The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

Anticipating Disruptions in Power Grid Frequency Data

by

Haoting Yu

Dissertation Presented for the Degree of
MSc in Operational Research with Data Science

August 2023

Supervised by
Dr. Amanda Lenzi

Abstract

In response to the increasing incorporation of renewable energy sources into power systems, this study is centered on the models suitable for anomaly detection and forecasting of system frequency. Drawing upon data sourced from the National Grid Electricity System Operator, the research aims to discern and pinpoint deviations in frequency patterns and predict ensuring system behaviors. Notably, the developed anomaly detection model achieved an accuracy rate of over 90%, emphasizing its reliability in detecting significant deviations. Concurrently, the forecasting model, designed around a CNN-LSTM architecture, demonstrated predictions with a deviation of merely 5%. Such results are indicative of their potential capability: they equip control engineers with robust tools for precise interventions, consequently enhancing the resilience of the power system against unforeseen disruptions. Given the challenges presented by the integration of renewable energy, the methodologies adopted in this study, encompassing both statistical and machine learning techniques, underscore the significance of refining our tools for better system management and forecasting.

Acknowledgments

I would like to thank my supervisor, Amanda, for her guidance and expertise throughout this project. My gratitude also goes to my family for their continuous support. Additionally, I appreciate the encouragement provided by my boyfriend, Xiangrui, during this journey.

Contents

List of Tables

List of Figures

1 Introduction

The following subsections provide a detailed account of the background, current solutions and their limitations, the proposed solution and its advantages, an overview of methodology, and the significance of this research.

1.1 Background

The pursuit of sustainable and reliable power systems is a key global concern. Electricity as one of the most wild-used energy, is distributed throughout Great Britain, and it's essential to regulate properties like voltage and frequency across the entire network. This regulation ensures that power produced in large-scale power stations can be safely utilized by domestic appliances connected to wall sockets. In this paper, we will use frequency as a primary measure to evaluate the operation and stability of the power grid system.

Frequency, in the context of electricity, refers to the number of times current oscillates between positive and negative voltage per second. For example, when an appliance such as a kettle or laptop charger is turned on, it operates on alternating current. This current "oscillates" between positive and negative voltage, a phenomenon known as electrical frequency. In the UK, this oscillation occurs 50 times per second, hence the frequency is defined as 50 hertz (50Hz). All appliances and electrical equipment in the UK are designed to operate at this frequency.

The National Grid Electricity System Operator (NG ESO) has the responsibility of preserving frequency within a narrow range around 50Hz. Deviations from this frequency can lead to equipment malfunction. Their operational target is to maintain frequency within 0.2Hz of 50Hz, a more stringent standard than the official requirement of 0.5Hz. In order to keep frequency around 50Hz, one key challenge is to maintain the balance of supply and demand, which is a fundamental aspect of power system reliability. The equilibrium of electricity supply and demand directly influences frequency. When demand surpasses supply, frequency descends; conversely, when supply outstrips demand, frequency ascends [9]. As the UK gravitates towards an increased reliance on zero carbon energy sources, frequency management has become more intricate due to the swifter responses to changes in supply and demand. Therefore, it's crucial to continuously monitor the frequency across the entire electricity network in Great Britain to ensure it remains close to 50Hz at all times. Nevertheless, this is a challenging task for the ESO. They must have specialists continually monitoring the frequency, a task so demanding that it necessitates a rotation every hour to ensure sustained attention and precision. Hence, it's important to develop and implement a tool that can aid these specialists in their frequency monitoring tasks. This tool would not only reduce the intense workload on the specialists but also improve the precision and effectiveness of the monitoring process, contributing to the overall stability of the power grid. In this context, the identification of frequency anomalies becomes crucial. Anticipating the trajectory of data changes not only provides valuable lead time for engineers but also enables proactive measures, enhancing system reliability and efficiency. Given the temporal variation of frequency, a wide range of time series analysis techniques are applicable in this project.

1.2 Objective and Contribution

The primary objective of this dissertation is to develop an effective method for anomaly detection and future trend prediction in NG ESO power grid frequency data. By leveraging machine learning techniques and time series analysis, we aim to provide a tool that can help engineers maintain the stability of the power grid and respond promptly to unexpected fluctuations. In terms of anomaly detection, we're proposing an innovative approach that combines autoencoders with recurrent neural networks (RNNs). Autoencoders, a type of artificial neural network, are adept at learning efficient encodings of input data. Meanwhile, RNNs, with their looped structure, can maintain information over time, making them suitable for time series data like ours. In our methodology, we will provide a more detailed explanation of both autoencoders and RNNs. However, it's crucial to understand that by integrating autoencoders with RNNs, we aim to construct a model that can effectively identify anomalies in the frequency data. We're also testing different types of RNNs, including Long Short-

Term Memory (LSTM) networks [16]. By evaluating the performance of these different models on our dataset, we aim to determine the most effective model for our specific task. This comparative analysis is a fundamental part of our research, as it will offer valuable insights into the applicability of these advanced neural network structures for power grid frequency data analysis. In terms of forecasting future trends, our approach employs both a benchmark model utilizing ARIMA (Autoregressive Integrated Moving Average) and a novel model that integrates an encoder-decoder structure with the combination of Convolutional Neural Networks (CNNs) and LSTM networks. This dual approach of anomaly detection and future prediction forms the core of our research, offering a comprehensive solution to managing the stability of power frequency data. In summary, the main contributions of this article are as follows:

- We developed an anomaly detection framework designed specifically for power grid frequency data. Given the limited existing literature on anomaly detection on this type of data, our work offers valuable insights for future research in this area. Additionally, we conducted a comparison of different deep learning architectures within auto-encoders, including SimpleRNN and LSTM, to identify the most effective model for anomaly detection in our study's context.

- We purposed a novel model using deep learning methods to forecast future frequency patterns. Despite the challenges presented by the uni-variate nature of our data, the model achieved decent results. Notably, Given the significant variations of data at each time stamp, this model offers future researchers a perspective on the potential of applying deep learning techniques to this kind of data.

- We found the ARIMA model's performance limited when applied to our volatile dataset. For short-term predictions, neural networks are notably flexible in handling this type of data.

1.3 Related Work

In the initial stages of anomaly detection research, the focus was primarily on statistical methods. Approaches like founding statistical characteristics as thresholds and autocorrelation are frequently used to detect anomalies in both uni-variate and multi-variate time series data [3]. In addition to this, clustering-based methods and hybrid unsupervised machine learning approaches have often been employed for identifying anomalies, introducing a data-driven solution to the issue, as highlighted by Pu et al. [23] and Riyaz et al. [10]. Meanwhile, taking use of deep learning algorithms has became common on anomaly detection tasks [17]. For instance, Recurrent Neural Networks (RNNs) and their advanced derivatives, like Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) networks, have gained recognition [2, 25, 11]. These networks differ from traditional Convolutional Neural Networks (CNNs) in that they have the ability to retain information from previous inputs in a sequence. This feature allows them to use this past information when processing current and future inputs, making them especially suitable for tasks involving sequential data, such as time series analysis. These models perform well in handling temporal dependencies in data, making them highly effective for anomaly detection tasks, [7, 29]. Parallel to these advancements, the integration of RNNs with autoencoders represents another significant development. This hybrid approach combines the strengths of both models to form a potent and efficient method for anomaly detection. The autoencoder's ability in data reconstruction, alongside the RNN's skill in managing temporal sequences, fosters an enhanced performance in detecting anomalies [6, 26, 13].

In terms of forecasting, predicting the trajectory of frequency data provides us with valuable insights into future performance. It's particularly beneficial to identify unexpected fluctuations ahead of time, enhancing our preparedness and response. Investigations into the dynamics and stability of power frequency data have been carried out using a variety of methodologies. Auto-regressive models, for instance, have been widely used due to their ability to capture the linear predictability in the data [4, 32, 15]. These models are particularly effective when dealing with data that exhibit a strong correlation with their past values. Stochastic models, on the other hand, have been employed to account for the inherent randomness in frequency data [9]. These models, which incorporate a degree of randomness in their predictions, are well-suited for handling the unpredictable nature of power frequency data. Additionally, some research has been undertaken to forecast both spatial and temporal aspects of the power grid frequency using spatiotemporal techniques [19]. Nearest neighbor algorithms have also been utilized in the analysis of frequency data [18, 22]. It works by identifying patterns in the data that are similar or "near" to each other, making them useful for detecting anomalies or shifts in the data patterns.

However, In this study, our focus shifts from a broad examination of the data's overall behavior to a more targeted analysis of the frequency data's patterns within a target period of time.

For power grid frequency data, there has been limited research aimed at investigating or detecting unusual values. The concept of thresholding is a viable strategy, as we are already familiar with the typical range of the performance of frequency data. However, anomalies in time series data may not be readily identifiable based solely on the individual values of the data points. An anomaly could be a data point or a series of points that deviate from the common collective pattern, or in another word, it could represent a change in behavior over time [8]. This highlights the need for an approach that can capture these subtle changes and identify anomalies based on the overall pattern and temporal dynamics of the data, not just individual data points. On the other hand, while there has been some research focused on forecasting future frequency, such as the work by Johannes et al. [18] who introduced system-specific null model based on daily profile, however, there is a lack of attention to short-term variations of the data, such as sudden spikes in frequency, which are crucial for maintaining system stability as allowing engineers to make quick interventions or actions.

1.4 Organization of the Paper

In this dissertation, we first introduce the study by outlining the background, stating the goals, reviewing related work, and describing the organization of the paper in Chapter 1. Chapter 2 focuses on preliminary data analysis, where we detail the data sources used, selection criteria, and perform an exploratory data analysis. Chapter 3 delves into anomaly detection, examining methodological approaches like Recurrent Neural Networks, Long Short-Term Memory Networks, and Autoencoders. This chapter also outlines the experiments conducted, including data sampling, pre-processing, and the training phase, followed by an evaluation of performance. Chapter 4 presents different forecasting techniques, starting with a benchmark model, its methodology, and an analysis of its performance. This is followed by an in-depth look at a neural network-based forecasting model, detailing its methods, experiments, and a thorough evaluation. Finally, Chapter 5 and 6 concludes the dissertation with a comprehensive discussion of the findings. Finally, there is a discussion which includes the insights gained from the study and some limilations.

2 Preliminary Data Analysis

In this section, the dataset is thoroughly examined, and an analysis is conducted on the entire dataset to identify the proportion and temporal position of anomalies. Based on the anomalies' temporal positions, a specific period of interest for modelling is then selected. An exploratory analysis is conducted on the selected data sample.

2.1 Data Source

The actual data comes from the National Grid Electricity System Operator Limited(NG ESO), the electricity system operator for Great Britain. The data set covers a period from January 1, 2014, to May 30, 2023 which is almost a decade of continuous recordings, providing an insight on the overall dynamic and behaviour of the electricity frequency.

Each entry in this dataset is marked with a timestamp in the "YYYY-MM-DD HH:MM:SS" format, as shown in figure 1. This consistent recording every second offers a high-resolution view into the power grid frequency changes. Such granularity can potentially reveal daily usage patterns, impacts of seasonal changes, and even short-lived anomalies or spikes that may occur due to unforeseen events or grid disruptions.

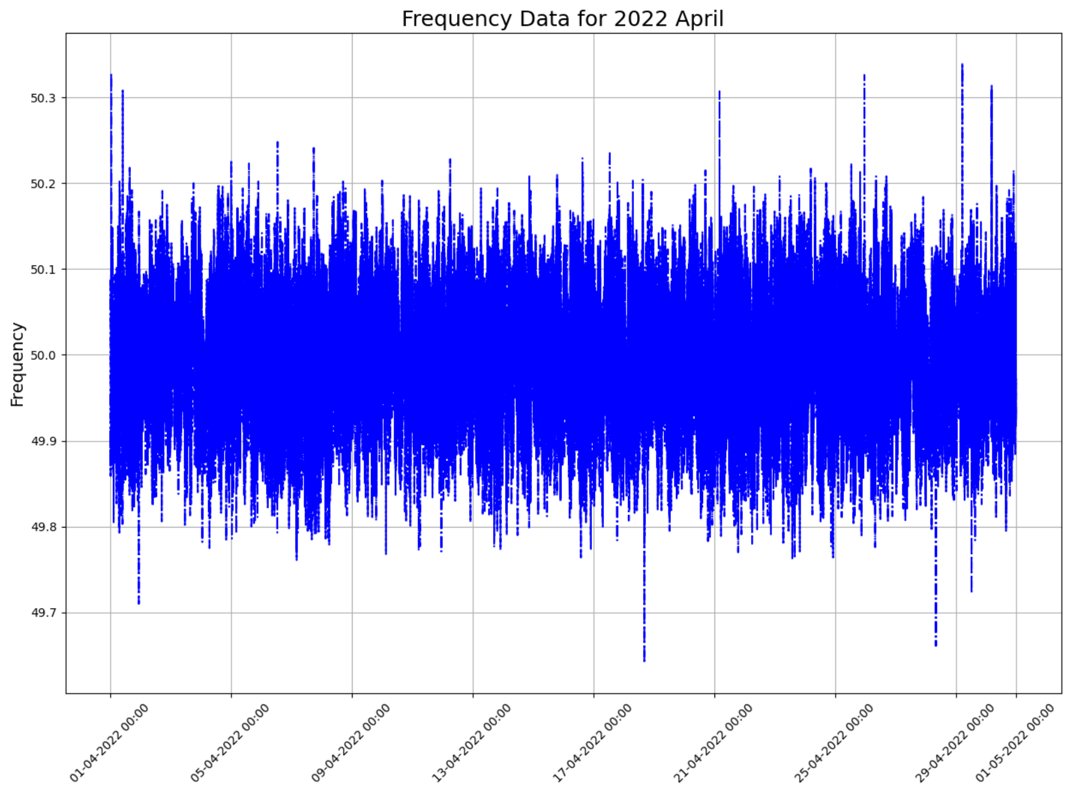


Figure 1: **Frequency Data for 2020 Feburary in second-resolution**.

As we proceed with the analysis, we aim to unpack these patterns, variations, and anomalies in the data. This in-depth exploration will lay the foundation for our subsequent tasks of anomaly detection and predictive modelling.

2.2 Data Selection

Visualizing extensive datasets, especially those with a high level of granularity as in our case, has certain challenges. Directly plotting near a decade of second-by-second frequency data would result in an overcrowded graph, making it difficult to discern meaningful patterns. Since the the NG ESO has provided guidelines for two target intervals within our data, we can take use of them to select the data of interest. These intervals are described as:

- Target Range 1: Spanning from $[50 - \alpha_1, 50 + \alpha_1]$, where $\alpha_1 = 0.5$.
- Target Range 2: Spanning from $[50 - \alpha_2, 50 + \alpha_2]$, where $\alpha_2 = 0.2$.

While both target ranges highlight the desirable bandwidth during operations, and it should be noted that Target Range 2 sets more restrictive boundaries compared to the 1.

To obtain more manageable datasets, we can utilize the target range and proceed with the subsequent steps:

Step 1: Begin by applying the forementioned target ranges as filters. This will allow us to isolate and select the period of interests to our study. By employing these target intervals, we aim to concentrate on the core components of our dataset.

Step 2: After isolating the key data, we then proceed to downsample it with an aim towards forecasting. This step is crucial in reducing the dataset's resolution, allowing for clearer long-term modelling without the complications presented by the original high-resolution data.

In the initial step, the entire dataset is examined to determine the total number of data points and pinpoint those that fall outside our defined ranges. This analysis will help us assess the overall structure, understand which data points deviate, and identify when these discrepancies occur.

The following plot figure 2 clearly show the data points outside of our defined ranges for each year by range 2.

