Master's Thesis

Masked Prediction of Time Series data Using Novel Machine Learning Models

Master's thesis in Complex Adaptive Systems

DineshKrishnan Gopalakrishnan
Niranjan Suresh

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024
www.chalmers.se
www.gu.se

Master's thesis 2024

Masked Prediction of Time Series data Using Novel Machine Learning Models

Development of a deep neural network model for the prediction of time-series data

DineshKrishnan Gopalakrishnan
Niranjan Suresh

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Department of Physics
University of Gothenburg
Gothenburg, Sweden 2024

Vehicle's Battery Modules Sensor Time Series Data Imputation using Time Series Transformer Networks
Development of a deep neural network model for the time-series data prediction
Dinesh Krishnan Gopalakrishnan
Niranjan Suresh

Supervisor: Henrik Ydreskog, Volvo Group
Supervisor/Examiner: Mats Granath, Department of Physics

Master's Thesis 2024
Master Program - Complex Adaptive Systems

Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

University of Gothenburg
SE-405 30 Gothenburg
Telephone +46 31-786 00 00

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Abstract

We are living in a time where data is the new oil, where the industries are data-driven. This change was expedited by the enormous amount of data that we are producing every second and the increase in computational power. At present the automotive sector is thriving in this data-driven model by their ubiquitous need for common and industrial purposes and the various data they are collecting to improve the sector as a whole. We can now see an automobile as not just a mechanical product but as a robot on wheels. Along with this data-driven model, the electrification of automobiles has revolutionized the industry. In the electrification process, the battery module is one of the key components that power the systems. To do this we must analyse the data from the battery modules for its efficient usage. However, due to certain hardware issues or if the vehicle is out of range and it cannot update the data we might lose data. This loss of data can obstruct the efficient usage of the data in machine learning models to optimize the system.

There are several methods to impute missing data, for example, there are statistical methods such as the Auto-regressive methods, which are limited by their time and the high cost of their computations. This thesis focuses on this problem and designing a neural network model for masked prediction of the Time series data. In this thesis, a Transformer Network is implemented for the masked prediction of the missing time series data.

In this thesis, we have built the machine learning model from scratch after weighing several factors. The data on which the model is trained is generated by the vehicle collected. This was led by pre-processing, later following the selection of the model. The model developed here is a variation of the transformer model, called the Time Series Transformer(TST), which predicts the missing values in the time series data. This model is then evaluated with suitable metrics by the model and the problem statement. The thesis aims to predict the missing values to improve the quality of the data collected and its quality usage to improve the performance of the vehicle.

Keywords: Time-series, Transformer, Deep learning, Time Series Transformer, Masked Prediction.

Acknowledgements

We extend our heartfelt gratitude to our thesis advisors, Deepak Guru Ganesan and Emil Johansson, for their continuous guidance during the thesis. Their profound insights and suggestions have been immensely valuable in this project. The accomplishment of the thesis objectives would not have been possible without their consistent support and motivation. We are equally thankful for the academic prowess and unwavering assistance from our supervisor and examiner, Mats Granath, particularly during challenging phases.

Furthermore, we wish to express our appreciation for the unwavering support and love from our friends and family. Their encouragement and backing have played a vital role in making this project a reality.

Dinesh Krishnan Gopalakrishnan and Niranjan Suresh,
Gothenburg, September 2023.

Contents

List of Figures

1
Introduction

Volvo Group AB is a globally renowned supplier of mobility solutions in the commercial vehicle sector. Their involvement and investment in data analytics are growing day after day, to ensure increased battery life of the vehicles.
A major component in the electrification process is the battery system. One key issue in this system is its efficient usage for its prolonged life. So it's important to have quality data without any missing values for its usage for improving this system, For this, the thesis contributes by creating a Time-series Transformer (TST).

1.1 About Volvo Groups AB

Volvo Group AB is the leading manufacturer of heavy trucks, construction equipment, busses, power solutions for marine and other industrial applications, and financing and services that increase the customers' up-time and productivity, having its headquarters in Goteborg, Sweden. They have production in 18 countries and serve around 160 markets worldwide employing 102,000. They have 8 distinct brands which consist of Volvo, Volvo Penta, Rokbak, Renault Trucks, Prevost, Nova Bus, Mack, and Arquus. They also partner in alliances and joint ventures with SDLG, Milence, Eicher, Dongfeng, and cellcentric.

Volvo Group is committed to changing society into a fossil-free environment and sustainable life can be seen from its introduction of a wide range of electric mobility solutions.

1.2 Thesis Objective

The objective of this thesis is to develop a reliable neural network model to do masked prediction of the multi-variate time series data. The crucial inquiries that can be addressed upon the completion of this research endeavor are outlined below.

1.2.1 Research Inquiries

1. Which network model can reliably predict the multi-variate time series data?
2. Whether the predicted data is by the original data generated?
3. Which evaluation metrics are appropriate for assessing the model's performance??

1.3 Scope

In an electric vehicle, the most important part is the battery module which powers the vehicle. One of the key factors affecting the vehicle's performance is the usage of the battery module. For high performance and increased range of the vehicle, the battery module must be used optimally. The battery module life and charging cycle too depend on how efficiently the module is being used by the drive system. To optimize this system, data collected from the vehicle can be used, but the issue is that there may be a loss of certain data in different time steps which makes the data hard to use for inference. To overcome this issue, this thesis aims to do masked prediction of the missing time series data.

The primary aim is to build a transformer-based neural network model that can be used to do masked prediction of the missing values and the complete data can be used to train any machine learning model over different use cases. The predicted data should also be compared with the original data by using evaluation metrics.

1.4 Goals

1. Conduct a comprehensive study of the data source to gain a thorough understanding of its significance. Employ various data visualization methods to analyze the data and enhance comprehension of each data point's importance.
2. Determine appropriate evaluation metrics during the preliminary investigation phase to assess the model's reliability. Provide sound justifications for the chosen metrics.
3. Perform data pre-processing to render the data suitable for project use. Aim to maintain the inherent properties of the data by exploring diverse pre-processing techniques and extracting valuable insights through visual representations.
4. Develop a robust neural network model as the subsequent goal. The model should align with the defined objectives, and its selection should be informed by the literature study conducted during the research phase.
5. The final goal is to compare the differences between the original and predicted data and incorporate methods to notch down the differences to ensure that the model performs robustly in all scenarios.

2

Theory

This chapter breaks down the technical ideas necessary for a detailed understanding of the thesis work.

2.1 Time-series Data

Time-series data is a sequence of data points that are recorded or collected at regular time intervals. This data tracks the evolution of a variable over time, such as stocks, temperature, etc. The regular time intervals can be daily, weekly, monthly, quarterly, or annually, and the data is often represented as a line graph or time-series plot.

Time-series data is grouped into various types using different criteria. Initially, the categorization can be based on the patterns observed over time, such as trends, seasonal variations, and irregularities. Another way to categorize is based on the number of features in the data, where it can either involve a single feature (Univariate Data) or multiple features (Multivariate Data) collected at a specific time..

2.1.1 The Trend

The data shows a trend when its value variably changes with time. An increasing value shows a positive trend and a decreasing value shows a negative trend. Analyzing the data for this trend is comparatively straightforward. Figure 2.1.1 shows a graph depicting a time series, illustrating a fairly straight downward trend without any recurring seasonal patterns, though there are some fluctuations observed throughout the timeframe..

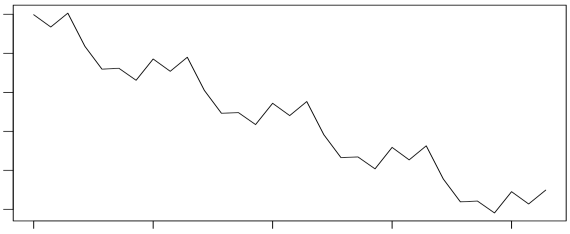

Figure 2.1.1: A linear downward trend time series plot

2.1.2 Seasonal time-series data

Periodicity in time series data becomes apparent when the data incorporates different seasonal elements such as weekly, quarterly, or annual repetitions. These seasonal variations may stem from natural factors like distinct seasons or weather conditions, or they could be artificial conventions like fashion or habits. Figure 2.1.2 illustrates a plot exhibiting seasonality, where a recurring pattern is observable.

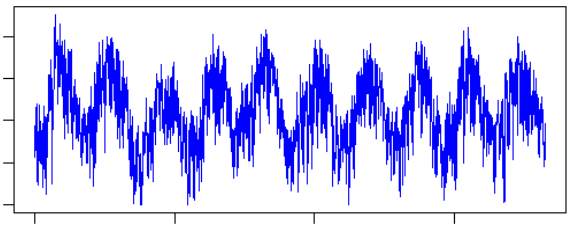

Figure 2.1.2: A periodic time-series plot

2.1.3 Irregular time-series Data

The time-series data, exemplified in figure 2.1.3, lacks a discernible pattern and is characterized by considerable noise. Such data poses challenges in understanding its distribution and conducting meaningful analyses.

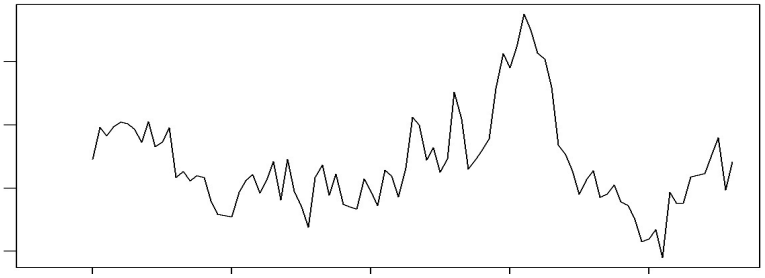

Figure 2.1.3: A representative time-series plot with irregularities and no discernible patterns

2.1.4 Univariate and multivariate Data

Time-series data can be classified according to the number of features it encompasses. When multiple features are collected or recorded at each time instance, it is referred to as multivariate data. Conversely, if only one feature exists in the data for each time instance, it is termed as univariate time-series data. This thesis specifically addresses multivariate time-series data.

