Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Optimising Battery Cell Dynamics in Electric Vehicles with Embedded Machine Learning

Implementing Real-Time Voltage Modelling in Electric Vehicle Subsystems

ISAK NYBERG

Optimising Battery Cell Dynamics in Electric Vehicles with Embedded Machine Learning

Implementing Real-Time Voltage Modelling in Electric Vehicle Subsystems

ISAK NYBERG

Degree Programme in Information and Communication Technology
Date: July 29, 2024

Supervisors: Jose Berengueres, Björn Bökelund, Johan Lundström
Examiner: Mats Nordahl
School of Electrical Engineering and Computer Science
Host company: Scania CV AB
Swedish title: Optimering av battericelldynamik i elfordon med inbyggd maskininlärning
Swedish subtitle: Implementering av realtidsmodellering av batterispänning i eldrivna fordon

Abstract

This thesis explores the application of machine learning for modelling battery cell dynamics in battery electric vehicles. The primary objective is to develop and implement machine learning based models, which accurately estimate the terminal voltage of lithium-ion battery cells and are able to run inference in real-time on embedded systems present in battery electric vehicles. Conventional methods, such as the equivalent circuit model, have limitations in handling the complex and dynamic environments encountered in battery electric vehicles. This thesis aims to improve upon these methods by leveraging the capabilities of machine learning in an embedded setting.

The research was conducted in collaboration with Scania CV AB, utilising data from their battery labs and electric trucks. The study involved preprocessing and feature engineering on the data, followed by training various machine learning models, including feedforward neural networks and long short-term memory networks. These models underwent training and evaluation based on their efficacy in interpreting data derived from battery tests conducted in a laboratory setting. The trained machine learning models were then adapted to run on the embedded systems within electric trucks, while considering the limited computational power and memory resources.

Both models were evaluated in a real-world electric truck during driving, charging, and idling scenarios. The long short-term memory network exhibited better performance when driving and idling while, the feedforward neural network performed better during the charging scenario. These findings are valuable as they demonstrate that machine learning models are the feasible for real-time applications in battery electric vehicles. It also highlights a promising area of further research, particularly for battery chemistries that are not easily modelled by the equivalent circuit model, paving the way for more intelligent, safe, and efficient battery management solutions in electric vehicles.

Keywords

Battery Electric Vehicles, Embedded Machine Learning, Lithium-Ion Battery Cell Modelling, Equivalent Circuit Model, Real-Time Voltage Modelling

Sammanfattning

Denna avhandling utforskar tillämpningen av maskininlärning för modellering av battericellsdynamik i batteridrivna elfordon. Det primära målet är att utveckla och implementera maskininlärningsbaserade modeller, som uppskattar terminalspänningen i litiumjonbattericeller i realtid på inbyggda system som finns i batteridrivna fordon. Konventionella metoder, såsom ekvivalentkretsmodellen, har begränsningar när det gäller att hantera de komplexa och dynamiska miljöer som finns i batteridrivna fordon. Den här avhandlingen syftar till att förbättra dessa metoder genom att utnyttja möjligheterna med maskininlärning i en inbäddad miljö.

Forskningen genomfördes i samarbete med Scania CV AB och använde data från deras batterilaboratorier och elektriska lastbilar. Studien omfattade förbearbetning och så kallad *feature engineering* av data, följt av träning av olika maskininlärningsmodeller, inklusive feedforward neuronnät och long short-term memory modeller. Dessa modeller genomgick träning och utvärdering baserat på data från batteritester som utförts i laboratoriemiljö. De tränade maskininlärningsmodellerna anpassades sedan för att köras på de inbyggda systemen i eldrivna lastbilar, med hänsyn tagen till den begränsade beräkningskraften och minnesresurserna.

Båda modellerna utvärderades i en eldriven lastbil under körning, laddning och tomgångskörning. Long short-term memory nätverket hade en bättre prestanda vid körning och tomgångskörning, medan feedforward nätverket presterade bättre under laddningsscenariot. Dessa resultat är värdefulla eftersom de visar att maskininlärningsmodeller är användbara för realtidsapplikationer i batteridrivna elfordon. De visar också på ett lovande område för vidare forskning, särskilt för batterikemikalier som inte enkelt kan modelleras med den ekvivalenta kretsmodellen, vilket banar väg för mer intelligenta, säkra och effektiva batterihanteringslösningar i elfordon.

Nyckelord

Elektriska Fordon, Inbyggd Maskininlärning, Litiumjonbattericeller Modellering, Ekvivalenta kretsmodell, Realtidsmodellering av Spänning

Zusammenfassung

In dieser Arbeit wird die Anwendung des maschinellen Lernens zur Modellierung der Dynamik von Batteriezellen in batteriebetriebenen Elektrofahrzeugen untersucht. Das Hauptziel ist die Entwicklung und Implementierung von auf maschinellem Lernen basierenden Modellen, die die Klemmenspannung von Lithium-Ionen-Batteriezellen in Echtzeit auf eingebetteten Systemen in batteriebetriebenen Elektrofahrzeugen abbilden. Herkömmliche Methoden, wie das Ersatzschaltbildmodell, sind für die komplexen und dynamischen Vorgänge in batteriebetriebenen Fahrzeugen nur bedingt geeignet. Diese Arbeit zielt darauf ab, diese Methoden zu verbessern, indem die Möglichkeiten des maschinellen Lernens in einer eingebetteten Umgebung genutzt werden.

Die Forschung wurde in Zusammenarbeit mit Scania CV AB durchgeführt, wobei Daten aus deren Batterielabors und Elektro-LKWs verwendet wurden. Die Studie umfasste die Vorverarbeitung und das Feature-Engineering der Daten, gefolgt vom Training verschiedener maschineller Lernmodelle, einschließlich neuronaler Feedforward-Netzwerke und Long short-term Memory-Netzwerke. Diese Modelle wurden anhand ihrer Effizienz bei der Interpretation von Daten aus Batterietests in einer Laborumgebung trainiert und bewertet. Die trainierten maschinellen Lernmodelle wurden dann so angepasst, dass sie auf den eingebetteten Systemen in Elektro-LKWs laufen, wobei die begrenzte Rechenleistung und die Speicherressourcen berücksichtigt wurden.

Beide Modelle wurden in einem realen Elektro-Lkw während der Fahrt, beim Aufladen und im Leerlauf getestet. Das Long short-term Memory-Netzwerke zeigte eine bessere Leistung während der Fahrt und im Leerlauf, während das neuronale Feedforward-Netzwerk im Ladeszenario besser abschnitt. Diese Ergebnisse sind wertvoll, da sie zeigen, dass Modelle des maschinellen Lernens für Echtzeitanwendungen in batteriebetriebenen Elektrofahrzeugen geeignet sind. Darüber hinaus wird ein vielversprechender Bereich für weitere Forschungen aufgezeigt, insbesondere für Batteriechemien, die nicht ohne weiteres durch das Ersatzschaltbildmodell modelliert werden können, wodurch der Weg für intelligentere, sicherere und effizientere Batteriemanagementlösungen in Elektrofahrzeugen geebnet wird.

Schlüsselwörter

Elektrofahrzeuge, Eingebettetes maschinelles Lernen, Modellierung von Lithium-Ionen-Batteriezellen, Ersatzschaltungsmodell,

Echtzeit-Spannungsmodellierung

Acknowledgments

I would like to thank Björn Bökelund, Johan Lundström, and the rest of the EVBSH team at Scania for their help and guidance throughout the process of the thesis. I would also like to thank Jose Berengueres for his role as supervisor for this project.

Stockholm, July 2024
Isak Nyberg

Contents

1 Introduction **1**
1.1 Background . . . 1
1.1.1 Current methods . . . 2
1.2 Problem . . . 3
1.2.1 Original problem and definition . . . 3
1.3 Purpose . . . 4
1.4 Goals . . . 4
1.5 Research methodology . . . 5
1.6 Delimitations . . . 5
1.7 Structure of thesis . . . 5

2 Background **7**
2.1 Lithium-ion battery cells . . . 7
2.1.1 Cell chemistry . . . 8
2.1.2 Battery modelling . . . 9
2.1.3 Equivalent circuit model . . . 9
2.1.3.1 Internal resistance . . . 10
2.1.3.2 Transient behaviour . . . 10
2.1.3.3 Cell hysteresis . . . 11
2.1.4 Cell state estimation . . . 13
2.1.4.1 Direct estimation . . . 13
2.1.4.2 Book-keeping through Coulomb counting . 14
2.1.4.3 Adaptive book-keeping . . . 14
2.1.5 Electrochemical model . . . 16
2.2 Machine learning . . . 16
2.2.1 Feature engineering . . . 17
2.2.2 Long short-term memory models . . . 17
2.2.3 Gated recurrent unit . . . 18
2.3 Embedded systems in battery electric vehicles . . . 19

2.3.1 Functional safety 20
2.3.2 Hardware specifications 20
2.3.3 Embedded machine learning 21
2.4 Related work 21
2.5 Summary 22

3 Methods 23
3.1 Research Process 23
3.2 Data collection 25
3.2.1 Data features 25
3.2.2 Data processing 26
3.2.3 Time-delayed data 27
3.2.4 Data derivatives 28
3.2.5 Time sequences 28
3.3 Model training 28
3.3.1 Model evaluation criteria 29
3.3.2 Training procedure 30
3.4 Hardware testing 30
3.4.1 INCA testing 32
3.5 Driving test procedure 32
3.6 Assessing reliability and validity of the data collected 33
3.6.1 Validity of method 33
3.6.2 Reliability of method 34
3.6.3 Validity and reliability of data 34
3.7 Planned data analysis 35
3.7.1 Software tools 35

4 Model Development 37
4.1 Feature engineering 37
4.1.1 Feature selection 38
4.2 Model selection 41
4.2.1 Model architecture 42
4.2.1.1 Feedforward architecture 42
4.2.1.2 Recurrent neural network architecture 42
4.2.2 Effect of model size 43
4.2.3 Hyper-parameters selection 46
4.3 Model implementation 47
4.3.1 Weights conversion 47
4.3.2 FFNN implementation 49

4.3.3 LSTM implementation . . . 49

5 Results and Analysis 51
5.1 Lab data testing . . . 51
5.2 Model simulation . . . 53
5.2.1 PyTorch and C comparison . . . 55
5.3 Live model testing . . . 55
5.3.1 Test drive results . . . 56
5.3.1.1 FFNN test results . . . 56
5.3.1.2 LSTM test results . . . 59
5.3.2 Performance comparison . . . 60
5.3.3 CPU utilisation comparison . . . 62
5.4 Major results . . . 63
5.5 Discussion . . . 64
5.5.1 Test drive discussion . . . 64
5.5.2 CPU utilisation discussion . . . 64

6 Conclusions and Future work 67
6.1 Conclusions . . . 67
6.2 Limitations . . . 68
6.2.1 Data limitations . . . 68
6.2.1.1 Data temporal resolution . . . 68
6.2.1.2 Data accuracy and precision . . . 68
6.2.1.3 Data generalisation . . . 68
6.2.1.4 Model initialisation . . . 69
6.2.2 Hardware limitations . . . 69
6.2.2.1 Truck availability . . . 69
6.3 Future work . . . 69
6.3.1 More extensive testing . . . 69
6.3.2 Accounting for battery health . . . 70
6.3.3 Alternative model targets . . . 70
6.3.4 Model initialisation . . . 70
6.3.5 Alternative machine learning models . . . 70
6.4 Ethical, societal, and sustainability considerations . . . 71
6.4.1 Ethical considerations . . . 71
6.4.2 UN sustainable development goals . . . 71

References 73

A Machine learning **83**
A.1 Perceptron 83
A.2 Feedforward neural network 84
A.2.1 Back propagation 84
A.3 Recurrent neural network 85

B Detailed feature selection **86**

C Model source code **91**

List of Figures

1.1 Lithium-ion battery cell equivalent circuit model with an ideal voltage source E, internal resistance R_0, and polarisation RC unit R_p and C_p. 2

2.1 Circuit diagram of the simplest equivalent circuit model with internal resistance of a battery cell. Source: Scania 10
2.2 Circuit diagram of equivalent circuit model with internal resistance and first order RC unit. Source: Scania 11
2.3 State of charge-open circuit voltage curve of battery cell including the hysteresis effect. Source: Scania 12
2.4 Circuit diagram of equivalent circuit model incorporating all components, including the hysteresis component Hys, internal resistance R_0, and the RC components R_p and C_p. Source: Scania 13
2.5 Illustration of adaptive book-keeping with Kalman filter. The model has an internal state that is updated based on time and the measured current draw. This model produces an estimate for the terminal voltage, which is subtracted from the measured terminal voltage. The result is then multiplied by the Kalman gain vector, producing the new internal state including a new state of charge [23]. 16

2.6 Illustration of the internal workings of an LSTM unit, including the forget gate, input gate, and output gate, and the transfer functions: sigmoid (σ) and hyperbolic tangent (tanh). The LSTM unit receives three inputs: the previous cell state C_{t-1}, the current input X_t, and the previous hidden state h_{t-1}. The outputs of the LSTM are the updated cell state C_t and the new hidden state h_t, where the hidden state also serves as the output data. In cases where the output data Y_t has different dimensions than the desired output, a simple linear layer can be placed to reduce the dimension of the output data and the desired output shape [35]. 18
2.7 Illustration of the structure of a gated recurrent unit including the update gate and reset gate. The model has two inputs h_{t-1} and X_t and the two outputs h_t and Y_t 19
2.8 Photograph of a battery management unit used in Scania's electric trucks. 20

3.1 Visual illustration of research process from data collection to test drive in truck. 24
3.2 Subset of terminal voltage data over a 60 hour period. On the y-axis, the terminal voltage of the battery cell is plotted with respect to time. The hue of the line represents the SOC of the cells at the given time, with red colours being high SOC and blue being low SOC. The test consists of periods of charge and discharge of the battery with both continuous current and shorter periods of high current, with time in between for the battery cell to relax. Source: Scania 25
3.3 Illustration of dynamic voltage behaviour over a period of high current discharge, for instance a strong acceleration of a battery electric vehicle. Source: Scania 26
3.4 Setup of hardware used for testing and diagnostics. In order from left to right, controller area network connector (1), battery management unit (2), Trucks Diagnostic Tool (3), and 24V power supply (4). 31
3.5 Screen image of the INCA software with spoofed signals. Source: Scania 32

4.1 Heatmap of correlation between reduced set of features. All coefficients are multiplied by 100 and lies in the range 100-0 instead of 1-0. 38

4.2 Violin plot of the percentage improvement over the baseline when adding new a feature in isolation to a linear regression models. 40
4.3 Violin plot of each improvement for the features $i, t, soc, i_{\alpha=0.0001}$, $i_{\alpha=0.001}$ di and $dt_{\alpha=0.0001}$ over the initial baseline 41
4.4 Performance of models of difference layer size with respect to the validation loss using the MSE. The x-axis displays the size of the hidden layers in the model using a logarithmic scale, while the loss is shown on the y-axis (a lower loss indicates better performance). The dashed red horizontal line represents the performance of the linear regression model, and each line represents a model subject to changes in the size of the hidden layer. The highlighted area behind each line shows the spread of the standard deviation over the 5-fold cross-validation. The FFNN models are marked with numbers indicating the total number of layers in the model. The clear trend observed is that as the size of the models increases, the loss decreases, indicating better performance. 44
4.5 Performance of models of different number of weights with respect to the validation loss. The vertical red line represents the approximate number weights (single precision floating point numbers) that the model can use give the constraints of the battery management unit as discussed in Section 2.3.2. The horizontal line is the performance of the previously established linear regression model. 45
4.6 Random sweep over the amount of weight decay for LSTM and FFNN using the procedure described in Section 4.2.3. . . 46
4.7 Graphical illustration of LSTM C implementation. The cell and hidden state represent the internal state of the LSTM, while $VectorA$ and $VectorB$ are temporary vectors used to store intermediary values. The final hidden state is the output of the entire model. 49

5.1 Subset of model estimation on the testing data.The temperature during this test subset hovered around $30 \pm 5°C$. The cell was charged to 100% SOC and discharged to 10% in bursts of approximately 50A, repeating this cycle three times over about ten hours each. After the third cycle, the cell was fully discharged to 0% SOC, leading to a voltage collapse visible around the 31-hour mark. Source: Scania 52
5.2 Simulation of the LSTM and FFNN with real driving data, with corresponding features. About 80 minutes of real driving data is used as the input to the Scania simulation software. The first two subplots of the figure show the measured and estimated terminal voltage for the two models. Third subplot of the figure shows the inputs given to the models. The SOC started at 79% and decreased to 66% over the course of the drive. The temperature remained steady at approximately 17°C. The current drawn from the battery is shown by the blue line, while the orange and green lines represent the exponential decay for different values of alpha. Source: Scania 54
5.3 Test drive of FFNN and measured voltage for a single battery cell for each driving test. The FFNN model (dashed green line) is compared with the measured voltage (solid blue line). The first subplot shows the full test drive, with the driving, charging, and idling sections are separated by the vertical grey lines. Between the driving and charging section there is a period of time which was discounted as the truck was in the process parking and the charger was activated. The next subplot isolates the driving section, followed by a combination of the charging and idling sections. 57
5.4 Test drive of LSTM with measured voltage for a single battery cell for each driving test. The layout of the figure is the same as in Figure 5.3, with the exception that this time the green line represents the LSTM model. 59
5.5 Violin plot of mean absolute error of each battery cell for different models and test types. Each element in the figure includes a box plot of the mean and interquartile range of the data, while the shape of the violin indicates the spread of the data from the different cells. The different test types are different parts of the test drive. The combined test is using the entire test drive as a single dataset 61

A.1 Perceptron with inputs, weights, transfer function and output. . 83

B.1 Heatmap of coefficient of correlation between all features. All coefficients are multiplied by 100 and lies in the range 100-0 instead of 1-0. 87
B.2 Violin-plot of features with their corresponding improvement with base features i, t, soc and $i_{\alpha=0.0001}$ 88
B.3 Violin-plot of features with their corresponding improvement with base features $i, t, soc, i_{\alpha=0.0001}$ and $i_{\alpha=0.001}$ 89
B.4 Heat-map of features with their corresponding improvement with base features $i, t, soc, i_{\alpha=0.0001}, i_{\alpha=0.001}$ and di 90

List of Tables

2.1 Cell voltage for different cathode materials. Source Nitta *et al.,* [15] 9
2.2 Hardware limitations of the project imposed by the battery management unit. 21

5.1 Comparison of mean squared error and mean absolute error for the best FFNN and LSTM models on lab data. The unit of the mean absolute error is millivolt. The 95% confidence interval was calculated on the mean absolute error using the t-distribution. 52
5.2 Mean absolute error comparison of PyTorch and C implementations of the LSTM and FFNN models when running inference on the same data. All values are in millivolt. 55
5.3 Absolute error and 95th percentile error for the FFNN and LSTM for each test and individual cell over the driving, charging and idling period of the test-drive. The entire test-drive is shows in the combined row. All values are in millivolt. 62
5.4 CPU utilisation of 1Hz and 10Hz loops for FFNN, LSTM, and Baseline. The baseline is the utilisation of the CPU when the battery management unit is running its existing modules. The FFNN and LSTM are added in isolation in addition to the existing modules that are running on the battery management unit. The utilisation was measured as the maximum utilisation over a 5 minute period. A lower utilisation is better. 63

Listings

4.1 Code to calulate performance of different features 39
4.2 Python code that takes a pytorch model-weights and returns a string of the weights formatted as C code 48
C.1 Python code of the three layer FFNN and LSTM pytorch classes 91

List of acronyms and abbreviations

ADAM	Adaptive Moment Estimation
AMD	Advanced Micro Devices
BEV	battery electric vehicle
BMU	battery management unit
CPU	central processing unit
ECM	equivalent circuit model
FFNN	feedforward neural network
GPU	graphics processing unit
GRU	gated recurrent unit
IBM	International Business Machines Corporation
INCA	Integrated Calibration and Application Tool
ISO	International Organisation for Standardisation
LSTM	long short-term memory
MAE	mean absolute error
MDA	Measure Data Analyser
ML	machine learning
MSE	mean squared error
OCV	open circuit voltage
ONNX	Open Neural Network Exchange
RAM	random access memory
ReLU	rectified linear unit
RNN	recurrent neural network
SDG	Sustainable Development Goal
SOC	state of charge

Chapter 1

Introduction

This thesis investigates machine learning (ML) methods for modelling the voltage across the terminals of the battery cells in a battery electric vehicle. Scania, the host company, is dedicated to electrifying their trucks to support the transition to a greener transport industry. One significant challenge in this transition is the real-time modelling of the battery state. To address this challenge, this thesis will leverage battery data from Scania's testing facilities to train a neural network capable of running in real time on the embedded battery management unit within their electric trucks. The findings of this study may suggest a viable alternative to the currently used equivalent circuit model.