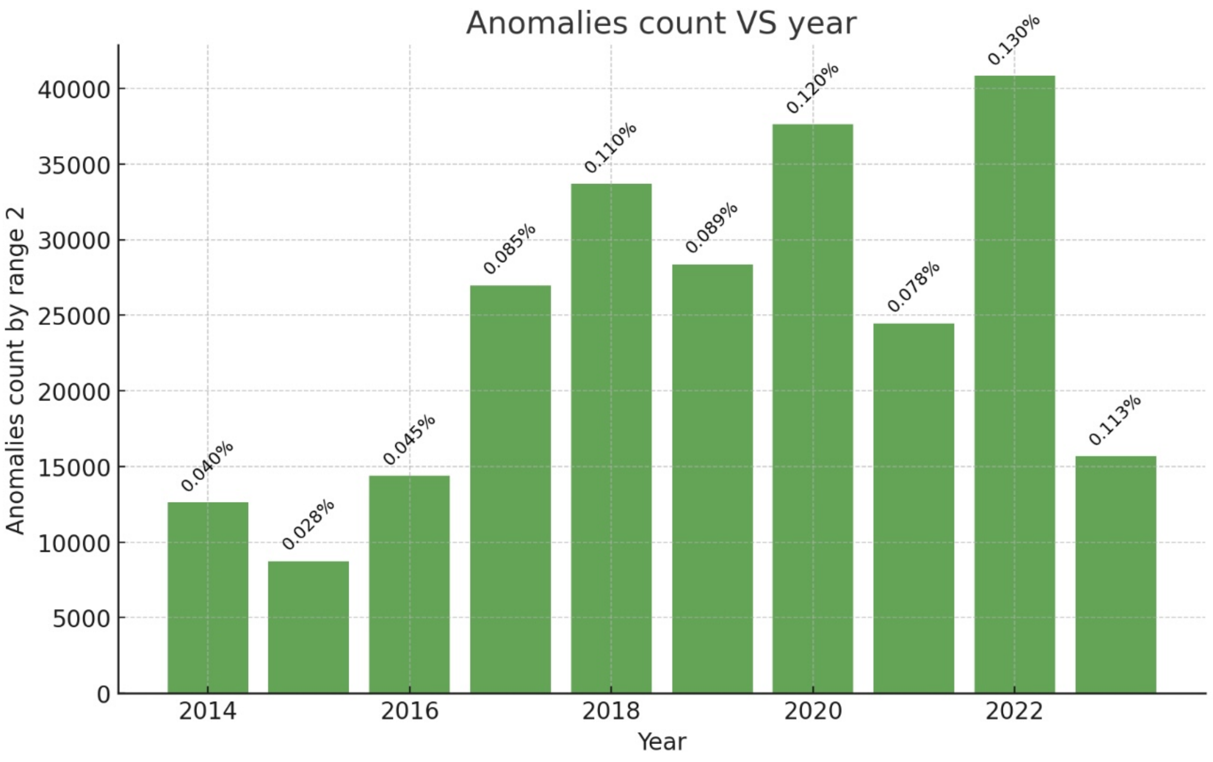


Figure 2: **Anomalies counts filtered by range 2 for the past decade**.

And we also include a table, displaying a breakdown of the total data points, anomalies, and their percentages for each year, as shown in the table 1.

From 2014 to 2022, there is a general upward trend in the histogram, with 2022 recording the highest count at 40,829 and 2015 the lowest at 8,741. Notably, 2017 witnessed a significant jump in anomalies from the previous year. In terms of percentages, which represent the ratio of anomalies to some unspecified total count per year, 2022 also had the highest at 0.130% while 2015 had the lowest at 0.028%. The years 2019 and 2020 show an interesting pattern: a drop in 2019 followed by a sharp rise in 2020, hinting at potential changes in data collection or genuine increases in anomalies. Data

year	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023 (to May)
Anomalies by range 1	0	0	0	0	0	142	0	0	0	0
Anomalies by range 2	12622	8741	14386	26979	33675	28362	37619	24467	40829	15687
Percentage by range 2	0.040%	0.028%	0.045%	0.085%	0.110%	0.089%	0.120%	0.078%	0.130%	0.113%

Table 1: **Anomalies and Percentage over Years.** A table of stating the total number of data of the year, the anomal points, the percentage.

for 2023, although only available until May, suggests a high anomaly count for the year if the current trend persists. The consistency in percentages for 2018 and 2019 suggests stable conditions during this period. Overall, while certain patterns emerge, understanding the root causes and implications of these anomalies would require a deeper look into external factors, data collection methods, and changing conditions over the years.

Based on the current information we have, the year 2019 stands out distinctly as it has 142 data points that fall outside Target Range 1, while no other year in the dataset has any anomalies for this range. This suggests that 2019 experienced a unique set of conditions or events that caused a significant number of data points to deviate from the narrower range. When we look at the percentage of anomalies, 2019 had 0.089%, which is not the highest, but still notably higher than its predecessor, 2018, and its successor, 2020. This underscores the unique characteristics of 2019 in terms of anomalies.

Upon examining the data for 2019, August stands out distinctly. In this specific month, all the anomalies, aligned with Range 1 criteria, occur. The subsequent plot illustrates these specific instances of anomalies. As depicted in the graph 3, these anomalies manifest as spikes.

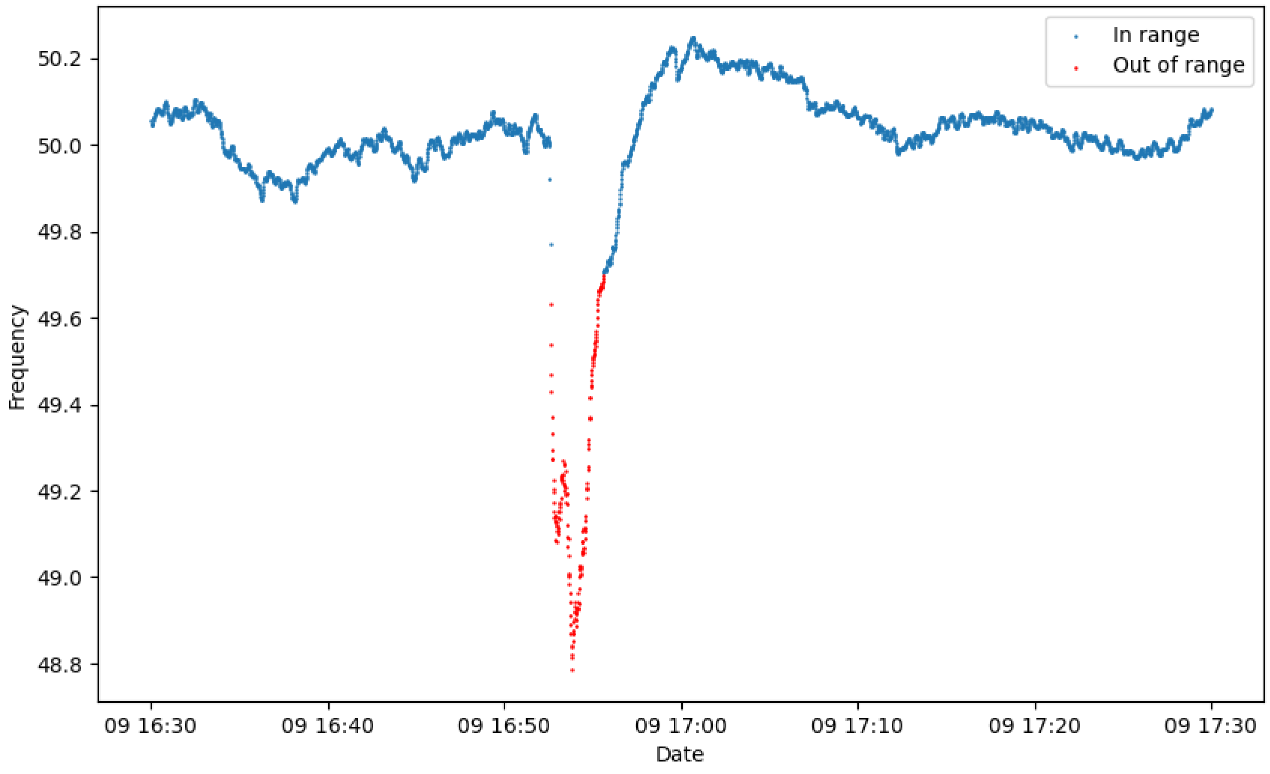


Figure 3: **The plot of the anomaly spikes**. All the anomalies filtered by range 1 occured at this specific interval.

To promptly detect anomalies as soon as they emerge and raise an immediate alert for engineers, an anomaly detection model can be developed specifically for this scenario. Such a model would serve as an early warning system, enabling engineers to take swift corrective actions. Thus, an anomaly detection model as explained in details in section 3 is constructed on a per-second basis.

Apart from the alert system, predicting the data trajectory also offers substantial benefits. This foresight provides valuable insights in advance, facilitating informed decision-making and effective planning. However, the original granularity of the selected dataset, with second-by-second observations, can be both a blessing and a challenge. On the one hand, it captures second fluctuations and details, offering a granular view of the data's behaviour. However, when the focus shifts to forecasting, it's essential to discern these broader patterns without getting entangled in the intricacies of short-term variations. Hence, we proceed step 2 to downsampling the data in a min-by-min basis by selecting the medium value recorded within each one-minute window from the original second-by-second dataset.

Here, we define the data set with the raw data as $D1$ and the downsampled data as $D2$.

2.3 Exploratory Analysis

Here we perform some basic analysis including visualizations, calculation of descriptive statistics as well as seasonal decomposition to have a better understanding on the selected dataset. The first step in our analysis involves understanding the general structure and characteristics of our dataset. To facilitate this, the subsequent plot 4 provides an overview of the entire dataset, with a detailed focus on a randomly selected day for a closer examination.

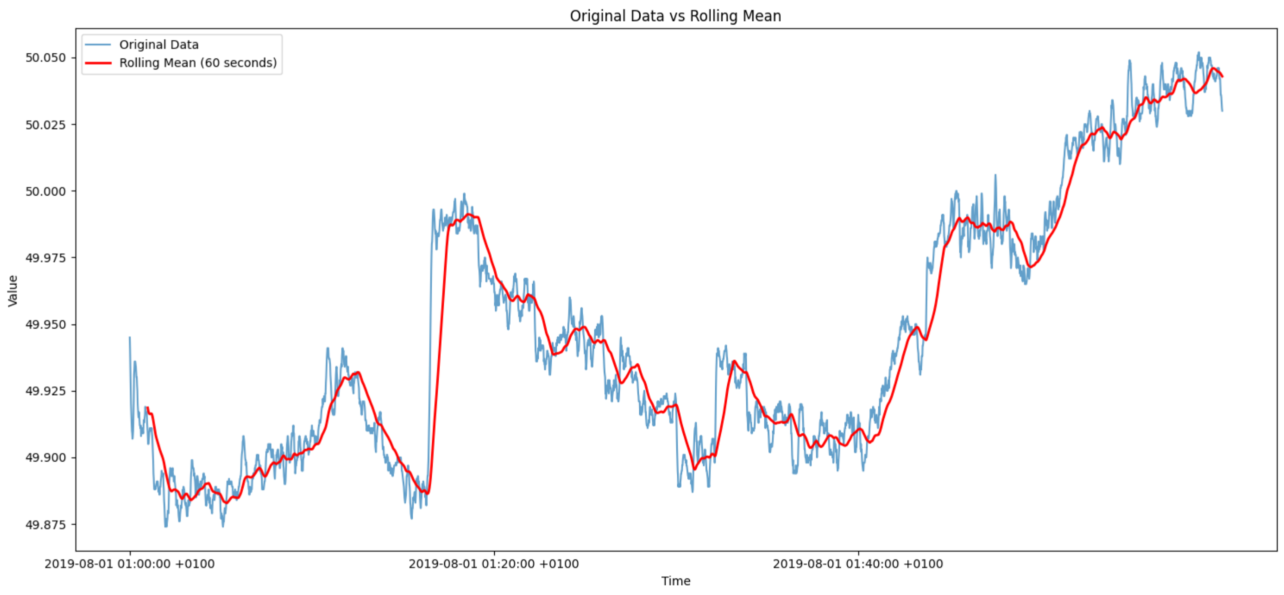


Figure 4: **Random select day from $D1$ with a rolling mean of 60 seconds.**

From the moving average evident in the selected chunk, it's clear that the dataset exhibits non-stationarity. This observation indicates that the statistical properties of our series, such as mean or variance, evolve over time. Before proceeding with modelling, it's imperative to address this non-stationarity, as many time series models require stationary data for reliable predictions. Techniques like differencing or detrending can be employed to stabilize such datasets.

Statistics	mean	std	min	25%	50%	75%	max
Value	50.02	0.059	49.268	49.977	50.023	50.062	50.274

Table 2: **Descriptive statistics from $D1$.**

As we further delve into the data, it's also crucial to understand its broader characteristics. To this end, the succeeding section presents a table with key descriptive statistics in table 2, complemented by a histogram. These tools help paint a comprehensive picture of the data's overall distribution and central tendencies, as shown in figure 5.

According to the statistical characteristics, we can see that the data's distribution is relatively symmetric around the central value with no appearance of significant skewness. It implies that extreme values on both sides of the mean are balanced and that there are not substantial deviations pushing

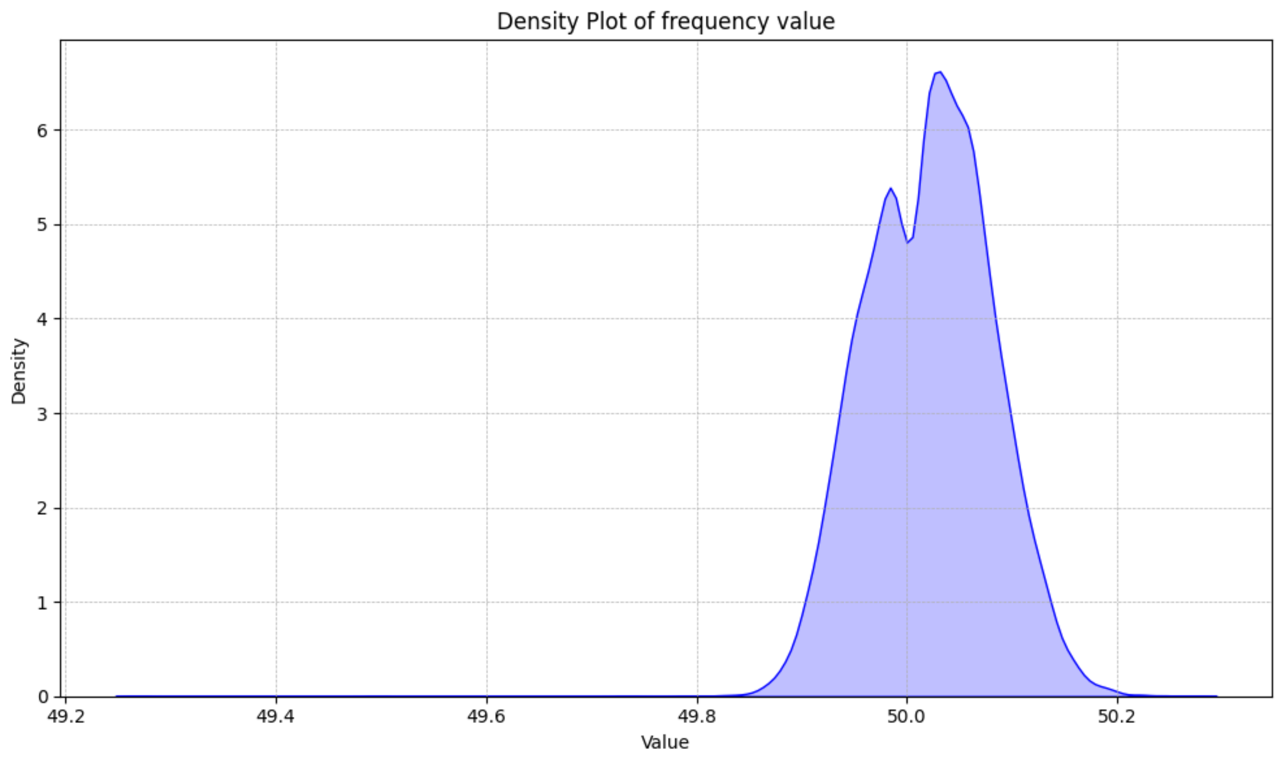


Figure 5: **Density plot for frequency value for $D1$.**

the data in one particular direction. It can be seen that there's a long tail on the left indicating that the outliers located below the average.