2.2 Artificial Neural Networks

This section elaborates on the concepts of artificial neural networks(ANNs) and their working. ANNs are a subset of machine learning and are pivotal for deep learning algorithms. ANNs are inspired by the human brain, mimicking the way

that biological neurons signal one another. An artificial neural network comprises of numerous neurons and edges connecting these neurons. In this system, learning happens by increasing or decreasing the strength(i.e, weights) of the connection between these neurons over the entirety of the training process.

2.3 Feed Forward Neural Networks

A feed-forward neural network is an artificial neural network in which the connections between the neurons do not form a cycle. The feed-forward model is the simplest form of a neural network as information is only processed in one direction, but it can pass through multiple hidden nodes in that direction and never backward.

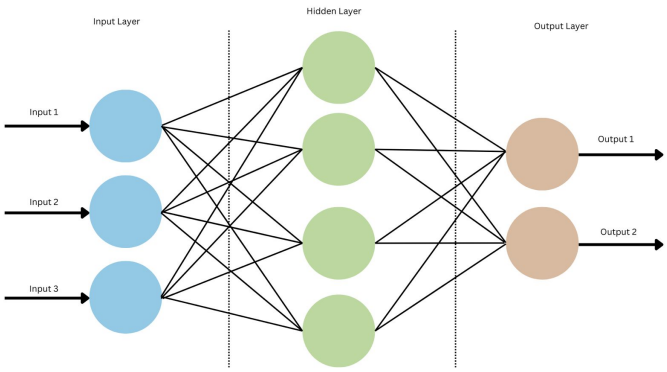

Figure 2.3.1: A simple feed-forward Neural Network model with one hidden layer

The purpose of a feed-forward neural network is to statistically approximate the function f(x) and perform downstream machine learning tasks. Training of a multilayer perceptron with hidden layers comprises forward propagation and backward propagation.[7]

For an n-dimensional input of $\vec{x} = \{x_1, \ldots, x_n\}$, and n- dimensional outputs of $\{y_1, \ldots, y_n\}$, the forward propagation is iteratively done to compute the value that each neuron in the next layer will take using the equation $\boldsymbol{y}^{(l)} = \sigma\left(\boldsymbol{W}\boldsymbol{y}^{l-1} + \boldsymbol{b}\right)$, where $\boldsymbol{W}$ is the weights and $\boldsymbol{b}$ is the bias present in the neurons of layer l. σ is the activation function used.

The difference is calculated between the predicted output $\boldsymbol{y}_{(out)}$ and the target y which is known as the cost and the function performing this the cost function. This quantifies how well a machine learning model performs. This cost function must be minimized by adjusting the models parameters(which are the weights and biases) during training.

2.4 Recurrent Neural Networks

A recurrent neural network (RNN) belongs to the category of artificial neural networks designed for processing sequential or time series data. They find frequent

application in addressing ordinal or temporal challenges, including tasks like language translation, natural language processing (NLP), and others.

RNNs are distinguished by their memory as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depends on the prior elements within the sequence. Figure 2.4.1 shows the unfolded and folded architecture of a RNN [4]. This image to the left shows the rolled version of RNN which shows the weight $\boldsymbol{W_{x_h}}$ between the input x and the RNN Cell h. The weight between the output sequence y and the RNN cell h is $\boldsymbol{W_{h_y}}$.

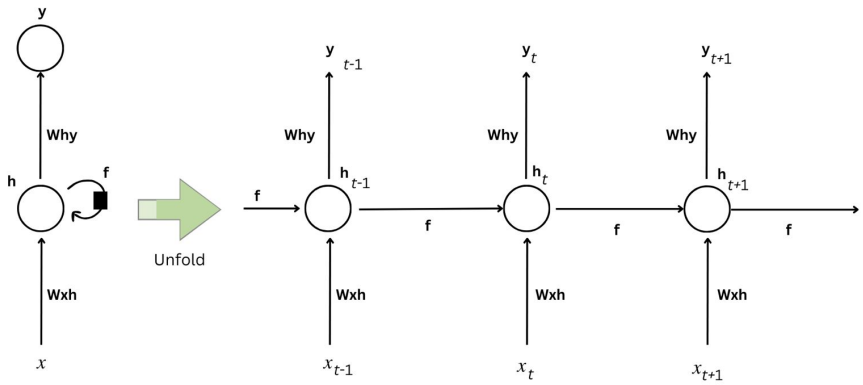

Figure 2.4.1: A RNN structure in both a folded and an unfolded configuration [3][2]

At each time step t, the RNN takes an input x_t and produces an output term y_t, along with an RNN term h_t, which serves as input for the next neuron. This sequential information transmission continues to subsequent neurons. Equations (2.1) and (2.2) outline the computation for h_t and the output term y_t:

$$h_t = g_1 \left(W_{hh} h_{t-1} + W_{xh} x_t + b_h \right) \tag{2.1}$$

$$y_t = g_2 \left(W_{hy} h_t + b_y \right) \tag{2.2}$$

Here, W_{x_h}, W_{h_h}, W_{h_y}, b_h, b_y are temporary coefficients, and g_1, g_2 are activation functions.

A notable feature of RNNs is their dynamic architecture, adapting to the length of the input sequence. Due to the temporal persistence of weights, updates occur across time, resulting in fewer trainable parameters. However, RNNs also exhibit several drawbacks, which will be explored in the subsequent section.

Even though the RNNs are advanced relative to a simple feed-forward neural network they too have their limitations. These limitations are given below:

- Inability of parallel computations
- Vanishing or Exploding gradient problem

To counteract these problems the transformer model was developed which is explained in the next section.

2.5 Transformer Model

A transformer model is a neural network model that learns the context and thus the meaning by tracking relationships in sequential data like the multivariate time series data in this thesis. It is primarily used in the field of Natural Language Processing and Computer Vision. Now let's look at the model architecture for a detailed understanding of the said network.

2.5.1 Transformer Architecture

The Transformer architecture eschews recurrence and instead relies entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization [8]. Below the model architecture is visualized.

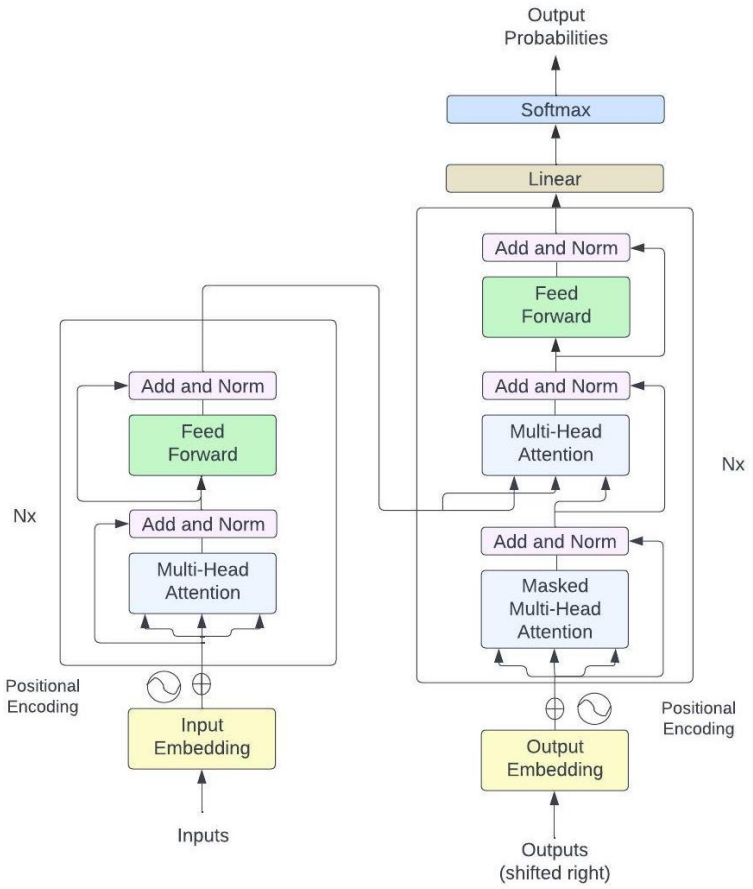

Figure 2.5.1: The Transformer Architecture as formulated in [8]

The transformer architecture consists of two main parts, namely the encoder block and the decoder block.

The encoder is composed of a stack of six identical layers and each layer has two sub-layers of which the first is the multi-head self-attention mechanism and the second is a simple position-wise fully connected feed-forward network, each of these sub-layers have a residual connection followed by a layer-normalization.

The decoder block is also composed of a stack of six identical layers, in addition to the two sub-layers as in the encoder block the decoder has a third sub-layer which performs multi-head attention over the output of the encoder stack. And the self-attention sub-layer in the decoder stack is modified to prevent positions from attending to subsequent positions. Given that the output embeddings are off-set by a position and also considering the masking, it only does prediction based only on the previous outputs.