1.1 Background

It is estimated that 16 million electric cars are in operation today, representing 9% of annual car sales [1], resulting in billions of battery cells currently in service [2]. The Lithium-ion battery is the standard choice of battery chemistry in battery electric vehicles due to their high capacity, energy density, and long life cycle compared to other battery-chemistries. It is estimated that one-third of all batteries consist of lithium-ion cells [3]. These batteries comprise many individual battery cells connected in series and in parallel to supply sufficient voltage and current output. Due to the energy usage characteristics in battery electric vehicles, there is a need to monitor the status and health of these batteries, both from a safety perspective and from a general functionality perspective. This monitoring includes information such as state of charge (SOC), state of health, cycle count, cell temperature, cell current, terminal voltage, and various other factors. These factors can indicate whether the battery is functioning correctly, to ascertain when the battery is due for disposal, and for fault detection such as lithium dendrites* or short circuits [4].

1.1.1 Current methods

A common method for modelling the characteristics of lithium-ion cells is through the equivalent circuit model. The equivalent circuit model is an electrical circuit composed of ideal components, which when combined, closely mimic the real behaviour of a lithium-ion cell. A widely used approach involves using a combination of a voltage source, capacitors, and resistors (RC circuit) to simulate the transient voltage characteristics of the battery cell [5]. An example of a first-order model with only one RC unit is shown in Figure 1.1.

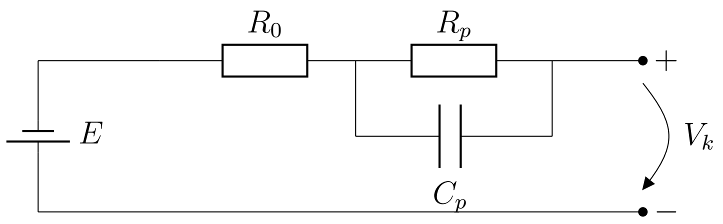


Figure 1.1: Lithium-ion battery cell equivalent circuit model with an ideal voltage source E, internal resistance R_0, and polarisation RC unit R_p and C_p.

The equivalent circuit model shown in Figure 1.1 has the advantage of being easy to create, as the values for R_0, R_p, and C_p can be parameterised. Once created, calculations are straightforward while providing good accuracy [5]. However, this idealised circuit struggles to adapt to more dynamic environments where factors such as air-conditioning, regenerative braking, and the power draw from internal components introduce noise that is too complex for the equivalent circuit model approach to model. This can be mitigated to some extent by introducing additional RC units, albeit at the expense of increased computational cost [6].

The modelling of the batteries in battery electric vehicles often occurs on embedded systems called battery management unit. These microcontrollers are responsible for the battery management and must be safety certified [7].

*Deposition of lithium to the anode that can affect cell capacity and cause short circuits

1.2 Problem

With the increasing use of lithium-ion battery cells in more complex applications such as battery electric vehicles, it becomes increasingly important to accurately manage and monitor these battery systems. Over time, as the battery is used, its total capacity decreases due to cell deterioration. Once the capacity falls below 80% of its original capacity, it is considered to have reached the end of life [8]. Using the battery beyond this point can lead to decreased system performance or even dangerous scenarios such as thermal runaway or fires [8]. Therefore, metrics such as SOC, state of health, and cell deterioration are used to ensure reliable and safe operation to fully utilise the battery until the end of its lifetime. These metrics are crucial for accurately predicting when a battery cell has reached its end of service, which is key to maintaining safe vehicle operations and calculating the total cost of operations.

Closed loop methods such as Coulomb counting* do not provide sufficient accuracy over a longer period of variation in currents as even the smallest imprecision in measurements will accumulate over time. Current state of the art proposes various solutions to this problem. One approach is to increase the order of the equivalent circuit model by adding another RC unit, which incurs a more complex calculation and parameter estimation process, or alternatively by polynomial fitting using genetic algorithms [9]. Another approach involves using a sophisticated electrochemical model of the battery cell [10]. However, due to the very limited computational resources of an embedded battery management unit used in a battery electric vehicle, these approaches are either too complex to be practical in a real-time setting or not advanced enough to provide an accurate measure of the battery cell's internal state.

1.2.1 Original problem and definition

This thesis will answer the following research question: *How can a machine learning model representing a powertrain propulsion battery cell be developed and effectively integrated into an on-board battery management unit in a battery electric vehicle?*

The research question can be split into multiple smaller questions of interest:

- What machine learning model achieve the lowest mean squared error

*Measuring the net flow of electric charge in and out of the battery over time by integrating the current with respect to time.

loss when modelling the voltage characteristics of a lithium-ion battery cell?

- How can a machine learning model be adapted to run on an electronic control unit?
- Which of the examined machine learning models is best suited for adaptation to the electronic control unit?

1.3 Purpose

The purpose of this thesis is to improve the current methods of monitoring lithium-ion battery cells in embedded systems in battery electric vehicles. From the perspective of Scania CV AB, this project aims to enhance their current methods by introducing a more accurate way of modelling their powertrain battery cells on their onboard battery management unit. Additionally, this thesis will contribute new research into both the modelling of lithium-ion batteries and the use of machine learning models on embedded platforms with limited computational resources.

1.4 Goals

The goal of this project is to create a machine learning model of a lithium-ion battery cell with accuracy equal to or superior to that of the equivalent circuit model, and then implement this model on an electronic control unit.

This has been divided into the following sub-goals:

- Conduct a study on the current research into monitoring methods for lithium-ion battery cells.
- Identify a machine learning model that can accurately simulate and predict the behaviour and characteristics of lithium-ion battery cells.
- Develop a methodology or framework for adapting machine learning models to be efficiently executed on a battery management unit with limited computational resources.
- Evaluate the performance, efficiency, and suitability of various machine learning models to determine the most appropriate model for implementation in a battery management unit for real-world applications.

1.5 Research methodology

Parts of this thesis will contain a literature study on the current academic work within the topics of lithium-ion battery modelling and machine learning for embedded systems. This will include research that addresses the use of machine learning for lithium-ion battery modelling, as well as research into applied machine learning in battery electric vehicles. These resources will be sourced from academic journals, as well as internal resources from Scania.

After the literature study, a machine learning model will be developed using insights gained from the literature review. The model will utilise modern machine learning frameworks such as PyTorch [11]. It will be trained with data provided by Scania and benchmarked against the test data as well as a parameterised equivalent circuit model.

The model will then be adapted to meet the constraints of the battery management unit and subsequently benchmarked to determine the impact of the adaptation process from its original framework to the framework on the battery management unit. Different methods of creating this implementation will be explored, including the use of the intermediate ONNX representation [12]. The final model will then be tested in a live setting by flashing the implementation onto a truck and carrying out a test drive.

1.6 Delimitations

While the model itself needs to be able to run on the embedded battery management unit with reasonable resource utilisation, the training process does not have to run on the battery management unit. This thesis will solely focus on the terminal voltage estimation of the battery cell, and not on the methods of using it to estimate the SOC.

1.7 Structure of thesis

Chapter 2 presents relevant background information about lithium-ion battery cells, ML, and the battery management unit. Chapter 3 outlines the methodology and methods used to train and evaluate the model and its results. Chapter 4 describes the development progress of the model, explaining the design decisions made throughout the process. Chapter 5 presents the results of the model, including both simulations and a test drive. Finally, Chapter 6 highlights the key takeaways, limitations, and further research that can be carried out.

Chapter 2

Background

The background chapter provides the information necessary for understanding of this thesis, covering topics such as lithium-ion battery cells, their modelling, particularly the equivalent circuit model, and methods for estimating the cells' state of charge (SOC). This chapter examines the current methods used to model a lithium-ion battery, including machine learning approaches and the electrochemical model. The final section offers a brief introduction to the battery management unit, discussing the constraints of the hardware.

2.1 Lithium-ion battery cells

The idea of lithium-ion battery cells originated from the Co Corporate Labs of Exxon in 1972, and the technology has come a long way since entering the consumer market in 1991 [13]. By 2015, lithium-ion battery cells accounted for more than 85% of all energy storage systems in use [14]. Lithium-ion cells consist of two terminals separated by an electrolyte through which lithium ions can move freely. The anode is the negatively charged terminal, while the cathode is the positively charged terminal. When the battery is discharging, the anode releases lithium ions that move through the electrolyte to the cathode [15]. Going forward the following definitions are used:

Capacity (Q) is the measure of the amount of charge in a battery, measured in Coulomb. When referring to the capacity of a battery it means the capacity when fully charged.* It is defined by Equation 2.1 where I is a positive current into the battery.

$$Q = \int I \, dt \tag{2.1}$$

*A battery is fully charged when it no longer accepts positive current at it's rated voltage.

For a battery cell the state of charge is defined as the percentage of the battery's current capacity relative to its capacity when full charged. This is equivalent to the fuel gauge of a car with an internal combustion engine. This is a value between 0 and 100 defined by Equation 2.2:

$$SOC = 100 \cdot \frac{Q_t}{Q} \tag{2.2}$$

When a battery ages it loses some of its ability to retain charge the capacity of the battery decreases. The ratio of a battery's capacity to its nominal capacity is known as the state of health expressed as a percentage, and is defined by Equation 2.3.

$$SOH = 100 \cdot \frac{Q}{Q_{max}} \tag{2.3}$$

2.1.1 Cell chemistry

In the battery, two equilibirum reactions take place: one between the cathode and the electrolyte, and one between the anode and the electrolyte. The cathode is made from metal oxides or phosphates, while the anode is typically made from carbon or silicon-based materials. The two reactions are as follows [16].

$$\mathrm{LiMO_2} \underset{\text{Discharge}}{\overset{\text{Charge}}{\rightleftharpoons}} \mathrm{MO_2} + \mathrm{Li^+} + \mathrm{e^-} \tag{2.4}$$

$$\mathrm{C_6} + \mathrm{Li^+} + \mathrm{e^-} \underset{\text{Discharge}}{\overset{\text{Charge}}{\rightleftharpoons}} \mathrm{LiC_6} \tag{2.5}$$

Equation 2.4 represents the reaction that takes place between the cathode and the electrolyte, while Equation 2.5 represents the reaction between the anode and the electrolyte. These are equilibrium reactions, meaning they can occur in either direction depending on the concentration of the reagents, temperature, and other factors. The reaction in the left-to-right direction indicates the battery charging, while the reaction in the right-to-left direction indicates the battery discharging. The element M in this context represents a transition metal, typically Cobalt, Manganese, or Nickel, or a combination of these [15]. When discharging, the electrons released in Equation 2.5 and consumed in Equation 2.4 cannot enter the electrolyte; they remain on the anode terminal.

Placing a conductive circuit between the anode and the cathode allows these electrons to flow, thereby causing the battery cell to discharge. This flow generates an electric current and maintains the necessary balance of charge for the reactions to proceed. In the absence of a closed circuit, the electrochemical reaction will stop. Applying an opposite voltage across the terminals will reverse the reaction and cause the battery cell to charge [17].

The cell voltage depends on the combination of cathode and anode materials [15]. A table of materials commonly used in battery electric vehicles with their corresponding voltages can be seen in Table 2.1 [15].

Cathode material	**Average cell potential**	
$LiNi_{0.33}Mn_{0.33}Co_{0.33}O_2$	3.7	V
$LiMn_2O_4$	4.1	V
$LiFePO_4$	3.4	V

Table 2.1: Cell voltage for different cathode materials. Source Nitta *et al.* [15]

Beyond the values visible in Table 2.1 [15], the voltages can vary depending on the chemical structure of the material and the different ratios of each material in the cathode [15].

2.1.2 Battery modelling

The simplest model of a battery cell is as an ideal voltage source, however under dynamic conditions the electrical behaviour of a battery cell does not match that of an ideal voltage source. Instead, more advanced models are used to capture these characteristics. Commonly used lithium-ion battery cell models include the equivalent circuit model explained in Section 2.1.3, electrochemical models explained in Section 2.1.5, and machine learning-based models explained in Section 2.2.

2.1.3 Equivalent circuit model

The battery cell generates both current and potential difference between its terminals due to the chemical reaction within. However, this reaction has properties that prevent the battery cell from behaving as an ideal voltage source. Consequently, a common approach to modelling a lithium-ion battery in a circuit is the equivalent circuit model. The equivalent circuit model represents the battery cell as an electronic component that includes an ideal voltage source to provide the cell's voltage, along with additional circuit components that simulate the behaviour of the voltage and current produced by the chemical reaction within the cell [18, 19].

2.1.3.1 Internal resistance

The chemical reactions occurring within a battery are not perfectly efficient and result in energy losses, predominantly in the form of heat. In the equivalent circuit model, this phenomenon is represented by incorporating an internal resistance for the battery cells [9].

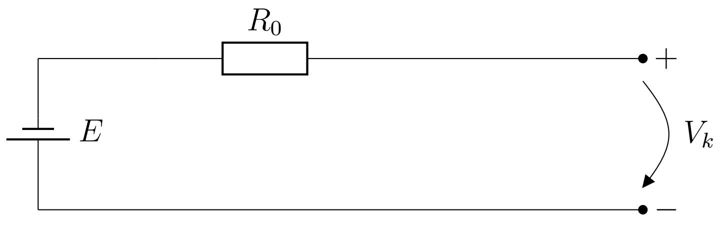


Figure 2.1: Circuit diagram of the simplest equivalent circuit model with internal resistance of a battery cell. Source: Scania

Figure 2.1 illustrates the internal resistance as resistor R_0. This modelling is particularly suitable because the rate of chemical reactions within the battery is directly proportional to the output current. Correspondingly, the energy dissipated across R_0 is also proportional to the current flowing through this resistor, thereby providing an accurate representation of the internal resistance's effect on battery performance [20].

2.1.3.2 Transient behaviour

When there is a sudden spike in current draw from a battery cell, the transfer of lithium ions between the electrodes exhibits some inertia. This results in transient behaviour that, in the equivalent circuit model model, is represented by RC units [21].

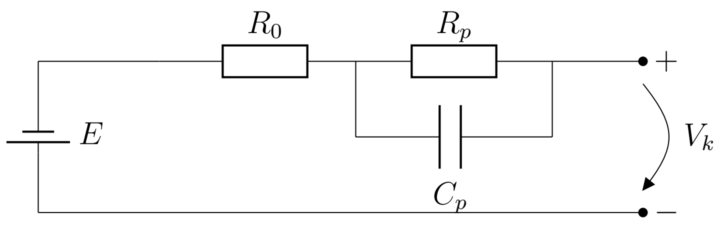


Figure 2.2: Circuit diagram of equivalent circuit model with internal resistance and first order RC unit. Source: Scania

As shown in Figure 2.2, the new circuit is constructed with the previous internal resistance R_0 as well as an RC circuit consisting of the resistor R_p and the capacitor C_p. These are known as the polarisation internal resistance and polarisation internal capacitance, respectively [9].

Estimating the parameters for these components allows the model to mimic the battery cell characteristics. The same model can be further improved by including multiple RC circuits, albeit at the cost of increased model complexity [22].

The resistance of the resistor and the characteristics of the RC circuit are dependent on the state of charge (SOC) of the cell as well as the operating temperature (T) of the entire circuit. This ultimately means that the cell terminal voltage V_k is a function of I, T and SOC as shown in Equation 2.6.

$$V_k = f(I, T, SOC) \tag{2.6}$$

2.1.3.3 Cell hysteresis

Lithium-ion cells exhibit hysteresis, this is observed when a cell is disconnected after having charged for a period of time, measured voltage will be higher than expected and it will take some time before it reach its relaxed open circuit voltage. The same is true after the cell has been discharging; the measured cell voltage will be lower than estimated [1]. This is illustrated in Figure 2.3.