Building on this foundation, our next step is to explore the seasonal decomposition of the data as shown in figure 6. Instead of using the original selected dataset, here the downsampling dataset is used to save computational power. Also, since for high temporal resolution data, such as second-by-second readings, often contain substantial noise that can obscure underlying trends and patterns.

From the seasonal decomposition plot 6, it's evident that the data maintains a relatively steady course, with only minor fluctuations evident in the trend component. As the month progresses, a slight upward inclination can be observed, indicating subtle changes in the variable represented. As anticipated, some seasonality is present, manifesting as a daily cycle. This cyclical pattern consistently repeats itself each day, showing two distinct peaks and troughs within a 24-hour period. The residuals are the deviations of the actual time series from the combined trend and seasonal components. They appear to be centred around zero, indicating the decomposition has captured most of the underlying structure.

After analysing the total trend and seasonality of the whole dataset,we can also check the ACF (Autocorrelation Function) to find if there's any correlation in my time series data within a day. ACF can measure the correlation between a time series and its lagged version, in another word, it can tell how a value in a series is correlated with its past values, as shown in figure 7. We randomly selected a day's data from $D2$ to construct an ACF plot, enabling an analysis of the time-based correlation.

In the ACF plot 7, the autocorrelation at lag 0 is inherently 1, denoting a series' perfect correlation with itself. As the number of lags increases, the autocorrelation values exhibit a decline, indicating reduced correlation between data points spaced further apart in time. The blue shaded region indicates the confidence intervals, and any lag with autocorrelation values outside this boundary is deemed statistically significant. Notably, some lags surpass this threshold, suggesting their relevance in the data's variability.

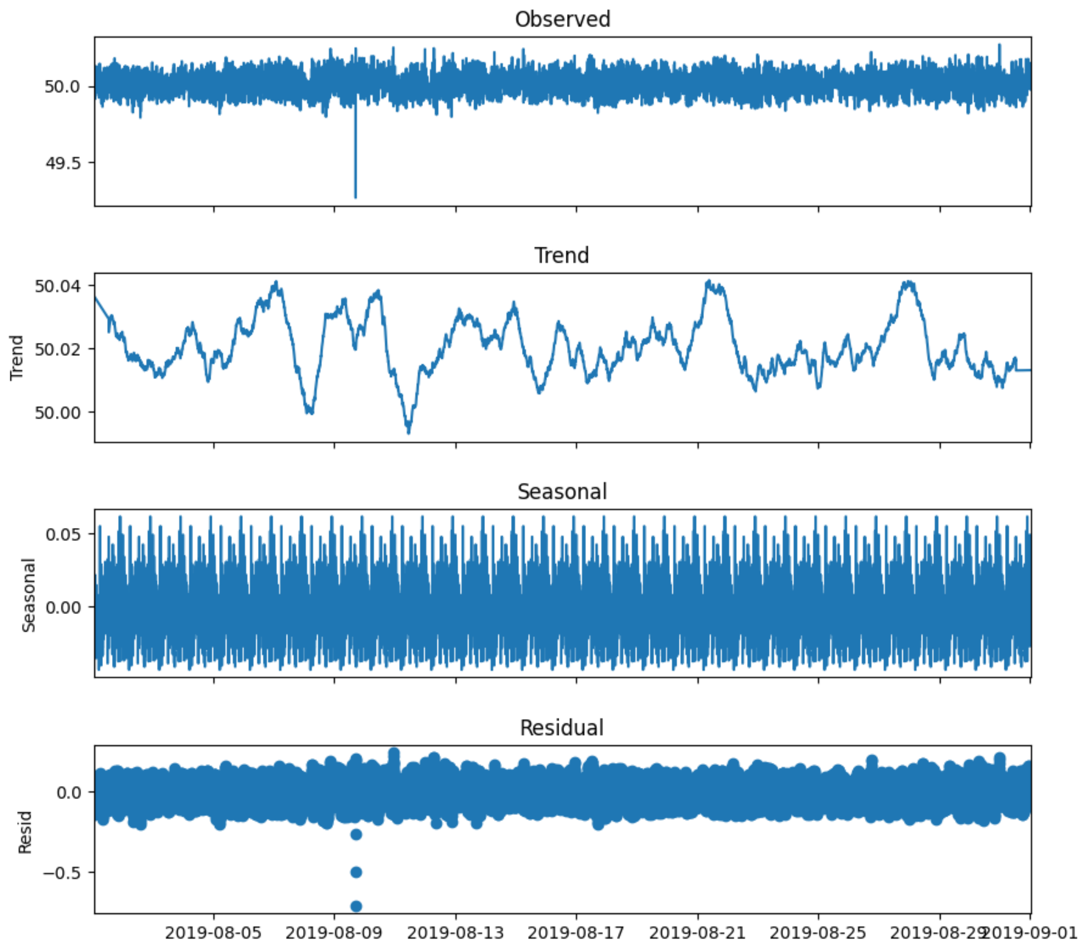


Figure 6: **Seasonality decomposition for $D2$.**

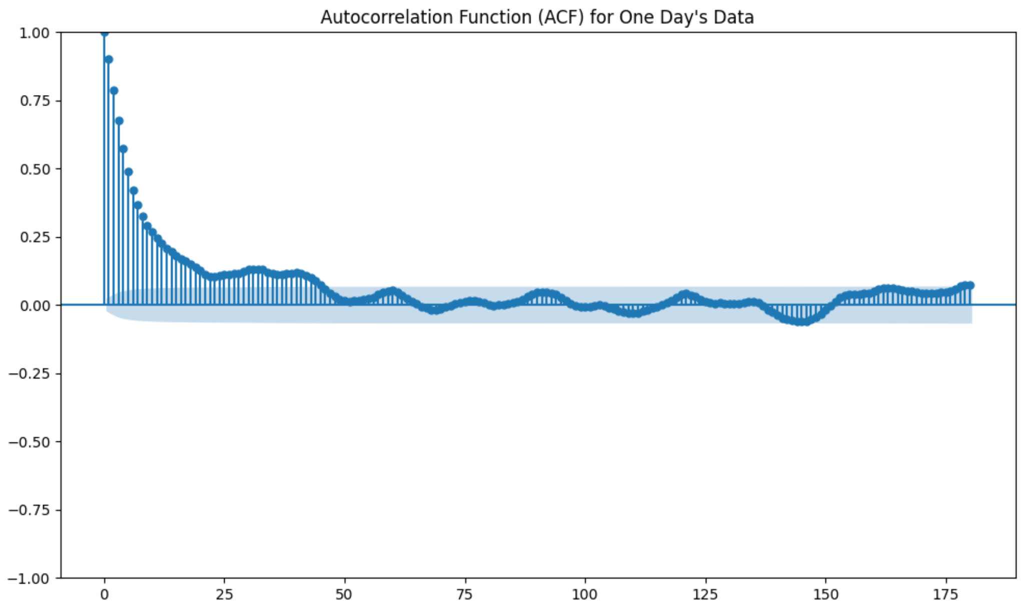


Figure 7: **Autocorrelation plot with $maxLag = 180$ for a random selected day from $D2$.**

3 Anomaly Detection

This section provides a comprehensive overview of the RNN-based approach to anomaly detection. Section 3.1 delves into the basic architecture of RNNs and elaborates on the training methodologies associated with the RNN-centric predictive anomaly detection autoencoder. In Section 3.2, we illustrate the application of our proposed method on a chosen sample. The concluding section offers a concise summary of the findings.

3.1 Methodology

3.1.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a specialized category of artificial neural networks characterized by connections between nodes that form a directed loop. Originally, Recurrent Neural Networks (RNNs) were introduced with tasks like Natural Language Processing (NLP) [1]. However, their inherent design, characterized by sequences and temporal dynamics, has proven to be especially effective for managing sequential data. As such, they've emerged as a preferred choice for time series analysis [28]. A generic RNN structure is shown in figure 8 and figure 9.

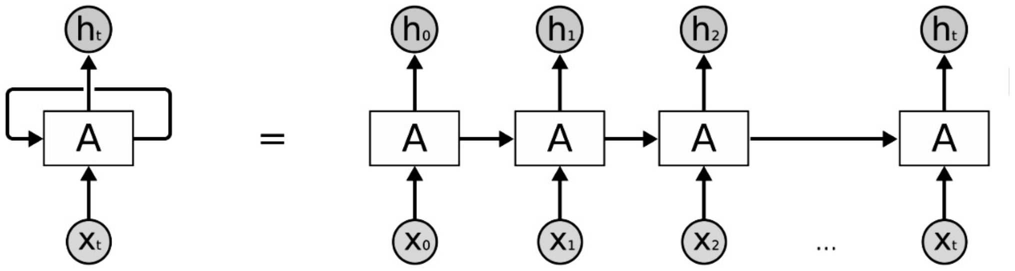


Figure 8: **RNN cell chain.** The figure shows how the recurrent neural network forms a loop by recurrently taking the output back into the cell and there is also an unfolded version of it.

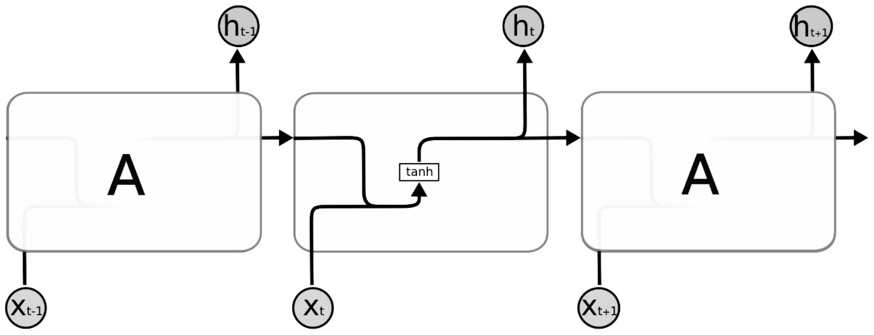


Figure 9: **RNN single cell.**

Inputs Each RNN cell takes in two main inputs: the current element from the sequence (denoted as x_t). The Hidden State from the previous time step (denoted as h_{t-1}) and it serves as the RNN's memory, capturing information from all previously seen elements in the sequence up to that point.

Operations Inside the cell, these inputs are processed with the aid of weighted matrices and the activation function, usually the tanh function 19 in the context of classic RNNs, plays a crucial role in this process as follows:

The recurrent update equation for the RNN is given by:

$$h_t = g_w(h_{t-1}, x_t)$$

Where h_t is the hidden state at time t, x_t is the input at time t, g_w is a function parameterized by weights w.

By keeping the output values limited within a range of -1 to 1, we ensure that the model remains consistent when processing sequential data. This focuses the model on genuine and significant trends or patterns, rather than being influenced by data from a specific moment.

Outputs The hidden state for the current time step is described as the "updated memory" and is represented by h_t. In parallel, the output for the current time step, denoted as y_t, can either match the hidden state or derive from a different function within the cell's operations.

However, while RNNs excel at capturing short-term dependencies due to their temporal dynamics, they encounter difficulties when trying to learn from long-range dependencies in sequences due to the gradient vanishing problem.

Since the inner workings of RNNs can be likened to a training process. Consider training a pet: immediate feedback post-action ensures effective learning. However, if the feedback is delayed, the pet might fail to associate its actions with the consequences, making the training ineffective. Similarly, when RNNs process sequences of data, they rely on gradients to adjust and improve their performance, akin to the feedback in our pet training analogy. Gradients represent the error or the adjustments the model needs to make based on its predictions. However, as RNNs delve deeper into sequences, especially long ones, the gradient - or the feedback signal - tends to diminish in strength. To address the vanishing gradient challenge inherent to traditional RNNs, Long Short-Term Memory (LSTM) networks were developed.

LSTM

Long Short Term Memory(LSTM) as a variant of recurrent neural networks, it has the same overall structure with RNN but LSTM cell has a more intricate design than the traditional RNN cell [12]. Here's a figure 10 showing a LSTM cell.

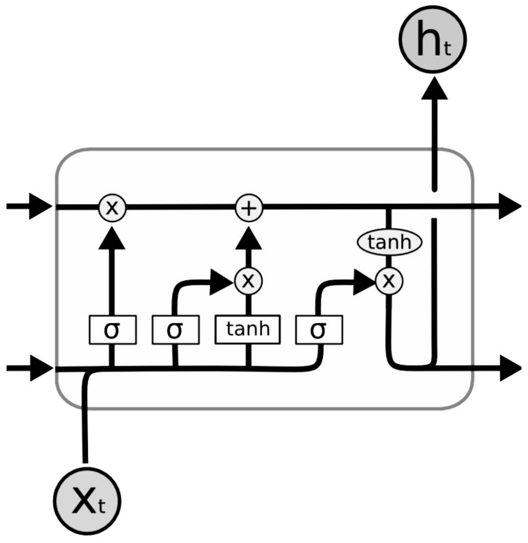


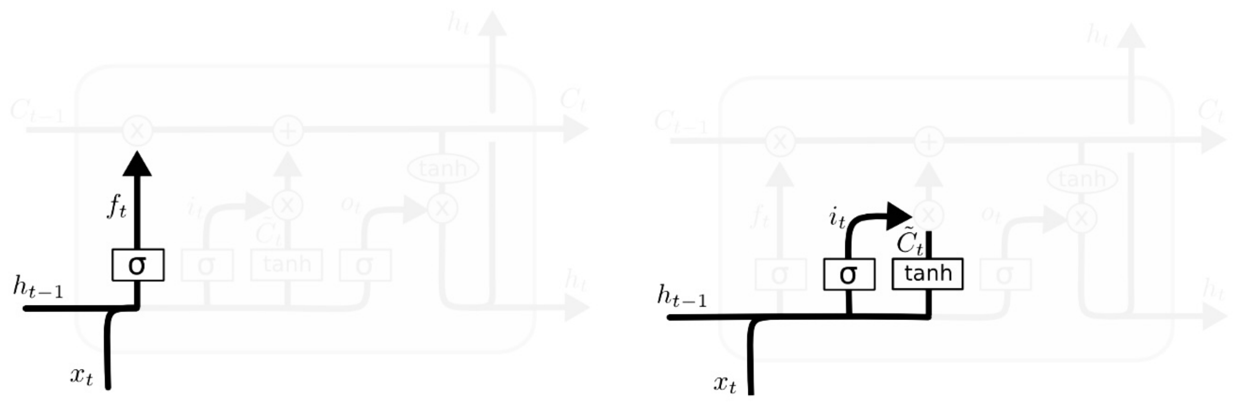
Figure 10: **Schematic representation of LSTM single cell.**

The main difference between a simple RNN cell and an LSTM cell is the introduction of gates and a dedicated cell state in the LSTM, which together enable more controlled and sustained memory retention across sequences.

Inputs Each LSTM cell takes in the current element from the sequence (x_t), the Hidden State from the previous time step (h_{t-1}) and the Cell State from the previous time step (c_{t-1}), acting as a longer-term memory than the hidden state.

Operations Inside the LSTM cell, these inputs are modulated by three gates:

- Forget Gate: Decides which information from the cell state should be deleted or kept.
- Input Gate: Updates the cell state with new values.
- Output Gate: Based on the cell state and the input, decides the next hidden state.



(a) Forget gate layer (b) Input gate layer

Figure 11: **inner working mechanism of single LSTM cell: forget gate and input gate**

These gates widely utilize sigmoid activation functions [refer to figure], which limit their outputs to a range between 0 and 1. Consider Forget gate as an example: As shown in Figure 11 plot (a), forget gate (f_t) decide what information will be kepted and deleted.

$$\text{Forget gate: } f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

As the out put is between 0 and 1, a value close to 0 suggests to the model to disregard a particular piece of historical data as it moves forward, while a value near 1 emphasizes the importance of retaining that specific information.