Working of the attention mechanism is based on mapping a query and a set of key-value pairs to an output, where all the said entities are vectors. The attention is

calculated as following:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_v}})V \tag{2.3}$$

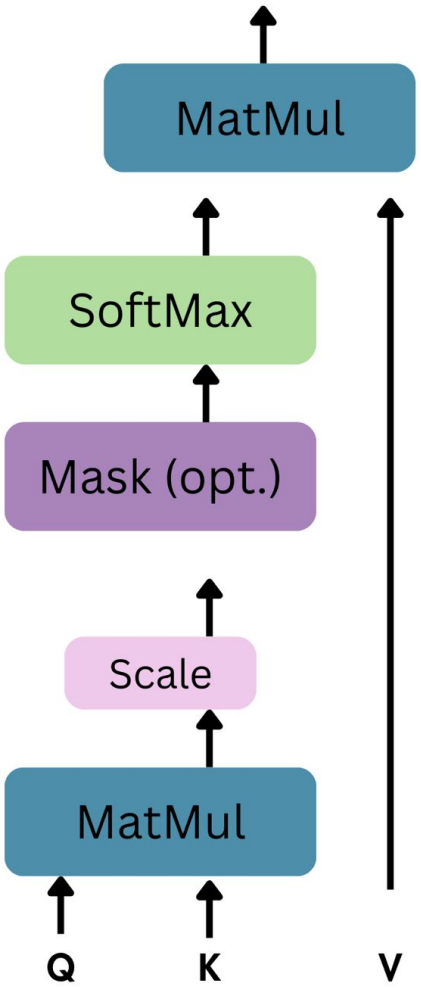

Figure 2.5.2: The Attention Mechanism

Where Q,K and V are the query, keys and value, and d_k is the dimension of the key. In-order to get the weights of the values, we do dot products of the query with all the keys divided by $\sqrt{d_k}$ and then apply softmax function.
Multi-head attention is an attention mechanism module which runs the attention mechanism several times in parallel. The independent attention outputs are then concatenated and linearly transformed into the expected dimension. Multi-head

attention allows the model to jointly attend to information from different representation sub spaces at different positions.[8]

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.4}$$

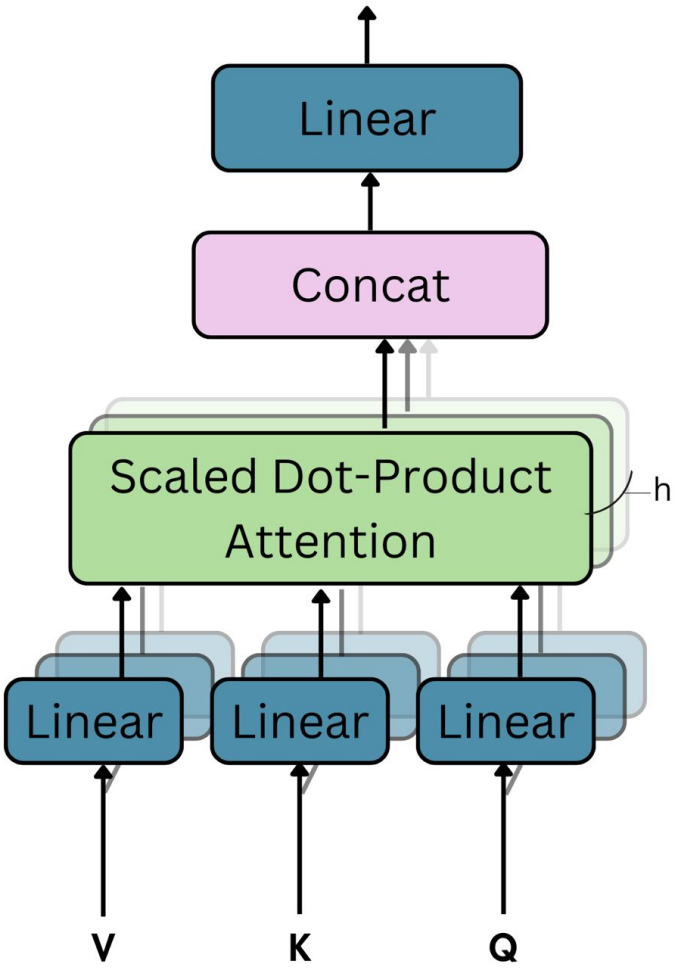

Figure 2.5.3: Multi-Head Attention

where $head_i = Attention(QW_i^Q)$. Because of this there is a reduced dimension of each head by which, the total computational cost is equivalent to that of single-head attention with full dimensionality.

3
Methodology

This part outlines the flow of the entire thesis, from the literature review to getting the predicted results in a detailed manner. The following flowchart explains the steps that were involved in the thesis project.

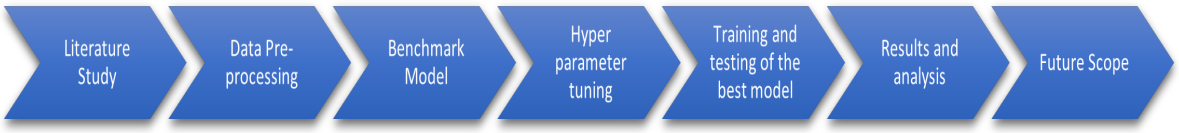

Figure 3.0.1: Work Flow of the Thesis

3.1 Literature Review

The literature review was started with the thought of getting the methods and procedures that were previously successful in predicting and forecasting similar types of time series data. The different articles, papers, and resources used were taken from Google and Google Scholar. The decision on pre-processing steps, data visualizations, and the model needed for the use case were all decided during the literature study. The detailed description of the literature studied for this thesis is mentioned in 2.

3.2 Data set

The data comes from the Field test trucks, which are processed and retrieved from the EMOB-General Signals category. The data from the sensors is transmitted as a Control Area Network (CAN) message, comprising time-dependent multivariate information. We took data from 9 different vehicles to incorporate different models under different conditions. Figure 3.2.1 below shows the data from one single vehicle with a gap less than 10 seconds between each timestamp.

3.3 Data Visualisation

Data visualization helps us understand the data that is present when not much information is provided. It helps us to track down trends, understand the underlying facts that the data represents and helps us to arrive at valuable conclusions out of

	actcurrent_ep1_x_b3	actsoc_ep1_x_b3	ambientairtemperature	actpackvoltage_ep1_x_b3
0	-0.150000	79.842499	24.937500	723.400024
1	-0.249999	79.842499	24.937500	723.400024
2	-0.400001	79.842499	24.937500	723.349976
3	-0.750000	79.842499	24.937500	723.299988
4	-1.250000	79.842499	24.937500	723.250000
...	...	...	...	...
1019	-12.685470	60.319023	9.726877	686.882446
1020	-12.685470	60.319023	9.726877	686.882446
1021	-12.685470	60.319023	9.726877	686.882446
1022	-12.685470	60.319023	9.726877	686.882446
1023	-12.685470	60.319023	9.726877	686.882446

Figure 3.2.1: A small slice of the dataset with the 10 important features

it. For a time series data set, it is very important to properly visualize the data as it helps us in understanding the trends that each feature follows as time progresses and helps to get an insight into how the forecast or prediction might look. Different visualization techniques are being employed for the time series data as shown below.

3.3.1 Plotly graphs

Various data features are plotted as pulse graphs in Python using the plotly module. The pulse graphs use unique colours to differentiate multiple features and plot each feature across time in the *x axis*. Each attribute's distribution range is shown by the *y axis*. Users have the option to choose a particular attribute within the data and concentrate exclusively on analyzing that feature using the real-time graphs.

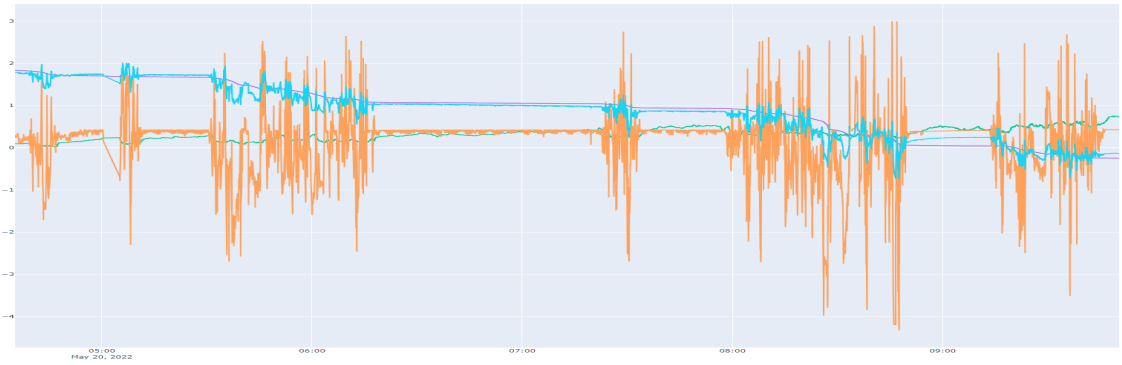

Figure 3.3.1: Standardized time series data for the 4 important features using plotly graph

3.4 Data Pre-processing

Pre-processing data is a critical step in the machine-learning process. Data pre-processing involves cleaning the data, removing missing values, and modifying the

data based on the use case. These processes are carried out because the raw data obtained from the sensors is frequently inconsistent, incomplete, and may contain errors. The pre-processing started with choosing the signals required to do the analysis. These signals were fetched from various domains in the form of CAN signals. These CAN signals are the different values that are received from the sensors placed at the battery modules that tell the characteristics of the module at each time stamp from the different field test trucks. The different steps of pre-processing performed are mentioned in the below sections.

3.4.1 Data Loading

The signals that we used for this thesis are taken from the the field test trucks in the form of CAN signals, as mentioned in 3.2 and we considered taking 10 important features from the signals obtained. This obtained data frame was processed later such that some columns were removed the outliers were dropped and other changes were performed to make the data clean enough to be sent inside the model.

3.4.2 Choosing of Battery Pack and dropping of values

The obtained signals contained almost 80 features for 6 different battery packs. For this process, we chose one battery pack that had less Nan values. Battery pack 3 had the least and only that pack is considered. Now, the most important 10 columns were extracted, which include current, voltage, ambient air temperature etc. The other columns were removed using the 'drop' function. After this process, for initially sending the data set inside the model, it is necessary to have no Nan values present in the data set. So, the Nan values can either be imputed or can be removed. We decided to remove them. It was removed under the condition that if any feature of one particular timestamp had a Nan value, then that data point was removed from the data set.