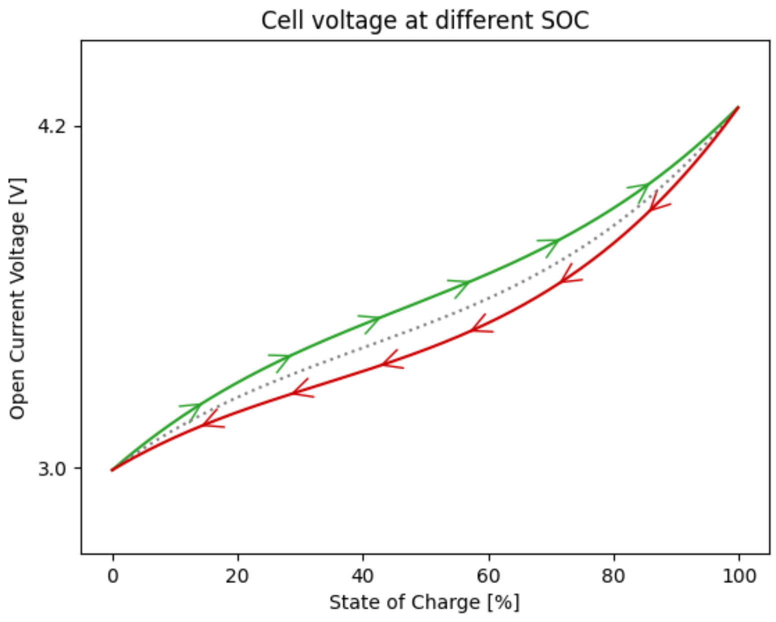


Figure 2.3: State of charge-open circuit voltage curve of battery cell including the hysteresis effect. Source: Scania

In Figure 2.3, the SOC of the battery cell is plotted against the open circuit voltage of the battery cell. When the cell has been charging (going from low SOC to high SOC, represented by the green line), the measured terminal voltage is slightly higher than the expected open circuit voltage (represented by the grey dotted line). Likewise, when the cell has been discharging (going from high SOC to low SOC), the measured voltage is lower than expected. This phenomenon is called hysteresis and cannot be explained by a simple equivalent circuit model. As a result, a new hysteresis component is added to the circuit to account for this difference. This final model is displayed in Figure 2.4.

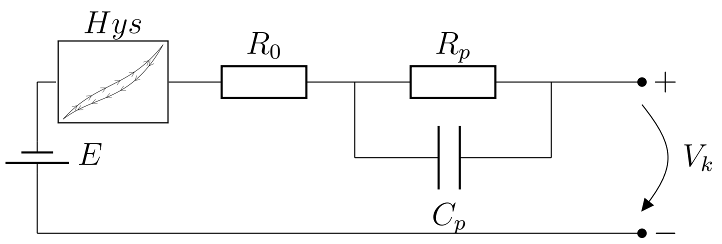


Figure 2.4: Circuit diagram of equivalent circuit model incorporating all components, including the hysteresis component Hys, internal resistance R_0, and the RC components R_p and C_p. Source: Scania

2.1.4 Cell state estimation

Accurately determining the SOC is vital to prevent overcharging and excessive discharging of the cell. Additionally, it helps in determining the state of health of the cell, assessing the remaining usage cycles, and diagnosing cell longevity, performance, and degradation [23]. It is important to distinguish the nominal capacity from the original capacity of the cell, as the cell's maximum capacity decreases with time and usage. Unlike temperature and current, the SOC cannot be directly measured and must instead be derived from historical data. There are several methods for deriving this value, each offering different benefits for various use cases [24].

2.1.4.1 Direct estimation

Methods that directly estimate the SOC take a known measurable value of the battery cell and derive the SOC through specific equations. The SOC can be estimated from several factors, such as open circuit voltage, current, terminal voltage, and cell impedance spectroscopy [23].

The open circuit voltage method uses the terminal voltage when the circuit is open, and there is no current flowing into or out of the cell. This value can be compared against an state of charge-open circuit voltage curve like the one seen in Figure 2.3 and illustrated by Equation 2.7.

$$SOC(t) = OCV(V_t) \tag{2.7}$$

However, this method does not account for the hysteresis effect discussed in Section 2.1.3.3. Additionally, to measure the open circuit voltage, the circuit must be *open*, which is not feasible in use cases where the battery cell is continuously under load, such as in a battery electric vehicle. Making an accurate terminal voltage measurement of a cell is further complicated by the fluctuations in voltage when the cell is in use [23].

An alternative approach is impedance spectroscopy, which involves measuring the battery impedances when exposed to a wide range of different alternating current frequencies and then applying regression techniques to fit the different impedances, from which the SOC can be inferred [23]. This method is feasible in a laboratory environment but is too advanced to implement in the context of a battery electric vehicle and will not be considered in this thesis.

2.1.4.2 Book-keeping through Coulomb counting

Book-keeping methods make use of historical data to account for the exact amount of charge that has left or entered the cell. The most common method is *Coulomb Counting*, which involves measuring the current that enters or leaves the battery cell and then integrating this value with respect to time to determine the amount of charge removed from the cell. Because the maximum capacity of the cell is known, the SOC can be calculated using Equation 2.8 [25].

$$SOC_t = SOC_{t_0} - \int_{t_0}^{t} \frac{I(t)}{Q} dt \tag{2.8}$$

In Equation 2.8, the state of charge at time t, (SOC_t) is calculated from the initial state of charge (SOC_{t_0}) and the integral of the current in/out of the cell I_t divided by the capacity (Q) of the cell [24]. This approach is feasible as long as the currents can be accurately measured and SOC_{t_0} can be precisely determined. However, any systematic error in the measurement of the current will accumulate over time and skew the SOC [23]. In the context of a battery electric vehicle, the main drawback of this approach is that it is an open-loop estimation process, where small deviations in measurements add up over time, causing the resulting SOC estimate to lose accuracy [24].

2.1.4.3 Adaptive book-keeping

The best approach combines book-keeping and adjusts it according to directly measured values. One such method is the use of Kalman filters [23]. A Kalman filter is a type of recursive filter that utilises measured current, voltage, and temperature to update and adjust the SOC. This approach uses the internal state from the equivalent circuit model (discussed in Section 2.1.3), updates it based on the measured current, and then refines this result using the measured voltage. The equation for the update of the internal state is shown in Equation 2.9 and Equation 2.10

$$\begin{bmatrix} \mathrm{SOC}_t \\ U_t \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-\frac{t_s}{\tau}} \end{bmatrix} \times \begin{bmatrix} \mathrm{SOC}_{t-1} \\ U_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{-t_s}{Q_n} \\ R_p\left(1 - e^{-\frac{t_s}{\tau}}\right) \end{bmatrix} \times i_{t-1} \tag{2.9}$$

$$U_{e,t} = f(SOC_t) - U_t - R_0 \times i_{t-1} \tag{2.10}$$

In Equation 2.9, the internal state is a linear function of the previous state plus a linear function of the current. Equation 2.10 estimates the terminal voltage based on the state of charge-open circuit voltage and subtracts the voltage across the RC unit and the voltage over the internal resistance [26]. This is the equivalent circuit model voltage over the terminals. The estimate of $U_{e,t}$ can then be compared to the measured terminal voltage U_m and used to adapt the internal state of the battery cell [23]. If the equivalent circuit model system has more than one RC unit, the internal state vector and matrix can increase their dimensions to accommodate this [27].

$$\begin{bmatrix} \mathrm{SOC}_{t+1} \\ U_{t+1} \end{bmatrix} = \begin{bmatrix} \mathrm{SOC}_t \\ U_t \end{bmatrix} + \begin{bmatrix} K_{\mathrm{SOC}} \\ K_U \end{bmatrix} (U_{m,t} - U_{e,t}) \tag{2.11}$$

Equation 2.11 is the final step of the process, where the internal state is updated by taking the internal state of the model and tuning it according to the difference between the measured terminal voltage and the estimated voltage, multiplied by a vector called the *Kalman gain* [27]. The Kalman gain is parameterised based on the properties of the cell and its internal state.

The adaptive book-keeping method is a closed-loop system that can adjust itself if the model representing the cell goes out of sync with the measured values. Note that in Equation 2.11, if $U_{m,t}$ and $U_{e,t}$ are equal, meaning that the model is perfectly accurate, their difference is zero, resulting in no alteration of the internal battery state in that step. This entire process is illustrated in Figure 2.5.

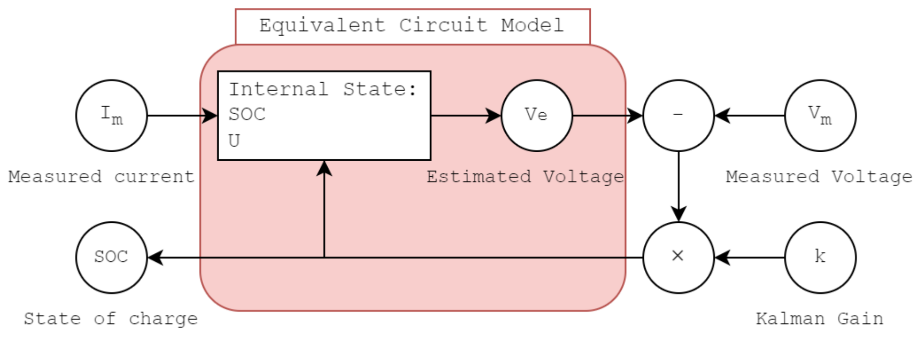


Figure 2.5: Illustration of adaptive book-keeping with Kalman filter. The model has an internal state that is updated based on time and the measured current draw. This model produces an estimate for the terminal voltage, which is subtracted from the measured terminal voltage. The result is then multiplied by the Kalman gain vector, producing the new internal state including a new state of charge [23].

2.1.5 Electrochemical model

The electrochemical model was originally proposed by Doyle *et al.,* in 1993 [10]. This approach addresses the problem from a chemical perspective by examining the reaction rates of the chemical processes inside the battery cell. It offers superior modelling and accuracy compared to the equivalent circuit model in terms of the behavioural dynamics of a lithium-ion battery cell, albeit at the expense of increased complexity and computational demand [28]. The model itself is exceedingly non-linear and depends on a multitude of parameters, making it infeasible for real-time estimations, especially on limited hardware [29].

2.2 Machine learning

Machine learning is the process of providing a program with a set of experiences related to a specific task, where its performance can be measured. If the program improves at the task using these experiences, it is said to *learn* [30]. In the context of this thesis, the experiences will consist of historical battery data containing the battery SOC, temperature, current, and terminal voltage. The performance measure will be the program's ability to estimate the terminal voltage given the SOC, temperature, and current. This section presupposes a basic understanding of machine learning concepts. A more in-depth introduction to these general concepts can be found in Appendix A. The paper will utilise three different machine learning models: long short-term memory, explained in Section 2.2.2; gated recurrent unit, explained in Section 2.2.3; and feedforward neural network, explained in Section A.2.

2.2.1 Feature engineering

Feature engineering is part of the preprocessing of the training data used for machine learning. It involves multiple approaches such as creating new domain specific features, transforming current features into a more suitable representation, extracting vital information from an existing feature, or discarding existing feature that are not believed to be useful. Severson *et al.,* have successfully employed feature engineering in life cycle prediction of lithium-ion batteries [31]. The same study was expanded upon by Geslin *et al.,* in 2023 which suggested that using the most suited features for the task is essential to create data-driven models that are suitable for real use cases [32].

2.2.2 Long short-term memory models

One example of a recurrent neural network (RNN) is the long short-term memory (LSTM), which effectively balances learning long and short-term dependencies in sequential data. LSTMs were first introduced by Hochreiter and Schmidhuber in 1997 [33]. The LSTM unit consists of three different gates: the input, forget, and output gates, each serving a specific purpose in retaining and forgetting data [34]. A diagram of an LSTM is shown in Figure 2.6.

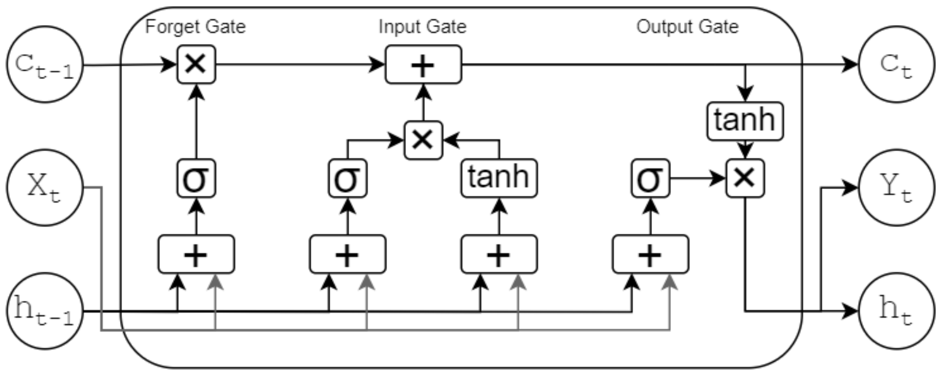


Figure 2.6: Illustration of the internal workings of an LSTM unit, including the forget gate, input gate, and output gate, and the transfer functions: sigmoid (σ) and hyperbolic tangent (tanh). The LSTM unit receives three inputs: the previous cell state C_{t-1}, the current input X_t, and the previous hidden state h_{t-1}. The outputs of the LSTM are the updated cell state C_t and the new hidden state h_t, where the hidden state also serves as the output data. In cases where the output data Y_t has different dimensions than the desired output, a simple linear layer can be placed to reduce the dimension of the output data and the desired output shape [35].

The LSTM shown in Figure 2.6 consists of gates with different functions. The first gate is the **forget gate**, which controls the amount of the previous cell state that is retained. Note that the sigmoid function returns a value between 0 and 1, and multiplying the cell value by this sigmoid output will result in a fraction of the original cell value. The second gate in the process is the **input gate**, which combines the hidden state with the new data and uses it to update the cell state through addition. Lastly, the **output gate** uses the product of the cell state and the input data to produce the new hidden state, which is also the output of the LSTM unit. The updated cell state and hidden state are then passed as inputs for the next input in the time series [35].

2.2.3 Gated recurrent unit

The gated recurrent unit (GRU) is an RNN similar to the LSTM. It was first introduced by Cho *et al.* [36] and was originally used in the context of natural language processing. Like the LSTM, it adapts to time series data and has an internal state. It also follows a similar architecture of having gated units that modulate the information inside the unit. The GRU has a simpler structure compared to the LSTM [37]. A diagram of a GRU is shown in Figure 2.7.

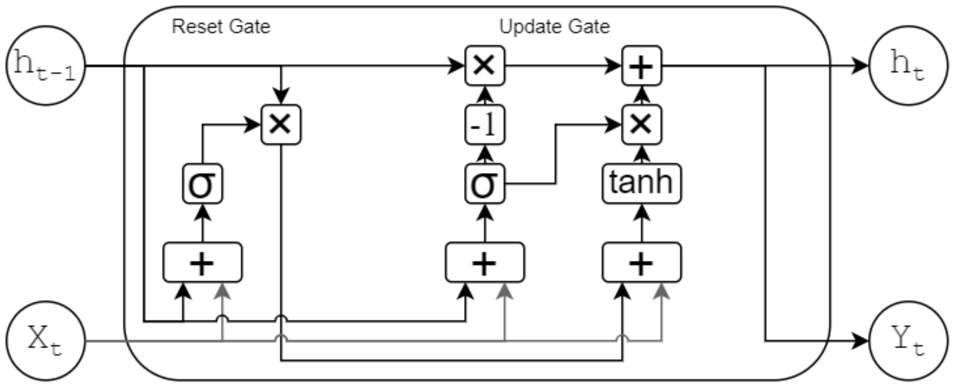


Figure 2.7: Illustration of the structure of a gated recurrent unit including the update gate and reset gate. The model has two inputs h_{t-1} and X_t and the two outputs h_t and Y_t

Figure 2.7 shows the structure of the GRU. Unlike the LSTM, the GRU only has two gates: the update gate and the reset gate. The update gate balances how much of the input will influence the new state of the GRU as well as what the unit will output, while the reset gate determines how much of the previous hidden state h_{t-1} is retained. The GRU is smaller and faster to train than the LSTM [38].

2.3 Embedded systems in battery electric vehicles

Embedded systems are minimal systems consisting of at least a processor, memory, and some kind of input/output. They are usually part of a larger system, which may consist of multiple embedded units [39]. Modern vehicles heavily rely on onboard embedded systems. These systems can have various tasks in managing different parts of the vehicle; some carry out safety-critical systems such as anti-lock braking system, while others handle more mundane tasks such as climate control [40]. This thesis will specifically focus on the battery management system in Scania's battery electric trucks.

2.3.1 Functional safety

When it comes to heavy machinery and vehicles such as trucks and buses, many systems are labelled as *safety-critical* [41]. This means that the reliability of the hardware and software must conform to established safety standards such as ISO 26262 [7]. Despite various suggestions, ISO 26262 does not yet cover any specifications for machine learning regarding data quality, model robustness, and model fault detection [42, 43].

2.3.2 Hardware specifications

The battery management unit is the particular embedded system that this thesis will focus on. The battery management unit used in this project belongs to the *MPC5 ultra-reliable MCU* family [44]. The micro-controller itself has a 32-bit central processing unit (CPU) with support for single-precision floating-point operations, 16KB data cache (D-Cache), and 512KB random access memory (RAM). Since the battery management unit has multiple use cases, the specific task of executing the trained machine learning model will be allocated approximately 1% of the memory when the task is not executing. As a result, the entire state of the model may not exceed 5120 bytes, or approximately 1280 single-precision floating-point numbers.

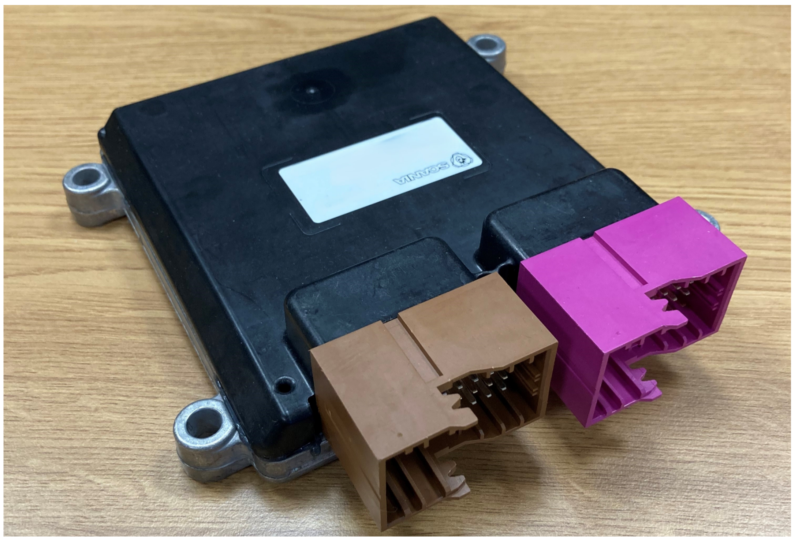


Figure 2.8: Photograph of a battery management unit used in Scania's electric trucks.

CPU	32 Bit
Operating Frequency	200-300 MHz
RAM	512kB
Flash	8000kB
Functional Safety	ISO 26262
Maximum Model size	5120 Bytes

Table 2.2: Hardware limitations of the project imposed by the battery management unit.