During the input processing phase as shown in Figure 11(b), the input gate (i_t) evaluates the significance of the new data in relation to past values. For example, in the context of a decreasing trend, the input gate might deem a newly introduced low value as highly significant, thereby giving it substantial weight.

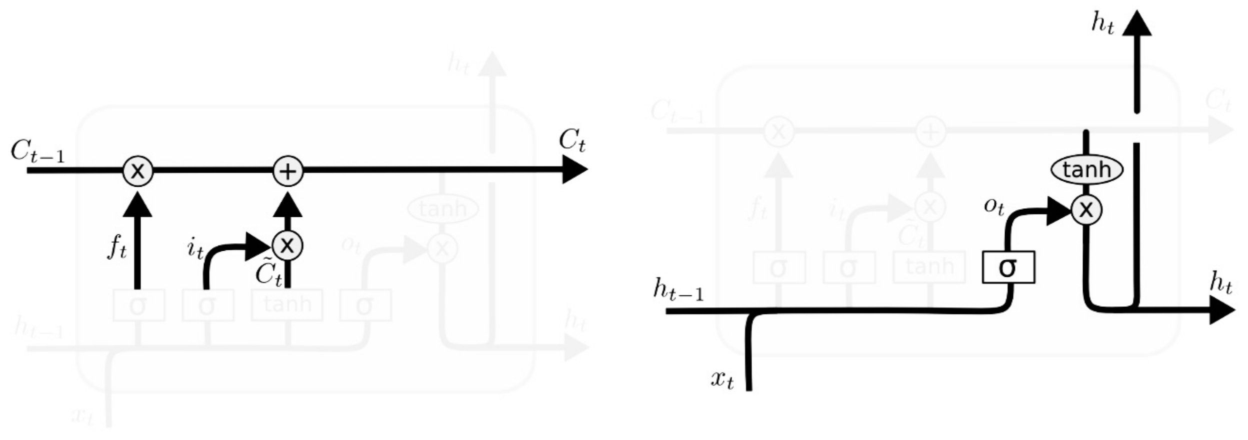
$$\text{Input gate: } i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.2}$$

$$\text{New memory cell: } \tilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.3}$$

The new candidate ($\tilde{C_t}$) values in LSTM provide a suggestion for updating the memory based on the current input and the previous hidden state. They are computed by combining these two elements, generating potential content for the cell state. After generating this suggestion ($\tilde{C_t}$),it is now the role of the input gate to decide how much of this suggestion should actually be incorporated into the cell state. If the input gate gives a high value for the current input (indicating its importance), more of the candidate value will be incorporated into the actual cell state.

From Figure 12 (a), The previous state C_{t-1} is updated (after information has been discarded using the forget gate) with the new candidate values, modulated by the input gate. Mathematically, this is represented as:

$$\text{Final memory cell: } C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C_t} \tag{3.4}$$



(a) State updating (b) Onput gate

Figure 12: **Inner working mechanism of single LSTM cell: updating and output gate**

where f_t is the forget gate's activation, i_t is the input gate's activation, and $\tilde{C}_t$ are the new candidate values.

Additionally, the output gate in an LSTM cell plays the role of a gatekeeper since it makes decisions based on past experiences and the current information it is presented with. Like other gates, sigmoid activation will be apply here as well. tanh will be applied as well to ensure the output stays between -1 to 1.

$$\text{Output gate: } o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.5}$$

$$\text{Hidden state: } h_t = o_t \cdot \tanh(C_t) \tag{3.6}$$

For the above formulas, x_t is the input at time t, h_t and C_t are the hidden state and memory cell at time t, respectively, σ denotes the sigmoid function, W_f, W_i, W_o, and W_C are weight matrices, b_f, b_i, b_o, and b_C are bias vectors.

Outputs Hidden state for the current time step h_t: Represents short-term memory. It's passed to the next LSTM cell and can also be used as the predicted output. Cell state for the current time step C_t: Acts as the long-term memory of the network. It processes the information with the influence of the forget and input gates and is passed along to the next LSTM cell.

While LSTMs mitigate the vanishing gradient challenge inherent to RNNs, they come with increased model complexity and higher memory requirements during training. In the following section, we will systematically evaluate and determine the most suitable neural network architecture for our dataset.

3.1.2 Autoencoder

Moving on to the pivotal aspect of constructing an anomaly detection system, the cornerstone of my proposed detection methodology is based on the use of autoencoders. The following is a generic autoencoder structure, as shown in figure 13.

Autoencoders are neural networks model designed to reproduce or reconstruct their input. Let $x(n)$ denote the input sequence, the autoencoder is a function f that outputs $f(x(n)) = \hat{x}(n)$. The architecture is often constructed as follows:

Inputs Normal Sequential data after pre-processing. The reason why the autoencoders are primarily trained with normal data is because by exposing the network only to typical patterns during training,

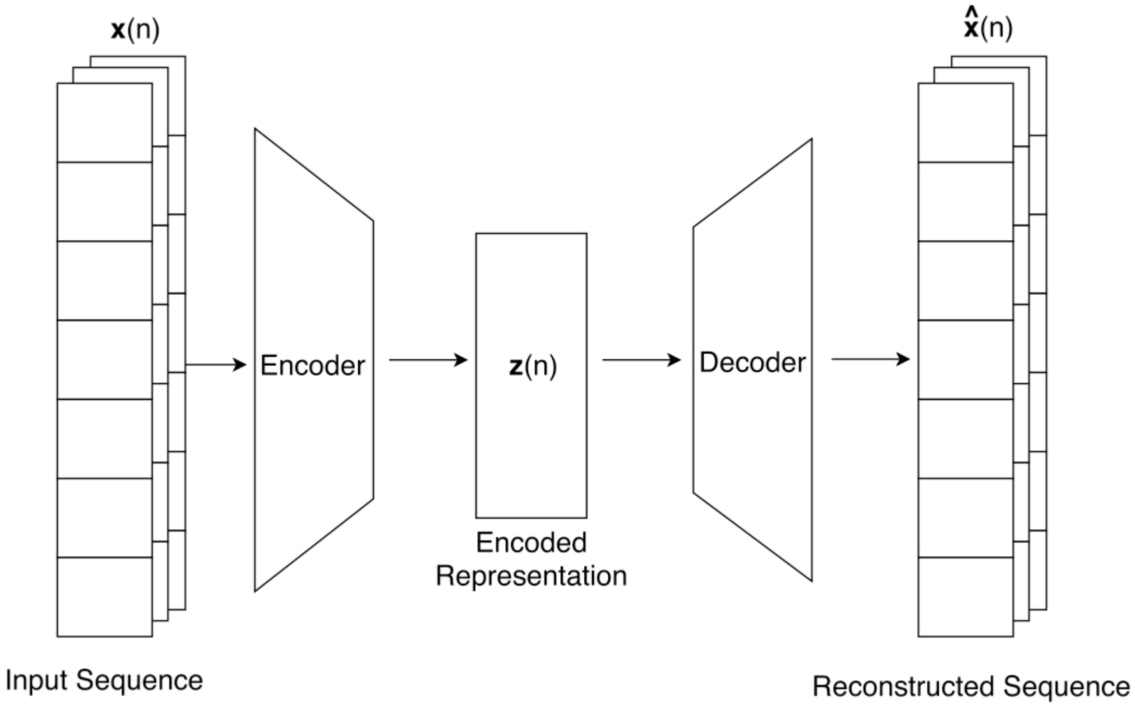


Figure 13: **Schematic representation of autoencoder structure.**

it will learn to reconstruct these patterns efficiently. Anomalies or deviations from these patterns will result in higher reconstruction errors, serving as a detection mechanism.

Encoder This part compresses the input into a latent-space (encoded) representation, and it essentially abstracts the primary characteristics of the input data into a lower-dimensional form.

Decoder This part reconstructs the input from the latent space (encoded) representation. For sequential data, this means reproducing the original sequence in its entirety. The decoder often mirrors the encoder's architecture but in reverse.

Output Reproduced input sequence by minimizing the reconstruction error. Proceeding to our unique scenario, the primary choice of encoder and decoder are made up with RNN and LSTM cells with their ability dealing with time series data like ours as I discussed in the previous section. We will primarily construct our first model using RNN cells for both encoder and decoder and second one using LSTM cells for comparison purpose. Before we put the model to the test, a crucial step is setting the threshold for detecting anomalies. This determines the acceptable limit for reconstruction error and when testing, any data point with a reconstruction error surpassing this threshold is marked as anomalous. This threshold can be set based on statistical methods such as Interquartile Range (IQR) and Z-score, or domain-specific knowledge.

In our current scenario, we introduce a new parameter k, to establish the anomaly detection threshold. The threshold is calculated by taking the average error from a set of data points and adding a certain number of standard deviations to it. The multiplier k determines how many standard deviations are added to the average error to set the threshold. The exact value of k can be fine-tuned based on test results.

3.2 Application to frequency data

3.2.1 Data Sampling and Pre-processing

From the data highlighted in chapter two 2.2, we can derive a subset containing only normal instances from $D1$, utilizing the defined target range. Given our intention to use an hour dense with anomalies as our primary testing dataset, we adopt an 80/20 distribution for training and testing. Consequently, five consecutive hours of data will serve as our training set. In addition, for data preparation we utilized a MinMax scaler to pre-process data in order to make them all stay in a specified range between 0 and 1. Then, we need to prepare sequences as our input of the autoencoders.

Let's denote the entire dataset as d. Given this dataset, we aim to transform it into sequences as input for our model. Each sequence is of fixed length $N = 15$, The sequences are generated by sliding a window of size N over the dataset, advancing one step at a time. These sequences are constructed as follows:

- The first sequence, $x(1)$, comprises the first n data points: $[x(1), x(2), \ldots, x(N)]$.
- The next sequence, $x(2)$, starts from the second data point, covering up to $[x(2), \ldots, x(N+1)]$.
- Following this pattern, the Nth sequence, $x(N)$, will span from $x(N)$ to $x(T)$ where $T = |d|$

By transforming the dataset in this manner, we ensure that each sequence captures a continuous chunk of the dataset, providing a temporal context for our model.

3.2.2 Training Phase

Since the variation in model parameters can significantly influence outcomes. In this experiment both RNN-RNN and LSTM-LSTM models share a consistent configuration of learning parameters to ensure the accuracy of model comparison.

We deploy 2 layers and 30 hidden neurons for the construction of both the encoder and decoder components. A detailed breakdown of the configuration is shown in table 3.

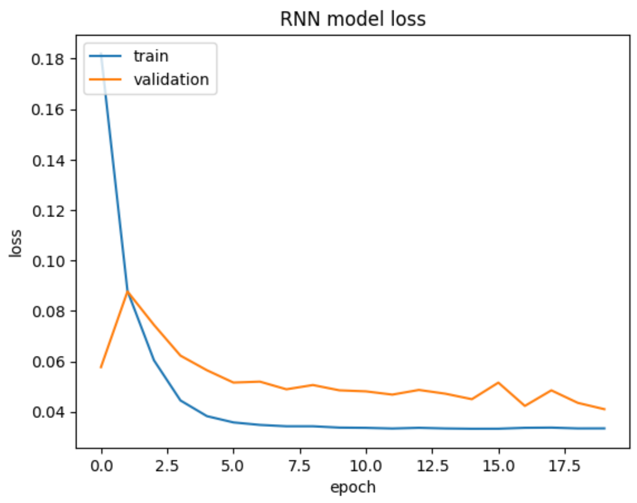
Parameters	**Value**
Number of layers of encoder	2
Number of layers of decoder	2
Number of hidden cells in each layer	30
Sequence length	15
Number of features	1
k	3

Table 3: Model Parameters for RNN/LSTM models

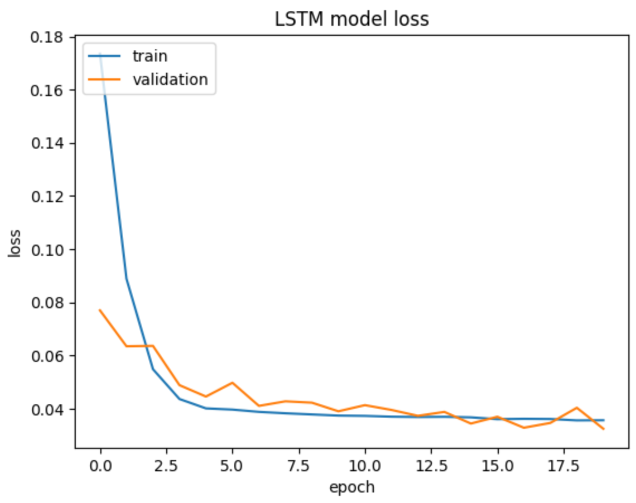
Both of model are trained using the same training set with validation at the same time and taking use of the same loss function, mean square error, as in equation 3.7.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f(x(n)) - x(n))^2 \tag{3.7}$$

Also, in order to avoid overfitting, we applied L1 regularization term to both models. It can be observed from figure 14, the error values start to converge around 10 epochs and also LSTM has a relatively better loss curve comparing to Simple RNN. This improved performance of the LSTM could be attributed to its ability to better handle long-term dependencies in the data.



(a) RNN loss plot.



(b) LSTM loss plot.

Figure 14: **Loss function plots for RNN and LSTM model.**

3.2.3 Performance Evaluation

Upon reviewing our results, we see a clear plot that displays errors and anomalies in test data for LSTM-LSTM, as shown in figure 15. This plot contrasts the reconstruction error with the identified anomalies. From the graph, it's evident that the model tends to label data points as anomalies when there is a significant increase in the reconstruction error, highlighting the model's ability to distinguish between usual and unusual patterns in the dataset.