3.4.3 Removal of outliers

After sorting the columns, the obvious outliers are removed from the data set.(i.e) soc in negatives, current values greater than +- 200 and voltage greater than +-700. These thresholds are fixed based on the standard set for the battery modules.

3.4.4 Consideration of only 1 status

The other values that come along with the features are the status at which the timestamp of the truck is currently at. There are 3 different statuses at which the truck can be. For initial considerations, the status 'Driving Not charging' is chosen. The reason to choose this status is that this status undergoes major changes in the sensor values than the other statuses and the prediction of the features in this status is more important than the rest.

3.4.5 Batch separation conditions

Batch separation is one of the most important processes of this thesis as it helps us in dividing the data set retrieved based on the different conditions under which the data was recorded. Doing this makes a batch have only a particular set of data. Also, the model is developed in such a way that it takes in data as batches, which helps the model to learn different scenarios from different batches.

For the batch separation process, the time difference between each timestamp is noted and that is stored as a separate feature. This is performed for all the 9 vehicles. The data is grouped based on the different vehicles present. By doing this, we ensure that each batch has data from one particular vehicle. That also acts as a condition for batching. After calculating the differences, the filename difference between the consecutive timestamps is also recorded as either 1(different filenames) or 0(same filenames). The filename is the labeling given to each timestamp value during the retrieval of data from the CAN signals by the data engineering department. The change in filename between timestamps indicates a change in the scenario or condition or the date on which the truck was tested.

After performing all of the above mentioned procedures, the entire data set is divided into batches based on the condition, and Batch IDs are also generated for the same. The Batch IDs are made as the indices of the data set.

3.4.6 Standardization using z-score

Scaling of the data is a very important process while performing data pre-processing. This helps us to maintain the data values in the same range, which helps us maintain the constancy of the dataset taken while sent inside the model for training. In this thesis, Standardization using z-score is considered to scale down the values. Standardization is done across each feature in the data set and is performed based on the formula

$$Z = \frac{X - \mu}{\sigma} \tag{3.1}$$

where:

- Z is the standardized value (z-score).
- X is the original value of the data point of that feature.
- μ is the mean of the distribution.
- σ is the standard deviation of the distribution.

Figure 3.3.1 is the plot of the standardized data for a small slice of the entire data set. After standardization, the minimum value of the dataset was -4 and the maximum was 4. After the process of standardization, the data set is ready to be used for training the model. .

3.5 Model

The core of our model is based on a Transformer encoder as developed in the original transformer work by Vaswani et al., however, we do not use the decoder part of this architecture.
Before we go into our model architecture let's see how the data is sent into the model. As we have described before we pre-process our data but since we are handling time series data we will have varying lengths of the data. In order to have consistency during training we have a set length of the data that is been sent into the model. That is if we have a dataset that exceeds the set length we slice the data and if it is below the set length we do padding to get it to the set length.

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.000000	-1.124762	0.711968	0.000000	0.132253	1.20566	0.187323	-5.533307	-6.594398	0.0	0.0	-6.443034
1	0.000000	-1.124762	0.711968	0.000000	0.132253	1.20566	0.187323	-5.533307	-6.517851	0.0	1.0	-6.443034
2	0.538398	-1.124762	0.711968	0.000000	0.132253	1.20566	0.000000	-5.533307	-6.517851	0.0	1.0	-6.443034
3	0.538398	-1.124762	0.711968	1.218130	0.140493	1.20566	0.187323	-5.533307	-6.517851	0.0	1.0	-6.443034
4	0.538398	-1.124762	0.711968	1.217726	0.137746	1.20566	0.187323	-5.533307	-6.517851	0.0	1.0	-6.443034
...	...	...	...	...	...	...	...	...	...	...	...	...
95	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000
96	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000
97	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000
98	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000
99	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.0	0.0	0.000000

Figure 3.5.1: A dataset being padded before inputted into the model.

Now let's see the mechanism of the model architecture. Each training sample X is a multi-variate time series of length w and m different variables, constitutes a sequence of w feature vectors (i.e., X = $x_1, x_2, x_3 ... x_w$). In short one row in our dataset is one feature vector(i.e., one data point).
These feature vectors x_t at each time step t is simultaneously inputted into the model. And then these are linearly projected onto a d-dimensional vector space, where the projections are indicated by u_t. This d-dimension is similar to the transformer model dimension.

$$u_t = W_p x_t + b_p \tag{3.2}$$

Where W_p and b_p are learnable parameters and u_t are the model input vectors, which are analogous to the word vectors in NLP Transformer.
The transformer model is independent of the position of the input data and since we are dealing with time series data it is important to include the temporal sense to the data, for which we add positional encoding to the input vectors before the data is inputted to the attention layers of the transformer encoder.

Once we have done with the input encoding, the encoded data is inputted to the transformer encoder where the learning happens. There are attention layers that will learn the inter-dependencies between the various features along with the positional embedding making the model learn the relation of the data with respect to

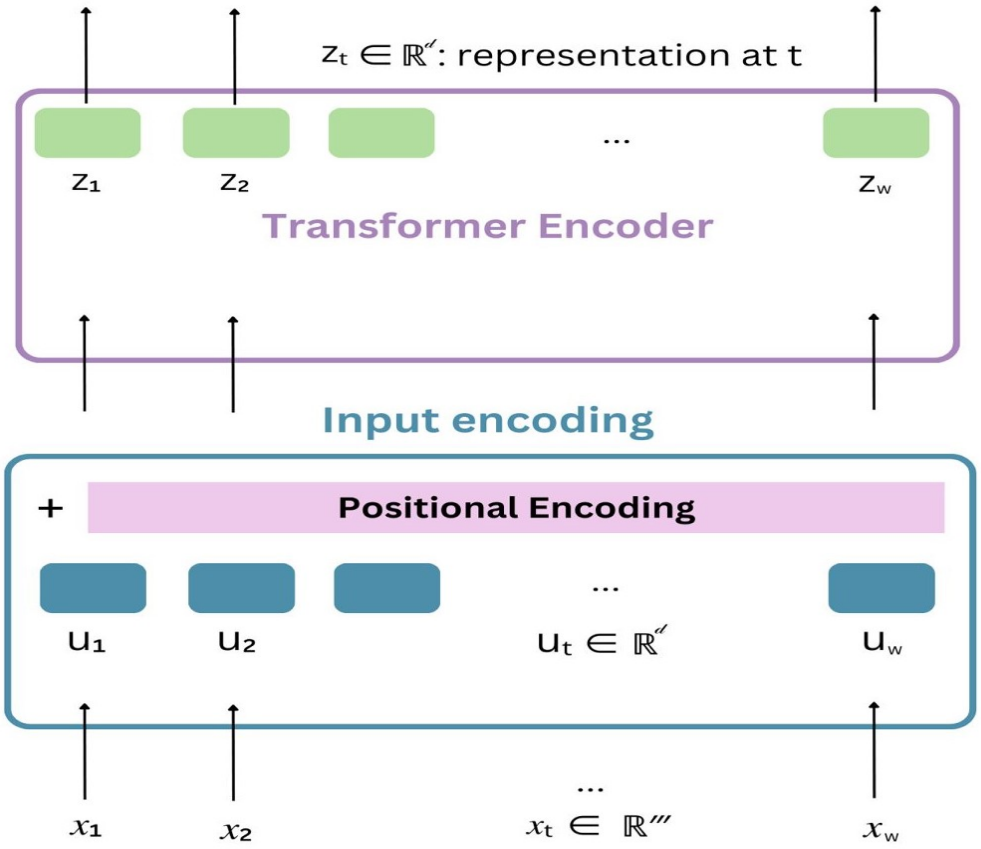

Figure 3.5.2: Model Architecture

time.

Previously we have seen how the data flows through the model, now let's see how the model is trained on the given data.

The initial step for the training is the creation of the mask. A binary mask(M) with dimensions similar to the training dataset is created, it is created independently for each training sample. Now the input is masked by element-wise multiplication with the binary mask, $X' = X \cdot M$.
Here masking is done column-wise of length w(length of the data), i.e., corresponding to a single variable in the multivariate time series. Also, the masking is controlled by a ratio called the masking ratio(r), the data is masked proportional to r.

Each masked segment will have a state transition probability such that the length of the masked segment follows a geometric distribution with mean l_m and this is followed by an unmasked segment of length $l_u = \frac{1-r}{r} l_m$.
Here we control the length of the mask instead of going for a constant length value because very short masked sequences (e.g., of 1 masked element) in the input can often be easily predicted with good approximation by replicating the immediately preceding or succeeding values or by the average thereof. To obtain enough long

	0	1	2	3	4	5	6	7	8	9	10	11
0	True	False	False	True	False	False	True	False	False	False	False	False
1	True	False	False	False	False	False	True	False	False	False	False	False
2	False	False	False	False	False	False	True	False	False	False	False	False
3	False	False	False	False	False	True	True	False	False	False	False	False
4	False	False	False	False	False	True	False	False	True	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
95	True	True	True	True	True	True	True	True	True	True	True	True
96	True	True	True	True	True	True	True	True	True	True	True	True
97	True	True	True	True	True	True	True	True	True	True	True	True
98	True	True	True	True	True	True	True	True	True	True	True	True
99	True	True	True	True	True	True	True	True	True	True	True	True

Figure 3.5.3: Binary Mask.

masked sequences with relatively high likelihood, a very high masking proportion r would be required, which would render the overall task detrimentally challenging. For this case, we have done this masking pattern because it tends the model to learn to attend both to preceding and succeeding segments in individual variables, as well as to existing contemporary values of the other variables in the time series, and thereby learning to model inter-dependencies between variables.