2.3.3 Embedded machine learning

From a machine learning perspective, the restrictions of ISO 26262 for embedded hardware can be detrimental because many machine learning algorithms achieve efficiency through parallel computation, particularly on hardware such as graphics processing unit (GPU) [45]. Embedded machine learning aims to tackle the challenge of running machine learning models on constrained systems. In 2017, Meta Platforms Inc., together with Microsoft Corp., released the Open Neural Network Exchange (ONNX) format [12]. The project has since gained widespread adoption from other large technology companies such as Advanced Micro Devices (AMD), International Business Machines Corporation (IBM), and Huawei [46]. ONNX allows neural networks to be formatted in a way that is widely supported by different frameworks, tools, and platforms [12]. ONNX has a runtime for C [47], which is a common language of choice for embedded systems. Additionally, there are open-source projects such as ONNX2C that can compile the ONNX model into C code [48]. However, this approach is not specifically tailored for low memory usage, and the extent to which these approaches can be adapted for the battery management unit will be an area of exploration in this thesis.

2.4 Related work

In recent years, advancements in machine learning have led to new research using machine learning approaches to model lithium-ion battery cells. This includes using feed-forward neural networks, convolutional neural networks, and long short-term memory networks to estimate cell state of health using voltage, current, and temperature profiles. These techniques have been shown to outperform conventional methods that use only voltage profiles by up to

25%-58% [49]. It has also been shown that, given sufficient training data and knowledge of a system, a machine learning model can accurately adapt and model physical systems [50], either for the purpose of improving existing models or to create entirely new models [51]. To model the characteristics of lithium-ion battery cells using ML, current research has employed the Python library Scikit-learn to predict voltage behaviour [52] and support vector machines for state of health estimation [53]. These results suggest that ML is a feasible approach, in addition to the equivalent circuit model, for accurately modelling battery cell characteristics. Similarly, terminal voltage, current, impedance, and capacity have been used to estimate the remaining useful life using LSTM models [54]. The use of support vector machines has been explored as an alternative to the Kalman filter, as shown in Section 2.1.4.3, to improve performance [55]. Other researchers have used combinations of neural networks and *teaching learning-based optimisation* to predict both open circuit voltage and terminal voltage, which have shown to have high accuracy [56]. Other researchers have used machine learning techniques to model the internal SOC of battery cells [57]. Additionally, other forms of machine learning have been employed for battery failure detection by monitoring impending voltage collapse in lithium-ion batteries [58].

2.5 Summary

Lithium-ion battery cells can store chemical potential energy, which is converted to electrical energy when a circuit is formed between the cell's terminals. The electrical behaviour of lithium-ion cells can be modelled using the equivalent circuit model. The behaviour of the battery cell is used to determine important properties such as the cell's state of charge. This approach can potentially be replaced with a machine learning (ML) model, such as a feedforward neural network (FFNN), a long short-term memory (LSTM), or a gated recurrent unit (GRU). This model would need to be capable of running on the battery management unit inside the battery electric vehicle.

Chapter 3

Methods

This chapter outlines the methodologies used in this thesis, covering the research process, data collection, experimental design, and evaluation process. It also highlights the scientific and engineering skills applied in this project. The research process described in Section 3.1 details the steps taken: data collection, preprocessing, feature engineering, model training, selection, hardware implementation, and real-world testing. The data collection described in Section 3.2 discusses techniques and ethical considerations, describing the data features collected. In data processing described in Section 3.2.2, the methods for preprocessing and feature extraction are explained, focusing on dynamic and static battery voltage components. The experimental design described in Section 3.3 outlines the construction and comparison of various machine learning models based on performance metrics like mean squared error (MSE), robustness, memory efficiency, and CPU usage. The reliability and validity is assessed in Section 3.6, discusses ensuring data and method accuracy and consistency, addressing potential generalisation issues. The planned data analysis in Section 3.7 explains the evaluation of model performance using mean absolute error (MAE) and details the software tools used.

3.1 Research Process

Figure 3.1 shows the steps conducted to carry out this research.

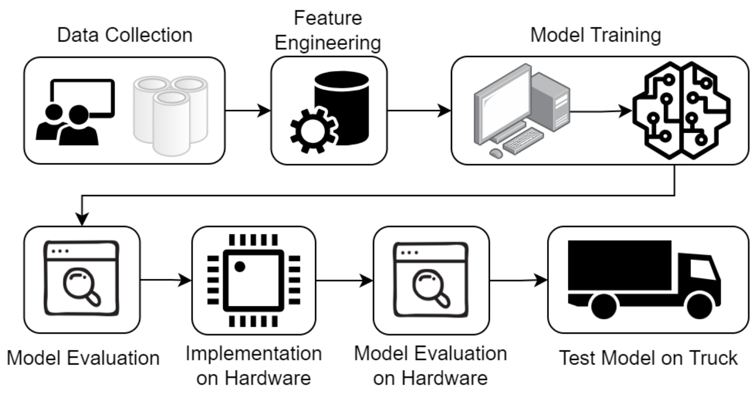


Figure 3.1: Visual illustration of research process from data collection to test drive in truck.

Figure 3.1 shows the broad steps taken in this thesis. It begins with the data collection carried out by Scania. This data is time series data of electric trucks including battery temperature, voltages, current, and state of charge over a period of the truck being used. Feature engineering will then be carried out on the data in order to extract relevant information and expand the input space. Thereafter an assortment of machine learning models will be trained on a subset of this data. Another subset of the collected data will be used as testing data for model selection based on performance. A few of the trained models will be selected to be implemented on the truck hardware, to ensure they comply with the computational restraints in both speed and memory usage. After this implementation it will be compared to the original model to determine the performance degradation of the adaptation process. This final model will then also be tested on a truck in a live usage setting.

These are the steps of the project:

Step 1 Collect truck data

Step 2 Pre-process data and divide into training, evaluation and test splits

Step 3 Train each proposed model with the training data

Step 4 Fine-tune model using evaluation data

Step 5 Select a model based on test data performance

Step 6 Implement trained model on hardware

Step 7 Evaluate performance loss in the implemented model

Step 8 Test model implementation on a real truck

3.2 Data collection

The data used to train the models comes from the battery lab facility at Scania. This data is very precise because it is collected in a controlled environment and consists of time series of cell charging and discharging at different temperatures, sampled once per second. Each sample contains the state of charge, current in or out of the cell, temperature of the cell, and the terminal voltage of the cell. There is a total of 18 continuous run where each run comprises around 100 hours of data. For each run the cells in the test are acclimated to a specific temperature of a minimum $-20°C$ and maximum $+45°C$ with variations of $\pm10°C$ within the test. From these series, 8 were from one cell, 4 from another cell, 2 from a third and then the last 4 series were from unique cells.

An extract of these series can be seen in Figure 3.2.

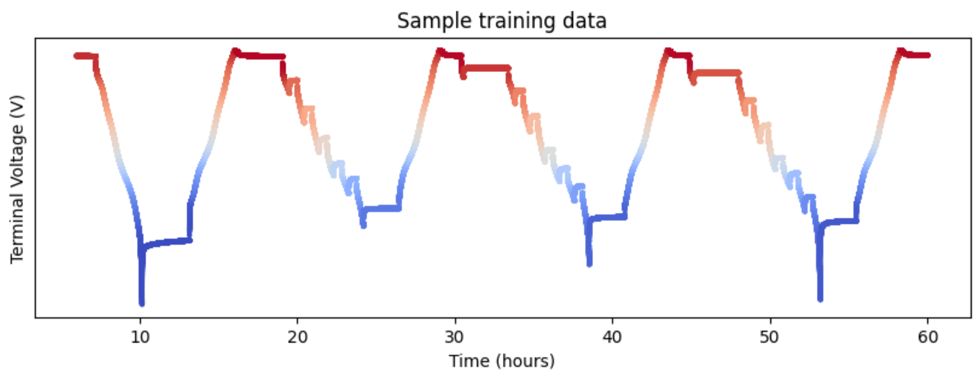


Figure 3.2: Subset of terminal voltage data over a 60 hour period. On the y-axis, the terminal voltage of the battery cell is plotted with respect to time. The hue of the line represents the SOC of the cells at the given time, with red colours being high SOC and blue being low SOC. The test consists of periods of charge and discharge of the battery with both continuous current and shorter periods of high current, with time in between for the battery cell to relax. Source: Scania

3.2.1 Data features

Cell Current: The cell current was measured with a unit placed between the cell and the load. It measured between $-150A$ and $+150A$ depending on whether the cell was charging or discharging. Sometimes the cell was given bursts of high current, while other times it was exposed to longer periods of constant current, as well as sinusoidal patterns to mimic different load behaviours of a real-world battery-driven truck.

Temperature: The temperature of the environment was moderated for different runs to examine the effects of various environmental temperatures. It was measured by a temperature probe placed in close proximity to the battery cells. One limitation is that the temperature sensor is not placed inside the cell, and the temperature of the electrolyte may impact the accuracy of the temperature value.

State of Charge: Since the battery state of charge is not directly measurable, it was derived through a process called Coulomb counting. The SOC is part of the internal state of the cell.

Terminal Voltage: The terminal voltage was simply measured across the terminals with a voltmeter.

3.2.2 Data processing

As mentioned in Chapter 2, a battery cell has both dynamic and static voltage behaviour. The static voltage can almost entirely be derived from the SOC, while the dynamic voltage is more difficult to model. The sum of these two components is the actual terminal voltage. This behaviour is illustrated in Figure 3.3.

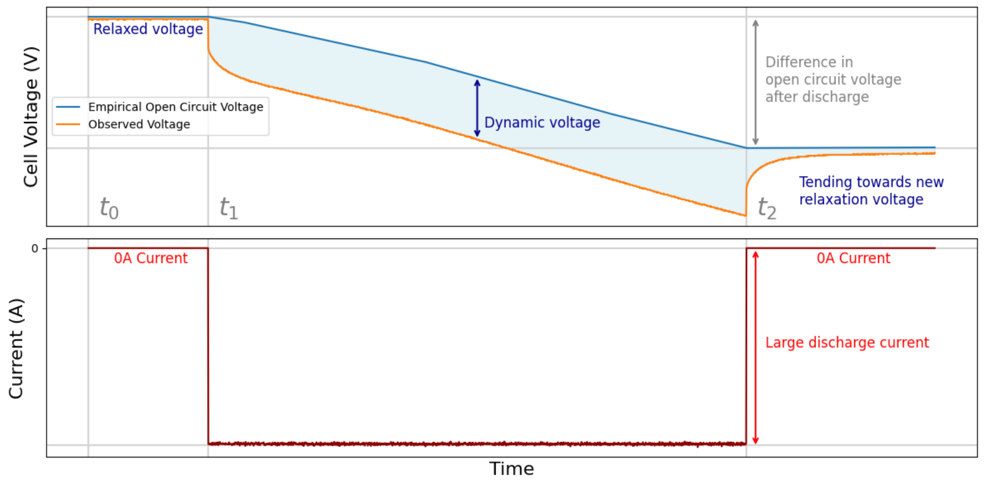


Figure 3.3: Illustration of dynamic voltage behaviour over a period of high current discharge, for instance a strong acceleration of a battery electric vehicle. Source: Scania

Figure 3.3 illustrates an example of dynamic voltage behaviour over a large current discharge. The blue line represents the open circuit voltage over the period of the discharge, which is an empirical value based on the SOC. The orange line represents the terminal voltage observed across the terminals. The bottom subplot of Figure 3.3 shows the current going out of the battery cell.

Initially, at time t_0, the open circuit voltage and measured voltage are identical as the current is $0A$, indicating that the cell is in its *relaxed* state. At time t_1, a large discharge occurs, causing the measured terminal voltage to drop significantly below the open circuit voltage. The difference between the open circuit voltage and the measured terminal voltage is the *dynamic voltage*. The dynamic voltage typically has a magnitude of hundreds of millivolts.

At time t_2, the discharge current stops and the current returns to 0A. At this point, the observed voltage will slowly tend towards the open circuit voltage. The open circuit voltage will also be lower than before since the SOC of the battery has decreased after the discharge.

This means the models task can be divided into two parts, the first is to estimate the static voltage and the second more difficult problem is to estimate the dynamic voltage. If the model can be trained on these two task separately it should have an easier time adapting to the data, as a result a new target can be created called DYNE which is the dynamic voltage calculated by subtracting the static voltage from the terminal voltage $DYNE = V - OCV$, this is the value the model should estimate.

3.2.3 Time-delayed data

As part of feature engineering, new features can be introduced using the existing features. One example of such a feature is the time-delayed current, where an *exponential moving window* can be used. This involves creating a new feature updated for each sample using the formula:

$$y_0 = x_0 \tag{3.1}$$

$$y_t = (1 - \alpha)y_{t-1} + \alpha x_t \tag{3.2}$$

In Equation 3.1 the initial value of y_0 is set to the first value of x. For every subsequent time step, the value of y_t is updated using the value of x_t and the previous y_{t-1}, using the coefficients based on the value $\alpha \in (0, 1)$. The value of α can be adjusted to determine the amount of change in y for each t. In the Python library `Pandas`, this can be implemented with the `ewm` function [59]. The values for α used in this thesis are (0.0001, 0.001, 0.01, and 0.1). This allows the model to use historical state without having an internal memory when making estimations. The size of the input vector increases with each additional feature, resulting in greater model complexity. The benefit of using this approach is that the implementation does not need to store long samples of historical data and instead only needs a single number to convey historical information.

3.2.4 Data derivatives

In addition to time delays, derivatives can be taken with respect to the time period of the data by subtracting each data sample from the value of the previous one. For example, if the current at time t is 100mA and at $t-1$ it is 75mA, the resulting derivative is $\frac{\partial i}{\partial t} = i_t - i_{t-1} = 100 - 75 = 25$mA. This method can be applied to each of the features i, SOC, and T. This value can be combined with the exponential moving window time delay approach to yield more information.

3.2.5 Time sequences

The LSTM and GRU both work on time series data. The training data consists of multiple hours of recordings, which will be split into sequences of continuous data. The optimal length of these sequences will be determined by searching for the length that yields the best performance during training.

3.3 Model training

Multiple architectures will be constructed, including linear regression, FFNN, LSTM, and GRU. Each model will be trained on the same training and evaluation data and benchmarked using the same test data. The linear regression and FFNN models will be given the data as simple feature-target pairs, while the LSTM and GRU models will be given the features and targets as continuous time sequences. The models will be developed with the PyTorch library [11] with various numbers of hidden layers. Each model will use validation data for early stopping, the Adaptive Moment Estimation (ADAM) optimiser, and weight decay. The code for the structure of each model is available in Appendix C. The trained model will be adapted for the specific hardware used by Scania, as mentioned in Section 2.3.2. In the final step, the model will also be tested on the Scania truck, with the methods outlined in Section 3.5.

3.3.1 Model evaluation criteria

The main model evaluation criterion used in the early stages of evaluation will be the mean squared error (MSE). MSE is a widely used loss function in machine learning that measures the average of the squares of the errors or deviations. Specifically, it quantifies the difference between the estimated values output by a model and the ground truth.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.3}$$

In Equation 3.3 $\hat{y}_i$ is the value estimated by the model, and y_i is the ground truth for *i'th* sample. n represent the total number of data samples.

In addition to MSE, it is essential for applications such as fault detection that the models maintain consistent performance without significant outliers. Therefore, robustness is also a critical evaluation criterion. Additionally, the deployment of these models in embedded hardware within trucks, must be memory-efficient and not consume excessive amounts of CPU resources, especially when making estimations across multiple battery cells. In essence, the models will be evaluated based on multiple criteria.

– **Mean Squared error loss:** The average performance of the model in terms of estimation accuracy.

– **Robustness:** Assessed by the model's 95th percentile error, focusing on minimising outliers to ensure reliable fault detection.

– **Memory efficiency:** The total size of the model in terms of number of weights must be considered to fit the constraints of the embedded hardware.

– **CPU utilisation:** The computational efficiency of the model, ensuring that it does not over-utilise CPU resources, particularly when processing multiple battery cells in sequence. This is measured as the increase in CPU utilisation over the baseline.

The CPU usage will be specific to the embedded hardware and will not be measurable until the trained model is implemented in embedded C. Therefore, it will only be measurable after a model has been selected and implemented for that purpose. The memory efficiency will be preemptively determined by the sizes of the models that are selected, this will be done based on the number of weights in the models.

3.3.2 Training procedure

The 18 data series will be separated into training, evaluation and testing groups. 15 series will be used for training, 2 will be used for validation and 1 will be used for final testing. When training the FFNN a single data sample from the training data was used at a time, while the two recurrent neural network will be given a sequence of 1000 continuous data samples as inputs. The evaluation data will be used as an early stopping indicator to prevent overfitting of the models. The testing data will serve as the final measure of accuracy and will be used to compare different model architectures. When evaluating and testing the models, the trained model estimates the voltage over the entire testing/evaluation data series and then the MSE is taken. For the FFNN this involves sequential estimates of each time step in the data, wile the recurrent neural network is given the entire series as a single sequence and thus updates the hidden state throughout. The first 1000 predictions are not taken into consideration for either model in order to account for the exponential decay filter and also to ensure the hidden state of the recurrent neural networks is given some time to adapt. As mentioned in Section 3.2 the different series have different temperatures. All the evaluation and test series were conducted at 30 ± 5°C as this is the expected operating temperature of the battery. The two evaluation and testing series were also taken from three different cells.

3.4 Hardware testing

When implementing the model on hardware, the ability to gauge its execution becomes limited. At this point, there is no way of altering the internal state of the model without spoofing input signals. However, the model can still run to determine the load on the CPU. This load will be measured when estimating the voltage of all 180 battery cells inside a module.

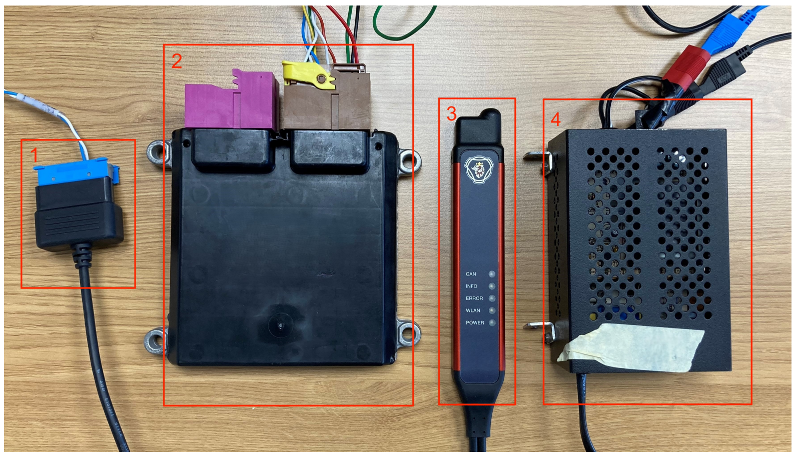


Figure 3.4: Setup of hardware used for testing and diagnostics. In order from left to right, controller area network connector (1), battery management unit (2), Trucks Diagnostic Tool (3), and 24V power supply (4).