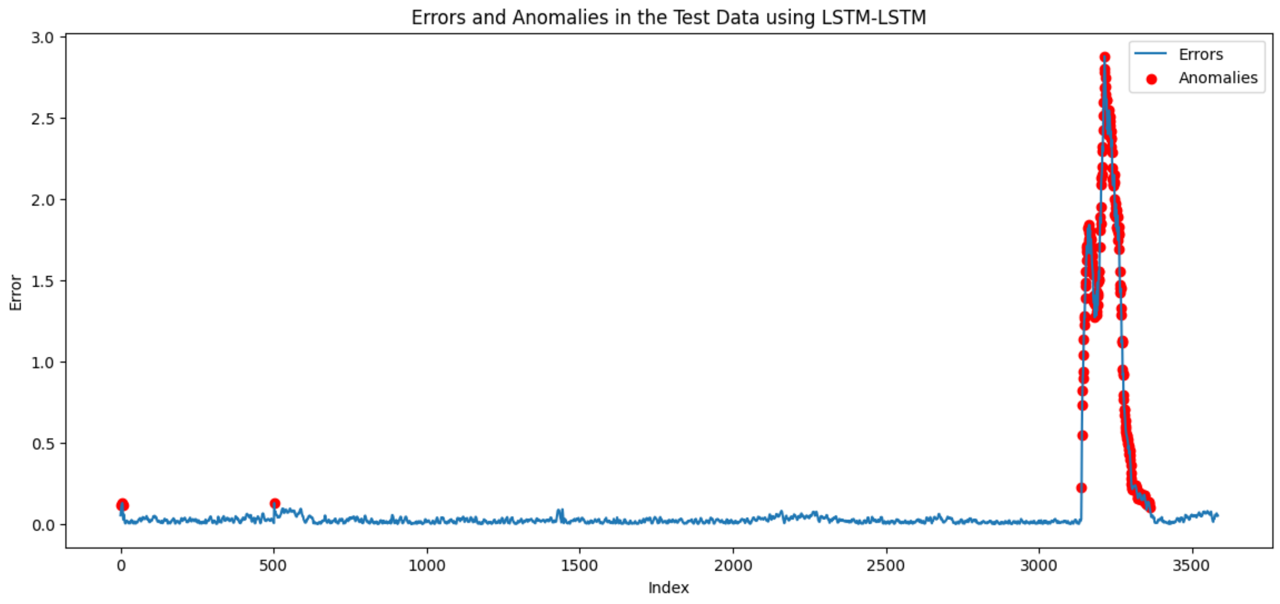
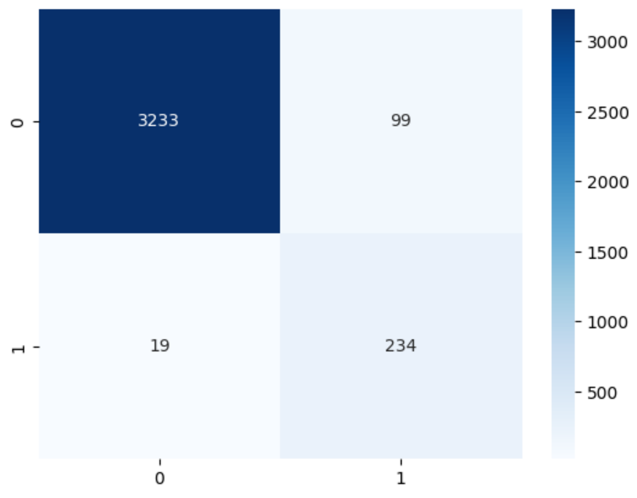


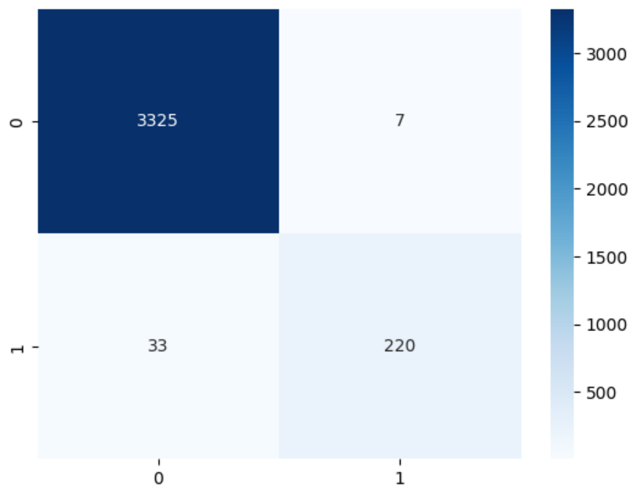
Figure 15: **Errors and anomalies in test data for LSTM-LSTM.**

Similarly, we have another plot for the RNN-RNN model, as shown in figure 27. Due to the detailed nature of our data, it's hard to determine which model is better just by looking at the plots. Some subtle differences in the visual data could be critical in practical situations and may go unnoticed.

To better understand the performance of each model, we used a confusion matrix, as shown in figure 16. This common tool in machine learning evaluation gives us information about true positives, true negatives, false positives, and false negatives. These metrics give a clearer picture of how each model performs. By analysing them, we can make a more grounded decision about the effectiveness of the LSTM-LSTM model compared to the RNN-RNN model, without being influenced by just visual impressions.



(a) Evaluation on RNN-RNN model.



(b) Evaluation on LSTM-LSTM model.

Figure 16: **Confusion matrix plots for RNN and LSTM model.**

Metric	**LSTM Model**	**RNN Model**
True Positives (TP)	220	234
True Negatives (TN)	3325	3233
False Positives (FP)	7	99
False Negatives (FN)	33	19

Table 4: evaluation metrics of LSTM and RNN Models

Table 4 and figure 16 shows the evaluation metrics in terms of accuracy mesurements of the anomaly detection model. Considering True Positives (TP) and False Negatives (FN): Given the higher TP and lower FN of the RNN, it appears to be more adept at correctly identifying anomalies. This makes the RNN potentially more appropriate for scenarios where failing to detect an anomaly could lead to significant repercussions.

On the other hand, when considering True Negatives (TN) and False Positives (FP): The LSTM, with its superior TN count and significantly reduced FP, showcases strong performance in accurately identifying non-anomalies and minimizing false alarms. This positions the LSTM as a prime choice for applications where ensuring non-anomalies are not misclassified as anomalies is crucial, especially when the implications of such misclassifications are substantial.

At this point, it is not obvious which model is superior to the other. To gain a more comprehensive understanding of their respective performances, it would be prudent to evaluate them using additional metrics such as Accuracy, Precision, Recall, and F1-score.

Accuracy Accuracy measures the ratio of correctly predicted instances to the total number of instances. It provides a high-level overview of the model's performance across both anomalies and non-anomalies classification.

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Instances}}$$

Precision Precision quantifies the proportion of instances that are correctly identified as anomalies among all instances that are labeled as anomalies by the model. It is especially crucial in contexts where falsely classifying a non-anomaly as an anomaly has significant consequences. A higher precision indicates that when the model detects an anomaly, it is highly likely to be a true anomaly.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Recall Recall measures the proportion of actual anomalies that the model correctly identified. It is particularly important in scenarios where not detecting an anomaly could have severe implications. A high recall ensures that most anomalies are captured by the model, even if there are some false alarms.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

F1-Score The F1-Score is the harmonic mean of Precision and Recall. It provides a single metric that balances the trade-off between the benefits of correctly identifying anomalies (precision) and the importance of not missing any actual anomalies (recall).

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metric	**LSTM Model**	**RNN Model**
Accuracy	98.88%	96.71%
Precision	96.92%	70.27%
Recall	86.96%	92.49%
F1-Score	91.67%	79.86%

Table 5: Performance Metrics of LSTM and RNN Models

From the performance results of the metrics in table 5, the LSTM model exhibits superior accuracy and precision in comparison to the RNN model. This implies that, on the whole, the LSTM model is more adept at classifying data points correctly. When it identifies a data point as an anomaly, it is more likely to be an actual anomaly. On the other hand, the RNN model showcases greater recall. This suggests that the RNN model excels at detecting genuine anomalies, even though it might mistakenly classify some non-anomalies as anomalies.

The F1-Score, acting as a equilibrium between precision and recall, is distinctly higher for the LSTM model, suggesting its superior capability in distinguishing anomalies from non-anomalies. Nonetheless, the differential in recall observed between the two models suggests that depending on particular contexts or data distributions, one model may offer a more favorable performance.

To conclude, while the LSTM model generally surpasses the RNN model in terms of metrics, the decision between the two should factor in the practical consequences of misidentifying non-anomalies as anomalies and vice versa, tailored to the specific use-case.

4 Forecast

In chapter 4, we delve into a comprehensive analysis of predictive models aimed at enhancing our understanding of the underlying data dynamics. The chapter is divided into two key sections, In Section 4.1, we explore a benchmark model based on the ARIMA framework, which serves as a foundational reference for our subsequent analyses. In Section 4.2, we propose a neural network model to improve the prediction performance. Furthermore, both sections comprehensively present the model's architecture, results and insights obtained from the experimental analyses, adding valuable context to the outcomes achieved.

4.1 Benchmark Model - ARIMA

We first fit a ARIMA as an initial check to obtain a benchmark performance from this simpler model. It gives a point of reference to evaluate how much better (or worse) more sophisticated models perform later.

4.1.1 Method

The ARIMA model, which stands for Autoregressive Integrated Moving Average, is one of the most widely used techniques for forecasting time series data. The strength of ARIMA lies in its ability to model several kinds of time series structures, including trends and seasonality, by encompassing three main components: Autoregressive (AR), Integrated (I), and Moving Average (MA).

Autoregressive (AR) Component

The AR component accounts for the relationship between an observation and a number of lagged observations. It is termed "autoregressive" because it's a regression of the variable against itself.

The autoregressive model of order p, denoted as AR(p), is given by:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \varepsilon_t \tag{4.1}$$

Where:

- X_t is the value of the series at time t.
- c is a constant.
- $\phi_1, \phi_2, \ldots, \phi_p$ are the parameters of the model.
- ε_t is the white noise (i.e., a random error term) at time t.

Integrated (I) component

The Integrated (I) component in time series analysis represents the number of differences required to transform a non-stationary time series into a stationary one. This differencing process helps in eliminating any trend or seasonality in the series. Mathematically, the first difference of a series Y is given by:

$$Y_t = X_t - X_{t-1} \tag{4.2}$$

The integrated component is responsible for making the time series stationary by differencing the series a certain number of times since ARIMA needs to ensure the series becomes stationary.

Moving Average (MA) Component

The Moving Average (MA) component models the relationship between an observation and a residual error from a moving average model applied to lagged observations. The formula for the MA(q) model is:

$$X_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \ldots + \theta_q\varepsilon_{t-q} \tag{4.3}$$

Where μ is the mean of the series, ε_t is the white noise or error at time t, $\theta_1, \theta_2, \ldots, \theta_q$ are the parameters of the model, q is the order of the MA model, indicating the number of lagged forecast errors in the prediction equation.

In practice, the ARIMA model is represented as ARIMA (p, d, q), where p is the order of the AR component, d is the order of differencing, and q is the order of the MA component. Each of these components helps the model capture different patterns present in the time series, and their combined effect can be leveraged to produce forecasts for different type of datasets [14].

Methods on parameters selection are vary. Instead of manually inspecting ACF and PACF plots to decide on the value of p and q, using AIC offers a more systematic and automated approach. By iterating over different combinations of parameters and computing the AIC for each, the optimal parameter set can be programmatically determined.

Since from chapter two, we can see find the seasonal decomposition, there's also a seasonality present on a daily basis, hence a seasonal component is also included in this model.

4.1.2 Analysis

The goal is to use 720 minutes to forecast the next 15 minutes right after. In order to forecast the next 15 minutes using the ARIMA model, a sliding window approach was employed for data processing. A fixed window of 720 consecutive minutes, representing a 12-hour interval, was utilized as the training dataset for each forecast. The initial 720-minute window was used to estimate the ARIMA model's parameters, which was then fitted to the data for generating a forecast for the subsequent 15 minutes. Following this, the window was shifted forward by 720 data points to avoid overlapping, and the process was iteratively repeated for 10 times.

The following plot 17 represent one of the forecasting results. In this representation, the blue line denotes the forecasted values, while the red line depicts the actual values. Even without employing specific evaluation metrics, a preliminary observation reveals that the model fails to adequately capture the inherent variations in the data. Since every time ARIMA will take the sequential data as new input and generate the best parameter combination possible. It's evident that there is room for model improvement to ensure more accurate forecasting in future applications.

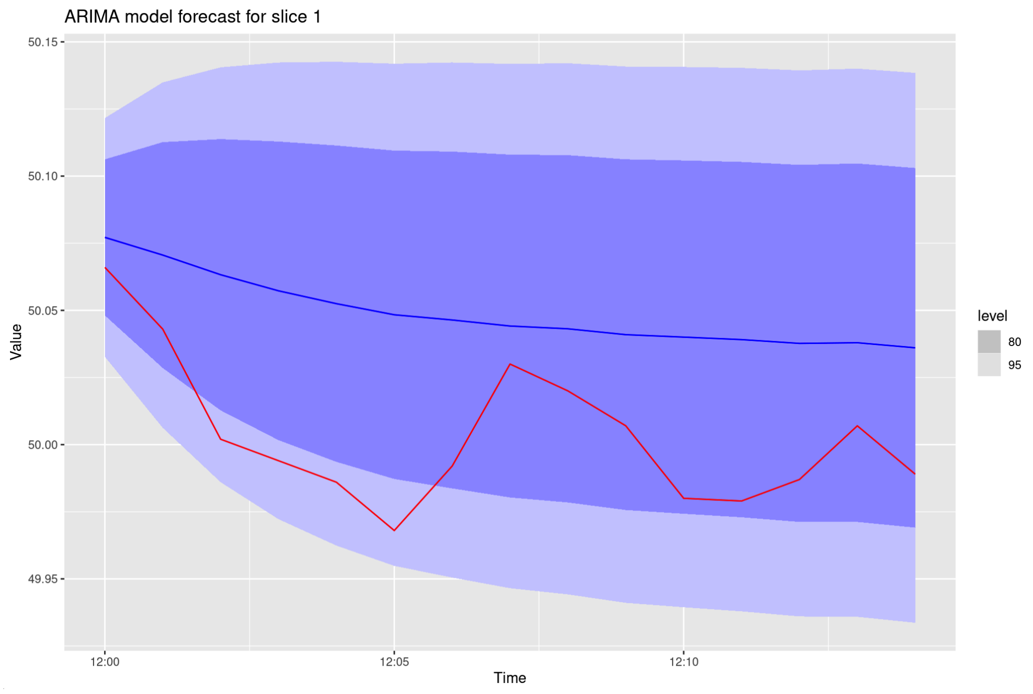


Figure 17: **Forcasting result using ARIMA model.**

Figure 17 is the zoom-in version of the forcasting part of figure 18.

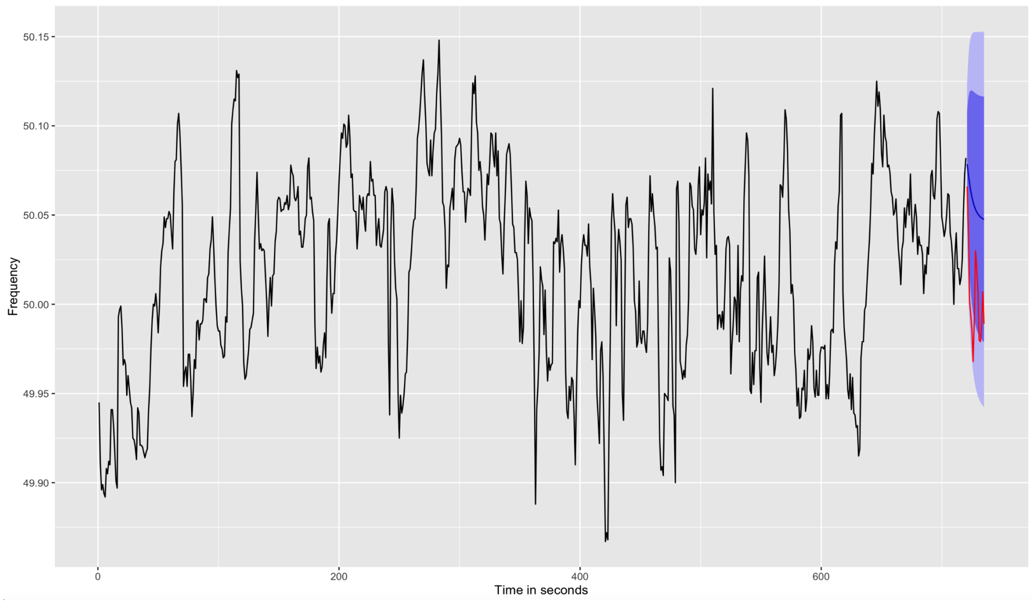


Figure 18: **Forcasting result using ARIMA model.**

The effectiveness of the ARIMA model is limited due to a few key challenges. One major concern is the data's granularity. Data with high granularity can sometimes display volatile patterns which can change rapidly. ARIMA models can struggle with such abrupt changes, as they're essentially linear models built on past observations. Another good example is forecasting stock price, it can swing dramatically due to various internal and external factors and the sheer variability can be overwhelming for forecasting.

In the subsequent section, a neural network based model is purposed and the model comparison is included between neural networks and ARIMA.

4.2 CNN-LSTM Seq2Seq Model

Since neural networks can capture complex non-linear relationships and have better adaptability to high granularity in the data, a LSTM based neural network model can be developed to forecast the future trajectory of our data. Given that the problem involves sequential inputs and outputs with a many-to-many mapping, a Sequence-to-Sequence (Seq2Seq) architecture is specifically employed to handle this complexity [20].

4.2.1 Methodology

CNN Convolutional Neural Network was firstly used to address image and visual recognition tasks as it shows great ability to automatically extract hierarchical features from images [30]. When applied to time series data, CNNs take the temporal sequence, capturing variations and patterns. This capability positions them as effective tools for analysing and understanding temporal data trends.