At the end of the data flow through the model the model outputs the estimates of the full input vector x_t, but only the predictions of the masked values will be considered during the computation of the loss function and hence it determines how the model is trained by its influence in the updation of the learnable parameters.
In the next section, we will discuss the evaluation metrics that were considered in the study.

3.6 Evaluation Metrics

After the prediction of the missing values from the best model, it is important to see how precise the values are in comparison to the original values so that to validate the predictions being made. This brings us the need to calculate the evaluation metrics for the test run performed. Different evaluation metrics can be performed for time series data based on the requirements, which can be both visual and quantitative.

3.6.1 Visual Metrics

The visual way of representing the obtained predictions is the simplest method of approaching or evaluating the model performance. The different visual techniques employed for this thesis are mentioned below

3.6.1.1 Prediction vs original plots

This contains the plot of the obtained predicted values in comparison to the ground truth of that particular timestamp. This helps us get a visual idea of how matching the predictions are with the ground truth values. The plots are obtained with the

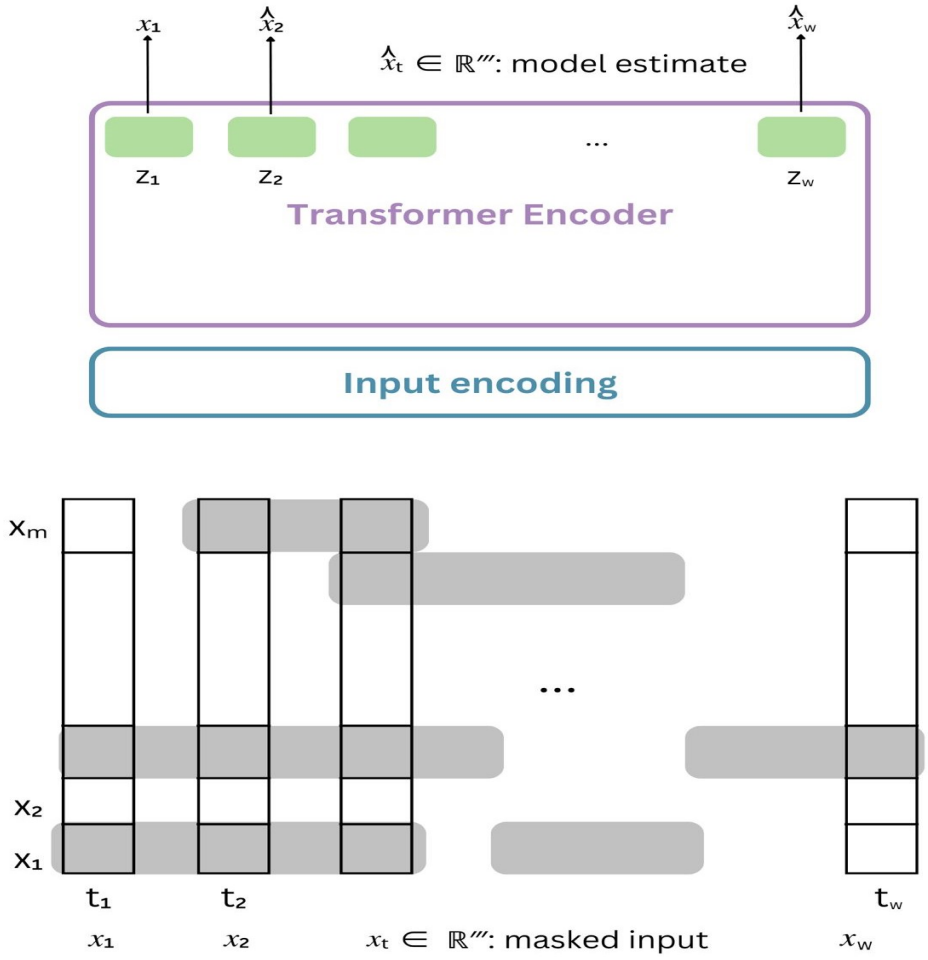

Figure 3.5.4: Model Training

help of the plotly package as it helps us to have an interactive view of the different features.

3.6.1.2 Boxplots of features

A boxplot can be used to graphically compare predicted and actual values by displaying their distributions and summary statistics. By plotting the features of predicted and ground truth values side by side, we can compare the medians, outliers range and IQRs. This helps us in understanding the spread of the values which helps us to find the differences between the data sets.

3.6.2 Quantitative Metrics

3.6.2.1 Mean absolute error

The mean absolute error (MAE) is characterized as the average difference between the actual values and the predicted values within the dataset.[6] This metric is very useful in predicting the performance of the model taken,(i.e.), it acts as an

accuracy metric. MAE is a straightforward measurement between ground truth and predictions. This helps us in knowing how much error the predictions hold and can be evaluated based on that. The range for MAE ranges between 0 to ∞ [5]. The equation for MAE is as below (3.3)

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \tag{3.3}$$

3.6.2.2 Root Mean Square Error

Root mean square error is the metric that calculates the error between the predicted and actual values by taking the mean of the squared differences between the predicted and the actual values,[1] i.e., the error, and then taking the square of it to calculate the metric. RMSE can be used as an evaluation metric to compare with different models as a comparison. RMSE is an even more strict metric than MAE as it gives strict penalties for the outliers between the ground truth and the predictions, as the errors are squared. The RMSE for a particular data set and its prediction is calculated using the formula (3.4) [1]

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.4}$$

4

Results

This chapter consists of the results and the corresponding discussions. During our study, we ran the network for different hyperparameters to analyze the network's behavior for different situations and to visualize them. The setup for each simulation will be explained below along with the graph of the Training & Validation curve and the actual & predicted values.

Each simulation setup will include, the number of features, learning rate, Early stopping patience, Learning rate reduction factor, Masking length, and the masking ratio. These are the Hyper-parameters associated with the machine learning algorithm we have developed. By changing these values in different combinations, we ran different simulation setups. In this study, we ran the network for numerous simulation setups out of which we have chosen five different setups which are delineated below.

4.1 Simulation Setup 1

- Number of Features - 4
- Learning rate - 0.0001
- Early stopping patience - 10
- Learning rate reduction factor - 0.5
- Masking length - 3
- Masking ratio - 0.2
- Optimizer - RAdam

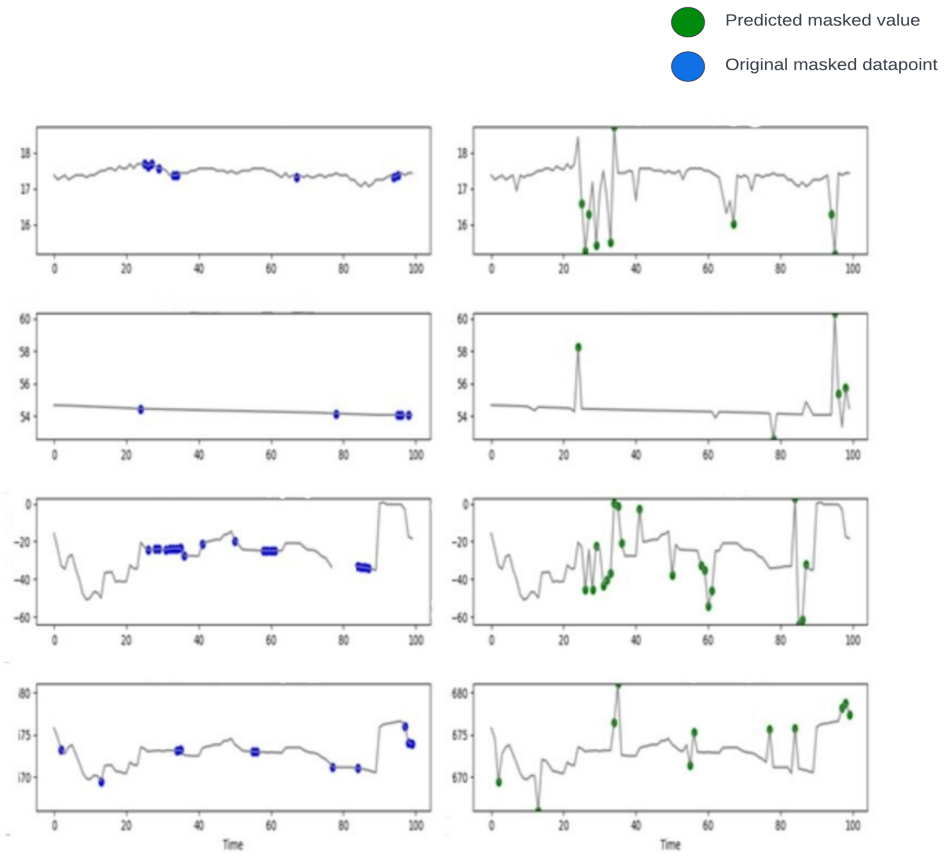

Figure 4.1.1: Original(left) vs Predicted(left) masked values for setup 1. The data points with circles are the masked values and the rest are the values not considered for prediction

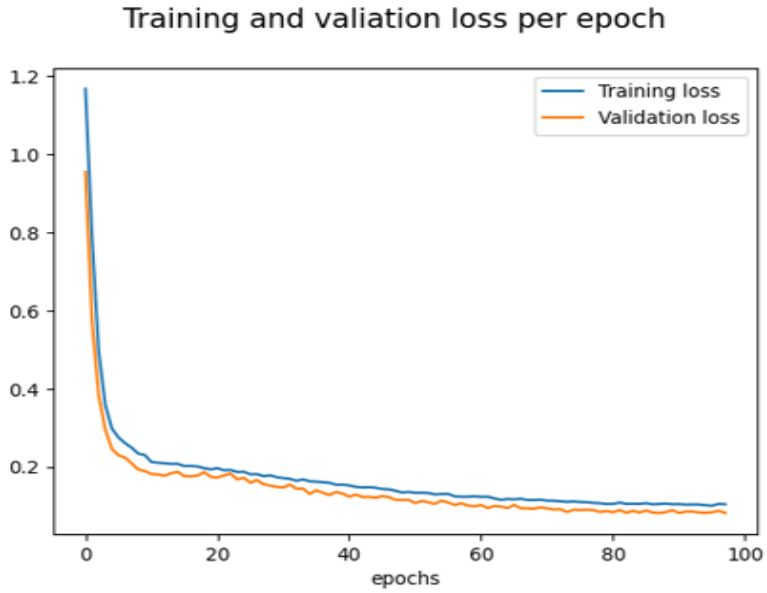

Figure 4.1.2: Training and validation loss curve for setup 1

Features	MAE score	RMSE Score
1	0.303	0.756
2	0.794	2.18
3	1.421	4.624
4	0.427	1.219

Figure 4.1.3: MAE and RMSE score of the features for setup 1

As detailed above this setup is run with the mentioned combination of the hyper-parameters. Here out of all the features collected from the truck we have selected only four features that have more impact in understanding the battery modules of the truck after a comprehensive exploratory data analysis.