The setup allows the embedded implementation to be run on the battery management unit in a controlled setting to verify that the model is stable and running without issues. The battery management unit can be interfaced with to measure performance in terms of CPU and memory usage.

In this context, CPU utilisation is defined as the amount of time used for a number of tasks to finish execution with respect to the amount of time they are allotted. For instance, if there are 5 tasks that are allotted 100ms and they all finish after 38ms, the utilisation is 38%. If the runtime of the tasks exceed the 100ms it is allotted, its deadline is violated, which can severely impact the performance of all systems that depend on or run on the battery management unit. Therefore, it is crucial that CPU utilisation is low and does not risk exceeding the time slot it is given. The tasks themselves are grouped into *loops* that the system is designed around. The separate loops are based on the number of times they execute per second: the $1Hz$, $10Hz$, $100Hz$, and $1000Hz$ loops. The loops with higher frequency run with a higher priority until completion, while loops with lower frequency yield until all loops with higher frequency have finished before executing. As a result, if a new module is added to the $10Hz$ loop, the $1000Hz$ and $100Hz$ loop are unaffected, while the $1Hz$ and $10Hz$ loops may observe higher CPU utilisation. Because both the LSTM and FFNN models execute in the $10Hz$ loop, they will only affect the performance of the $10Hz$ and $1Hz$ loops.

3.4.1 INCA testing

Because the model is running on the hardware, it becomes increasingly difficult to debug. A software called Integrated Calibration and Application Tool (INCA) [60] can be used to spoof signals to the battery management unit in real-time in order to observe the model's behaviour when executing on the battery management unit. INCA allows for the behaviours to be verified visually as a sanity check before the model is flashed onto a truck for driving.

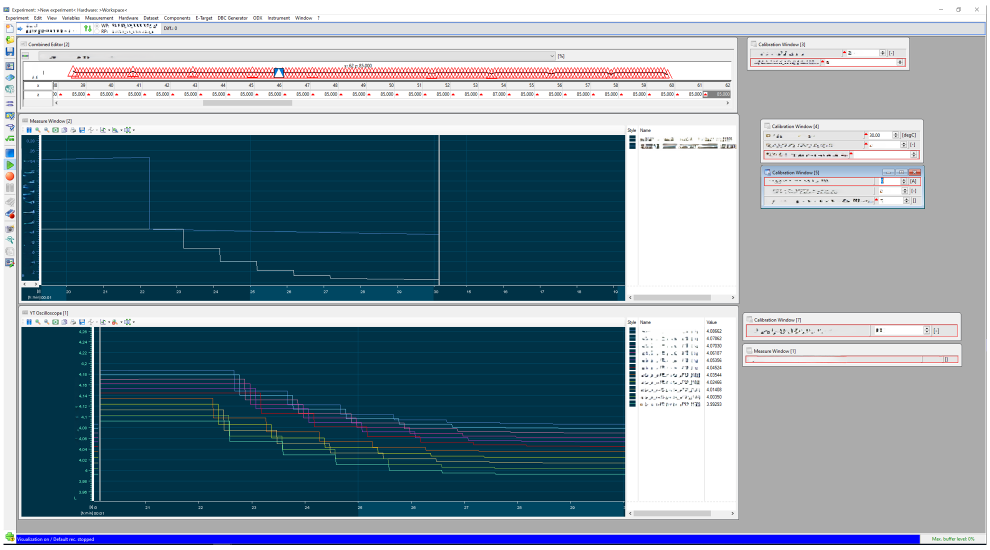


Figure 3.5: Screen image of the INCA software with spoofed signals. Source: Scania

Figure 3.5 shows the INCA software in use. The colourful graphs display the output voltage estimation for a selection of cells. On the right-hand side, spoofed measurements can be entered to observe how the model behaves when custom values for temperature, SOC, or current are input. INCA does not simulate the measured voltage, so the values in the graph cannot be compared to any ground truth, and the performance can only be validated by visual inspection. Once the model behaves as expected, it is ready to be tested in a live setting.

3.5 Driving test procedure

Since the model is expected to work in various scenarios, each needs to be tested to determine the model's ability to generalise from the lab data to a live setting. After the model is flashed onto the battery management unit, it will be tested in three separate scenarios listed below.

1. **Driving test:** The model will be tested by driving the truck on the test track at the Scania facilities in Södertälje. The track includes both uphill and downhill parts, as well as turns and straights. This type of driving is characterised by mostly negative but highly varying currents, generally decreasing SOC, and increasing battery temperatures.
2. **Charging test:** After the driving test, the truck will be connected to a charger for a period of time. The charging test is characterised by high but constant positive currents, high temperatures, and increasing SOC.
3. **Idle test:** After charging, the truck will be disconnected and remain stationary to measure the behaviour of the model as the batteries tended towards relaxation. This test is characterised by very low currents, constant SOC, and decreasing battery temperatures.

3.6 Assessing reliability and validity of the data collected

The data collected in the lab is measured in a controlled enrolment with precise equipment and runs little risk of being a source of error. The main issue is one of generalisation, where the scenarios used to test the cells in a lab may not accurately represent the cell behaviours when the truck is driving in a real setting with respect to the currents, temperature, and usage of SOC. Additionally, the state of health of the battery in the lab may not be an accurate representation of what can be expected in a real setting, thus potentially giving unreliable results when used as training data. The series from different cells may also have differences in behaviour between them.

3.6.1 Validity of method

The validity of the methods is supported by the multiple steps of model evaluation. The selected model was first tested using the PyTorch framework with truck data, followed by testing the same C implementation on the hardware test bench. Subsequently, the model was tested in INCA as well as in a real-time setting on the Scania truck. These separate steps ensured that the model's performance remained consistent. Additionally, testing the model on the truck in real time serves as a good way of examining the model in similar environment to where the model would be deployed.

3.6.2 Reliability of method

The initial data collection was carried out by the testing team at Scania, which consists of experienced engineers who can make accurate measurements using high-quality equipment. The feature engineering was conducted using a linear regression model, which is different from the final model architecture. Therefore, there are concerns that the chosen features suitable for the linear regression model may not be ideal for the final, more complex model. This is ultimately a limitation since training each complex model for each feature over multiple trials is a time-consuming task, and thus a simple baseline model was used instead. During training, each model will be tested at multiple stages throughout the development process to ensure that performance remains reliable. This will be combined with cross-validation to reduce variance in training conditions.

3.6.3 Validity and reliability of data

During the research, different types of data were collected, including model performance, model size, CPU utilisation, and driving data. Each of these has varying levels of validity. Firstly, the model size is trivial to calculate based on the number of weights. Model CPU utilisation will be measured using the INCA software; the measured value is the maximum utilisation over a period of time. Naturally, there will be some variation in utilisation due to random fluctuations and balancing of all the tasks running on the same battery management unit, thus a sufficiently long period of time needs to be monitored to obtain an accurate and generalisable result. Model performance can be calculated using any loss function, provided the data used is sufficiently accurate. This accuracy is limited by the precision of either the simulation software or the data used to measure model performance.

In terms of the reliability of the data, the main concerns arise from the actual test drive in the truck. Since multiple factors can affect the model during driving, these need to be taken into account when making conclusions regarding the data collected. These factors include the ambient temperature, the initial SOC of the truck, the driving style, and the relation between the amount of time spent driving, charging, and idling.

3.7 Planned data analysis

The data collected will be analysed to evaluate the model's performance. When evaluating the results, the mean absolute error (MAE) will be used to provide a meaningful measure of the model's accuracy. The results will be analysed with respect to the simulation in Section 5.2 and the test drive in Section 5.3.

3.7.1 Software tools

Various tools will be used throughout the research process. Initially, the following Python libraries will be used for training the model: `Numpy`, `pandas`, and `pytorch`. To create graphs, the following tools will be used: `matplotlib` and `seaborn`. To simulate the battery cell, proprietary software at Scania will be used. When interfacing with the battery management unit, the software Integrated Calibration and Application Tool (INCA) will be utilised. Lastly, the software Measure Data Analyser (MDA) will be employed to process the data from the truck.

Chapter 4

Model Development

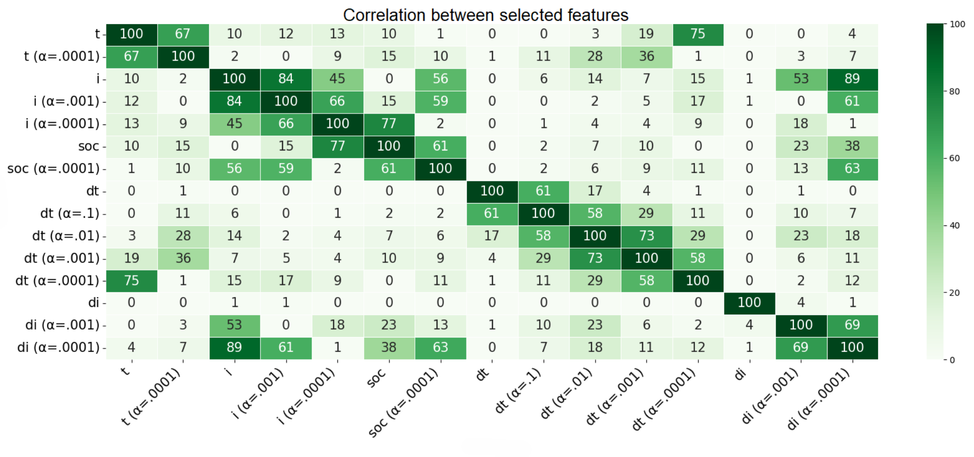
In this chapter, the model development is discussed, starting with feature engineering for identifying the most relevant features for the models, as detailed in Section 4.1.

Next, model selection is conducted in Section 4.2, where different neural network model architectures are examined, including feedforward neural network (FFNN), long short-term memory (LSTM), and gated recurrent unit (GRU) models. The architecture of each selected model is discussed in Section 4.2.1.

Finally, the implementation of the selected model in embedded C is discussed in Section 4.3. This includes the process of converting model weights from Python to C, and the specific considerations for implementing FFNN and LSTM models in embedded systems. The detailed implementation schematic for the LSTM model is illustrated in Figure 4.7.

4.1 Feature engineering

Following the creation of features in Section 3.2.2, the next task is to determine which of these are the most salient and should be selected for the models. After the features are developed, the number of features needs to be explored by checking feature correlation. We want to minimise the number of features in order to reduce the computational cost, so introducing a new feature whose information is already covered by another feature is redundant and may introduce issues with multicollinearity [61]. The correlation between all the features can be seen in a correlation matrix in Figure B.1 in Appendix B. After removing correlated features a correlation matrix of the remaining features is visible in Figure 4.1.



	t	t (α=.0001)	i	i (α=.001)	i (α=.0001)	soc	soc (α=.0001)	dt	dt (α=.1)	dt (α=.01)	dt (α=.001)	dt (α=.0001)	di	di (α=.001)	di (α=.0001)
t	100	67	10	12	13	10	1	0	0	3	19	75	0	0	4
t (α=.0001)	67	100	2	0	9	15	10	1	11	28	36	1	0	3	7
i	10	2	100	84	45	0	56	0	6	14	7	15	1	53	89
i (α=.001)	12	0	84	100	66	15	59	0	0	2	5	17	1	0	61
i (α=.0001)	13	9	45	66	100	77	2	0	1	4	4	9	0	18	1
soc	10	15	0	15	77	100	61	0	2	7	10	0	0	23	38
soc (α=.0001)	1	10	56	59	2	61	100	0	2	6	9	11	0	13	63
dt	0	1	0	0	0	0	0	100	61	17	4	1	0	1	0
dt (α=.1)	0	11	6	0	1	2	2	61	100	58	29	11	0	10	7
dt (α=.01)	3	28	14	2	4	7	6	17	58	100	73	29	0	23	18
dt (α=.001)	19	36	7	5	4	10	9	4	29	73	100	58	0	6	11
dt (α=.0001)	75	1	15	17	9	0	11	1	11	29	58	100	0	2	12
di	0	0	1	1	0	0	0	0	0	0	0	0	100	4	1
di (α=.001)	0	3	53	0	18	23	13	1	10	23	6	2	4	100	69
di (α=.0001)	4	7	89	61	1	38	63	0	7	18	11	12	1	69	100

Figure 4.1: Heatmap of correlation between reduced set of features. All coefficients are multiplied by 100 and lies in the range 100-0 instead of 1-0.

The features from Figure 4.1 were then further examined in order to find the best performance in Section 4.1.1.

4.1.1 Feature selection

Before training a complex model, the relevant features created in Section 3.2.2 need to be determined. This was done using a simpler linear regression model. The model was provided with the base three features t, i, and soc. The performance of this model was measured and became the *baseline*. Thereafter, another model was given one additional feature at a time and trained. The new performance was measured again, and the relative increase as a percentage was calculated. This process was repeated with each engineered feature to determine which ones provided the model with the most useful information.

The data used was a random sub-sample of 1,000,000 data points from the training data, and the performance was measured against 1,000,000 unseen data points. This method of training the linear regression model is showcased in Listing 4.1.

```
base_features  = ['soc','t','i']
extra_features = [
    'soc_0001', ... 'soc_75',
    't_0001',   ... 't_75',
    'i_0001', ... 'i_75',
    'dsoc',   ... 'dsoc_75',
    'dt',    ... 'dt_75',
    'di',   ... 'di_75',
]
y      = df['dyn']
y_test = df_test['dyn']

results = {}
for feature in extra_features:
    X = df[base_features + [feature]]
    X_test = df_test[base_features + [feature]]
    model  = LinearRegression(X,y)
    results[feature] = model.score(X_test, y_test)

print(results)
```

Listing 4.1: Code to calulate performance of different features

This process was repeated 5 times, and the average increase in performance was used as the final improvement. The results of the search for relevant features are visible in Figure 4.2 as a heat-map. The value of each box represents the percentage increase in performance over the baseline.

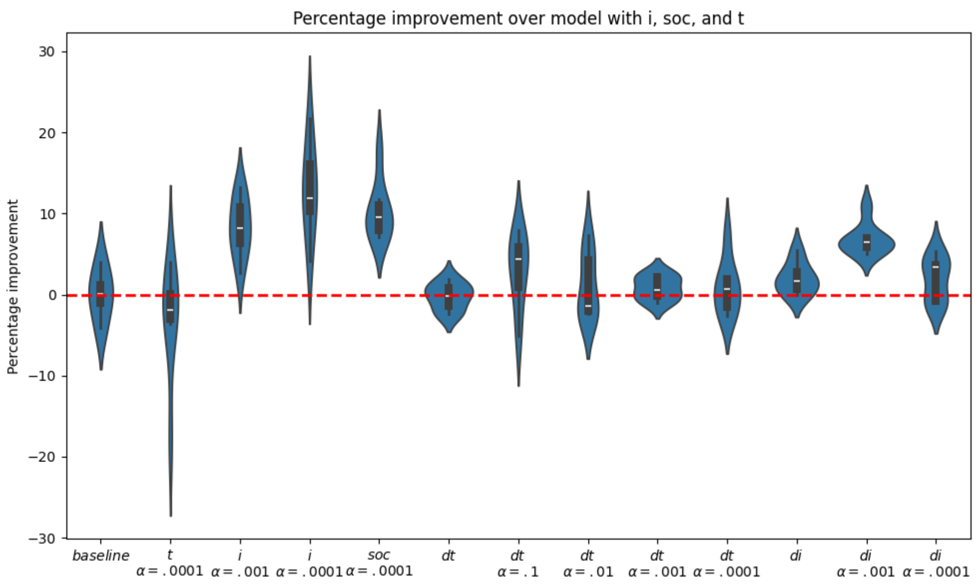


Figure 4.2: Violin plot of the percentage improvement over the baseline when adding new a feature in isolation to a linear regression models.

In Figure 4.2, the best improvement over the baseline was achieved with the current feature (i) having an exponential decay with an alpha of $\alpha = 0.0001$, which resulted in an average decrease in loss of 12% over the baseline. From the accompanying violin chart, it is also apparent that this improvement is statistically significant as the 1.5x interquartile range lies above the baseline. This is reasonable since the feature provides the model with information about the long-term usage of the battery cell, for example, whether the battery has been continually discharging or charging for a long period of time. However, the same feature does not differentiate between long periods of zero usage and the same time period of high oscillating use.

The next step is to repeat the same process but with $i_{\alpha=0.0001}$ added as a base feature in order to determine which next feature should be added. The results of each iteration are shown in Appendix B, while the overall results for each feature are shown in Figure 4.3.

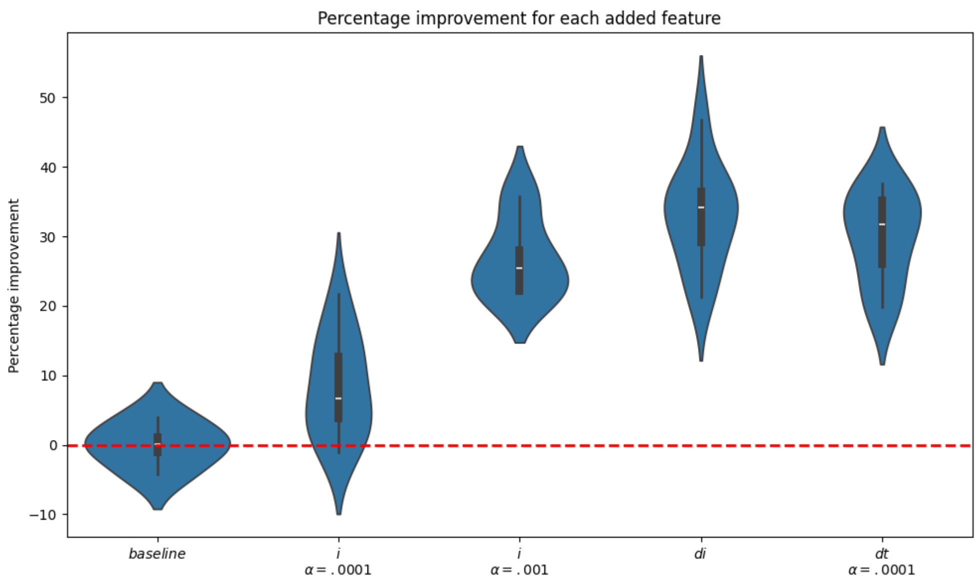


Figure 4.3: Violin plot of each improvement for the features $i, t, soc, i_{\alpha=0.0001}$, $i_{\alpha=0.001}$ di and $dt_{\alpha=0.0001}$ over the initial baseline

Figure 4.3 shows the percentage improvement over the original baseline for each added feature. The best features tended to be those related to different aspects of the current, which is logical as it is the feature with the most variation and likely the one that affects the terminal voltage the most in the short-term. The feature $dt_{\alpha=0.0001}$ was decided to be the stopping point since it did not provide significant improvement over the previous features. The final features selected for the model are i, t, soc, $i_{\alpha=0.0001}$, $i_{\alpha=0.001}$, and di.