Input The CNN receives the sequential data as its input.

Operation As the primary operation in CNNs, convolution involves the use of filters that slide across the input sequence to detect patterns. This operation can capture local dependencies or patterns in the data. The result is a feature map that highlights the identified patterns at various positions in the sequence. After the convolution, the raw feature map is passed through a non-linear activation function, often the Rectified Linear Unit (ReLU), as shown in figure 19. This introduces non-linearity into the model, enabling it to learn more complex patterns. A padding is also incorporate in the convolutional layers to ensure the same sequency length of the input.

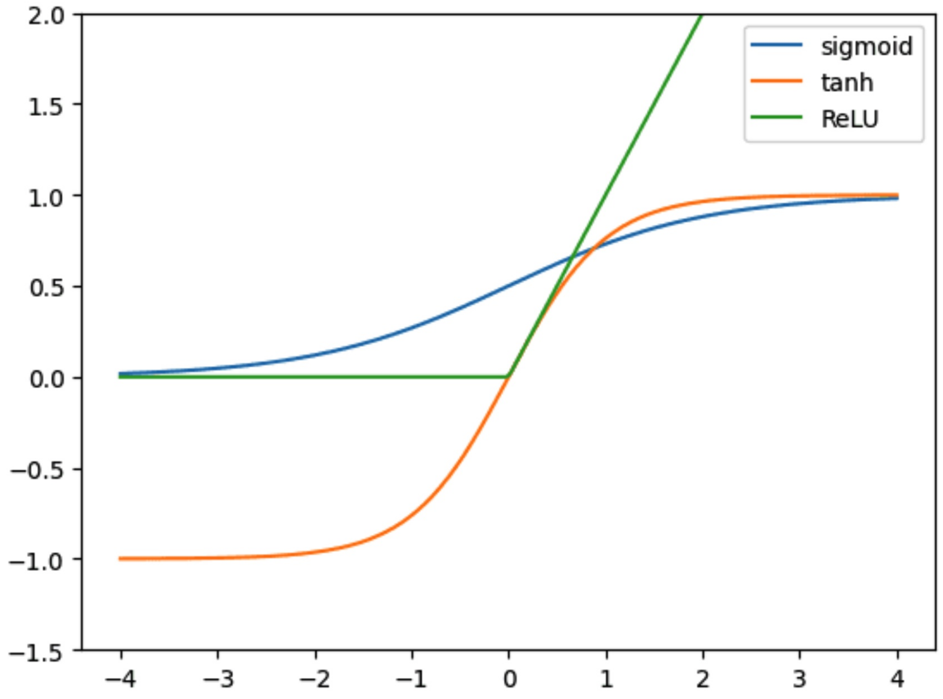


Figure 19: **Plots of non-linear activation functions.**

Output A processed version of input is processed with emphasizing certain characteristics and patterns. Given the use of padding, this output sequence maintains the same length as the input.

Seq2Seq In chapter 3, the encoder and decoder structure is mentioned inherent to Seq2Seq models. Autoencoders actually represent a specific case of this structure, primarily tasked with reproducing input data. However, Seq2Seq models are frequently employed for forecasting purposes.

The following diagram shows the primary process of how the Seq2seq model works at training, as shown in figure 20.

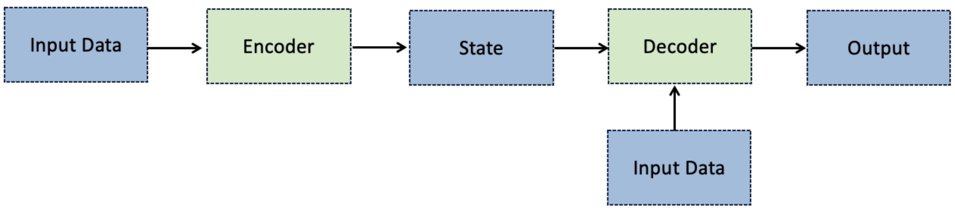


Figure 20: **Diagram of the encoder-decoder model during training**

Instead of solely relying on LSTMs for both the encoder and decoder, we integrate convolutional layers within the encoder. This addition allows for a more intricate extraction of characteristics from the input, enhancing the model's capability to discern various patterns since CNNs are adept at identifying local features through convolutional operations, they can capture nuances in the data that might be overlooked by LSTMs alone.

A comprehensive structure is depicted in figure 21. For clarity and illustration purpose, the diagram illustrates just a single layer of both the encoder and decoder.

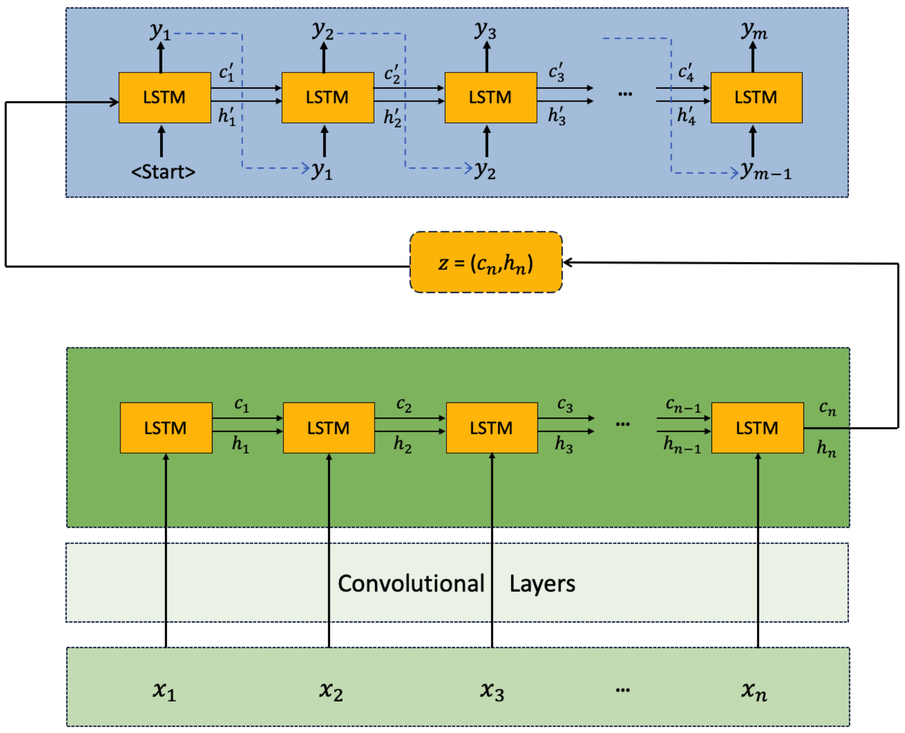


Figure 21: **Schematic representation of the inner mechanism of Seq2seq model**

Encoding process

Let's consider an input sequence represented as $x = [x_1, x_2, \ldots, x_n]$. After the input sequence passes through CNN, it will enter the LSTM layer. We initialize both hidden state h and cell state c to all zeros. Every time the data passes through the LSTM cell, these values are updated. The hidden states, represented as $h = [h_1, h_2, ..., h_n]$, along with a cell state c, symbolized as $c = [c_1, c_2, ..., c_n]$, are both generated and updated as detailed in chapter 3. Now the intermediate state vector becomes the final hidden state and the final cell state, represented as $z = (h_n, c_n)$. And they are associated with the encoder with formula 4.4:

$$(h_n, c_n) = \text{Encoder}(x_n, (h_{n-1}, c_{n-1})) \tag{4.4}$$

Decoding process

Taking use of the final hidden state and cell state vector from encoder and also the last point of the input sequency as the starting point to predict y1 and so on. To clarify: hidden states of the decoder are denoted as $h' = [h'_1, h'_2, ..., h'_m]$; cell states of the decoder are represented as $c' = [c'_1, c'_2, ..., c'_m]$. This offers formulas that bear a close resemblance to those in the encoder, which leads to eqation 4.5.

$$(h'_m, c'_m) = \text{Decoder}(y_m, (h_{m-1}, c_{m-1})) \tag{4.5}$$

Also, there is mechanism in decoder called ‘teacher forcing’ to speed up the training phase. Instead of feeding the model’s own predictions into the next time step of the decoder, “teacher forcing” feeds the true training data to help the model learn faster as it is being “guided” by the correct outputs, rather than potentially compounding errors from its own predictions.

4.2.2 Application to Frequency Data

Data sampling and pre-processing With the same idea, we utilize the downsampled data, $D2$ and establish a sliding window of length 720. For each iteration, this window captures 720 minutes of data, forming an input sequence to forecast the subsequent 15 minutes. Instead of extracting non-overlapping input sequences every 720 time steps, we adjust our approach to select sequences at intervals of 60 data points. This method provides a larger number of input sequences for training purposes. This procedure continues until the entirety of the training set is covered. Instead of adopting the MinMaxScaler for data pre-processing, we have chosen to apply the z-score normalization method to the dataset.

The formula for the z-score of a value x is given by:

$$z = \frac{x - \mu}{\sigma} \tag{4.6}$$

The resulting z-scores represent the number of standard deviations a data point is from the mean. Since the frequency values are all fluctuated around 50, the raw scale might not be the most intuitive for monitoring and interpretation, especially during training. By centring the data around zero, it becomes easier to understand positive and negative deviations. It is useful when monitoring training progress and make it easier to set early stopping criteria.

We take full month D2 data using 8/2 split as train and validation set.

Training phase The configuration of parameters of the Seq2seq model with autoencoder structure is shown in the table 6.

The choice of Loss function is Mean Square Error to minimize the prediction error, same as equation 3.7.

To avoid overfitting, another dropout layer is added to our model functioning as a regularization technique that randomly deactivates a fraction of neurons during each training iteration, thereby preventing any single neuron from becoming overly specialized to the training data.

4.2.3 Model Evaluation

Overall Prediction Evaluation for Each Sequential Prediction

The following plot 22 and plot 23 show one of the best predictions and one of the worst ones. One slice means one forecasting window. One week’s data is chosen as our test set hence there are 155 slices in total.

Parameters	Value
Number of CNN Layers	3
Number of Encoder Layers (LSTM)	3
Number of Decoder Layers (LSTM)	3
Number of hidden cells in each layer	256
Input sequence length	720
Output sequence length	15
Batch size	128
Dropout	0.05
Number of features	1

Table 6: Model Parameter configuration of the Seq2seq neural network model

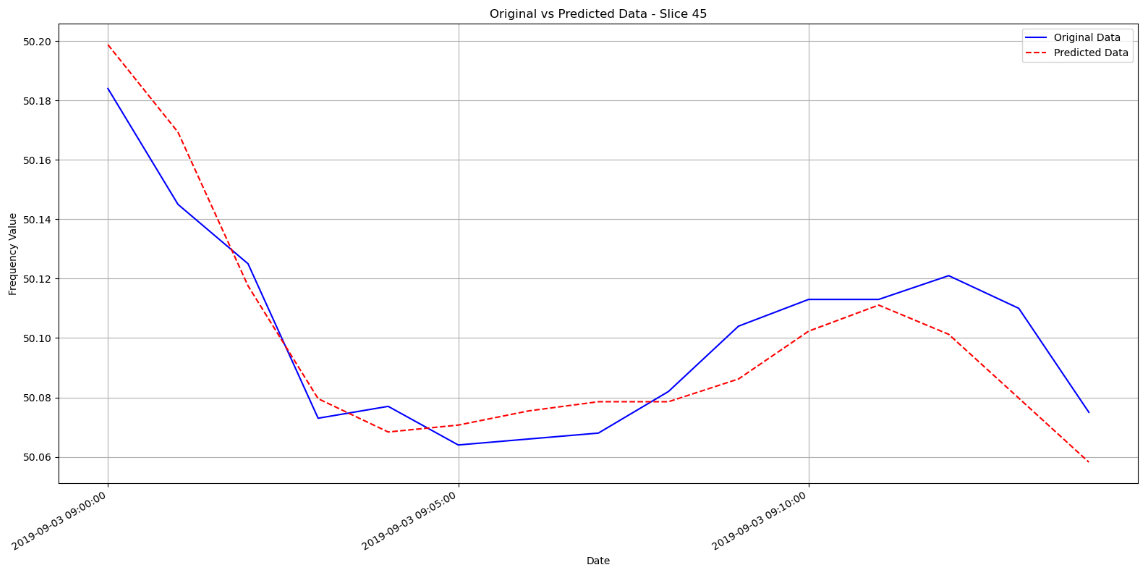


Figure 22: **One of the best prediction data vs original data for a slice of sequence.**

As the graph indicates, the model effectively captures the general trajectory of the data, mirroring its trend over time.

Since it is difficult to view all the visualizations (slices) to evaluate the model performance, we will use some evaluation metrics such as MAE and MSE.

The Mean Absolute Error (MAE) measures the average absolute difference between the actual values and the predicted values per prediction slice. It provides an idea of how wrong the predictions were, without considering their direction.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.7}$$

Where y_i represents the true values, $\hat{y}_i$ represents the predicted values, n is the number of data points or samples.

The Mean Squared Error (MSE) measures the average squared difference between the estimated values and the actual values per prediction slice. It quantifies the spread of the residual errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.8}$$

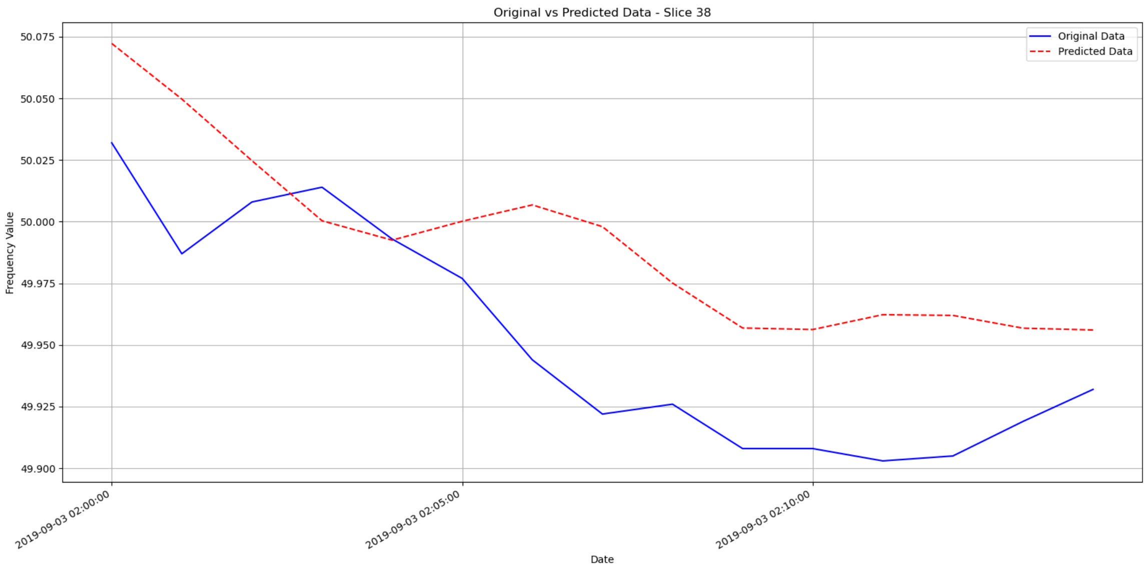


Figure 23: **One of the worst prediction data vs original data for a slice of sequence.**

All of them provide a measure to quantify the average error between the model's predictions. We can utilize these metrics to systematically assess the performance of our model. For each slice, the metrics are calculated, and with all 155 slices from test data, we can get accurate measurements for the forcasting prediction with the Seq2seq neural network model.