From observing Figure 4.1.2 we can see that both the training loss and validation loss are close enough to indicate a good fit but there is a relatively slight dip in the validation loss going slightly below the training loss. The MAE and RMSE scores show that predictions are low, but if we look at Figure 4.1.1 we can see that the predictions seem to be wavering from the original values and for some features, the scale of variation is quite tolerable in the range of -10 to $+10$.

4.2 Simulation Setup 2

- Number of Features - 4
- Learning rate - 0.0001
- Early stopping patience - 10
- Learning rate reduction factor - 0.5
- Masking length - 6
- Masking ratio - 0.3
- Optimizer - RAdam

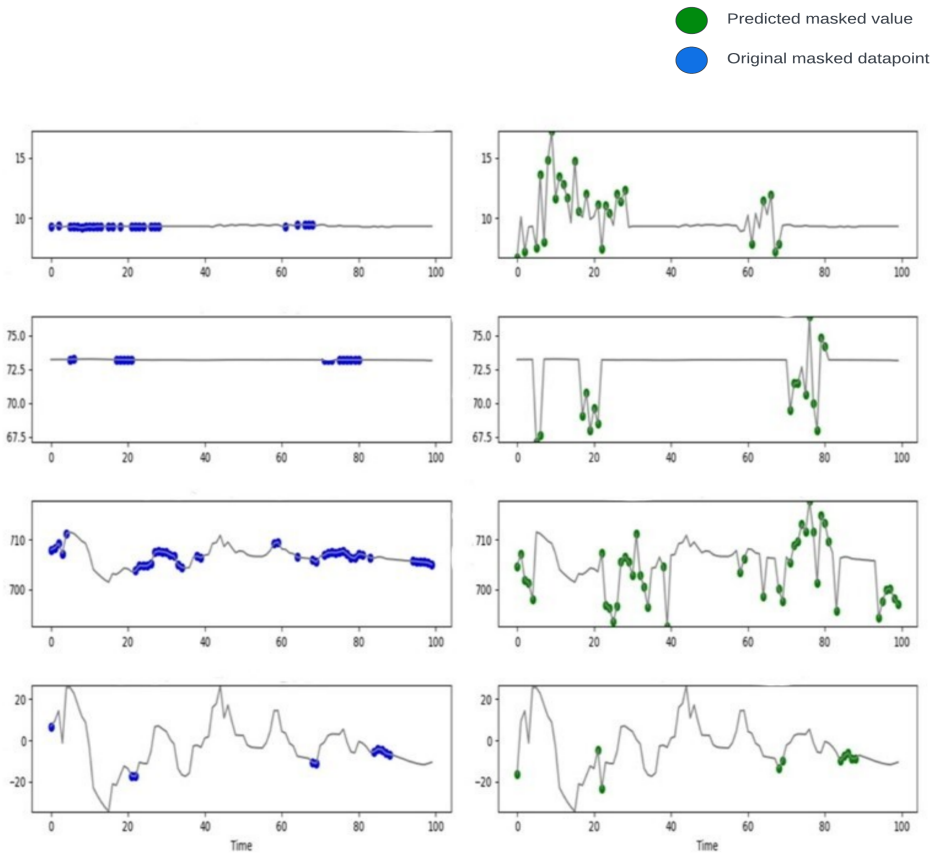

Figure 4.2.1: Original(left) vs Predicted(left) masked values for setup 2. The data points with circles are the masked values and the rest are the values not considered for prediction

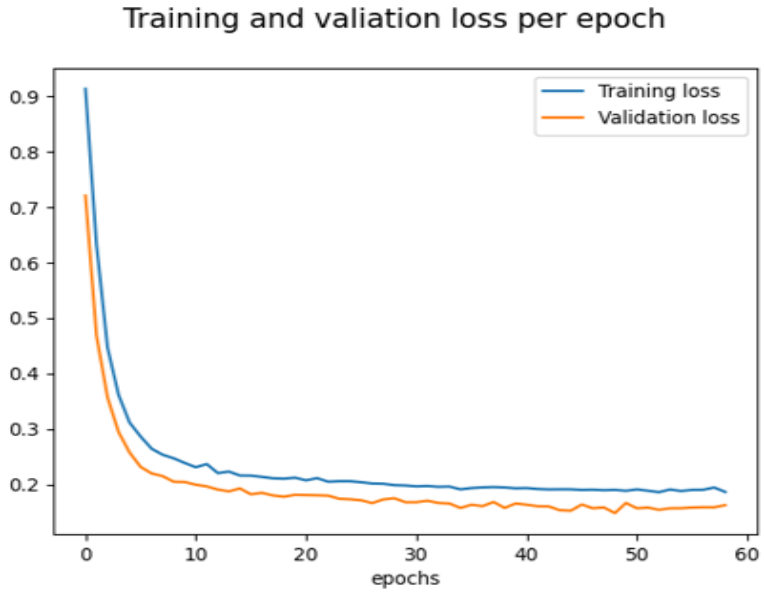

Figure 4.2.2: Training and validation loss curve for setup 2

Features	MAE score	RMSE score
1	0.457	1.058
2	1.598	3.362
3	2.422	6.008
4	1.046	2.034

Figure 4.2.3: MAE and RMSE score of the features for setup 2

This simulation setup is run with the hyper-parameters mentioned above which are quite similar to the simulation setup 1 but here we have increased the masking length to 6 and the masking ratio to 0.3. Here by observing Figure 4.2.2 we can see that there is a clear gap between both the training loss curve and the validation loss curve with no overlaps. And also the MAE and RMSE scores have increased as compared to the first setup. Here the values of the predicted as compared to the original one vary much more than the setup 1. Knowing that masking length and masking ratio are the only differences from setup 1 we infer from this that the increase in the mask must have made demanding for the model to predict longer sequences. Also, the increased masking ratio has masked more features increasing the training loss and the model might be exposed to more information during validation, leading to lower loss, as observed from Figure 4.2.2.

4.3 Simulation Setup 3

- Number of Features - 4
- Learning rate - 0.0001
- Early stopping patience - 10
- Learning rate reduction factor - 0.5
- Masking length - 10
- Masking ratio - 0.25
- Optimizer - RAdam

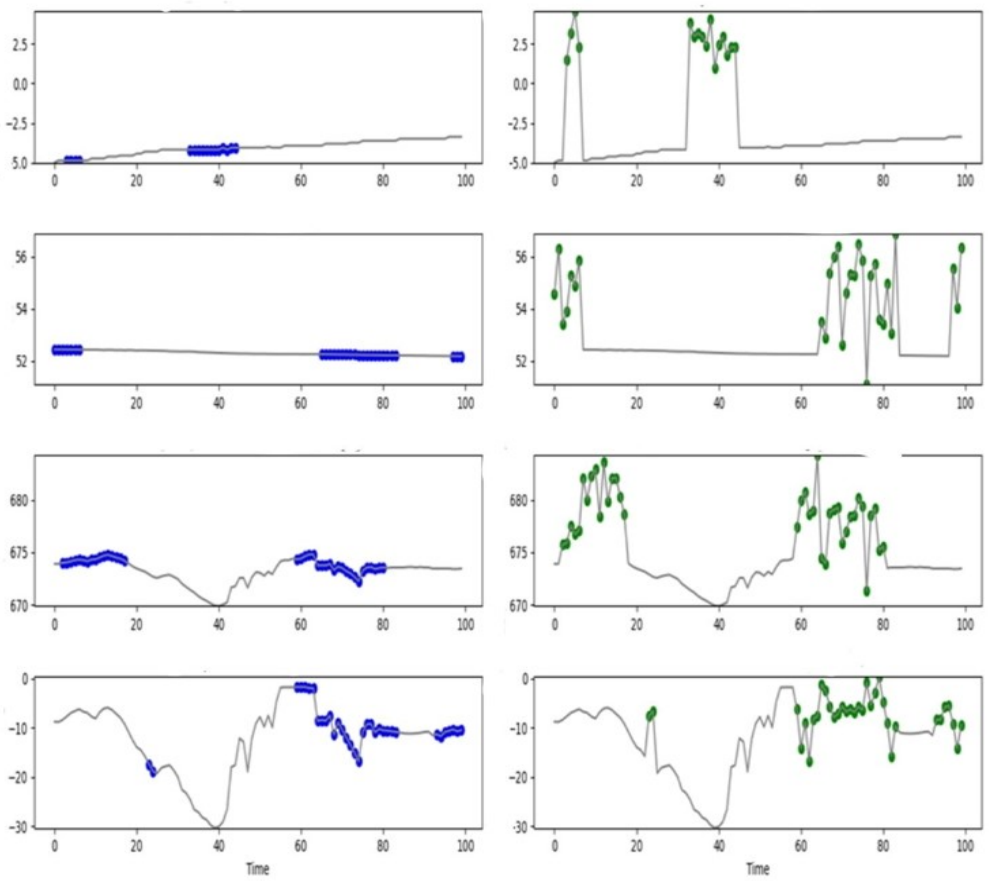

Figure 4.3.1: Original(left) vs Predicted(left) masked values for setup 3. The data points with circles are the masked values and the rest are the values not considered for prediction