4.2 Model selection

Once the features of the model are decided, the structure and architecture of the model must be examined in order to select the most feasible models configuration. Initially, different model architectures are discussed in Section 4.2.1, including long short-term memory (LSTM), gated recurrent unit (GRU), and feedforward neural network (FFNN) models. Key structural elements such as layer size (width) and number of layers (depth) are also examined. Next, Section 4.2.1.1 delves into the FFNN architecture, highlighting the impact of varying layers and nodes, as well as the importance of regularisation to prevent overfitting. Section 4.2.1.2 then addresses LSTM and GRU models, emphasising their ability to manage temporal sequences with internal memory. Finally, Section 4.3 discusses the implementation of selected models in embedded C, covering the conversion of model weights and specific considerations for deploying FFNN and LSTM models in embedded systems, as illustrated in Figure 4.7.

4.2.1 Model architecture

In Chapter 2, three main model architectures were examined: LSTM, GRU, and FFNN. Each model comes with a set of hyperparameters that significantly impact its performance. When running the model, there are two main parameters that determine the model characteristics in terms of implementation: width and depth. The width refers to the size of each layer in the model, while the depth refers to the total number of layers.

The initial sizes of hidden layers that will be used are 8, 16, 32, 64, 128, and 256 nodes to provide a range of sizes spanning multiple orders of magnitude. The number of layers for the FFNN will range from 0 (linear regression) to 4 layers.

4.2.1.1 Feedforward architecture

The feed-forward neural networks have two main important factors to consider in terms of structure: depth and width. Depth refers to the number of layers, and width refers to the number of nodes in each layer. Additionally, there is the regularisation factor, which is used to prevent the model from overfitting. Notably, the model will not contain an internal memory, and the only source of time dependency will be derived from the time-dependent features determined in Section 4.1.

4.2.1.2 Recurrent neural network architecture

The LSTM and GRU models also have concepts of depth and width. However, due to the significant increase in size, only single-layer LSTM and GRU models will be considered for this report. Instead, the effects of regularisation and the size of the hidden layer will be examined. Compared to the FFNN, the two recurrent neural networks will have a concept of internal memory and, as a result, will be less sensitive to variations in the input data if trained correctly.

4.2.2 Effect of model size

In order to gauge the complexity the model needs to have in order to solve the problem, a sweep over model type and complexity was carried out. Each model was trained using 1 million data samples from the training dataset and then tested on an entire dataset of approximately 50,000 samples. Each model was given a total of 30 epochs to train using the ADAM optimiser. For the LSTM and GRU, a batch size of 10 and a sequence length of 1000 were used, while the FFNN had a batch size of 10,000 (resulting in an equal number of training samples for each model). Early stopping was implemented in case the model diverged from its best result for more than 3 epochs in a row. 5-fold cross-validation was used to gauge the distribution of the results. The results are shown in Figure 4.4.

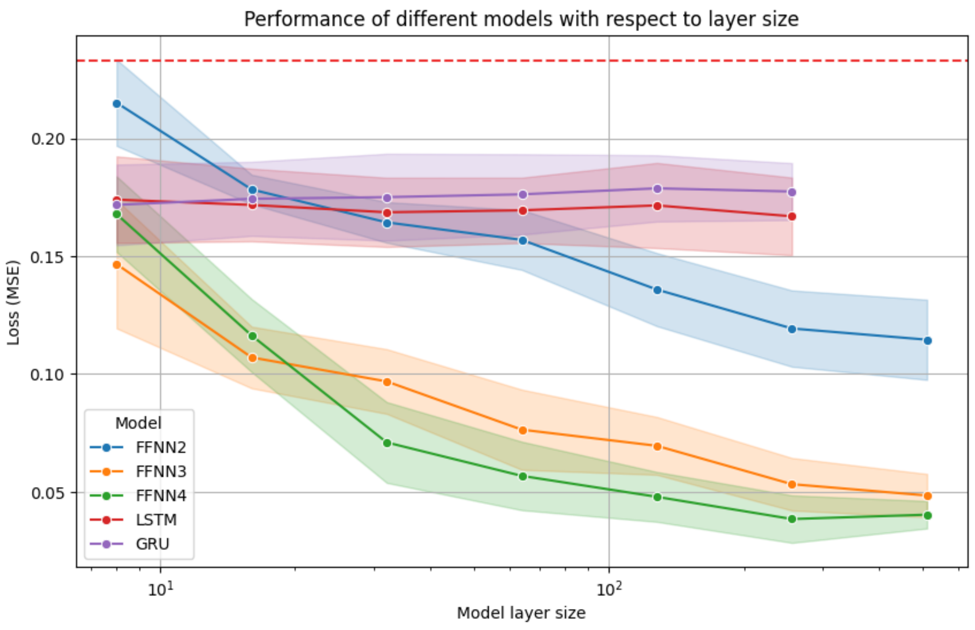


Figure 4.4: Performance of models of difference layer size with respect to the validation loss using the MSE. The x-axis displays the size of the hidden layers in the model using a logarithmic scale, while the loss is shown on the y-axis (a lower loss indicates better performance). The dashed red horizontal line represents the performance of the linear regression model, and each line represents a model subject to changes in the size of the hidden layer. The highlighted area behind each line shows the spread of the standard deviation over the 5-fold cross-validation. The FFNN models are marked with numbers indicating the total number of layers in the model. The clear trend observed is that as the size of the models increases, the loss decreases, indicating better performance.

While Figure 4.4 provides an indication of the number of units in the hidden layer, it does not accurately show the increase in size in terms of the total amount of memory used to store the model. To illustrate this, the total size of the model in terms of the number of individual weights can be plotted instead, as it gives a more realistic indication of the models' overall size. This is shown in Figure 4.5.

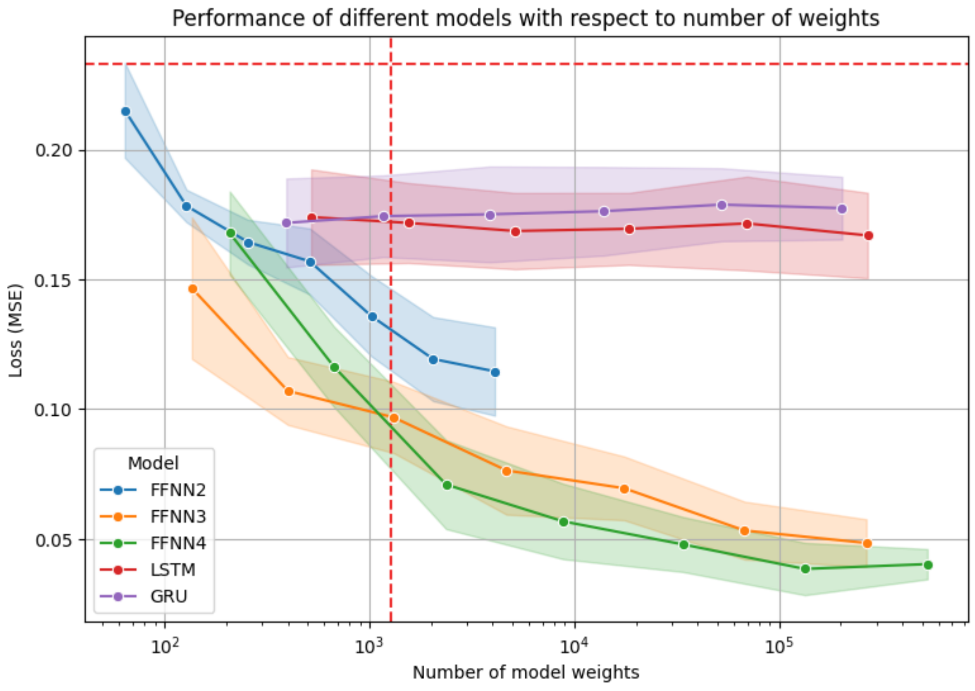


Figure 4.5: Performance of models of different number of weights with respect to the validation loss. The vertical red line represents the approximate number weights (single precision floating point numbers) that the model can use give the constraints of the battery management unit as discussed in Section 2.3.2. The horizontal line is the performance of the previously established linear regression model.

Looking at Figure 4.5, the difference in size between the FFNN2 and FFNN4 is clearly indicated as they both now span a different range on the x-axis. Notably, the largest two-layer FFNN models are comparable in size to a three-layer FFNN with a hidden layer size of 64.

At this point, a number of further design decisions were made. The first one is to proceed with one FFNN model and one RNN model. This is because ultimately only two models can be tested in the truck at the same time. The models chosen were the LSTM with a hidden size of 16, as it was the best performing RNN with the size closest to the vertical red line, and the FFNN with three layers.

4.2.3 Hyper-parameters selection

Beyond the general structure of the models, there are a few parameters that can be tuned, including sequence length for LSTM, learning rates, weight decay, dropout, and the number of epochs of training.

When training the LSTM with different sequence lengths, the length had no effect on the final performance of the model. Therefore, the original sequence length of 1000 was used, as it represents about 17 minutes, which was deemed a reasonable amount of time for battery cell behaviour to be captured. Similarly, dropout was experimented with for the FFNN, but it only had a negative effect on the performance.

Learning rates were dynamic throughout the training using the ADAM optimiser, where the maximum learning rate was decreased when the model loss did not decrease for consecutive epochs. The number of epochs was increased while still using early stopping, this time with a patience of 5 epochs.

A random sweep over different amounts of weight decay (L2 regularisation) was conducted to find the best value for each model. The results are visible in Figure 4.6.

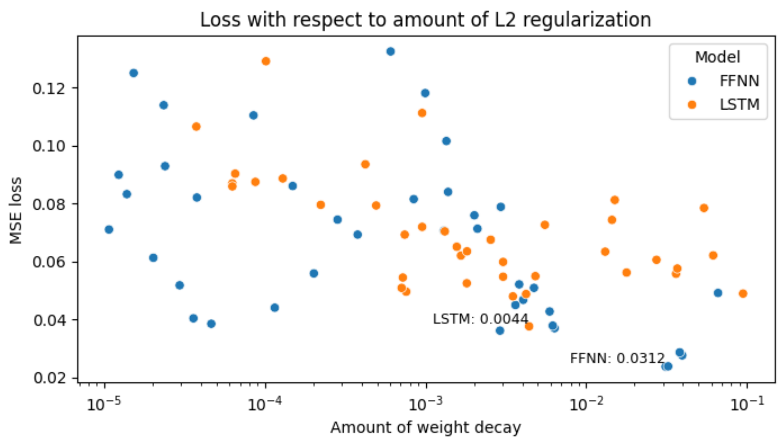


Figure 4.6: Random sweep over the amount of weight decay for LSTM and FFNN using the procedure described in Section 4.2.3.

In the random sweep over the different amounts of weight decay, the FFNN seemed to benefit more from weight decay than the LSTM. The best weight decay values were 0.0044 for the LSTM and 0.0312 for the FFNN.

4.3 Model implementation

The trained model must be implemented in embedded C to run on the battery management unit. Because the model is pre-trained, only the forward pass of the model must be implemented, simplifying the process.

4.3.1 Weights conversion

Implementing the selected model in C involves extracting the weights from the model in PyTorch. The initial approach was to use the ONNX library discussed in Section 2.3.3, however, this library did not provide the necessary tailoring and customisation required by the battery management unit. As a result, each model was implemented and tested by hand instead of through code generation, posing a new challenge. First, the weights of each model need to be converted to C code. The method to do this is showcased in Listing 4.2.

```
import torch

def as_c_array(python_list):
    # takes python list returns string of C array
    # input: [1.0,2.0,3.0]
    # output: "{1.0f,2.0f,3.0f}"
    element_str = "f, ".join(map(str, python_list))
    return "{" + element_str + "f}"

def weights_to_c_code(weight_tensor, name):
    # Take a python array and return the correpsonding C code
    # input: [[1.0, 2.0], [3.0, 4.0]], "fc.1"
    # output: """const float[2][2] fc_1 = {
    #   {1.0f, 2.0f},
    #   {3.0f, 4.0f}
    # }"""
    np_array = weight_tensor.numpy()
    array_type = "const float " + name.replace(".", "_")
    array_dimension = "[" + "][".join(np_array.shape) + "]"
    array_values = " = "
    if np_array.ndim == 1:
        array_values += as_c_array(np_array.tolist())
    else:
        element_str = [as_c_array(row) for row in np_array]
        array_values += "{\n"
        array_values += ",\n".join(element_str)
        array_values += "}\n"
    return array_type + array_dimension + array_values

path = "/path/to/model"
model = torch.load(path)

for name, param in model.named_parameters():
    print(weights_to_c_code(param.data, name))
```

Listing 4.2: Python code that takes a pytorch model-weights and returns a string of the weights formatted as C code

With the code in Listing 4.2, the weights in the Python-based PyTorch model can be converted to arrays of floats that can be used in C code. One thing to note is that the weights are stored as double precision floats in Python, and for the particular use case of the Scania trucks, they need to be represented as 32-bit floats. In this conversion process, some precision is lost, which may affect the performance of the model after being implemented in C. This will be examined later in Section 5.2.1.

4.3.2 FFNN implementation

The nature of feedforward neural networks consists of simple matrix multiplications, the implementation of which is straightforward enough not to be covered in this report.

4.3.3 LSTM implementation

When designing the code implementation for the LSTM, the specific embedded requirements of the target hardware need to be considered. These include the amount of memory used as well as CPU utilisation, but also the predictability of these factors. While the source code of the implementation will remain a property of Scania, the schematic for the implementation is shown in Figure 4.7.

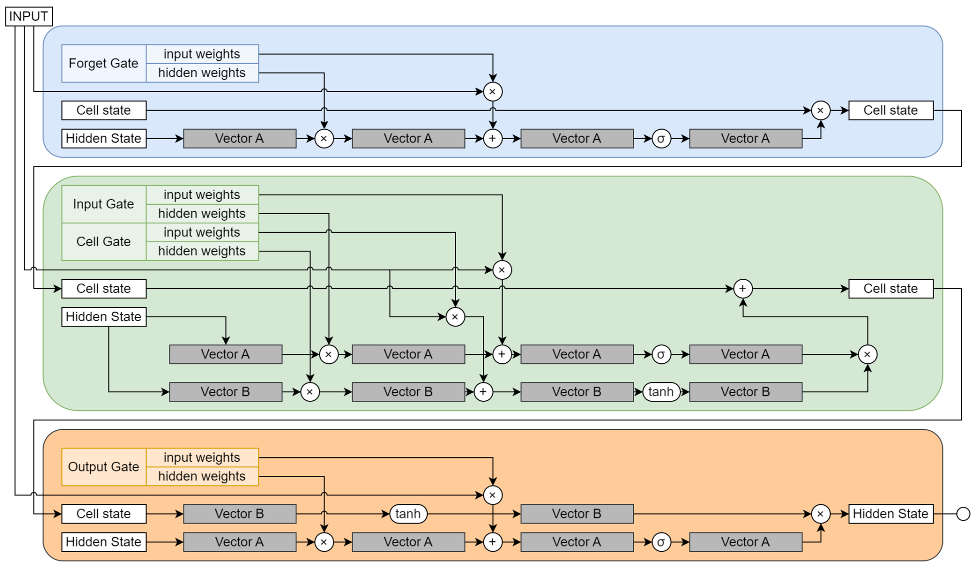


Figure 4.7: Graphical illustration of LSTM C implementation. The cell and hidden state represent the internal state of the LSTM, while $VectorA$ and $VectorB$ are temporary vectors used to store intermediary values. The final hidden state is the output of the entire model.

From Figure 4.7, the code implementation of the LSTM is visible. All operations are either matrix multiplication with a vector, element-wise multiplication of two vectors, or applying a function to each element in a vector. The weights themselves are retrieved using the code in Listing 4.2. The cell state and hidden state are initialised to 0, which is a limitation of the current implementation. After the hidden state of the cell is produced, a linear transformation is used to produce a single output value; this is not shown in Figure 4.7.

Chapter 5

Results and Analysis

This chapter is concerned with the results of the model simulation, hardware testing, and live model testing, with a focus on comparing the performance of long short-term memory (LSTM) and feedforward neural network (FFNN) models used in this project. Section 5.1 discusses the results of the trained models on the lab data. In the model simulation (Section 5.2), the C implementation of the models is tested using Scania's in-house simulation software. The performance of the embedded LSTM and FFNN models is compared with the Python-based models using a simulation on real driving data, providing an initial evaluation of the models before hardware implementation. The live model testing is done in (Section 5.3), the models are tested in real-time on a Scania truck. The test performance of the LSTM and FFNN models is evaluated during driving, charging, and idling scenarios. Finally, In hardware testing (Section 5.3.3), the models are deployed on the hardware to assess CPU utilisation and overall performance in a controlled environment.

5.1 Lab data testing

After having selected the hyperparameters for the the LSTM and FFNN the model was trained on the whole training dataset with the evaluation series used for early stopping with the evaluation procedure described in Section 3.3.2. The test series consisted of 240 616 samples. The results in terms of mean squared error and mean absolute error are showcased in Table 5.1.

Model	Mean Squared Error	Mean Absolute Error	95% Confidence Interval
FFNN	0.005171	4.522	4.493 to 4.551
LSTM	0.008063	4.355	4.312 to 4.396

Table 5.1: Comparison of mean squared error and mean absolute error for the best FFNN and LSTM models on lab data. The unit of the mean absolute error is millivolt. The 95% confidence interval was calculated on the mean absolute error using the t-distribution.

In Table 5.1 the mean squared error of the FFNN is smaller than that of the LSTM, while for the mean absolute error the opposite is true. This discrepancy implies that while the LSTM model generally makes more accurate predictions, it occasionally makes larger errors. The 95% confidence intervals between the two mean absolute errors do not overlap, suggesting that there is a statically significant difference between the results. Figure 5.1 is a subset of the estimations of the two models on the testing data compared to the measured voltage.