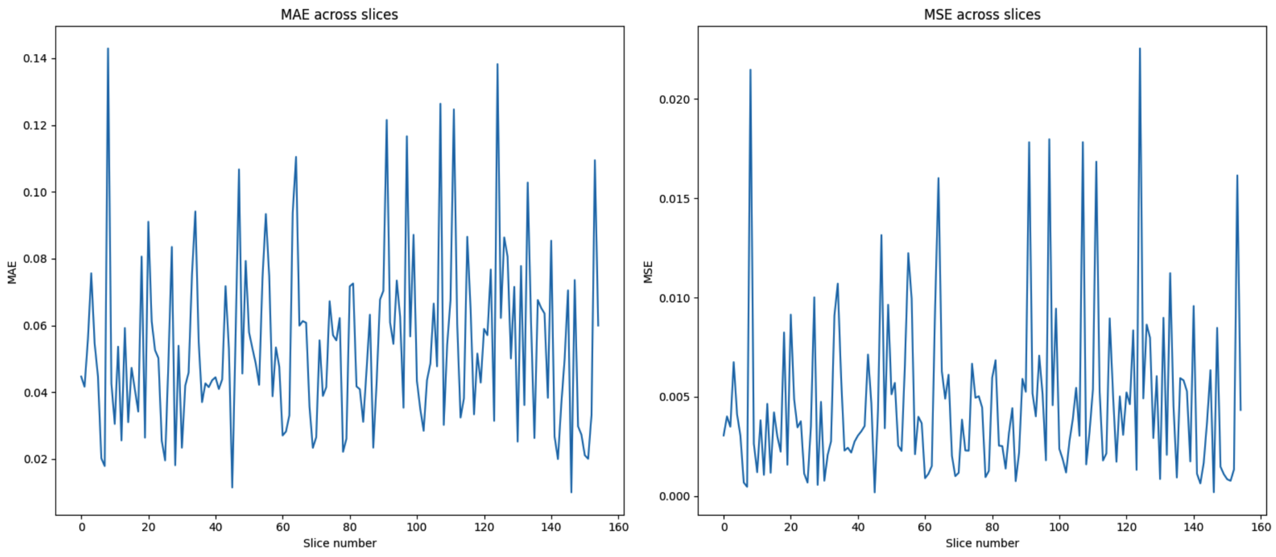


Figure 24: **Performance evaluation in terms of MAE and MSE across slices in test data.**

Figure 24 representing the MAE, MSE values and for each prediction slice and the figure 25 presents the distribution plots of these error metrics. Distribution plots can visually show the spread and skewness of data. As it can be noticed, they all have a skewness to right trend among these distributions which indicates that the majority of predictions are aligned with the actual values, as evidenced by the error values being less than the average. However, there are occasional predictions with notably larger errors, leading to a rightward stretch in the distribution's tail.

By taking the mean and standard deviation among all slices for MAE and MSE we can further analysis on the model performance, as shown in table 7.

The model's Mean Absolute Error (MAE) stands at approximately 5.43%, indicating that, on average, predictions deviate by this percentage from actual values. However, with a standard deviation of 2.566% for the MAE, errors can range between roughly 2.87% to 7.996%. Meanwhile, the Mean Squared Error (MSE) is 0.00479, emphasizing the presence of larger errors, as evidenced by the

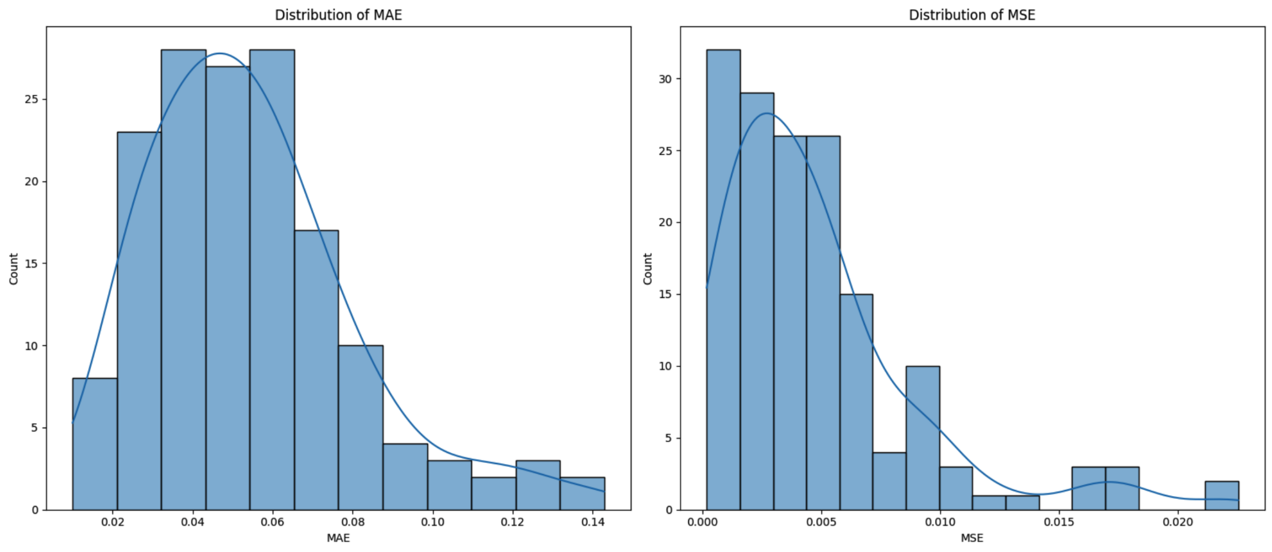


Figure 25: **Performance statistics for MAE and MSE for test data.**

Metric	**Mean**	**Standard Deviation**
MAE	0.05431	0.02566
MSE	0.00479	0.00420

Table 7: Metric statistics for forcasting prediction performance evaluation

closeness of its standard deviation (0.00420) to its mean. This means that while the model often predicts with reasonable accuracy, there are instances of significant deviations potentially due to some certain challenging data points challenging points. Hence, there is still room for improvement and further refinement for enhanced consistency.

Additionally, we aim to compare the performance of the CNN-LSTM model to the ARIMA model. Given that we used a 720-minute sliding window for ARIMA to predict the subsequent 15 minutes, while using only a 60-minute window for the CNN-LSTM since the model requires a sufficient sequences for training purpose, it's essential to align their predictions for a fair comparison. In terms of the test set, the first prediction from the ARIMA model should be matched with the 12th prediction from the CNN-LSTM model. This will allow us to accurately compare the Mean Absolute Error (MAE) and Mean Squared Error (MSE) of both models. The performance comparison of ARIMA and CNN-LSTM models on these metrics is shown in table 8.

Table 8: Overall Forecasting Performance of ARIMA and CNN-LSTM Models

Model	Metric	Mean	Std
CNN-LSTM model	MAE	**0.03410**	**0.00706**
	MSE	**0.00189**	**0.00081**
ARIMA model	MAE	0.05324	0.02575
	MSE	0.00472	0.00459

In a comparative analysis of forecasting performance, the CNN-LSTM model demonstrated a marked improvement over the ARIMA model in terms of both MAE and MSE. Specifically, the CNN-LSTM model yielded a 35.97% lower MAE and a 59.96% lower MSE. These quantitative metrics suggest that the neural network model is considerably more accurate and reliable for this particular forecasting application with this specific power grid frequency dataset. Thus, based on these metrics, the neural network model offers a more precise forecasting solution compared to its ARIMA counterpart.

Timestamp-Wise Evaluation

Given that the CNN-LSTM model has shown superior trajectory accuracy for each slice, we aim to delve deeper into its performance by assessing it at individual timestamps. With a total of 155 predictions, each spanning a forecast of 15 minutes, we can compute the average error across these predictions for each minute, from t=1 to t=15, to understand how the model's accuracy evolves over time. Figure 26 shows the mean average error for the outputs of 15 continuous timesteps.

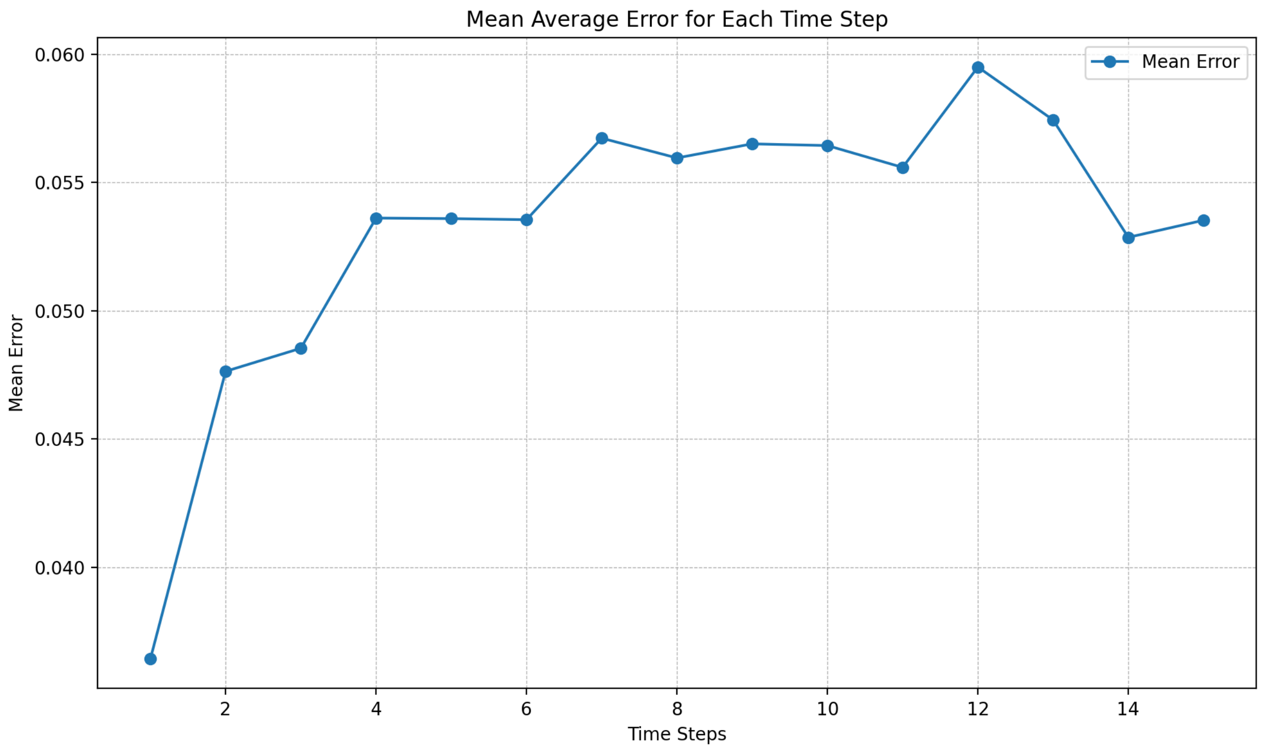


Figure 26: **Mean average error for timesteps $t = 1$ to $t = 15$.**

In figure 26, the trend in prediction errors across time steps is evident. Initially, the errors are minimal, highlighting the LSTM model's efficacy in near-term forecasting. As the timeline progresses, the error keeps increasing slightly until 7 minutes. Then it remains relatively consistent around a value of 0.055, suggesting the LSTM model's capability for longer-term predictions. A subsequent reduction in prediction errors towards the latter stages can be attributed to the nuanced architecture of our neural network model, emphasizing its versatility across varying forecasting horizons. From the general trend, the increase in mean average error over time, indicates potentials for further refinement of the current model, more details on suggestions for improvement can be found in section 5.2.

To thoroughly analyze prediction errors (MAE) across timesteps, refer to figure 28 which utilizes the entirety of the test data. The heatmap's gradient from light to dark serves as an indicative measure of accuracy. Rows denote distinct data slices and columns pinpoint specific timesteps, facilitating a detailed assessment of the model's performance across the timesteps and all the slices in test set.

5 Discussion

Recall the core of this thesis is to design two separate models: the first for detecting anomalies in power grid frequency data, and the second for forecasting future trajectories of the same. These models are intended to provide grid operators with essential tools, enabling them to not only monitor but also anticipate frequency deviations, thus ensuring consistent grid operations. The subsequent sections will elaborate on both models and provide the implications to real life scenarios, strengths and limitations and some insights for future researchers.

5.1 Anomaly Detection

For the anomaly detection framework, auto-encoder embedded with recurrent neural network model was established to do the job and achieve promising results. Instead of the labor-intensive task of continuously monitoring frequency alterations, engineers can leverage this model. It serves as an automated oversight mechanism, flagging deviations that could be symptomatic of potential system disturbances or failures. Essentially, it acts as an early warning system, ensuring that the operating frequency remains within safe bounds. Control engineers can also adjust the model selection accordingly using the provided performance metrics.

Moreover, the control engineers can utilize this model to real life as the training process of as the model does not necessitate intricate training procedures. Contrary to some deep learning architectures which may be computationally demanding and complex, the auto-encoder embedded with RNN is relatively straightforward. This allows engineers to implement it without requiring extensive computational resources or expertise in deep learning.

Beyond anomaly detection, the model can also be repurposed for stability analyses. This is achieved by calibrating the classification threshold manually. By doing so, engineers can determine the sensitivity of the model, making it either more tolerant to minor fluctuations or more stringent in its detection criteria, depending on the specific requirements of the system under observation. For instance, if a control engineer wishes to assess the frequency stability for the past hour, by manually adjusting the threshold, they can determine the consistency of the frequency.

As the model achieve over 90 percent accuracy in detecting potential anomalies, in contexts where decisions are driven by data, such a high accuracy rate demonstrates that the model is dependable on this specific case.

However, there are still some limitations. Utilizing only normal data to train the autoencoders presents a challenge. Training exclusively on normal data means the model is well-versed in recognizing what constitutes a 'normal' pattern, but it may lack the exposure to detect subtle or novel anomalies effectively. Furthermore, as the definition of 'normal' can evolve over time due to changes in system behavior, operational practices, or external influences, there is a pressing need to adjust the training set frequently. Relying on a static training set could reduce the model's efficacy in detecting anomalies in the long run. In essence, the approach is not a "set it and forget it" solution. For sustained accuracy and relevance, the training data needs regular reviews and updates to encapsulate the evolving nature of 'normal' in the system being monitored. This iterative process ensures that the autoencoder remains sensitive to both known and emerging anomalies.

In the future, there is potential in exploring deeper or wider autoencoder architectures and integrating additional data types for a holistic view, such as grid equipment status or localized power consumption rates. Additionally, instead of doing a binary classification denoting anomalies and non-anomalies, we can use multi-classification instead discreetly denoting how far the anomaly apart from the safe range.

Building on the current work, future endeavors could also focus on integrating more diverse datasets to expose the models to a variety of anomalies, enhancing their robustness. Incorporation of auxiliary data, like grid load forecasts, could enrich the context for better anomaly detection. Research can also delve into ensemble methodologies, combining diverse models to enhance detection robustness.

5.2 Forecast

The fundamental difference between ARIMA and CNN-LSTM lies in their model building and validation processes. Specifically, CNN-LSTM follows a two-phase approach: training and testing. In this approach, a substantial portion of data is used to 'train' or 'build' the model, which essentially means fine-tuning its internal parameters based on the patterns in the training data. Once this training phase is complete, the model's predictive performance is evaluated using a separate set of test data. In the context of this dissertation, we trained the CNN-LSTM model using an entire month's data and subsequently tested it using the following week's data, resulting in 155 sequential predictions.

On the other hand, ARIMA operates differently. It does not follow the notion of building a model from scratch or refining it iteratively. Instead, ARIMA uses a given sequence of data to forecast future points without an explicit separate training phase. This means that for ARIMA, you typically provide a sequence, and it projects forward based on the intrinsic patterns in that sequence. there is no iterative refinement of parameters to achieve an optimal combination as with deep learning models like CNN-LSTM.

Moreover, the ARIMA model, commonly used for time series forecasting, displayed limitations when used on our power grid frequency dataset. This data, characterized by sudden changes and unpredictability, often lacks consistent patterns. ARIMA's dependency on identifying and extrapolating these patterns resulted in its less than optimal performance for this specific dataset. Additionally, ARIMA faced challenges in addressing sharp, sudden variations, rendering it less appropriate for such data.

