:

- Hard Disk: One Terabyte byte of storage
- Processor: Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz, 2301 Mhz, 2 Core(s), 4 Logical Processor(s)
- Installed Memory: Physical Memory (RAM) 12.00 GB
- OS: Microsoft Windows 10 Pro, x64-based PC, Version 10.0.19044 Build 19044
- System Model: - Vostro 15-3568

5.3. Setup Environment

The model is implemented using python programming language. The following useful Libraries and packages are employed.

- Python 3.10.0: The latest version of python programming language at the time of experimenting is used to implement the DL models.
- TensorFlow 2.9.1: Used as a backend for keras
- Keras 2.9.0: Used as an API to develop and fit the deep learning models
- Matplotlib 3.5.0: To visualize the electricity load dataset and the results of the models
- NumPy 1.21.4: Used to manipulate the dataset
- Pandas 1.3.4: For data manipulation, analysis and visualization
- Scikit-learn 1.1.1: To evaluate the amount of error in deep learning models

5.4. Data Preprocessing Implementation

The collected Gofa substation electricity load dataset is a CSV file containing timestamps of when load data was recorded, the amount of load data in KW, and other columns that are not needed in this implementation. After these unnecessary columns are removed, the values are sorted, and the missing timestamps are added and filled with null values, then the dataset is loaded into the pandas dataFrame.

```
dataframe=read_csv("E:/Research/Data/New demand load/load_no_missing_value_feeder1_gofa_15min _jan1_2020_2022.csv",index_col=0)
```

Code-Fragment 5. 1 Loading substation electricity load dataset

The dataset is then resampled hourly and daily, resulting in three different datasets. The dataset resampled hourly contains 19,993 records and its implementation is displayed in code-fragment 5.2. The daily dataset, on the other hand, contains 906 data points.

```
dataframe.index=pd.to_datetime(dataframe.index)
dataframe=dataframe.resample('h').max()
dataframe.plot(figsize=(30,10),xlabel="Datetime", ylabel="Load in KW")
```

Code-Fragment 5. 2 Resample 15 minute load data to hourly load data

Each of the resampled datasets has gone through normalization. MinMaxScaler, which is from the sklearn. A preprocessing module is used to convert the given values in the dataframe to a scale of 0 to 1. The minimum value of the dataset is set to 0 and the maximum value of the dataset is set to 1, and all values in between to a number between 0 and 1. The sample code for the normalization processes is outlined in code-fragment 5.3.

```
scaler=MinMaxScaler(feature_range=(0,1))
dataset=scaler.fit_transform(dataset)
```

Code-Fragment 5. 3 Normalize with MinMaxScaler

The dataset is split into training, validation, and test sets. Train sets are used to fit the model, so the model learns the dataset's pattern from it. The validation sets are used to help the model reset its parameters by giving the model insight into how it performs on data it has not seen before.

The validation set is utilized as the model is being trained. Finally, the test set is used to evaluate the model's performance after it finishes training and is used to compare the model's performance to other models. This thesis used 80% of the dataset to train the model, 10% of the dataset as a validation set, and the remaining 10% of the dataset to test the model's accuracy. The sample code of splitting the dataset is presented in code-fragment 5.4.

```
train_size=int(len(dataset) * 0.8)
validation_size=int(len(dataset)*0.9)
train,validation,test=dataset[0:train_size,:],dataset[train_size:validation_size,:],dataset[validation_size:len(dataset),:]
```

Code-Fragment 5. 4 Split dataset

5.5. Creating and Compiling the Model Implementation

To create a sequential model, a set of hyperparameters, such as the number of layers and the number of neurons in each layer, needs to be decided. The Keras Sequential API is used to stack the layers. In the proposed model, we built a model with two LSTM layers stacked sequentially after the input layer that takes one step early data point and uses a dense layer containing one neuron as an output layer. The code snapshot to create the LSTM model is displayed in code-fragment 5.5.

```
model=Sequential()
model.add(LSTM(32,return_sequences=True,input_shape=(None,seq_size)))
model.add(LSTM(32,input_shape=(None,seq_size)))
model.add(Dense(1))
adam=optimizers.Adam(learning_rate=0.1,beta_1=0.9,beta_2=0.999,epsilon=None, decay=0.0,amsgrad=False)
```

Code-Fragment 5. 5 Model creation

The model uses the Adam optimization algorithm (which is an extension of stochastic gradient descent) which is known to be computationally efficient, requires little memory, and typically requires little tuning. It is used to change the attributes of a neural network, such as weights and learning rate, in order to reduce the losses. The model uses mean-squared error to evaluate error and punish the model while training.

```
model.compile(loss='mean_squared_error',optimizer='adam')
```