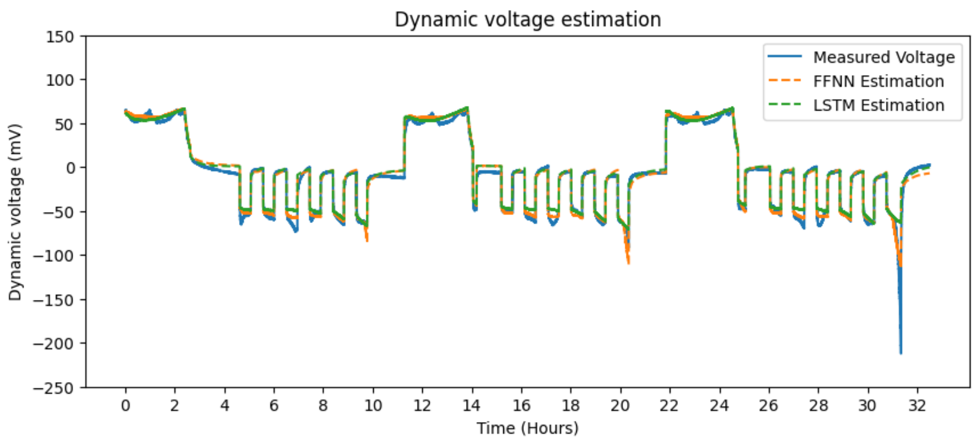


Figure 5.1: Subset of model estimation on the testing data.The temperature during this test subset hovered around $30 \pm 5°C$. The cell was charged to 100% SOC and discharged to 10% in bursts of approximately 50A, repeating this cycle three times over about ten hours each. After the third cycle, the cell was fully discharged to 0% SOC, leading to a voltage collapse visible around the 31-hour mark. Source: Scania

From Figure 5.1 the two models have clearly learned the cells' general dynamic behaviour. However both models seem to not catch the temperature variations peaks. The FFNN does a better job than the LSTM at modelling the dips when the cell reaches a low SOC. Neither model accurately models the voltage collapse at hour 31, but the FFNN does a slightly better job than the LSTM. The voltage variations at the 0, 11, and 22-hour marks are due to temperature changes, something neither model seems to have fully learned.

5.2 Model simulation

The implemented C module can be simulated using in-house software available at Scania. This allows custom values to be provided in order to compare its performance with the Python-based model. The simulation is based on a version of the equivalent circuit model, and thus is limited in terms of accuracy. However, it provides a good indication of how well the model performs and acts as a sanity check before the model is run on the truck's hardware. The results of the simulation are shown in Figure 5.2.

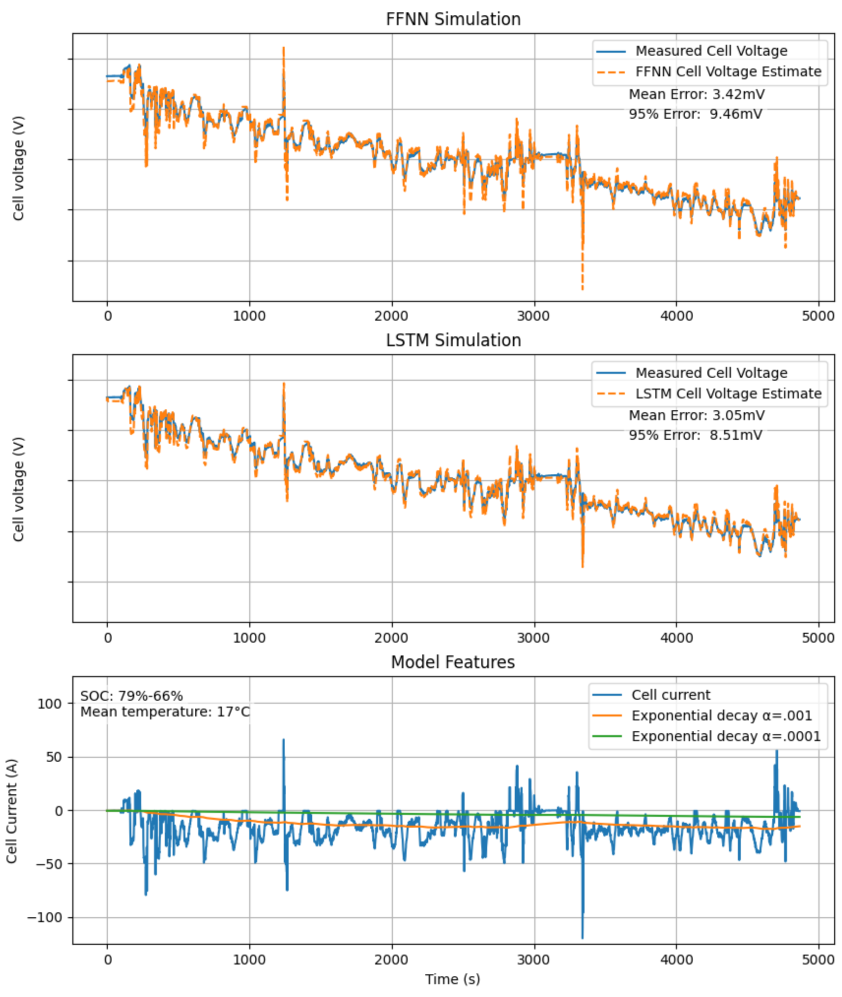


Figure 5.2: Simulation of the LSTM and FFNN with real driving data, with corresponding features. About 80 minutes of real driving data is used as the input to the Scania simulation software. The first two subplots of the figure show the measured and estimated terminal voltage for the two models. Third subplot of the figure shows the inputs given to the models. The SOC started at 79% and decreased to 66% over the course of the drive. The temperature remained steady at approximately 17°C. The current drawn from the battery is shown by the blue line, while the orange and green lines represent the exponential decay for different values of alpha. Source: Scania

In the simulation, it is clear that both the LSTM and FFNN models have captured the dynamics of the cell voltage. The orange line closely follows the blue line. The mean absolute error (MAE) for the LSTM was 3.05mV, with

95% of all errors being less than 8.51mV, while the FFNN had a slightly higher MAE of 3.42mV and a 95% percentile error of 9.46mV. Note that Figure 5.2 shows the terminal voltage compared to Figure 5.1 which only shows the dynamic voltage, this means that the orange lines is the sum of the models' estimates and and open circuit voltage. Given these results, both models are feasible candidates for hardware implementation moving forward.

5.2.1 PyTorch and C comparison

In order to ensure that the C implementation of the model performs on par with the PyTorch implementation, the features created in Figure 5.2 were given to the PyTorch implementation of the same models. The results of this comparison are presented in Table 5.2.

Model	LSTM	FFNN
C	3.05	3.43
PyTorch	3.13	3.39

Table 5.2: Mean absolute error comparison of PyTorch and C implementations of the LSTM and FFNN models when running inference on the same data. All values are in millivolt.

From Table 5.2, it is apparent that the difference between the PyTorch and C implementations is minuscule, both being on the order of tens of μV. This difference is likely explained by the C implementation using 32-bit floating point numbers, while the PyTorch model uses 64-bit floats. Moving forward, it is assumed that the difference in performance between the Python and C implementations of the models is negligible.

5.3 Live model testing

Once the models were verified to function on hardware, they were then flashed onto a Scania truck to operate in a real-time setting. A similar setup to Figure 3.4 was used, except for the power supply, which was not needed. The test was carried out using the testing procedure outlined in Section 3.5.

5.3.1 Test drive results

The tests were carried out on a battery electric vehicle Scania truck in sunny weather with an ambient air temperature of 18°C on Scania's test track, with no coupled trailer. The initial state of charge was 86.0%, which decreased to 83.0% over the course of the drive. The truck was later charged back up to 87.5%, which is the same SOC that the truck was idling at.

The test provided about 35 minutes of data: 21 minutes of driving, 11 minutes of charging, and 3 minutes of idling. The two models were run simultaneously on two separate battery packs. Although the models were running on all 180 cells in each pack, due to the limited bandwidth of the controller area network bus, only the results of 5 cells were recorded for each model in parallel. The results for each model are shown in Figure 5.3 and Figure 5.4, respectively.

5.3.1.1 FFNN test results

The FFNN model was run on the target battery management unit, while the LSTM was run on the controller battery management unit*. The results for a single battery cell can be seen in Figure 5.3.

*Due to terminology concerns the terms master/slave are replaced by controller/target

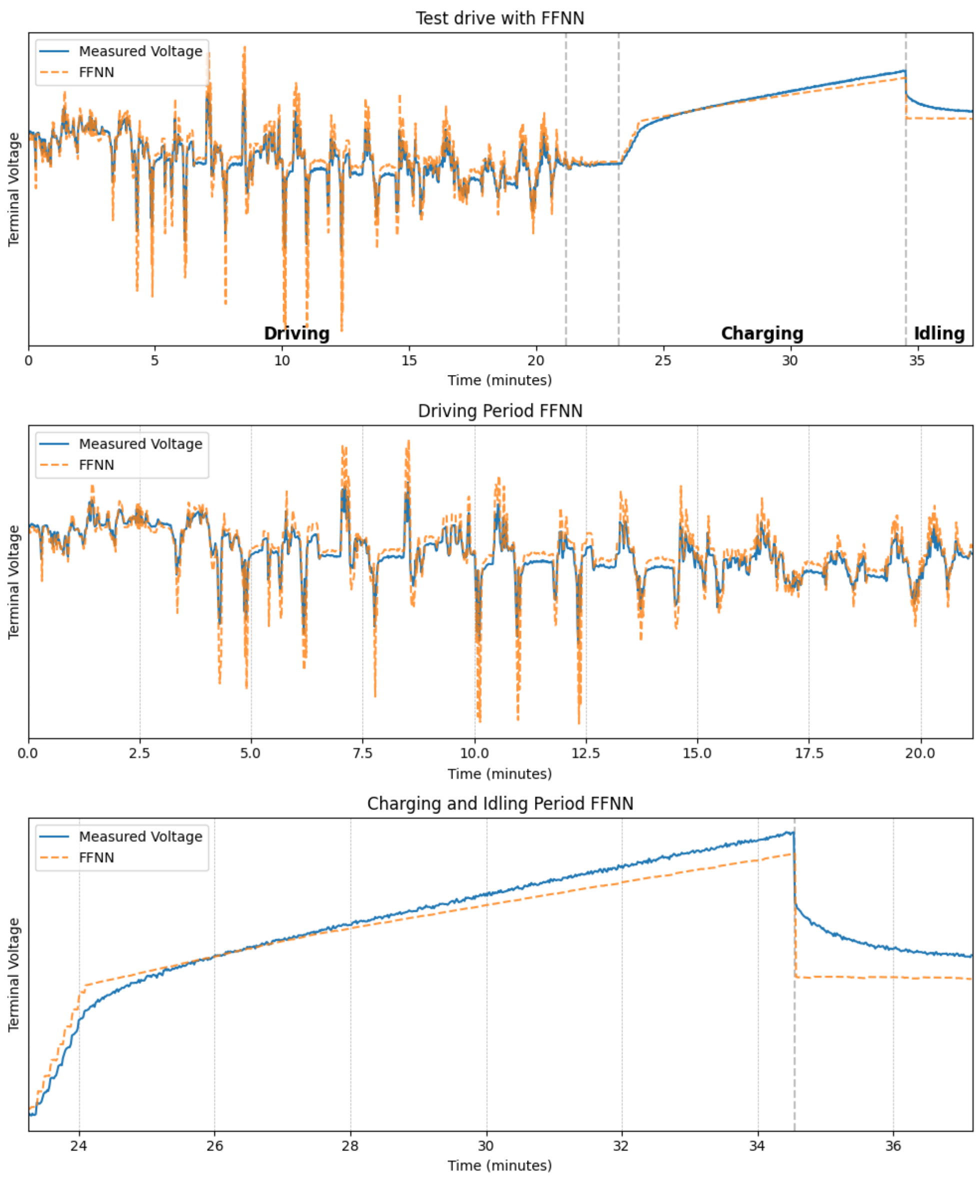


Figure 5.3: Test drive of FFNN and measured voltage for a single battery cell for each driving test. The FFNN model (dashed green line) is compared with the measured voltage (solid blue line). The first subplot shows the full test drive, with the driving, charging, and idling sections are separated by the vertical grey lines. Between the driving and charging section there is a period of time which was discounted as the truck was in the process parking and the charger was activated. The next subplot isolates the driving section, followed by a combination of the charging and idling sections.

During the driving test shown in Figure 5.3, the model follows the measured voltage closely during periods of low variation in terminal voltage. However, during large changes, the FFNN appears to be too sensitive to large variations, overestimating the voltage in both directions. The relative performance of the FFNN seems consistent throughout the entire driving test, suggesting it does not drift away from the measured voltage. However, given that the driving test is limited to only 20 minutes further testing is need in order for a conclusion to be ascertained.

During the charging test, the FFNN model does a good job capturing both the initial ramp-up of the charging rate and then following the linear characteristics of the increasing voltage during charging. However, the model does not find the exact slope of the increasing voltage, with the gradient being a bit too shallow compared to the measured voltage.

In the idling test, the FFNN does not perform well. It seems not to have learned the slower dynamic properties and instead locks down to the open circuit voltage almost instantly. This could be due to this type of scenario not being well represented in the training data or the features given to the model not being complex enough to model this behaviour, because the FFNN has no internal state. Ideally, the idling period should have been longer; however, due to time constraints on the testing day, this was not possible.

5.3.1.2 LSTM test results

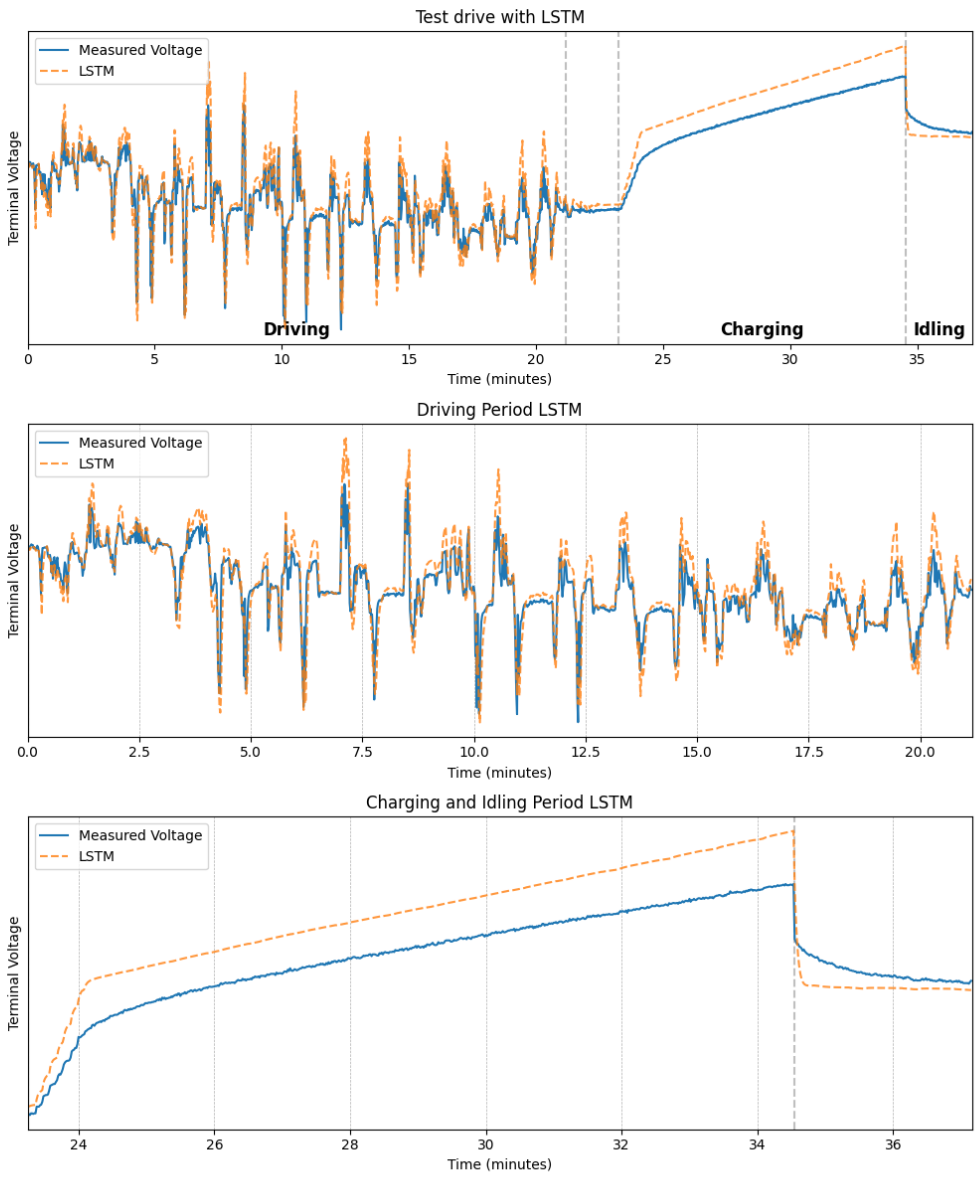


Figure 5.4: Test drive of LSTM with measured voltage for a single battery cell for each driving test. The layout of the figure is the same as in Figure 5.3, with the exception that this time the green line represents the LSTM model.

As stated previously, this was run on a separate battery pack from the FFNN, but because the drive happened simultaneously, the data will not be different enough to have a substantial impact on the conclusions that can be drawn. Similar to the FFNN, the LSTM model follows the measured voltage closely during periods of low variation in terminal voltage in the driving test. It also tends to overestimate the spikes in voltage. However, unlike the FFNN, when the current dips significantly, the LSTM is more careful and accurate in its estimation. Just like the FFNN, the relative performance of the LSTM seems to be consistent throughout the entire driving test.

During the charging test, the LSTM consistently overestimated the voltage. While the gradient is accurate and seems to mimic the measured voltage, there is an offset of approximately 17mV.

Similar to the FFNN, the LSTM does not seem to have captured the idling relaxation behaviour of the cell. There is some visible smoothing over the first few seconds of the test, likely a result of the internal state of the LSTM, but it is not comparable to the expected behaviour.

5.3.2 Performance comparison

Comparing the results of the different models and cells makes a few assumptions about the data. The first assumption is that the different cells within a battery pack behave similarly, and the second is that the behaviour between battery packs is similar. To measure performance, the mean absolute error was taken between the model whose performance is to be measured and the measured terminal voltage. This was first done for the entire test session and then for each individual test to see in which parts the models perform well. The results are visible in Figure 5.5.