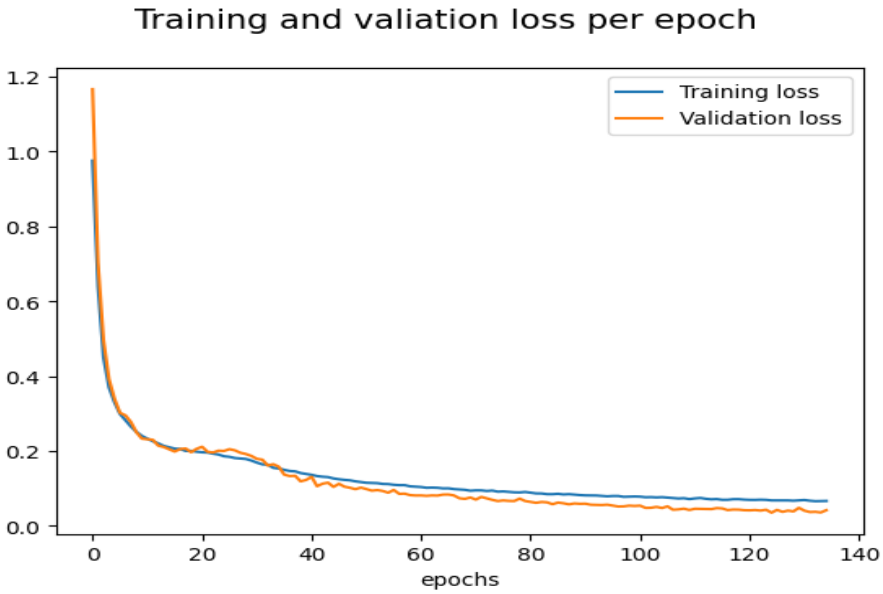

Figure 4.3.2: Training and validation loss curve for setup 3

Features	MAE score	RMSE score
1	1.301	3.2801
2	1.423	2.614
3	1.413	4.507
4	1.546	3.378

Figure 4.3.3: MAE and RMSE score of the features for setup 3

In this simulation setup, we run the network with the above-mentioned hyper-parameters which are similar to setup 2, where the masking length and masking ratio are changed. But here we have increased the masking length and decreased the masking ratio. By observing Figure 4.3.2 we can see that the gap between the training loss and the validation loss has decreased as compared to setup 2 and also the MAE and RMSE scores for this setup lie between that of the previous two setups, lower than the of setup 2. Again here when observing Figure 4.3.1 we can see that the predicted values are varying within a tolerable range as that observed in setup1. And similar to setup1 some features show huge variations, here the first and second graphs of Figure 4.3.1 show relatively huge variations while the third and fourth graphs show good predictions within the tolerable range.

4.4 Simulation Setup 4

- Number of features - 10
- Learning rate - 0.0001
- Early stopping patience - 10
- Learning rate reduction factor - 0.5
- Masking length - 3
- Masking ratio - 0.25
- Optimizer - RAdam

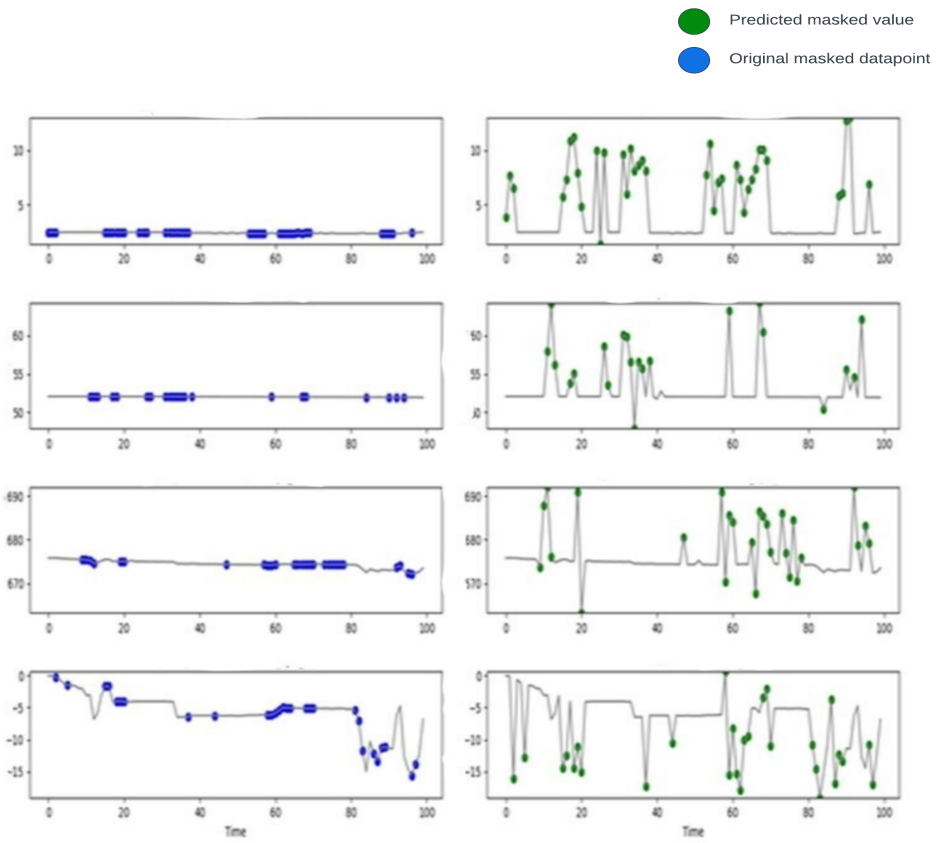

Figure 4.4.1: Original(left) vs Predicted(left) masked values for setup 4. The data points with circles are the masked values and the rest are the values not considered for prediction

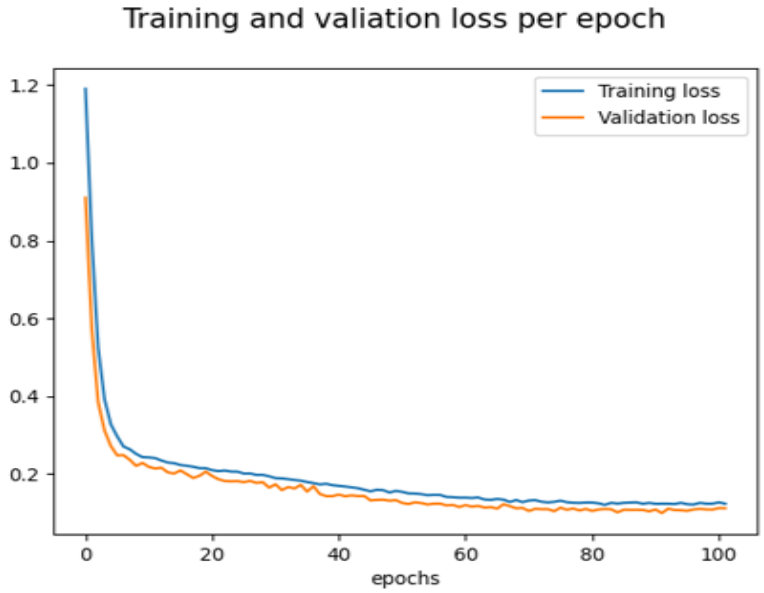

Figure 4.4.2: Training and validation loss curve for setup 4

Features	MAE score	RMSE score
1	2.602 e+06	8.13 e+06
2	5.131e+04	1.615 e+04
3	0.753	2.263
4	0.807	2.638
5	0.838	2.992
6	0.816	3.244
7	0.189	0.719
8	0.115	0.439
9	0.083	0.297
10	0.734	0.291

Figure 4.4.3: MAE and RMSE score of the features for setup 4

In this simulation setup we have run the network with the above-mentioned hyper-parameters and the major difference between this setup and all the previous setups is that, here we have used 10 input features to train the model as compared to the 4 input feature in the previous setups. The remaining values of the hyper-parameter are similar to setup1. This inclusion of more features has a noticeable decrease in the MAE and RMSE scores. However, the predicted values as opposed to the original values as observed in figure 4.4.1 are similar to those observed in setup1 and setup3.