Code-Fragment 5. 6 Compiling LSTM model

5.6. Model Training Implementation

After a model is developed with chosen hyperparameters and compiled with chosen parameters, it is trained with a batch number of 20 and runs for 20 rounds.

```
model.fit(trainX,trainY, validation_data=(validX,validY),epochs=20,verbose=2,batch_size=20 )
```

Code-Fragment 5. 7 Training LSTM model

Epoch vs. Loss graph is one way of understanding how well the model is performing on training and validation sets.

```
training_loss = model.history.history['loss']
validation_loss = model.history.history['val_loss']
epoch_count = range(len(training_loss))
plt.plot(epoch_count, training_loss)
plt.plot(epoch_count, validation_loss)
plt.legend(['Training Loss', 'Val Loss'])
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()
```

Code-Fragment 5. 8 Plotting training, validation loss vs. Epoch

In order to evaluate the model's accuracy, the final version of the model must be used to forecast given a set of input set. In this case, we used the developed LSTM model to predict on all three sets: training, validation, and test.

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

Code-Fragment 5. 9 Predict using developed LSTM model

The developed model makes the prediction on a normalized set of values. Thus, its output is also in a normalized form. To visualize the results of the model and evaluate them (compare with actual values), we need to denormalize the outputs back to the original data ranges. To do that,

we call the inverse transform method on the scaler object created earlier with the set of the output passed as a parameter.

```
#Denormalize
trainPredict=scaler.inverse_transform(trainPredict)
trainY=scaler.inverse_transform([trainY])
validPredict=scaler.inverse_transform(validPredict)
validY=scaler.inverse_transform([validY])
testPredict=scaler.inverse_transform(testPredict)
testY=scaler.inverse_transform([testY])
```

Code-Fragment 5. 10 Denormalize electricity load data

A great way to visualize how well a model has performed is to plot the predicted values against the actual values graph. We used the matplotlib. Pyplot API to plot the actual vs. predicted electricity load graph.

```
plt.plot(testY[0])
plt.plot(testPredict)
plt.legend(["Actual Load", "Predicted Load"])
plt.ylabel("Load in KW")
plt.rcParams["figure.figsize"] = (20,6)
plt.show()
```

Code-Fragment 5. 11 Plotting actual load vs. predicted load

5.7. Evaluating the Model Implementation

In this thesis, three evaluating metrics are implemented: RMSE, MAE, and MAPE. Because, the test set of 15 minute and daily frequency contains zero values, MAPE could not be applied to all of the horizons. Since the paper we compared to, (Venkataramana Veeramsetty, 2020), used MAPE to demonstrate their accuracy and their forecasting horizon was an hour, MAPE was implemented only on hourly forecast.

```
def RMSE(Y_actual,Y_Predicted):
    rmse = math.sqrt(mean_squared_error(Y_actual,Y_Predicted))
    return rmse

def MAE(Y_actual,Y_Predicted):
    mae = np.mean(np.abs(Y_actual - Y_Predicted))
    return mae
def MAPE(Y_actual,Y_Predicted):
    mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return mape
```

Code-Fragment 5. 12 Evaluation metrics

CHAPTER SIX

6. RESULT AND DISCUSSION

6.1. Overview

In this chapter, the results of the experiments from implementing three different deep learning models, which are ANN, RNN, and LSTM, are outlined. The electricity power load history dataset of Gofa substation is fitted into these models. After the necessary preprocessing steps are carried out, three different forms of the dataset are fitted into the models. These are the 15-minute, hourly, and daily electricity load datasets. Evaluation metrics like MAE, RMSE, and MAPE are used to evaluate the accuracy of each of the models on each of the datasets mentioned above. ANN model and preprocessing steps from (Venkataramana Veeramsetty, 2020) are used to train the datasets for the ANN model. The comparison between the results of the proposed model and the other methods is also outlined.

6.2. Dataset Analysis

The dataset does not have a trend but exhibits, mostly, daily seasonality. Electricity load is the lowest between 12 am to 5 am, rises from 5 am to 7 am, stays high but fluctuates between 7 am to 9 pm and declines from 9 pm to 12 am. Figure 6.1 and 6.2 demonstrate the daily seasonality of the dataset.

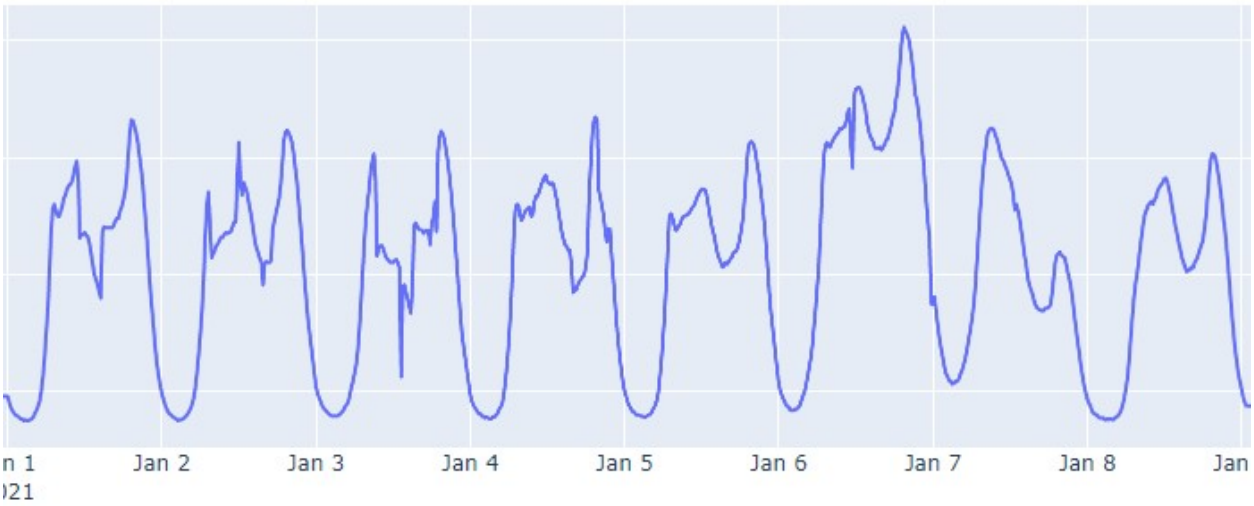

Figure 6.1 Sample dataset plot displaying daily seasonality

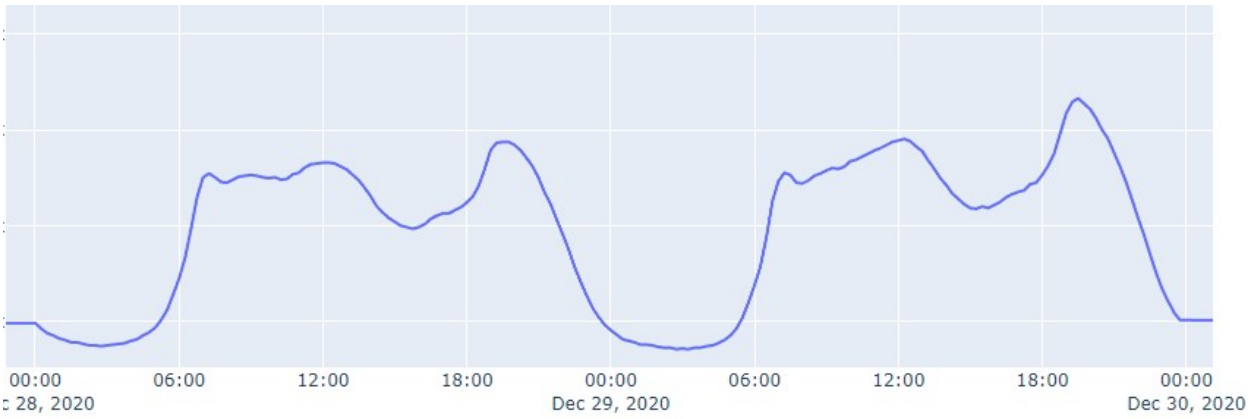

Figure 6.2 Sample dataset plot displaying daily seasonality, closer look

6.3. Deep Learning Models Selection

From literature review, we identified three candidate deep learning models suitable for electricity load forecasting. (Venkataramana Veeramsetty, 2020) is the one paper we found that used a substation dataset. Hence, the ANN model from the paper is used as one of a benchmark models. The paper also used temperature wind-index as external factor. RNN is another implemented deep learning model, as it is a good deep learning model for time series forecasting problems. The evaluation result of the proposed LSTM model and the comparison between these models is outlined in the following section.

6.4. LSTM Model's Results

When training a model, one way of finding out if a model overfits or underfits is by plotting the training loss vs. validation loss graph. Sample train vs. validation loss of a 15 minute forecasting with LSTM model is displayed in Figure 6.3.

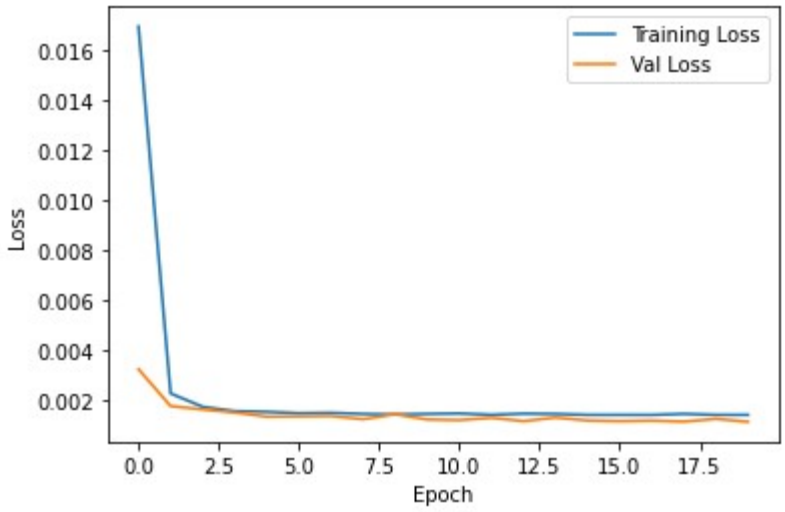

Figure 6.3 Loss vs. Epoch plot for 15-min forecast

In this particular case, as displayed in Figure 6.3, the training loss significantly decreased on the second round of training the model and since then has steadily decreased until round 20. Figure 6.4, 6.5, and 6.6 exhibit the actual vs. predicted electricity load data for 15 minute, hourly, and daily datasets respectively. For the 15-minute data, the proposed Long Short-Term Model performed very well, the actual and predicted values pretty much overlap. In the hourly forecasting, the model did well too. For the daily forecast, the proposed model performed the best. The gap between the actual and predicted values actually seem higher than the previous two because, the number of data points in the daily test set is smaller, hence the plot is zoomed in compared to the earlier two plots

.

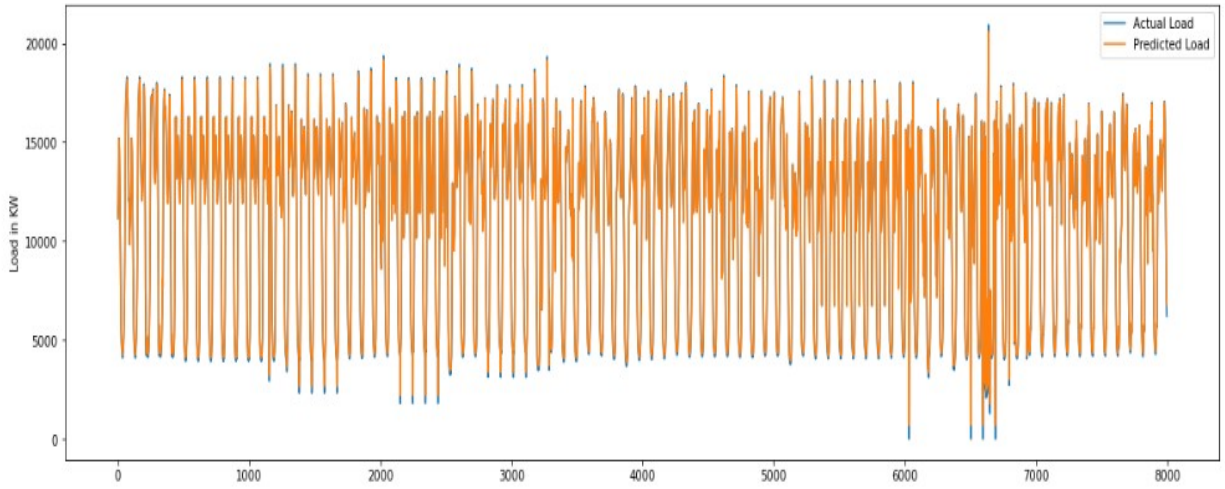

Figure 6.4 LSTM model 15 min forecasted load vs. actual load

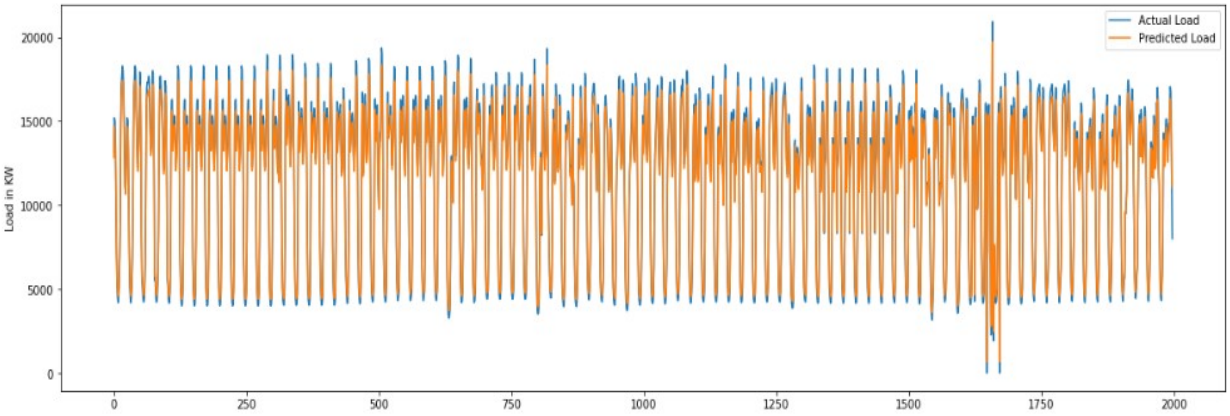

Figure 6.5 LSTM model hourly forecasted load vs. actual Load

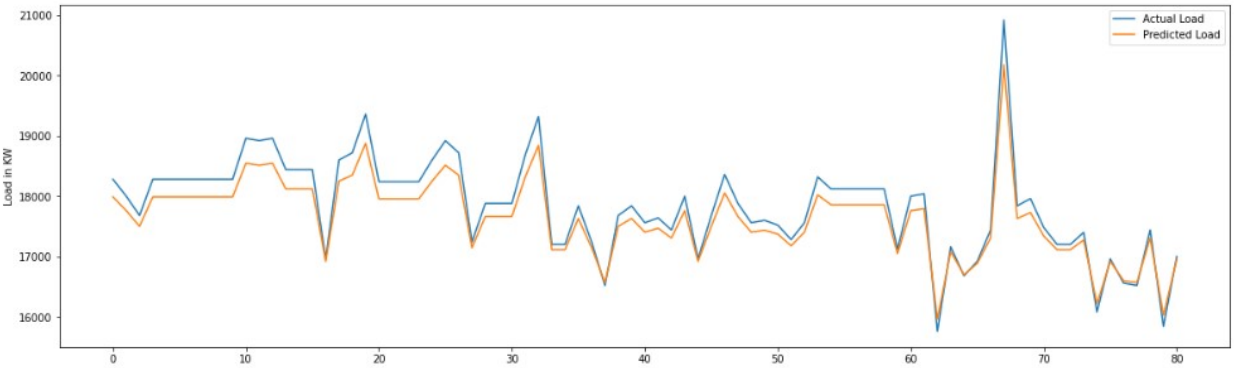

Figure 6.6 LSTM model daily forecasted load vs. actual Load

6.5. Evaluation of Results

To compare the results of electricity load forecasting models implemented in these theses, RMSE, MSE, and MAPE metrics are used. The comparison between ANN, RNN, and the LSTM model in terms of RMSE and MAE is listed in Table 6.1 and Table 6.2, respectively.

Table 6.1 RMSE evaluation result of LSTM model vs. RNN and ANN

	15 min	Hourly	Daily
Proposed LSTM model	86.57	424.45	256.03
RNN	415.75	718.42	267.26
ANN	524.45	799.3	462.23

Table 6.2 MAE evaluation result of LSTM model vs. RNN and ANN

	15 min	Hourly	Daily
Proposed LSTM model	56.51	381.36	222.64
RNN	334.73	805.86	286.39
ANN	424.88	1178.21	823.87

Tables 6.1 and 6.2 show that the LSTM model performs better in terms of RMSE and MAE than the ANN base line model and the RNN model.

In table 6.3 the proposed model's training and validation MSE and MAPE of the testing set are compared with the corresponding results of the ANN based paper. It demonstrates that the LSTM model has far smaller error values.

Table 6.3 Result comparison between ANN and hour ahead LSTM model

	Train(MSE)	Validation(MSE)	Test(MAPE)
Proposed model	0.0067	0.0057	4.64
ANN	0.2	0.26	9.1

6.6. Discussion

In this study, we collected electricity power historical load data of Gofa substation. Then the dataset is analyzed to get more understanding of the data. From the analysis, we understood that the dataset has missing values and extreme values. We also noticed there is weakly seasonality, and adjacent electricity load values have the strongest autocorrelation. These phenomenons are presented in Figure 4.5, Figure 6.1 and Figure 6.2 respectively. The extreme values are treated according to the law of maximum substation load. The missing values are imputed with the day before values, and the autocorrelation result helped in deciding the window size of the LSTM model. We have resampled the dataset in three forms, thus giving us with three different cases. We have also performed our experiment on three different deep learning models namely ANN, RNN and LSTM, which are chosen after careful literature review on electricity load forecasting and forecasting in general. From the results outlined in this chapter, it can be seen that there is a significant difference in the error values of RMSE and MAE between the proposed LSTM model and the other two models. From the actual electricity load vs. predicted electricity load plots and comparisons between the LSTM model and models like RNN and ANN, it can be seen that the proposed model outperforms these models.

Generally, we can conclude that using LSTM on electricity load data at substation level can increase forecasting performance to a significant level. Electricity load forecasting at substation level using only historical load dataset was also, able to give good performance.

CHAPTER SEVEN

7. CONCLUSION AND RECOMMENDATION

7.1. Conclusion

In this study, it is justified that LSTMs are a good candidate for time series forecasting. It was able to capture not only the long-term relationship that spanned weeks to years, but also the relationship between the adjacent or neighboring data points and the weekly relationship. When developing the models, we discovered that as the forecast horizon increased, the complexity of the models also increased slightly. The proposed LSTM model outperformed both the Recurrent Neural Network and Artificial Neural Networks model. It has shown a significant error reduction in both Root Mean Squared Error and Mean Absolute Percentage Error compared to (Venkataramana Veeramsetty, 2020) that forecasted an hour ahead power load using 3 hours earlier and 4 days back from the same time.

7.2. Recommendation and Future Work

In this study, we encountered a dataset with a large percentage of missing values and hence had to use only a portion of the dataset. If more datasets with accurate values could be collected, forecasting for longer horizons would be possible. Future work can work with hybrid models such as LSTM-RNN and CNN-LSTM. In the future, this research can be extended to forecast for longer forecasting horizons using multistep forecasting other electricity load influencing factors.

Special Acknowledgment: This research was funded by Adama Science and Technology University with a grant number **ASTU/SM-R/500/22**

REFERENCE

Abbasimehr, H., Shabani, M., & Yousefi, M. (2020). An optimized model using LSTM network for demand forecasting. *Computers & Industrial Engineering*, *143*, 106435. https://doi.org/10.1016/j.cie.2020.106435

Alicia Tuovila. (2022, September 17). *Forecasting Definition*. https://www.investopedia.com/terms/f/forecasting.asp

Almeshaiei, E., & Soltan, H. (2011). A methodology for Electric Power Load Forecasting. *Alexandria Engineering Journal*, *50*(2), 137–144. https://doi.org/10.1016/j.aej.2011.01.015

Ayush Gupta. (2022). *A Comprehensive Guide on Deep Learning Optimizers*. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

Bikal Basnet. (2016). *LSTM Optimizer Choice? – Data Science & Deep Learning*. https://deepdatascience.wordpress.com/2016/11/18/which-lstm-optimizer-to-use/

Chen, C. S., Tzeng, Y. M., & Hwang, J. C. (1996). The application of artificial neural networks to substation load forecasting. *Electric Power Systems Research*, *38*(2), 153–160. https://doi.org/10.1016/S0378-7796(96)01077-2

Chng, Z. M. (2022, April 27). Google Colab for Machine Learning Projects. *Machine Learning Mastery*. https://machinelearningmastery.com/google-colab-for-machine-learning-projects/

docker hub. (2022). *python—Official Image | Docker Hub*. https://hub.docker.com/_/python

Gabreyohannes, E. (2010). A nonlinear approach to modelling the residential electricity consumption in Ethiopia. *Energy Economics*, *32*(3), 515–523. https://doi.org/10.1016/j.eneco.2009.08.008

Gebreyohans, G., Saxena, N. K., & Kumar, A. (2018a). Long-Term Electrical Load Forecasting of Wolaita Sodo Town, Ethiopia Using Hybrid Model Approaches. *2018 IEEE 8th Power India International Conference (PIICON)*, 1–6. https://doi.org/10.1109/POWERI.2018.8704408

Gebreyohans, G., Saxena, N. K., & Kumar, A. (2018b). Long-Term Electrical Load Forecasting of Wolaita Sodo Town, Ethiopia Using Hybrid Model Approaches. *2018 IEEE 8th Power India International Conference (PIICON)*, 1–6. https://doi.org/10.1109/POWERI.2018.8704408

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

Hasan-Al-Shaikh, Rahman, Md. A., & Zubair, A. (2019). Electric Load Forecasting with Hourly Precision Using Long Short-Term Memory Networks. *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 1–6. https://doi.org/10.1109/ECACE.2019.8679244

IGI Global. (2022). *What is Artificial Neural Network (ANN) | IGI Global*. https://www.igi-global.com/dictionary/artificial-neural-network-ann/1518

Indeed Editorial Team. (2021). *What Is MAPE? (Plus How To Calculate MAPE in 3 Steps) | Indeed.com*. https://www.indeed.com/career-advice/career-development/what-is-mape

International Energy Agency. (2021). *Ethiopia Energy Situation—Energypedia*. https://www.iea.org/statistics/statisticssearch/report

Jian Zheng, Cencen Xu, Ziang Zhang, & Xiaohua Li. (2017). Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network. *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, 1–6. https://doi.org/10.1109/CISS.2017.7926112

Kuster, C., Rezgui, Y., & Mourshed, M. (2017). Electrical load forecasting models: A critical systematic review. *Sustainable Cities and Society*, *35*, 257–270. https://doi.org/10.1016/j.scs.2017.08.009

Lopez-Martin, M., Sanchez-Esguevillas, A., Hernandez-Callejo, L., Arribas, J. I., & Carro, B. (2021). Novel Data-Driven Models Applied to Short-Term Electric Load Forecasting. *Applied Sciences*, *11*(12), 5708. https://doi.org/10.3390/app11125708

Metaxiotis, K., Kagiannas, A., Askounis, D., & Psarras, J. (2003). Artificial intelligence in short term electric load forecasting: A state-of-the-art survey for the researcher. *Energy Conversion and Management*, 10.

pandas. (2022). *pandas—Python Data Analysis Library*. https://pandas.pydata.org/

Potapov, V., Khamitov, R., Makarov, V., Gritsay, A., Chervenchuk, I., & Tyunkov, D. (2018). Short-Term Forecast of Electricity Load for LLC "Omsk Energy Retail Company" Using Neural Network. *2018 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, 1–5. https://doi.org/10.1109/Dynamics.2018.8601430

Setiawan, A., Koprinska, I., & Agelidis, V. G. (2009). Very short-term electricity load demand forecasting using support vector regression. *2009 International Joint Conference on Neural Networks*, 2888–2894. https://doi.org/10.1109/IJCNN.2009.5179063

Shao, X., Kim, C.-S., & Sontakke, P. (2020). Accurate Deep Model for Electricity Consumption Forecasting Using Multi-channel and Multi-Scale Feature Fusion CNN–LSTM. *Energies*, *13*(8), 1881. https://doi.org/10.3390/en13081881

simplilearn. (2021). *What is Keras and Why it so Popular in 2021 | Simplilearn*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras

tensorflow. (2022). *TensorFlow*. https://www.tensorflow.org/

Torres, J. F., Martínez-Álvarez, F., & Troncoso, A. (2022). A deep LSTM network for the Spanish electricity consumption forecasting. *Neural Computing and Applications*, *34*(13), 10533–10545. https://doi.org/10.1007/s00521-021-06773-2

Zor, K., Timur, O., & Teke, A. (2017). A state-of-the-art review of artificial intelligence techniques for short-term electric load forecasting. *2017 6th International Youth Conference on Energy (IYCE)*, 1–7. https://doi.org/10.1109/IYCE.2017.8003734

APPENDIXES

Appendix A: Sample Collected Dataset

Substation	Feeder Name	Actual meter reading date	Field of type TIMS	Interval Length	D_KW
Goffa	Incomer 1	3/21/2019	3:15:00 AM	15	464.00
Goffa	Incomer 1	3/21/2019	3:30:00 AM	15	464.00
Goffa	Incomer 1	3/21/2019	3:45:00 AM	15	475.00
Goffa	Incomer 1	3/21/2019	4:00:00 AM	15	478.00
Goffa	Incomer 1	3/21/2019	4:15:00 AM	15	501.00
Goffa	Incomer 1	3/21/2019	4:30:00 AM	15	506.00
Goffa	Incomer 1	3/21/2019	4:45:00 AM	15	543.00
Goffa	Incomer 1	3/21/2019	5:00:00 AM	15	574.00
Goffa	Incomer 1	3/21/2019	5:15:00 AM	15	643.00
Goffa	Incomer 1	3/21/2019	5:30:00 AM	15	718.00
Goffa	Incomer 1	3/21/2019	5:45:00 AM	15	838.00
Goffa	Incomer 1	3/21/2019	6:00:00 AM	15	967.00
Goffa	Incomer 1	3/21/2019	6:15:00 AM	15	1,192.00
Goffa	Incomer 1	3/21/2019	6:30:00 AM	15	1,462.00

Appendix B: Sample Code to Preprocess the Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import plotly
import numpy as np
def missing_value_percentage(dataset):
    return (dataset.isnull().sum()/len(dataset))*100
df = pd.read_excel("E:/Research/Data/New demand load/orignal_demand_feeder1_gofa.xlsx")
df['datetime'] = df['date'].astype(str) +" "+ df['time'].astype(str)
#sort
df.sort_values(by='datetime', inplace=True)
#remove duplicates
df.drop_duplicates(subset='datetime',keep='first',inplace=True)
# save the result to a file
df.set_index('datetime',inplace=True)

df.index= pd.to_datetime(
                          df.index,
                          format='%Y-%m-%d %H:%M:%S')
new_index = pd.date_range(start='2019-03-21 00:00:00 ', end='2022-04-13 00:00:00', freq='15min')
df=df.reindex(index=new_index)
for i in range(len(df)):
    if df['D_KW'][i] < 0:
        df['D_KW'][i] = 0
    elif df['D_KW'][i] >30000:
        df['D_KW'][i] = 30000
```

```
for i in range(len(df1_no_missing)):
    if pd.isna(df1_no_missing['D_KW'][i]):
        df1_no_missing['D_KW'][i]= df1_no_missing['D_KW'][i-96]
df.index=pd.to_datetime(df.index)
df=df.resample('d').max()
df.plot(figsize=(30,10),xlabel="Datetime", ylabel="Load in KW")
df.plot(figsize=(30,10),xlabel="Datetime", ylabel="Load in KW")
scaler=MinMaxScaler(feature_range=(0,1))
dataset=df.values
dataset=scaler.fit_transform(dataset)
train_size=int(len(dataset) * 0.8)
validation_size=int(len(dataset)*0.9)
train,validation,test=dataset[0:train_size,:],dataset[train_size:\
        validation_size,:],dataset[validation_size:len(dataset),:]
def to_sequence(dataset,seq_size=1):
    x=[]
    y=[]
    for i in range(len(dataset)-seq_size-1):
        window=dataset[i:(i+seq_size),0]
        x.append(window)
        y.append(dataset[i+seq_size,0])
    return np.array(x),np.array(y)
seq_size=1
trainX,trainY=to_sequence(train)
validX,validY=to_sequence(validation)
testX,testY=to_sequence(test)
print("Shape of training set:{}".format(trainX.shape))
trainX=np.reshape(trainX,(trainX.shape[0],1,trainX.shape[1]))
validX=np.reshape(validX,(validX.shape[0],1,validX.shape[1]))
testX=np.reshape(testX,(testX.shape[0],1,testX.shape[1]))
```

Appendix C: Sample Code to Create, Compile and Train the Proposed Model

```
model=Sequential()
model.add(LSTM(32,return_sequences=True,input_shape=(None,seq_size)))
model.add(LSTM(32,input_shape=(None,seq_size)))
model.add(Dense(1))
adam=optimizers.Adam(learning_rate=0.1,beta_1=0.9,beta_2=0.999,epsilon=None, decay=0.0,amsgrad=False)
model.compile(loss='mean_squared_error',optimizer='adam')

model.summary()
model.fit(trainX,trainY, validation_data=(validX,validY),epochs=20,verbose=2,batch_size=20 )
#Loss graph
training_loss = model.history.history['loss']
validation_loss = model.history.history['val_loss']
epoch_count = range(len(training_loss))
plt.plot(epoch_count, training_loss)
plt.plot(epoch_count, validation_loss)
plt.legend(['Training Loss', 'Val Loss'])
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()
```

Appendix D: Sample Code for Evaluation

```
testPredict=model.predict(testX)
testPredict=scaler.inverse_transform(testPredict)
testY=scaler.inverse_transform([testY])
testPredict=testPredict[1:]
plt.plot(testY[0])
plt.plot(testPredict)
plt.legend(["Actual Load", "Predicted Load"])
plt.ylabel("Load in KW")
plt.rcParams["figure.figsize"] = (20,6)
plt.show()
def RMSE(Y_actual,Y_Predicted):
    rmse = math.sqrt(mean_squared_error(Y_actual,Y_Predicted))
    return rmse

def MAE(Y_actual,Y_Predicted):
    mae = np.mean(np.abs(Y_actual - Y_Predicted))
    return mae
def MAPE(Y_actual,Y_Predicted):
    mape = np.mean(np.abs((Y_actual - Y_Predicted)/Y_actual))*100
    return mape
testScore_RMSE=RMSE(testY[0][:-1],testPredict[:,0])
testScore_MAPE=MAPE(testY[0][:-1],testPredict[:,0])
print("Test score: %.2f RMSE"%(testScore_RMSE))
print("Test score: %.2f MAPE"%(testScore_MAPE))
```