Quantitative Trading in North American Power Markets

Dominic Isaac Andoh

A Thesis
in the Department
of
Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Science (Mathematics) at
Concordia University
Montréal, Québec, Canada

January 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis is prepared

By: Mr. Dominic Isaac Andoh

Entitled: Quantitative Trading in North American Power Markets

and submitted in partial fulfillment of the requirements for the degree of

Master of Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Chair
Dr. Cody Hyndman

Examiner
Dr. Frederic Godin

Examiner
Dr. Arusharka Sen

Thesis Supervisor(s)
Dr. Cody Hyndman

Thesis Supervisor(s)
Dr. Xiaowen Zhou

Approved by:
Dr. Galia Dafni — Chair of Department or Graduate Program Director

Dr. Pascale Sicotte — Dean

Abstract

Quantitative Trading in North American Power Markets
Dominic Isaac Andoh

Short-term load forecasting (STLF) in electrical grids is critical for efficiency and reliability. Many countries in the west have deregulated their electric power industry, allowing for a free and competitive market; this has made load forecasting a more critical task for estimating future spot prices. Load forecasting is a complex task due to seasonal variation and the non-stationarity of historical load data. Also, multicollinearity among exogenous variables adds to the complexities of STLF. We use data provided by Plant-E Corp, an investor in the New York Control Area electricity market. We propose STLF models and test them against the benchmark New York Independent System Operator (NYISO) model; we propose three different models for STLF. The first is a hybrid model consisting of a clustering part and a weighted Euclidean distance norm component; we name it the `Cluster-WED` model. The other two models are deep learning models we shall call `BiLSTM` and `AttnLSTM` model. Both are based on the encoder-decoder architecture. The encoder part of the `BiLSTM` model comprises a bidirectional layer. The decoder is a unidirectional LSTM layer. We incorporate the attention mechanism into the `AttnLSTM` model, where we assign weights to all the hidden states of the encoder before making the next predictions in the decoder. The encoder and decoder for the `AttnLSTM` model are both unidirectional LSTM layers.

Though none of our proposed models outperformed the NYISO model due to the disparities in the input information, the results from the `Cluster-WED` model prove that we can perform a complex task like STLF using simple nonlinear models. Also, the results from the `BiLSTM` model demonstrate the applicability of deep learning when adapted for structured time series data. The generalization and consistency of the `BiLSTM` model suggest that we could achieve competitive performance against the benchmark once the information gap is bridged.

Keywords: Short-term load forecast, clustering, weighted Euclidean distance, deep learning, bidirectional, unidirectional, attention.

Acknowledgements

My utmost and profound gratitude is to my supervisor Dr. Cody Hyndman; without whose support, I would not have completed this august institution, Concordia. I am very grateful for the numerous support you gave me. Not forgetting your constructive criticisms; they challenged me to become a better version of myself.

I would also use this opportunity to thank my co-supervisor, Dr. Xiaowen Zhou, who also supported me during my studies. I want to express my sincere gratitude to Dr. Galia Dafni for her wise counsel. I also thank the staff and management of Plant-E Corp for allowing me to work with them. It was a great experience working with them; I am thankful for their numerous support and contributions towards this work.

Finally, I would like to thank my beloved wife, Obaahemaa Rhoda. She has been a pillar of support and encouragement through all challenging times. Your constant encouragement gave me hope and strength to keep pushing myself to the very limit. Though it is hard being apart, I hope the future will be kind to bring us together, never to be apart.

To all family and friends, many and sundry, who have one way or the other contributed to this journey of life, I say thank you.

Contents

List of Figures vi

List of Tables viii

1 Introduction 1

2 Literature Review 4
- 2.1 Classical Statistical Models . . . 4
- 2.2 Machine Learning Models . . . 5
- 2.3 Deep Learning Models . . . 7
  - 2.3.1 Bidirectional RNN . . . 7
  - 2.3.2 RNNs with Attentions . . . 8
- 2.4 Hybrid Models . . . 10

3 Cluster-WED 14
- 3.1 K-means Clustering . . . 15
- 3.2 Dominance Analysis . . . 18
- 3.3 Input Data Analysis & Parameter Identification . . . 19
  - 3.3.1 Cumulative Temperature & Humidity Index (CTHI) . . . 19
- 3.4 `Cluster-WED` Architecture . . . 25
- 3.5 Chapter Summary . . . 27

4 Recurrent Neural Network 28
- 4.1 Adam Optimization Algorithm . . . 31
- 4.2 Seq2Seq Bidirectional LSTM Architecture . . . 32
- 4.3 LSTM with Luong Attention . . . 37
- 4.4 Chapter Summary . . . 38

5 Numerical Implementation & Results 40
- 5.1 Analysis of Results . . . 43
- 5.2 Chapter Summary . . . 48

6 Conclusion 55

Bibliography 57

List of Figures

3.1 A scattered plot of K-means clustering on a data with two variables TS_1 and TS_2. The plots shows seven clusters with their corresponding centroids. 17
3.2 Daily Average load against daily average weather variables, 2015 21
3.3 Daily load curves of NY 2015 January and February 22
3.4 Load curves of 1 Jan. 2018 (Monday) and its similar days of NY 23
3.5 Load curves of 1 Jan. 2018 (Monday), weekend and anomalous days of NY 24
3.6 Detailed architecture of the `Cluster-WED` model.The day-type data is clustered into pre-defined K-samples. We select the five closest days to the forecast day from the forecast cluster C_f. the average of these days become the 24-hour forecast (Section 3.4). 26

4.1 Unfolded RNN architecture 28
4.2 Graph showing dependencies for RNN model modified from [59], with three timesteps. Plain boxes represent variables and shaded parameters. Circles represent operators. . 29
4.3 A single cell LSTM architecture 34
4.4 LSTM encoder-decoder (`BiLSTM`) architecture. The encoder is bidirectional and the decoder is unidirectional. 36
4.5 LSTM encoder-decoder with Loung attention (`AttnLSTM`) model. The encoder and the decoder are unidirectional. 38

5.1 New York Control Area Load Zones, NYISO website 40
5.2 Real-load and load-forecast concatenation. For each calendar day in the load time series, we concatenate the first 5 hours of the real-load with the last 19 hours of the NYISO-forecast to give us the adjusted-load. 43
5.3 Input-output overlapping windows. At sample 0, we move the window from the 0^{th} to the 47^{th} member of the time series; this becomes our sample 0 input mapping the following 24 observations as output. In sample 1, we move the input window one timestep forward and repeat the process. 44
5.4 Plot of loss function, `BiLSTM`. 44
5.5 Plot of loss function, `AttnLSTM`. 45
5.6 Hourly plots of real and predicted loads by `Cluster-WED` and `BiLSTM`, 2019 49
5.7 Daily average plots of real-load and `BiLSTM` forecast, 2019 49
5.8 Daily average plots of real-load and `Cluster-WED` forecast, 2019 50
5.9 Residual distribution of `BiLSTM`, `AttnLSTM` and `Cluster-WED`, 2019 51
5.10 Fitted t-distribution for `BiLSTM` residuals with 4.16 degree of freedom. 52

5.11 Fitted t-distribution for `AttnLSTM` residuals with 4.33 degree of freedom. 52
5.12 Fitted t-distribution for `Cluster-WED` residuals with 5.58 degree of freedom. 53
5.13 Residuals of `AttnLSTM`, `Cluster-WED` and `BiLSTM`, 2019 53

List of Tables

1.1 Comparison of conventional and artificial intelligence (AI) based models [22] 2

2.1 Comparison of three attention-based models, based how spatial-temporal information is extracted and how final predictions are made [21]. 13

3.1 Additional variance contribution computation for x_2 and x_3 19
3.2 Relative importance of x_1, x_2 and x_3.The first two columns show the subset models and their corresponding $R^2_{Y \cdot X}$ values. 20
3.3 RSD metric for successive days and Saturdays 22
3.4 RSD metric for Sundays and Mondays 23
3.5 RSD metric of the anomalous day compared with similar days and other weekend-anomalous days 24
3.6 Day-type codes are numeric values assigned to each day of the week. We use the codes to sample the historic input dataset for the day we wish to forecast. 25
3.7 Input data set dimensions 25

4.1 Activation functions 29
4.2 `BiLSTM` layer specifications and hyperparameters. After some repeated trials with different values, this set of hyperparameter values gave the best results. 37
4.3 `AttnLSTM` layer specifications and hyperparameters. After several repeated trials with different values, this set of hyperparameter values gave the best model results. 39

5.1 Weather station weights imputed to each zone [41] and applied to the NYISO model. 41
5.2 City-weather weights imputed to each zone and applied to the `Cluster-WED, BiLSTM` and `AttnLSTM` models. The weights are computed using the linear combination approach proposed by [50]. 42
5.3 APEs of `BiLSTM` model on some in-sample and out-sample data. APE compares the hourly forecast against the actual hourly observation and expresses the absolute error as a percentage of the actual observation. 46
5.4 APEs of `AttnLSTM` model on some in-sample and out-sample data. 46
5.5 APE statistics of the eleven zones of NYCA trained with `BiLSTM` and tested on 2019 data. 47
5.6 APE statistics of the different models after testing on the out-sample 2019 data. . . . 47
5.7 MAPE statistics of the different models after testing on the out-sample 2019 data. MAPE compares the mean daily forecast against the actual daily mean observation and expresses the absolute error as a percentage of the actual daily mean observation. 48

Chapter 1

Introduction

Electricity load forecasting is very crucial in power systems management for several reasons, namely, the sound management of many tasks such as market trade, day-ahead demand and outage planning, energy storage management, plant maintenance, to name a few. A long-term forecast of peak electricity demand is needed to plan the generation and distribution of electricity [36]. To control and plan power systems, electricity service providers need short-term load forecasts (STLF), more importantly for estimating future spot prices [36].

Many countries in the west are deregulating their power industry. The understanding is that a competitive free electricity market would enhance the reliability of energy supply at a low-cost [39]. Usually, an electricity market comprises two bodies to facilitate trade between consumers and producers; the *pool*, an eCommerce marketplace, and a framework that enables bilateral contracts. Power producers submit their loads and their corresponding prices at the pool level. Transmission firms, utility distributors and other players also present their needs and their respective prices [39].

The market operator uses a market-clearing tool based on a single-round auction [48]. The market operator makes the market clearing prices public with no other information. Producers and consumers (transmission & distribution firms, market participants) rely on forecast information to prepare their trading strategies [39]. A producer with a good forecast for the following day's market-clearing prices can develop an optimum bidding technique to maximize profit. In the same way, a consumer can also devise a strategy to maximize one's utility. There are grave consequences for wrong predictions; Bunn and Farmer [9] estimated that for any 1% increase in error when forecasting U.K. electrical load, there is a corresponding increase of £10 million in operating cost annually.

Several classical approaches such as regression and other complex statistical techniques for STLF require heavy computations due to seasonal variation and the non-stationary nature of load data. Jain and Satish [22] classified STLF approaches into two main categories. One approach treats load patterns as time series analysis. The other recognizes that load demands are heavily dependent on weather variables and finds a functional relationship between the weather variables and load demands.

The time-series approach does not utilize weather information, and most regression methods use a piece-wise linear relationship between weather parameters and load without any justification [22]. However, studies have shown that the functional relationship between weather variables and load demands follows spatio-temporal patterns. The time series and regression approaches converge slowly and, in some instances, may diverge. Jain and Satish [22] compared machine learning, also known as artificial intelligence (AI) methods, to the conventional time series and regression approach. Table 1.1 shows the superiority of the AI-based model to the classic statistical models in terms of their

flexibility, computations and mathematical assumptions.

Important features	Time series Model	Regression Model	AI Model
Load information	Considered	Considered	Considered
Weather information	Ignored	Considered	Considered
Functional relation between load and weather variables	Ignored	Required	Not required
Complex mathematical calculations	Required	Required	Not required
Time required for predictions	More	More	Less
Adaptability	Less	Less	More

Table 1.1: Comparison of conventional and artificial intelligence (AI) based models [22]

However, due to seasonal variation and non-stationarity in the time series data, traditional machine learning algorithms like support vector machines (SVM), random forest, and others fail to capture the sequential and seasonal patterns for accurate load forecast [47]. In contrast, deep learning models yield better forecasting accuracy [47]. Deep learning algorithms are widely studied and applied in different data science domains such as natural language processing (NLP), video summarization, image classification and many more. STLF strives to achieve enhanced accuracy by integrating different deep learning and hybrid models [47].

Jain and Rao [24] proposed a hybrid technique of combining the weighted Euclidean distance measure between weather parameters of the forecast day and historic days, with clustered data subsets to obtain hourly load forecasts. The authors considered hourly load, temperature, humidity and the day-type variables of the dataset. The input dataset for the clustering algorithm follows a distinct day-type similarity criterion for normal and anomalous days.

A recurrent neural network (RNN) is a deep learning architecture that has proved very efficient in NLP. In recent years researchers have successfully adapted RNN for STLF. In theory, RNNs can capture the temporal correlation of time series by using a historical sequence of arbitrary length [61]. However, the shortcoming of exploding or vanishing gradient during traditional RNN training has necessitated the development of long short-term memory (LSTM) cells that can learn long-term dependencies. Unlike the RNN cells, where neurons apply activation functions on a linear set of inputs, LSTMs, use memory cells that contain input, forget, and output gates. The cells use these gates to eliminate the exploding/vanishing gradient effects. Wang et al. [58] applied the LSTM model for an STLF, proving superior with excellent generalization capabilities.

Compared to other machine learning models, one advantage of all RNN cells is their ability to recall historical information related to previous time steps, which is essential for time-series predictions [4]. The basic LSTM model is the unidirectional network, which only learns the current state through past states' information, but getting information from future time series is an added advantage. Hence the bidirectional architecture. Several researchers have implemented the bidirectional LSTM architecture for STLF. For instance, Atef and Eltawil [4] investigated the performance of stacked layers of the unidirectional and bidirectional LSTM models for STLF. The bidirectional

LSTM model outperformed all the unidirectional LSTM models. This thesis proposes a bidirectional LSTM sequence-to-sequence architecture, which we shall call `BiLSTM` for a day-ahead load forecast of the New York Control Area (NYCA).

The sequence-to-sequence architecture has one main limitation: the encoder summarizes information from the input sequence into a fixed dimension vector irrespective of the length of the input sequence. This bottleneck architecture of the sequence-to-sequence leads to the loss of essential information when summarizing long input sequences into a fixed dimensional vector. To find a more robust STLF, we propose another model with an attention mechanism. We call this LSTM attention-based model `AttnLSTM`.

The attention mechanism is a new technique that is recently emerging in deep learning and has since been applied successfully in many data science tasks. The attention mechanism is a physiological process that initiates selective attention in the human brain and its visual system [35]. When processing images, optical modules in the human cerebral cortex determine the target area that needs more focus and allocates more attention and resources to the target area. The attention mechanism in deep learning is conceptually similar to this selective visual attention of the human brain [35].

In this thesis, we consider a temporal attention mechanism proposed by Luong et al. [34]; they proposed a global and local attention mechanism. The global attention model considers all hidden states of the encoder when computing the context vector for the next prediction. In contrast, its counterpart, the local attention, considers some of the hidden states. In an STLF task, each part of the input sequence may have features influencing forecast results and these features are distributed across the input sequence [35]. Hence extracting only partial data for analysis distorts the temporal integrity of the input sequence. In the `AttnLSTM` architecture, we consider the global attention mechanism.

Plant-E Corp is an investor in the New York Control Area (NYCA) power markets, based in Montreal, and all data used in this research are sourced from Plant-E Corp. This research aims to propose and develop a short-term load forecast (STLF) for the entire NYCA. We started with the hybrid model proposed by Jain and Rao [24] and called it the `Cluster-WED` model. However, we make some changes like adding CTHI as an input variable to temperature and humidity and using Dominance Analysis to assign weights of importance to the exogeneous input variables instead of the least square method recommended by Jain and Rao [24]. To develop a more robust and consistent model, we next considered the two deep learning models, `BiLSTM` and `AttnLSTM`. We compared our forecast results with the New York Independent System Operator (NYISO) forecasts as our baseline model.

The remainder of this thesis is organized as follows: Chapter 2 provides a literature review that summarizes and expounds different researchers' work related to electricity load forecasting. Chapter 3 introduces K-means clustering and gives a detailed description of its computations. We explain the Dominance Analysis and give a detailed description of the `Cluster-WED` architecture. Chapter 4 introduces recurrent neural networks, gives a detailed description of the `BiLSTM` and `AttnLSTM` models, and introduces the `Adam` optimization algorithm; Chapter 5 presents the results from our analysis. In Chapter 6, we give conclusions and remarks.

Chapter 2

Literature Review

Two main aspects of an accurate short-term load forecast (STLF) are identifying appropriate variables and selecting suitable techniques for implementation [24]. Khotanzad et al. [26] considered the effect temperature and humidity have on STLF. Khotanzad et al. [27] looked at the effect of wind speed and humidity with a linear transformation of temperature in their work. Amjady [3] presented a time series approach that can accurately forecast the hourly loads on weekdays, weekends, and public holidays.

Methods for STLF are based on statistical models, machine learning models like neural networks, support vectors, fuzzy logic and many other approaches. With advancements in computer capabilities, robust models like recurrent neural networks (RNN), deep learning, and other hybrid models have recently emerged in applications to STLF.

2.1 Classical Statistical Models

Several statistical techniques have been adapted for STLF, ranging from linear and non-linear regression, curve fitting, least-square approximation and time series analysis. Papalexopoulos and Hesterberg [42] proposed a linear regression-based model to calculate STLF. Their model included accurate holiday modelling using binary variables, temperature modelling using heating and cooling degree functions and parameter estimation by weighted least-squares linear regression techniques. Taylor and McSharry [54] used electricity demand data from 10 European countries as the basis for an empirical comparison of a day-ahead load forecasting. The authors analyzed performances of the Autoregressive Integrated Moving Average (ARIMA), the periodic autoregressive (AR), and the principal component analysis (PCA) models. The results were disappointing for the AR model, but ARIMA and the PCA methods performed well.

Elsaraiti et al. [14] applied a seasonal ARIMA (SARIMA) model to forecast electricity consumption in southern Tripoli, Libya, for two weeks. The authors used the auto-correlation (ACF) and the partial autocorrelation function (PACF) to determine (p,d,q). The best-performing model achieved a mean absolute percentage error (MAPE) of 4.33%. Kaur and Ahuja [25] compared the performance of the SARIMA and ARIMA techniques, applying both methods to 11 years' healthcare's electricity demand data. The comparative analysis showed that the SARIMA model had superior performance than the ARIMA model. Many researchers have adapted several other time-series approaches to forecast electricity demand. For instance, Lee et al. [32] compared the performance of the simple moving

average (SMA), weighted moving average (WMA), simple exponential smoothing (SMA), the Holt linear trend (HL), Holt-Winter (HW) and centred moving average (CMA) on the monthly electricity load consumption data. The HW model is a time series forecasting technique based on exponential smoothing. It considers trend and seasonality whiles forecasting; the HW model gave the smallest MAPE value.

Fumo and Biswas [16] implemented simple and multiple linear regression and quadratic regression analysis on hourly and daily load data. They realized that the time interval of the data was significant in determining the quality or performance of the model. The results illustrated that as the time granularity of the data increases, the quality of their model increases; this is partly because, with more extended time intervals, discrepancies that are prevalent in shorter time intervals average out. They also suggested the quadratic regression could offer better results for the shorter time intervals. Again using multiple regression, Vu et al. [56] predicted monthly load demands, using historical load data of New South Wales in Australia. With multicollinearity and backward elimination processes, [56] selected the most appropriate variables before applying the multiple regression. The analysis results showed that temperature, humidity, and rainy days predominantly affect the state's electricity consumption. The monthly forecast from their model closely matched the actual demands. Amber et al. [2] compared the performance of a multiple regression model to a genetic programming model to forecast the daily load consumption of buildings. The genetic programming model outperformed the multiple regression model. Also, using temperature and humidity, Braun et al. [7] predicted gas and electricity consumption of a supermarket based on multiple regression.

2.2 Machine Learning Models

With increasing computing power and advancement in artificial intelligence, STLF techniques have seen breakthroughs in efficiently and accurately dealing with different load scenarios [24]. Popular artificial intelligence techniques used for STLF include artificial neural networks, genetic algorithms, fuzzy logic systems, support vector systems, and others [24]. Jain and Satish [23] proposed a clustering-based STLF using support vector machines (SVM). They calculated the daily average load of each day for all training and testing patterns. They then clustered using a threshold value between the daily average load of the testing and training patterns. The results showed that clustering the input patterns yielded a more accurate forecast.

Paudel et al. [43] did similar work using SVM to predict the energy consumption of low energy buildings (LEB). The authors first divided their training datasets into two categories, "all data" and "relevant data". The "relevant data" uses a small section of the daily dataset that represents and addresses the non-linear dynamics and complexity of the building. Numerical results showed that the "relevant data" modelling approach performed better than the "all data" modelling approach.

Bogomolov et al. [6] took an entirely different approach by using human activities aggregated from anonymized phone data as a proxy to model energy consumption. Using the energy consumption dataset and mobile phone records dataset, Bogomolov et al. [6] could predict average daily consumption and the peak energy consumption for the next seven days. The authors formulated the task as a non-linear regression on 32 dimension features extracted from the mobile phone dataset as the independent variable and the load consumption data as the response variable. The authors used ensemble learning methods (random forest) to solve the optimization problem. From a preliminary analysis of the response variable (electricity consumption), the authors identified three clusters based on the temporal regularities of the response variable. These regularities were characteristics of dif-

ferent locations of the province, namely: the residential areas, touristic areas, the city center, and the industrial areas.

Consumption behaviour of the residential areas showed uneven seasonality on a weekly scale. It also depicted a varying seasonality during day and night. The consumption also varies during the weekdays and low during holidays. The touristic areas showed uneven seasonality on a weekly scale, a particularly upward sloping trend toward holidays, and an abnormally high load during holidays. Interestingly, Bogomolov et al. [6] found no significant differences in consumption of the city and the industrial areas. Both showed stationary seasonality weekly and during day or night. Also, both showed steady consumption during weekdays and low during weekends.

The fuzzy logic model is a machine learning algorithm that maps a set of input variables to output variables using simple `IF-THEN` statements. The inputs and outputs are not necessarily numeric [46]. Pombeiro et al. [44], studied the efficiency of simple non-linear architectures to predict the energy consumption of intelligent buildings. They use explanatory variables like the time of day, weather conditions, and occupancy estimated from WiFi traffic. They developed a fuzzy logic and a neural network system which achieved considerably better performance than linear regression models. Their results suggest that low complexity non-linear models can accurately predict load demands.

Hernández et al. [19] presented an STLF in microgrids based on a three-stage architecture. The first stage of the architecture is a *self-organizing map*, a neural network algorithm for dimension reduction. The second stage is a K-means clustering algorithm. Finally, the third stage predicts the forecast for each cluster using a multilayer perceptron (MLP). This architecture gave smaller error margins than other simple models that do not employ classification and clustering.

Khwaja et al. [28] proposed an improved neural network technique called the boosted neural network (BooNN). This model combines a set of artificial neural networks (ANNs) trained iteratively. The motivation for using boosting is that it can reduce both bias and variance of the prediction error; it is also robust to overfitting. Khwaja et al. [28] first generate an initial model $\hat{h}_i$ at iteration i using the training data that consists of independent variable $\mathbf{X}$ and dependent variable $\mathbf{y}$. At the next iteration, $i+1$, the authors generate a second model by replacing the target output at i with the error of the predictions from the first model at iteration i. Thus,

$$\hat{f}_{i+1} : \mathbf{X} \rightarrow \alpha_i \mathbf{h}_i, \tag{2.2.1}$$

where $\mathbf{h}_i$ is the difference between the target output and the predicted output at the i^{th} model, and α_i is the weight for each model. Khwaja et al. [28] iterate the process until they generate the M number of models. They put all these models together to form a collection of models $\mathcal{F}$ given by:

$$\mathcal{F} = \hat{f}_i, \quad i = 1, 2, \cdots, M. \tag{2.2.2}$$

Finally, [28] derives the ensembled model by the linear combination of the different members of $\mathcal{F}$:

$$f_f = \sum_{i=1}^{M} \alpha_i \hat{f}_i. \tag{2.2.3}$$

The authors discovered that this process reduced the forecast error compared with the single ANN and bagged neural network (BNN).

2.3 Deep Learning Models

According to Sajjad et al. [47], machine learning, deep learning, and hybrid models form the three baseline models for electricity load predictions. However, machine learning models have inadequate prediction accuracy due to multicollinearity in independent variables and electricity consumption. Hence, several studies have been conducted on load forecast using recurrent neural networks as load consumption data follow a time sequence. A typical example is the work of Wang et al. [58], who proposed a model for forecasting energy consumption based on a long-short term memory (LSTM) network, a type of RNN. The authors first extracted secondary features from the raw data using autocorrelation and *mechanism analysis* (a collection of statistical tools employed to understand data behaviour and extract relevant input variables). These secondary features were then used as input to the constructed LSTM model to forecast the sequential data. The empirical results demonstrated that the proposed algorithm has excellent generalization capability.

Agrawal et al. [1] were able to account for the long-term time correlation in the electricity load data of New England over 12 years using an LSTM-based model. They found their model robust with a MAPE value of 6.54%. Also, Zhang et al. [60] conducted an STLF on an hourly load consumption of Toronto from January to July 2016 using an LSTM-based algorithm. The authors found that their algorithm could learn the correlation of the load series and output lower mean square error (MSE) with a short convergence time. They again tested the ability of their model to detect anomalies in data by introducing artificial anomalies. The algorithm was able to detect those anomalies and repair those anomalies adaptively. Zheng et al. [63] used a gated recurrent network (GRU) for STLF. A GRU is a type of RNN like the LSTM but with fewer parameters. Using the least absolute shrinkage and selection (Lasso) and partial correlation analysis, the authors concluded that the average temperature affects load the most among other exogenous variables like humidity, rainfall and wind speed, hence adding temperature as an input variable. The results indicated that the model was faster with similar forecast accuracy than the LSTM model. Taking inspiration from their work, we consider temperature as an input feature in the different STLF architectures proposed.

2.3.1 Bidirectional RNN

Recently, bidirectional RNN has gained popularity due to its ability to factor in historical and future information. Deng et al. [11] used a bidirectional GRU model to predict wind-generated electric power. Wahab et al. [57] presented an architecture for an STLF with limited data based on a two-layer bidirectional sequential model. They named their model the deep derived feature fusion (DeepDeFF). Their model used raw input data and hand-crafted features derived from the primary input features by employing statistical tools to aid the learning of their model. Similarly, Cai et al. [10] presented a multilayer stacked bidirectional LSTM architecture to model STLF. The model extracted features from the data and made accurate predictions.

Atef and Eltawil [4] investigated the performance of stacked unidirectional and bidirectional LSTM architecture on STLF. They compared two and three stacked layers to a single layer for both the unidirectional and bidirectional RNN. Their findings indicated that deep-stacked LSTM layers add no significant improvement in prediction accuracy. Nonetheless, they consume almost twice the computation time as a single-layer model. The single-layered bidirectional LSTM model generally outperformed all other models considered in [4].

2.3.2 RNNs with Attentions

Most multi-step STLF adopt RNN architectures based on the sequence-to-sequence (encoder-decoder) framework. One significant shortfall of this model is that irrespective of the input sequence, the encoder part of the architecture summarizes all the input sequences into a fixed-dimension vector, the encoder-vector. The encoder-vector passes to the decoder for interpretation. This mechanism creates an architectural bottleneck that hampers accurate predictions for long input sequences. Additionally, these models do not consider that input vectors have different impacts on forecast results, so redundant information could interfere with or obscure important information necessary for forecasting accuracy [35].

Bahdanau et al. [5] were the first to propose the attention mechanism, and its superiority is seen across a myriad of tasks like image processing, speech recognition and NLP. Later Luong et al. [34] improved on previous attention models and proposed the global and local attention models. These attention classes differ in terms of whether to place “attention" on all source positions or only a few source positions [34]. The local attention is out of the scope of this thesis.

Global attention considers all hidden states of the encoder when computing the context vector c_t. At each time t, a variable-length alignment vector a_t, whose length is the number of timesteps on the encoder side, is computed based on the hidden state of the previous decoder $\mathbf{H}_{dec}^{t-1}$ and all hidden states of the encoder $\mathbf{H}_{enc}$. The global context vector is the weighted average of a_t overall encoder hidden states $\mathbf{H}_{enc}$, thus $c_t = a_t^\top \mathbf{H}_{enc}$. The softmax of the score function gives the alignment vector a_t. The score function is content-based, and Luong et al. [34] proposed three alternatives :

$$\text{score}(\mathbf{H}_{dec}^{t-1}, \mathbf{H}_{enc}) = \begin{cases} \mathbf{H}_{dec}^{t-1\top} \mathbf{H}_{enc} & \text{(dot)} \\ \mathbf{H}_{dec}^{t-1\top} \mathbf{W}_a \mathbf{H}_{enc} & \text{(general)} \\ v_a^\top \tanh(\mathbf{W}_a[\mathbf{H}_{dec}^{t-1}; \mathbf{H}_{enc}]) & \text{(concat),} \end{cases} \tag{2.3.1}$$

where v_a is a weight vector and $\mathbf{W}_a$ is a trainable matrix.

Lin et al. [33] presented a dual-stage attention RNN (`DA-RNN`) model for STLF. In the first stage, [33] develops a feature-attention-based encoder to compute the correlation of input features with electricity at each timestep. Typical traditional encoder-decoder architecture ignores that various features extracted from the input data contribute differently to load forecasting so that the generated context vector can achieve optimum performance. The feature-attention-based encoder computes the correlation of input features to electricity load at each timestep; thus, assigning higher weights to more relevant features and vice versa. The authors used the input attention mechanism (IAM) proposed by Qin et al. [45]. The proposed IAM adaptively selects the relevant features that a meaningful for predictions.

Given the k^{th} input feature series $\mathbf{X^k} = \{x_1^k, x_2^k, \cdots, x_n^k\} \in \mathbb{R}^n$, where n is the number of observations in the mini-batch. We let d be the number of driving features; we can construct an IAM through a feed-forward dense layer by referring to the previous hidden state $\mathbf{H}_{enc}^{t-1} \in \mathbb{R}^h$ and cell state $\mathbf{C}_{enc}^{t-1} \in \mathbb{R}^h$ in the encoder LSTM, thus:

$$e_t^k = \mathbf{v}_e^\top \tanh(\mathbf{W}_e[\mathbf{H}_{enc}^{t-1}; \mathbf{C}_{enc}^{t-1}] + \mathbf{U}_e \mathbf{X}^k) \tag{2.3.2}$$

$$\alpha_t^k = \frac{\exp(e_t^k)}{\sum_{i-1}^{d} \exp(e_t^i)}, \tag{2.3.3}$$

where $\mathbf{v}_e \in \mathbb{R}^n$, $\mathbf{W}_e \in \mathbb{R}^{n \times 2h}$, and $\mathbf{U}_e \in \mathbb{R}^{n \times n}$ are trainable parameters. We omit the bias terms for

clarity; α_t^k is the attention weight assigning importance to the k^{th} input at time t. We apply the softmax function to e_t^k to ensure that α_t^k sums to 1. IAM is a feed-forward network trained jointly with the LSTM components.

The process of IAM is an attempt to form a new feature representation that would focus on the most relevant features. We denote this new feature representation as:

$$\tilde{\mathbf{X}}_t = (\alpha_t^1 x_t, \alpha_t^2 x_t, \cdots, \alpha_t^d x_t) \in \mathbb{R}^{1\times d} \tag{2.3.4}$$

The encoder output is the set of hidden states at each timestep of the encoder LSTM denoted by $\{\mathbf{H}_{enc}^t\}_{t=1}^n$. Each $\mathbf{H}_{enc}^t$ is a selective expression of the feature at time t given by:

$$\mathbf{H}_{enc}^t = f_{LSTM}^t(\mathbf{H}_{enc}^{t-1}, \tilde{\mathbf{X}}_t) \in \mathbb{R}^{1\times h}, \tag{2.3.5}$$

where f_{LSTM}^t is the function set of the encoder LSTM.

Lin et al. [33] construct a temporal attention-based decoder to extract the time dependencies in the second stage. The authors used the temporal attention by Qin et al. [45], which adaptively selects relevant encoder hidden states at all timesteps. The attention weight for each encoder output $\mathbf{H}_{enc}^i$ at time t is computed based on previous decoder hidden states$\mathbf{H}_{dec}^{t-1} \in \mathbb{R}^p$ and cell state $\mathbf{C}_{dec}^{t-1} \in \mathbb{R}^p$ with

$$l_t^i = \mathbf{v}_d^\top \tanh(\mathbf{W}_d[\mathbf{H}_{dec}^{t-1}; \mathbf{C}_{dec}^{t-1}] + \mathbf{U}_d \mathbf{H}_{enc}^i), \quad 1 \le i \le n \tag{2.3.6}$$

$$a_t = \frac{\exp(l_t^i)}{\sum_{j=1}^n \exp(l_t^j)}, \tag{2.3.7}$$

where $\mathbf{v}_d \in \mathbb{R}^h$, $\mathbf{W}_d \in \mathbb{R}^{h\times 2p}$, and $\mathbf{U}_d \in \mathbb{R}^{h\times h}$ are the parameters to learn. We omit the bias term for clarity. The temporal attention weight a_t represents the importance of the i^{th} encoder hidden state for forecasting. The context vector c_t is the weighted sum of all the encoder hidden-states $\{\mathbf{H}_{enc}^1, \mathbf{H}_{enc}^2, \cdots, \mathbf{H}_{enc}^n\}$, thus:

$$c_t = \sum_{i=1}^n a_t \mathbf{H}_{enc}^i. \tag{2.3.8}$$

Instead of using traditional loss functions, Lin et al. [33] used the pinball loss function between the predicted quantile value $\hat{y}_{n+1}$ and the actual value y_{n+1}. The pinball loss function captures the probabilistic load fluctuations of future forecast and is defined as:

$$L(\hat{y}_{n+1}, y_{n+1}, \Omega, q) = \sum_{i:\hat{y}_{n+1}^i < y_{n+1}^i} \left(1 - \frac{q}{100}\right)(y_{n+1}^i - \hat{y}_{n+1}^i) + \sum_{i:\hat{y}_{n+1}^i \ge y_{n+1}^i} \frac{q}{100}(\hat{y}_{n+1}^i - y_{n+1}^i), \tag{2.3.9}$$

where Ω is the set of all parameters in our model to train, i is the sample index, and $q \in (1, 100)$ is the predefined quantile value.

Prior to [33], Zhang et al. [62] proposed an STLF architecture based on RNN with input attention mechanism and hidden connection mechanism (HCM). HCM implements the residual connection mechanism proposed by He et al. [18] to address the gradient degradation challenge of deep networks.

Theoretically, networks should get better as they get deeper. For instance, given that there are no more features to learn from a dataset by adding extra layers to the network, any additional layer learns the identity mapping. By so doing, information from the previous layer is preserved and can do no

worse than shallower networks. In theory, a network should learn at least the identity mapping if there is no other information. However, in practice, it is difficult to optimize deeper networks. As we add layers to the network, the network gets better until additional layers decrease accuracy. He et al. [18] attempts to solve this gradient degradation problem by using *skip connections* in their residual block networks. Skip connections in neural networks bypass some layers in the network. It then feeds the output of one layer as input to the next layer.

Zhang et al. [62] also used IAM in their proposed architecture; they proposed a novel type of IAM that could bypass the traditional encoder-decoder network, and by doing so, significantly reduce the learnable parameters. Let $\mathbf{X}^k = (x_1^k, x_2^k, \cdots, \mathbf{x}_n^k) \in \mathbb{R}^n$ be the k^{th} feature out of d features of the inputs, and n the number of observations in the mini-batch; then Zhang et al. [62] defines IAM as:

$$h_t^k = \tanh(\mathbf{W}_h \mathbf{X}^k + \mathbf{b}_h) \tag{2.3.10}$$

$$\alpha_t^k = \frac{\exp(h_t^k)}{\sum_{i-1}^{d} \exp(h_t^i)} \tag{2.3.11}$$

$$\tilde{\mathbf{X}}_t = (\alpha_t^1 x_t, \alpha_t^2 x_t, \cdots, \alpha_t^d x_t) \in \mathbb{R}^{1 \times d}, \tag{2.3.12}$$

where $\mathbf{W}_h \in \mathbb{R}^{n \times n}$ is the trainable matrix and $\mathbf{b}_h \in \mathbb{R}^n$ is the bias term. The softmax function is applied to h_t^k to force the attention weights to sum to one.

2.4 Hybrid Models

Increasing complexities of innovative grid technologies have placed high demands for more sophisticated and efficient load forecasting tools. These needs have prompted the proposal of many hybrid techniques, using proven statistical models and emerging machine and deep learning techniques. One such hybrid technique is the work of Jain and Rao [24], which combined weighted Euclidean distance measure with clustered data subsets to forecast hourly loads. The hybrid model proposed by Jain and Rao [24] comes in two parts. In the first part, the authors extract the inputs from historical datasets. The authors applied K-means clustering algorithm to group the input dataset into predefined K clusters; this includes the forecast day vector. The clustering is based on the weather parameter metrics (temperature, humidity). With the forecast cluster, [24] finds the weights for temperature and humidity at the second stage. The weighted Euclidean distance is applied to the forecast cluster to sample five similar days whose average becomes the 24 hourly forecasts. The authors evaluated the performance of the proposed model against the model implemented by NYISO; in most situations, their model outperformed the NYISO model.

Jain and Satish [22] proposed a novel hybrid technique that uses SVM and ANN to predict the next 24 hours load. A typical weekly load profile consists of steady-state, peak and valley components, average components, periodic and random components [22]. Many single module STLF architectures cannot extract all these features from past load data in a well-defined manner for accurate predictions. Jain and Satish [22] proposed a solution to the STLF problem by using statistical and artificial intelligence techniques to forecast the next 24 hours load. Their architecture can factor in the components above by employing four independent modules: basic SVM, peak and valley ANN, Averager & forecaster, and the adaptive combiner.

The basic SVM tracks the steady-state and random component of the load profile and predicts the subsequent 24 hours load (L_d^b); peak and valley ANN tracks and forecasts the peak and valley load of

the next day. The authors arrange the data as inputs mapping to outputs to predict the peak load, with peak load, peak temperature and average temperature serving as the driving inputs and forecasted peak load (PL_d) as the output. The same is done to predict the valley load, but with valley load, valley temperature, and average temperature as the inputs, mapping to forecasted valley load (VL_d). The averager & forecaster comes in two parts. The averager samples the ten latest historical days closest to the day to be forecast in terms of their weather parameters and computes the average of those days' 24 hourly loads (AVL_d). The forecaster part predicts the load by:

$$L_d^f = VL_d + (PL_d - VL_d)(AVL_d), \tag{2.4.1}$$

where VL_d is the forecasted valley load, PL_d is the forecasted peak load, AVL_d the averaged load and L_d^f the forecasted 24 hours load.

The adaptive combiner is a weighted combination of outputs from the basic SVM L_d^b and the output from the forecaster L_d^f. For weekdays like Tuesday, Wednesday, Thursday and Friday, the final 24 hourly loads are computed as

$$L_d = \begin{cases} 0.15L_d^b + 0.85L_d^f & \text{for 0 to 5 hours} \\ 0.10L_d^b + 0.90L_d^f & \text{for 6 to 11 hours} \\ 0.50L_d^b + 0.50L_d^f & \text{for 12 to 17 hours} \\ 0.15L_d^b + 0.85L_d^f & \text{for 18 to 23 hours.} \end{cases} \tag{2.4.2}$$

For weekends (Saturdays, Sundays and holidays) and the day after a holiday, the final 24 hourly loads are computed as

$$L_d = 0.50L_d^b + 0.50L_d^f \quad \text{for 0 to 23 hours.} \tag{2.4.3}$$

The final forecasted load L_d obtained from the adaptive combiner gave a more accurate prediction than those obtained individually from the basic SVM and forecaster. The authors, hence concluded that the performance of the hybrid model is better than the single modules in general.

RNN cells cannot always generalize the temporal and spatial features of electricity load data. Several researchers have used a convolutional neural network (CNN) to extract features and temporal patterns from electricity load data. Kim and Cho [29] proposed a CNN-LSTM neural network model to predict housing energy consumption. The CNN layer can extract several features that affect the electricity consumption forecast, denoising the resultant data. The outputs of the CNN layer become the input to the LSTM layer. The final output of the LSTM layer passes to a fully connected layer for predictions. This novel CNN-LSTM model achieved excellent prediction of electricity consumption. Similarly, Sajjad et al. [47] also proposed a hybrid CNN-GRU-based STLF model. Their proposed model achieved the lowest error rate on household electric power consumption compared with other baselines. The authors found it an effective alternative to previous hybrid models considering computational complexity and prediction accuracy.

Lv et al. [35] presented a hybrid STLF model called the `EGA-STLF` model, based on distributed representation, bidirectional GRU (`BiGRU`) and attention mechanism. When training neural networks with input parameters like days of the week, time, and other non-numerical characters, there is the need to encode these non-numerical variables into characters that computers would understand. For instance, integer encoding encodes non-numerical input variables before training.

Distributed representation is an NLP technique where non-numeric characters like words are rep-

resented as real-valued vectors in a predefined vector space through neural networks. Each element in the distributed vector is a coordinate value of points representing each distinct non-numeric character. The Euclidean distance between any two points with similar meaning in practice is short, and vice versa. Thus, the distributed vector can express correlations between different variables and capture the context information for each variable.

We implement the distributed representation at the data preprocessing level. The encoder comprises a `BiGRU` layer whose primary function is to extract the non-linear characteristics from the input layer and then output values to the decoder for predictions. The decoder comprises a GRU layer and a fully connected layer interacting with the temporal attention mechanism.

Lai et al. [31] proposed a Long-and-Short-Term Time-series Network (`LSTNet`) model based on CNN, recurrent-skip, similar to skip connections mentioned in Section 2.3.2, and AR. RNN models like the GRU and LSTM in shallow layers usually fail to capture long-term correlation in practice [31]. Increasing the depth of the architecture should increase the architecture's performance in theory, but this is not always the case in practice and mostly leads to gradient degradation [18]. Lai et al. [31] developed the recurrent-skip network based on GRU cell units to capture very long-term dependencies patterns to deal with this challenge. The recurrent skip is similar to the work of He et al. [18]; both used skip connections (Section 2.3.2). CNN and the GRU-based recurrent-skip network extract short-term dependency patterns among variables and capture very long-term trends. `LSTNet` also leverages the traditional autoregression model to take care of the volatility insensitive nature of the non-linear CNN and RNN. An ablation study by [31] found that `LSTNet` models without the AR component could not capture the sudden change in variance, hence concluding that AR is generally robust to volatile time series data. Lai et al. [31] incorporated AR as the linear component of their model.

A typical attention mechanism like that of Bahdanau et al. [5] and Luong et al. [34] would place a weight of importance on each of the hidden states from previous timesteps, placing more weights on hidden states that contribute more to the output of the current time and vice versa. Thus, the typical attention mechanism would review information at each previous timestep and select relevant information for the following output. This attention mechanism is very efficient for NLP tasks where each time step corresponds to a single word. However, in a multivariate time series, the attention mechanism fails to ignore noisy variables in terms of forecasting utility [49].

Shih et al. [49] proposed a new attention mechanism to select relevant variables instead of the relevant timestep. They developed a novel time series forecasting model and called it temporal pattern attention LSTM (`TPA-LSTM`). The first layer is an LSTM network that produces the hidden states; a CNN kernel is applied to the hidden states matrix except for the hidden state at the current time to capture time-invariant temporal patterns across multiple timesteps. We next use the novel attention mechanism to select relevant spatial variables from the output matrix of the CNN layer. This spatial attention mechanism seeks to achieve the same purpose as the input attention mechanism proposed by Qin et al. [45] and Zhang et al. [62], the idea is to separate more significant variables for forecasting from less significant ones. Eventually, the attention yields a context vector that encodes spatial and temporal information of the current time step for prediction at the dense layer.

Huang et al. [21] performed a comparative study of the `LSTNet`, `TPA-LSTM` and the `DA-RNN` (Table 2.1). They tested these attention-based models on historical wind power generation and wind speed records to forecast wind power in the short term. After empirical real-world data analysis, the `DA-RNN` yielded the best performance by giving the least mean absolute error. The authors found that the dual attention in the `DA-RNN` architecture caused redundancy in the attention weights since they were compulsory at each timestep. They recommended combining beneficial aspects of the three

	Temporal info.	Spatial info.	Final pred.
LSTNet [31]	GRU-based recurrent skip, CNN	CNN	Dense layer, ARIMA
TPA-LSTM [49]	CNN	LSTM, Attention	Dense layer
DA-RNN [45]	Attention-based LSTM	Attention-based LSTM	Dense layer

Table 2.1: Comparison of three attention-based models, based how spatial-temporal information is extracted and how final predictions are made [21].

models, for instance, skipping some timesteps in the spatial and temporal attention of `DA-RNN`, as demonstrated by recurrent skip of `LSTNet`.

Chapter 3

Cluster-WED

Jain and Rao [24] proposed a hybrid approach of combining the weighted Euclidean distance measure with clustered data subsets to obtain hourly load forecasts. The authors considered hourly load, temperature, humidity and day-type as variables of the dataset. The input dataset for the clustering algorithm follows a distinct day-type similarity criterion for normal and anomalous days.

To implement an efficient, robust and reliable model, Jain and Rao [24] made some assumptions so that their model could accommodate the innovative grid implementations of power systems. Human activities in time resolution are an essential factor that affects load consumption. The week type, day time and season of the year are all-time factors affecting load consumption. To address this aspect of non-stationarity in historical load demand data, the authors proposed a day-type similarity criterion for building the input dataset. We select inputs according to the day of the week.

Jain and Rao [24] divided days of the week into two groups, namely, normal and anomalous days. The authors meant weekdays or working days by normal days, and anomalous days are weekends and holidays. Many works of literature [3, 6] indicate that STLF developed for normal days may not always be appropriate for anomalous days. Jain and Rao [24] adapted a distinct input data selection criterion for normal and anomalous days.

The literature review of Chapter 2 outlines several studies where atmospheric conditions affect load forecast modelling. Exogenous weather and calendar variables like temperature, humidity, wind speed, and others are given significant consideration in load forecast literature. Often such inclusions make STLF tools too sensitive and parameter-dependent [24]. The authors only considered two weather variables, temperature and humidity, in their STLF model.

Subsequently, Jain and Rao [24] presented a clustering and a weighted Euclidean distance measure hybrid approach for STLF of a day-ahead hourly load for normal and anomalous days. The authors considered hourly load, temperature, humidity, and day-type variables of the dataset used. Their model comes in two parts. We group the input data set into a pre-defined number of clusters using K-mean clustering on the weather parameters in tier one. Secondly (tier two), the forecast day subset is reduced further to obtain five similar days to the forecast day using the weighted Euclidean distance norm. The mean hourly loads of the five similar days give the 24 hours load forecast of the day in question. We adapt the work of Jain and Rao [24] to predict STLF for the whole of New York. We then compare the performance of the `Cluster-WED` STLF model with other deep learning STLF models.

3.1 K-means Clustering

Clustering techniques are data analysis techniques applied to multivariate data sets to uncover the underlying structure of the data. It is an unsupervised classification problem, as clusters are formed by evaluating the similarities and dissimilarities of features between different variables. The grouping of cases is based on emergent similarities and not external criteria [37].

K-means clustering technique belongs to a partitioning-based grouping technique that iteratively relocates data points between clusters. It divides the cases of the data into non-overlapping clusters based on the characteristics uncovered [37]. This clustering technique aims to produce cases with high similarity within each group and low similarities between groups [17]. *K*-means clustering is very useful in exploratory data analysis tools. They assist researchers in gaining valuable qualitative and quantitative insight into large multivariate datasets.

A secondary important property of *K*-means clustering is the reduction in data complexity. Faber [15] cited an excellent example of consolidated students' test scores expressed as percentages and categorized into A, B, C, D, and E. The test scores are the data points, and each cluster's point is the average test score in that cluster. That is a classic example of one-dimensional data clustering, where each point is a single measured quantity.

Finally, *K*-means clustering is a good initialization step for a more computationally expensive process. Which often groups data and reduces the data's noise. Several works of literature, like Jain and Rao [24] that use a hybrid technique for STLF, use the *K*-means algorithm as their starting algorithm.

A robust and efficient clustering algorithm uses as few clusters as possible while still capturing all statistically significant clusters. Usually, similarities within clusters measure how close members are to one another. We use different metrics to compute these similarities. Euclidean distance, which is the most utilized, is given by:

$$dE = \sqrt{\sum_{i=1}^{k}(c_i - x_i)^2}, \tag{3.1.1}$$

where c is the cluster center or the centroid, x is the cases we compare to c; and i is the dimension of x or c in comparison. Let k be the total number of dimensions. The following standard metrics used are the squared Euclidean distance:

$$dE^2 = \sum_{i=1}^{k}(c_i - x_i)^2, \tag{3.1.2}$$

the *Manhattan distance*, also called the "city block distance" [52], is a sum of the distance between centroids c and their respective cases x, $dMht = \sum_i^k |c_i - x_i|$; and lastly, the maximum distance between vectors $dMAx = \max_{i=1,\cdots,k} \sum_i |c_i - x_i|$.

Morissette and Chartier [37] presented three major *K*-means clustering algorithms used in the literature. Algorithm 1 is one of those elaborated in their work. We would first need to know the number of clusters present in the data for each algorithm they presented. Since this information is mostly not readily available, multiple trials are often necessary to find the optimum K value [37]. Figure 3.1 shows the implementation of the K-means clustering with two variables, TS1 and TS2. The figure shows a scatter plot in seven clusters and their corresponding centroids.

Algorithm 1 *K-means clustering*.
We let $\mathbf{X_t} = \{TS1_t, TS2_t\}$ for $t = 1, \cdots, T$ be the input vector. Let $S_k = \{t|$ if $\mathbf{X_t}$ belongs to the k^{th} cluster$\}$.

Require: K ▷ We choose the number of clusters.
Require: $c_1, c_2, \cdots, c_K$ ▷ We randomly generate K centroids from our dataset.
Require: I ▷ Maximum number of iteration
Require: dE ▷ We choose the metric measure like equation 3.1.1.
  for $i = 1, \cdots, I$ **do**
    Update cluster assignments:
    for $t, \cdots, T$ **do**
      $\alpha_t = \underset{k=1,\cdots,K}{\arg\max}[dE(c_k - \mathbf{X}_t)]$
    end for
    $S_k \leftarrow t$ ▷ We assign index point of $\mathbf{X}_t$ to the closest cluster S_k according to α_t.
    Update centroid locations:
    for $k = 1, \cdots, K$ **do**
      $c_k \leftarrow \frac{1}{|S_k|} \sum_{t \in S_k} \mathbf{X}_t$ ▷ where S_k is the set of indices of $\mathbf{X}_t$ assigned to the k^{th} centroid.
    end for
  end for
  Update cluster assignments using last centroids
  for $t, \cdots, T$ **do**
    $\alpha_t = \underset{k=1,\cdots,K}{\arg\max}[dE(c_k - \mathbf{X}_t)]$
  end for
  return S_k and c_k ▷ Resulting parameters.

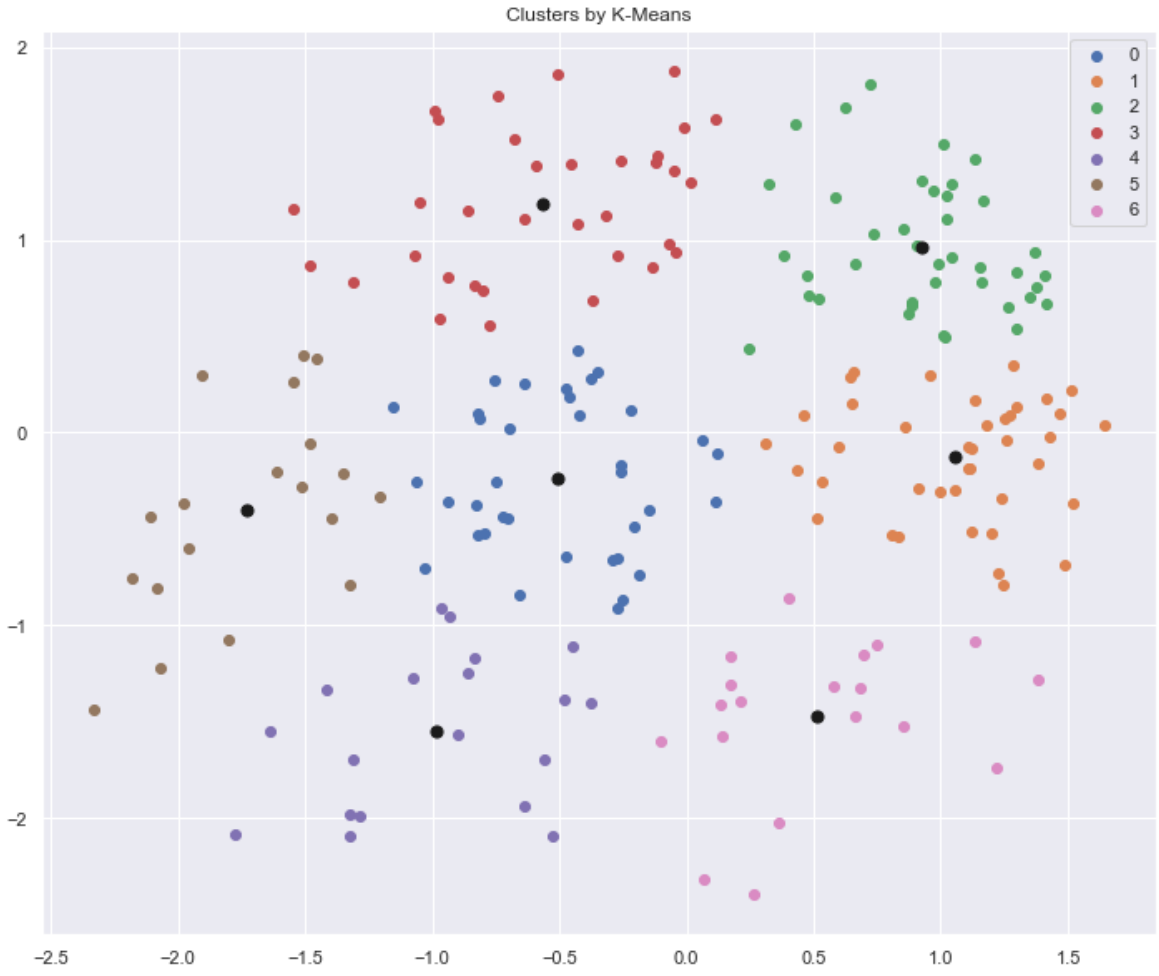

Figure 3.1: A scattered plot of K-means clustering on a data with two variables TS_1 and TS_2. The plots shows seven clusters with their corresponding centroids.

3.2 Dominance Analysis

Dominance Analysis (DA) is a technique to compute the relative importance of predictors in multiple regression. Budescu [8] was the first to propose the concept of DA. It approaches the problem of relative importance by examining the change in R^2 resulting from adding a predictor to all possible subset regression models [55]. Intuitively, DA determines the dominance of one predictor over others by comparing their incremental R^2 contribution across all subset models.

Predictors are compared in a pair-wise fashion across all subsets of the models to establish the hierarchy of dominance. Suppose we have 'p' predictors. We build $2^p - 1$ models (all possible subsets models) and compute the incremental R^2 contribution of each predictor to the subset model of all predictors.

We consider an example of three predictors x_1, x_2 and x_3. We would create a total of $2^3 - 1 = 7$ models. We denote any subset model that does include the the predictors x_i and x_j as $x_{(-i,j)}$. As example, if $i = 1$ and $j = 2$, then $x_{(-i,j)}$ would include any subset of $[x_3, x_4, \cdots, \mathbf{x}_p]$. In this case where we have three predictors, $x_{(-1,2)}$ would be the subset models $x_{(-1,2)} = [\{\emptyset\}, \{x_3\}]$. Similarly, if we take, x_1, the additional contributions of x_1 are computed as the increase in the proportion of variance accounted for when we add x_1 to each subset of the remaining predictors $\mathbf{x}_{(-1)} = [\{\emptyset\}, \{x_2\}, \{x_3\}, \{x_2, x_3\}]$, this applies to all other variables.

The coefficient of determination, also known as R^2, is a statistical measure in a regression model that determines the proportion of variance in the dependent variable that the independent variable can explain, and given mainly by:

$$R^2 = 1 - \frac{\text{Unexplained Variation}}{\text{Total Variation}}. \tag{3.2.1}$$

The variance proportion in Y that predictors account for in model X is $R^2_{Y \cdot X}$. For instance, $R^2_{Y \cdot x_1 x_2}$ is the proportion of variance of Y accounted for by the model consisting of x_1 and x_2. We measure the additional contribution of a given predictor by the proportional increase of variance that results from adding that predictor to the regression model.

Consider the three predictors as example again. We are interested in the individual contribution of each predictor in the variance of Y given the model $x_1 x_2 x_3$. We first create seven separate models of the subsets of $[x_1, x_2, x_3]$ except for the empty set $\emptyset$; thus models with the following parameters $[\{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}]$. Once the regression models are created, we compute the corresponding values $[R^2_{Y \cdot x_1}, R^2_{Y \cdot x_2}, R^2_{Y \cdot x_3}, R^2_{Y \cdot x_1 x_2}$,
$R^2_{Y \cdot x_1 x_3}, R^2_{Y \cdot x_2 x_3}, R^2_{Y \cdot x_1 x_2 x_3}]$, with these values, we can compute the additional contribution of each predictor to the variance proportion in Y for the entire model $x_1 x_2 x_3$.

For instance, let us compute the additional contribution of the predictor x_1 to the variance proportion of Y given $x_1 x_2 x_3$. Let kk be the subset model size. We assume that from empirical test analysis $[R^2_{Y \cdot x_1}, R^2_{Y \cdot x_2}, R^2_{Y \cdot x_3}, R^2_{Y \cdot x_1 x_2}, R^2_{Y \cdot x_1 x_3}, R^2_{Y \cdot x_2 x_3}, R^2_{Y \cdot x_1 x_2 x_3}]$= $[0.360, 0.090, 0.160, 0.450,$
$0.477, 0.228, 0.574]$. The additional contribution for x_1 would be the average of $R^2_{Y \cdot x_1 x(-1)} - R^2_{Y \cdot x(-1)}$,

where $\mathbf{x}_{(-1)} = [\{\emptyset\},\{x_2\},\{x_3\},\{x_2,x_3\}]$. Thus,

$$x_{1+}^{kk} = \begin{cases} R^2_{Y\cdot x_1} = \mathbf{0.360} \text{ for } kk = 0 \\ \frac{(R^2_{Y\cdot x_1x_2} - R^2_{Y\cdot x_2}) + (R^2_{Y\cdot x_1x_3} - R^2_{Y\cdot x_3})}{2} = \frac{(0.450-0.090)+(0.477-0.160)}{2} = \mathbf{0.339} \text{ for } kk = 1 \\ R^2_{Y\cdot x_1x_2x_3} - R^2_{Y\cdot x_2x_3} = 0.574 - 0.228 = \mathbf{0.346} \text{ for } kk = 2. \end{cases} \tag{3.2.2}$$

We then compute the additional contribution of x_1 to the variance proportion of Y given $x_1x_2x_3$ as :

$$R^2_{Y\cdot x_{1+}} = \frac{x_{1+}^{kk=0} + x_{1+}^{kk=1} + x_{1+}^{kk=2}}{3} = \frac{0.360 + 0.339 + 0.346}{3} = 0.348 \tag{3.2.3}$$

We iterate the above computations to find the additional contribution for the other predictors. Table 3.1 summarizes how to compute the variance contribution of x_2 and x_3 within the different subset models. Table 3.2 illustrates DA for a hypothetical example with three predictors. The first two columns of Table 3.2 show the subset models and their corresponding $R^2_{Y\cdot X}$ values. The rest of the table computes the additional contribution of each predictor. Eventually, the additional contribution of each predictor $[x_1,x_2,x_3]$ should sum to the variance in Y explained by the model of the entire predictors $x_1x_2x_3$. Thus, $R^2_{Y\cdot x_{1+}} + R^2_{Y\cdot x_{2+}} + R^2_{Y\cdot x_{3+}} = 0.348 + 0.089 + 0.137 = 0.574 = R^2_{Y\cdot x_1x_2x_3}$.

Subset model size(kk)	Average additional contributions of:	
	x_2	x_3
0	$R^2_{Y\cdot x_2}$	$R^2_{Y\cdot x_3}$
1	$[(R^2_{Y\cdot x_1x_2} - R^2_{Y\cdot x_1}) + (R^2_{Y\cdot x_2x_3} - R^2_{Y\cdot x_3})]/2$	$[(R^2_{Y\cdot x_1x_3} - R^2_{Y\cdot x_1}) + (R^2_{Y\cdot x_2x_3} - R^2_{Y\cdot x_2})]/2$
2	$(R^2_{Y\cdot x_1x_2x_3} - R^2_{Y\cdot x_1x_3})$	$(R^2_{Y\cdot x_1x_2x_3} - R^2_{Y\cdot x_1x_2})$

Table 3.1: Additional variance contribution computation for x_2 and x_3

3.3 Input Data Analysis & Parameter Identification

One assumption governing the architecture proposed by Jain and Rao [24] is that atmospheric weather conditions significantly impact load demands. We investigate how the average temperature, humidity, dew point and cumulative temperature-humidity index (CTHI) affect the load demands. We compare 2015 historical load demand for New York against same-year weather variables. We see that variation in load consumption is very sensitive to temperature and CTHI. Figure 3.2 supports the assertion of Jain and Rao [24], and it also shows a positive impact of increased average humidity on load consumption in New York.

3.3.1 Cumulative Temperature & Humidity Index (CTHI)

In their recent report, NYISO [40] applied the CTHI variable to model Load Forecast Uncertainty. Experimentally, we realized that including CTHI as input data to the daily average temperature and

Subset model(X)	$R^2_{Y\cdot X}$	Additional contributions of:		
		x_1	x_2	x_3
Null and $kk=0$ average	0	0.360	0.090	0.160
x_1	0.360		0.090	0.117
x_2	0.090	0.360		0.138
x_3	0.160	0.317	0.068	
$kk=1$ average		0.339	0.079	0.128
x_1x_2	0.450			0.124
x_1x_3	0.477		0.097	
x_2x_3	0.228	0.346		
$kk=2$ average		0.346	0.097	0.124
$x_1x_2x_3$	0.574			
overall average		0.348	0.089	0.137
% Relative importance		**60.63%**	**15.50%**	**23.87%**

Table 3.2: Relative importance of x_1, x_2 and x_3.The first two columns show the subset models and their corresponding $R^2_{Y\cdot X}$ values.

humidity improves the model's performance. The CTHI is a three-day weighted average of dry bulb and wet bulb temperature where:

1. We calculate the temperature-humidity Index (THI) as a weighted average of the dry bulb temperature (DB) and the wet-bulb temperature (WB). For any day d:

$$THI_{d_i} = 0.6 \cdot DB_{d_i} + 0.4 \cdot WB_{d_i} \tag{3.3.1}$$

   where $i = 0, 1, \cdots, 23$ indicate the hours of a day.

2. Compute $THI_{max} = \max(THI_{d_i})$.

3. We compute CTHI using a weighted average of three days (the day for which CTHI is being calculated and the two preceding days):

$$CTHI_d = 0.7 \cdot THI_{max_d} + 0.2 \cdot THI_{max_{d-1}} + 0.1 \cdot THI_{max_{d-2}} \tag{3.3.2}$$

The CTHI calculations consider both dry bulb and wet bulb temperatures, adjusting for the load-increasing impacts of heat and humidity. Also, it incorporates lag values, accounting for the load-increasing effects of heat buildup over multiple days [40].

The two main assumptions by Jain and Rao [24] that govern the model are the day-type criterion and the distinct selection criterion applied to input data for normal and anomalous days. Weekdays like Mondays to Friday, except holidays, are grouped as normal days. In contrast, weekends like Saturdays and Sundays, including holidays, are classified as anomalous days. This assumption claims that normal days would have load patterns distinct from anomalous days. For instance, if we wish to forecast Monday, which is not a holiday. Our input data selection would consist of historic weather variables for only Mondays that were not holidays.

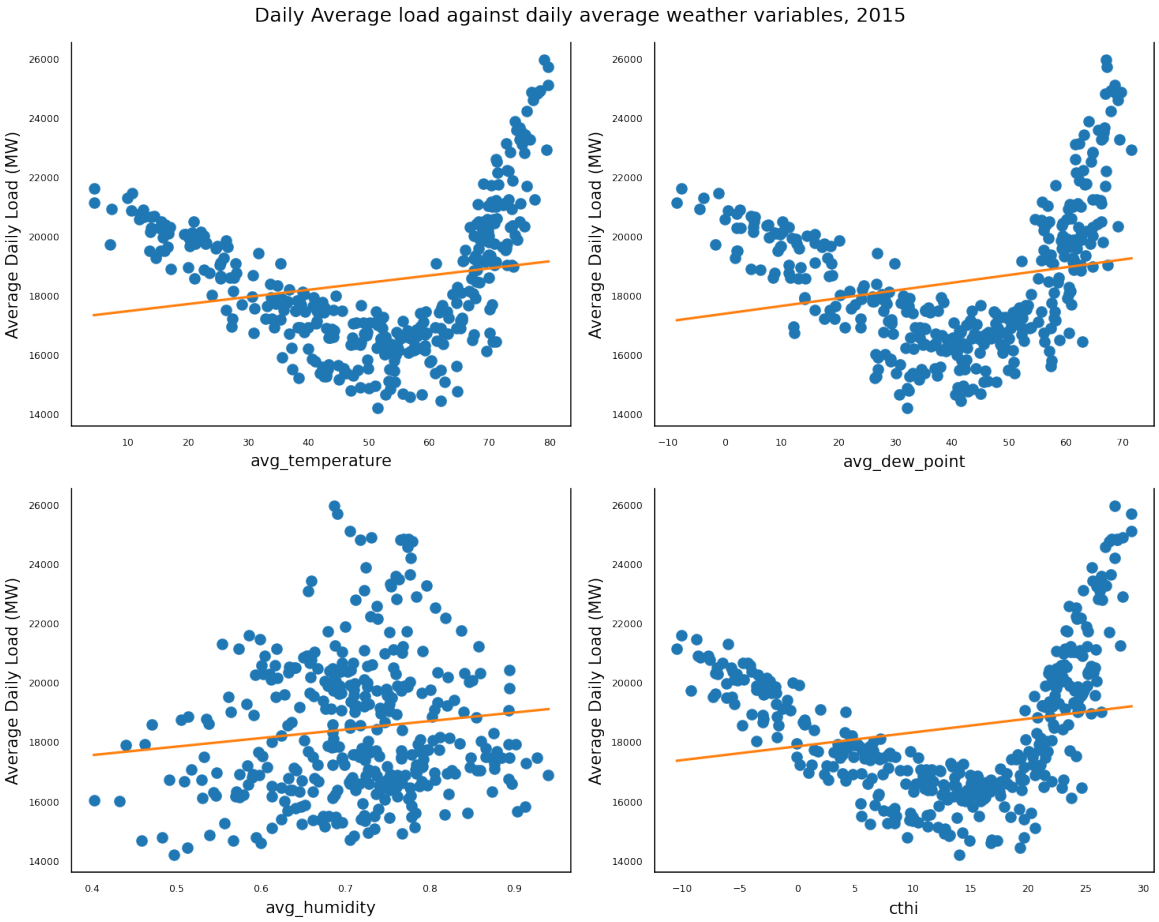

Figure 3.2: Daily Average load against daily average weather variables, 2015

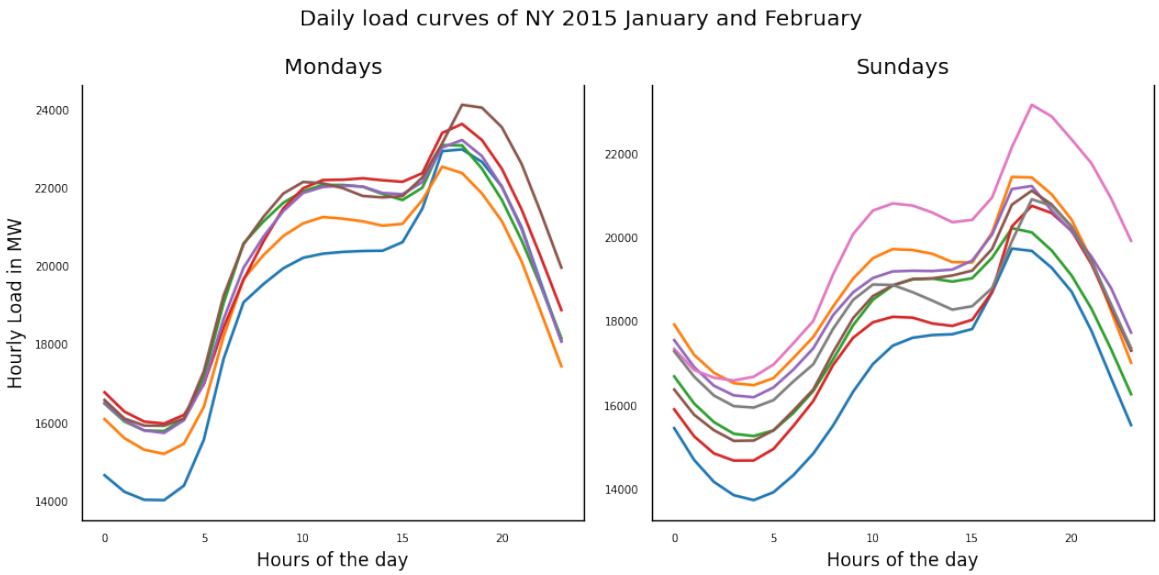

Figure 3.3: Daily load curves of NY 2015 January and February

In Figure 3.3, we have the daily load plots of Mondays and Sundays in two consecutive months, January and February. These plots help us to understand how load patterns behave for different day-type. From Figure 3.3, we see that all the Mondays that are not holidays have a similar trend distinct from the consecutive Sundays in January and February 2015. The similarity in calendar data is explained further using the relative standard deviation index (RSD):

$$\text{RSD} = \frac{\text{St. Dev. 24H load} \cdot 100}{\text{Mean 24H load}}. \tag{3.3.3}$$

From Tables 3.3 and 3.4, we observed that the RSD index for the successive days is more diverging than similar days, which are much closer.

date	day_of_week	%SD	date	day_of_week	%SD
2015-02-23	Monday	13.07	2015-01-24	Saturday	9.70
2015-02-24	Tuesday	8.87	2015-01-31	Saturday	8.10
2015-02-25	Wednesday	8.89	2015-02-07	Saturday	7.77
2015-02-26	Thursday	10.55	2015-02-14	Saturday	7.35
2015-02-27	Friday	8.72	2015-02-21	Saturday	6.82
2015-02-28	Saturday	6.36	2015-02-28	Saturday	6.36

Table 3.3: RSD metric for successive days and Saturdays

In forecasting anomalous days, we adopt a different approach. From Figure 3.4, we identified that the load curve of the anomalous day (Monday, 1 Jan. 2018) follows a different trend than that of the same day-types (Mondays) from the previous two months. However, as shown in Figure 3.5, the load curve of Monday, 1 Jan. 2018 follows a similar load pattern as those of other anomalous

date	day_of_week	%SD	date	day_of_week	%SD
2015-01-18	Sunday	9.19	2015-01-05	Monday	15.36
2015-01-25	Sunday	10.77	2015-01-12	Monday	12.65
2015-02-01	Sunday	8.20	2015-01-26	Monday	12.50
2015-02-08	Sunday	10.56	2015-02-02	Monday	12.80
2015-02-15	Sunday	10.61	2015-02-09	Monday	12.74
2015-02-22	Sunday	8.00	2015-02-23	Monday	13.07

Table 3.4: RSD metric for Sundays and Mondays

days and weekends from the previous two months. These have led to the conclusion made by Jain and Rao [24] that any anomalous day load curve is similar to the load curves of other anomalous days and weekends. The RSD value of the anomalous day is close to that of other anomalous days and weekends. However, it shows higher variance for its same day-type as depicted by Table 3.5. Therefore, a similar day selection criterion for normal and anomalous days would not generate a good forecast [24]. Table 3.6 summarizes the criterion for sampling anomalous and normal day-type input datasets.

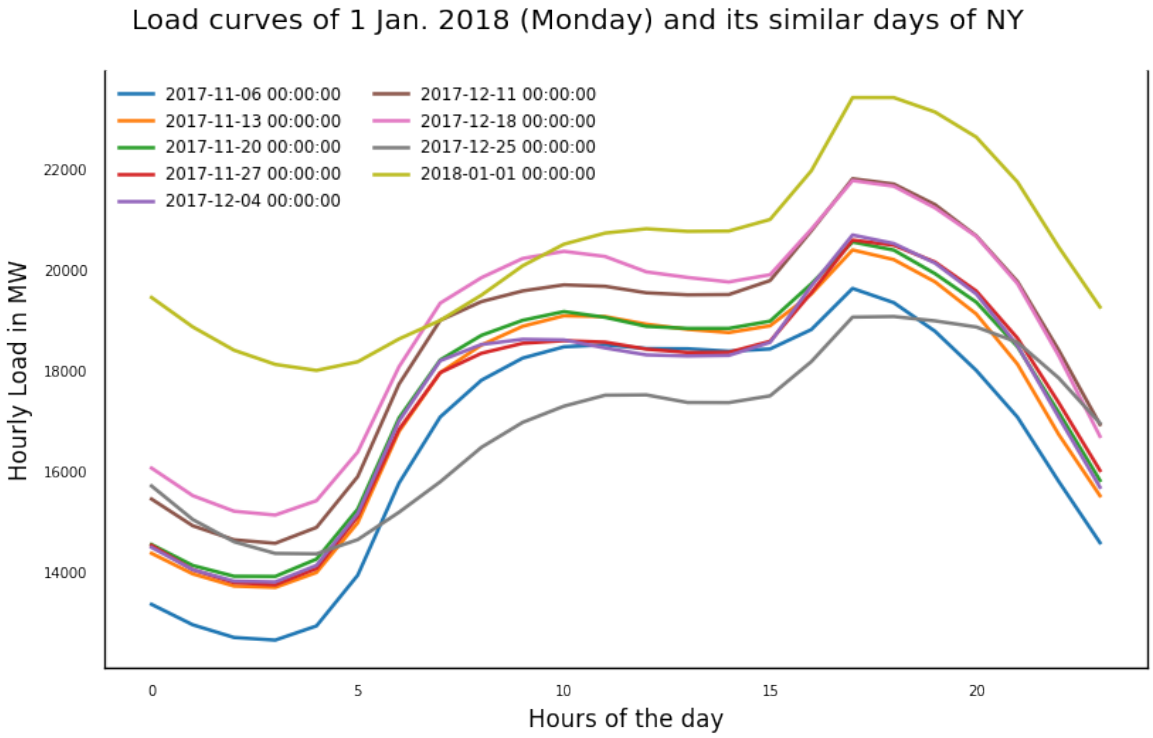

Figure 3.4: Load curves of 1 Jan. 2018 (Monday) and its similar days of NY

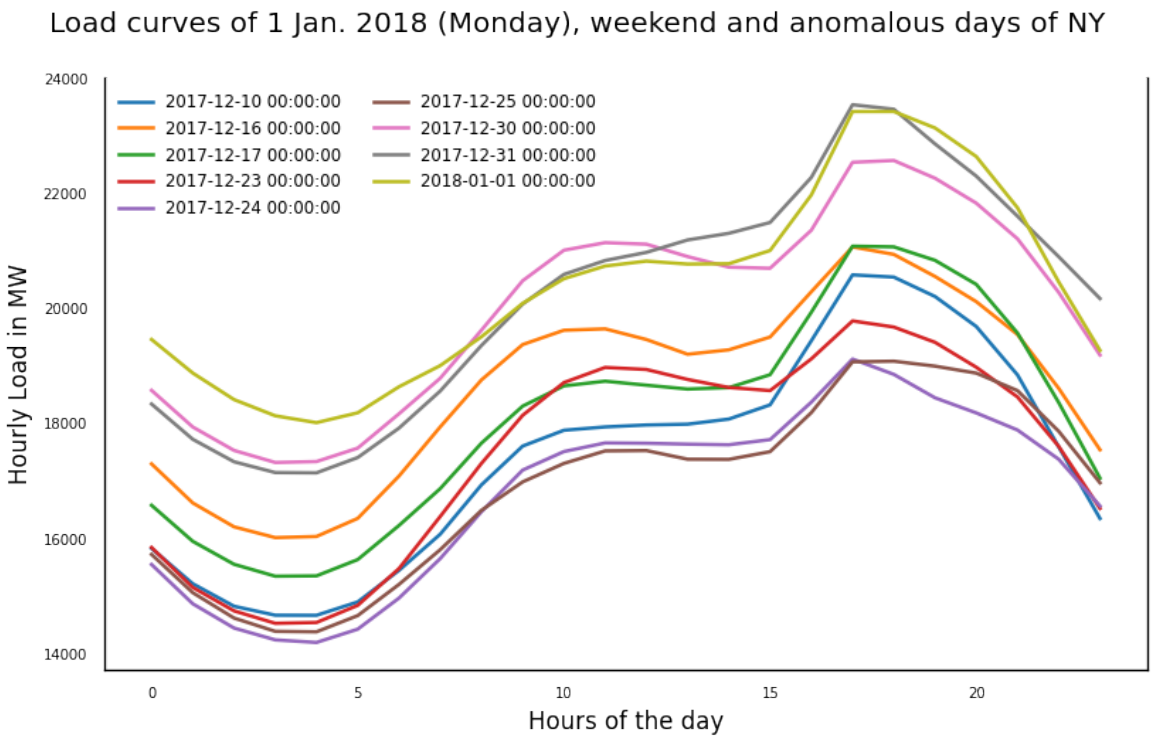

Figure 3.5: Load curves of 1 Jan. 2018 (Monday), weekend and anomalous days of NY

date	day_of_week	%SD	date	day_of_week	%SD
2017-12-17	Sunday	10.07	2017-11-20	Monday	12.31
2017-12-23	Saturday	10.23	2017-11-27	Monday	12.33
2017-12-24	Sunday	9.22	2017-12-04	Monday	12.37
2017-12-25	Monday	9.12	2017-12-11	Monday	12.33
2017-12-30	Saturday	8.47	2017-12-18	Monday	11.40
2017-12-31	Sunday	10.02	2017-12-25	Monday	9.12
2018-01-01	Monday	8.15	2018-01-01	Monday	8.15

Table 3.5: RSD metric of the anomalous day compared with similar days and other weekend-anomalous days

Day	Forecast day-type code	Input dataset day-type code
Sunday	000	000
Monday	001	001
Tuesday	010	010
Wednesday	011	011
Thursday	100	100
Friday	101	101
Saturday	110	110
Anomalous Day	111	111, 110 and 000

Table 3.6: Day-type codes are numeric values assigned to each day of the week. We use the codes to sample the historic input dataset for the day we wish to forecast.

3.4 Cluster-WED Architecture

The hybrid STLF architecture Cluster-WED proposed by Jain and Rao [24] comes in two phases (Figure 3.6). We sample the input data from the historical dataset and then apply the K-means clustering algorithm to divide the dataset into a pre-defined K number of clusters. The input dataset should include the day whose 24 hourly loads we wish to forecast. We cluster based on similarities in daily weather conditions. The weather parameters considered in this paper are the daily average temperature (T_{avg}), average humidity (H_{avg}) and CTHI; thus $X_t = \{T^t_{avg}, H^t_{avg}, CTHI^t\}$ for $t = 1, 2, \cdots, T_f$, where T_f is the position count of the day to be forecast. From empirical tests and results, we select the average temperature, average humidity, CTHI, historic hourly load data, and day-type as the dimension of our input dataset (Table 3.7). After splitting the input weather data set into K non-

Parameter	**Description**
L	Hourly load
$\mathbf{T}_{avg}$	Daily average temperature
$\mathbf{H}_{avg}$	Daily average humidity
CTHI	CTHI
DT	Day-type

Table 3.7: Input data set dimensions

overlapping clusters in the first phase, we identify the cluster that holds the day to be forecast. We label this cluster as C_f. Using C_f, we determine the weights for each weather parameter to compute the weighted Euclidean norm. We decide the optimal value for K by using the *silhouette analysis*, which is a measure of how close points in one cluster are close to points in a neighbouring cluster. It is a technique that validates the consistency of points in clusters, and this measure is within the range of [-1,1]. Jain and Rao [24] suggested a least squares regression approach given by

$$L^t_{avg} = L^{t-1}_{avg} + w_1 T^t_{avg} + w_2 H^t_{avg} + w_3 CTHI^t, \tag{3.4.1}$$

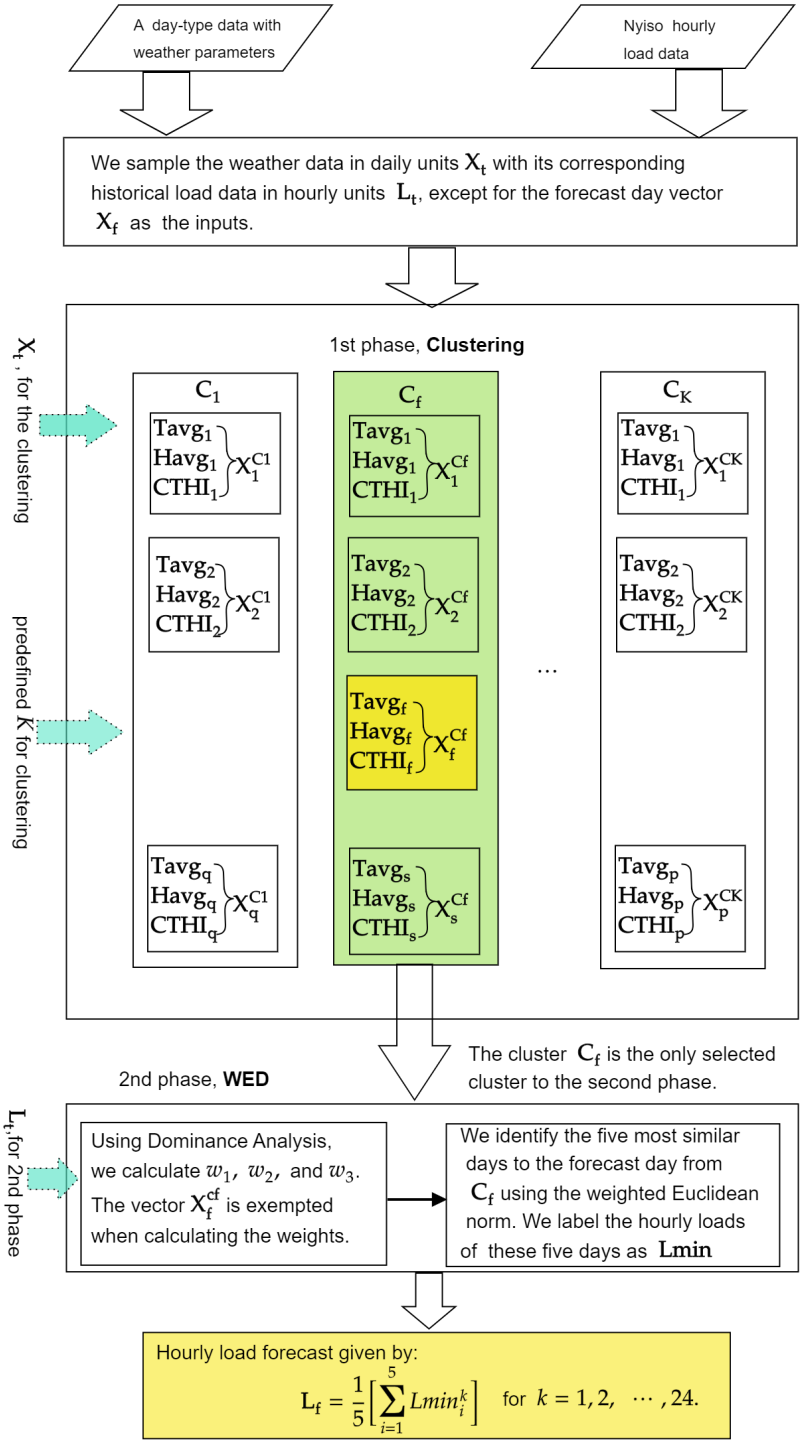

Figure 3.6: Detailed architecture of the `Cluster-WED` model.The day-type data is clustered into pre-defined K-samples. We select the five closest days to the forecast day from the forecast cluster C_f. the average of these days become the 24-hour forecast (Section 3.4).

where L^t_{avg} is the average load of the current day, and L^{t-1}_{avg} is the average load of the previous day. This linear regression approach potentially leads to a negative value for w_1, eventually, a negative sign under the root. Also, multicollinearity among the weather parameters interferes with the weight estimation and reduces the precision of the estimated coefficient. Hence, we use DA to find w_1, w_2 and w_3. The DA method yields positive weights since the additional contribution of individual predictors always sum to the coefficient of determination $R^2 \in (0,1)$ of the entire model. We exempt the forecast day vector from the weight calculation. We choose five days closest to the forecast day from C_f using the weighted Euclidean norm (Equation 3.4.2).

$$\text{ED} = \sqrt{w_1(\Delta T_{\text{avg}})^2 + w_2(\Delta H_{\text{avg}})^2 + w_3(\Delta CTHI)^2}, \tag{3.4.2}$$

where $\Delta T_{\text{avg}} = T_{\text{avg}_f} - T^c_{\text{avg}}$, $\Delta H_{\text{avg}} = H_{\text{avg}_f} - H^c_{\text{avg}}$ and $\Delta CTHI = CTHI_f - CTHI^c$. T_{avg_f}, H_{avg_f} and $CTHI_f$ are the average temperatures, humidity and CTHI for the forecast day; T^c_{avg}, H^c_{avg} and $CTHI^c$ are the average temperatures, humidity and CTHI of other similar days in C_f.

We sampled the 24 hourly loads of the five nearest days that gave the minimum ED (by Equation 3.4.2) and labelled these loads as *Lmin*. We forecast the 24 hourly loads of the forecast day, taking the average of the *Lmin* as in Equation 3.4.3.

$$L_f = \frac{1}{5}\left[\sum_{i=1}^{5} Lmin_i^k\right], \text{ for } k = 1,2,\cdots,24. \tag{3.4.3}$$

3.5 Chapter Summary

Jain and Rao [24] proposed a clustering and weighted Euclidean distance hybrid approach to an STLF. This chapter describes the proposed model. We first outlined the assumptions governing this model's development. In subsequent sections, we elaborate on the different mathematical ideas used to implement this model.

In this thesis, we found that adding the CTHI as an exogenous weather parameter to temperature and humidity improved performance than just using temperature and humidity. We also investigated the load patterns of different calendar dates, anomalous and normal days with the relative standard deviation. The authors of [24] suggested the least square method in assigning weights to the exogenous input parameters; this could not be generalized for all cases, as it potentially leads to a negative under the root in some instances. In this report, we use the Dominance Analysis proposed by Budescu [8]. The Dominance Analysis would always yield positive weights since each parameter's additional contribution should sum to the coefficient of determination $R^2 \in (0,1)$.

Section 3.1 introduces the K-means clustering algorithm, explains the fundamental mathematical equations, and gives its implementation pseudo-code (Algorithm 1). In Section 3.2, we give a detailed description of Dominance Analysis, which we use to calculate the weights of our predictors. The last two sections explain the different analyses made to choose the input dimensions and give a detailed description of the model's architecture. We discuss numerical results for this chapter in Chapter 5.

Chapter 4

Recurrent Neural Network

Power operators have used ARMA, ARMAX and SARIMA techniques for a long time to forecast load. In recent years, Artificial Intelligence (AI) techniques such as neural networks and deep learning are emerging in time series analysis. To this end, Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs) are being explored and show promise to give better forecasts compared to some other traditional and hybrid methods [38].

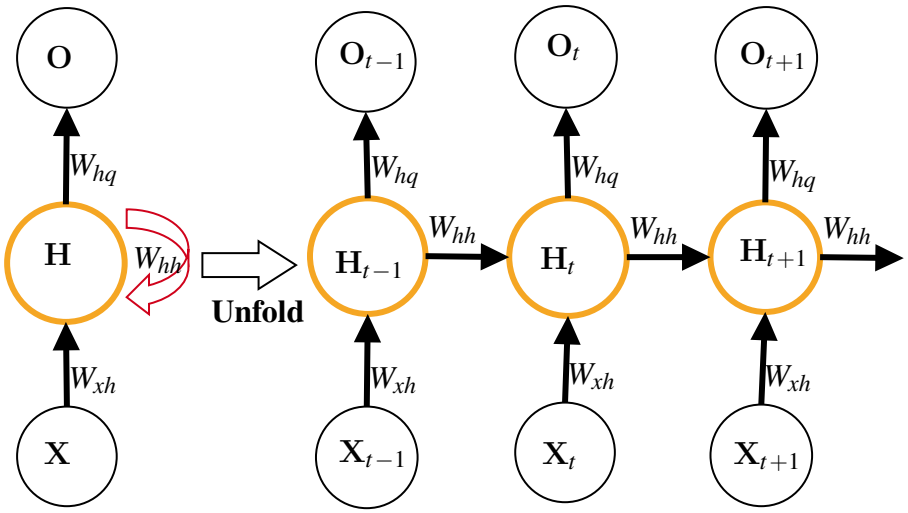

Figure 4.1: Unfolded RNN architecture

In 1990 Elman [13] became the first to propose the RNN model. RNN is more capable of using past information to predict the future and hence is widely used in natural language processing, voice processing and other time series applications. Figure 4.1 shows a standard RNN architecture unfolded by time.

Given n to be the mini-batch size and d the number of features in our dataset, we let the matrix $\mathbf{X}_t \in \mathbb{R}^{n\times d}$ represent the input at time t. The matrix $\mathbf{O}_t \in \mathbb{R}^{n\times q}$ represents the output at time t, $\mathbf{W}_{xh} \in \mathbb{R}^{d\times h}$ and $\mathbf{W}_{hq} \in \mathbb{R}^{h\times q}$ denote the hidden and output layer weights, respectfully, and $\mathbf{W}_{hh} \in \mathbb{R}^{h\times h}$ is

the same for different time steps. We express the output at time t as:

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h), \tag{4.0.1}$$

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q, \tag{4.0.2}$$

where $\phi(\cdot)$ is the activation function of the hidden layers. Typically a *tanh* or a *reLu* function; $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$ is the previous hidden state, $\mathbf{b}_h \in \mathbb{R}^{1 \times h}$ and $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ are the hidden and output layer biases, respectively.

Name	Function
Sigmoid	$f(x) = \dfrac{1}{1+\exp(-x)}$
Tanh	$f(x) = \tanh(x)$
ReLu	$f(x) = \max(0, x)$

Table 4.1: Activation functions

Training RNN or other neural networks modifies its parameters through gradient optimization algorithms like stochastic gradient descent (SGD) or Adam. An optimization algorithm finds weights that minimize the loss function when mapping inputs to outputs. It consists of two basic steps until the network converges. The first is the forward pass characterized by Equations 4.0.1 and 4.0.2. In the second step, known as the backward pass, the gradient of the loss function given the weights $\mathbf{W}_{xh}$, $\mathbf{W}_{hq}$ and $\mathbf{W}_{hh}$ are backpropagated through time to update the weight values.

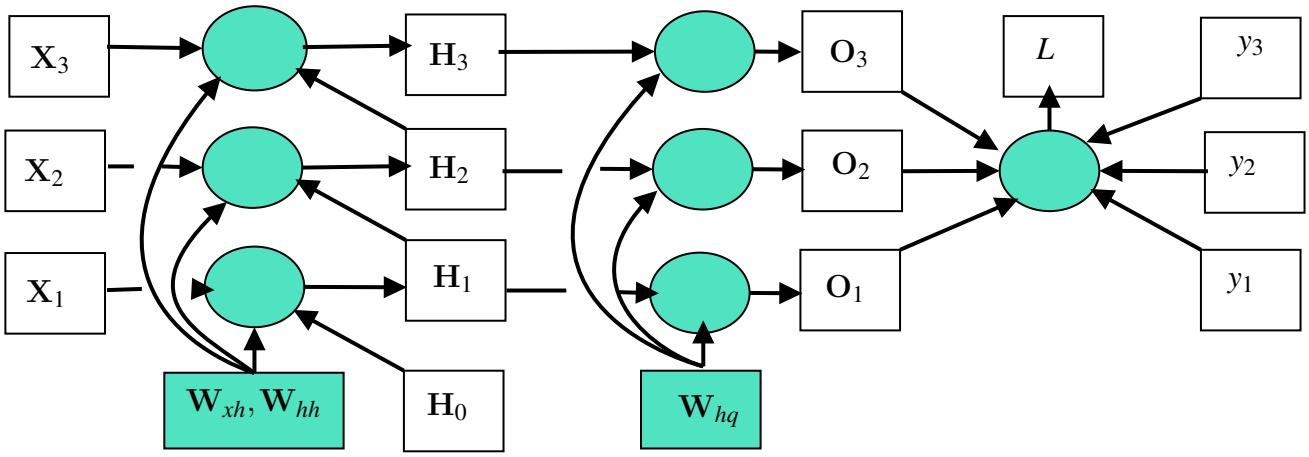

Figure 4.2: Graph showing dependencies for RNN model modified from [59], with three timesteps. Plain boxes represent variables and shaded parameters. Circles represent operators.

To be succinct in our approach, we consider an RNN without the bias parameter, whose activation function is identity such that $\phi(x) = x$. For the timestep t, we let the input be $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ and the label

y_t. We compute the hidden state $\mathbf{H}_t$ and output $\mathbf{O}_t$ as:

$$\mathbf{H}_t = \mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh}, \tag{4.0.3}$$

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq}. \tag{4.0.4}$$

We let $l(\mathbf{O}_t, y_t)$ represent the stochastic loss function at the time step t. The objective loss function over the entire observation T is given by:

$$L(\mathbf{W}) = \frac{1}{T} \sum_{t=1}^{T} l(\mathbf{O}_t, y_t), \tag{4.0.5}$$

where $\mathbf{W}$ is the generic representation of all the weights. Figure 4.2 gives a picturesque description of how variables and parameters interplay to compute RNN. For instance, computing $\mathbf{H}_2$ would depend on $\mathbf{W}_{xh}$ and $\mathbf{W}_{hh}$, the past hidden state $\mathbf{H}_1$ and the current input $\mathbf{X}_2$.

To backpropagate, we first have to find how much the total error changes relative to the output.

$$\frac{\partial L}{\partial \mathbf{O}_t} = \frac{\partial l(\mathbf{O}_t, y_t)}{T \cdot \partial \mathbf{O}_t} \in \mathbb{R}^{n \times q} \tag{4.0.6}$$

With Equation 4.0.6, we can compute the gradient of the objective loss function given all the weights. Based on Figure 4.2, we see that the objective function depends on $\mathbf{W}_{hq}$ through $\mathbf{O}_1, \cdots, \mathbf{O}_T$, using the chain rule

$$\frac{\partial L}{\partial \mathbf{W}_{hq}} = \sum_{t-1}^{T} \left(\frac{\partial L}{\partial \mathbf{O}_t} \frac{\partial \mathbf{O}_t}{\partial \mathbf{W}_{hq}} \right) = \sum_{t=1}^{T} \mathbf{H}_t^\top \frac{\partial L}{\partial \mathbf{O}_t} \in \mathbb{R}^{h \times q}. \tag{4.0.7}$$

As shown in Figure 4.2, we see that the objective function L depends on $\mathbf{H}_T$ via OT at the last timestep T. We, therefore, find the gradient $\partial L / \partial \mathbf{H}_T \in \mathbb{R}^{n \times h}$ using the chain rule:

$$\frac{\partial L}{\partial \mathbf{H}_T} = \frac{\partial L}{\partial \mathbf{O}_t} \frac{\partial \mathbf{O}_t}{\partial \mathbf{H}_T} = \frac{\partial L}{\partial \mathbf{O}_t} \mathbf{W}_{hq}^\top. \tag{4.0.8}$$

However, it becomes cumbersome when $t < T$, where the objective function L depends on $\mathbf{H}_t$ through $\mathbf{H}_{t+1}$. For instance, given that $T = 3$, then at $t = 1$, L would depend on $\mathbf{H}_1$ through $\mathbf{H}_2$ and $\mathbf{H}_3$; at $t = 2$, L would depend on $\mathbf{H}_2$ via $\mathbf{H}_3$. According to the chain rule, we recurrently compute the gradient $\partial L / \partial \mathbf{H}_t \in \mathbb{R}^{n \times h}$ at $t < T$ as:

$$\frac{\partial L}{\partial \mathbf{H}_t} = \frac{\partial L}{\partial \mathbf{H}_{t+1}} \frac{\partial \mathbf{H}_{t+1}}{\partial \mathbf{H}_t} + \frac{\partial L}{\partial \mathbf{O}_t} \frac{\partial \mathbf{O}_t}{\partial \mathbf{H}_t} = \frac{\partial L}{\partial \mathbf{H}_{t+1}} \mathbf{W}_{hh}^\top + \frac{\partial L}{\partial \mathbf{O}_t} \mathbf{W}_{hq}^\top. \tag{4.0.9}$$

Thus expanding the recurrent computation of any timestep $1 \leq t \leq T$ gives

$$\frac{\partial L}{\partial \mathbf{H}_t} = \sum_{i=t}^{T} \frac{\partial L}{\partial \mathbf{O}_{T+t-i}} \mathbf{W}_{hq}^\top (\mathbf{W}_{hh}^\top)^{T-i}. \tag{4.0.10}$$

Though RNN can learn the correlation in data through time, it suffers from the long-term dependency problem [59], and we see the evidence of this in Equation 4.0.10, which involves potentially vast powers of $\mathbf{W}_{hh}^\top$. This is numerically unstable, manifesting itself as the vanishing/exploding gradient.

Finally, according to Figure 4.2, L depends on $\mathbf{W}_{xh}$ and $\mathbf{W}_{hh}$ through interactions with the hidden states $\mathbf{H}_1, \cdots, \mathbf{H}_T$. To compute gradients $\partial L/\partial \mathbf{W}_{xh} \in \mathbb{R}^{d\times h}$ and $\partial L/\partial \mathbf{W}_{hh} \in \mathbb{R}^{h\times h}$, we once again apply the chain rule to give us:

$$\frac{\partial L}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{H}_t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \mathbf{X}_t^\top \frac{\partial L}{\partial \mathbf{H}_t}, \tag{4.0.11}$$

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial \mathbf{H}_t} \frac{\partial \mathbf{H}_t}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \mathbf{H}_{t-1}^\top \frac{\partial L}{\partial \mathbf{H}_t}. \tag{4.0.12}$$

Backpropagation, through time, computes and stores gradients in turn. Intermediate values like $\partial L/\partial \mathbf{H}_T$ are stored and used to compute $\partial L/\partial \mathbf{W}_{xh}$ and $\partial L/\partial \mathbf{W}_{hh}$ to avoid calculations duplicated.

4.1 Adam Optimization Algorithm

Kingma and Ba [30] first proposed the adaptive moment (Adam) estimation algorithm. This algorithm can adapt individual learning rates for different parameters using the gradients' first and second moments estimates. Adam combines advantages of the AdaGrad method [12], which works well in sparse gradient, and the RMSProp, which works well in online and non-stationary settings [30].

Let $l(\mathbf{W})$ be a differentiable loss function with respect to $\mathbf{W}$. To minimize the expected value $\mathbb{E}[l(\mathbf{W})]$ given $\mathbf{W}$; we define $l_1(\mathbf{W}), \cdots, l_T(\mathbf{W})$ as the values of the loss function at each timestep $1, \cdots, T$. Stochasticity of the loss function $l(\mathbf{W})$ may be due to the random sampling of the data points in mini-batches, or possibly the inherent noise in the function [30]. We let $g_t = \nabla_{\mathbf{W}} l_t(\mathbf{W})$ be the function's gradient.

Adam uses estimations of the first (mean) and second moments (uncentered variance) of the gradient to adapt the learning rates at each timestep of the neural network. The authors used exponential moving averages, computed on the gradient on the mini-samples at each timestep t:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{4.1.1}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{4.1.2}$$

where m and v are the first and second moments (moving averages) of the gradient g_t. We choose $[\beta_1, \beta_2] \in [0,1)$, such that the moving averages assign small weights to gradients too far in the past. Most literature sets it to default values of 0.9 and 0.999, respectively.

The vectors m and v are initialized by a zero vector, leading to moment estimates biased towards zero, especially at the initial timesteps and when the betas are closer to one. We show for m, and the same is analogous to v:

$$\begin{aligned}
m_0 &= 0 \\
m_1 &= \beta_1 m_0 + (1-\beta_1) g_1 = (1-\beta_1) g_1 \\
m_2 &= \beta_1 m_1 + (1-\beta_1) g_2 = \beta_1 (1-\beta_1) g_1 + (1-\beta_1) g_2 \\
m3 &= \beta_1 m_2 + (1-\beta_1) g_3 = \beta_1^2 (1-\beta_1) g_1 + \beta_1 (1-\beta_1) g_2 + (1-\beta_1) g_3 \\
&\cdots
\end{aligned}$$

The more we iterate m, the lesser the influence of the first term on the values for each m. They get to

be multiplied by increasing powers of $\beta_1 < 1$. We can therefore capture these patterns for m and v as:

$$m_t = (1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}g_i \tag{4.1.3}$$

$$v_t = (1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}g_i^2. \tag{4.1.4}$$

Since m and v are the estimates for the first and second moments of g_t, $\mathbb{E}[m_t] = \mathbb{E}[g_t]$ and $\mathbb{E}[v_t] = \mathbb{E}[g_t^2]$. Thus, the expected values of an unbiased estimator should equal the parameter we are estimating. However, this is not the case, and our estimators are biased towards zero.

To correct the bias, we first investigate how $\mathbb{E}[m_t]$, the exponential moving average's expected value correlates with the actual first moment $\mathbb{E}[g_t]$. The same is also analogous to $\mathbb{E}[v_t]$ and $\mathbb{E}[g_t^2]$.

$$\mathbb{E}[m_t] = \mathbb{E}\left[(1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}g_i\right] \tag{4.1.5}$$

$$= \mathbb{E}[g_t](1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i} + \zeta \tag{4.1.6}$$

$$= \mathbb{E}[g_t](1-\beta_1)\frac{\beta_1^t - 1}{\beta_1 - 1} + \zeta \tag{4.1.7}$$

$$= \mathbb{E}[g_t](1-\beta_1^t) + \zeta \tag{4.1.8}$$

We fix m_t in Equation 4.1.3 into Equation 4.1.5. In Equation 4.1.6, we approximate g_i with g_t and bring it out of the summation. Since we are approximating, we introduce the error term ζ. We transform from Equation 4.1.6 to 4.1.8 using the finite sum of geometric series. We have an extra term of $(1-\beta_1^t)$ that is caused by initializing m_0 to be zero. In the case of $v_0 = 0$, we would have $(1-\beta_2^t)$. To correct the initialization bias, the authors divided m by $(1-\beta_1^t)$ and v by $(1-\beta_2^t)$ (Algorithm 2).

4.2 Seq2Seq Bidirectional LSTM Architecture

In their attempt to solve the vanishing/exploding gradient of RNN, Hochreiter and Schmidhuber [20] proposed the long short-term memory (LSTM) architecture. A standard RNN cell is composed of only a superficial tanh layer. However, a typical LSTM cell contains three neural network (NN) layers with a sigmoid function called the Forget ($\mathbf{F}$), Input ($\mathbf{I}$) and the Output ($\mathbf{O}$) gate. There is also a fourth layer with a tanh function called the Candidate ($\overline{\mathbf{C}}$) layer (Figure 4.3). These gates decide whether or not to store information based on the weights of importance assigned to the information [60].

Suppose the batch size is n, h is the number of hidden units, and the number of features for our dataset is d. At any given time t, the input of the LSTM cell is $\mathbf{X}_t \in \mathbb{R}^{n\times d}$, the previous hidden state $\mathbf{H}_{t-1} \in \mathbb{R}^{n\times h}$ and the previous memory state $\mathbf{C}_{t-1} \in \mathbb{R}^{n\times h}$. The output at time t is the current hidden state $\mathbf{H}_t$ and the current memory state $\mathbf{C}_t$. We initialize the values of the Forget, Input and Output

Algorithm 2 Adam, as proposed by Kingma and Ba [30]; g_t^2 is the element-wise product of g_t, thus $g_t \odot gt$. A good default setting for machine learning are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. All operations on vectors are element-wise, with β_1^t and β_2^t denoting β_1 and β_2 to the power of t.

Require: α ▷ Stepsize
Require: $\beta_1, \beta_2 \in [0,1)$ ▷ Exponential decay rates for the moment estimates
Require: $l(\mathbf{W})$ ▷ Stochastic loss function with parameter $\mathbf{W}$
Require: θ_0 ▷ Initial parameter vector
  $m_0 \leftarrow 0$ ▷ Initialize 1^{st} moment vector
  $v_0 \leftarrow 0$ ▷ Initialize 2^{nd} moment vector
  $t \leftarrow 0$ ▷ Initialize timestep
  while $\mathbf{W}_t$ do not converge **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_{\mathbf{W}} l_t(\mathbf{W}_{t-1})$ ▷ Get gradient w.r.t. loss at timestep t
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$ ▷ update biased first moment estimate
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$ ▷ update biased second moment estimate
    $\hat{m}_t \leftarrow m_t / (1-\beta_1^t)$ ▷ Compute bias-corrected first moment estimate
    $\hat{v}_t \leftarrow v_t / (1-\beta_2^t)$ ▷ Compute bias-corrected second moment estimate
    $\mathbf{W}_t \leftarrow \mathbf{W}_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$ ▷ Update parameters
  end while
  return $\mathbf{W}_t$ ▷ Resulting parameters

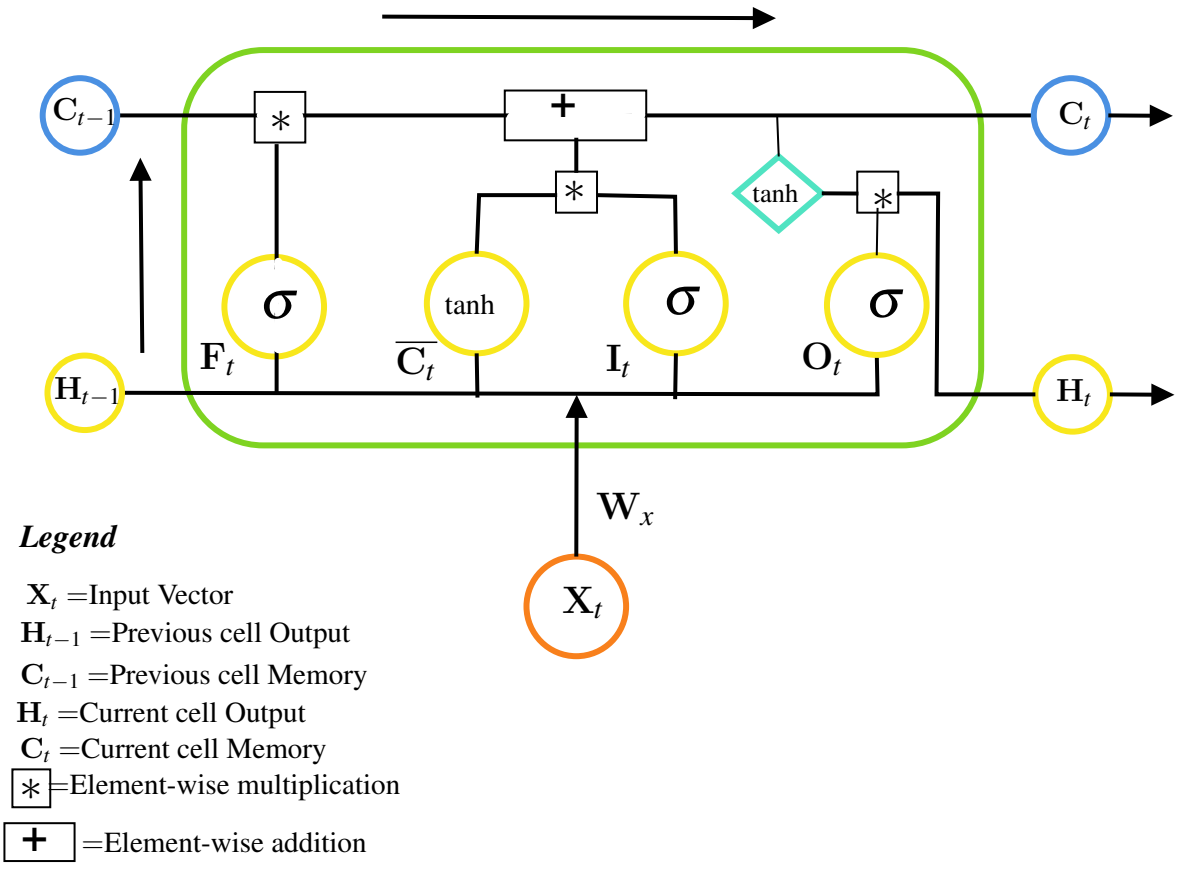

Figure 4.3: A single cell LSTM architecture

gates; and the Candidate layer by the following equations:

$$\mathbf{F}_t = \sigma(\mathbf{X}_t * \mathbf{W}_{xf} + \mathbf{H}_{t-1} * \mathbf{W}_{hf} + \mathbf{b}_f), \tag{4.2.1}$$

$$\overline{\mathbf{C}}_t = \tanh(\mathbf{X}_t * \mathbf{W}_{xc} + \mathbf{H}_{t-1} * \mathbf{W}_{hc} + \mathbf{b}_c), \tag{4.2.2}$$

$$\mathbf{I}_t = \sigma(\mathbf{X}_t * \mathbf{W}_{xi} + \mathbf{H}_{t-1} * \mathbf{W}_{hi} + \mathbf{b}_i), \tag{4.2.3}$$

$$\mathbf{O}_t = \sigma(\mathbf{X}_t * \mathbf{W}_{xo} + \mathbf{H}_{t-1} * \mathbf{W}_{ho} + \mathbf{b}_o), \tag{4.2.4}$$

where $\mathbf{W}_{xf}$, $\mathbf{W}_{xc}$, $\mathbf{W}_{xi}$, $\mathbf{W}_{xo} \in \mathbb{R}^{d\times h}$ and $\mathbf{W}_{hf}$, $\mathbf{W}_{hc}$, $\mathbf{W}_{hi}$, $\mathbf{W}_{ho} \in \mathbb{R}^{h\times h}$ are the weights and $\mathbf{b}_f$, $\mathbf{b}_c$, $\mathbf{b}_i$, $\mathbf{b}_o \in \mathbb{R}^{1\times h}$ are the bias.

Thus, the three gates ($\mathbf{F_t}, \mathbf{I_t}, \mathbf{O_t}$) and the candidate ($\overline{\mathbf{C_t}}$) layer first take the pair-wise multiplication of the input vector ($\mathbf{X_t}$) and their corresponding weights $\mathbf{W}_x$, concatenate (pair-wise addition) them with products of $\mathbf{H_{t-1}}$ and their corresponding $\mathbf{W}_h$ before applying the activation functions (Figure 4.3). The cell takes the previous memory state $\mathbf{C_{t-1}}$ performs a pair-wise multiplication with the forget gate $\mathbf{F_t}$ (i.e. $\mathbf{C}_t = \mathbf{C}_{t-1} \cdot \mathbf{F}_t$). The cell completely forgets the previous state memory if the forget gate value is 0 and passes it to the next timestep if the forget state value is 1.

Given the current memory state $\mathbf{C}_t$, we update its value by $\mathbf{C}_t = \mathbf{C}_t + (\mathbf{I}_t \cdot \overline{\mathbf{C}_t})$. Finally, we compute the output $\mathbf{H_t}$ based on the updated cell state (memory) $\mathbf{C_t}$ filtered through a tanh function. We then perform an element-wise multiplication with the output state $\mathbf{O}_t$ to give us $\mathbf{H}_t = \mathbf{O}_t \cdot \tanh(\mathbf{C}_t)$. Thus,

$$\mathbf{C}_t = \mathbf{F}_t \cdot \mathbf{C}_{t-1} + \mathbf{I}_t \cdot \overline{\mathbf{C}_t} \tag{4.2.5}$$

$$\mathbf{H}_t = \mathbf{O}_t \cdot \tanh(\mathbf{C}_t). \tag{4.2.6}$$

The sequence-to-sequence architecture, also known as the encoder-decoder architecture, was first developed by Sutskever et al. [51] to solve a speech translation problem from English to French. The encoder-decoder model comprises two main parts, the encoder and the decoder. The encoder takes the input sequence and summarizes them into a fixed-length vector called encoder-vector. LSTM cells are effective and can learn long-term dependencies because of their memory cells. They are, however, unidirectional and can only learn past context. To use both forward and backward dependencies in our input sequence, we implement the bidirectional LSTM in our encoder.

We use two LSTM layers that process the input sequence in opposing directions. At each timestep, the hidden and memory cell state are concatenated and passed to the next to compute the next hidden and cell state (i.e $\mathbf{H}_t = [\overrightarrow{\mathbf{H}}_t, \overleftarrow{\mathbf{H}}_t]$ and $\mathbf{C}_t = [\overrightarrow{\mathbf{C}}_t, \overleftarrow{\mathbf{C}}_t]$). We discard all encoder output, preserving only the last hidden ($\mathbf{H}_T = [\overrightarrow{\mathbf{H}}_T, \overleftarrow{\mathbf{H}}_T]$)) and memory cell ($\mathbf{C}_T = [\overrightarrow{\mathbf{C}}_T, \overleftarrow{\mathbf{C}}_T]$) states as the encoder-vector.

The encoder-vector encapsulates all information of the input vector, passing them to the decoder for predictions. Each LSTM cell unit in the decoder predicts output at time t. The decoder takes the encoder-vector as its initial state to make its first predictions. We pass the output of the first timestep and the previous hidden states to the second timestep for prediction. We iterate until we compute all predictions. Figure 4.4 is a picturesque representation of this sequence-to-sequence model, which we would name the `BiLSTM` model. We train the `BiLSTM` architecture with the following layer specifications in Table 4.2.

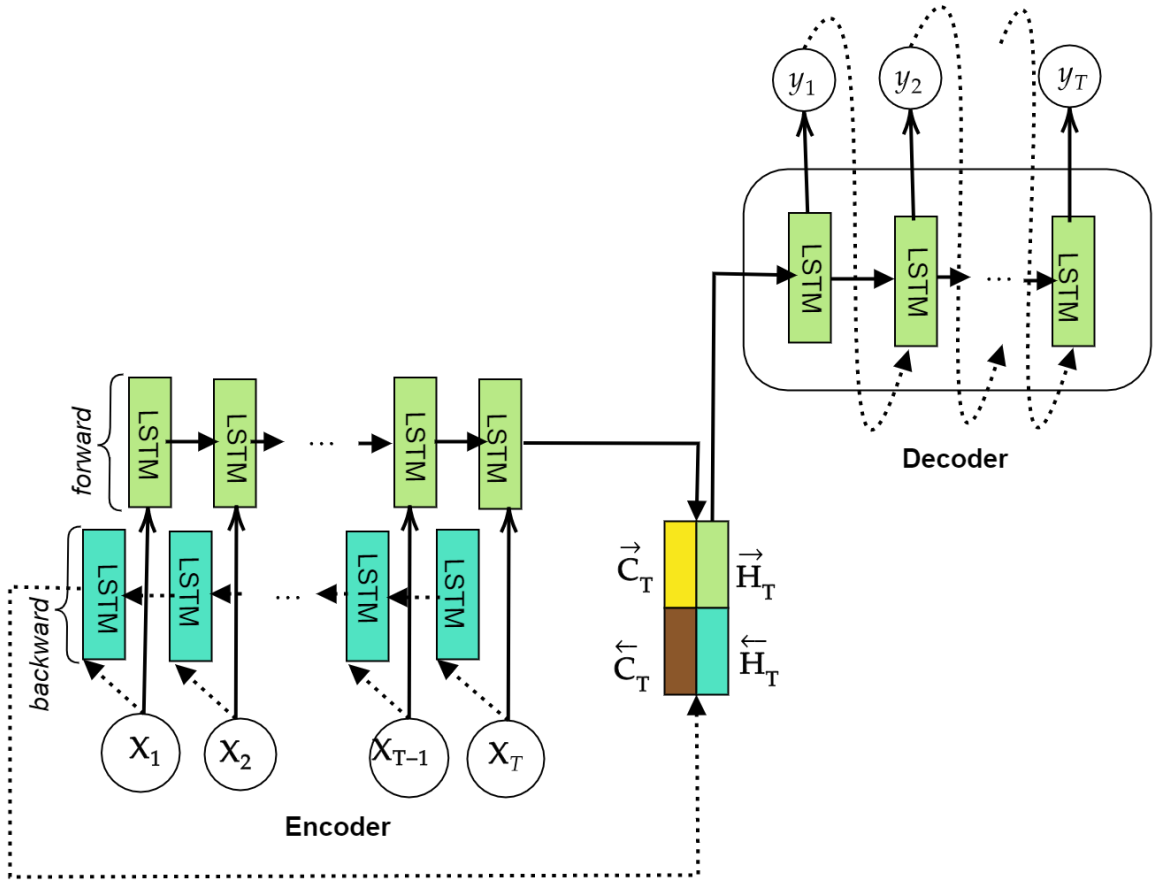

Figure 4.4: LSTM encoder-decoder (`BiLSTM`) architecture. The encoder is bidirectional and the decoder is unidirectional.

Hyperparameter	Value
Input time series	2 (load & temperature)
Output time series	1 (load)
RNN cell variant	LSTM
Embedding dim	200
Encoder	Bidirectional
Encoder depth	2 (a layer in each direction)
Decoder	Unidirectional
Decoder depth	1
Attention	None
Optimizer	Adam
Epochs	100
Initial learning rate	0.0005
batch size	16

Table 4.2: `BiLSTM` layer specifications and hyperparameters. After some repeated trials with different values, this set of hyperparameter values gave the best results.

4.3 LSTM with Luong Attention

The encoder-vector in the seq2seq architecture summarizes all information from the input, which creates a bottleneck in the architecture, making it difficult to cope with long input sequences. Bahdanau et al. [5] were the first to present an extensive model that learns to align and predict jointly. This model makes predictions based on the context vector and previous predictions. Later, Luong et al. [34] extended the work of Bahdanau et al. [5] to propose a more efficient global attention. Global attention, just like Bahdanau et al. [5], considers all the hidden states of the encoder when computing the context vector. However, it differs in how the alignment score is computed and the position of attention to the decoder. Luong et al. [34] proposed three ways to calculate the alignment score function: the dot, general and concatenate methods. In this paper, we implemented the dot scoring function, which is given by:

$$\text{score}_t = \mathbf{H}_{dec}^{t-1^\top} \mathbf{H}_{enc}, \tag{4.3.1}$$

where $\mathbf{H}_{enc}$ is the stack of all hidden sequences from the encoder and $\mathbf{H}_{dec}^{t-1}$ is the hidden states of the previous decoder.

We first generate a stacked hidden sequence, passing all our input into the encoder model (Figure 4.5). Instead of using the hidden state at the last time step T, we carry forward all the hidden states of the encoder $\mathbf{H}_{enc}$ to compute the alignment score. The alignment score is the attention mechanism's bedrock, which quantifies the amount of weight the decoder will place on each encoder output when producing the next output. We then apply the softmax function on the score to give us the attention weight:

$$a_t = \frac{\exp(\text{score}_t)}{\sum \exp(\text{score}_t)} \tag{4.3.2}$$

Once we have the attention weights, we compute the context vector by performing a pair-wise multiplication of attention weights a_t and the encoder output $\mathbf{H}_{enc}$. The softmax function's essence is to assign probabilities to the encoder outputs. Suppose the weight of any member of the encoder is closer to one, then its influence on the subsequent decoder output is amplified and vice versa. We concatenate the context vector with the previous decoder's hidden state $\mathbf{H}_{dec}^{t-1}$; the combined vector is then passed through the dense layer to predict an output. We name the model `AttnLSTM`; Table 4.3 gives all layer specifications and hyperparameters.

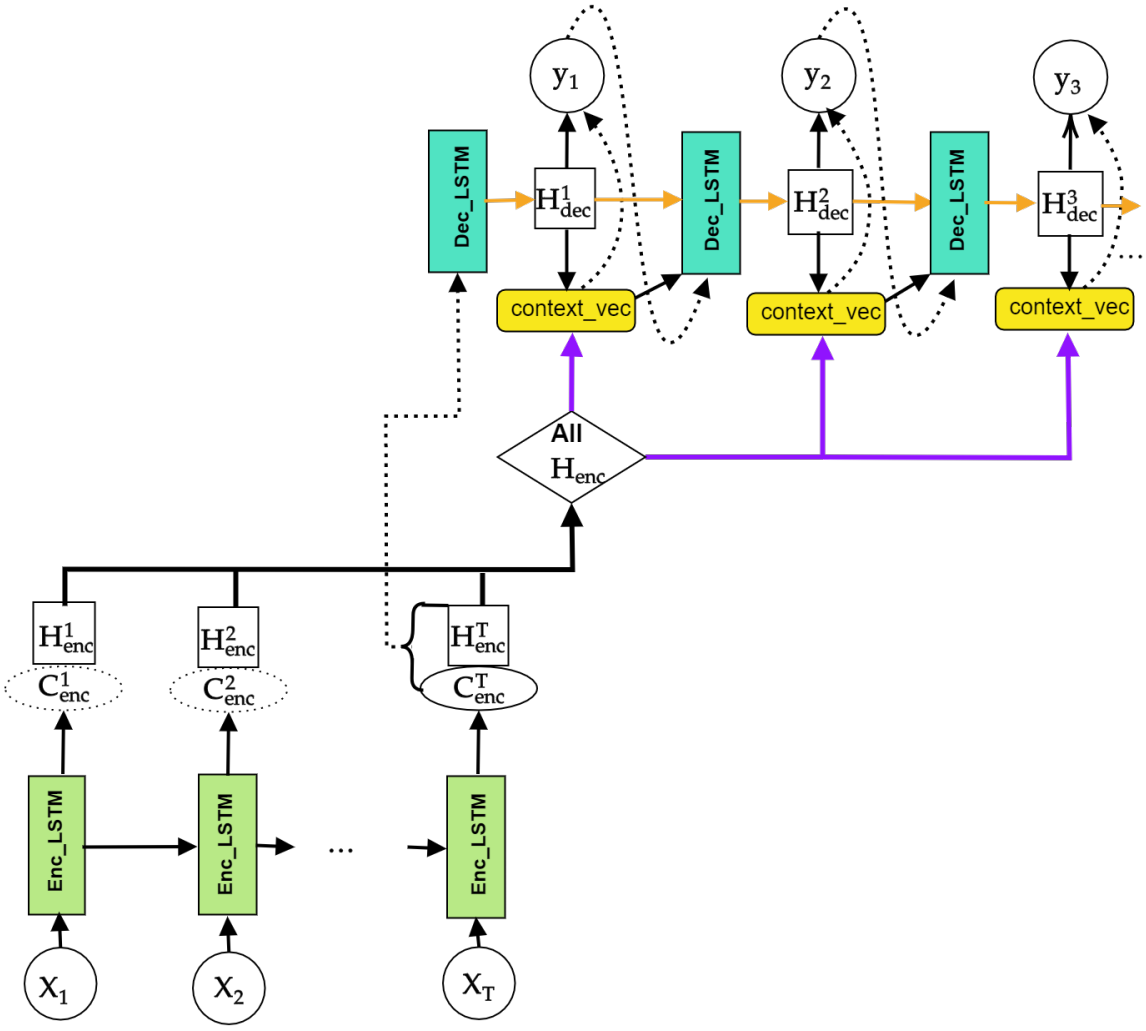

Figure 4.5: LSTM encoder-decoder with Loung attention (`AttnLSTM`) model. The encoder and the decoder are unidirectional.

4.4 Chapter Summary

Empirical testing of the `Cluster-WED` model suggests it sometimes becomes inconsistent in its predictions, leading to overfitting and high variance of the predicted and the observed data. Even Sajjad

Hyperparameter	Value
Input time series	2 (load & temperature)
Output time series	1 (load)
RNN cell variant	LSTM
Embedding dim	200
Encoder	Unidirectional
Encoder depth	1
Decoder	Unidirectional
Decoder depth	1
Attention	Luong (Global attention)
Optimizer	Adam
Epochs	100
Initial learning rate	0.0005
batch size	16

Table 4.3: `AttnLSTM` layer specifications and hyperparameters. After several repeated trials with different values, this set of hyperparameter values gave the best model results.

et al. [47] asserts these empirical facts. This chapter is a quest to find a model that would be more consistent. We open the chapter with the introduction of recurrent neural networks. We expound on its architecture and the different equations that govern its operation. We briefly touch on the two parts of its operation: the forward pass and the backpropagation through time.

Section 4.1 introduces the adaptive moment optimization algorithm (Adam). Adam combines AdaGrad and RMSProp optimization methods; Algorithm 2 gives pseudo-codes for its computation. We also introduce the long short-term memory (LSTM) and outline the different mathematical equations implemented in the section. We introduce the sequence-to-sequence architecture and finally give a detailed description of the `BiLSTM` architecture. Section 4.3 briefly introduces the attention mechanism and describes the `AttnLSTM` model. We give numerical results of this chapter in the next chapter.

Chapter 5

Numerical Implementation & Results

We use historic NYISO load demand and weather data from 2012 to 2020. We refine the input data before training on our different models since even the best model can yield poor results with the wrong data. We employ data preprocessing methods to normalize our data, fill empty spaces, and combine the load and weather data in hourly granularity according to the corresponding timestamp.

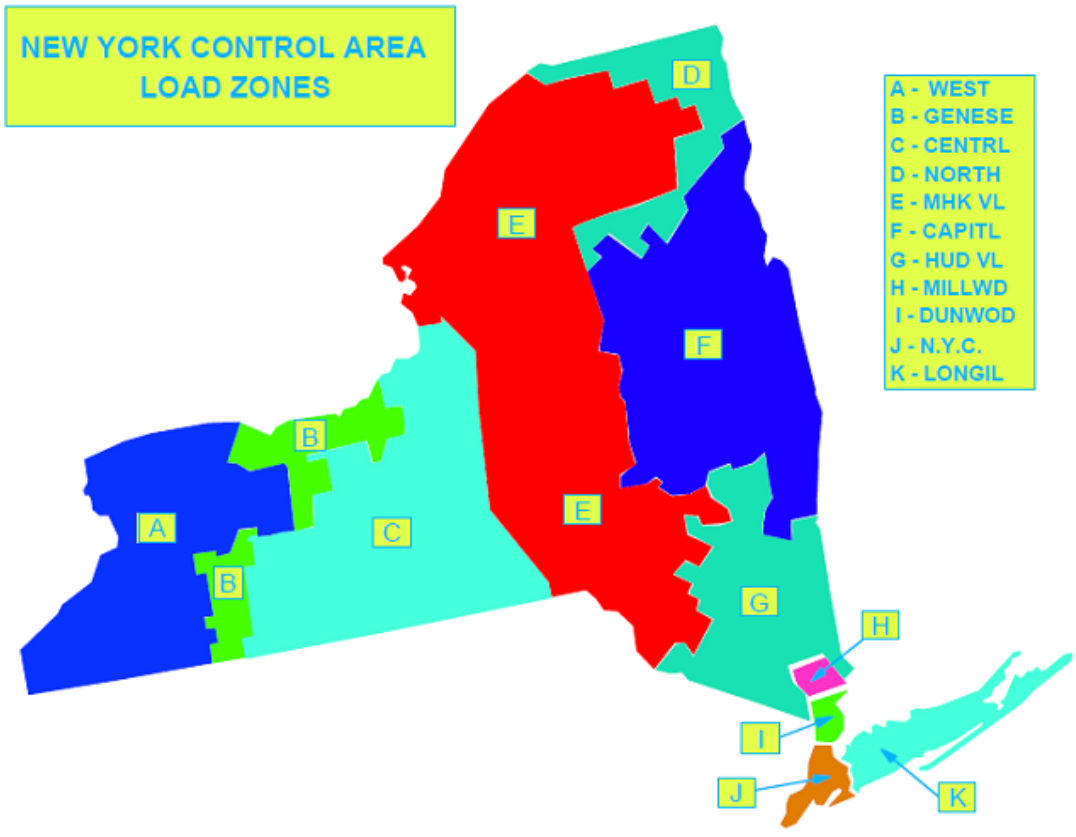

Figure 5.1: New York Control Area Load Zones, NYISO website

Since we are dealing with input parameters of varying scales, it is necessary to rescale the input

parameters to force them on the same scale while preserving the differences in their ranges of values. We use the `Min-Max` normalization technique that helps us achieve this without assuming the distribution of each parameter. We compute the `Min-Max` scale for each input parameter by:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{5.0.1}$$

where x is the input vector, x_{min} is the minimum and x_{max} the maximum values of the input vector, and x_{norm} is the rescaled input parameter.

We benchmark our models with the NYISO forecasting model based on ANN and statistical regression techniques to forecast hourly load [24]. The New York Control Area comprises eleven load zones (Figure 5.1). The NYISO model forecasts each zone independently using economic, load data, and historical weather parameters like temperature, dew points, wind speed, and cloud cover [41]. The zonal forecast models use weather information gathered from seventeen weather stations across New York. The information from the weather stations is appropriately aggregated to reflect each zone [41]. Consequently, information from the seventeen weather stations is aggregated into eleven zone weather sets and one-state weather (Table 5.1).

Zone	Stations	Station Weights	Zone	Stations	Station Weights
1 - WEST	Buffalo	91%	**2 - GENESE**	Elmira	5%
	Elmira	5%		Rochester	85%
	Syracuse	4%		Syracuse	10%
3 - CENTRL	Binghamton	23%	**4 - HUD VL**	Newburgh	68%
	Elmira	14%		Poughkeepsie	27%
	Syracuse	55%		White Plains	4%
	Watertown	9%		Albany	2%
5 - MHK VL	Binghamton	20%	**6 - CAPITL**	Albany	76%
	Massena	17%		Binghamton	3%
	Monticello	13%		Plattsburgh	5%
	Utica	35%		Poughkeepsie	6%
	Watertown	15%		Utica	10%
7 - N.Y.C.	JFK	21%	**8 - NORTH**	Plattsburgh	100%
	LGA	79%	**9 - LONGIL**	Islip	100%
10 - MILLWD	White Plains	100%	**11-DUNWOD**	White Plains	100%

Table 5.1: Weather station weights imputed to each zone [41] and applied to the NYISO model.

However, Plant-E Corp summarizes historical weather data into ten `city_ids`. To find the weather condition for the whole of New York State, we implement the linear combination approach proposed by Sobhani et al. [50]. We first collect weather parameters like temperature, humidity and temperature-humidity index (THI) according to `city_ids`. We use these parameters as our dependent variables and the NYISO load as the dependent variable. We perform a regression analysis for

each `city_id`. At each instance, we compute the overall mean absolute error and allocate decreasing linear weights to each computed error ranked in the increasing order, and then normalized to sum to one (Table 5.2).

For example, we have four `city_ids` arranged according to their mean absolute error in increasing order. Accordingly, the linear weights assigned to these `city_ids` are four, three, two, and one. We normalize the weights to keep the combined temperature and humidity profile in the same range as the individual ones. We let lin_u_i be the linear weights, u_i is the normalized weight, and n be the total number of `city_ids`. The normalized weight u_i is given by:

$$u_i = \frac{lin\_u_i}{\sum_{i=1}^{n} lin\_u_i} \tag{5.0.2}$$

where,

$$lin\_u_i = \begin{cases} n, & i = 1 \\ n-1, & i = 2 \\ \vdots & \vdots \\ 1, & i = n \end{cases} \tag{5.0.3}$$

City-ID	City-name	Zones	City-weights
12	Albany	Capital, Mohawk Vl.	0.109
13	Syracuse	Central	0.073
14	Buffalo	West	0.036
15	Rochester	Genese	0.018
16	NYC	NYC	0.145
17	Huntington	Long Is.	0.127
18	Plattsburgh	North	0.091
19	Genese	Genese	0.055
20	White Plains	Dunwod, Millwod	0.164
21	Newburgh	Hudson Vl.	0.182

Table 5.2: City-weather weights imputed to each zone and applied to the `Cluster-WED`, `BiLSTM` and `AttnLSTM` models. The weights are computed using the linear combination approach proposed by [50].

To implement the `BiLSTM` and `AttnLSTM` models described in Chapter 4, we first have to transform the data into a supervised learning problem. We predict the following 24 hours of load demands, given the past 48 hours of load ($\mathbf{L}$) and temperature ($\mathbf{T}$). Plant-E Corp receives actual load demands up to the fifth hour of the current day. That implies that we would be running short of 19 hours of the current day's data to predict the day-ahead load demand. To remedy this, we concatenate the 5 hours of the current day with 19 hours of the NYISO load forecast for that same day. We create a new column dubbed adjusted-load (Figure 5.2). This new column now becomes our new time series $\mathbf{L}$ to train the `BiLSTM` and `AttnLSTM` models. This adjusted load also replaces L (Table 3.7) in training

the Cluster-WED model, whose description is outlined in Chapter 3. We do not alter the temperature time series since we have all 48 hours of temperature data to forecast the next 24 hours of load demand.

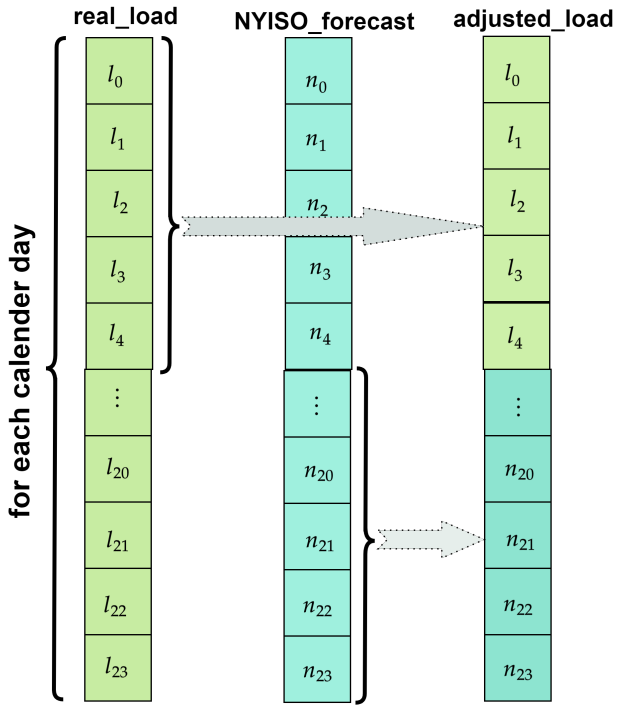

Figure 5.2: Real-load and load-forecast concatenation. For each calendar day in the load time series, we concatenate the first 5 hours of the real-load with the last 19 hours of the NYISO-forecast to give us the adjusted-load.

To transform our time series into a supervised learning problem, we first divide the data into training and test datasets to avoid leakage before normalizing them. We then divide the train and test set into standard 24 hours samples. We iterate over the entire training set, dividing the data into an overlapping window of 48 hours sample inputs mapping to 24 hours output (Figure 5.3). Overlapping windows increase the number of observations and are applied only to the training set. Figures 5.4 and 5.5 are plots of the training loss function at each epoch for BiLSTM and AttnLSTM whose hyperparameter specifications are summarized in the Tables 4.2 and 4.3. Both figures show a training loss that decreases to the point of stability, which shows that the BiLSTM and AttnLSTM are not underfitting but keep learning the training set until they can no longer learn any new information.

5.1 Analysis of Results

We present results from three major STLF models. The Cluster-WED, BiLSTM and the AttnLSTM models. There is a fourth model we would call the ZonalBiLSTM, where we train the 11 zones independently with the BiLSM model. We compute their aggregate as the forecast for the entire New York. We use data from 2012 to 2018 as the training set for Cluster-WED, BiLSTM and AttnLSTM

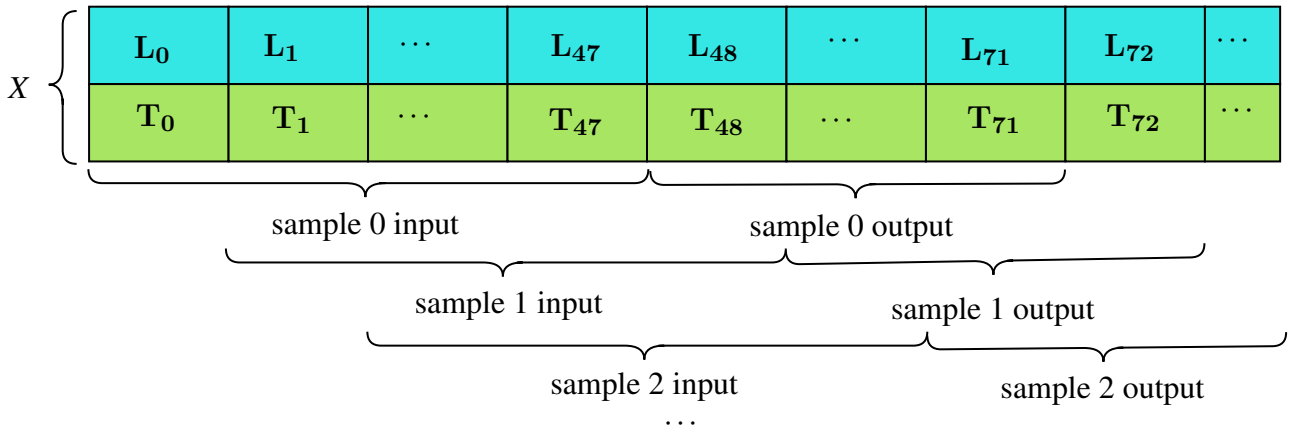

Figure 5.3: Input-output overlapping windows. At sample 0, we move the window from the 0^{th} to the 47^{th} member of the time series; this becomes our sample 0 input mapping the following 24 observations as output. In sample 1, we move the input window one timestep forward and repeat the process.

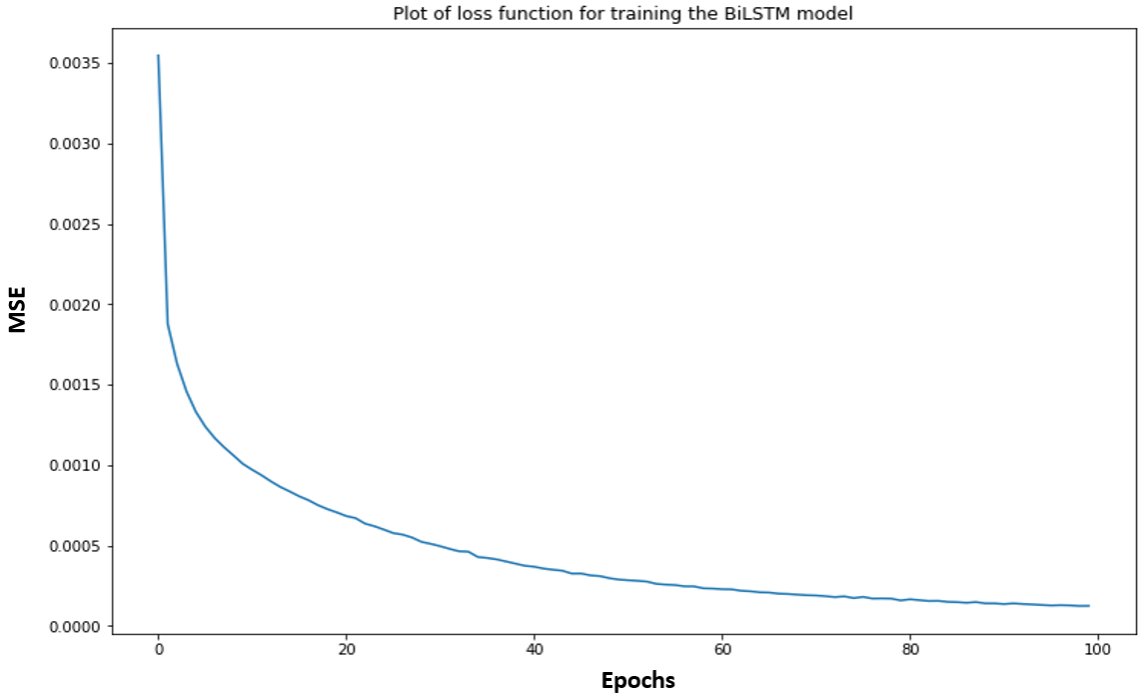

Figure 5.4: Plot of loss function, `BiLSTM`.

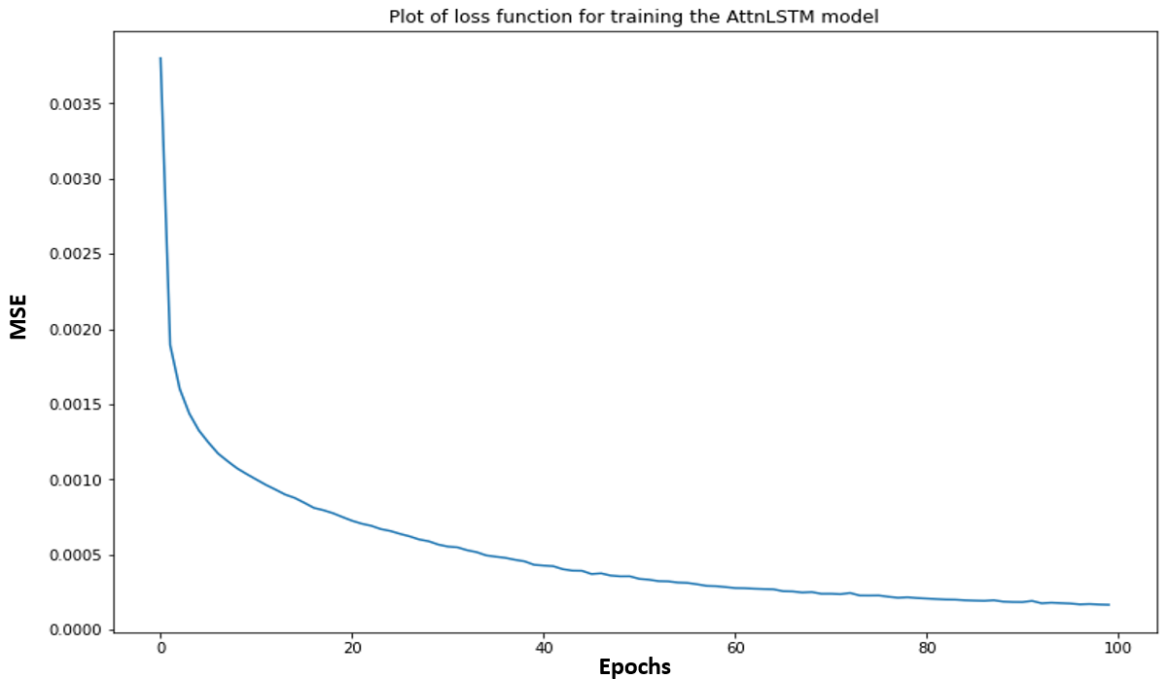

Figure 5.5: Plot of loss function, `AttnLSTM`.

and 2015 to 2018 as the training set for `ZonalBiLSM`. We test all models on data for 2019. We did not use 2020 data as a test set because of the pandemic's impact on load demands.

There is the need to define a measure of accuracy to measure the performance of models. In STLF literature, the absolute percentage error (APE) and the mean absolute percentage error (MAPE) is probably the most used error measure. APE compares the hourly forecast value $\hat{y}$ to the hourly actual load demands y, whereas the MAPE is over 24 hours. Thus,

$$\text{APE}_t = \frac{|y_t - \hat{y}_t|}{y_t} * 100 \tag{5.1.1}$$

$$\text{MAPE}_t = \frac{1}{24} * \sum_{i=1}^{24} \text{APE}_i. \tag{5.1.2}$$

We make predictions with our models using a walk-forward validation scheme. We predict 24 hours; then, the observable data is made available to the model for predictions of the next 24 hours.

A model overfits if it performs well on the training set but poorly on the test set. Since we use the mean squared error as our loss function and APE/MAPE as the performance measure, we sample 2016, 2017, and 2018 from the training set; then apply the `BiLSTM` and `AttnLSTM` on each in-sample yearly data sampled from the training set. We compute the APEs for each in-sampled yearly data. Finally, we compare it to performance on the test set (out-sample), 2019. From Tables 5.3 and 5.4, we do not have enough reason to say the two models are overfitting. The average APE for in-sample years 2016, 2017, and 2018 are 3.57, 4.56 and 4.17, respectively. Compared to the out-sample year 2019, we observe the `BiLSTM` performing better for the out-sample year than it did for 2017 and 2018. We observed a similar result for the `AttnLSTM` model; it performs better in 2019 than it did for in-sample years 2017 and 2018. With these observations, we do not have enough grounds to suggest

	insample_2016	insample_2017	insample_2018	outsample_2019
mean	3.57	4.56	4.17	3.92
std	2.98	3.68	3.44	3.34
min	0.00	0.00	0.00	0.00
25%	1.40	1.77	1.59	1.48
50%	2.87	3.79	3.39	3.07
75%	4.89	6.31	5.82	5.45
max	21.64	21.91	27.89	23.36

Table 5.3: APEs of `BiLSTM` model on some in-sample and out-sample data. APE compares the hourly forecast against the actual hourly observation and expresses the absolute error as a percentage of the actual observation.

	insample_2016	insample_2017	insample_2018	outsample_2019
mean	3.92	5.02	4.72	4.17
std	3.30	3.97	3.73	3.60
min	0.00	0.00	0.01	0.00
25%	1.50	2.01	1.87	1.53
50%	3.13	4.14	3.88	3.23
75%	5.32	6.89	6.56	5.81
max	20.87	26.08	23.30	28.07

Table 5.4: APEs of `AttnLSTM` model on some in-sample and out-sample data.

that `BiLSTM` and `AttnLSTM` are overfitting the training data.

	Capital	Central	West	Genese	NYC	Long-Is
mean	5.27	9.26	5.73	14.65	4.80	6.19
std	4.58	7.45	5.34	11.66	4.89	6.07
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	1.87	3.48	2.00	5.54	1.38	1.95
50%	4.11	7.39	4.32	12.12	3.15	4.26
75%	7.40	13.14	7.78	20.55	6.41	8.45
max	34.38	43.42	50.68	71.87	32.00	46.86

	North	Dunwod	Milwod	Mhk-Vl	Hud-Vl
mean	4.17	5.92	7.74	11.64	6.28
std	3.53	5.68	6.51	6.58	5.39
min	0.00	0.00	0.00	0.01	0.00
25%	1.56	1.95	2.73	6.44	2.19
50%	3.41	4.24	6.02	11.79	4.92
75%	5.93	8.10	11.03	16.21	8.86
max	84.17	64.62	52.40	33.65	36.84

Table 5.5: APE statistics of the eleven zones of NYCA trained with `BiLSTM` and tested on 2019 data.

	bilstm	attn_lstm	clus_wed	zonal_bilstm	NYISO
mean	3.93	4.18	3.90	4.39	2.97
std	3.36	3.61	3.30	3.72	1.87
min	0.00	0.00	0.00	0.00	0.00
25%	1.48	1.54	1.48	1.64	1.54
50%	3.07	3.23	3.12	3.47	2.81
75%	5.45	5.81	5.39	6.09	4.11
max	23.36	28.07	27.39	25.64	14.44

Table 5.6: APE statistics of the different models after testing on the out-sample 2019 data.

We also assess the performance of `BiLSTM` on the individual zones for the year 2019 (Table 5.5). Tables 5.6 and 5.7 compare different models' performance by comparing their APE and MAPE values. The `ZonalBiLSTM` is the model with the worst performance; this is counter-intuitive. We had expected that treating each zone individually, like the NYISO model, would improve performance. We are getting such results partly due to differences in information whiles putting exogenous weather parameters together; Tables 5.1 and 5.2 show the disparity in weather information for the NYISO and our models.

	bilstm	attn_lstm	clus_wed	zonal_bilstm	NYISO
mean	3.93	4.18	3.90	4.39	2.97
std	2.24	2.47	2.38	2.66	1.29
min	0.99	0.82	0.84	0.95	0.54
25%	2.39	2.38	2.41	2.54	2.04
50%	3.21	3.45	3.26	3.57	2.93
75%	4.92	5.27	4.74	5.37	3.68
max	13.34	16.87	19.22	14.66	10.34

Table 5.7: MAPE statistics of the different models after testing on the out-sample 2019 data. MAPE compares the mean daily forecast against the actual daily mean observation and expresses the absolute error as a percentage of the actual daily mean observation.

The `Cluster-WED` model has the smallest MAPE of 3.90% compared to 3.93% and 4.18% for the `BiLSTM` and `AttnLSTM`, respectively. However, the statistics indicate that the `Cluster-WED` model is not as robust as the `BiLSTM`; `Cluster-WED` shows higher variance predicting values further away from the mean than the `BiLSTM` model. For instance, we see from Table 5.5 that the maximum value for `BiLSTM` is 23.36% compared to 27.39% for the `Cluster-WED`. Table 5.7 shows a standard deviation of 2.24% for the `BiLSTM` compared to 2.38% for the `Cluster-WED` model.

Figure 5.6 is the plot of hourly observations for the real-load and forecast loads from the `BiLSTM` and `Cluster-WED` models. The first plots are observations in January and February, the second and third from May to June and September to October, respectively. Observing the second and third plots, we notice how mostly `BiLSTM` is under-predicting values whereas `Cluster-WED` is over-predicting. An example is the second plot before 2019-05-22. The `Cluster-WED` model is not very robust and sometimes predicts very much higher values than those observed; this is evident when we compare the daily averages of the real-load and the `Cluster-WED` forecast (Figure 5.8). We see at least three instances where the model predicted higher values than the observed load. The `Cluster-WED` model is a simple non-learning algorithm that does not require any complex assumptions and formulation. Given the historical input data, it uses clustering and weighted Euclidean distance measure to sample five closest days to the forecast day in terms of their weather parameters. The averages of these five days become our forecast values. The `Cluster-WED` model does not take into consideration the seasonal variation. As such, we expect it to be less robust compared to the `BiLSTM` model, which is learnable and able to capture the seasonal variation and trends.

Figures 5.7 and 5.8 compare the daily average forecast of the `BiLSTM` and `Cluster-WED` models with the real-load. Figure 5.13 is a plot of the residuals of the three models, `BiLSTM`, `AttnLSTM` and `Cluster-WED`, and Figure 5.9 plots the distribution of the residual for each of the models analyzed. Figures 5.10, 5.11 and 5.12, fit a student t-distribution to the residuals of `BiLSTM`, `AttnLSTM` and `Cluster-WED`, respectively.

5.2 Chapter Summary

This chapter presents the results from training and testing our models on NYISO load data and historical weather data. We compare the performance of `Cluster-WED, BiLSTM, AttnLSTM`, and

Figure 5.6: Hourly plots of real and predicted loads by `Cluster-WED` and `BiLSTM`, 2019

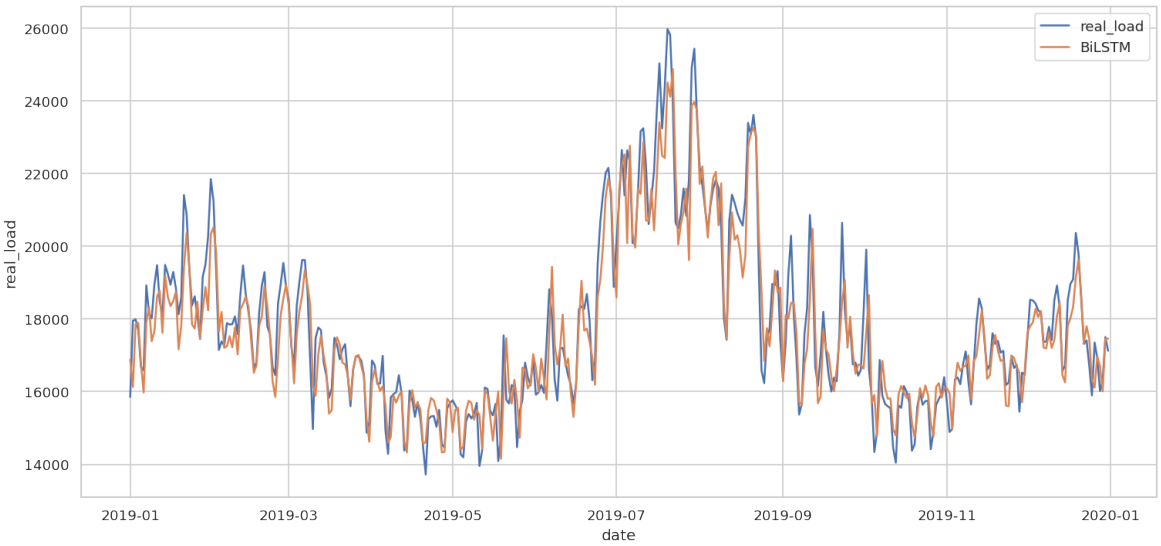

Figure 5.7: Daily average plots of real-load and `BiLSTM` forecast, 2019

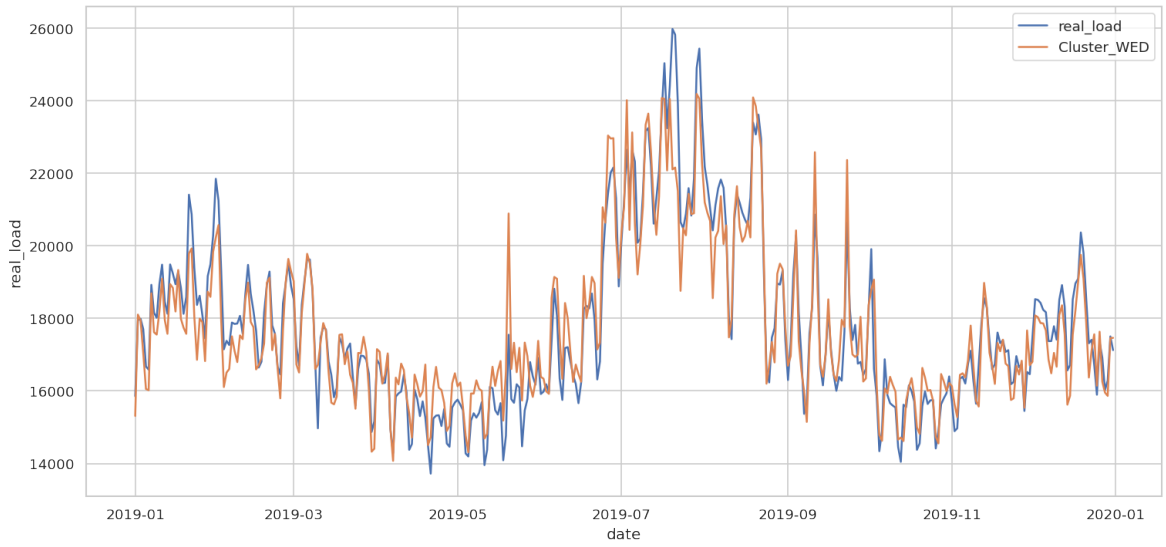

Figure 5.8: Daily average plots of real-load and `Cluster-WED` forecast, 2019

`ZonalBiLSTM` with that of the NYISO model. The `Cluster-WED` is a hybrid model that uses K-means clustering and weighted Euclidean distance to cluster and sample five closest days according to weather parameters, day-type and normal-anomalous selection criterion.

The `BiLSTM`, `AttnLSTM`, and `ZonalBiLSTM` models are all special RNN models that use LSTM cells. RNN is very efficient for language processing and can be adapted for structured time series data. The `BiLSTM` model is an effort to find a model that would outperform the `Cluster-WED` and NYISO model. The encoder-vector of a seq2seq model like `BiLSTM` creates a bottleneck making it difficult for the model to cope with long input sequences. The `AttnLSTM` model alleviates this defect by considering all hidden states of the encoder when computing the context vector.

Tables 5.6 and 5.7 give the statistical summary of the performance of the different models. The `BiLSTM` outperforms the `AttnLSTM` and `ZonalBiLSTM`. The `Cluster-WED` model slightly outperforms the `BiLSTM` model. However, the `BiLSTM` is more consistent at forecasting values that are closer to the actuals than `Cluster-WED` would do. For instance, we see from Table 5.6 where `BiLSTM` has its maximum APE being 23.36%, whereas `Cluster-WED` is 27.39%. We see the same for Table 5.7, where the maximum value for `BiLSTM` and `Cluster-WED` are 13.34% and 19.22%, respectively.

Neither of these models outperformed the NYISO model partly due to the loss of information in aggregating the weather parameters for New York. NYISO uses seventeen weather stations to aggregate the weather conditions for the state of New York, whereas, in this thesis, we use only ten to aggregate for New York. They also use economic indicators [41], which we have no access to. To find the aggregate weather parameter for the entire New York state, We used the linear combination approach first proposed by Sobhani et al. [50]. This linear combination method gave a better-performing model than the simple average.

NYISO publishes historic load demand up to the fifth hour each day of the week; this implies the two RNN models would be missing nineteen hours to make 24 hours ahead forecast. Hence the adjusted load in Figure 5.2. Despite all these shortfalls in information, we achieved an average APE of 3.93% for `BiLSTM` and 3.90% for the `Cluster-WED` model, which is still within the permissible error limit for a robust forecast [53]. Load forecasting is an arduous task, partly due to the non-

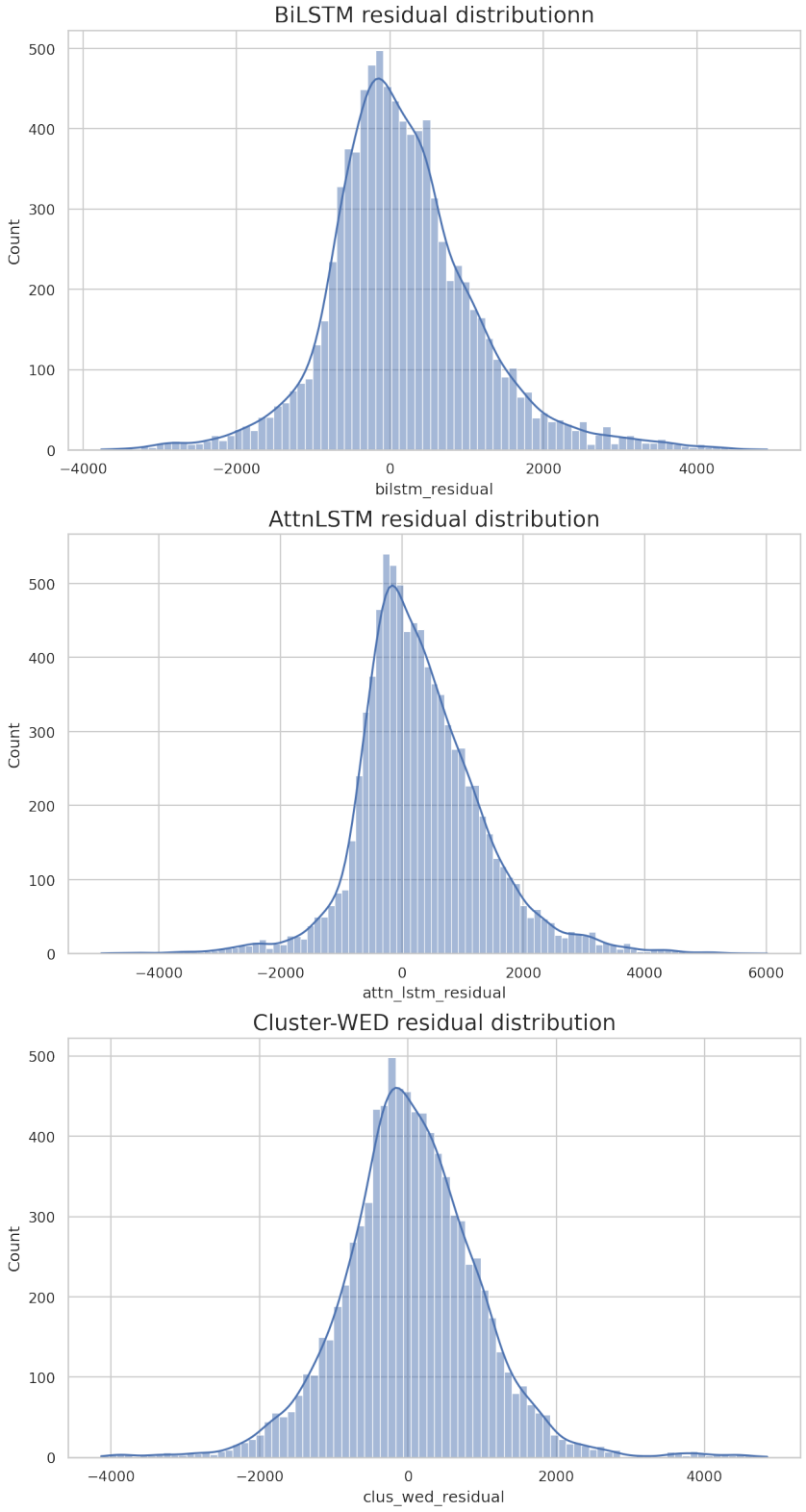

Figure 5.9: Residual distribution of `BiLSTM`, `AttnLSTM` and `Cluster-WED`, 2019

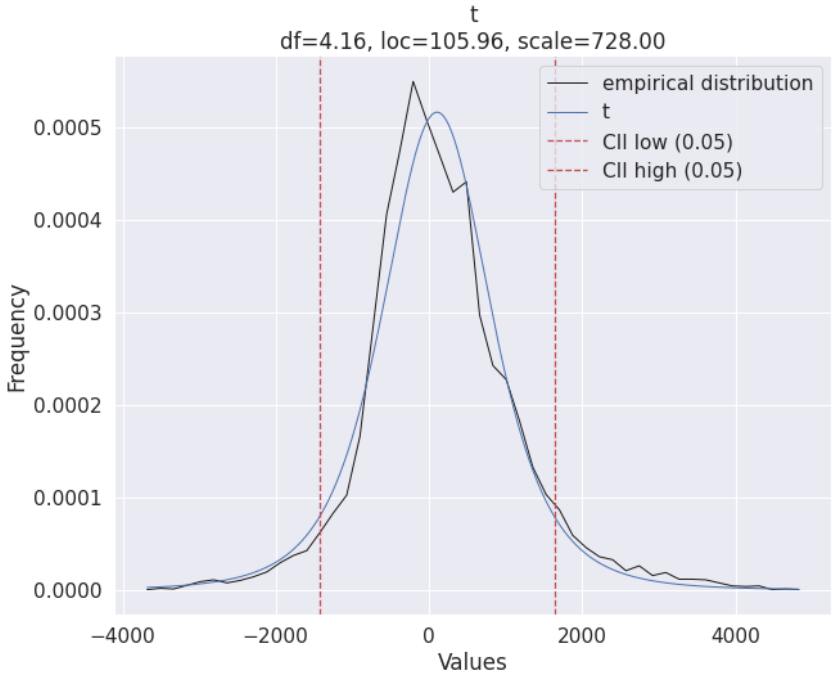

Figure 5.10: Fitted t-distribution for `BiLSTM` residuals with 4.16 degree of freedom.

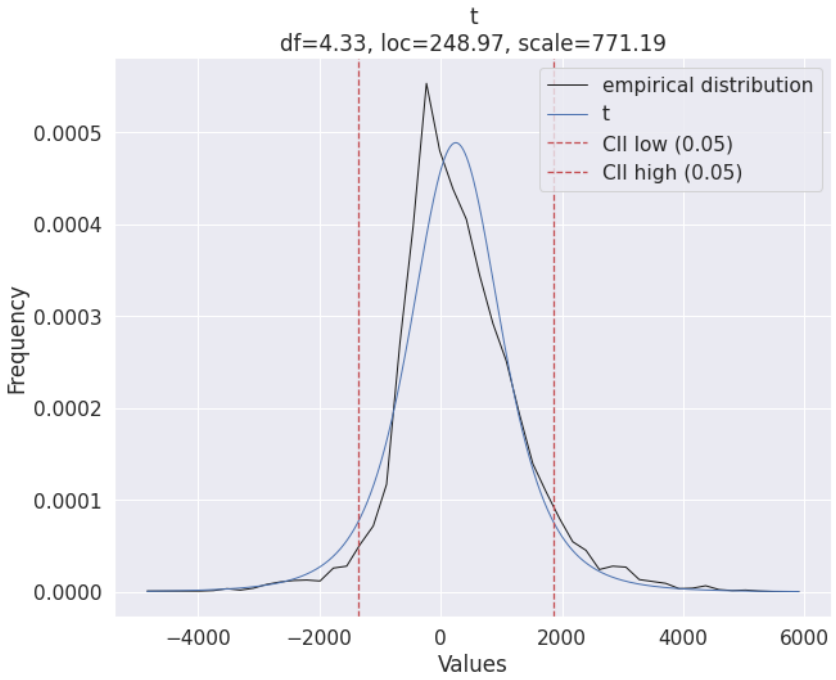

Figure 5.11: Fitted t-distribution for `AttnLSTM` residuals with 4.33 degree of freedom.

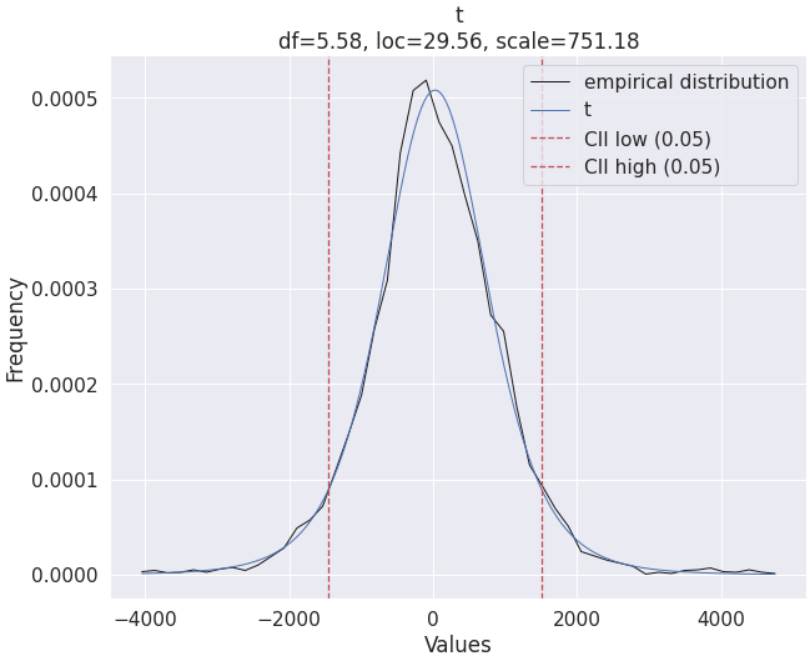

Figure 5.12: Fitted t-distribution for `Cluster-WED` residuals with 5.58 degree of freedom.

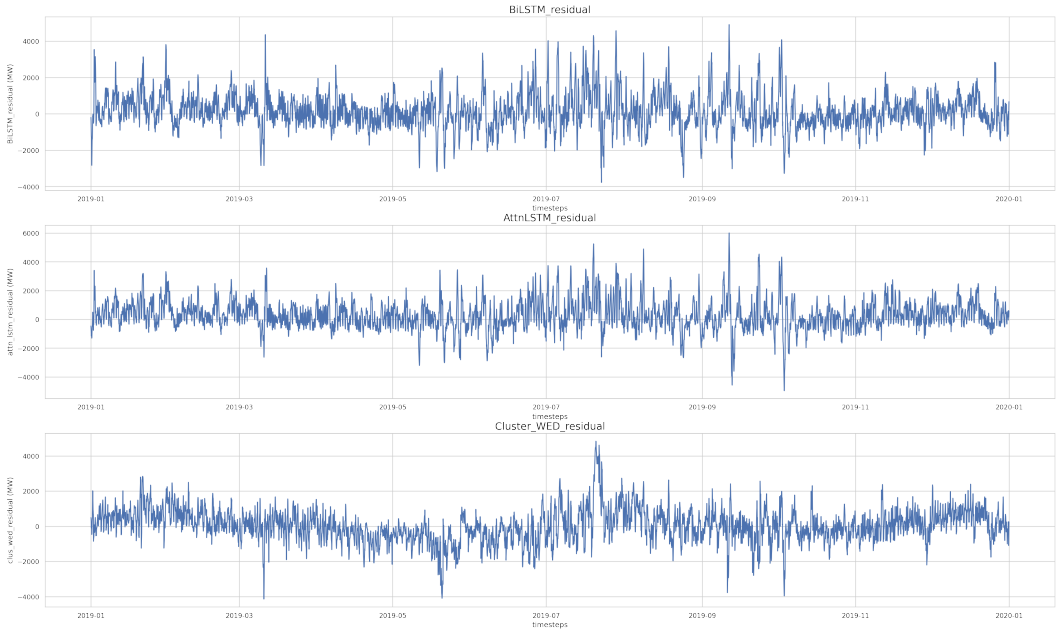

Figure 5.13: Residuals of `AttnLSTM`, `Cluster-WED` and `BiLSTM`, 2019

stationarity and seasonality of the time series data. Also, multicollinearity in exogenous weather parameters influencing load consumption adds complexity. However, the result obtained from the `Cluster-WED` suggests that a complex task like STLF could be executed with comparatively simple nonlinear models that are computationally less expensive.

Deep learning for STLF has shown excellent capabilities these past few years [60]; they can effectively capture temporal features from data and model complex nonlinear relations between the load and exogenous parameters. The LSTM cell deals with the vanishing/exploding gradient within RNN cells. Numerical results from the `BiLSTM` model show good generalization capabilities and are consistent in its forecast, indicating more room for improved performance once we bridge the information gap.

Chapter 6

Conclusion

Short-term load forecast (STLF) is complex due to the nonlinearity and non-stationarity of historic load demand data. Multicollinearity among exogenous input parameters also adds to the complexity. This thesis seeks to develop an STLF model for Plant-E Corp to forecast the day-ahead loads that rival the NYISO model. NYISO uses ANN to forecast load for each of the eleven zones, using economic, historical weather and load data.

Eventually, we developed three main models, the `Cluster-WED`, `BiLSTM` and `AttnLSTM` models. The `Cluster-WED` is a hybrid technique that uses clustering and weighted Euclidean distance to sample five closest days to the forecast day. We use a similar day selection criterion, quantitative variables like weather and load, and qualitative variables like normal-anomalous days without any complex formulation. We also realized that adding CTHI to temperature and humidity increased the forecasting accuracy of the `Cluster-WED` model. The second tier of the `Cluster-WED` involves finding weights for each for the exogeneous variables, temperature, CTHI and humidity. The authors of [24] recommended the least square approach. Our findings indicated that the least square method could not be generalized and could lead to a negative value under the root sign when computing the weighted Euclidean distance norm. We propose the Dominance Analysis to compute the weights for CTHI, temperature and humidity. Dominance Analysis allows us the opportunity to partition the overall model's R^2 in the attributable contribution of each predictor. It has a good generalization property and avoids a negative value under the root sign. The `Cluster-WED` model is simple to implement and does not require a lot of computing power and time. However, we realized that they lead to high variance in some cases, making them inconsistent sometimes.

The `BiLSTM` and `AttnLSTM` are recurrent neural networks (LSTM) sequence-to-sequence architecture. The LSTM deals with the vanishing/exploding gradient challenge inherent in RNN cells. A typical LSTM cell consists of the forget, input, and output gates, which are neural network layers with a sigmoid activation function. These gates decide which information to store or discard.

The encoder in the `BiLSTM` model summarizes all information from the input into a fixed dimension encoder-vector, creating a bottleneck in the architecture. This fixed dimension encoder-vector affects the performance of a typical sequence-to-sequence model. However, `AttnLSTM` also a sequence-to-sequence model considers all the hidden states of the encoder by assigning weights of importance to each hidden state when producing the following output in the decoder. The attention mechanism is to alleviate the bottleneck effects of the `BiLSTM` model.

The `Cluster-WED` model outperformed the the `BiLSTM` model by a small margin, with an average MAPE of 3.90 compared to 3.93 for `BiLSTM` over the entire year 2019. None of the proposed mod-

els outperformed the NYISO model, partly due to the gap in the information when aggregating the weather parameters for the whole of New York; we also do not have access to the economic data used in the NYISO model. This result does not come as a surprise since the performance of a forecasting model is as good as the input data.

However, given the correct input information, this research has proved that one could use simple non-linear models like `Cluster-WED` for complicated tasks like short-term load forecasting. It also shows how powerful and consistent deep learning models are when adapted and applied to structured time series data. Due to the structure of the LSTM cell with features like the hidden and memory cell state in LSTM cells, RNNs can capture the correlation of the load series in a time dimension, giving them leverage over traditional non-learning algorithms. The sequence-to-sequence architecture was initially developed for natural language processing (NLP) problems but adapted to solve myriad time series problems. This aspect of RNNs for STLF is a vibrant research area that needs more studies.

In the future, the immediate logical step would be to bridge the information gap between inputs presented by Plant-E Corp and NYISO. We consider it reasonable to consider individual zones when incorporating weather variables; this would give us grounds to compare the performance of our models against the NYISO model. Finally, deep learning is one area in artificial intelligence that has seen rigorous research and application across many fields of data science. In future work on STLF, we would like to explore the performance of a double attention model [45] and an input attention mechanism with skip connections [62] on STLF.

Bibliography

[1] Rahul Kumar Agrawal, Frankle Muchahary, and Madan Mohan Tripathi. Long term load forecasting with hourly predictions based on long-short-term-memory networks. *Texas Power and Energy Conference (TPEC)*, pages 1–6, 2018.

[2] K. P. Amber, M. W. Aslam, and S. K. Hussain. Electricity consumption forecasting models for administration buildings of the UK higher education sector. *Energy and Buildings*, 90:127–136, 2015.

[3] Nima Amjady. Short-term hourly load forecasting using time-series modeling with peak load estimation capability. *IEEE Transactions on Power Systems*, 16(4):798–805, 2001.

[4] Sara Atef and Amr B. Eltawil. Assessment of stacked unidirectional and bidirectional long short-term memory networks for electricity load forecasting. *Electric Power Systems Research*, 187, 2020.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*, 2015.

[6] Andrey Bogomolov, Bruno Lepri, Roberto Larcher, Fabrizio Antonelli, Fabio Pianesi, and Alex Pentland. Energy consumption prediction usingpeople dynamics derived from cellular network data. *EPJ Data Science*, 5:1–13, 2016.

[7] M. R. Braun, H. Altan, and S. B. M. Becka. Using regression analysis to predict the future energy consumption of a supermarket in the UK. *Applied Energy*, 130:305–313, 2014.

[8] David V. Budescu. Dominance analysis: A new approach to the problem of relative importance of predictors in multiple regression. *Psychological Bulletin*, 114(3):542–551, 1993.

[9] Derek W Bunn and E. D. Farmer. *Comparative models for electrical load forecasting*. Wiley, New York, 1985.

[10] Changchun Cai, Yuan Tao, Tianqi Zhu, and Zhixiang Deng. Short-term load forecasting based on deep learning bidirectional LSTM neural network. *Applied Sciences*, 11, 2021.

[11] Yaping Deng, Hao Jia, Pengcheng Li, Xiangqian Tong, Xiaodong Qiu, and Feng Li. A deep learning methodology based on bidirectional gated recurrent unit for wind power prediction. *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 591–595, 2019.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2011.

[13] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[14] Meftah Elsaraiti, Gama Ali, Hmeda Musbah, Adel Merabet, and Timothy Little. Time series analysis of electricity consumption forecasting using ARIMA model. *IEEE Green Technologies Conference (GreenTech)*, 2021.

[15] Vance Faber. Clustering and the continuous k-means algorithm. *Los Alamos Science*, 1(22), 1994.

[16] Nelson Fumo and M.A. Rafe Biswas. Regression analysis for prediction of residential energy consumption. *Renewable and Sustainable Energy Reviews*, 47:332–343, 2015.

[17] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. New York : Springer, second edition, 2009.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the 2016 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] Luis Hernández, Carlos Baladrón, Javier M Aguiar, Belén Carro, Antonio Sánchez-Esguevillas, and Jaime Lloret. Artificial neural networks for short-term load forecasting in microgrids environment. *Energy*, 75, 2014.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[21] Bin Huang, Yuying Liang, and Xiaolin Qiu. Wind power forecasting using attention-based recurrent neural networks: A comparative study. *IEEE access*, 9:40432–40444, 2021.

[22] Amit Jain and B. Satish. Integrated approach for short term load forecasting using SVM and ANN. *TENCON 2008 - 2008 IEEE Region 10 Conference*, pages 1–6, 2008.

[23] Amit Jain and B. Satish. Clustering based short term load forecasting using support vector machines. *In: Proceedings of IEEE Bucharest Power Tech*, 2009.

[24] M. Babita Jain and Venu Gopala Rao. A clustering and weighted Euclidean norm-based short-term load forecasting of normal and anomalous days. *Innovations in Electrical and Electronic Engineering*, pages 61–77, 2021.

[25] Harveen Kaur and Sachin Ahuja. SARIMA modelling for forecasting the electricity consumption of a health care building. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(12), 2019.

[26] A Khotanzad, R Afkhami-Rohani, T L Lu, A Abaye, M Davis, and D J Maratukulam. ANNSTLF-artificial neural network short-term load forecaster- generation three. *IEEE Trans Neural Network*, 8(4), 1997.

[27] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam. ANNSTLF-artificial neural network short-term load forecaster- generation three. *IEEE*, 1998.

[28] A.S. Khwaja, X. Zhang, A. Anpalagan, and B. Venkatesh. Boosted neural networks for improved short-term electric load forecasting. *Electric Power Systems Research*, 143, 2017.

[29] Tae-Young Kim and Sung-Bae Cho. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy*, 182:72–81, 2019.

[30] Diederik P. Kingma and Jimmy Lei Ba. ADAM: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. ICLR, 2015.

[31] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, page 95–104, 2018.

[32] Y. W. Lee, Tay Kim Gaik, and Choy Yaan Yee. Forecasting electricity consumption using time series model. *International Journal of Engineering & Technology*, 7(4):218–223, 2018.

[33] Jun Lin, Jin Ma, Jianguo Zhu, and Yu Cui. Short-term load forecasting based on LSTM networks considering attention mechanism. *International Journal of Electrical Power & Energy Systems*, 2022.

[34] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to Attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*, page 1412–1421, 2015.

[35] Pin Lv, Song Liu, Wenbing Yu, Shuquan Zheng, and Jing Lv. EGA-STLF: A hybrid short-term load forecasting model. *IEEE access*, 8:31742–31752, 2020.

[36] Patrick E. McSharry, Sonja Bouwman, and Gabriël Bloemhof. Probabilistic forecasts of the magnitude and timing of peak electricity demand. *IEEE Transactions on Power Systems*, 20(2): 1166–1172, 2005.

[37] Laurence Morissette and Sylvain Chartier. The k-means clustering technique: General considerations and implementation in Mathematica. *in Tutorials in Quantitative Methods for Psychology*, 9(1):15–24, 2013.

[38] Shahzad Muzaffar and Afshin Afshari. Short-term load forecasts using LSTM networks. *Elsevier Ltd.*, 2019.

[39] Francisco J. Nogales, Javier Contreras, Antonio J. Conejo, and Rosario Espínola. Forecasting next-day electricity prices by time series models. *IEEE Transactions on Power Systems*, 17(2): 342–348, 2002.

[40] NYISO. Load forecast uncertainty modeling: New York temperature distribution analysis. Technical report, New York Independent System Operator, 2021.

[41] NYISO. *Day-Ahead SchedulingManual*. New York Independent System Operator, 11 edition, September 2021.

[42] Alex D. Papalexopoulos and Timothy C. Hesterberg. A regression-based approach to short-term system load forecasting. *IEEE Transactions on Power Systems*, 5(4):1535–1547, 1990.

[43] Subodh Paudel, Mohamed Elmitri, Stéphane Couturier, Phuong H. Nguyen, René Kamphuis, Bruno Lacarrière, and Olivier Le Corre. A relevant data selection method for energy consumption prediction of low energy building based on support vector machine. *Energy and Buildings*, 138:240–256, 2017.

[44] Henrique Pombeiro, Rodolfo Santos, Paulo Carreira, Carlos Silva, and Joâo M. C. Sousa. Comparative assessment of low-complexity models to predict electricity consumption in an institutional building: Linear regression vs. fuzzy modeling vs. neural networks. *Energy and Buildings*, 146, 2017.

[45] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *International Joint Conference on Artificial Intelligence (IJCAI)*, 4, 2017.

[46] D. K. Ranaweera, N. F. Hubele, and G. G. Karady. Fuzzy logic for short term load forecasting. *Electrical Power & Energy Systems*, 18(4):215–222,, 1996.

[47] Muhammad Sajjad, Zulfiqar Ahmad Khan, Amin Ullah, Tanveer Hussain, Waseem Ullah, Mi Young Lee, and Sung Wook Baik. A novel CNN-GRU-based hybrid approach for short-term residential load forecasting. *IEEE access*, 2020.

[48] Gerald B. Sheble. *Computational Auction Mechanisms for Restructured Power Industry Operation*. Springer, 1999.

[49] Shun-Yao Shih, Fan-Keng Sun, and Hung yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441, 2019.

[50] Masoud Sobhani, Allison Campbell, Saurabh Sangamwar, Changlin Li, and Tao Hong. Combining weather stations for electric load forecasting. *Energies*, 12, 2019.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *arXiv:1409.3215 [cs.CL]*, 2014.

[52] R. Suwanda, Z. Syahputra, and E. M. Zamzami. Analysis of Euclidean distance and Manhattan distance in the K-Means algorithm for variations number of centroid K. *Journal of Physics: Conference Series*, 2020.

[53] David A. Swanson. On the relationship among values of the same summary measure of error when it is used across multiple characteristics at the same point in time: An examination of MALPE and MAPE. *Review of Economics & Finance*, 2015.

[54] James W. Taylor and Patrick E. McSharry. Short-term load forecasting methods: An evaluation based on european data. *IEEE Transactions on Power Systems*, 22(4), 2007.

[55] Scott Tonidandel and James M. LeBreton. Relative importance analysis: A useful supplement to regression analysis. *Journal of Business and Psychology*, 26(1), 2011.

[56] D.H. Vu, K.M. Muttaqi, and A.P. Agalgaonkar. A variance inflation factor and backward elimination based robust regression model for forecasting monthly electricity demand using climatic variables. *Applied Energy*, 140:385–394, 2015.

[57] Abdul Wahab, Muhammad Anas Tahir, Naveed Iqbal, Faisal Shafait, Syed Muhammad, and Raza Kazmi. Short-term load forecasting using bi-directional sequential models and feature engineering for small datasets. *IEEE Transactions on Power Systems*, 2020.

[58] Jian Qi Wang, Yu Du, and Jing Wang. LSTM based long-term energy consumption prediction with periodicity. *Energy*, 197, 2020.

[59] Aston Zhang, Zachary C. Lipton, Mu Li, , and Alexander J. Smola. Dive into deep learning. Release 0.17.0, November 2021.

[60] Lei Zhang, Linghui Yang, Chengyu Gu, and Da Li. LSTM-based short-term electrical load forecasting and anomaly correction. *CPEEE*, 2020.

[61] Mingfei Zhang, Zhoutao Yu, and Zhenghua Xu. Short-term load forecasting using recurrent neural networks with input attention mechanism and hidden connection mechanism. *IEEE access*, 8:186514–186529, 2020.

[62] Mingfei Zhang, Zhoutao Yu, and Zhenghua Xu. Short-term load forecasting using recurrent neural networks with input attention mechanism and hidden connection mechanism. *IEEE access*, 8:186514–186529, 2020.

[63] Jiaxiang Zheng, Xingying Chen, Kun Yu, Lei Gan, Yifan Wang, and Ke Wang. Short-term power load forecasting of residential community based on GRU neural network. *2018 International Conference on Power System Technology (POWERCON)*, pages 4862–4868, 2018.