4.5 Simulation Setup 5

- Number of Features - 4
- Estimators - 1000
- Learning rate - 0.01

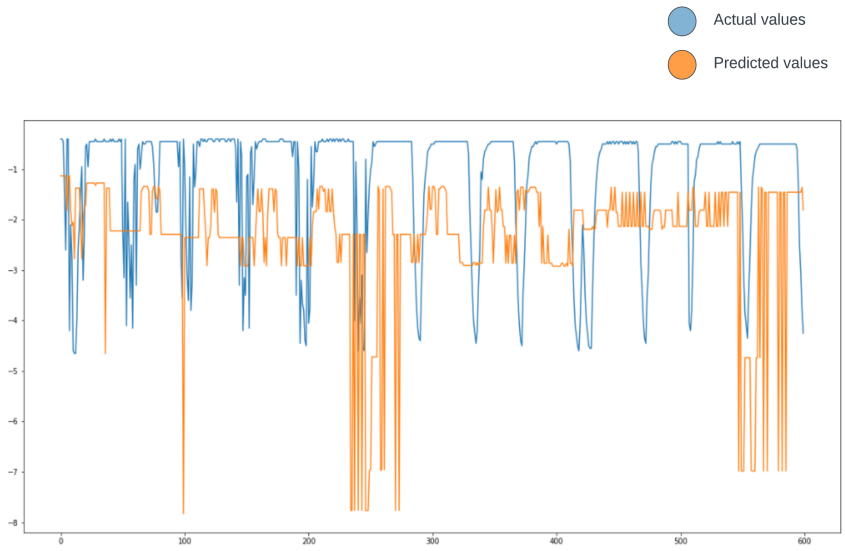

Figure 4.5.1: Original vs Predicted values of the 1st feature for setup 5 using XGBoost

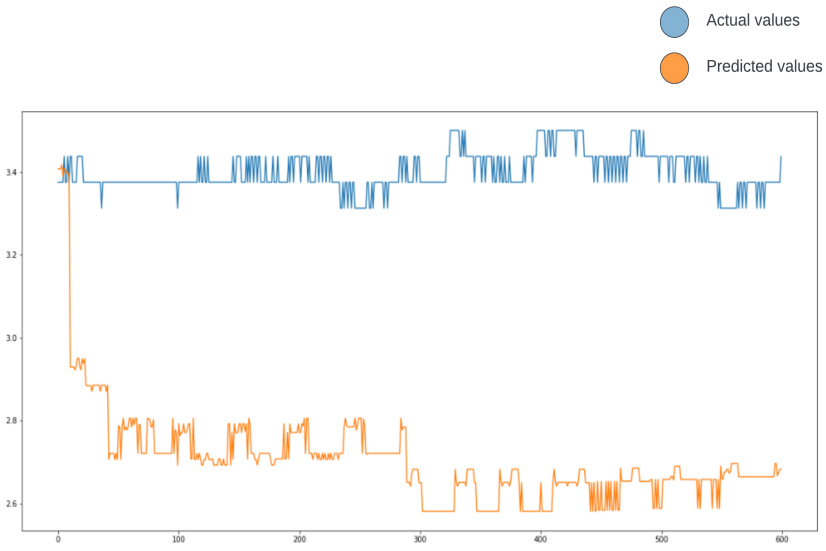

Figure 4.5.2: Original vs Predicted values of the 2nd feature for setup 5 using XGBoost

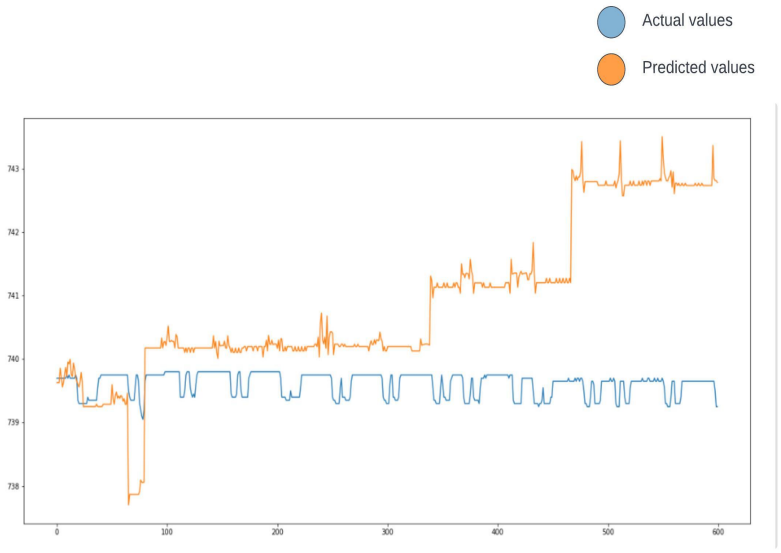

Figure 4.5.3: Original vs Predicted values of the 3rd feature for setup 5 using XGBoost

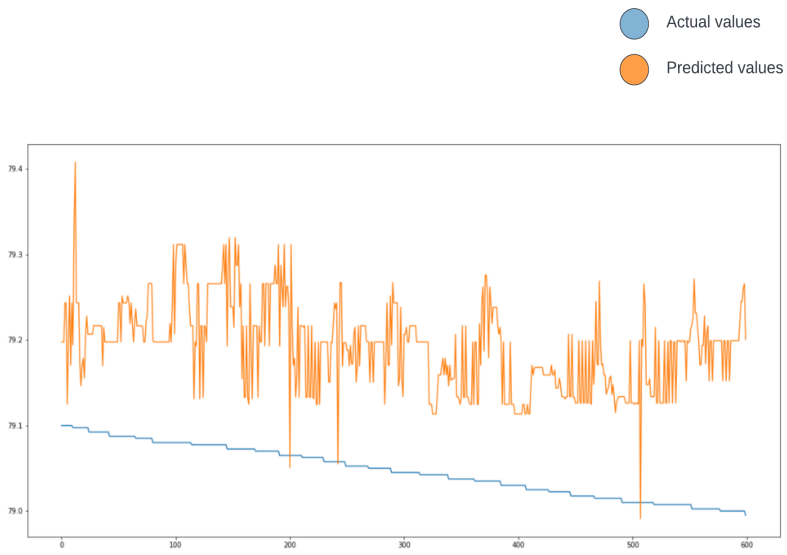

Figure 4.5.4: Original vs Predicted values of the 4th feature for setup 5 using XGBoost

Features	MAE score
1	1.68
2	0.81
3	0.69
4	0.23

Figure 4.5.5: MAE score of the features for setup 5

Up until now the model under study is based on neural networks and since we had varying predictions for certain features of the data these features had values that were closely ranged, making them approximately similar for a significant length of time. Therefore we trained the truck data on a statistical model to see how well the model is performing for such a feature that we have identified as said previously along with the features that have a diverse range of values.

For this case, we used the XGBoost statistical model as this model is best suited to train data with a time sense i.e., they are one of the best statistical models to train time series data. Here for this setup, we have chosen the four features that we have been using for the study up until now. From the figure 4.5.4 we can see the predicted values and the original values of the chosen four features. These MAE scores show that the model is predicting within a highly feasible tolerance range. But the problem with this model is that our transformer-based neural network model

used in the previous setups does representation learning in which the model learns both the time dependency of the data under study and also the inter-dependencies within the features simultaneously, whereas, in this statistical model, we do target based supervised learning that learns the time sense of the data, therefore we need to run the model for different targets for the model to learn to predict a single feature at a time which our neural network model learns how to predict all the features simultaneously.

5
Conclusion

This thesis systematically investigated the performance of a neural network model designed for analyzing battery module data from trucks. Through various simulation setups with different hyperparameters, we explored the behaviour of the model under diverse conditions. The key findings from the selected setups shed light on the intricate relationship between hyperparameters and model performance.

Simulation Setup 1 demonstrated a relatively good fit between training and validation loss, indicating a potential risk of overfitting. The increase in masking length and ratio in Simulation Setup 2 led to a wider gap between the training and validation loss curves, suggesting challenges in predicting longer sequences. Simulation Setup 3, with an increased masking length and decreased ratio, showed improved performance compared to Setup 2.

Simulation Setup 4 incorporated a higher number of features, resulting in decreased Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) scores, highlighting the potential benefit of including more input features in the model. Lastly, Simulation Setup 5 explored the use of the XGBoost statistical model, revealing its capability to predict within a feasible tolerance range but emphasizing its limitation in capturing interdependencies among features compared to the neural network model.

After seeing the results, we observed that our model still needs a lot of improvements in the accuracy of predictions. This is because the data it learned from was limited and not diverse enough. Despite this, the model has a unique strength – it learns from all the features in the dataset to make predictions. This sets it apart from traditional statistical models. Also, the differences in predicted values compared to the original ones are in the range of what the company considers acceptable. This can make us say that this model has set the right base for further developments that can yield better-performing results. Right now, our model is an experimental study showing how neural networks can predict features compared to the usual statistical methods. We've made progress, but there's more work needed to make the model ready for real-world use. This thesis gives us a good starting point and shows the potential of using neural networks for predictions.
In summary, we successfully developed and applied a Neural Network pipeline, considering various parameters to ensure versatility. However, upon closer examination, we identified certain limitations in the network's performance based on the given data. To explore alternative approaches, we conducted a comparison between our

Neural Network and a statistical model. This comparison sheds light on the distinct strengths and weaknesses of each approach. Looking ahead, the future scope of this thesis would be to train the model with large and varying data, refine the Neural Network architecture, adjust hyperparameters, and explore different masking methods that can ensure the forecasting of data values as the masking method and the Neural network models hold scope for improvements on these. This could enhance predictive accuracy and broaden the applicability of our model in future studies.

Bibliography

[1] Tianfeng Chai and R.R. Draxler. "Root mean square error (RMSE) or mean absolute error (MAE)?– Arguments against avoiding RMSE in the literature". In: *Geoscientific Model Development* 7 (June 2014), pp. 1247–1250. DOI: 10.5194/gmd-7-1247-2014.

[2] OpenGenus Foundation. *When to use Recurrent Neural Networks (RNN)?* Nov. 2018. URL: https://iq.opengenus.org/when-to-use-recurrent-neural-networks-rnn/.

[3] Raouf Ganda and Ausif Mahmood. "Deep Learning approach for sentiment analysis of short texts". In: Apr. 2017, pp. 705–710. DOI: 10.1109/ICCAR.2017.7942788.

[4] Dishashree Gupta. *Recurrent Neural Network: Fundamentals Of Deep Learning*. Nov. 2020. URL: https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/.

[5] Jj. *MAE and RMSE - Which Metric is Better?* Mar. 2016. URL: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

[6] *Mean Absolute Error - an overview | ScienceDirect Topics — sciencedirect.com*. https://www.sciencedirect.com/topics/engineering/mean-absolute-error. [Accessed 01-02-2024].

[7] *Synthetic time-series data generation using Generative Adversarial Networks — hdl.handle.net*. https://hdl.handle.net/20.500.12380/302525. [Accessed 01-02-2024].

[8] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY