Electric Load Forecasting Using Deep Neural Networks

By

Zain Ahmed

A thesis submitted to the

School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

May, 2025

St. John's Newfoundland and Labrador Canada

Abstract

Short-Term Load Forecasting (STLF) is a critical and complex task that plays a vital role in the efficient management of electricity generation, transmission, and distribution. Recent research has made strides in this field through the application of advanced deep learning techniques to enhance the accuracy and reliability of load predictions. The first study introduces a novel deep neural network tailored for STLF at Memorial University of Newfoundland (MUN). This model integrates electric load data with meteorological information and features a 1D Convolutional Neural Network followed by an Encoder-Decoder Network with an attention mechanism, showing superior performance compared to traditional Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) models. The study also focuses on optimizing the input horizon using the algorithm.

The second study focuses on Multi-Energy Systems (MES) and presents a Multi-Task Learning-based approach for load forecasting. It features a cutting-edge deep learning architecture designed to forecast multiple loads simultaneously. Applied to the University of Austin Tempe Campus, this approach employs a Deep Temporal Convolutional Neural Network (D-TCNet) to effectively capture spatial and temporal correlations in the data, resulting in improved forecasting accuracy across different energy types and seasons.

The third study compares various Recurrent Neural Network (RNN)-based time-series forecasting algorithms, including LSTM, GRU, Bi-directional GRU, and Bi-directional LSTM, on electric load data from MUN. The Bi-directional GRU model emerged as the top performer, achieving the highest R2 score and the lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE) for day-ahead predictions.

Collectively, these studies demonstrate the power of deep learning in enhancing the precision and effectiveness of short-term load forecasting, offering promising avenues for optimizing energy system operations.

Acknowledgments

First and foremost, I would like to thank my supervisor and co-supervisor Dr. Ashraf Ali Khan and Dr. Mohsin Jamil for their utmost guidance, support and flexibility in conducting this research and completing this thesis.

I would like to express my gratitude to the School of Graduate Studies and the Faculty of Engineering and Applied Sciences for deeming me a deserving candidate for this degree.

Finally, I would like to thank my family and friends for their unwavering support and help during these years.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RNN – Recurrent Neural Network

LSTM – Long Short Term Memory

GRU – Gated Recurrent Unit

ARIMA – Autoregressive Integrated Moving Average

STLF – Short-Term Load Forecasting

MSE – Mean Squared Error

MAE – Mean Absolute Error

MIC – Mutual Information Coefficient

DC – Distance Correlation

CNN – Convolutional Neural Network

SVM – Support Vector Machine

TCN – Temporal Convolutional Network

Chapter 1: Introduction

1.1 Literature Review

Load forecasting is a crucial tool for a power service provider. In the short term, it can significantly aid daily operations. Load predictions help operators anticipate peak hours and prevent issues like load shedding. Over a longer horizon, it assists maintenance staff in planning activities necessary to keep equipment in good running condition. Accurate load demand predictions can also influence policy and the future direction of the organization. Decisions regarding the expansion of current facilities or purchasing new units depend on these predictions. Financially, an increase in forecasting error correlates with higher operating costs. A one percent increase in forecasting error is associated with a $10 million increase in operating costs [1].

The load forecasting problem is often formulated as a time-series problem, where past values are used as inputs for the forecasting algorithm. Based on the prediction horizon, the problem can be divided into different types:

Ultra/Very Short-Term Load Forecasting: This type of forecasting predicts load a few minutes into the future. It is used in preventive control and emergency management of power systems [2]. Different organizations can use these predictions to adjust power scheduling in real time.

Short-Term Load Forecasting: Predicting the electric load one hour to one week ahead falls under short-term load forecasting. This type of forecasting significantly impacts system reliability. Underestimation can put the system under immense pressure, leading to power shortages or outages, while overestimation results in excess power generation and wasted

resources [3]. This forecasting is also crucial for generation scheduling, such as optimizing the release of hydro reservoirs for hydro-power generation and determining the unit commitment function for thermal generation units to minimize operation costs [3].

Medium-Term Load Forecasting: This involves predicting electric load from several weeks to several months ahead. Medium-term load forecasting is important for setting electricity prices, network management scheduling, and deciding power distribution mechanisms [4].

Long-Term Load Forecasting: Forecasting electricity demand over a planning horizon longer than one year is known as long-term load forecasting. System adequacy is established based on these forecasts. As peak load is the worst-case scenario, it is used as the prediction target. Peak load predictions are tested against system capacity for generation, transmission, and distribution [5]. These predictions are crucial for decisions related to expansions or additions of new units to the power utility company's present capacity.

Another classification for forecasting is based on the number of factors involved in making predictions. Algorithms can be divided into univariate and multivariate approaches:

Univariate Time-Series Forecasting: This method uses only past load values to predict future values.

Multivariate Time-Series Forecasting: This method uses multiple types of data to predict electric load demand. Since electric load can depend on various factors other than past load values, this method is preferable. It allows for more flexible and sophisticated approaches for future load predictions. Meteorological data and features describing holidays and workdays can be included. Some prediction methods use a multistep approach, constructing complex

features with an algorithm. These derived features are then used to make predictions, as exemplified in [6].

Prediction algorithms can also be categorized into parametric and non-parametric methods:

Parametric Algorithms: These algorithms predict based on a fixed number of parameters, independent of the amount of data. Examples include linear regression, Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA).

Non-Parametric Algorithms: These algorithms do not assume any structure about the data and freely learn and predict based on patterns and structures in the data. Decision trees, support vector machines, and deep neural networks are considered non-parametric methods.

Some of the major challenges faced in electric load forecasting are as follows:

1. Data Quality and Availability:

Incomplete or Noisy Data: Load data often contains gaps or noise, which can reduce the accuracy of forecasting models. Data might be missing due to equipment malfunctions, communication errors, or human mistakes.

Granularity and Resolution: The available data may lack the necessary granularity (e.g., hourly or sub-hourly intervals) or sufficient historical coverage, making it difficult to capture detailed consumption trends.

2. Integration of Renewable Energy Sources:

Variability and Uncertainty: Renewable energy sources like wind and solar are highly variable and unpredictable, adding complexity to load forecasting. Their intermittent nature introduces more uncertainty, especially in short-term forecasts.

Grid Flexibility: Balancing renewable energy sources with traditional power generation requires highly accurate load forecasts to maintain grid stability.

3. Evolving Consumption Patterns:

Changing Consumer Behavior: Shifts in consumer energy usage, influenced by factors like smart appliances, electric vehicles, and demand-side management, require load forecasting models to adapt to these evolving patterns.

Demand Response Programs: Increased participation in demand response programs, where consumers alter their usage based on price signals or incentives, adds variability to consumption patterns, making forecasting more difficult.

4. External Factors and Uncertainty:

Weather Conditions: Weather plays a major role in electricity demand (e.g., changes in temperature affecting heating and cooling). Sudden weather fluctuations or inaccurate weather predictions can reduce the accuracy of load forecasts.

Economic and Social Factors: Economic growth, industrial activity, and population changes impact energy demand, but these factors are often difficult to predict and incorporate into forecasting models.

5. Complexities in Multi-Energy Systems:

Interdependence Between Energy Sources: Multi-energy systems (e.g., electricity, gas, heat) involve interactions that must be modeled together. Predicting how different types of energy sources interact adds complexity to the forecasting process.

Distributed Energy Resources (DERs): The growing use of distributed generation, like rooftop solar, and energy storage, such as batteries, shifts the energy supply model away from centralized power plants, complicating load predictions.

6. High Dimensionality of Data:

Large Input Space: Forecasting models often require a wide range of variables, such as weather data, socioeconomic factors, and past load values. Managing and processing this high-dimensional data efficiently is a challenge.

Feature Selection: Identifying the most important variables (e.g., meteorological, economic, or time-series factors) is crucial to improving accuracy while reducing computational costs.

7. Temporal and Spatial Variations:

Short-Term vs. Long-Term Forecasting: Different forecasting horizons (short-term, medium-term, and long-term) come with distinct challenges, requiring models to account for immediate fluctuations and long-term trends.

Spatial Resolution: Forecasting across different spatial scales, from building-level to regional or national, demands that models be both accurate and adaptable to varying levels of aggregation.

8. Model Selection and Scalability:

Complexity of Models: AI-based models, such as neural networks and LSTMs, show promise but often require vast amounts of data, expertise, and computational power to develop and deploy. Balancing complexity and interpretability is an ongoing challenge.

Generalization: Ensuring that a forecasting model trained on one dataset or region can generalize effectively to other locations or time periods is difficult. Ensuring scalability across different grid sizes is also a concern.

9. Uncertainty in Predictions:

Confidence Intervals and Risk Assessment: Forecasting models need to provide not only a point estimate but also a measure of the uncertainty around that prediction, especially in scenarios where miscalculations could lead to significant financial or operational issues.

10. Integration with Smart Grids:

Real-Time Adjustments: As smart grids evolve, the need for real-time load forecasting becomes more critical. Continuously updating and adjusting forecasts in near real-time presents both technical and computational challenges.

Cybersecurity Risks: With the growing reliance on digital infrastructure and smart technology, forecasting systems are increasingly vulnerable to cyber-attacks, which could undermine data integrity and prediction accuracy.

Exponential Smoothing and Autoregressive methods have traditionally been the baseline for time series forecasting. However, these approaches require manual selection of inputs and rely on prior assumptions about the data. Specifically, ARIMA, one of the most widely used methods, assumes a linear relationship between future values and past observations, making it less effective in modeling highly non-linear patterns.

In contrast, time series data often display temporal dependencies, meaning that similar time points can lead to different future behaviors. Deep learning models, by comparison, have shown far greater potential for time series prediction because of their ability to capture complex

features and patterns within the data. For example, Artificial Neural Networks (ANNs) have demonstrated superior generalization ability in tasks like water quality prediction. Long Short-Term Memory (LSTM) networks, in particular, have outperformed traditional methods in short-term load forecasting, and both LSTM and Bi-directional LSTM have consistently delivered better results for day-ahead forecasting in independent buildings compared to more conventional approaches.

For campus load forecasting, traditional machine learning methods have also been explored. Rational Quadratic Gaussian Process Regression (RQ-GPR) was found to perform best among conventional methods. In another study, a combination of k-means clustering followed by an LSTM algorithm was used for campus load prediction. Additionally, a CNN + sequence-to-sequence model, trained on residential data, outperformed conventional RNN and LSTM models, further highlighting the advantages of newer techniques. While earlier models in this field have utilized the Luong attention mechanism other attention mechanisms have not been explored.

The goal of this research is to provide a state-of-the-art deep learning based short-term load forecasting method. The said method is an encoder-decoder model with Bahdanau Attention to capture long-range dependencies. The model also utilizes a 1D-Convolutional Neural Network to capture dependencies among different features. The proposed method aims to solve some of the challenges mentioned above which include high dimensionality of data and model generalization ability.

The second part of the research is concerned with multi-energy systems and provides a new deep learning based custom architecture for load forecasting. This proposed approach utilizes

multi-task learning to improve the accuracy of the current models. This method also aims to provide an efficient solution for high dimensionality data by utilizing distance correlation.

Below is an introduction of major techniques and methods used in load forecasting. It also introduces major components used in the proposed architectures.

1.2 Major Techniques

In this section we will be reviewing the algorithms typically used for electric load forecasting. The latest work done by using those techniques is also explained in detail. We will start by describing the statistical models followed by artificial intelligence based models. We end this section by describing the common evaluation metrics used for electric load forecasting and their pros and cons.

1.2.1 Statistical Models

Statistical models can be divided into 2 main categories based on the number of variables they employ for forecasting. These include univariate and multi-variate load forecasting models.

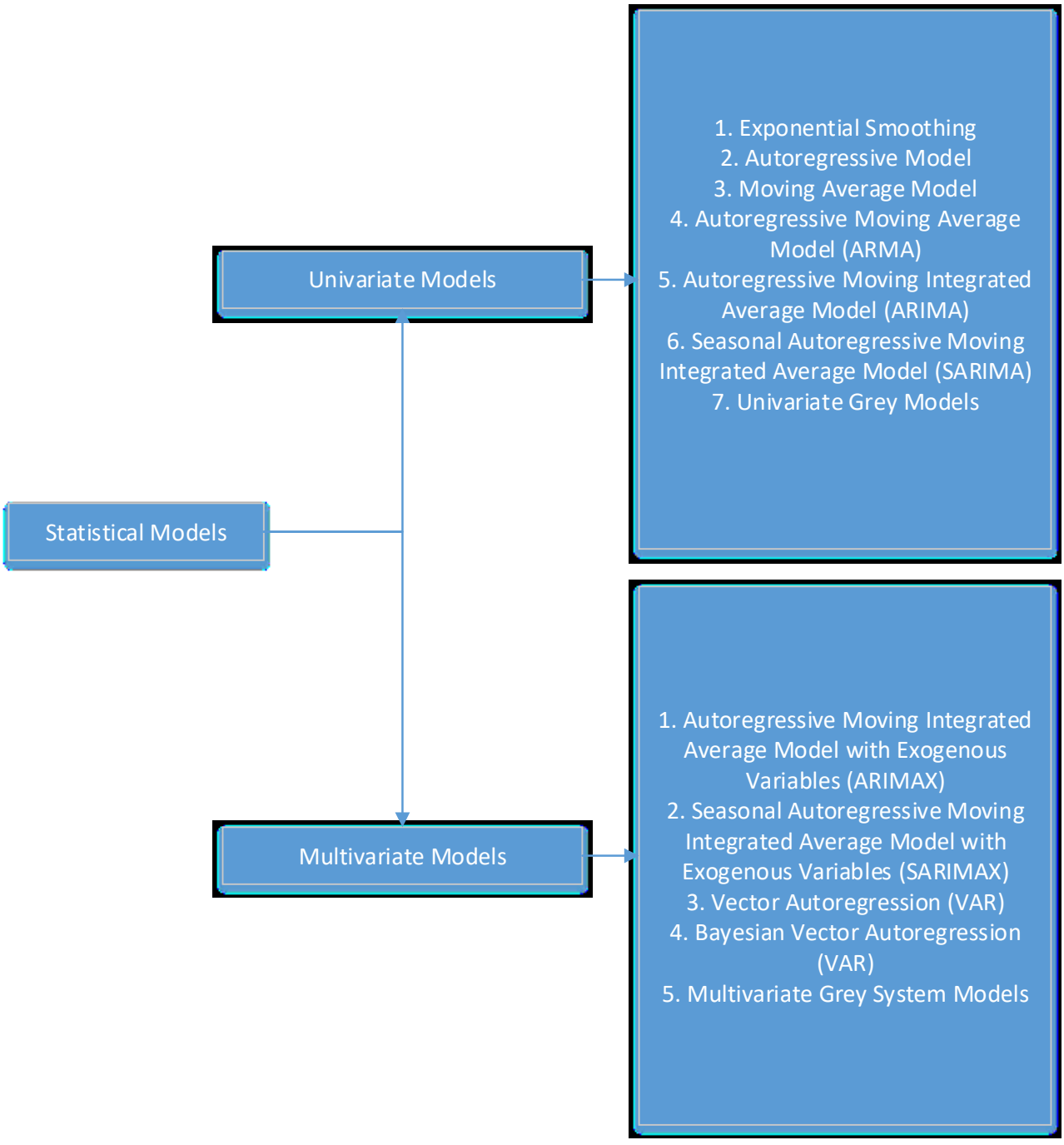


Figure 1: Basic Overview of Statistical Models

1.2.1.1 Univariate Models:

Univariate Models only employ past values from a single variable to make future predictions for the same variable. These models try to capture the seasonalities and trends in the data to estimate future values.

a. Exponential Smoothing:

Exponential Smoothing is a type of classical univariate approach to forecasting in which the weighted averages of past values are used to predict values in the future. The weights decay exponentially for older values. There can be up to nine different types of exponential smoothing methods based on the seasonality and trend (and additive, multiplicative or damped additive nature of their appearance) [7]. Some of these methods include:

- Simple exponential smoothing
- Holt's linear method
- Additive damped trend method
- Additive Holt's Winter method
- Multiplicative Holt's Winter method
- Holt-Winters' damped method

b. Autoregressive Model:

Autoregressive Model is similar to application of regression method using the lagged values of the target variable itself. The order of an autoregressive model depends on the number of lagged values used by it. An autoregressive model of order p represented by AR(p) would use past p values to

forecast the future. If we consider $\boldsymbol{P_t}$ to be the value of interest, then a pth order autoregressive model would use this equation for prediction.

$$\boldsymbol{P_t = A_1 P_{t-1} + A_2 P_{t-2} + \cdots + A_p P_{t-p} + B + C_t} \quad (\mathbf{1})$$

Here $\boldsymbol{C_t}$ represents the white noise and $\boldsymbol{A_1}$-$\boldsymbol{A_p}$ are the coefficients and B is a constant.

c. Moving Average Model:

In a moving average model, the predicted value (or target value) can be thought of as a weighted moving average of past forecast errors [7]. A moving average model of order q would utilize past q forecast errors and is represented as MA(q).

$$\boldsymbol{P_t = T_1 C_{t-1} + T_2 C_{t-2} + \cdots + T_q C_{t-q} + C_t + M} \quad (\mathbf{2})$$

Here $\boldsymbol{T_1}$-$\boldsymbol{T_q}$ represent the coefficients for the residual terms. The residual terms ($\boldsymbol{C_{t-1}}$-$\boldsymbol{C_{t-q}}$) are the differences between the predicted and actual values at their respective time steps. $\boldsymbol{M}$ represents the mean.

d. Autoregressive-Moving Average (ARMA) Model:

When moving average and autoregressive predictions are combined in a single model then we get a ARMA model. The prediction for a time step t represented by $\boldsymbol{P_t}$ would be written as follows for a ARMA(p,q) model where p represents the order of autoregressive portion while q represents the order of moving average part:

$$\boldsymbol{P_t = M + \sum_{i=1}^{p} A_i P_{t-i} + \sum_{i=1}^{q} T_i C_{t-i} + C_t} \quad (\mathbf{3})$$

Here autoregressive and moving average predictions are combined.

e. Autoregressive Integrated Moving Average (ARIMA) Model:

ARIMA model is obtained by combining the autoregression algorithm with the moving average algorithm after performing differencing on a homogenous nonstationary time series. An ARIMA(p,d,q) model consists of an autoregression model of order p, a differencing of order d ,and a moving average model of order q. Differencing is done to make the model stationary. Hence the ARIMA model can be written as:

$$P'_t = M + \sum_{i=1}^{p} A_i P'_{t-i} + \sum_{i=1}^{q} T_i C_{t-i} + C_t \quad (4)$$

The only difference is that $\boldsymbol{P'_t}$ is the differenced value and the order of differencing can be higher than 1. A first-order differencing is given as:

$$P'_t = P_t - P_{t-1} \quad (5)$$

f. Seasonal Autoregressive Integrated Moving Average (SARIMA) Model:

A SARIMA model is used to forecast univariate time-series which exhibits a seasonality. Seasonality can be described as presence of strong periodic patterns [8]. SARIMA model introduces other parameters which handle seasonality. A SARIMA(p,d,q)(P,D,Q)s has the parameters p,d and q which are same as a non-seasonal ARIMA model but also introduces P,D and Q to represent the order of seasonal auto regression, seasonal differencing and seasonal moving average respectively. The subscript "s" represents the frequency of repeating pattern.

g. Univariate Grey System theory-based models:

Grey system theory was introduced in 1980s and is well known method for studying systems with partially known information. This approach requires limited amount of data to study systems with unknown characteristics. The methods classify systems based on the amount of information available related to system. A black system has no available information while a white system is

completely known. A system where only partial information is available is termed a grey system [9]. Univariate grey system models include:

G (1,1): It would be described as a first order grey system model for single variable. It employs a differential equation to model the behavior of a data series.

G (2,1): It would be described as a second order grey system model for single variable. It would be able to capture more complex phenomenon compared to first order model.

1.2.1.2. Multivariate Models:

Multivariate methods employ multiple interdependent variables to capture patterns in the data and use that to improve the forecasting accuracy. Multivariate models include:

a. *Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX) Model:*

An ARIMAX model is a modification of ARIMA model to handle multivariate time-series. This model uses exogenous variables in addition to the lagging values of the time-series to make forecasts.

b. *Seasonal Autoregressive Integrated Moving Average with Exogenous Variables (SARIMAX) Model:*

Like ARIMAX, SARIMAX model is a modification of SARIMA model which can be used for a multivariate time series. This model uses additional or exogenous variables to predict the future values of the target variable while also handling seasonality.

c. *Vector Autoregression (VAR):*

Vector auto regression is a type of multivariate time series forecasting algorithm which treats the future values of a particular variable in a time series as a function of past values of variable itself and past values of other variables in the system.

d. Bayesian Vector Auto egression (BVAR):

BVAR is a type of vector autoregression which uses Bayesian framework to improve the forecasting capability of standard VAR Model. Rather than using fixed values for lagged values in a traditional VAR model, BVAR uses random variables with prior probabilities as input for the model. BVAR models offer regularization by incorporating priors to prevent overfitting. BVAR offers more robust prediction performance as it leverages prior distributions.

e. Multivariate Grey System theory-based models:

Grey system theory-based models can also be used for multiple variables. It would be described as a G (1, N) model. This type of model would model a primary variable by using N-1 related variables.

1.2.2 Artificial Intelligence-Based Models:

1.2.2.1 Artificial Neural Networks:

Artificial Neural Networks (ANNs) are at the heart of deep learning. They consist of various node layers where each node takes input from nodes of the previous layer and assigns weights to each of these inputs. This computation is passed through an activation function (Sigmoid, ReLU, or Hyperbolic Tangent), which allows the network to learn highly non-linear features as computation proceeds deeper into the model. A simple artificial neural network consists of an input layer, an output layer, and a hidden layer.

ANNs can be divided into various types based on the types of inputs they can handle and the type of computations they perform. Different types of ANNs are as follow:

a. Recurrent Neural Networks:

Recurrent neural networks (RNNs), which were first introduced as Hopfield networks [10], are ideal as they can handle sequential data, and information from past data points is used to carry out

computation, which not only affects the output but also the information carried to other data points. A recurrent neural network can take arbitrarily long sequences of input data and can generate arbitrarily long output. Based on the number of inputs and outputs, an RNN can be divided into one-to-one, one-to-many, many-to-one, and many-to-many architectures.

Based on the computations in a single RNN unit, this network can be divided into Simple RNNs, Gated Recurrent Units (GRUs) [11], and Long Short-Term Memory (LSTM) [12].

Based on the information flow, an RNN can be divided into unidirectional and bidirectional. A unidirectional RNN only uses information from previous time steps in its computation, while a bidirectional RNN [13] uses information from past and future time steps to generate the output.

A model which uses two separate RNNS is called sequence to sequence model [14] or an encoder-decoder model. Such a model can generate arbitrarily long sequence of output from an input. They can also capture complex dependencies in the data to generate predictions. One other major advantage of sequence to sequence models is that attention mechanism [15] can be applied to these models which allows the model to focus on different parts of the input while generating each part of the output sequence. The figure below shows a sample encoder decoder model. Each of the block represents an RNN block that could either be an LSTM or GRU block. It takes input and passes information among different blocks. The output is not shown here as it depends on the problem statement. In a many-to-many problem for example, the output of each decoder would be one complete output. In a many-to-one scenario, only output from the final block would be used.

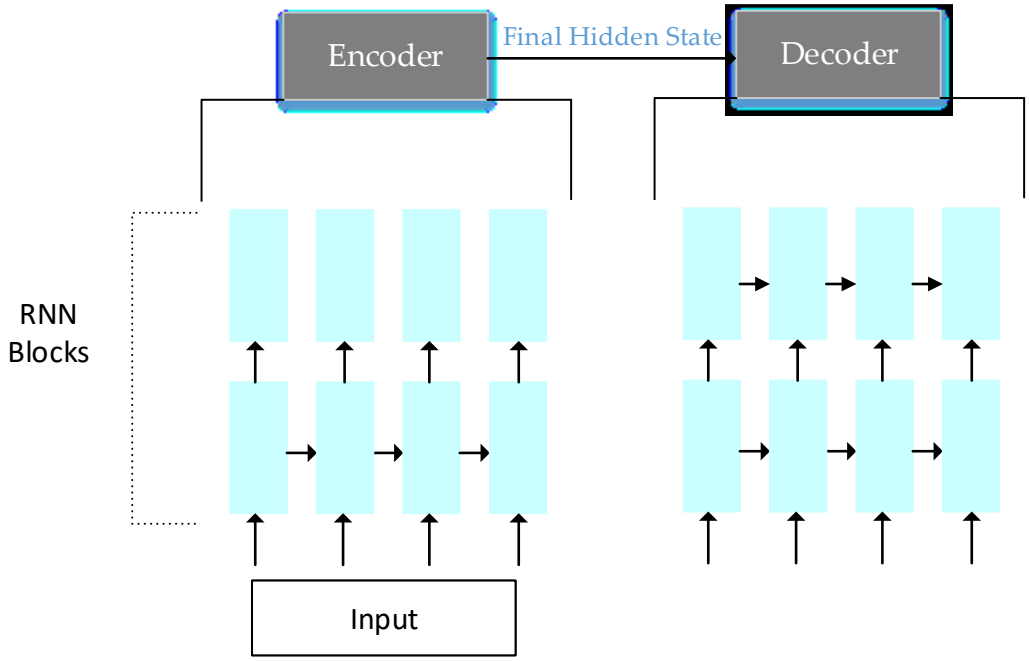


Figure 2: An RNN-based Encoder-Decoder Model

b. Transformer Networks:

Transformer Networks were originally introduced in [15]. They are they are part of many state-of-the-art models in Natural Language Processing (NLP). Models. They employ multiple mechanisms which include self-attention, multi-head attention, positional encoding and encoding-decoding. Self-attention mechanism allows the model to weigh the importance of different parts of the input sequence relative to each other. Multi-head attention breaks down a sequence and performs self-attention of each portion simultaneously Each of these attention operations is called a head. Encoder-decoder architecture divides the model into two separate portions. Encoder takes the input sequence and passes it along to the decoder which generates an output. Transformers do not consider position of the inputs (words or values). To remedy this, positional encoding is used to feed information about each input in the sequence. All these modifications to the architecture allow parallelization during model training unlike RNNs which process the information sequentially.

c. Convolutional Neural Networks (CNNs):

CNNs [16] are a type of neural network which are used on structured grid data like an image. Primarily they consist of convolution layers, pooling layers and fully connected layers. Convolutional layers consist of a convolution operation which generates a feature map by passing a kernel through the data. Pooling layer reduces the size of the feature map and fully connected layer is obtained by unrolling the feature map and making the final prediction.

Some of the major CNNs used for time series forecasting include 1D-CNNs [17], Temporal Convolutional Networks (TCNs) [18]and WaveNet [19].

1D-CNNs can be used to capture temporal features in a time series. The convolution is applied over the temporal dimension of the data. In modern time series forecasting research, 1D-CNNs are typically used along with other state-of-the-art algorithms to achieve better performance. In [20], 1D-CNNs are combined with Transformers for intraday stock prices. CNNs capture short term dependencies while Transformers model long term dependencies. The model performs better than exponential moving average model, ARIMA model and DeepAR model. In [21], RNN based LSTM model is combined with 1D-CNN for predicting Euro/United States Dollar (USD) exchange rate. Here, CNN is used to capture short-term dependencies while RNN based model captures long term dependencies.

TCNs are another type of convolutional neural networks which use a combination of techniques to make them suitable for time series forecasting including casual convolutions, dilated convolutions and residual connections. Casual convolutions modify the original convolution operation so temporal order of the data is respected. Dilated convolutions ensure a bigger receptive field which allow us to capture long range dependencies. Residual connections are also employed.

These allow the training of deeper networks preventing exploding or diminishing gradients. In [22], TCN outperforms LSTM based RNN network in load forecasting on two datasets.

WaveNet is a deep generative model to produce raw audio waveforms. It utilizes dilated convolutions to model long range temporal dependencies. This type of network has been used for time series forecasting. In [23], a WaveNet based encoder-decoder architecture is utilized to make short-term load forecasting predictions on French Grid data. In this work, the architecture performs better than multiple models such as LSTM based RNN, GRU based RNN, ARIMA, 1D-CNN and ConvLSTM.

d. Neural Basis Expansion Analysis of Time Series (N-BEATS):

N-BEATS is a deep neural network introduced in 2019 in [24]. The architecture consists of blocks in a stacked architecture. Each block consists of a trend component to capture long-term dependencies and a seasonal component to capture periodic and seasonal patterns in the data. Each block performs a neural basis expansion which transforms an input time-series into a higher dimensional space to extract temporal features. Multiple block makes a stack. And prediction from one stack is used by another to further refine the predictions. In [25], an evolutionary method using N-BEATS to predict energy consumption for individual customers in a smart grid. In [26], a hybrid algorithm is employed using N-BEATS algorithm, Transformer mechanism, Convolutional layers and Mish activation function for forecasting cryptocurrency portfolios.

e. Hybrid Networks:

These models combine different methodologies to cover weaknesses and improve forecasting performance of algorithms. One of the examples is Auto encoders + Recurrent Neural Networks. Autoencoders reduce dimensionality of the data and denoise it while RNN is used capturing temporal features and forecasting. Another example is CNN + LSTM. CNNs are used for capturing

spatial features while LSTM is focused on temporal features and forecasting.

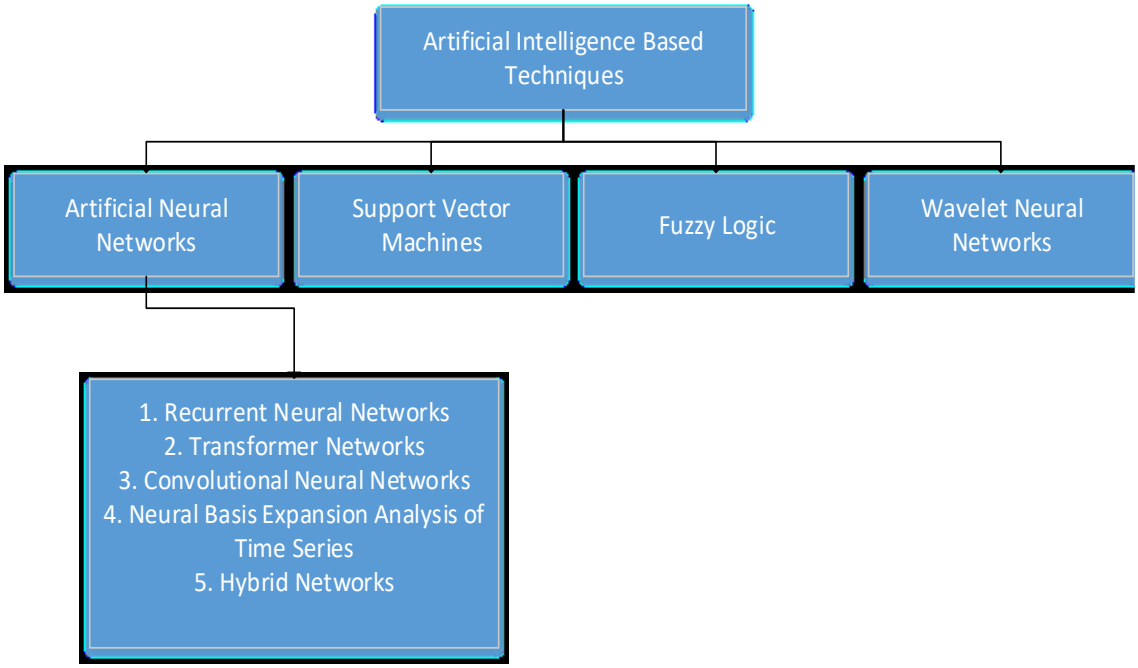


Figure 3: Artificial Intelligence based Algorithms

Latest work in hybrid models for time series forecasting involve [27] which combines Recursive Empirical Model Decomposition (REMD) with LSTM model to improve forecasting performance. REMD is used to decompose the data in intrinsic mode functions (IMFs) and then LSTM is used to predict the future values of those IMFs. Final value is obtained by accumulating different IMFs. In [28], LSTM is combined with random forests (RF) and CNN for forecasting. And it shows superior performance compared to conventional forecasting techniques. In this model, CNN is responsible for capturing spatial features while LSTM captures temporal dependencies. Random Forest (RF) algorithm is then used for making the final prediction.

1.2.2.2. Support Vector Machines:

Support vector machines (SVMs) are a type of supervised machine learning algorithm used in regression and classification. In a classification problem, SVMs use a kernel function to generate a decision boundary (hyperplane) between different classes using only the data points close to the hyperplane. These points are called support vectors. A variation of SVM called Support Vector Regression (SVR) [29] uses only a subset of data outside the preset margin and generates a best-fit hyperplane. A time series can be formulated as a regression problem, and support vector regression can be used to generate predictions [30].

1.2.2.3 Fuzzy Logic:

Fuzzy logic [31] is a branch of propositional calculus with truth values between 0 and 1. It is different from classical logic, which only has binary truth values. In fuzzy logic, 0 and 1 represent the extreme cases, while the values in between represent intermediate degrees of truth, imitating human reasoning. Fuzzy logic does not require the input and output values to be numerical, so natural language can also be used. This type of approach can be quite beneficial when historical data is not in the form of numerical data but rather in the form of linguistic values [32]. This technique has been applied to short-term load forecasting [33].

1.2.2.4. Wavelet Neural Network:

A wavelet neural network combines the classical neural network with wavelet analysis. All the variables are fed to the input layer, which is connected to the hidden layer with units called wavelons. They convert the input variables to dilated and translated versions of the mother wavelet. This layer is followed by the output layer, which generates the prediction [34]. Wavelet neural networks have been used for short-term load forecasting, where they are trained in fewer

Table 1: Summary of Major Techniques in Literature

Reference	Contributions
[7]	. Exponential smoothing: Uses weighted averages of past values with exponential decay. . Autoregressive (AR): Predicts using lagged values of the target variable. . Moving Average (MA): Uses past forecast errors as a moving average for predictions. . ARMA: Combines AR and MA for time series predictions. . ARIMA: Adds differencing to ARMA for non-stationary series. . ARIMAX: Incorporates exogenous variables with ARIMA. . SARIMAX: Handles seasonality and external variables.
[8]	. SARIMA: Extends ARIMA to handle seasonal patterns in data.
[9]	. Grey system models (G1,1 and G2,1): Models with limited data using grey system theory.
[10]	. Artificial Neural Networks (ANN): Learns complex non-linear patterns in data.
[11]	. Recurrent Neural Networks (RNN): Handles sequential data with memory from past steps.
[15]	. Transformer Networks: Uses attention mechanisms for parallel sequence processing.

[16]	. Convolutional Neural Networks (CNN): Captures short-term dependencies in structured data.
[24]	. N-BEATS: Uses stacked blocks to capture trends and seasonality.
[27]	. Hybrid Networks (e.g., CNN + LSTM): Combines models for better accuracy.
[29]	. Support Vector Machines (SVM): Applies support vectors for time series regression.
[31]	. Fuzzy Logic: Uses fuzzy logic for forecasting with linguistic data.
[34]	. Wavelet Neural Network: Combines wavelet analysis with neural networks for forecasting.

1.2.3. Evaluation Metrics

1.2.3.1. Mean Squared Error:

Mean Squared Error (MSE) is an evaluation metric used to judge the performance of a forecasting technique. It is calculated by averaging the squares of the residuals (the difference between the predicted and actual values). Due to squaring, the value is always positive and is quite sensitive to outliers. As the estimator or algorithm improves, the value of MSE approaches zero.

$$MSE = \frac{\sum_{i=1}^{N}(P_{i,prediction} - P_{i,actual})^2}{N} \quad (6)$$

1.2.3.2. Root Mean Squared Error:

Root Mean Squared Error (RMSE) is an evaluation metric obtained by taking the square root of the Mean Squared Error (MSE). Its values range from 0 to infinity, with 0 representing a model with residuals amounting to 0. This metric is more sensitive to outliers compared to Mean Absolute Error (MAE) due to the squaring of the residuals. However, RMSE is preferred over MSE because it is on the same scale as the data.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(P_{i,prediction} - P_{i,actual})^2}{N}} \tag{7}$$

1.2.3.3. Mean Absolute Error:

Mean Absolute Error (MAE) is an evaluation metric obtained by taking the average of the absolute values of the residuals. Since the absolute value is used, the value is always positive. As a more accurate algorithm is used, the residuals become smaller, and the average value decreases. Hence, a better estimator or algorithm would have a value closer to 0.

$$MAE = \frac{\sum_{i=1}^{N}\left|P_{i,prediction} - P_{i,actual}\right|}{N} \tag{8}$$

MAE is less sensitive to outliers since residuals are not squared. It has been argued that MAE is the most natural measure of average error and is unambiguous; hence, it should be used for inter-comparison of average model-performance error. Compared to MAE, RMSE does not have a clear interpretation as it is a function of MAE, the distribution of error magnitudes, and $N^{1/2}$ [38].

1.2.3.4. R2 Score:

R^2 Score is also referred to as the coefficient of determination. It is calculated as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(P_{i,prediction} - P_{i,actual})^2}{\sum_{i=1}^{N}(P_{i,actual} - M)^2} \quad (9)$$

It can be seen from the mathematical formulation that the coefficient of determination can have a maximum value of 1 for an estimator that predicts all the ground truth values with complete accuracy. However, this evaluation metric can also have a negative value. A lower value would indicate a worse model. The value of the coefficient of determination would be 0 if the model predicts a constant mean value as the output.

Here M is the mean of all the observed values and is defined as:

$$M = \sum_{i=1}^{N}(G_i) \quad (10)$$

1.2.3.5. Mean Absolute Percentage Error:

Mean absolute percentage error (MAPE) can be defined by the equation below:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{P_{i,actual} - P_{i,prediction}}{P_{i,actual}}\right| \% \quad (11)$$

It is a percentage value obtained by averaging the absolute values of the ratio of error ($P_{i,actual} - P_{i,prediction}$) and actual value $P_{i,actual}$. The mean is multiplied by 100 to obtain the percentage. A lower value would imply that a model is better while a higher value would indicate a worse performing model. This evaluation metric is scale independent hence it can be used to measure model performance across multiple datasets. But one of the major issues with percentage based

evaluation metrics is that they become infinite or undefined when P_{actual} becomes 0. Another weakness of this metric is that it assumes a meaningful 0 so they make no sense when forecasting temperatures on Celsius scale. Finally, the MAPE also suffers from penalizing positive errors much more heavily compared to negative errors [39]. this has led to the use of symmetric errors [40].

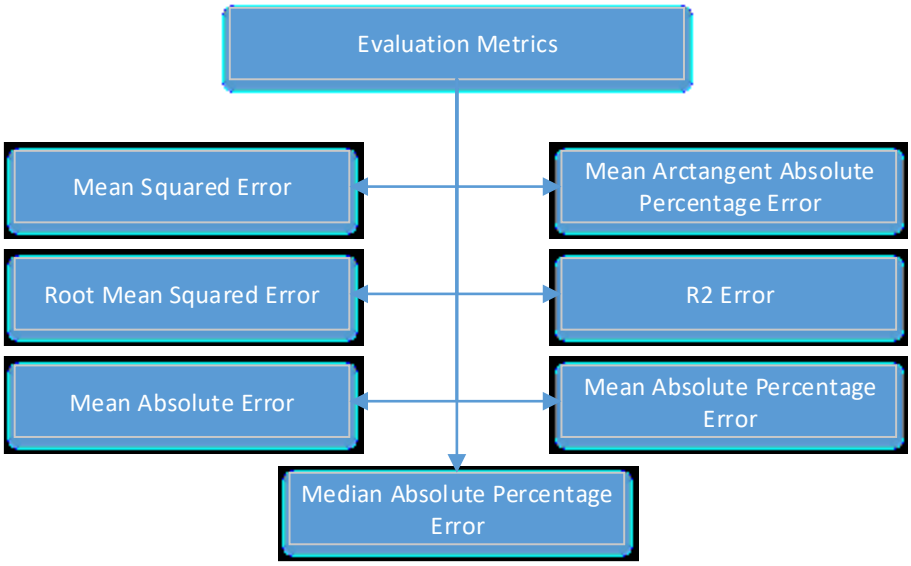


Figure 4: Evaluation Metrics

1.2.3.6. Mean Arctangent Absolute Percentage Error:

This new metric has been proposed to overcome the issues with MAPE metric producing indefinite or infinite values when actual values are close to zero. If we look at the adjacent side of a triangle as absolute actual value and opposite side as absolute value of difference between actual and forecast value, the MAPE can be seen as a slope in the form of a ratio. Mean arctangent absolute percentage error (MAAPE) allows us look at the error as the slope but in the form of an angle.

It can be defined as:

$$\text{MAAPE} = \frac{1}{N}\sum_{i=1}^{N} \arctan\left(\left|\frac{P_{i,actual} - P_{i,prediction}}{P_{i,actual}}\right|\right) \quad (12)$$

The arctangent function applied to absolute values is bounded with the range $[0, \frac{\pi}{2}]$. This property of the function prevents infinite values for zero or close to zero actual values. It is also more robust to outliers due to its bounded nature. If extremely large errors have important implications for algorithm performance, then MAAPE is not a suitable metric [41].

1.2.3.7. Median Absolute Percentage Error:

Median absolute percentage error can be defined as follow:

$$MdAPE = \text{median}(\left|\frac{P_{i,actual} - P_{i,prediction}}{P_{i,actual}}\right|) \quad (13)$$

As median is used instead of mean as the measure of central tendency in this case, the effect of outliers is reduced. This evaluation metric is preferred when we need to reduce the effect of outliers on algorithm selection or evaluation.

1.2.3.8. Mean Absolute Scaled Error:

To define Mean Absolute Scaled Error (MASE), we first define the scaled error (SE) as

:

$$SE_i = \frac{P_{i,prediction} - P_{i,actual}}{\sum_{i=2}^{N}\left|P_{i,actual} - P_{i-1,actual}\right| / N - 1} \quad (14)$$

We simply take the mean of the scaled error to obtain MASE as:

$$MASE = mean(|SE_i|) \quad (15)$$

This method compares the MAE for a given forecast with the mean absolute error of a one-step naïve prediction model. If the metric is lower than one, it indicates that on average, the performance of the forecasting algorithm is better than the naïve model. Conversely, if the value is higher than one, then it shows that MAE calculated for the forecasts generated by the algorithm is worse than a naïve mode [39].

1.3 Major Contributions of the Research Work

Major Contributions of the research work are summarized below.

Part 1:

1. A novel deep learning network architecture for short-term electric load forecasting. The architecture uses a hybrid neural network to make use of spatiotemporal patterns in the data for forecasting. It consists of a 1D CNN and a GRU-based encoder-decoder with Bahdanau attention for short-term electric load forecasting. Author has not observed the use of this attention mechanism in the literature for multivariate short term load forecasting for campuses.
2. Input horizon optimization for more accurate forecasting results, This novel neural network decouples the input window length from number of training parameters. This allows much faster training times for longer input windows.
3. Robustness testing framework for deep neural network algorithms. A robustness testing framework is developed and utilized to assess the change in model performance in case of addition of noise to the data.
4. Model performance comparison between the proposed architecture and other popular algorithms for campus electric load forecasting. Data used for this comparison study is taken from Memorial University St. John's Campus along with metrological data of the

city. Comparison is carried out with other deep learning techniques which include a GRU-based RNN, an LSTM-based RNN, and 1D CNN + Encoder Decoder model without attention. The model shows a higher R2 Score and lower MAPE, MAE, and MSE.

Part 2:

The main purpose of the work in part two is focused on electric load forecasting for multi energy systems using multi task learning. The major contributions of that work are summarized below:

1. Investigate the coupling relationship between cooling, heating and electric loads using distance correlation across 4 seasons of the year.
2. Propose a novel multi-task learning based approach for short-term load forecasting. First, distance correlation analysis is used to investigate coupling relationship between multiple load time series. Appropriate input variable selection is carried out based on the analysis. The selected load data along with exogenous variables are used as input to D-TCNet which is a TCN based forecasting network.

1.4 Thesis Outline

The thesis consists of five chapters. First chapter is focused on the problem statement and the literature review.

Second chapter is mainly focused on the comparison between different electric load forecasting algorithms using a dataset from MUN. The main focus of this chapter is the use of conventional recurrent neural networks and their use in electric load forecasting.

The third chapter proposes a novel electric load forecasting algorithm which combines 1D convolution, sequence to sequence model and attention mechanism for electric load forecasting. The chapter also investigates the performance of this algorithm on MUN electric load dataset. The

input horizon optimization is also performed to ensure optimal performance and finally a robustness testing framework is proposed and tested on the algorithm.

The fourth chapter is concerned with multi energy systems and load forecasting for those systems. A novel architecture is proposed which uses distance correlation and multi task learning to perform load forecasting for heating, electric and cooling loads.

The final chapter ends with concluding remarks and research contributions of the research. The chapter also proposes future prospects of the research in the field of electric load forecasting as well as load forecasting of multi energy systems.

1.5 References

[1] W. Zhang, Q. Chen, J. Yan, S. Zhang, and J. Xu, "A novel asynchronous deep reinforcement learning model with adaptive early forecasting method and reward incentive mechanism for short-term load forecasting," *Energy*, vol. 236, p. 121492, Dec. 2021, doi: 10.1016/j.energy.2021.121492.

[2] M. Tan, S. Yuan, S. Li, Y. Su, H. Li, and F. He, "Ultra-Short-Term Industrial Power Demand Forecasting Using LSTM Based Hybrid Ensemble Learning," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 2937–2948, Jul. 2020, doi: 10.1109/TPWRS.2019.2963109.

[3] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352–1372, Oct. 2015, doi: 10.1016/j.rser.2015.04.065.

[4] P. Matrenin, M. Safaraliev, S. Dmitriev, S. Kokin, A. Ghulomzoda, and S. Mitrofanov, "Medium-term load forecasting in isolated power systems based on ensemble machine learning models," *Energy Reports*, vol. 8, pp. 612–618, Apr. 2022, doi: 10.1016/j.egyr.2021.11.175.

[5] M. Dong and L. Grumbach, "A Hybrid Distribution Feeder Long-Term Load Forecasting Method Based on Sequence Prediction," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 470–482, Jan. 2020, doi: 10.1109/TSG.2019.2924183.

[6] A. Wahab, M. A. Tahir, N. Iqbal, A. Ul-Hasan, F. Shafait, and S. M. Raza Kazmi, "A Novel Technique for Short-Term Load Forecasting Using Sequential Models and Feature Engineering," *IEEE Access*, vol. 9, pp. 96221–96232, 2021, doi: 10.1109/ACCESS.2021.3093481.

[7] R. J. Hyndman and G. Athanasopoulos, "Forecasting: Principles and Practice," p. 504.

[8] D. C. Montgomery, "Introduction to Time Series Analysis and Forecasting," p. 469.

[9] E. Kayacan, B. Ulutas, and O. Kaynak, "Grey system theory-based models in time series prediction," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1784–1789, Mar. 2010, doi: 10.1016/j.eswa.2009.07.064.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, Art. no. 6088, Oct. 1986, doi: 10.1038/323533a0.

[11] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," Oct. 07, 2014, *arXiv*: arXiv:1409.1259. doi: 10.48550/arXiv.1409.1259.

[12] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[13] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, pp. 2673–2681, Dec. 1997, doi: 10.1109/78.650093.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," arXiv.org. Accessed: Jul. 11, 2024. [Online]. Available: https://arxiv.org/abs/1409.3215v3

[15] A. Vaswani *et al.*, "Attention Is All You Need," arXiv.org. Accessed: Jul. 11, 2024. [Online]. Available: https://arxiv.org/abs/1706.03762v7

[16] Y. LeCun *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.

[17] Z. Wang, W. Yan, and T. Oates, “Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline,” Dec. 14, 2016, *arXiv*: arXiv:1611.06455. doi: 10.48550/arXiv.1611.06455.
[18] S. Bai, J. Z. Kolter, and V. Koltun, “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,” arXiv.org. Accessed: Jun. 01, 2023. [Online]. Available: https://arxiv.org/abs/1803.01271v2
[19] A. van den Oord *et al.*, “WaveNet: A Generative Model for Raw Audio,” Sep. 19, 2016, *arXiv*: arXiv:1609.03499. doi: 10.48550/arXiv.1609.03499.
[20] K. Kim, “Financial time series forecasting using support vector machines,” *Neurocomputing*, vol. 55, no. 1, pp. 307–319, Sep. 2003, doi: 10.1016/S0925-2312(03)00372-2.
[21] M. Markova, “Forex Time Series Forecasting Using Hybrid Convolutional Neural Network/Long Short-Term Memory Network Model,” 2023, pp. 295–305. doi: 10.1007/978-3-031-21484-4_26.
[22] “Applied Sciences | Free Full-Text | Temporal Convolutional Networks Applied to Energy-Related Time Series Forecasting.” Accessed: Jul. 15, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/10/7/2322
[23] F. Dorado Rueda, J. Durán Suárez, and A. del Real Torres, “Short-Term Load Forecasting Using Encoder-Decoder WaveNet: Application to the French Grid,” *Energies*, vol. 14, no. 9, Art. no. 9, Jan. 2021, doi: 10.3390/en14092524.
[24] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, “N-BEATS: Neural basis expansion analysis for interpretable time series forecasting,” Feb. 20, 2020, *arXiv*: arXiv:1905.10437. doi: 10.48550/arXiv.1905.10437.
[25] “DIGWO-N-BEATS: An evolutionary time series prediction method for situation prediction.” Accessed: Jul. 16, 2024. [Online]. Available: https://www.researchgate.net/publication/378242728_DIGWO-N-BEATS_An_Evolutionary_Time_Series_Prediction_Method_for_Situation_Prediction
[26] A. Sbrana and P. A. Lima de Castro, “N-BEATS Perceiver: A Novel Approach for Robust Cryptocurrency Portfolio Forecasting,” *Comput Econ*, Sep. 2023, doi: 10.1007/s10614-023-10470-8.
[27] Y. Yang, C. Fan, and H. Xiong, “A novel general-purpose hybrid model for time series forecasting,” *Appl Intell*, vol. 52, no. 2, pp. 2212–2223, Jan. 2022, doi: 10.1007/s10489-021-02442-y.
[28] “Hybrid machine learning model combining of CNN-LSTM-RF for time series forecasting of Solar Power Generation - ScienceDirect.” Accessed: Jul. 16, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S277267112400216X
[29] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, “Support Vector Regression Machines,” p. 7.
[30] E. Ceperic, V. Ceperic, and A. Baric, “A Strategy for Short-Term Load Forecasting by Support Vector Regression Machines,” *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4356–4364, Nov. 2013, doi: 10.1109/TPWRS.2013.2269803.
[31] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965, doi: 10.1016/S0019-9958(65)90241-X.
[32] S. Rahman, “Formulation and analysis of a rule-based short-term load forecasting algorithm,” *Proceedings of the IEEE*, vol. 78, no. 5, pp. 805–816, May 1990, doi: 10.1109/5.53400.

[33] D. K. Ranaweera, N. F. Hubele, and G. G. Karady, “Fuzzy logic for short term load forecasting,” *International Journal of Electrical Power & Energy Systems*, vol. 18, no. 4, pp. 215–222, May 1996, doi: 10.1016/0142-0615(95)00060-7.
[34] A. K. Alexandridis and A. D. Zapranis, “Wavelet neural networks: A practical guide,” *Neural Networks*, vol. 42, pp. 1–27, Jun. 2013, doi: 10.1016/j.neunet.2013.01.008.
[35] Z. Bashir and M. E. El-Hawary, “Short term load forecasting by using wavelet neural networks,” in *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8492)*, May 2000, pp. 163–166 vol.1. doi: 10.1109/CCECE.2000.849691.
[36] S. J. Yao, Y. H. Song, L. Z. Zhang, and X. Y. Cheng, “Wavelet transform and neural networks for short-term electrical load forecasting,” *Energy Conversion and Management*, vol. 41, no. 18, pp. 1975–1988, Dec. 2000, doi: 10.1016/S0196-8904(00)00035-2.
[37] Y. Chen, B. Yang, and J. Dong, “Time-series prediction using a local linear wavelet neural network,” *Neurocomputing*, vol. 69, no. 4, pp. 449–465, Jan. 2006, doi: 10.1016/j.neucom.2005.02.006.
[38] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Climate Research*, vol. 30, no. 1, pp. 79–82, Dec. 2005, doi: 10.3354/cr030079.
[39] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, Oct. 2006, doi: 10.1016/j.ijforecast.2006.03.001.
[40] S. Makridakis, “Accuracy measures: theoretical and practical concerns,” *International Journal of Forecasting*, vol. 9, no. 4, pp. 527–529, Dec. 1993, doi: 10.1016/0169-2070(93)90079-3.
[41] S. Kim and H. Kim, “A new metric of absolute percentage error for intermittent demand forecasts,” *International Journal of Forecasting*, vol. 32, no. 3, pp. 669–679, Jul. 2016, doi: 10.1016/j.ijforecast.2015.12.003.

Chapter 2: Short-Term Campus Load Forecasting using CNN-based Encoder-Decoder Network with Attention

The chapter is currently under review in Energies. The manuscript was submitted on 20th of August 2024. The manuscript has been accepted and article has been published.

Abstract

Short-Term Load Forecasting is a challenging research problem and has a tremendous impact on electricity generation, transmission, and distribution. A robust forecasting algorithm can help power system operators to better tackle the ever-changing electric power demand. This paper presents a novel deep neural network for short-term electric load forecasting for the St. John's campus of Memorial University of Newfoundland (MUN). The electric load data is obtained from the Memorial University of Newfoundland and it is combined with metrological data of St. John's. This is used to formulate a multivariate time-series forecasting problem. A novel deep learning algorithm is presented consisting of a 1D Convolutional Neural Network which is followed by an Encoder Decoder Based Network with attention. The input used for this model is the electric load consumption and metrological data while the output is the hourly prediction of the next day. The model is compared with Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) based Recurrent Neural Network. A CNN-based Encoder-Decoder Model without attention is also tested. The proposed model shows a lower Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and a higher R2 Score. These evaluation metrics show an improved performance compared to GRU and LSTM-based RNNs as well as the CNN-Encoder Decoder model without attention. An MAE of 407 kW and a MAPE of 3.37% has been achieved our proposed architecture.

Keywords: Load Forecasting; Convolutional Neural Network; Time Series Forecasting

2.1 Introduction

Modern power systems require an uninterrupted supply of electricity and which demands the least amount of error when determining electric load demand. [1]. Accurate day-ahead load forecasting is critical the for system's operation [2]. The financial impact of load forecasting accuracy is also very critical. 1% raise in forecasting error is associated with a 10-million-dollar increase in operating costs [3]. Hence, it is extremely important to increase the accuracy of load forecasting algorithms. Based on the forecasting horizon, load forecasting problem can be divided into 3 major categories: (1) Short-term load forecasting deals with intervals ranging from one hour to one week, (2) Medium-term forecasting deals with predicting electricity demand from one week up to 1 year and (3) Long-term forecasting deals with predictions longer than 1 year [4]. Forecasting methods can also be divided based on the number of inputs required by the algorithm. They can be divided into a multi-factor forecasting approach which uses multiple variables to predict the load demand and a time-series approach in which the algorithm only uses the univariate load data [5]. Forecasting algorithms can also be divided into statistical models, artificial intelligence-based models, and hybrid methods [6].

Statistical Algorithms can be divided into [6]:

1. Autoregressive (AR) Model
2. Moving Average (MA) Model
3. Autoregressive Moving Average (ARMA) Model
4. Autoregressive Integrated Moving Average (ARIMA) Model
5. Seasonal Autoregressive Integrated Moving Average (SARIMA) Model
6. ARIMAX and SARIMAX models

7. Kalman Filtering Algorithm
8. Grey Models
9. Exponential Smoothing (ES)

Artificial Intelligence and computational intelligence-based algorithms can be divided into:

1. Artificial Neural Network Algorithms (ANN)
2. Extreme learning machines (ELM)
3. Support Vector Machines (SVM)
4. Fuzzy Logic
5. Wavelet Neural Networks (WNN)
6. Genetic Algorithms (GA)
7. Expert System

Exponential Smoothing and Autoregressive approaches have been considered the baseline for time series forecasting but for these approaches, we have to manually set the number of inputs to use and also make apriori assumptions about the data [7]. In particular, ARIMA is one of the most popular approaches but the algorithm works under the assumption that future values are linearly related to the observed data points, hence it is unsuitable for modeling highly non-linear behaviour [8].

Time series exhibit temporal dependencies which cause two identical points in time to exhibit different behaviour in the future. For time series prediction tasks, deep learning architectures show greater potential due to their ability to learn complex features and patterns in the data. For water quality prediction tasks, ANN was found to have a greater generalization ability [9]. For short-term forecasting, Long Short Term Memory (LSTM) architecture showed superior performance

[10]. For day-ahead forecasting in independent buildings, either LSTM or Bi-directional LSTM showed superior performance compared to more conventional techniques [11].

For campus load forecasting, conventional machine learning techniques were employed in [12] and it was found that Rational Quadratic Gaussian Process Regression (RQ-GPR) showed the best performance. In [13], k-means clustering algorithm is used followed by LSTM algorithm for prediction of campus load data. In [14], a CNN + sequence to sequence model is proposed which is trained on residential data to achieve better performance compared to conventional RNN or LSTM models. Previous works in similar ANNs have used Luong attention mechanism but use of Bahdanau attention is absent from any literature for 1D CNN + Sequence to Sequence attention models. Work focuses on that model for campus load forecasting. Additionally, an extensive input horizon optimization study is performed to find optimal input horizon for the algorithm. Robustness of the algorithm is also studied by adding noise to the input data.

The work focuses on a novel deep-learning technique for electric load forecasting. It consists of a convolutional neural network followed by an encoder-decoder model with attention. The main contribution of the work includes:

1. A novel deep learning network architecture for short-term electric load forecasting. The architecture uses a hybrid neural network to make use of spatiotemporal patterns in the data for forecasting. It consists of a 1D CNN and a GRU-based encoder-decoder with Bahdanau attention for short-term electric load forecasting. Author has not observed the use of this attention mechanism in the literature for multivariate short term load forecasting for campuses.

2. Input horizon optimization for more accurate forecasting results, This novel neural network decouples the input window length from number of training parameters. This allows much faster training times for longer input windows.
3. Robustness testing framework for deep neural network algorithms. A robustness testing framework is developed and utilized to assess the change in model performance in case of addition of noise to the data.
4. Model performance comparison between the proposed architecture and other popular algorithms for campus electric load forecasting. Data used for this comparison study is taken from Memorial University St. John's Campus along with metrological data of the city. Comparison is carried out with other deep learning techniques which include a GRU-based RNN, an LSTM-based RNN, and 1D CNN + Encoder Decoder model without attention. The model shows a higher R2 Score and lower MAPE, MAE, and MSE.

The paper is divided into 5 Sections. In Section II, data would be described (campus load information and metrological data) are provided and the basic characteristics of the data are presented. In Section III, the algorithms are discussed. In Section IV, the procedure and simulation results are discussed. Finally, Section V concludes the paper.

2.2 Data Description

2.2.1. Campus Load Information

Memorial University of Newfoundland (MUN) is located in St. John's Newfoundland and Labrador (NL). The average temperature of the city can range from 6 degrees Celsius to as high as 21 Degrees Celsius. Due to this harsh weather, the heating system represents a very significant

portion of the total electric load. MUN employs hot water plants to fulfil its heating requirements. During the harsh weather of winter, the boilers operate at full capacity while during summer the load is decreased and usually half of the boilers are used. The boilers use oil to operate and can consume up to 70000 barrels of oil annually to heat more than 50 buildings across the campus. University uses two meters to monitor the electric load. Meters log data every 15 minutes. We will conduct our analysis on hourly data. Figure 1 is a line plot which shows the trend of electricity consumption for MUN St. John's campus with a data resolution of 1 hour.

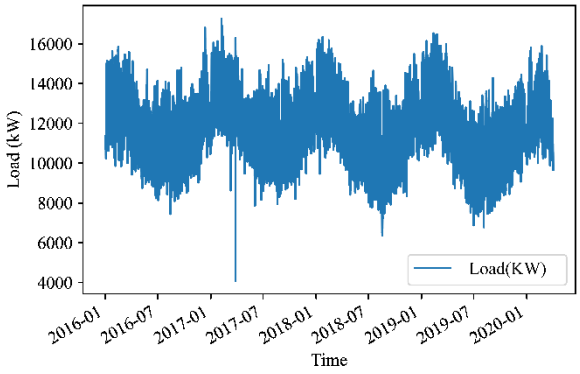


Figure 5: MUN electric load

The average electric load by weekday can be seen in the figure 2 below. The decreased load during weekends is quite evident. There are minor differences across the week but a major change is observed between electricity consumption for weekdays and weekends. Understandably, weekdays have a higher consumption of electricity compared to weekends. Figure 3 is a bar plot similar to Figure 2 but it represents the average electricity consumption for each month. Given the weather of St. John's, it is quite evident that months with higher average temperature (summer & spring) have a lower electricity consumption compared to months with lower ambient temperature (winter & fall).

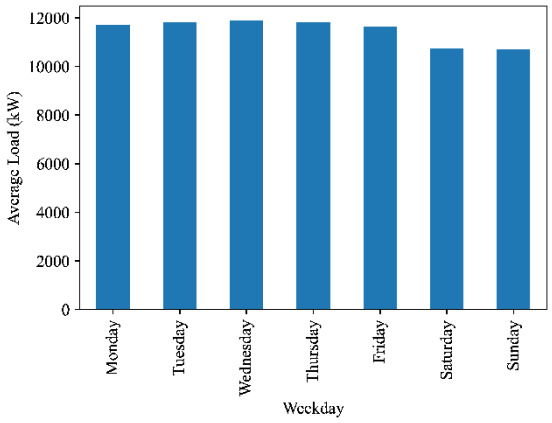


Figure 6: Average electric load by day

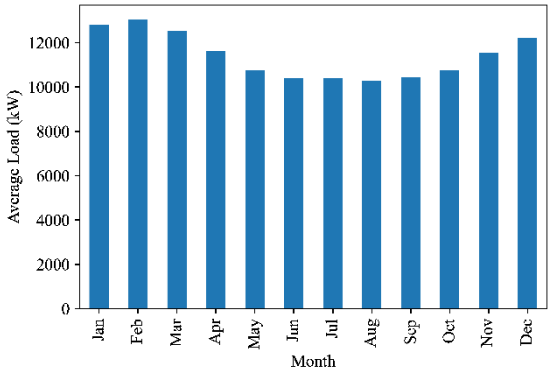


Figure 7: Average electric load by month

2.2.2. Metrological Data

Metrological data has a significant impact on energy consumption so this data will be used in the forecasting strategy. Typically used variables for forecasting include temperature, relative humidity, visibility, cloud cover, rainfall, precipitation, dew point, wind speed, and wind chill [15] [16]. This metrological data was obtained from weather.gc.ca. Our data will be using the following metrological factors [12]:

1. Dry Bulb Temperature: The temperature visible on a thermometer when it is exposed to the air in the absence of moisture and radiation is called dry bulb temperature. It is proportional to the mean kinetic energy of the air molecules.

2. Dew Point: The temperature required at constant pressure to achieve a relative humidity of 100% is called the dew point of the air. It is directly proportional to the moisture in the air.

3. Relative Humidity: At any given temperature, the water mass ratio to air mass represents absolute humidity. Relative humidity is obtained by dividing absolute humidity by the maximum possible humidity at any given temperature.

4. Wind Direction: It represents the direction of the blowing wind. A value of 0 denotes that the wind is calm. A value of 9 represents that wind is blowing from the east while a value of 36 means that the wind is blowing from the north.

5. Wind Speed: It is the speed of the wind. It is measured at a distance of 10m from the ground. In our data, we are using km/h as the measuring unit.

6. Visibility: Visibility is the distance at which an object of tangible size can be observed. In our data, we are using kilometers as the measuring unit.

7. Atmospheric Pressure: The atmospheric pressure is the force exerted per unit area at the height of the measuring station.

After concatenating metrological and electric data, we obtain our complete training dataset. Table 2 shows all the chief features of the complete dataset which is then fed to pre-processing pipeline.

Table 2: Training dataset parameters

Parameter	Value
Data start date	January 2nd 2016
Data end date	March 31st 2020
Data interval	1 hour
Total data points	37221
Features	8

2.3. Background Knowledge

2.3.1. Algorithms

2.3.1.1. Seasonal Autoregressive Integrated Moving Average (SARIMA)

The SARIMA algorithm is the variation of the Autoregressive Integrated Moving Average (ARIMA) algorithm which handles the seasonality in the data. ARIMA itself is a generalized case of the Autoregressive Moving Average (ARMA) algorithm. An ARIMA algorithm the has following components:

Autoregressive Model: Autoregressive model is a common way to model a time series. Equation 1 represents the time series as an autoregressive model. It means that the future values of a time series are linearly related to the previous values in the time series. The order of the time series represents the number of previous values. For example, if a time series has an autoregressive order of p, then it means that the any value in that time series can be written as a linear combination of previous p values. Such a time series would be called an AR(p) time series [17]:

$$y_t = \delta + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \epsilon_t \quad (1)$$

Here ϵ_t represents the white noise Here y_t can be seen as a regression p lagging values hence it is an AR(p) model.

Moving Average Model: Moving average model is a way to model a time series as a linear combination of past forecasting errors. Equation 2 represents the time series as a moving average model. A moving average model of order q can be represented as MA(q). Each value in such a time series can be thought of as a moving average of past q forecasting errors. This can be observed in equation 2.

$$y_t = \mu + \epsilon_t - \varepsilon_{t-1}\theta_1 - \cdots - \varepsilon_{t-q}\theta_q \quad (2)$$

Here μ represents the mean. (ϵ_t,… ε_{t-q}) represent the past errors while ($\theta_1 \ldots \theta_q$)

An ARIMA Model is obtained by combing these two models with differencing to induce stationarity on the time series. The order of differencing is represented by the parameter d. Hence the ARIMA(p,d,q) model is represented by three parameters i.e. p,d, and q, each representing the order of Auto regression, differencing, and moving average of the model.

2.3.1.2. Long Short-Term Memory Recurrent Neural Network

Recurrent neural networks are a type of artificial neural network which are capable of handling sequential data. A simple RNN network takes the input value of the sequence and combines it with the hidden state from the previous timestamp and uses an activation function to generate the hidden state for the next timestamp. RNNs are susceptible to exploding and vanishing gradients. To resolve this issue, RNN cells are modified using various gates. Long Short Term Memory (LSTM) [18] artificial neural network consists of LSTM cells and each cell has an input gate, output gate

and forget gate. Through these gates, an LSTM cell can forget or pay attention to different parts of a sequential input [19].

Forget Gate: The forget gate (f_t) determines what data needs to be forgotten from a network's long-term memory /cell state (c_t) based on the new input x_t and hidden state obtained from the previous time step (h_{t-1}).

$$f_t = sigmoid\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \tag{3}$$

Input Gate: The input gate (I_t) determines which information needs to be added to the network based on the current input and hidden state from the previous step. The gate also generates a candidate hidden state ($\hat{c}_t$) by using a tangent hyperbolic activation function. Finally, the cell state at time t represented by c_t is calculated as well:

$$I_t = sigmoid(W_{Ix}x_t + W_{Ih}h_{t-1} + b_I) \tag{4}$$

$$\hat{c}_t = tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \tag{5}$$

$$c_t = f_t.c_{t-1} + \hat{c}_t.I_t \tag{6}$$

Output Gate: The output gate (O_t) generates the hidden state (h_t) which is passed over to the next LSTM cell.

$$O_t = sigmoid(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{7}$$

$$h_t = O_t \otimes tanh(c_t) \tag{8}$$

All the variables represented by W and b represent weights and biases (respectively) learned by the network on training. The $\otimes$ symbol represents pointwise multiplication.

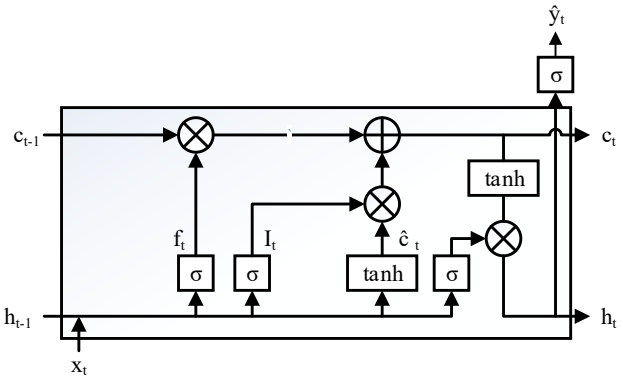


Figure 8: A basic LSTM unit

2.3.1.3. Gated Recurrent Unit

Gated Recurrent Units are a type of Recurrent neural network in which a single unit has 2 gates (reset gate and update gate). [20]. GRUs also do not have cell states and instead, they use hidden states to pass the information to the next time step. Compared to LSTMs, this network is simpler and takes less time to train.

The basic description of the gates is given below:

Reset Gate: The reset gate decides how much of the previous information needs to be forgotten.

$$r_t = sigmoid(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \tag{9}$$

Update Gate: The update gate in the GRU decides how much of the information from previous time steps need to be passed to the next blocks.

$$z_t = sigmoid(W_{zx}x_t + W_{zh}h_{t-1} + b_z) \tag{10}$$

First, the reset gate is used to create a candidate vector ($\hat{h}_t$) following:

$$\hat{h}_t = tanh(W_z x_t + W_h(r_t \otimes h_{t-1}) + b_{ht}) \tag{11}$$

Finally, the update gate is used to generate the hidden state as follows:

$$h_t = z_t \otimes \hat{h}_t + (1 - z_t) \otimes h_{t-1} \quad (12)$$

2.3.1.3. Bi-Directional Recurrent Neural Networks

Bi-directional RNNs process the data in both directions using two separated hidden layers before feeding it to

the same output layer [21]. This architecture can be used for both, LSTM Recurrent Neural Networks (Bi-Directional LSTM) and Gated Recurrent Networks (Bi-Directional GRU).

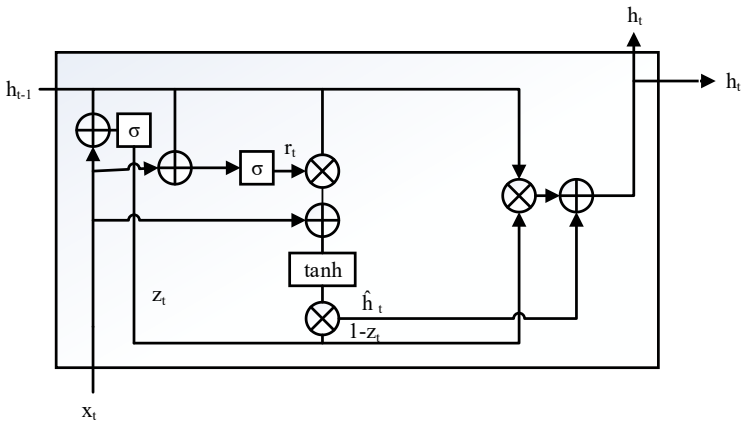


Figure 9: A basic GRU unit

2.3.1.4. Convolutional Neural Network

Convolutional neural networks (also referred to as Shift Invariant Artificial neural networks) consist of filters that slide along the input features to create feature maps. Typical convolutional neural network architectures usually consist of convolutional layers with a pooling layer between two convolutional layers and finally dense or fully connected layers.

A convolutional layer performs convolutional operations using filters or kernels to generate feature maps. Important parameters include the stride and filter size. Each neuron of a feature map is connected to neurons in the previous layer referred to as the receptive field of that neuron.

The activation value of a neuron located at (i,j) location in the kth feature map of layer l is obtained by passing the feature map through an activation function which adds the ability to learn nonlinear patterns to a network. The activation value can be represented as:

$$a_{i,j,k}^{l} = Activation(W_k^{l^T} x_{i,j}^{l} + B_h^{l}) \quad (13)$$

Here, $x_{i,j}^{l}$ represents the input while $W_k^{l^T}$ and B_h^{l} are weights and biases. Activation functions used could be ReLU, tanh, or sigmoid.

Pooling operation is used to sample the feature map. It introduces shift invariance to the network. Max pooling [21] and average pooling [22] are the typical pooling operations employed in convolutional networks. Pooling layers are placed between 2 convolutional layers.

A network usually consists of a number of convolutional layers with pooling layers in between. The earlier layers in a network learn low-level features while the layers after that learn more abstract or higher-level features. In case when a convolutional neural network is used for image classification, the earlier layers will learn features such as lines, edges, and corners while later layers would learn more abstract features which are built on top of previous low-level features.

After several convolutional and pooling layers, one or more fully connected layers are added at the end of the network. This fully connected layer performs high-level reasoning, although it can be replaced by a 1x1 convolution layer [23]. This layer is then connected to the output layer. A loss function is generated using the output of this layer and true outputs from the training dataset. This loss function is then minimized to obtain network parameters (weights and bias terms). A popular optimization algorithm used for minimizing loss function is stochastic gradient descent [24].

2.3.1.5. Sequence to Sequence (Seq2Seq) Model

As the name implies, this model converts an input sequence to an output sequence. The length of both sequences can be different. The model consists of two recurrent neural networks. The input is fed to a network called encoder network while we get an output from a model called the decoder network.

The encoder model is a recurrent neural network that takes an input sequence. At each time step in the recurrent neural network, the GRU or LSTM block takes the input of that time step and produces an output and a hidden state. The hidden state is passed along to the next LSTM/GRU block which uses it along with input from that time step to generate a hidden state for the next block. The hidden state generated by the last LSTM/GRU block is used as the initial hidden state for the decoder part. It is called the context vector.

The decoder is an RNN consisting of either GRUs or LSTMs. The first decoder block uses a hidden state to generate output value and a hidden state for the next LSTM/GRU block. Each block will use the output and hidden state from the previous block.

2.3.1.6. Sequence to Sequence (Seq2Seq) Model with Attention

The motivation for using the attention mechanism stems from the underperformance of the basic encoder-decoder model for long sequences. This is caused by the use of a fixed-length context vector.

When using the attention mechanism, an alignment score is calculated by using a vector consisting of all the encoder hidden state vectors as well as the output generated from the decoder block of the previous time step. Using alignment scores, attention weights are generated using the

Softmax Activation function and finally, a context vector is generated by using attention weights and encoder hidden states.

2.3.1.7. Proposed Custom Architecture:

The proposed custom architecture consists of a 1D convolution neural network (CNN). The 1D CNN provides temporal features to the next model which is the encoder-decoder model. The encoder-decoder model also has an attention mechanism. The attention mechanism helps the encoder-decoder model to pay attention to more important patterns in the extracted temporal features. Both the encoder and decoder consist of GRU units. After the encoder-decoder model, a fully connected layer is used. The 24 output cells of the fully connected layer are the predicted outputs representing the predicted hourly load profile of the model.

2.3.2. Evaluation Metrics

2.3.2.1. Mean Absolute Error

Mean absolute error is the average of the absolute difference between a predicted/forecasted value and ground truth. It can be written as:

$$Mean\ Absolute\ Error = \frac{\sum_{i=1}^{N} |P_i - G_i|}{N} \tag{14}$$

Here, P and G are predictions and ground values respectively. N is the number of values in the dataset. If an estimator is more accurate, it would result in a lower MAE. An inaccurate estimator would result in a higher MAE. One major drawback of the MAE is that it is scale dependent (it uses same scale as that of the dataset). Being scale dependent means that it cannot be used to compare algorithm performance across different datasets.

2.3.2.2. Mean Squared Error

Mean Squared Error is the average of the squared difference between the predicted values and ground truth. This evaluation metric is based on Euclidean distance. It can be written as:

$$Mean\ Squared\ Error = \frac{\sum_{i=1}^{N}(P_i - G_i)^2}{N} \tag{15}$$

This evaluation metric is based on Euclidian distance. An accurate estimator would have a lower MSE while an inaccurate estimator would have a higher MSE. Compared to MAE, MSE is more sensitive to outliers since errors are squared. MSE is scale dependent so it should only be used to compare algorithm performance for same dataset.

2.3.2.3. R2 Score

It is also referred to as the coefficient of determination. It can have a maximum value of 1which would indicate that the model has predicted every ground truth value correctly. It can also have negative values with no limit since an estimator can be arbitrarily worse. If the ground truth is non-constant and the model predicts a constant mean value, then the R2 value would be 0. The formula for this metric is given by the relationship below:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(P_i - G_i)^2}{\sum_{i=1}^{N}(G_i - M)^2} \tag{16}$$

Here M is the mean of all the observed values and is defined as:

$$M = \sum_{i=1}^{N}(G_i) \tag{17}$$

It is evident from 15 that R^2 would have a value of 1 for a model with all the predicted values equal to ground truths as $(P_i - G_i)^2$ would be 0. For a baseline model that only predicts mean

values, $\frac{\sum_{i=1}^{N}(P_i-G_i)^2}{\sum_{i=1}^{N}(G_i-M)^2}$ would be 1, and the overall value would be 0. Adding high number of features to the model can increase the R^2 score of the model even though it might have lower prediction power. Similarly, a model with non-random residuals can have high score while still being fairly inaccurate. So it is always advisable to use coefficient of determination with other metrics such as mean squared error, mean absolute error and mean absolute percentage error.

2.3.2.4. Mean Absolute Percentage Error (MAPE):

Mean Absolute Percentage Error (MAPE) is an evaluation metric that is defined as:

$$MAPE = \frac{100}{n}\sum_{i=1}^{N}\left|\frac{G_i - P_i}{G_i}\right| \tag{18}$$

This metric is independent of the scale of data and can be used to compare an algorithm across different time series. Although if a time series has zero or near zero values, this evaluation metric becomes infinite. It also penalizes negative and positive errors asymmetrically. A more accurate algorithm would have lower MAPE compared to a less accurate model.

2.3. Proposed Solution

2.4.1 Methodology

2.4.1.1. Data Cleaning

Data cleaning and data preprocessing are divided into two different steps as they represent different set of procedures used on the dataset. Typically, data cleaning consists of initial set of methods while data preprocessing occurs downstream of data cleaning and involves more advanced techniques.

Data cleaning is performed on the data to make it consistent and remove any discrepancies. Some of the major data cleaning procedures involve:

a. Removal of outliers which could be extremely large values, extremely small values or non-plausible values (such as negative values for temperature in Kelvin). They might be a part of the dataset because of a number of reasons including faulty instrument or some errors in data collection software.

b. Removal of any NA (Not available) values from the dataset. These values represent gaps or breaks in the data.

The outliers are detected based on inter quartile range (IQR). These outliers are then represented as NA values. These NA values along with any gaps in the dataset are filled using forward filling method. The method involves filling any breaks or outliers in the data with values from previous timestamp. Data was analyzed again to find any missing values and outliers. None were found after the procedure.

2.4.1.2. Data Preprocessing

In this step, data standardization is performed. Data Standardization is carried out for each feature separately. This process helps the model converge faster. It is also called Z-Score Normalization. It is more robust to any outliers in the data compared to data normalization. Equation 19 shows the basic formula for the standardization of a single feature.

$$x_{normalized} = \frac{x - M}{\alpha} \tag{19}$$

Here M is the meanwhile α is the standard deviation.

Secondly, a sliding windowing function is implemented. Data windowing is done to create input-output pairs from a continuous time series. Since we are using the hourly input, we will be dividing the time series into continuous slices while striding 24-time steps which correspond to one complete day. The output of the network would be an hourly prediction of one complete day.

After data windowing, we divide the dataset into 3 portions. These are the training set, validation set, and test set. 75% of the data is used for training, 15% for validation while remaining 10% of the data is used for testing.

2.4.1.3. Network Architecture

The networks used for training include RNNs (LSTM, GRU), a 1D CNN + Encoder-Decoder model as well as a custom architecture consisting of a 1D CNN followed by an encoder-decoder model with attention. The difference between the above 2 models is the attention mechanism. Following this, a fully connected layer generates the desired output. The complete proposed architecture is shown in Figure 8.

Model consists of three main parts. The first part is the 1D CNN model which aims to embed the spatial features in the model. These features include metrological features present in the original data. The second part is the RNN network based on LSTM model. This portion of the model is concerned with temporal features of the data. The dataset is fed to network in the form of windows so LSTM model tries to capture the temporal patterns and features in the data. Finally, we have the attention mechanism which is part of the temporal network. Attention mechanism is typically used for capturing long range dependencies. Here, this mechanism would capture long range temporal features would could be missed by a simple RNN or LSTM.

2.4.1.4. Training

Finally, the training set is used to train the data. Huber loss is used along with Adams optimizer.

$$Huber\ loss = \begin{cases} \frac{1}{2}(P-G)^2 & for\ (P-G) \leq \delta \\ \delta\ .\left(|P=G| - \frac{1}{2}\delta\right), & otherwise \end{cases} \quad (20)$$

Equation 20 defines the Huber loss. This loss function is non-linear (quadratic) for small values of the residual (P-G) and linear for larger residuals. As a result, this function combines the sensitivity of Mean Squared Error (MSE) with the robustness of Mean Absolute Error (MAE).

Each of the algorithms is trained up to 100 epochs. An early stopping routine is implemented for a more efficient training process and to save time. Validation loss is monitored and the best results are saved based on its minimum value. Finally Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and R2 values are tabulated on the test data to evaluate the model performance. Table 4 shows training and inference times on GTX 970m and i7 4720 HQ with 16 GB Memory for the model with best performance.

Table 3: Training Hyperparameters

Hyperparameters	**Values**
Epochs	100
Loss Function	Huber Loss
Batch Size	128
Optimizer	Adams Optimizer
Early Stopping Patience	20
Early Stopping Parameter	Validation Loss
Input Features	10
Output Window Width	24
Input Window Width	672

Table 4: Training and Inference time

Parameter	Value (seconds)
Training time	257.4
Inference time	.21

2.4.1.5. Results

a. Model Comparison

Data cleaning and preprocessing were done in Python 3.9 using pandas and Numpy. Keras was used to create deep learning models and compare various evaluation metrics. We can see graphs of electric load prediction by all the models. By looking at the prediction graphs in Figure 9, it can be seen that the models have learned the daily electric usage profile quite well. The proposed model has the lowest MAPE, MSE, and MAE. The proposed model also shows the highest R2 score. We also compare this to some of the other forecasting models found in literature which include Rational Quadratic Gaussian Process Regression (RQ-GPR), which is used in [12] for interpolation however we are interested in forecasting future values while training the model on past information. We compare our model with this algorithm. Similarly model proposed in [14] called Sequence to Sequence + Luong Attention is also trained and compared with our trained algorithm. RQ-GPR has the weakest performance compared to all the other algorithms as it is a conventional machine learning techniques compared to rest of the methods which employ deep learning. GRU based RNN is the second weakest performer followed by LSTM. Using 1D CNN + Sequence to Sequence model significantly improves the performance. Adding Luong attention as mentioned in [14] improves the performance further but the best performance is achieved by employing the proposed method which shows lower MAPE, MSE and MAE. It also shows a higher R2 score which corresponds to better forecasting potential compared to rest of the algorithms.

b. Input Horizon Optimization

We have obtained all the results for an input window of 672 time steps, 672 time steps correspond to 28 days (24 x 28 = 672). Since the model uses a 1D CNN Network, the number of learning variables is independent of the input size of each training example. This allows us to further optimize the model by varying the input horizon. Using a larger input window would allow the network to capture more dependencies among variables but it also reduces the number of training samples we can have. This can lead to overfitting as a more complex model requires more training examples. The use of a 1D CNN-based network allows the optimization of the input horizon without increasing the number of trainable parameters in the network. Another similar architecture with an LSTM-based encoder-decoder is also used for comparison. The input window was changed from 24 to 672 with an interval of 24. This corresponds to the input horizon ranging from 1 to 28 days. The results are tabulated in Table 7. The evaluation metrics used are MAPE, MAE, MSE, and R2 Score. It can be observed from the table below that the GRU-based proposed model performs the best overall with an 8-day (or 192 time steps) input horizon.

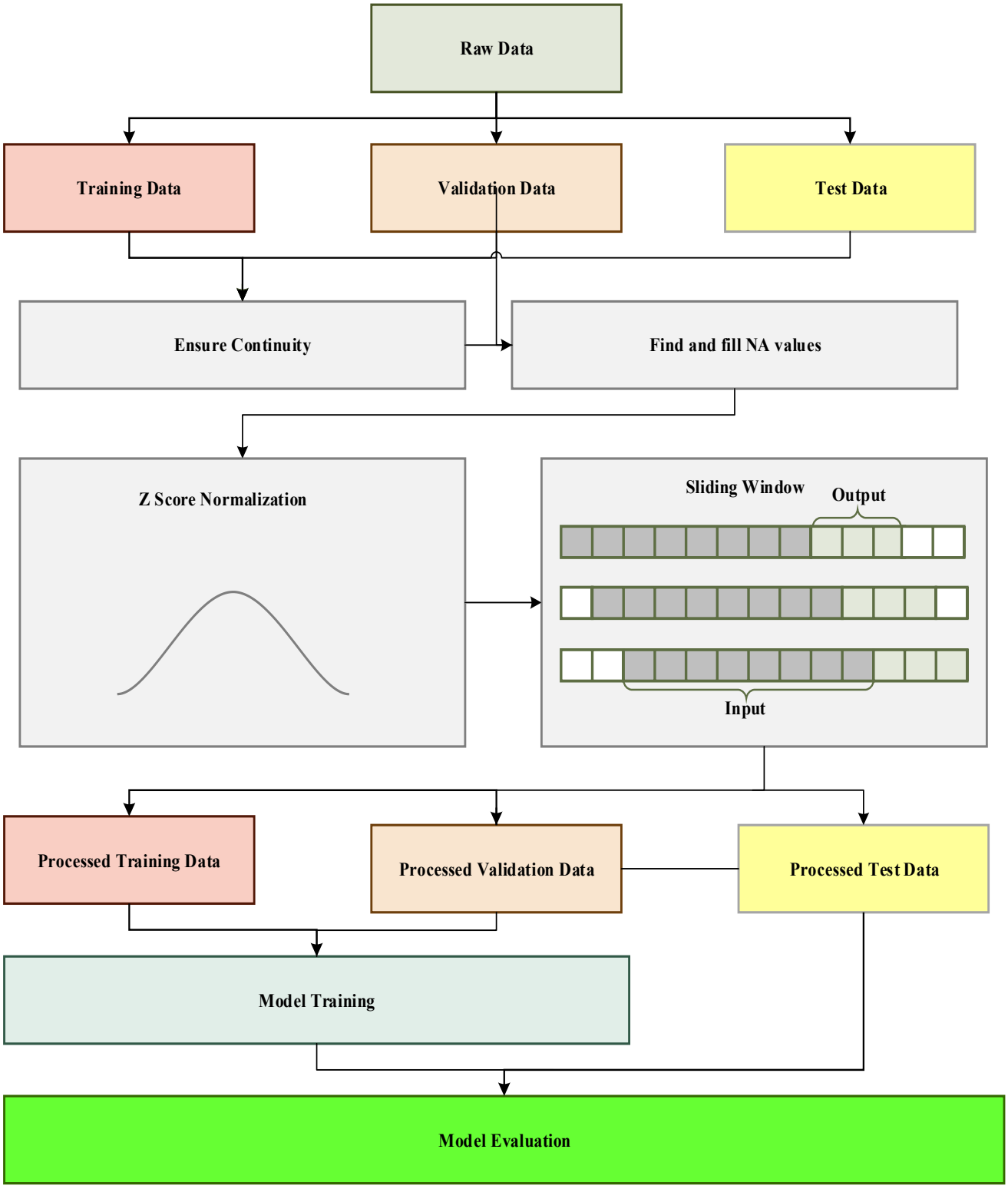


Figure 10: Model Pipeline

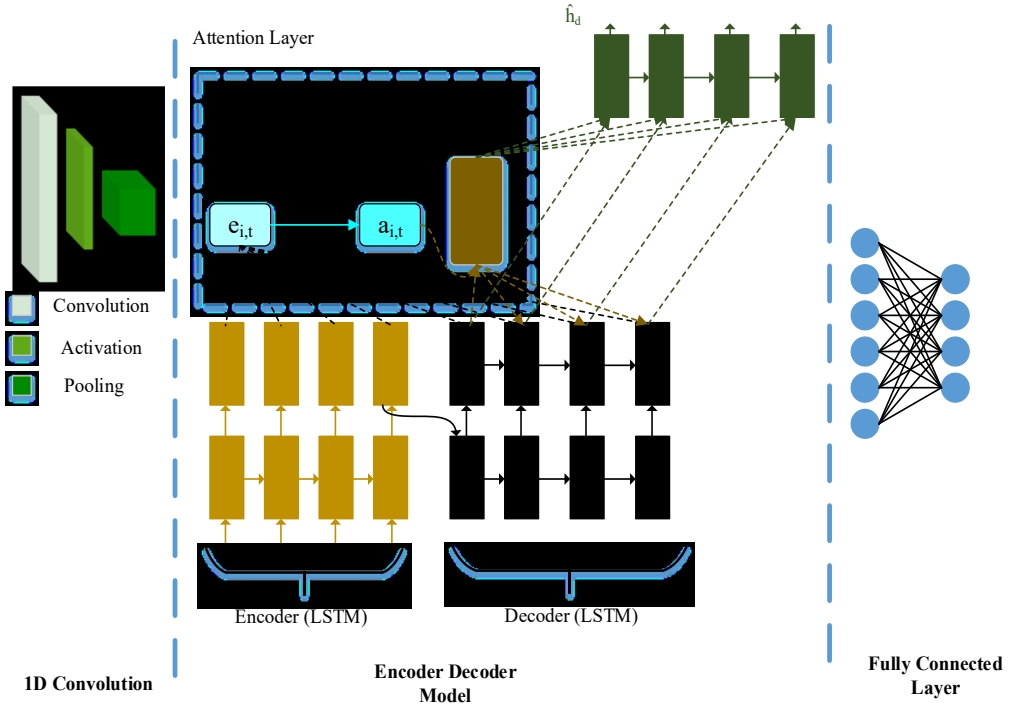
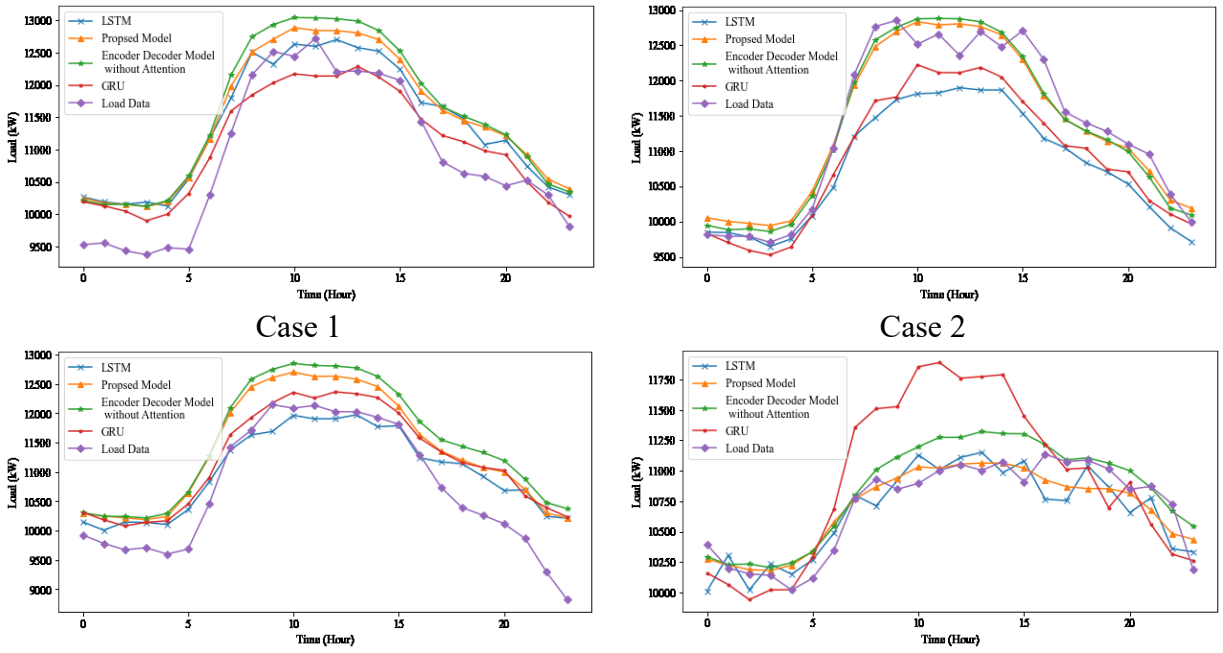


Figure 11: Proposed Network Architecture



Case 1 Case 2

Case 3 Case 4

Figure 12: Predictions for various days using the algorithms

Table 5: Results for various algorithms

Algorithm/Metric	MAE (kW)	R^2 Score	MSE (kW^2)	MAPE (% age)
LSTM	442.94	0.77	340026.33	3.64
GRU	493.72	0.739	386019.62	4.05
1D CNN + Encoder Decoder	423.326	0.8011	294162.21	3.51
RQ-GPR [12]	450.21	0.71	401264.54	4.98
Sequence to Sequence + Luong Attention [14]	410.11	0.795	298765.45	3.40
Proposed Network Architecture	407.308	0.805	287346.33	3.37

c. Robustness Testing

To ensure that the trained algorithm is sufficiently robust and invariant to perturbations in the data, noise is added to the testing set, and evaluation metrics are reevaluated to check for any major changes. Gaussian White Noise is added to the load variable in the data with 0 mean and different standard deviations.

Table 6: Robustness testing with different cases

Case	Study Details	Test Parameters	Evaluation Metrics			
			% ΔMAE	% ΔMSE	% ΔMAPE	% ΔR2
1	**[0,50]**	**Average**	1.241453	4.06674	0.417076	0.89951
		Maximum	1.863907	5.261097	1.003151	1.157584
		Standard Deviation	0.283457	0.509466	0.285956	0.112588
2	**[0,75]**	**Average**	1.85939	5.20309	1.051538	1.111195

		Maximum	2.71684	6.87143	1.879937	1.437032
		Standard Deviation	0.393231	0.663161	0.397011	0.13982
3	**[0,100]**	**Average**	2.796778	6.779848	2.024935	1.405992
		Maximum	4.115265	9.339956	3.419123	1.944015
		Standard Deviation	0.546864	0.948765	0.559865	0.203264
4	**[0,125]**	**Average**	4.02232	8.966153	3.297807	1.807394
		Maximum	5.781688	12.74125	5.005656	2.662423
		Standard Deviation	0.624861	1.205047	0.629848	0.268467

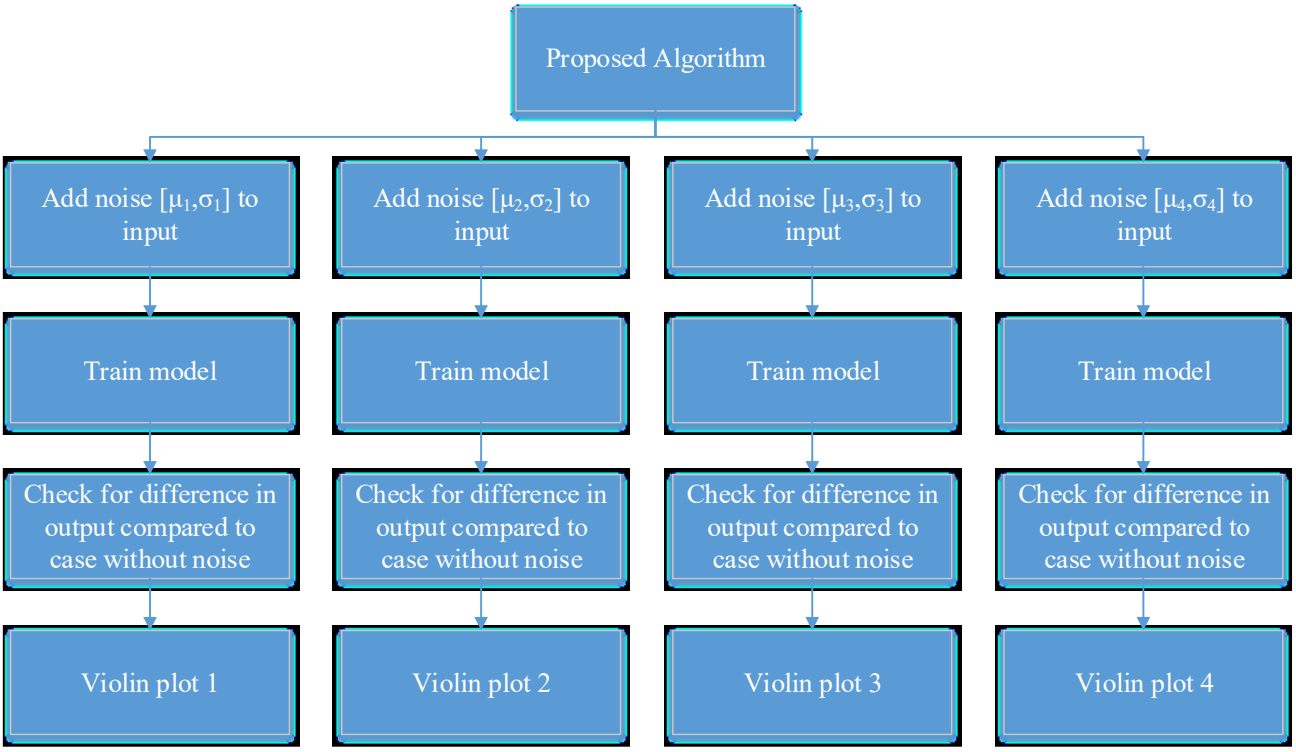


Figure 13: Basic robustness testing pipeline

Cases are studied with over 100 tests for each case. A case is defined as [μ,σ], where μ is the meanwhile σ is the standard deviation of the normal distribution of the added noise.

Figure 9 shows the basic robustness testing pipeline while Figure 11 shows the probability distribution functions of the 4 cases studied of added noise. These results obtained after conducting over 100 tests are tabulated in Table 5. The highest percentage deviation was observed for MSE which was close to 12%, this can be explained by the fact that any outliers in the added noise would have affected the score in a more significant manner compared to MAE, MAPE, and the R2 score. The rest of the metrics show a maximum deviation of around 6%. The average percentage deviation of the metrics scores also does not exceed 9% (for MSE). Overall the model is fairly unaffected by the added noise.

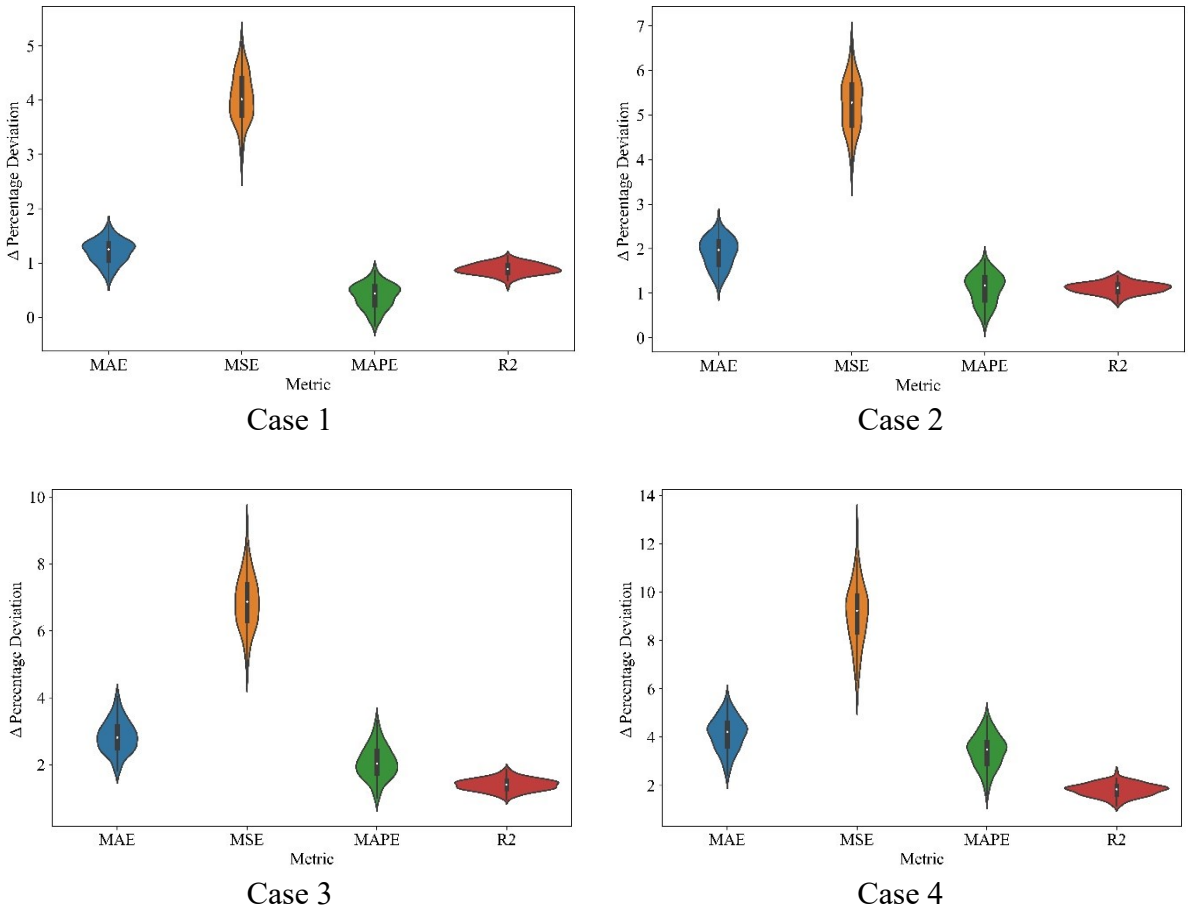


Figure 14: Violin plot showing variation in evaluation metrics for each case

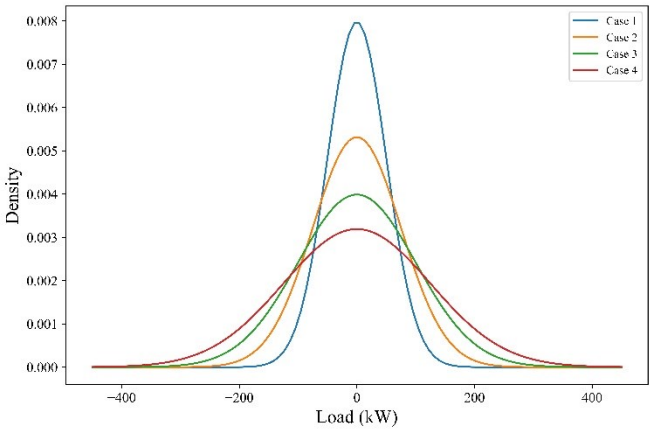


Figure 15: Probability distribution functions of added noise

2.5. Conclusion

The model is trained on real-world data and shows a higher R2 Score and lower MSE, MAE, and MAPE compared to other conventional LSTM-based and GRU-based models. The difference in performance achieved by the attention mechanism is also evident as the model with attention performs better (Higher R2, lower MAE, MSE, and MAPE) compared to the similar model without attention. An analysis was also carried out to find optimal input window sizes while also changing the basic RNN block from GRU to LSTM. The input window size of 8-day or 192 time steps with a GRU-based encoder decoder performs the best with the lowest MAE, MAPE, and the highest R2 Score. Robustness testing of the proposed method was also conducted and shows that the proposed model is unaffected by the added perturbations to the data.

Table 7: Different input horizons

Architecture	Train Steps	MSE kW^2	MAE kW	R2 Score	MAPE (*100 %)
GRU	24	536352.076	541.330	0.663	0.045
	48	464480.438	501.336	0.708	0.043
	.	.	.	.	.
	.	.	.	.	.
	168	328631.031	435.952	0.786	0.037
	192	**284889.734**	**408.822**	**0.814**	**0.034**
	216	287705.298	412.394	0.812	0.035
	.	.		.	.
	.	.	.	.	.
	.	.	.	.	.
	648	364983.889	458.885	0.754	0.038
	672	385770.860	471.314	0.739	0.039

LSTM	24	508925.840	533.137	0.680	0.045
	48	488299.198	532.989	0.693	0.045
	.	.	.	.	.
	.	.	.	.	.
	168	344335.384	448.178	0.776	0.038
	192	356604.350	465.067	0.767	0.039
	216	345513.005	452.298	0.775	0.038
	240	**302698.183**	**421.951**	**0.802**	**0.035**
	264	400005.317	477.321	0.738	0.040
	.	.	.	.	.
	.	.	.	.	.
	648	413153.544	505.153	0.721	0.042
	672	368893.385	473.734	0.751	0.039

2.6. References

[1] A. A. Mamun, M. Sohel, N. Mohammad, M. S. Haque Sunny, D. R. Dipta, E. Hossain, A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models, IEEE Access 8 (2020) 134911–134939. https://doi.org/10.1109/ACCESS.2020.3010702.

[2] M. Bashari, A. Rahimi-Kian, Forecasting Electric Load by Aggregating Meteorological and History-based Deep Learning Modules, in: 2020 IEEE Power Energy Soc. Gen. Meet. PESGM, IEEE, Montreal, QC, Canada, 2020: pp. 1–5. https://doi.org/10.1109/PESGM41954.2020.9282124.

[3] W. Zhang, Q. Chen, J. Yan, S. Zhang, J. Xu, A novel asynchronous deep reinforcement learning model with adaptive early forecasting method and reward incentive mechanism for short-term load forecasting, Energy 236 (2021) 121492. https://doi.org/10.1016/j.energy.2021.121492.

[4] E. Mocanu, P.H. Nguyen, M. Gibescu, W.L. Kling, Deep learning for estimating building energy consumption, Sustain. Energy Grids Netw. 6 (2016) 91–99. https://doi.org/10.1016/j.segan.2016.02.005.

[5] Y. Lin, H. Luo, D. Wang, H. Guo, K. Zhu, An Ensemble Model Based on Machine Learning Methods and Data Preprocessing for Short-Term Electric Load Forecasting, Energies 10 (2017). https://doi.org/10.3390/en10081186.

[6] M.A. Hammad, B. Jereb, B. Rosi, D. Dragan, Methods and models for electric load forecasting: a comprehensive review, Logist. Supply Chain Sustain. Glob. Chall. 11 (2020) 51–76.

[7] F.M. Bianchi, E. Maiorino, M.C. Kampffmeyer, A. Rizzi, R. Jenssen, An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting, 2017. https://doi.org/10.1007/978-3-319-70338-1.
[8] Jian Zheng, Cencen Xu, Ziang Zhang, Xiaohua Li, Electric load forecasting in smart grids using Long-Short-Term-Memory based Recurrent Neural Network, in: 2017 51st Annu. Conf. Inf. Sci. Syst. CISS, 2017: pp. 1–6. https://doi.org/10.1109/CISS.2017.7926112.
[9] X. Wang, W. Tian, Z. Liao, Statistical comparison between SARIMA and ANN's performance for surface water quality time series prediction, Environ. Sci. Pollut. Res. 28 (2021) 33531–33544. https://doi.org/10.1007/s11356-021-13086-3.
[10] S. Muzaffar, A. Afshari, Short-Term Load Forecasts Using LSTM Networks, Energy Procedia 158 (2019) 2922–2927. https://doi.org/10.1016/j.egypro.2019.01.952.
[11] M. Jain, T. AlSkaif, S. Dev, Are deep learning models more effective against traditional models for load demand forecasting?, in: 2022 Int. Conf. Smart Energy Syst. Technol. SEST, IEEE, Eindhoven, Netherlands, 2022: pp. 1–6. https://doi.org/10.1109/SEST53650.2022.9898424.
[12] M. Madhukumar, A. Sebastian, X. Liang, M. Jamil, M.N.S.K. Shabbir, Regression Model-Based Short-Term Load Forecasting for University Campus Load, IEEE Access 10 (2022) 8891–8905. https://doi.org/10.1109/ACCESS.2022.3144206.
[13] L. Yi, J. Zhu, J. Liu, H. Sun, B. Liu, Multi-level collaborative short-term load forecasting, in: 2022 25th Int. Conf. Electr. Mach. Syst. ICEMS, IEEE, Chiang Mai, Thailand, 2022: pp. 1–5. https://doi.org/10.1109/ICEMS56177.2022.9983442.
[14] M. Aouad, H. Hajj, K. Shaban, R.A. Jabr, W. El-Hajj, A CNN-Sequence-to-Sequence network with attention for residential short-term load forecasting, Electr. Power Syst. Res. 211 (2022) 108152. https://doi.org/10.1016/j.epsr.2022.108152.
[15] M. Jawad, M. S. A. Nadeem, S. -O. Shim, I. R. Khan, A. Shaheen, N. Habib, L. Hussain, W. Aziz, Machine Learning Based Cost Effective Electricity Load Forecasting Model Using Correlated Meteorological Parameters, IEEE Access 8 (2020) 146847–146864. https://doi.org/10.1109/ACCESS.2020.3014086.
[16] R. Schaeffer, A.S. Szklo, A.F. Pereira de Lucena, B.S. Moreira Cesar Borba, L.P. Pupo Nogueira, F.P. Fleming, A. Troccoli, M. Harrison, M.S. Boulahya, Energy sector vulnerability to climate change: A review, Energy 38 (2012) 1–12. https://doi.org/10.1016/j.energy.2011.11.056.
[17] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, G.M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.
[18] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Comput. 9 (1997) 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
[19] P.T. Yamak, L. Yujian, P.K. Gadosey, A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting, in: Proc. 2019 2nd Int. Conf. Algorithms Comput. Artif. Intell., ACM, Sanya China, 2019: pp. 49–55. https://doi.org/10.1145/3377713.3377722.
[20] Q. Tao, F. Liu, Y. Li, D. Sidorov, Air Pollution Forecasting Using a Deep Learning Model Based on 1D Convnets and Bidirectional GRU, IEEE Access 7 (2019) 76690–76698. https://doi.org/10.1109/ACCESS.2019.2921578.
[21] A. Graves, A. Mohamed, G. Hinton, Speech Recognition with Deep Recurrent Neural Networks, (2013). http://arxiv.org/abs/1303.5778 (accessed November 14, 2022).
[22] Y.-L. Boureau, J. Ponce, Y. LeCun, A Theoretical Analysis of Feature Pooling in Visual Recognition, (n.d.).

[23] T. Wang, D.J. Wu, A. Coates, A.Y. Ng, End-to-end text recognition with convolutional neural networks, in: Proc. 21st Int. Conf. Pattern Recognit. ICPR2012, 2012: pp. 3304–3308.
[24] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recognit. 77 (2018) 354–377. https://doi.org/10.1016/j.patcog.2017.10.013.
[25] R.G.J. Wijnhoven, P.H.N. de With, Fast Training of Object Detection Using Stochastic Gradient Descent, in: 2010 20th Int. Conf. Pattern Recognit., 2010: pp. 424–427. https://doi.org/10.1109/ICPR.2010.112.

Chapter 3: A Novel Multi-Task Learning Based Approach to Multi-Energy System Load Forecasting

The chapter is currently under review in IEEE Open Access Journal of Power and Energy. A revision for the manuscript was requested on August 20th.

Abstract

Multi-Energy Systems (MES) allow optimal interactions between different energy sources. Accurate load forecasting for such intricate systems would greatly enhance the performance and economic incentive to employ them. This article proposes a state-of-the-art deep learning based architecture to forecast multiple loads. The algorithm utilizes load correlations to select optimal input parameters. These optimal inputs are fed to D-TCNet (Deep – Temporal Convolution Network). This network uses multi-layer perceptrons (MLP) to encode the spatial relationship among exogenous variables which is fed to a Temporal Convolutional Network (TCN). The TCN resolves temporal information in the multi-load time series which is used for forecasting these loads for fixed output horizon. The proposed novel method is used on the energy consumption data for multi energy system of University of Austin Tempe Campus. The proposed method shows improved performance across all three energy types as well as all four seasons.

Index Terms— Multi-Energy Systems, Multi-Task learning, Temporal Convolutional Network

3.1. Introduction

Multi-Energy Systems (MES) offer a unique solution for overall efficient energy usage by employing interaction between energy sources. Compared to conventional independent energy systems, MES offer better performance in technical, environmental and economic terms [1]. MES achieve a much better balance between energy supply and demand using the interactions [2]. Accurate load forecasting algorithms would prove extremely beneficial in operation and maintenance of such systems. However, the interactivity of energy sources gives MES an inherent complexity not found in independent systems. The relationships between electric, cooling and heating loads vary round the year and the interactions between these loads gives rise to complex relationships which further increase the challenge associated with MES load forecasting.

In a conventional energy system, future energy or load requirements are highly dependent on metrological features and past energy consumption. In MES, it is very important to quantify the coupling relationships among different loads. Researchers have also used Pearson correlation analysis to quantify correlations before feeding data to the prediction algorithm [3] [4]. Pearson correlation analysis only quantifies linear dependencies among the data and fails to account for intricate non-linear relationships [5]. In the current literature Maximum Information Coefficient (MIC) has been used to mitigate this issue [6]. In this article we will be using Distance Correlation (Dcor) for coupling analysis. Although, both the MIC and Dcor can be used for non-linear relationships, Dcor exhibits higher statistical power and ordering of dependencies is preserved

even with the introduction of noise in the data [7]. On top of that Dcor is simpler to calculate and is not an approximation like MIC.

3.2 Literature Review

Deep Learning is a state of the art technology capable to modeling complex non-linear functions using hierarchical structure. This structure allows these models to extract high level features from low level mappings. Complexity of the model depends on the number of hidden layers in a hierarchy [8]. Deep learning has revolutionized multiple fields including image processing, natural language processing and time-series forecasting.

In case of time-series forecasting, the hidden layer based structure allows the deep learning models to extract complex spatial and temporal features from the time-series. These features are used by downstream layers to identify key patterns which are used for forecasting. In case of integrated energy systems, another factor in the form of complex coupling relationships emerge due to inherent interplay among energy variables (heating, cooling and electricity). Researches have used multiple forecasting techniques to untangle this challenge.

In [9], researches create a digital twin for a IES and used deep neural network (DNN) for load forecasting. In [10], the researchers use MIC to characterize load correlations and utilize stacking based ensemble learning consisting of Random Forests, Gradient Boosting Decision Trees and Support Vector Regression. Researchers use multi-task learning (MTL) with bootstrapping, improved slap swarm algorithm and multi-kernel extreme learning machine for load forecasting of integrated energy system in [11].

Authors used Bi-directional Long Short Term Memory (Bi-LSTM) Network for short-term load forecasting but only considered using MIC for coupling relationship [6].

Temporal Convolutional Network (TCN) [12] is an architecture utilizing casual and dilated convolutions. It shows strong capability for time-series forecasting and there has not been enough research in employing this network for short-term load forecasting while employing multi-task learning for multi-energy systems. Combining the temporal convolution network with multi-layer perceptron would allow the model to learn complex spatiotemporal features.

Table 1 consolidates all the current literature on multi energy load forecasting.

The main purpose of the work is summarized below as:

1. Investigate the coupling relationship between cooling, heating and electric loads using distance correlation across 4 seasons of the year.
2. Propose a novel multi-task learning based approach for short-term load forecasting. First, distance correlation analysis is used to investigate coupling relationship between multiple load time series. Appropriate input variable selection is carried out based on the analysis. The selected load data along with exogenous variables are used as input to D-TCNet which is a TCN based forecasting network.

3.2. Data Description

3.2.1. Generic Multi-Energy System

A Multi-Energy system utilizes the transfer among different forms of energy to improve efficiency, sustainability and environmental performance. Such systems also have a higher degree of flexibility. Figure 1 shows the energy interaction structure of a generic multi-energy system.

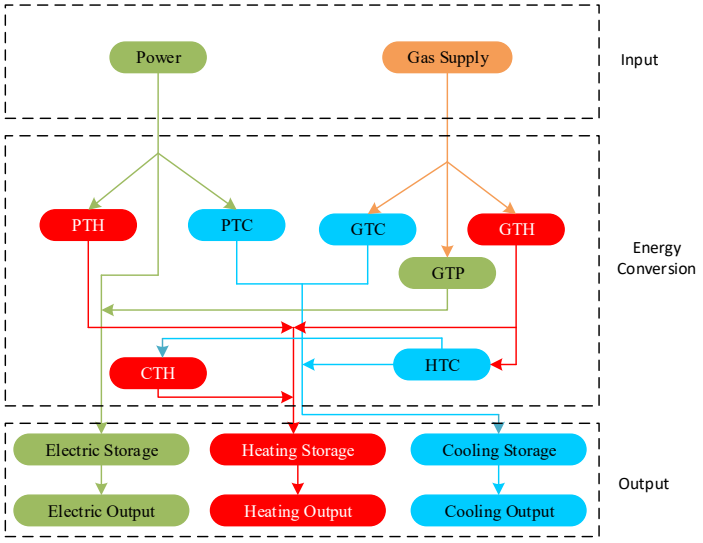
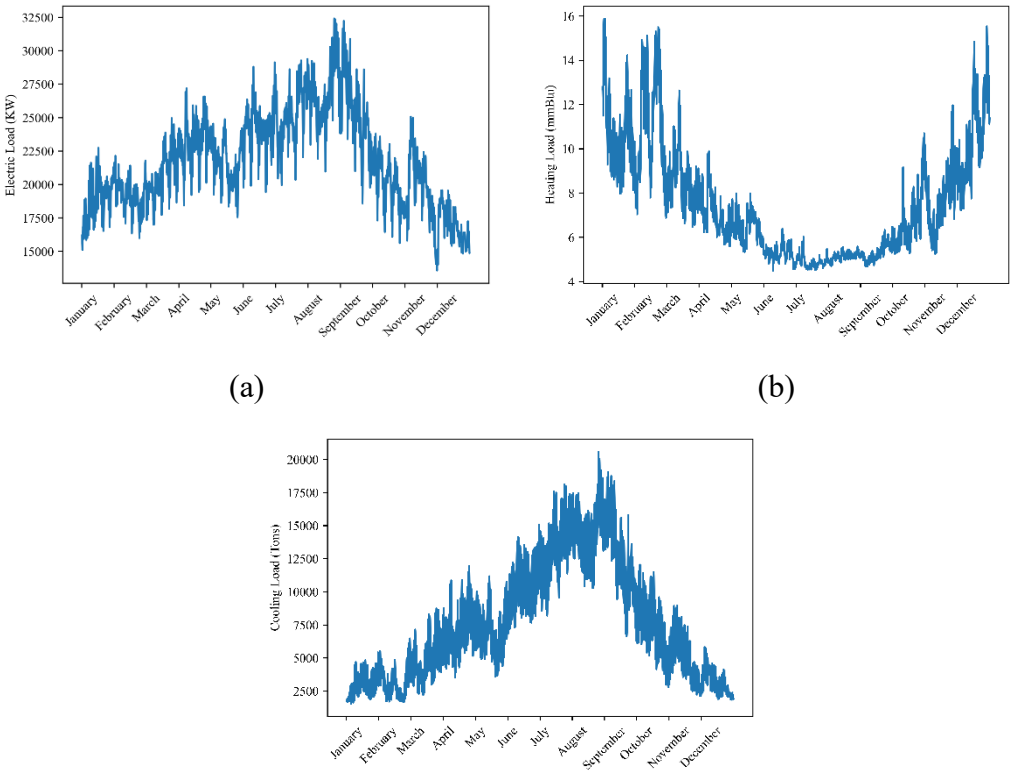


Figure 16: Generic Multi-Energy System

In the figure, power and gas supply are stored and/or converted to electricity, heating and cooling. There are various devices/modules in the second or energy conversion layer to achieve the transformation of energy from one type to the other. Here PTH is the device used for converting power to heating and GTP converts gas to power such as a gas turbine. CTH and HTC are the devices for converting cooling to heating and heating to cooling respectively. Depending on the system design that can be achieve simultaneously by a heat exchanger.

3.2.2. Campus Data Analysis

The dataset in use is from University of Austin Tempe Campus. Data is extracted using Campus Metabolism (a website for energy monitoring and historic data storage). It is the heating, cooling and electricity consumption data for the integrated energy system powering the campus ranging from 2016 to 2019.



(a) (b)

(c)

Figure 17: (a) Electric load data from 2019

(b) Heating load data from 2019

(c) Cooling load data from 2019

Table 2 shows all the major data features of the input data including the number of temporal and metrological variables used. Figure 2 shows the electric, cooling and heating data from 2019 for the campus. It can be seen from the energy usage profiles that there is a clear increase in the use of electric and cooling loads during summer and fall. High number of students on campus coupled with relatively high temperatures on campus create a surge in electricity and cooling demand during early fall. Heating loads show a different trend here as demand is understandably lower during summer but very high in winter. Next, the dataset is divided into different seasons and exploratory data analysis is performed to analyze if data actually belongs to same or different

distribution. Violin plot is a great tool to assess this. We not only get the mean for each of the different seasons but we can also observe the distribution of target variables. Figure 3 shows the violin plots.

Table 8: Current research in multi energy system load forecasting

Reference	Publication Date	Predictive Model	Output	Data Interval
[13]	2016	Long Short Term Memory based RNN	Cooling, electricity and heating	1 hour
[14]	2020	Deep Belief Network	Gas, electricity and heating	1 hour
[15]	2021	Deep Belief Network	Gas, electricity and heating	1 hour
[16]	2014	Non-linear autoregressive model for exogenous variables (NARX)	Cooling, electricity and heating	1 hour
[17]	2019	CNN	Gas, electricity and heating	1 hour
[18]	2022	Bi-directional GRU	Cooling, electricity and heating	1 hour
[19]	2022	CNN-GRU	Cooling, heating, electricity and gas	1 hour
[20]	2023	CNN	Cooling, electricity and heating	1 hour
[21]	2023	Custom Architecture with Fully connected layer, convolution and residual block	Cooling, electricity and heating	1 hour

It's quite evident that apart from having quite different means, the distribution of data is also quite different. Moreover, the daily energy usage profiles are also quite different for different seasons. So it's logical to train, validate and test the data for separate seasons independently.

3.3. Algorithm

The algorithm proposed in this article first involves dividing the data into different seasons so they can be processed separately. After data division, coupling between three energy variables is analyzed using distance correlation analysis. Energy variables (cooling, heating and power) with higher correlations are coupled together. As a result, we have three different sets of input variables used for each season. These sets are fed to the preprocessing pipeline and then to the multi-task learning deep neural network. The network proposed for this task is D-TCN. It consists of multi-layer preceptrons (MLP) followed by temporal convolution network. The MLP encodes the spatial information while TCN deals with the temporal dependencies in the data. The output from three MLP + TCN streams is concatenated and used for forecasting.

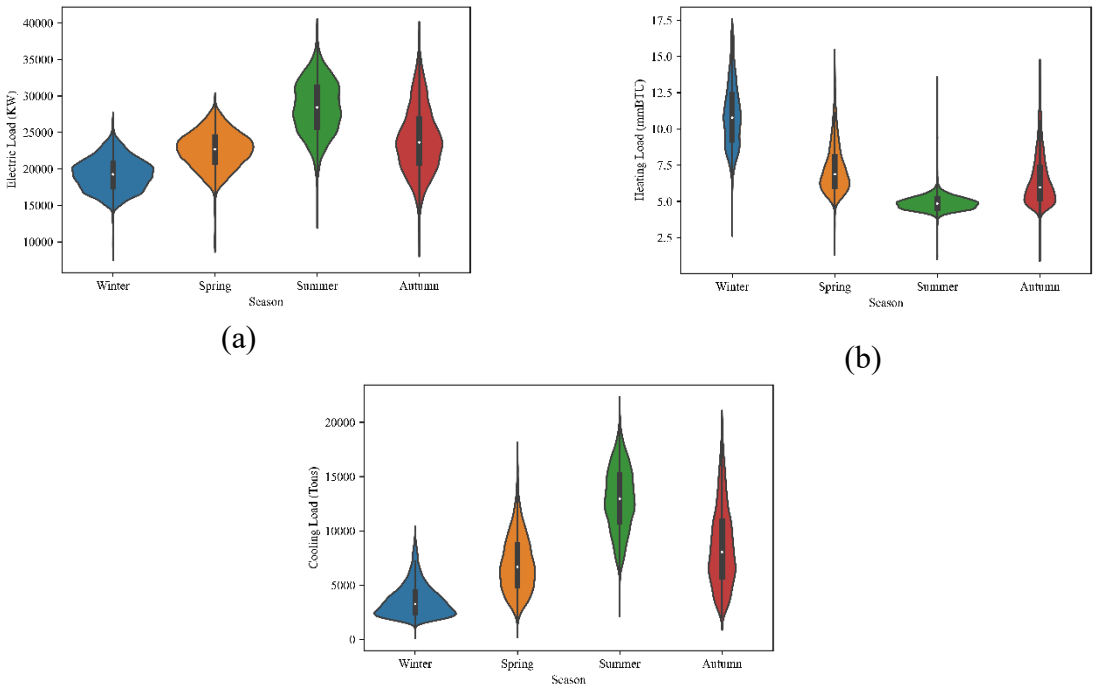


Figure 18: (a) Electric load violin plot

(b) Heating load violin plot

(c) Cooling load violin plot

Data is first divided into seasons based on the months. Summer data is from June to August. Following that, till November it is fall. After fall, till February it is winter. Finally, autumn starts in March and ends in May. After the data is divided, each of the dataset undergoes distance correlation analysis to find out the optimal input variables for D-TCN algorithm.

3.3.1 Distance Correlation Analysis

Distance Correlation or Distance Covariance is a metric used to measure the degree of dependence or interrelationships between two random variables or vectors. Unlike Pearson correlation coefficient, a value of zero implies independence between two vectors or variables.

Distance Correlation is defined as:

$$dCor^2(x,y) = \frac{dCov^2(x,y)}{\sqrt{dVar^2(x)dVar^2(y)}} \quad (1)$$

We will be using distance correlation to calculate the coupling between all three load variables

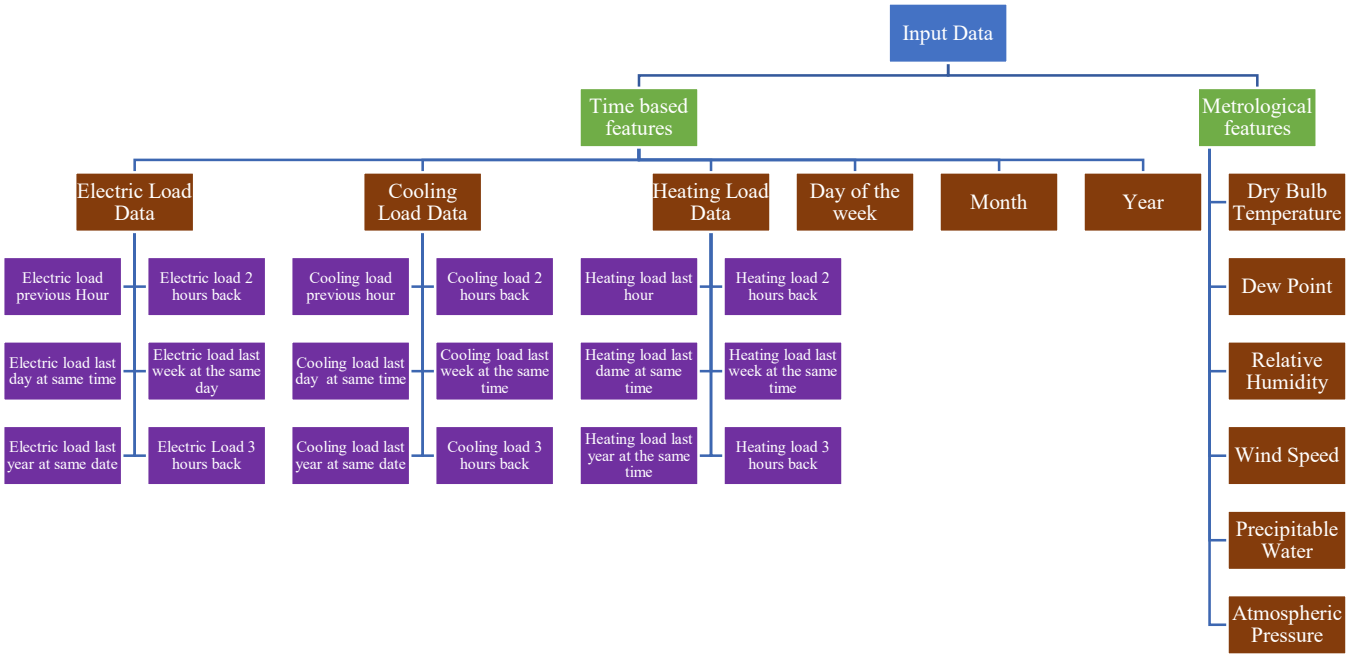


Figure 19: Features used in the forecasting algorithm

across four seasons (Spring, Summer, Winter and Autumn). And based on the strength of the relationship, input variables are selected for algorithm.

Table 9: Salient data features

Parameter	**Value**
Data start date	1st January 2016
Data end date	31st December 2019
Data interval	1 hour
Number of observations	35063
Number of metrological features	6
Number of temporal features	21

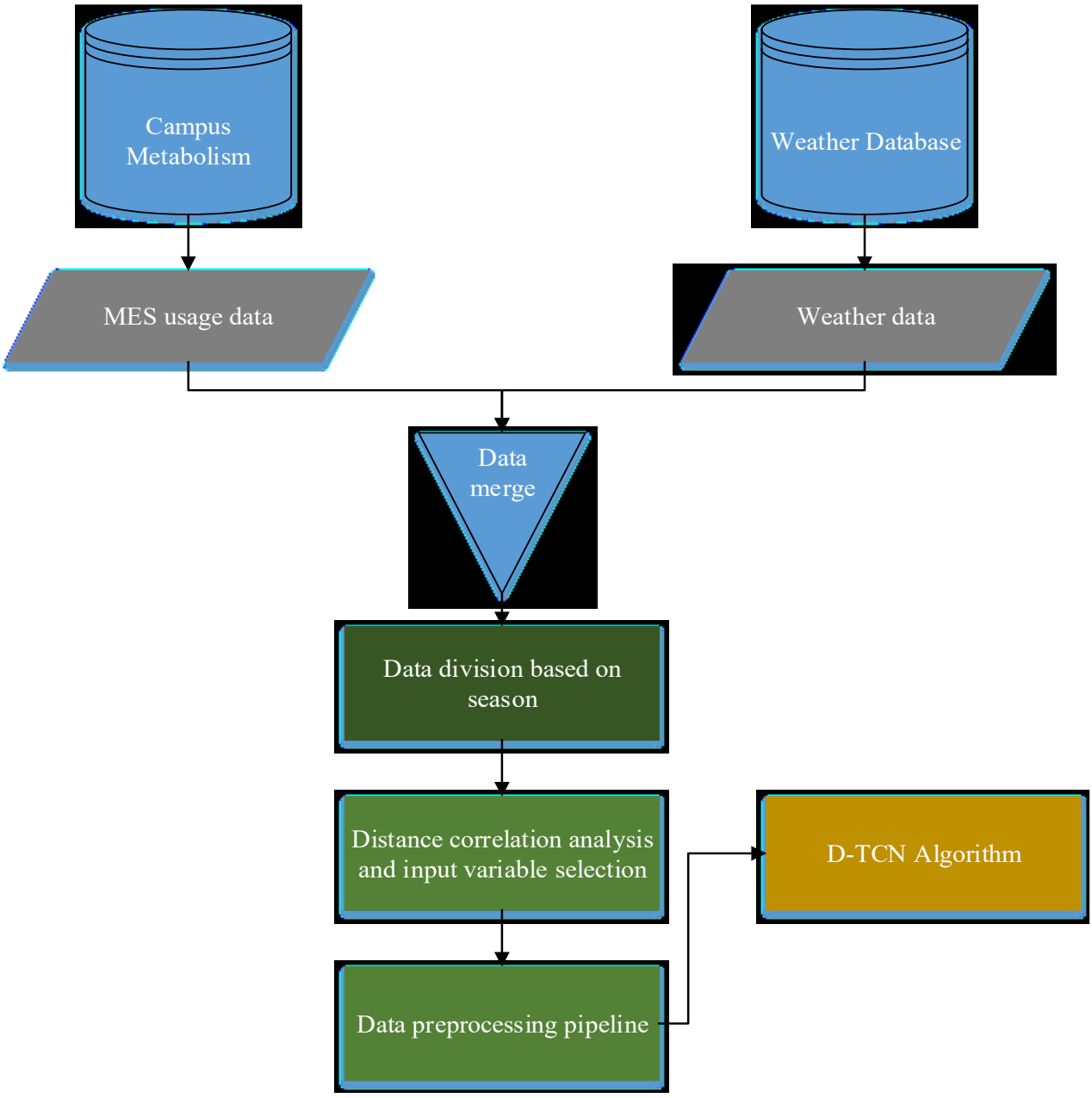


Figure 20: Basic algorithm for MES load forecasting

Distance correlation analysis for the energy variables is carried out and the results show that correlation between energy variables vary between different seasons. For example, heating and cooling data have a very high coupling during winter season as the MES might be using the excess heat for cooling but the relationship is very weak during summer months. We will be using a cut-

off value of 0.4 to separate the highly coupled energy vectors from loosely coupled energy consumption data for each season. The highly coupled data would be used as input for MTL algorithm for each season separately.

The variables with distance correlation of less than 0.4 are not used in the MTL algorithm while the variables with a distance correlation above this limit would be considered as inputs for the proposed forecasting algorithm.

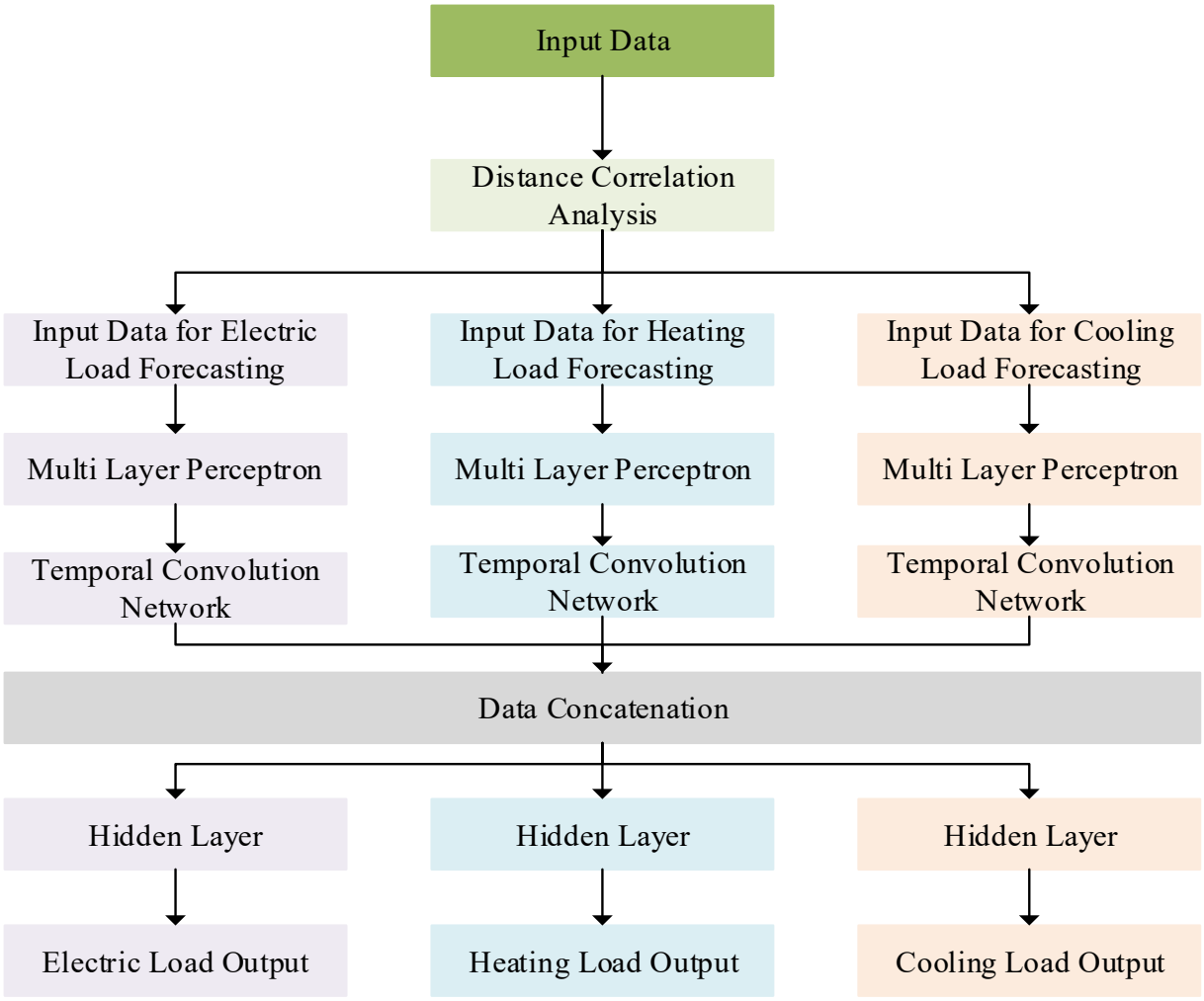


Figure 21: Proposed Architecture

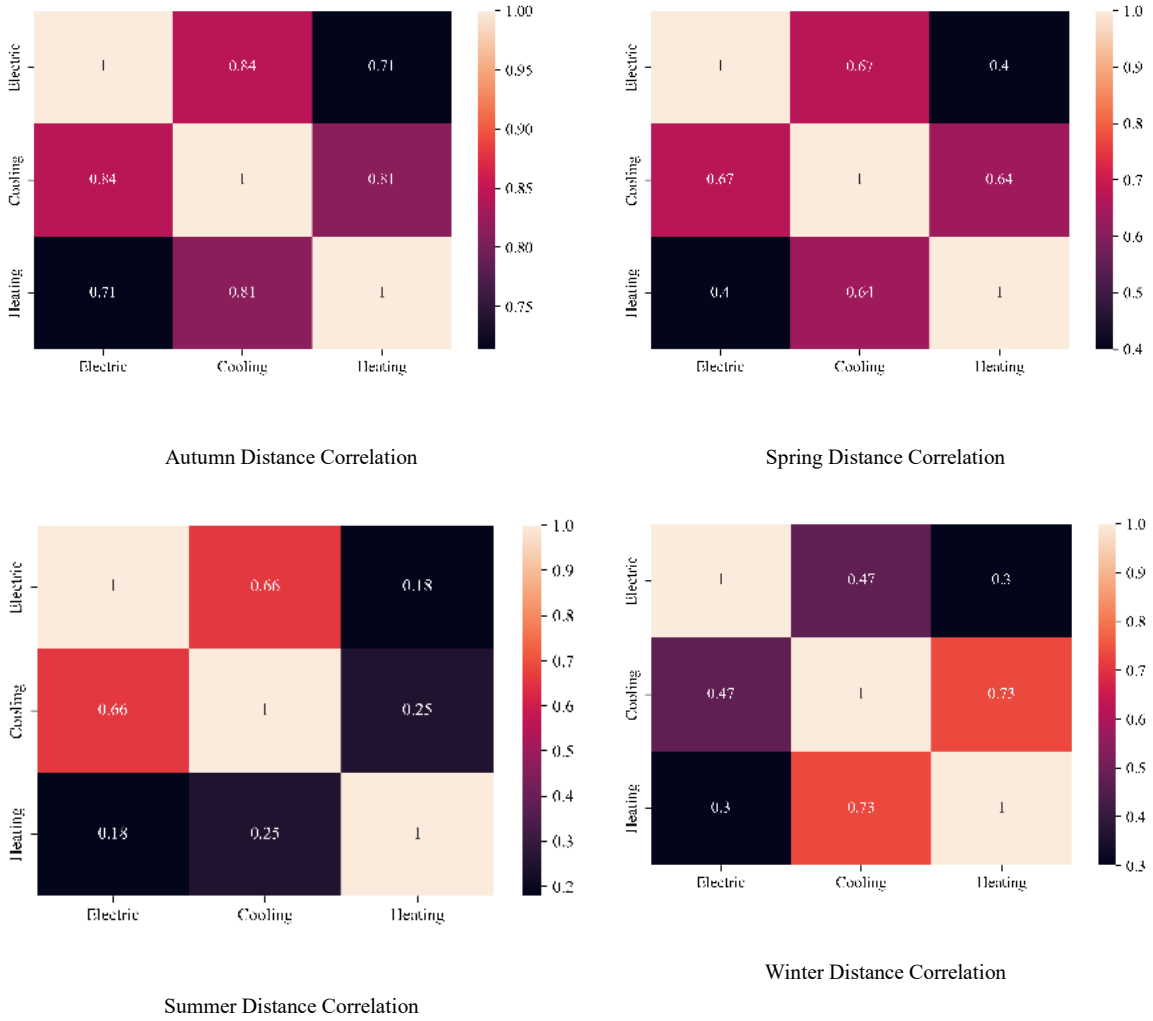


Figure 22: Distance Correlation heat maps

Table 10: Table for selected input variables based on distance correlation analysis

Season	Energy Variable	Input variable		
		Heating	Cooling	Electric
Summer	Heating			
	Cooling			
	Electric			
Winter	Heating			
	Cooling			

	Electric	
Autumn	Heating	
	Cooling	
	Electric	
Spring	Heating	
	Cooling	
	Electric	

Table 3 consolidates the results for distance correlation analysis by highlighting the selected input variables for each season and each input stream for D-TCN algorithm. Autumn and spring use all the available energy variables due to strong correlations observed in the coefficients while the same is not true for summer and winter. There is a very coupling between heating and electricity as well as heating and cooling in both of these

seasons. For this reason, these variables are removed as inputs when predicting heating loads.

3.3.2. Data Preprocessing

First basic exploratory data analysis is performed before data preprocessing pipeline. This is done to ensure there are no extreme outliers or NaN (not a number) values in our data. Outliers are identified and removed using inter-quartile range method. Any data point which is outside of the **[Q1 – 3 * IQR, Q3 + 2 * IQR]** window is replaced using forward filling method. Here Q1 represents the first quartile, Q3 is the third quartile.

Data preprocessing consists of three main steps. Data windowing, data partition and finally data standardization. Data windowing is necessary to pose a load forecasting problem for the neural network to learn.

Data is divided based on stride (s), window size (w) and output target size (o). As we are working with hourly data, we pose the problem as taking weekly data and generating the next hour prediction. We will be using a stride of 1, window size of 168 and output target size of 1.

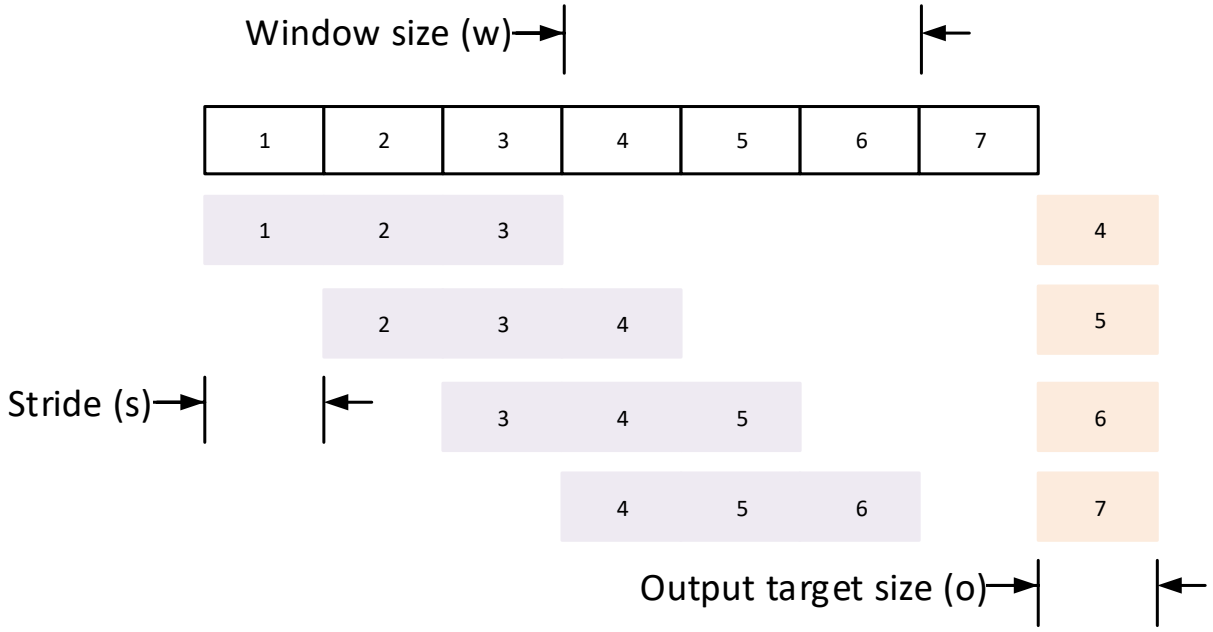


Figure 23: Data Windowing

Figure 8 shows data windowing where stride (s) is 1, output target size is also 1 and window size is 3. After data windowing, we divide the data into train, validation and test split. We use 80% of data for training, 10% for validation and test respectively. It is made sure that training data is always from the past. Next step is data standardization.

Data standardization (or z-score normalization) is employed to ensure that model converges faster. For each feature, we take the difference between feature mean and that value and divide it by standard deviation. This is done for each data window separately. The data standardization for ith feature is done as:

$$\hat{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{2}$$

Here μ_i is the mean while σ_i is the standard deviation for ith feature while $\hat{x}_i$ is the standardized feature . μ_i is defined;

$$\mu_i = \frac{1}{J}\sum_{j=1}^{J} x_{i,j} \tag{3}$$

Here J is the total number of samples for ith feature. σ_i is defined as:

$$\sigma_i = \sqrt{\frac{1}{J}\sum_{j=1}^{J}\left(x_{i,j} - \mu_i\right)^2} \tag{4}$$

3.3.3. Deep Temporal Convolutional Network (D-TCNet)

The proposed deep neural network for forecasting is based on TCN and MLP. MLP is used for finding the spatial correlations among different features as well as to reduce the dimensions of the data similar to word embedding in natural language processing. Data first passes through the MLP and then it moves to TCN which resolves the temporal relations in the data. The data from three input streams (heating, cooling and electricity) is concatenated after passing through MLP and TCN which is then used for forecasting of three energy types in our MES.

3.3.3.1. Temporal Convolutional Neural Network:

TCN was originally proposed in [12] for action segmentation. Since TCN allows parallel computation of outputs, it shows better performance while training.

TCN consists of casual and dilated convolutions. Adding residual block is also a major modification worth discussing. Casual convolutions ensure that the model learns only from the

past time steps. In a casual convolution, the output only depends on the previous input time steps only. However, in order to have a full history coverage or to have a big receptive field for the model, a very deep model would be required. Dilations are introduced to mitigate this issue. Dilated convolutions ensure a fixed distance between input time steps to ensure a much bigger receptive field. In a TCN model, dilation value is varied across layers exponentially and in order to ensure that there are no holes in the receptive field, the kernel size must be as big as the base of the exponential.

Other than adding dilated convolutions, residual blocks are introduced. A residual block consists of a few layers with same dilation factor and a residual connection. It further reduces the number of layers required to achieve a larger receptive field.

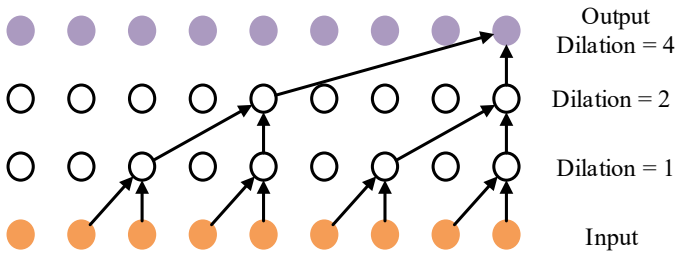


Figure 24: Simple TCN block

Figure 9 shows a TCN architecture with casual and dilated convolutions from input to output. The illustrated network increases dilation exponentially in each layer.

A complete TCN uses residual blocks which constitute of simple TCN blocks, weight regularization, activation function and dropout layer for regularization. Figure 10 shows a complete residual block in detail.

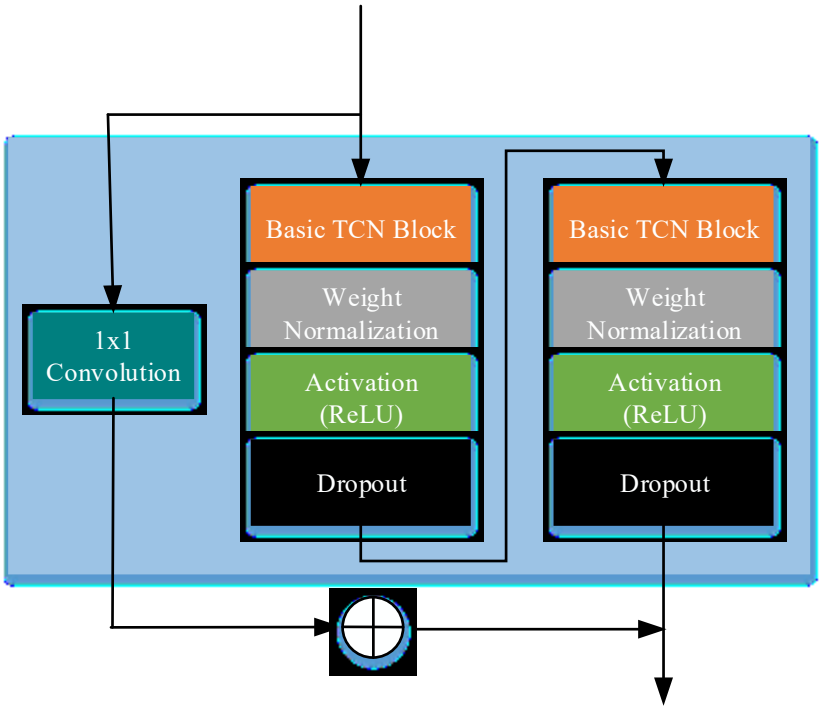


Figure 25: A complete residual block

3.3.3.2. Multi-layer Perceptron (MLP):

The other major component of the D-TCNet is the MLP layer. It is used before the TCN block to reduce the dimension of input to TCN blocks. The MLP encodes the spatial information between variables while TCN resolves the temporal component. The proposed network consists of three separate MLP layers for 3 sets of inputs for the multi-task learning.

The complete network consists of three input streams which take data from three input datasets. Each dataset is used for predicting the respective energy demand in this multi-task learning problem. Following the input stream, we have MLP followed by residual TCN blocks. The output from the three TCN blocks is concatenated and fed to another MLP which produces three outputs. Each of the output is the energy forecast.

After extensive hyperparameter optimization, the most optimal features for the network are given below:

Table 11: Network Parameters

Parameter	Value
MLP Embedding dimension	8
TCN kernel size	8
TCN Dropout	0.2
TCN activation function	Leaky ReLU
Number of TCN Blocks	3

3.3.3.3. Loss Function, Optimizer and Training Routine:

Loss function is the metric which is minimized by the neural network algorithm using an optimizer. Mean squared error is the loss function employed for training the neural network. This loss function is defined as:

$$Mean\ Squared\ Error = \frac{1}{N}\sum_{i=1}^{N}(A_i - P_i)^2 \quad (5)$$

Here A is the vector of actual values while P is the vector of predicted values.

We employ Adam's optimizer [22] for minimizing the loss function. Rest of the training routine features are given in table 5 below.

Table 12: Training Routine Parameters

Parameter	Value
Max epochs	200
Initial Learning Rate	0.01

Learning rate reduction on plateau	Reduction Patience	10
	Reduction factor	0.5

3.3.4. Evaluation Metrics:

3.3.4.1. Mean Absolute Percentage Error (MAPE):

MAPE is a scale independent evaluation metric which can be used to compare algorithm performance across different time series as well. For any forecasting algorithm, a lower MAPE translates to better performance. Although, one major drawback faced by this metric is that it does not penalize positive and negative errors symmetrically. Equation 4 describes MAPE.

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{A_i - P_i}{A_i}\right| \tag{6}$$

Here A is the actual value while P is the predicted value by the algorithm. Number of samples is N.

3.4.4.2. Mean Absolute Error (MAE):

Mean absolute error is obtained by taking the mean of absolute difference between actual and predicted values. A model with higher accuracy would have a lower MAE. Although mean absolute error is symmetric i.e. it penalizes positive and negative errors equally, it is scale dependent so it cannot be used to compare algorithms across different data sets.

$$Mean\ Absolute\ Error = \frac{\sum_{i=1}^{N}|A_i - P_i|}{N} \tag{7}$$

3.4. Results

First we start with a brief description of the evaluation metrics used for algorithm assessment.

3.4.1. Performance Evaluation:

As algorithm is trained separately for each season, algorithm performance is compared separately as well. We will compare our proposed model to support vector machine (SVM), Random Forest (RF) algorithm and LSTM. We also compare the algorithm to state-of-the-art load forecasting algorithms such as multi-task learning least square support vector machine (MTL-LSSM), mutual information coefficient based bidirectional LSTM (MIC + Bi-LSTM) and separation fusion + improved CNN. The proposed model shows improved performance compared to these algorithms. Lastly we compare it to single-task learning approach similar to proposed method with only difference being that there is no concatenation so there are three separate input-output streams. This comparison is done to show the effectiveness of multi-task approach and information sharing layer compared to learning tasks separately.

Table 13: Results for Autumn

Algorithm	Load Type	Evaluation Metrics	
		MAPE (%)	RMSE (KW)
SVM	Electric	7.7	2154.1
	Heating	9.8	251.5
	Cooling	12.1	3698.2
RF	Electric	6.2	1854.2
	Heating	7.4	212.3

	Cooling	10.1	2918.5
LSTM	Electric	4.2	997.6
	Heating	5.1	129.7
	Cooling	8.7	2314.5
MIC + Bi-LSTM [6]	Electric	2.2	N/A
	Heating	**4.1**	N/A
	Cooling	7.4	N/A
Single-Task Approach	Electric	2.4	756.1
	Heating	4.4	108.4
	Cooling	7.9	1917.8
Multi-Task Approach	Electric	**1.9**	605.1
	Heating	4.2	101.2
	Cooling	**6.9**	1645.1

Table 14: Results for Spring

Algorithm	Load Type	Evaluation Metrics	
		MAPE (%)	RMSE (KW)
SVM	Electric	6.1	1976.2
	Heating	13.1	250.8
	Cooling	11.9	3218.3
RF	Electric	5.7	1716.6
	Heating	8.4	212.7
	Cooling	8.1	2798.5
LSTM	Electric	3.4	1046.7
	Heating	5.5	140.9
	Cooling	7.9	2567.7
MIC + Bi-LSTM	Electric	**2.1**	N/A
	Heating	3.7	N/A

[6]	Cooling	3.5	N/A
Single-Task Approach	Electric	2.2	811.1
	Heating	3.8	101.1
	Cooling	3.5	1134.7
Multi-Task Approach	Electric	**2.1**	768.6
	Heating	**3.5**	99.8
	Cooling	**3.1**	976.8

Table 15: Results for Summer

Algorithm	Load Type	Evaluation Metrics	
		MAPE (%)	RMSE (KW)
SVM	Electric	7.2	1956.8
	Heating	9.5	298.1
	Cooling	12.5	5067.1
RF	Electric	6.1	1838.4
	Heating	5.0	135.4
	Cooling	7.7	4567.6
LSTM	Electric	3.6	1428.5
	Heating	2.6	53.4
	Cooling	3.9	2547.5
MIC + Bi-LSTM [6]	Electric	2.4	N/A
	Heating	2.1	N/A
	Cooling	2.2	N/A
Separation Fusion + CNN [20]	Electric	2.3	N/A
	Heating	**1.5**	N/A
	Cooling	2.7	N/A
	Electric	2.3	N/A

MTL-LSSM [11]	Heating	3.3	N/A
	Cooling	3.8	N/A
Single-Task Approach	Electric	2.5	1129.6
	Heating	2.3	50.4
	Cooling	2.1	1545.5
Multi-Task Approach	Electric	**2.1**	801.5
	Heating	**1.5**	48.1
	Cooling	**2.0**	1359.5

Table 16: Results for Winter

Algorithm	Load Type	Evaluation Metrics	
		MAPE (%)	RMSE (KW)
SVM	Electric	6.6	2413.3
	Heating	10.5	768.4
	Cooling	17.1	5472.9
RF	Electric	6.0	2187.4
	Heating	7.6	345.7
	Cooling	12.1	2098.6
LSTM	Electric	3.8	1015.4
	Heating	4.9	210.7
	Cooling	9.1	1456.8
MIC + Bi-LSTM [6]	Electric	2.5	N/A
	Heating	3.2	N/A
	Cooling	5.1	N/A
Separation Fusion + CNN [20]	Electric	1.5	N/A
	Heating	3.0	N/A
	Cooling	**1.9**	N/A

MTL-LSSM [11]	Electric	1.9	N/A
	Heating	3.5	N/A
	Cooling	3.9	N/A
Single-Task Approach	Electric	2.6	710.3
	Heating	3.6	154.8
	Cooling	5.2	870.4
Multi-Task Approach	Electric	**1.4**	680.5
	Heating	**2.7**	125.7
	Cooling	2.1	774.7

It can be seen that proposed Deep TCN algorithm has shown the lowest (and hence best) MAPE and RMSE. It performs better than the single-task learning approach in every scenario as the information sharing layer helps it combine information from all three input streams.

There are two big leaps in accuracy when analyzing the metrics. First jump is observed when we move from conventional machine learning algorithms to deep learning approaches and second leap is observed when moving from deep learning to distance correlation (DCor) based approach which accounts for coupling information among energy variables.

A significant decrease in training time is also observed when shifting from single-task to multi-task learning as instead of training three independent models, one model outputs all three energy variables. Figure 11 shows the improvement in training time when employing multi-task learning compared to single task learning. The training time has been reduced to a third here.

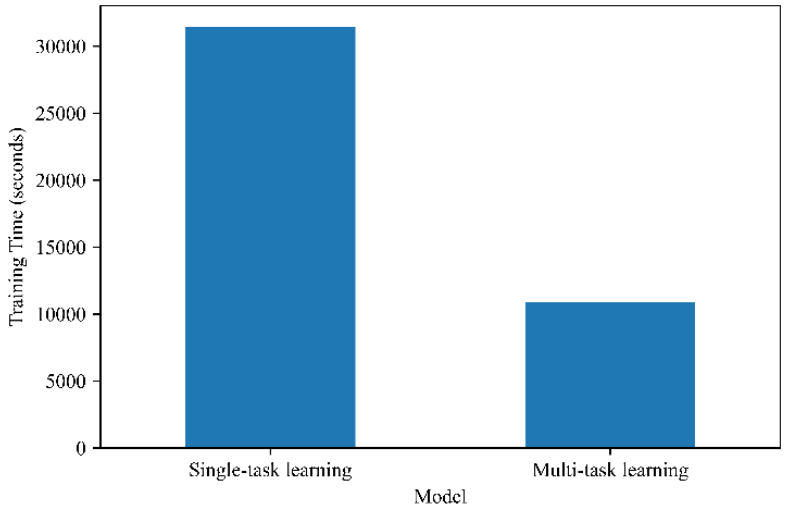


Figure 26: Training time comparison

3.5. Conclusion

The work presents a new feature selection method and a novel TCN-based forecasting model for multi-energy systems. The feature selection method uses distance correlation to quantify coupling among heating, cooling and electricity variables. Highly correlated variables are then selected. The TCN-based network uses multi-task learning approach to forecast all three energy variables simultaneously. The effectiveness of the proposed method is verified by increased accuracy compared to other algorithms in a detailed analysis which divides the data seasonally. The use of proposed method significantly improves MAPE and RMSE compared to other state-of-the-art algorithms.

3.6. References

[1] P. Mancarella, "MES (multi-energy systems): An overview of concepts and evaluation models," *Energy*, vol. 65, pp. 1–17, Feb. 2014, doi: 10.1016/j.energy.2013.10.041.

[2] N. Liu, J. Wang, and L. Wang, "Hybrid Energy Sharing for Multiple Microgrids in an Integrated Heat–Electricity Energy System," *IEEE Trans. Sustain. Energy*, vol. 10, no. 3, pp. 1139–1151, Jul. 2019, doi: 10.1109/TSTE.2018.2861986.

[3] Y. Yan and Z. Zhang, “Cooling, Heating and Electrical Load Forecasting Method for Integrated Energy System based on SVR Model,” in *2021 6th Asia Conference on Power and Electrical Engineering (ACPEE)*, Apr. 2021, pp. 1753–1758. doi: 10.1109/ACPEE51499.2021.9436990.
[4] “Energies | Free Full-Text | Short-Term Load Forecasting for CCHP Systems Considering the Correlation between Heating, Gas and Electrical Loads Based on Deep Learning.” Accessed: Feb. 02, 2024. [Online]. Available: https://www.mdpi.com/1996-1073/12/17/3308
[5] J. Zheng *et al.*, “Multiple-Load Forecasting for Integrated Energy System Based on Copula-DBiLSTM,” *Energies*, vol. 14, no. 8, Art. no. 8, Jan. 2021, doi: 10.3390/en14082188.
[6] Y. Guo *et al.*, “BiLSTM Multitask Learning-Based Combined Load Forecasting Considering the Loads Coupling Relationship for Multienergy System,” *IEEE Trans. Smart Grid*, vol. 13, no. 5, pp. 3481–3492, Sep. 2022, doi: 10.1109/TSG.2022.3173964.
[7] M. Clark, “A Comparison of Correlation Measures”.
[8] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, Art. no. 7553, May 2015, doi: 10.1038/nature14539.
[9] M. You, Q. Wang, H. Sun, I. Castro, and J. Jiang, “Digital twins based day-ahead integrated energy system scheduling under load and renewable energy uncertainties,” *Appl. Energy*, vol. 305, p. 117899, Jan. 2022, doi: 10.1016/j.apenergy.2021.117899.
[10] B. Chen and Y. Wang, “Short-Term Electric Load Forecasting of Integrated Energy System Considering Nonlinear Synergy Between Different Loads,” *IEEE Access*, vol. 9, pp. 43562–43573, 2021, doi: 10.1109/ACCESS.2021.3066915.
[11] H. Zhao and S. Guo, “Uncertain Interval Forecasting for Combined Electricity-Heat-Cooling-Gas Loads in the Integrated Energy System Based on Multi-Task Learning and Multi-Kernel Extreme Learning Machine,” *Mathematics*, vol. 9, no. 14, Art. no. 14, Jan. 2021, doi: 10.3390/math9141645.
[12] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, “Temporal Convolutional Networks: A Unified Approach to Action Segmentation,” in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 47–54. doi: 10.1007/978-3-319-49409-8_7.
[13] “Cooling, heating and electrical load forecasting method for CCHP system based on multivariate phase space reconstruction and kalman filter - Google Search.” Accessed: Apr. 21, 2024. [Online].
[14] L. Zhang, J. Shi, L. Wang, and C. Xu, “Electricity, Heat, and Gas Load Forecasting based on Deep Multitask Learning in Industrial-Park Integrated Energy System,” *Entropy Basel Switz.*, vol. 22, no. 12, p. 1355, Nov. 2020, doi: 10.3390/e22121355.
[15] B. Zhou *et al.*, “Multi-energy net load forecasting for integrated local energy systems with heterogeneous prosumers,” *Int. J. Electr. Power Energy Syst.*, vol. 126, p. 106542, Mar. 2021, doi: 10.1016/j.ijepes.2020.106542.
[16] K. M. Powell, A. Sriprasad, W. J. Cole, and T. F. Edgar, “Heating, cooling, and electrical load forecasting for a large-scale district energy system,” *Energy*, vol. 74, pp. 877–885, Sep. 2014, doi: 10.1016/j.energy.2014.07.064.
[17] R. Zhu, W. Guo, and X. Gong, “Short-Term Load Forecasting for CCHP Systems Considering the Correlation between Heating, Gas and Electrical Loads Based on Deep Learning,” *Energies*, vol. 12, no. 17, Art. no. 17, Jan. 2019, doi: 10.3390/en12173308.

[18] D. Niu, M. Yu, L. Sun, T. Gao, and K. Wang, “Short-term multi-energy load forecasting for integrated energy systems based on CNN-BiGRU optimized by attention mechanism,” *Appl. Energy*, vol. 313, p. 118801, May 2022, doi: 10.1016/j.apenergy.2022.118801.
[19] C. Li, G. Li, K. Wang, and B. Han, “A multi-energy load forecasting method based on parallel architecture CNN-GRU and transfer learning for data deficient integrated energy systems,” *Energy*, vol. 259, p. 124967, Nov. 2022, doi: 10.1016/j.energy.2022.124967.
[20] K. Li, Y. Mu, F. Yang, H. Wang, Y. Yan, and C. Zhang, “A novel short-term multi-energy load forecasting method for integrated energy system based on feature separation-fusion technology and improved CNN,” *Appl. Energy*, vol. 351, p. 121823, Dec. 2023, doi: 10.1016/j.apenergy.2023.121823.
[21] P. Zhao *et al.*, “Geometric Loss-Enabled Complex Neural Network for Multi-Energy Load Forecasting in Integrated Energy Systems,” *IEEE Trans. Power Syst.*, pp. 1–13, 2023, doi: 10.1109/TPWRS.2023.3345328.
[22] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. Accessed: Feb. 28, 2024. [Online]. Available: http://arxiv.org/abs/1412.6980

Chapter 4: Campus Electric Load Forecasting Using Recurrent Neural Networks

The work was presented in the 12th International Conference on Smart Grid (icSmartGrid), Setubal, Portugal, 2024

Abstract

Short-Term Load Forecasting is a challenging research problem and has a tremendous impact on the generation, transmission, and distribution of electricity. In this paper, different Recurrent Neural Network (RNN) based time-series forecasting algorithms are applied to the electric load data obtained from the Memorial University of Newfoundland. These algorithms include Long Short Term Memory (LSTM), Gated Recurrent Network (GRU), Bi-GRU and Bi-LSTM. The data is trained on the previous week and its accuracy for the next day is measured using hourly forecasts. The accuracy of these algorithms is compared for day-ahead forecasting potential. Bi-GRU based RNN shows best performance among the tested algorithms with the highest R2 score and lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Keywords: Time-series Forecasting, Deep Neural Networks, Long Short Term Memory, Gated Recurrent Unit

4.1 Introduction

Modern power systems require an uninterrupted electricity supply and that demands the least amount of error when determining electric load demand. [1]. Accurate day-ahead load forecasting is critical the for system's operation [2]. The financial impact of load forecasting accuracy is also very critical. 1% raise in forecasting error is associated with 10 million dollars in operating costs [3]. Hence, it is extremely important to improve the accuracy of load forecasting algorithms.

Load forecasting problem can be divided into 3 major categories based on the duration or length of forecasting horizon: (1) Short-term load forecasting deals with intervals ranging from one hour to one week, (2) Medium-term forecasting deals with predicting electricity demand from one week up to 1 year and (3) Long-term forecasting deals with predictions longer than 1 year [4].

Forecasting methods can also be divided based on the number of inputs required by the algorithm. They can be divided into multi-factor forecasting approach which uses multi variables to predict the load demand and time-series approach in which the algorithm only uses the univariate load data [5]. Forecasting algorithms can also be divided into artificial intelligence-based models, statistical models and hybrid methods [6].

Statistical Algorithms can be divided into [6]:

1. Autoregressive (AR) Model
2. Moving Average (MA) Model
3. Autoregressive Moving Average (ARMA) Model
4. Autoregressive Integrated Moving Average (ARIMA) Model
5. Seasonal Autoregressive Integrated Moving Average (SARIMA) Model
6. ARIMAX and SARIMAX models
7. Kalman Filtering Algorithm
8. Grey Models
9. Exponential Smoothing (ES)

Artificial Intelligence and computational intelligence-based algorithms can be divided into:

1. Artificial Neural Network Algorithms (ANN)
2. Extreme learning machines (ELM)
3. Support Vector Machines (SVM)

4. Fuzzy Logic
5. Wavelet Neural Networks (WNN)
6. Genetic Algorithms (GA)
7. Expert System

Exponential Smoothing and Autoregressive approaches have been considered baseline for time series forecasting but for these approaches we have to manually set the number of inputs to use and also make a-priori assumption about the data [7]. In particular, ARIMA is among the most popular approaches but the algorithm works under the assumption that future values are linearly related to the observed data points, hence it is unsuitable for modeling highly non-linear behavior [8]. Time series exhibit temporal dependencies which cause two identical points in time to exhibit different behavior in the future. For time series prediction tasks, deep learning architectures show greater potential due to their ability to learn complex features and patterns in the data. Among the conventional machine learning techniques, Rational Quadratic Gaussian Process Regression (GPR) and exponential GPR are the best performing algorithms for short-term load forecasting on campus loads [9]. For water quality prediction tasks, ANN was found to have a greater generalization ability [10]. Machine Learning algorithms have also been used for state of health (SOH) prediction of batteries for electric vehicle [11]. Machine learning algorithms have been used in multiple other applications as shown in [12], [13], [14], [15] and [16]. For short-term forecasting, Long Short Term Memory (LSTM) architecture showed superior performance [17]. For day-ahead forecasting in independent buildings, either LSTM or Bi-directional LSTM showed superior performance compared to more conventional techniques [18].

The work focuses on a comparison between various forecasting methods for day-ahead load prediction for the Memorial University of Newfoundland. The paper is divided into 5 Sections. In

Section II, data would be described (campus load information and metrological data) is provided and basic characteristics of the data are presented. In Section III, the algorithms are discussed. In Section IV, the procedure and simulation results are discussed. Finally, Section V concludes the paper.

4.2 Data Description

Memorial University of Newfoundland (MUN) is located in St. John's Newfoundland and Labrador (NL). On the average temperature of the city can range from -6 Degrees Celsius to as high as 21 Degrees Celsius.

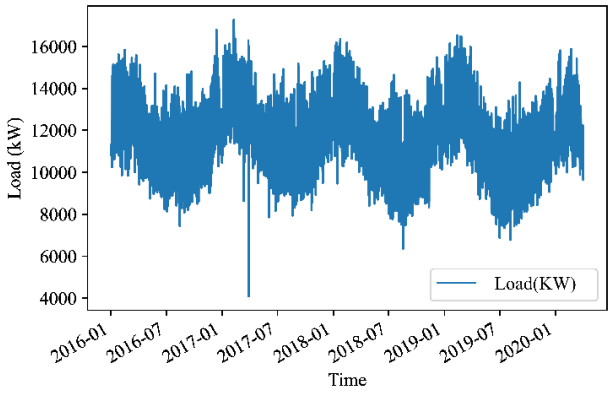


Figure 27 Electric Load Data

Due to this harsh weather, heating system represents a very significant portion of the total electric load. MUN employs hot water plants to fulfill its heating requirements. During the harsh weather of winter, the boilers operate at full capacity while during summer the load is decreased and usually half of the boilers are used.

The boilers use diesel to operate and can consume up to 70000 barrels of oil annually to heat more than 50 buildings across the campus. University uses two meters to monitor the electric load. Meters log data every 15 minutes. We will conduct our analysis on hourly data.

The average electric load by weekday can be seen in the graph below. The decreased load during weekends is quite evident.

We can also break down the average load consumption on monthly basis. It is evident from the figure 3 that load during summer months is lower than compared to winter as heating requirements are comparatively lower

Metrological data has a significant impact on energy consumption so this data will be used in the forecasting strategy. Typically used variables for forecasting include temperature, relative humidity, visibility, cloud cover, rainfall, precipitation, dew point, wind speed and wind chill [19] [20].

Our data will be using the following metrological factors:

1. Dry Bulb Temperature: The temperature visible on a thermometer when it is exposed to the air in absence of moisture and radiation is called dry bulb temperature. It is proportional to mean kinetic energy of the air molecules.

2. Dew Point: The temperature required at constant pressure to achieve a relative humidity of 100% is called the dew point of air. It is directly proportional to the moisture in air.

3. Relative Humidity: At any given temperature, the ratio of water mass to air mass gives represents absolute humidity. Relative humidity is obtained by dividing absolute humidity by maximum possible humidity at any given temperature.

4. Wind Direction: It represents the direction of blowing wind. A value of 0 denotes that the wind is calm. A value of 18 represents that wind is blowing from the south while a value of 27 means that the wind is blowing from the west.

5. Wind Speed: It is the speed of the wind. It is measured at a distance of 10m from the ground. In our data we are using km/h as the measuring unit.

6. Visibility: Visibility is the distance at which an object of tangible size can be observed. In our data, we are using kilometers as the measuring unit.

7. Atmospheric Pressure: The atmospheric pressure is the force exerted by the atmosphere per unit area at the height of the measuring station.

Table 1 shows all the dataset parameters which will be used in training simulation. It shows the interval between data points, data start date, end date and number of features used for the simulation. This metrological data was obtained from weather.gc.ca.

The metrological data is concatenated with load data to make the input dataset.

Table 17: Training dataset parameters

Parameter	Value
Data start date	January 1st 2015
Data end date	March 31st 2019
Data interval	1 hour
Total data points	37221
Features	8

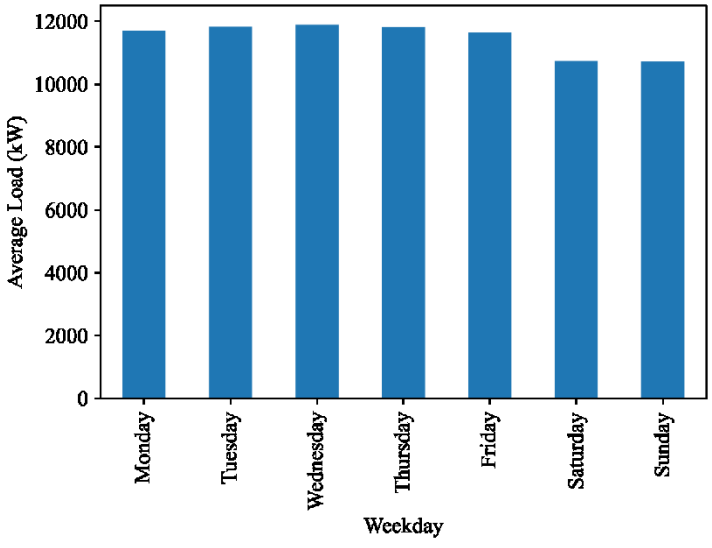


Figure 28 Average Electric Load by Weekday

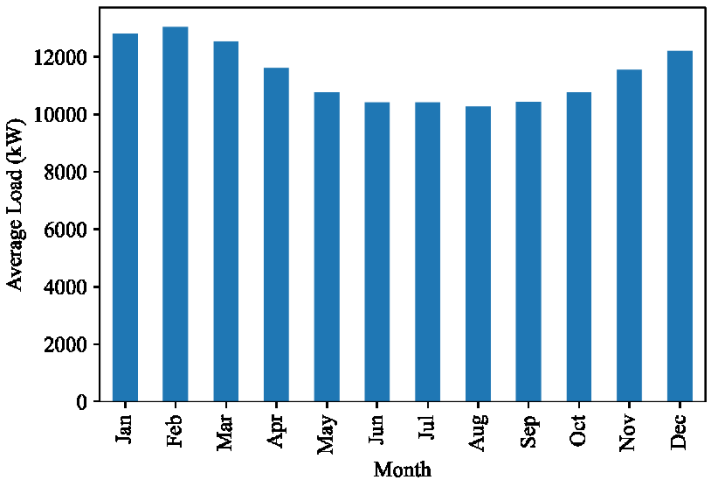


Figure 29 Average Electric Load by Month

4.3. Background Information

4.3.1. Algorithms

4.3.1.1. Long Short-Term Memory Recurrent Neural Network:

Recurrent neural networks are a type of artificial neural network which are capable of handling sequential data. A simple RNN network takes the input value of the sequence and combines it with hidden state from previous timestamp and uses an activation function to generate the hidden state for the next timestamp. RNNs are susceptible to exploding and vanishing gradients. To resolve this issue, RNN cells are modified using various gates. Long Short Term Memory (LSTM) [21] artificial neural network consists of LSTM cells. Each cell performs a set of mathematical operations called gates. Each cell has a forget gate, input gate and output gate. Through these gates, an LSTM cell is able to forget or pay attention to different parts of a sequential input [22].

Forget Gate: The forget gate (f_t) determines what data needs to be forgotten from a network's long-term memory /cell state (c_t) based on the new input x_t and hidden state obtained from the previous time step (h_{t-1}).c

$$f_t = \text{sigmoid}(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \quad (1)$$

Input Gate: The input gate (I_t) determines which information needs to be added to the network based on the input and hidden state from the last step. Input gate also generates a candidate hidden state ($\hat{c}_t$) by using a tangent hyperbolic activation function. Finally the cell state at time t represented by c_t is calculated as well:

$$I_t = \text{sigmoid}(W_{Ix}x_t + W_{Ih}h_{t-1} + b_I) \quad (2)$$

$$\hat{c}_t = \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \quad (3)$$

$$c_t = f_t . c_{t-1} + \hat{c}_t . I_t \quad (4)$$

Output Gate: The output gate (O_t) is responsible for generating the hidden state (h_t) which is passed over to the next cell in the sequence.

$$O_t = \text{sigmoid}(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad (5)$$

$$h_t = O_t \otimes \tanh(c_t) \quad (6)$$

All the variables represented by W and b represent weights and biases (respectively) learned by the network on training. The $\otimes$ symbol represents pointwise multiplication.

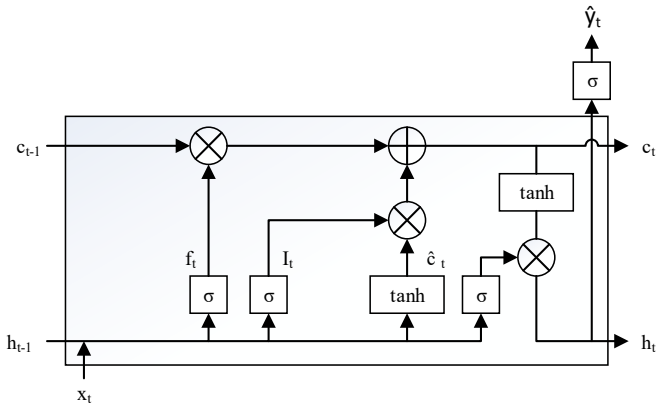


Figure 30: A basic LSTM Network

4.3.1.2. Gated Recurrent Unit

Gated Recurrent Units are a type of Recurrent neural network in which a single unit has 2 gates (reset gate and update gate). [23]. GRUs also do not have cell states and instead they use hidden state to pass the information to the next time step. Compared to LSTMs, this network is simpler and takes less time to train.

The basic description of the gates is given below:

Reset Gate: The reset gate decides what portion of the information from previous time step can be forgotten.

$$r_t = \text{sigmoid}(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \tag{7}$$

Update Gate: Update gate in the GRU decides what portion of the information from previous time steps can be passed to the next blocks.

$$z_t = \text{sigmoid}(W_{zx}x_t + W_{zh}h_{t-1} + b_z) \tag{8}$$

First, the reset gate is used to create a candidate vector ($\hat{h}_t$) follow:

$$\hat{h}_t = \tanh(W_z x_t + W_h(r_t \otimes h_{t-1}) + b_{ht}) \tag{9}$$

Finally, the update gate is used to generate the hidden state as follow:

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

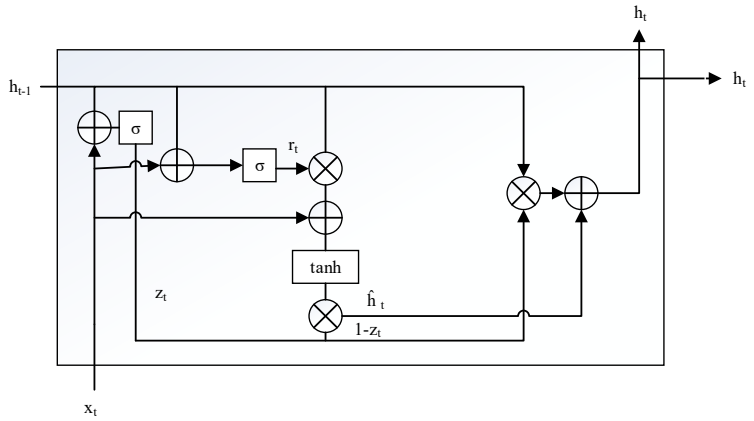


Figure 31: A Basic GRU Unit

4.3.1.3. Bi-Directional Recurrent Neural Networks

Bi-directional RNNs process the data in two directions using two different hidden layers before feeding it to

the same output layer [24]. This architecture can be used for both, LSTM Recurrent Neural Networks (Bi-Directional LSTM) and Gated Recurrent Networks (Bi-Directional GRU). The difference between two architectures can be observed from the comparison of Figure 6 and Figure 7.

4.3.2 Evaluation Metrics:

4.3.2.1. Mean Absolute Error (MAE)

Mean absolute error is the average of the absolute difference between a predicted/forecasted value and ground truth. It can be written as:

$$\text{Mean Absolute Error} = \frac{\sum_{i=1}^{N} |P_i - G_i|}{N} \quad (11)$$

Here, P and G are prediction and ground values respectively. N is the number of values in the dataset. If an estimator is more accurate, it would result in a lower MAE. An inaccurate estimator would result in a higher MAE. It is a scale dependent metric, which means that the MAE values cannot be compared across different datasets to compare performance. MAE is also not differentiable at zero. As most optimization algorithms rely on differentiation, they would have trouble updating model parameters around that point. As a result, the learning process might face

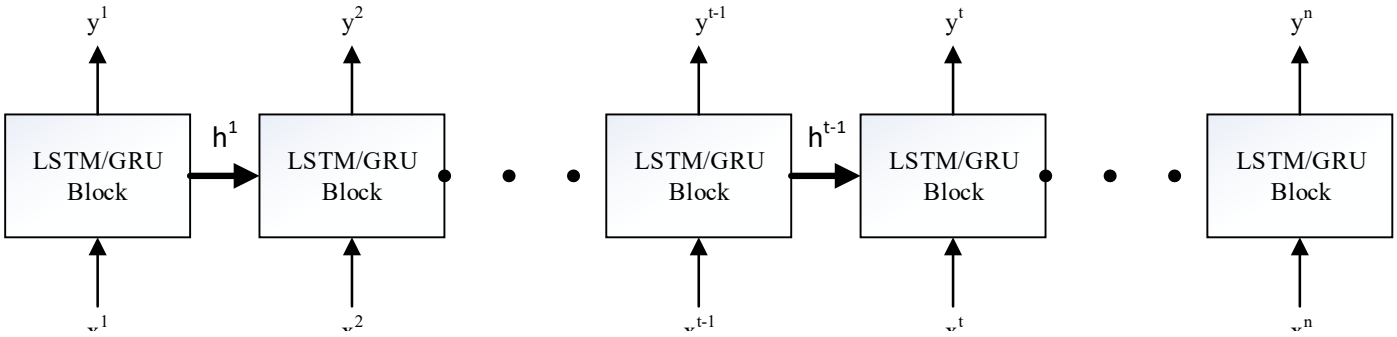
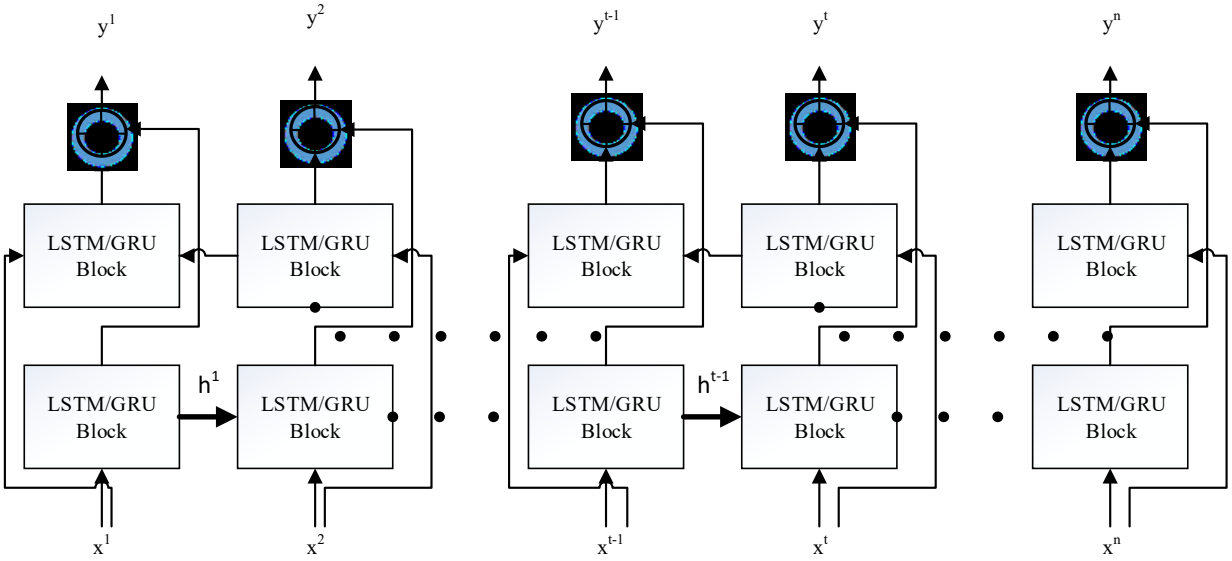


Figure 32: A Unidirectional Network

slow convergence or oscillatory behavior. This could be a problem especially when this metric is used as a loss function. Other optimization algorithms such as mean squared error could alleviate this problem as it is differentiable at x = 0.

4.3.2.2. Mean Squared Error

Figure 33: A bidirectional network



Mean Squared Error is an evaluation metric obtained by calculating the average of the squared difference between the predicted values and ground truth. This evaluation

metric is based on Euclidean distance. It can be written as:

$$\text{Mean Squared Error} = \frac{\sum_{i=1}^{N}(P_i\text{-}G_i)^2}{N} \tag{12}$$

This evaluation metric is based on Euclidian distance. An accurate estimator would have a lower MSE while an inaccurate estimator would have a higher MSE. Compared to MAE, MSE is more sensitive to outliers since errors are squared. MSE, like MAE, is also scale dependent which limits its ability to measure algorithm performance across different datasets.

4.3.2.3. R2 Score

R2 score is also called the coefficient of determination. It can have a maximum value of 1which would indicate that the model has predicted every ground truth value correctly. It can also have negative values with no limit since an estimator can be arbitrarily worse. If the ground truth is non-constant and the model predicts a constant mean value, then R^2 value would be 0. The formula for this metric is given by the relationship below:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(P_i - G_i)^2}{\sum_{i=1}^{N}(G_i - M)^2} \quad (13)$$

Here M is the mean of all the observed values and is defined as:

$$M = \sum_{i=1}^{N}(G_i) \quad (14)$$

It is evident from 15 that R^2 would have a value of 1 for a model with all the predicted values equal to ground truths as $(P_i - G_i)^2$ would be 0. For a baseline model that only predicts mean values, $\frac{\sum_{i=1}^{N}(P_i - G_i)^2}{\sum_{i=1}^{N}(G_i - M)^2}$ would be 1 and overall value would be 0.

4.4. Methodology & Simulation Results

4.4.1. Methodology

The basic process after the data collection includes data cleaning, data preprocessing and training. The basic description of these processes is as follow:

4.4.1.1. Data Cleaning

Data cleaning and data preprocessing are distinct steps, each involving different procedures applied to the dataset. Typically, data cleaning is the initial phase, while data preprocessing follows

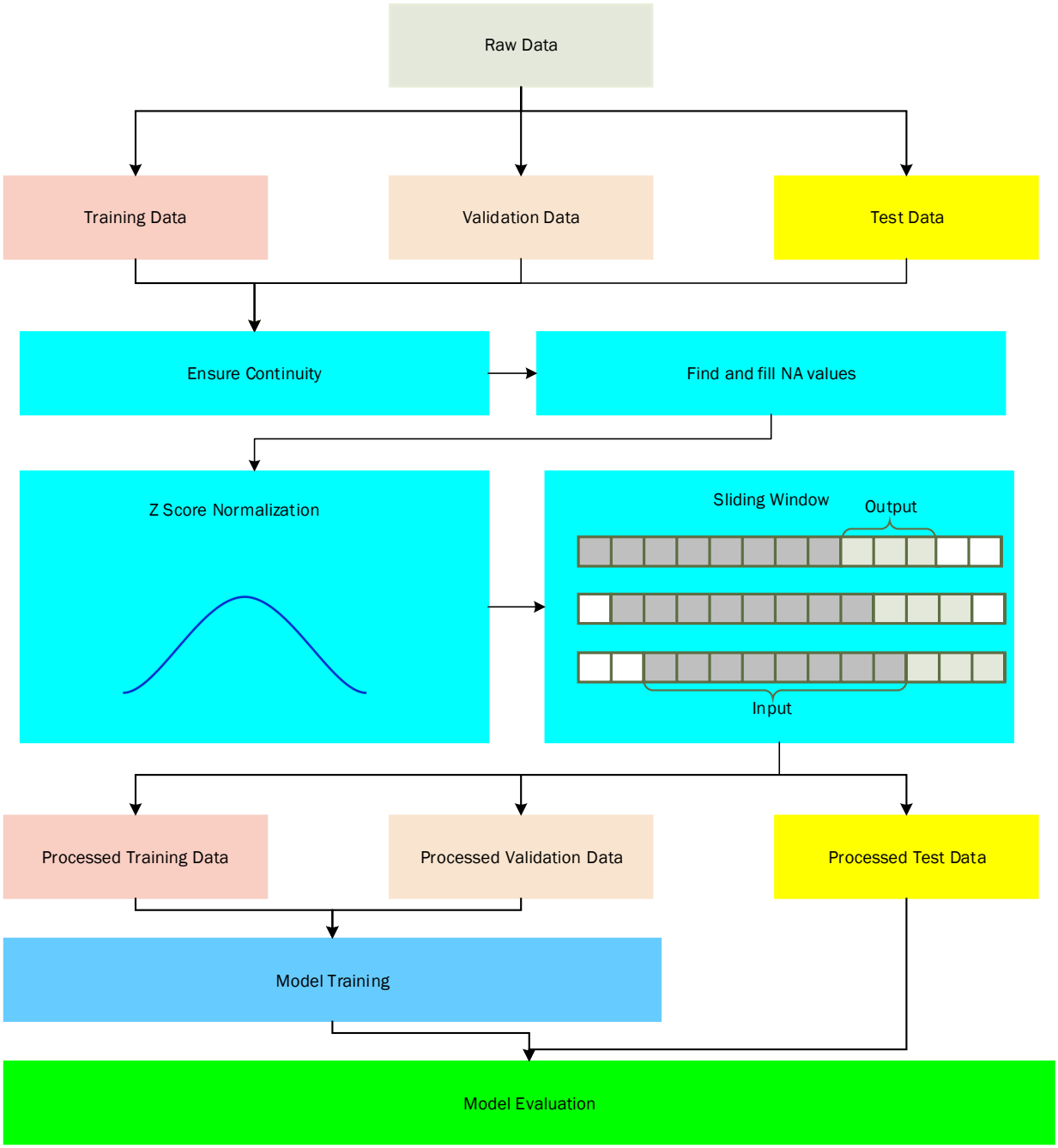


Figure 34: Basic Model Pipeline

and involves more advanced techniques. Data cleaning aims to ensure consistency and remove discrepancies in the data. Major procedures in data cleaning include:

a. Outlier Removal: Identifying and eliminating extremely large, extremely small, or implausible values (e.g. negative temperatures in Kelvin). Outliers can result from faulty instruments or data collection errors.

b. NA (Not Available) Values Removal: These values indicate gaps or breaks in the data and need to be addressed.

Outliers are detected using the interquartile range (IQR) method and are then marked as NA values. These NA values, along with any gaps, are filled using the forward filling method, which replaces breaks or outliers with values from the previous timestamp. The dataset is reanalyzed to ensure no missing values or outliers remain, and none are found after this process.

4.4.1.2. Data Preprocessing

In this step, **data standardization** is performed. **data standardization** is carried out for each feature separately. This process helps the model converge faster. It is also called Z-Score Normalization. It is more robust to any outliers in the data compared to data normalization. Equation 15 shows the basic equation for normalization of a single feature.

$$x_{normalized} = \frac{x - M}{\alpha} \tag{15}$$

Here M represents the mean while α is the standard deviation. This technique enhances the convergence speed of optimization algorithms by preventing larger values from overshadowing the learning process. It also aids in detecting outliers. Additionally, it reduces bias by ensuring that all features are on the same scale, minimizing the influence of features with higher magnitudes.

By standardizing the scale of data points, it facilitates easier comparison between different datasets and features.

Secondly, windowing function is implemented. Data windowing is done to create input-output pairs from a continuous time series. Since we are using the hourly input from last 7 days, we will be dividing the time series into continuous slices of 168 time steps while striding 24 time steps. The output of the network would be hourly prediction of one complete day (24 time steps).

After data windowing, we divide the dataset into 3 portions. These are training set, validation set and test set. 80% of the data makes up the training set while 10 % of the data is used for validation and finally the remaining 10% is used for testing.

4.4.1.3. Training

Finally, the training set is used to train the data. Huber loss is used along with Adams optimizer. Huber loss can be defined as:

$$\text{Huber loss} = \begin{cases} \frac{1}{2}(\text{P-G})^2 & \text{for (P-G)} \leq \delta \\ \delta \cdot \left(|\text{P} = \text{G}| - \frac{1}{2}\delta\right), & \text{otherwise} \end{cases} \tag{16}$$

This loss function is non-linear (quadratic) for small values of the residual (P-G) and linear for larger residuals. As a result, this function combines the sensitivity of Mean Squared Error (MSE) with the robustness of Mean Absolute Error (MAE). Huber loss is also more suitable for gradient based optimization methods as it provides a smooth gradient at 0 unlike MAE which is non-differentiable at that point. Huber loss is also quite flexible as changing the parameter δ allows the control over the transition between quadratic and linear behavior. Huber loss also helps with data that has a lot of noise as it offers a smooth transition between linear and quadratic loss.

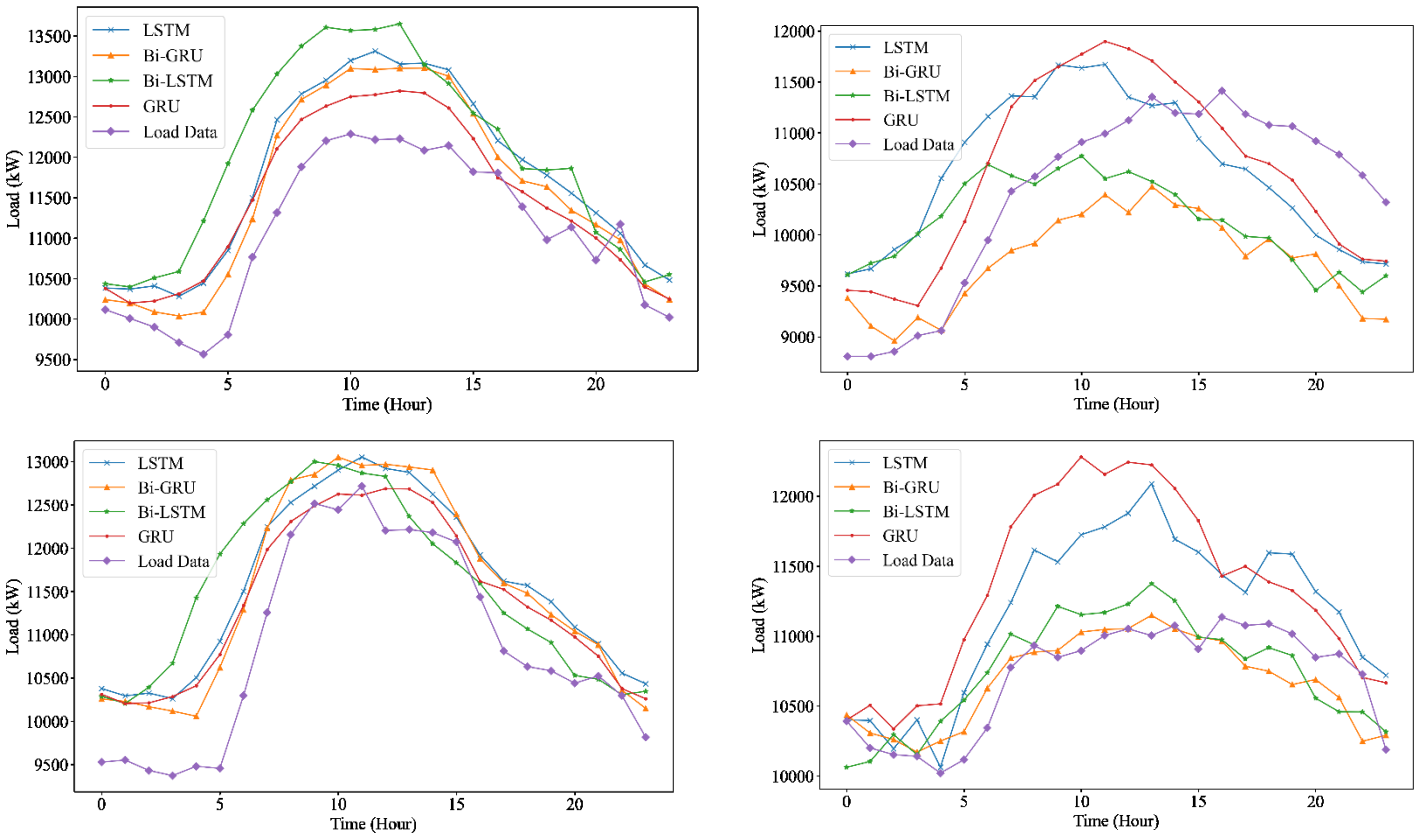


Figure 35: Predictions of various days using the algorithm

For comparison, each of the algorithms is trained up to 100 epochs. Early Stopping routine is implemented for a more efficient training process and save time. Validation loss is monitored and the best results are saved based on its minimum value. Finally Mean Absolute Error (MAE), Mean Squared Error (MSE) and R2 values are tabulated on the test data to evaluate the model performance. The architecture used for training of deep learning models is shown in Figure 10. We use 3 RNN units (either LSTM, GRU or their bidirectional counter parts), followed by 2 dense layers. The first dense layer has 128 units while the second one is the output layer with 24 units.

The system used for training and inference is a GTX 970m and i7 4720 HQ with 16 GB Memory.

4.4.2. Simulation Results:

Data cleaning and preprocessing were done in python 3.9 using pandas and Numpy. Keras was used to create the deep learning models and compare the various evaluation metrics. We can see graphs of electric load prediction by all the models. By looking at the prediction graphs in Figure 9, it can be seen that the models have learned the daily electric usage profile quite well. GRU and Bi-GRU models show better MAE and MSE compared to their counter parts (LSTMs and Bi-LSTMs). The Mean Absolute Error and Mean Squared Error are also given in the table below:

Table 18: Evaluation Metrics on test set

Algorithm	**LSTM**	**Bi-LSTM**	**GRU**	**Bi-GRU**
MAE (kW)	573.13	503.05	490.13	424.09
R^2 Score	0.654	0.73	0.731	0.790
MSE (kW^2)	530192.57	414496.58	411932.35	321941.44

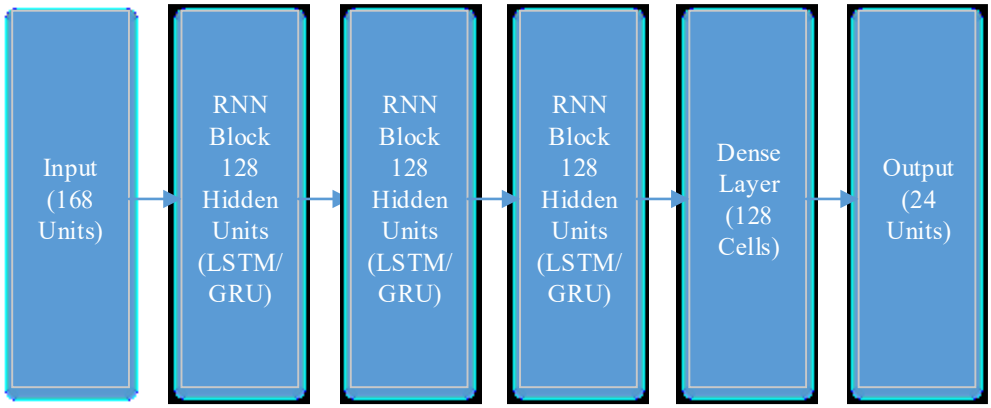


Figure 36: Basic Architecture Used for Deep Learning Algorithm

4.5. Conclusion

It can be seen that GRU and Bi-GRU algorithms perform better than the conventional LSTM and Bi-LSTM algorithms despite being simpler models. As LSTM and Bi-LSTM models have higher

complexity and higher number of learnable parameters, they tend to overfit the data while GRU and Bi-GRU are simpler models.

4.6. References

[1] A. A. Mamun, M. Sohel, N. Mohammad, M. S. Haque Sunny, D. R. Dipta, and E. Hossain, "A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models," *IEEE Access*, vol. 8, pp. 134911–134939, 2020, doi: 10.1109/ACCESS.2020.3010702.

[2] M. Bashari and A. Rahimi-Kian, "Forecasting Electric Load by Aggregating Meteorological and History-based Deep Learning Modules," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, Montreal, QC, Canada: IEEE, Aug. 2020, pp. 1–5. doi: 10.1109/PESGM41954.2020.9282124.

[3] W. Zhang, Q. Chen, J. Yan, S. Zhang, and J. Xu, "A novel asynchronous deep reinforcement learning model with adaptive early forecasting method and reward incentive mechanism for short-term load forecasting," *Energy*, vol. 236, p. 121492, Dec. 2021, doi: 10.1016/j.energy.2021.121492.

[4] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustainable Energy, Grids and Networks*, vol. 6, pp. 91–99, Jun. 2016, doi: 10.1016/j.segan.2016.02.005.

[5] Y. Lin, H. Luo, D. Wang, H. Guo, and K. Zhu, "An Ensemble Model Based on Machine Learning Methods and Data Preprocessing for Short-Term Electric Load Forecasting," *Energies*, vol. 10, no. 8, 2017, doi: 10.3390/en10081186.

[6] M. A. Hammad, B. Jereb, B. Rosi, and D. Dragan, "Methods and models for electric load forecasting: a comprehensive review," *Logistics, Supply Chain, Sustainability and Global Challenges*, vol. 11, no. 1, pp. 51–76, 2020.

[7] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, *An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting*. 2017. doi: 10.1007/978-3-319-70338-1.

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

[9] M. Madhukumar, A. Sebastian, X. Liang, M. Jamil, and M. N. S. K. Shabbir, "Regression Model-Based Short-Term Load Forecasting for University Campus Load," *IEEE Access*, vol. 10, pp. 8891–8905, 2022, doi: 10.1109/ACCESS.2022.3144206.

[10] X. Wang, W. Tian, and Z. Liao, "Statistical comparison between SARIMA and ANN's performance for surface water quality time series prediction," *Environ Sci Pollut Res*, vol. 28, no. 25, pp. 33531–33544, Jul. 2021, doi: 10.1007/s11356-021-13086-3.

[11] K. Akbar, Y. Zou, Q. Awais, M. J. A. Baig, and M. Jamil, "A Machine Learning-Based Robust State of Health (SOH) Prediction Model for Electric Vehicle Batteries," *Electronics*, vol. 11, no. 8, Art. no. 8, Jan. 2022, doi: 10.3390/electronics11081216.

[12] H. Ali, S. O. Gilani, M. J. Khan, M. Jamil, and M. K. Khattak, "Stacked Bin Convolutional Neural Networks based Sparse Low-Rank Regressor: Robust, Scalable and Novel Model for Memorability Prediction of Videos," *Multimed Tools Appl*, vol. 82, no. 26, pp. 40799–40817, Nov. 2023, doi: 10.1007/s11042-023-15128-z.

[13] M. Awais, L. Khan, S. G. Khan, Q. Awais, and M. Jamil, "Adaptive Neural Network Q-Learning-Based Full Recurrent Adaptive NeuroFuzzy Nonlinear Control Paradigms for Bidirectional-Interlinking Converter in a Grid-Connected Hybrid AC-DC Microgrid," *Energies*, vol. 16, no. 4, Art. no. 4, Jan. 2023, doi: 10.3390/en16041902.quation

[14] K. Zafar *et al.*, "Skin Lesion Segmentation from Dermoscopic Images Using Convolutional Neural Network," *Sensors (Basel)*, vol. 20, no. 6, p. 1601, Mar. 2020, doi: 10.3390/s20061601.

[15] A. R. Asif *et al.*, "Performance Evaluation of Convolutional Neural Network for Hand Gesture Recognition Using EMG," *Sensors (Basel)*, vol. 20, no. 6, p. 1642, Mar. 2020, doi: 10.3390/s20061642.

[16] U. Rashid, M. Jamil, S. Gilani, and I. Niazi, "LQR Based Training of Adaptive Neuro-Fuzzy Controller," vol. 54, 2016, pp. 311–322. doi: 10.1007/978-3-319-33747-0_31.

[17] S. Muzaffar and A. Afshari, "Short-Term Load Forecasts Using LSTM Networks," *Energy Procedia*, vol. 158, pp. 2922–2927, Feb. 2019, doi: 10.1016/j.egypro.2019.01.952.

[18] M. Jain, T. AlSkaif, and S. Dev, "Are deep learning models more effective against traditional models for load demand forecasting?," in *2022 International Conference on Smart Energy Systems and Technologies (SEST)*, Eindhoven, Netherlands: IEEE, Sep. 2022, pp. 1–6. doi: 10.1109/SEST53650.2022.9898424.

[19] M. Jawad *et al.*, "Machine Learning Based Cost Effective Electricity Load Forecasting Model Using Correlated Meteorological Parameters," *IEEE Access*, vol. 8, pp. 146847–146864, 2020, doi: 10.1109/ACCESS.2020.3014086.

[20] R. Schaeffer *et al.*, "Energy sector vulnerability to climate change: A review," *Energy*, vol. 38, no. 1, pp. 1–12, Feb. 2012, doi: 10.1016/j.energy.2011.11.056.

[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[22] P. T. Yamak, L. Yujian, and P. K. Gadosey, "A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, Sanya China: ACM, Dec. 2019, pp. 49–55. doi: 10.1145/3377713.3377722.

[23] Q. Tao, F. Liu, Y. Li, and D. Sidorov, "Air Pollution Forecasting Using a Deep Learning Model Based on 1D Convnets and Bidirectional GRU," *IEEE Access*, vol. 7, pp. 76690–76698, 2019, doi: 10.1109/ACCESS.2019.2921578.

[24] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," Mar. 22, 2013, *arXiv*: arXiv:1303.5778. Accessed: Nov. 14, 2022. [Online]. Available: http://arxiv.org/abs/1303.5778

Chapter 5: Conclusion and Future Work

5.1. Conclusion

Electric load forecasting is a challenging research problem. It has a great impact on maintenance planning, future expansions and organization policy. It can also help alleviate problems such as load shedding. An increase in forecasting error actually correlates with operating costs. Our work focuses on using state-of-the-art artificial intelligence based algorithms to optimize the electric load forecasting.

When comparing the performance of conventional load forecasting algorithms using recurrent neural networks, it is evident that GRU and Bi-GRU outperform LSTM and Bi-LSTM. Bi-GRU is the best-performing algorithm, followed closely by GRU. Since LSTM and Bi-LSTM have more trainable parameters, they are more prone to overfitting the data compared to Bi-GRU and GRU models.

Next we compare the sequence-to-sequence models to the conventional RNN models and present a novel sequence-to-sequence based network with attention. The model, trained on real-world data, demonstrates a higher R2 score and lower MSE, MAE, and MAPE compared to other conventional LSTM-based and GRU-based models. The impact of the attention mechanism is also clear, as the attention-enhanced model outperforms its counterpart without attention, achieving a higher R2 score and lower MAE, MSE, and MAPE. Additionally, an analysis was performed to identify optimal input window sizes while switching the basic RNN block from GRU to LSTM. The model with a GRU-based encoder-decoder and an 8-day (192 time steps) input window size achieved the best performance, with the lowest MAE, MAPE, and the highest R2 score. Robustness testing of the proposed method indicates that the model remains unaffected by added perturbations to the data.

Finally, the multi-energy systems are studied. We utilize the multi-task learning framework for load forecasting for such systems. This work introduces a new feature selection method and a novel TCN-based forecasting model for multi-energy systems. The feature selection method employs distance correlation to measure the coupling among heating, cooling, and electricity variables, selecting those with the highest correlation. The TCN-based network utilizes a multi-task learning approach to simultaneously forecast all three energy variables. The effectiveness of the proposed method is validated by a detailed, seasonally divided analysis, demonstrating increased accuracy compared to other algorithms. The proposed method significantly improves MAPE and RMSE compared to other state-of-the-art techniques.

5.2 Future Work

Future work for load forecasting would include investigating the performance of the novel electric load forecasting algorithm presented in chapter 2 on various other datasets. In our investigation we focused on real world data from a single MUN campus but investigating the algorithm on other datasets would provide valuable insight into the performance of the algorithm in much more comprehensive manner.

Another horizon for research would be to change the architecture and introduce a bi-directional encoder decoder instead of the standard encoder decoder and investigate its performance on multiple electric load datasets.

Encoder decoders have an advantage in generating arbitrarily long sequences of output. Use of attention mechanism would allow the network to handle long-range dependencies in a much better fashion. So using the proposed architecture or a modified version could be used for medium or long-term electric load forecasting.

For the multi-energy system based forecasting algorithm proposed in chapter 3, more research can be done on other multi-energy based datasets to investigate the performance. Another variation for the architecture would be to replace the TCN with an encoder decoder based model. Investigating the performance for this modification could allow for a better performing algorithm.

List of Publications

1.	Ahmed, Z.; Jamil, M.; Khan, A.A. Short-Term Campus Load Forecasting Using CNN-Based Encoder–Decoder Network with Attention. *Energies* 2024, *17*, 4457. https://doi.org/10.3390/en17174457
2.	Z. Ahmed and M. Jamil, "Campus Electric Load Forecasting Using Recurrent Neural Networks ," *2024 12th International Conference on Smart Grid (icSmartGrid)*, Setubal, Portugal, 2024, pp. 412-417, doi: 10.1109/icSmartGrid61824.2024.10578153.