In many instances, ARIMA's predictions deviated from actual observations due to the inherent complexities of the dataset and its intermittent frequency shifts. The model's difficulty in handling unexpected frequency fluctuations, which are essential to consider in power grid contexts, can compromise timely and proactive grid management.

The neural network based model is not hard to implement for real application: for example, the engineer can take one month of power grid frequency data as training data and feed into the neural networks, then use the subsequent one week of data as test data to predict the future time-steps afterwards. The model is trained in this way and normally it only takes short time before converging to optimal solution for the dataset. If the enginner takes more data, make sure the data is consistent. As for the test data, the length of input sequence can vary between one week to one day, normally 12 hours (720 mins) should be fine to give a decent prediction. But in order to compensate for other normalization and other pre-processing of data, a few more datapoints are needed to keep consistency and mitigate the problem of distribution shift between training data and testing data. In the training phrase, this is not directly modeled but we would assume the distribution follow regular pattern and the process of data pre-processing involves z normlzation or MinMax Scaler operations.

In terms of the choice for neural network models, it proves effective in capturing the majority of data trends and variations. The architecture of this specific neural network, designed to predict sequences, provides a holistic approach to forecasting. A notable strength of the CNN-LSTM model is its computational efficiency. It can process large datasets, segmenting them into sequences, and subsequently update model parameters during each training iteration. This ensures continuous refinement and improvement in predictive accuracy.

However, since of our data is univarate, the limited features of training data could potentially hinder the model to learn more meaningful representations. Therefore, incorporating additional features like electricity demand could be beneficial, offering the model more context and data points to work with. Also, the "black box" nature of deep neural networks makes them hard to interpret. Even if they predict well, it is difficult to explain how they make those predictions, especially when compared to simpler models. This can be an issue when we need to understand and explain the model's decisions in important situations.

Additionaly, further research is more likely to focus on hybrid models, combining traditional statistical models with neural networks. Research could also focus on augmenting the dataset with synthetic data or even external features like weather conditions or special events that might influence power grid frequencies. This can lead to a more robust model capable of handling a broader spectrum of real-world scenarios. And in this case of forecasting with multi-variants, the neural network model is expected to excel as well.

In future research, there is potential to integrate self-attention mechanisms into the existing model structure [24]. This can place heightened emphasis on vital segments of the sequence during predictions. Notably, transformer-based models have seen a significant rise in their application within the time series analysis domain [21, 27].

6 Conclusion

In this dissertation, an extensive analysis of power grid frequency data was conducted. For anomaly detection, we employ a methodology based on recurrent neural networks with an autoencoder framework, utilizing its capabilities for feature representation and anomaly identification. In the part of time-series forecasting, we purposed a Seq2Seq model that integrated both CNN and LSTM networks.

The study identified that the conventional ARIMA model, though widely recognized for time-series forecasting, faced substantial challenges when deployed on our specific power grid dataset. This was attributed to its inherent difficulties with the abrupt shifts and volatile behaviors exhibited by the dataset. To overcome these challenges, the proposition and implementation of a neural network-based approach became imperative. Such an approach can excel particularly in scenarios with complex, non-linear dependencies, offering the flexibility and adaptability that our dataset demanded.

In the findings from our comprehensive analysis, the anomaly detection model we implemented exhibited an accuracy exceeding 90%, with the LSTM autoencoder framework reaching an impressive 98%, underscoring its capability in pinpointing deviations. Simultaneously, the CNN-LSTM forecasting model yielded predictions with a mean deviation of approximately 5%, marking a 59.96% reduction in MSE when contrasted with the ARIMA model. Furthermore, a timestamp-wise assessment revealed a progressive increase in prediction error over time, indicating potential avenues for further refinement.

In summary, this dissertation not only highlights the shortcomings of traditional statistical methods like ARIMA in our dataset but also posits neural network architectures as a more reliable alternative. Through systematic investigation and evaluation, we have offered a detailed insight into power grid frequency data analysis, setting a foundation for subsequent research in this domain.

References

[1] J. Alvarez-Cercadillo, J. Ortega-Garcia, and L. A. H. Gómez. Context modeling using RNN for keyword detection. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '93, Minneapolis, Minnesota, USA, April 27-30, 1993*, pages 569–572. IEEE Computer Society, 1993.

[2] K. Bandara, C. Bergmeir, and S. Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Syst. Appl.*, 140, 2020.

[3] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *CoRR*, abs/2002.04236, 2020.

[4] A. Bolzoni, R. Todd, A. J. Forsyth, and R. Perini. Real-time auto-regressive modelling of electric power network frequency. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society, Lisbon, Portugal, October 14-17, 2019*, pages 515–520. IEEE, 2019.

[5] J. S. Costa, A. Lunardi, P. C. Ribeiro, I. B. D. Silva, D. A. Fernandes, and A. J. S. Filho. Performance-based tuning for a model predictive direct power control in a grid-tied converter with l-filter. *IEEE Access*, 11:8017–8028, 2023.

[6] G. D'Angelo and F. Palmieri. Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial-temporal features extraction. *J. Netw. Comput. Appl.*, 173:102890, 2021.

[7] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi. Deep learning for time series anomaly detection: A survey. *CoRR*, abs/2211.05244, 2022.

[8] K. Feasel. *Finding Ghosts in Your Data: Anomaly Detection Techniques with Examples in Python.* Apress, New York, New York, 2022.

[9] L. R. Gorjão, M. Anvari, H. Kantz, C. Beck, D. Witthaut, M. Timme, and B. Schäfer. Data-driven model of the power-grid frequency dynamics. *IEEE Access*, 8:43082–43097, 2020.

[10] R. A. A. Habeeb, F. Nasaruddin, A. Gani, M. A. Amanullah, I. A. T. Hashem, E. Ahmed, and M. Imran. Clustering-based real-time anomaly detection - A breakthrough in big data technologies. *Trans. Emerg. Telecommun. Technol.*, 33(8), 2022.

[11] V. Hnamte, H. Nhung-Nguyen, J. Hussain, and Y. Kim. A novel two-stage deep learning model for network intrusion detection: LSTM-AE. *IEEE Access*, 11:37131–37148, 2023.

[12] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 473–479. MIT Press, 1996.

[13] R. Jiang, Y. Xue, and D. Zou. Interpretability-aware industrial anomaly detection using autoencoders. *IEEE Access*, 11:60490–60500, 2023.

[14] R. L. Kashyap. Optimal choice of AR and MA parts in autoregressive moving average models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(2):99–104, 1982.

[15] M. Khashei and M. Bijari. A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Appl. Soft Comput.*, 11(2):2664–2675, 2011.

[16] T. Kieu, B. Yang, C. Guo, and C. S. Jensen. Outlier detection for time series with recurrent autoencoder ensembles. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2725–2732. ijcai.org, 2019.

[17] R. Kozma, M. Kitamura, M. Sakuma, and Y. Yokoyama. Anomaly detection by neural network models and statistical time series analysis. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 5, pages 3207–3210 vol.5, 1994.

[18] J. Kruse, B. Schäfer, and D. Witthaut. Predictability of power grid frequency. *IEEE Access*, 8:149435–149446, 2020.

[19] A. Lenzi, J. Bessac, and M. Anitescu. Power grid frequency prediction using spatiotemporal modeling. *Stat. Anal. Data Min.*, 14(6):662–675, 2021.

[20] Y. Mu, M. Wang, X. Zheng, and H. Gao. An improved lstm-seq2seq-based forecasting method for electricity load. *Frontiers in Energy Research*, 10:1093667, 2023.

[21] H. S. Oliveira and H. P. Oliveira. Transformers for energy forecast. *Sensors (Basel, Switzerland)*, 23(15):6840, 2023.

[22] T. L. Onsaker, H. S. Nygård, D. Gomila, P. Colet, R. Mikut, R. Jumar, H. Maaß, U. G. Kühnapfel, V. Hagenmeyer, D. Witthaut, and B. Schäfer. Predicting the power grid frequency of european islands. *CoRR*, abs/2209.15414, 2022.

[23] G. Pu et al. A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Science and Technology*, 26(2):146–153, 2021.

[24] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2627–2633. ijcai.org, 2017.

[25] C. Tang, L. Xu, B. Yang, Y. Tang, and D. Zhao. Gru-based interpretable multivariate time series anomaly detection in industrial control system. *Comput. Secur.*, 127:103094, 2023.

[26] H. D. Trinh, E. Zeydan, L. Giupponi, and P. Dini. Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach. *IEEE Access*, 7:152187–152201, 2019.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv: Computation and Language*, 1706.03762v7, 2017.

[28] B. V. Vishwas and A. Patel. *Hands-on Time Series Analysis with Python: From Basics to Bleeding Edge Techniques.* Apress, Berkeley, CA, 1st edition, 2020.

[29] J. Wang, X. Li, J. Li, Q. Sun, and H. Wang. NGCU: A new RNN model for time-series data prediction. *Big Data Res.*, 27:100296, 2022.

[30] Z. Yang, Y. Nishio, and A. Ushida. Image processing of two-layer cnns-applications and their stability-. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 85-A(9):2052–2060, 2002.

[31] C. Zhang, G. Zhang, and S. Sun. A mixed unsupervised clustering-based intrusion detection model. In *2009 Third International Conference on Genetic and Evolutionary Computing, WGEC 2009, Guilin, China, 14-17 October 2009*, pages 426–428. IEEE Computer Society, 2009.

[32] G. P. Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175, 2003.

Appendices

A Anomaly plots based on RNN-RNN autoencoder model

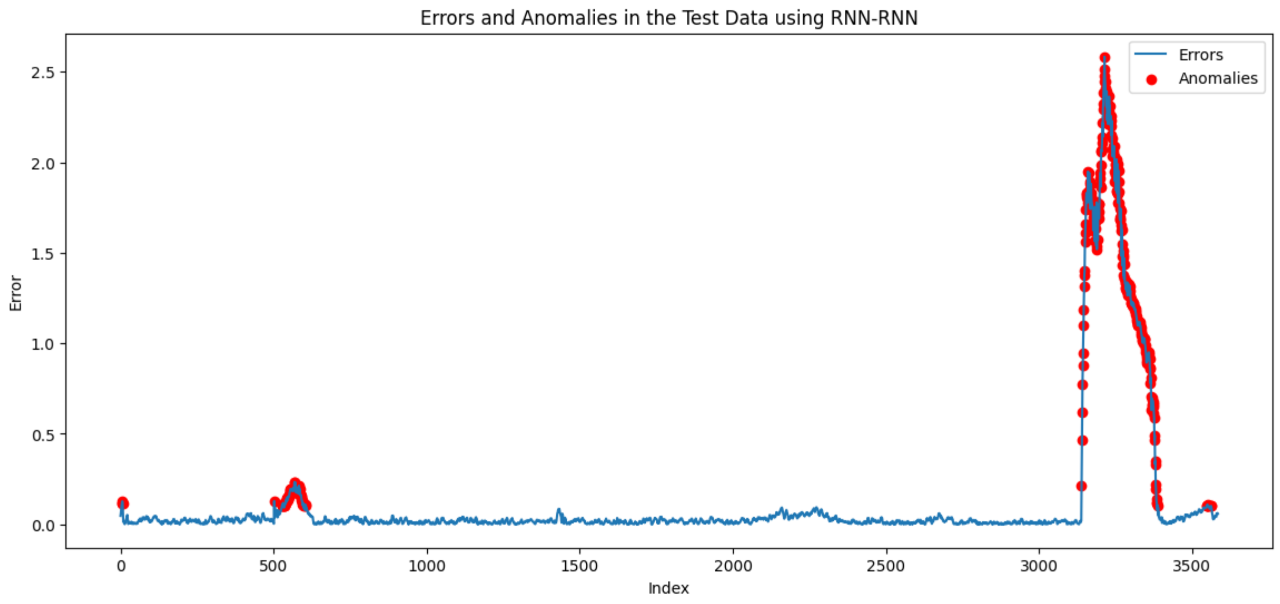


Figure 27: **Errors and anomalies in test data for RNN-RNN model.**

B Heatmap of Prediction Errors for Test Results

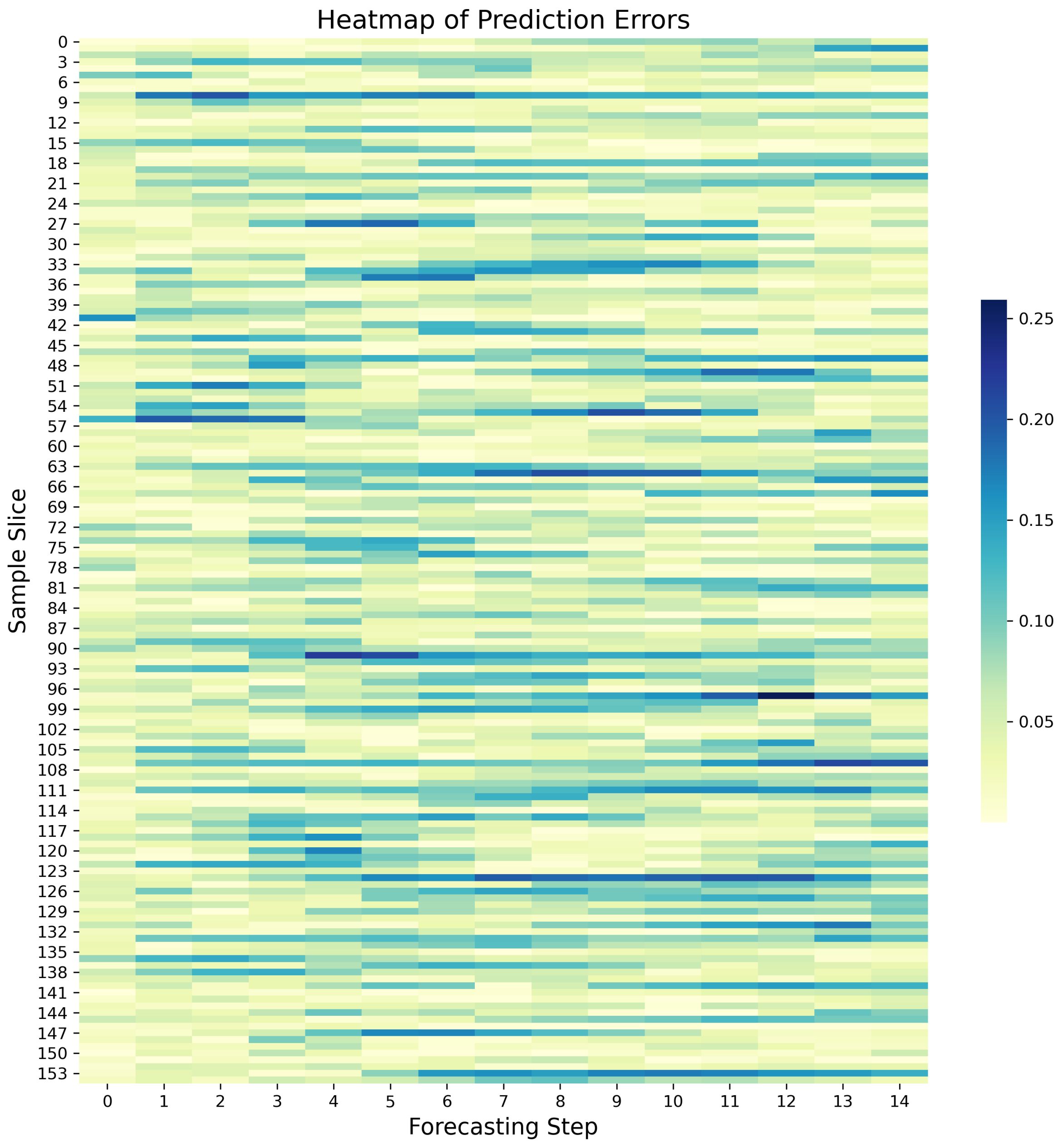


Figure 28: **Detailed MAE heatmap with all test results for timesteps $t = 1$ to $t = 15$.**