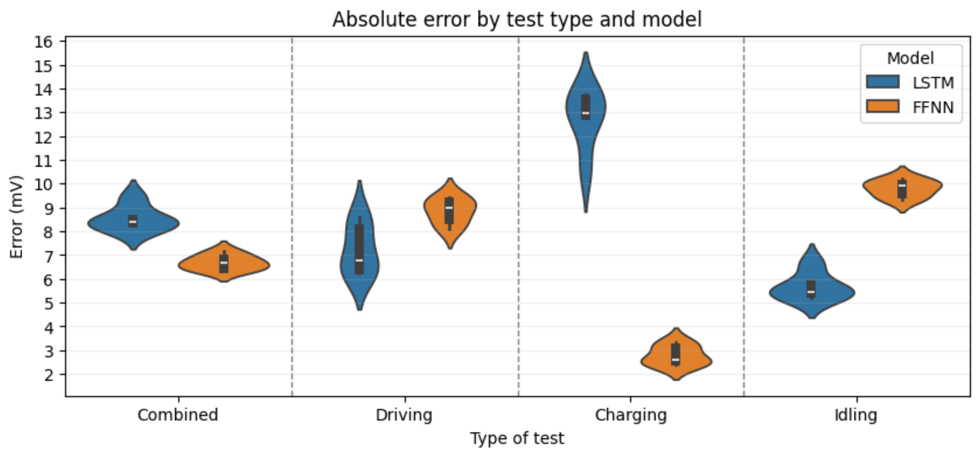


Figure 5.5: Violin plot of mean absolute error of each battery cell for different models and test types. Each element in the figure includes a box plot of the mean and interquartile range of the data, while the shape of the violin indicates the spread of the data from the different cells. The different test types are different parts of the test drive. The combined test is using the entire test drive as a single dataset

From Figure 5.5, the ML models, the MAE across the combined tests is 8.54mV for the LSTM and 6.69mV for the FFNN. It is clear that the performance varies greatly across the different tests. The LSTM has the worst overall performance in the combined test, likely due to the significant errors from the charging test, despite being more accurate than the FFNN for both the driving and idling tests. These results are further examined in Table 5.3. The FFNN is the ML model with the best performance, particularly in the modelling of the charging scenario. However, for the other tests, the FFNN is the worst-performing model.

Error type	Test	Model	Cell 0	Cell 18	Cell 36	Cell 56	Cell 72	Average
MAE	Combined	LSTM	8.52	8.34	8.01	9.41	8.41	**8.54**
		FFNN	7.14	6.82	6.68	6.36	6.44	**6.69**
	Driving	LSTM	6.77	6.38	6.26	8.61	8.08	**7.22**
		FFNN	9.45	9.22	8.99	8.51	8.10	**8.85**
	Charging	LSTM	13.58	13.74	12.87	12.96	10.63	**12.75**
		FFNN	3.11	2.52	2.57	2.38	3.34	**2.78**
	Idling	LSTM	5.20	5.44	5.40	5.72	6.65	**5.68**
		FFNN	9.94	9.57	9.33	9.98	10.23	**9.81**
95% Percentile Error	Combined	LSTM	17.74	17.98	16.88	21.43	20.33	**18.87**
		FFNN	21.06	20.86	19.68	19.56	18.06	**19.84**
	Driving	LSTM	21.22	19.72	18.62	28.76	26.42	**22.95**
		FFNN	27.63	27.64	26.80	27.46	24.44	**26.79**
	Charging	LSTM	17.25	17.70	16.69	16.31	13.98	**16.38**
		FFNN	6.12	6.66	6.31	6.37	6.41	**6.37**
	Idling	LSTM	10.64	10.71	10.45	11.30	12.15	**11.05**
		FFNN	17.45	16.60	16.64	17.13	18.13	**17.19**

Table 5.3: Absolute error and 95th percentile error for the FFNN and LSTM for each test and individual cell over the driving, charging and idling period of the test-drive. The entire test-drive is shows in the combined row. All values are in millivolt.

Table 5.3 shows the errors and 95% percentile error of each individual cell for each model in a table format. Note that the results may not be exactly comparable because each model runs on different battery packs. The 95% percentile error is the value that is greater than exactly 95% of all measured error values. This metric is used because, in addition to having good average performance, a model that is also consistent is preferable.

Inspecting the 95% percentile error, the LSTM seems to consistently outperform the FFNN, with the exception of the charging test. Another notable finding is that for the LSTM, cell number 36 consistently has the lowest error, while cell 56 has the highest.

5.3.3 CPU utilisation comparison

The maximum CPU utilisation of the models were tested in accordance with Section 3.4, the results are visible in Table 5.4.

Model	10Hz CPU Utilisation	1Hz CPU Utilisation
Baseline	18.5%	6.6 %
FFNN	19.8% (+1.3%)	6.7% (+0.1%)
LSTM	24.0% (+5.5%)	6.9% (+0.3%)

Table 5.4: CPU utilisation of 1Hz and 10Hz loops for FFNN, LSTM, and Baseline. The baseline is the utilisation of the CPU when the battery management unit is running its existing modules. The FFNN and LSTM are added in isolation in addition to the existing modules that are running on the battery management unit. The utilisation was measured as the maximum utilisation over a 5 minute period. A lower utilisation is better.

From Table 5.4, it is clear that the increase in CPU utilisation of the FFNN is significantly smaller than that of the LSTM. This difference is due to the added number of calculations needed for the LSTM, as well as the more complex activation functions used (sigmoid and tanh instead of ReLU). While a number such as 5.5% may not appear large by itself, it also means that all future features must share this CPU utilisation. These figures suggest that there is no possibility of running the LSTM on all 180 cells at a frequency higher order of magnitude than $1Hz$, as the tenfold utilisation of more than 50% for a single module is not acceptable. Overall, the CPU utilisation of the FFNN is preferable over the LSTM utilisation.

5.4 Major results

The results indicate that, both machine learning models demonstrated a good ability to generalise to the battery behavioural characteristics. These models are also portable and can be implemented on the battery management unit, allowing them to run in a real-time setting within the limitations of the embedded hardware. The LSTM performed better than the FFNN in both the driving and idling scenarios, while the FFNN outperformed the LSTM in the charging scenario.

5.5 Discussion

5.5.1 Test drive discussion

Overall, the FFNN performed better than the LSTM across the entire test drive. However, the proportions of each scenario type during the test could affect the overall results depending on the time spent in each scenario. For example, since the different tests varied in length, the combined results would differ if the distribution of time for driving, charging, and idling changed. Therefore, each scenario should be discussed in isolation. It is also important to note that the models were evaluated based on a single test drive, and the specific data from this drive could influence the conclusions drawn.

In the driving scenario, the LSTM model outperformed the FFNN, indicating that the LSTM's ability to maintain an internal state over time is beneficial for capturing the dynamics of battery cell behaviour during driving. Particularly during large positive or negative voltage spikes, the FFNN tended to overestimate the voltage in either direction. These spikes usually coincide with significant variations in cell current, suggesting that the FFNN might be overly sensitive to this feature. The LSTM's internal state likely helps prevent such overshooting in voltage estimation.

In the charging scenario, the FFNN outperformed the LSTM, suggesting that the simpler architecture of the FFNN is more suitable for modelling the relatively steady and predictable voltage increase during charging. The LSTM exhibited a systematic error with a constant offset, the cause of which is unclear. The charging test was limited in the range of states of charge tested. A longer and more comprehensive evaluation of charging scenarios could provide a more complete picture of performance.

Both models struggled with the idling scenario, but the LSTM showed a slight advantage. This advantage could be due to the internal state introducing some inertia in the estimations, as evidenced by a slight curvature in Figure 5.4. The FFNN's tendency to react too sensitively to large changes in current is also apparent in this scenario.

5.5.2 CPU utilisation discussion

When examining the CPU utilisation, the FFNN demonstrated significantly lower usage compared to the LSTM in both the 10Hz and 1Hz loops. Both models had minimal impact on the 1Hz loop, which is encouraging. However, in the 10Hz loop, the FFNN exhibited much lower CPU utilisation. This indicates that the complexity of the LSTM arises not only from its size but also from a more complex inference process.

Given the comparable performance in the driving test, this suggests that the FFNN has more potential for improvement within the constraints of CPU utilisation. Enhancements could include developing a more complex model with additional layers or incorporating new features to better capture the dynamic behaviour of the current. Although not examined in this thesis, these results also suggest the potential for the models to make inferences at a higher frequency, estimating cell voltages more frequently than once per second.

Chapter 6

Conclusions and Future work

This chapter introduces the conclusions that can be made from the results of the thesis as a whole. This includes the conclusions as well as the limitation with respect to the data and hardware. As well as the further extensions and future work. Lastly the reflection on the social, economical and ethical impacts of the thesis.

6.1 Conclusions

The major conclusions that can be drawn from this thesis are:

- A machine learning approach is a feasible method to model the battery cell characteristics in a battery electric vehicle.
- Such a model can be implemented in embedded C and run on the battery management unit without depleting its resources.

Beyond these major conclusions, some more specific observations can be made. Both the FFNN and LSTM produced comparable results in terms of mean absolute error performance during the test drive. The LSTM performed better in driving and idling scenarios, while the FFNN excelled during charging, as shown in Table 5.3. When adapting these models to run on embedded systems, manually converting the weights and implementing the model inference function was preferable to using code generation tools like ONNX. This approach allowed for better tailoring to the existing systems on the battery management unit and resulted in a smaller footprint compared to a library import. Comparing the Python and C implementations revealed no degradation in performance. However, among the two models explored, the LSTM had a higher maximum CPU utilisation than the FFNN when running inference on the battery management unit, as shown in Table 5.4. Consequently, further exploration into inference at a higher frequency should not consider the LSTM.

6.2 Limitations

There have been limitations throughout this project that can either be addressed with further research or need to be considered when making conclusions about this work. They are explored in each of the following subsections.

6.2.1 Data limitations

The limitations of the project mostly revolve around the data used for training the model. As with many machine learning projects, data often determines the success of a project. In this case, the available data has been very valuable; however, it also posed some limitations that needed to be considered.

6.2.1.1 Data temporal resolution

The update frequency of the data is $1Hz$. This limits the resolution at which the model can both return data and the quality of the input, meaning that the model might not capture finer dynamics within the battery, which ultimately limits the accuracy of the model.

6.2.1.2 Data accuracy and precision

Since the model uses current, temperature, and SOC as features, each of these is limited in accuracy by the hardware used to measure them or the existing models used to estimate the SOC. This means that if these models or measurements are imprecise or inaccurate, it will ultimately affect the results in this thesis.

6.2.1.3 Data generalisation

The data used for training is also created under different conditions than those under which the model is expected to perform. Since the lab data is both measured at a greater precision than in the Scania trucks, the battery cell behaviour may be different. In addition, battery ageing is not considered in this project, a factor which could impact the accuracy of the model when battery cells are older or at a different state of health. Additionally, the model is only trained on lithium nickel manganese cobalt oxide commonly known as NMC battery cells, so there are no guarantees that neither the model nor the method of training said model are effective for different battery chemistries.

6.2.1.4 Model initialisation

The nature of the model is constructed with both a recurrent neural network as well as the exponential decaying inputs. The initial time that the model is running, the outputs are to some extent undefined as the initial conditions are not accurately captured. As a result, the model needs some time to *warm up* before it produces reliable results.

6.2.2 Hardware limitations

When referring to the hardware, this includes everything from the truck to the individual battery management units used in this project. Because these are heavily tailored to the needs of Scania, the findings of this paper are also tailored to the needs of Scania.

6.2.2.1 Truck availability

The testing in a truck setting is very limited in scope. It is costly for Scania to allot a time slot for a truck and to take time away from the engineers to drive the truck. As a result, the test drive is limited in both range and time. The truck was limited to the terrain of the test track as well as the temperature on the day the test was carried out. Consequently, the model was not tested under different temperature and driving conditions, which could provide a more holistic picture of the model's ability to generalise.

6.3 Future work

Since the thesis was limited both by data and by time, there are a few areas that can be further explored within this thesis.

6.3.1 More extensive testing

The real test of the model was limited to a single test run on a real Scania truck. Ideally, the model should be tested under a greater extent of varying conditions. This would include different ambient temperatures, driving characteristics, states of charge, charging at different speeds, and idling.

6.3.2 Accounting for battery health

The model can be further fine-tuned to accommodate ageing batteries, exploring to what extent the ageing of the battery affects the results of this thesis and how it can be accounted for. This would require the acquisition of more data with new features at a high enough resolution.

6.3.3 Alternative model targets

Alternatively, a model can be created to estimate other targets such as the SOC or state of health of the battery cell. This adds the challenge that the SOC and state of health cannot be directly measured, meaning that a model could never outperform existing models of the battery cell SOC and state of health. However, with accurately collected data, the model could perhaps perform better over time compared to an open-loop model. The possibility of this remains to be explored.

6.3.4 Model initialisation

Since recurrent neural networks, by nature, have an internal state, and the inputs with exponential decay are also time-dependent. However, since the histories cannot be known when the model is created, these values are currently initialised to an array of zeroes. Exploring better ways to initialise these values could improve model performance.

6.3.5 Alternative machine learning models

Another avenue worth exploring is alternative machine learning models that could serve as feasible alternatives to the FFNN and LSTM. In particular, given the restrictions of the battery management unit, an ensemble of simpler networks could potentially outperform a single model. Another model that can be explored is the physics-informed neural network, which has shown promising results in modelling physical systems [50], particularly for state of charge and state of health estimation [62]. Additionally, to maximise performance in terms of CPU utilisation, exploring low bit-width integer neural networks [63] might be more suited for the battery management unit.

6.4 Ethical, societal, and sustainability considerations

Beyond the results of the thesis, the greater implications of the research need to be placed within the context of the ethical, social, and environmental impacts.

6.4.1 Ethical considerations

There are some ethical considerations about the safety implications of using software in heavy machinery such as trucks. These considerations need to be taken into account, and systems need to be in place to prevent errors from causing dangerous situations for the driver and the people around them.

Materials in lithium-ion batteries such as lithium and cobalt are known as *conflict minerals*. Together with Volkswagen Group, Scania is part of the *Cobalt for Development* initiative [64]. The initiative aims to improve health and safety conditions and enhance the management of cobalt mining in the Democratic Republic of the Congo [65]. The battery cell supplier Northvolt is part of the *Fair Cobalt Alliance*, a non-government organisation that aims to contribute to local economic development and improve the livelihoods of people in the Democratic Republic of the Congo [66].

6.4.2 UN sustainable development goals

This thesis relates to a number of United Nations Sustainable Development Goals (SDG). In particular, those that relate to sustainable infrastructures and responsible consumption. The following list is a breakdown for each SDG.

- **SDG 7: Affordable and Clean Energy** - This goal aims to ensure access to affordable, reliable, sustainable, and modern energy for all [67]. Enhancing the efficiency and reliability of battery systems in electric vehicles aligns with this goal by promoting the use of clean energy. Research into accurate battery modelling improves performance and longevity of batteries, making electric vehicles more viable and attractive, thereby reducing reliance on fossil fuels and thus contributing to target 7.a [67].

- **SDG 9: Industry, Innovation, and Infrastructure** - This goal focuses on building resilient infrastructure, promoting inclusive and sustainable industrialisation, and fostering innovation [68]. Scania, in particular, is a large supplier of industrial machines such as trucks. This thesis provides new insights into improvements in the electrification of these machines to promote sustainable industrialisation, contributing to target 9.4 of retrofitting industries to make them sustainable.

- **SDG 11: Sustainable Cities and Communities** - This goal is to make cities inclusive, safe, resilient and sustainable. By contributing to the development of more efficient and reliable electric vehicles, this research supports the creation of sustainable urban environments [69]. Electric vehicles are key components of sustainable urban transportation systems as part of target 11.2, and helping to reduce air pollution and improve urban air quality as part of target 11.6.
- **SDG 12: Responsible Consumption and Production** - This goal emphasises sustainable consumption and production patterns [70]. Improved battery management through advanced modelling techniques can ensure that batteries are used more efficiently, giving them a longer lifespan in adherence to target 12.5. This reduces the need for frequent battery replacements, minimising waste and the environmental impact from the reliance on natural resources as per target 12.2.

References

[1] M.-K. Tran, A. Mevawala, S. Panchal, K. Raahemifar, M. Fowler, and R. Fraser, “Effect of integrating the hysteresis component to the equivalent circuit model of Lithium-ion battery for dynamic and non-dynamic applications,” *Journal of Energy Storage*, vol. 32, p. 101 785, Dec. 2020, ISSN: 2352-152X. DOI: 10.1016/j.est.2020.101785.

[2] J. Zhao, H. Ling, J. Liu, J. Wang, A. F. Burke, and Y. Lian, “Machine learning for predicting battery capacity for electric vehicles,” *eTransportation*, vol. 15, p. 100 214, Jan. 2023, ISSN: 2590-1168. DOI: 10.1016/j.etran.2022.100214.

[3] S. K. Kauwe, T. D. Rhone, and T. D. Sparks, “Data-Driven Studies of Li-Ion-Battery Materials,” en, *Crystals*, vol. 9, no. 1, p. 54, Jan. 2019, ISSN: 2073-4352. DOI: 10.3390/cryst9010054.

[4] K. J. Siczek, “Chapter Eight - Negative Electrode (Anode) Materials,” in *Next-Generation Batteries with Sulfur Cathodes*, K. J. Siczek, Ed., Academic Press, Jan. 2019, pp. 117–131, ISBN: 978-0-12-816392-4. DOI: 10.1016/B978-0-12-816392-4.00008-6.

[5] X. Liu, C. Zheng, J. Wu, J. Meng, D.-I. Stroe, and J. Chen, “An Improved State of Charge and State of Power Estimation Method Based on Genetic Particle Filter for Lithium-ion Batteries,” en, *Energies*, vol. 13, no. 2, p. 478, Jan. 2020, ISSN: 1996-1073. DOI: 10.3390/en13020478.

[6] S. Wang *et al.*, “Chapter 7 - Battery state-of-power evaluation methods,” in *Battery System Modeling*, S. Wang *et al.*, Eds., Elsevier, Jan. 2021, pp. 227–254, ISBN: 978-0-323-90472-8. DOI: 10.1016/B978-0-323-90472-8.00004-4.

[7] 14:00-17:00, *ISO 26262-1:2018*, en. [Online]. Available: https://www.iso.org/standard/68383.html (visited on 03/08/2024).

[8] X. Li, D. Yu, V. Søren Byg, and S. Daniel Ioan, “The development of machine learning-based remaining useful life prediction for lithium-ion batteries,” *Journal of Energy Chemistry*, vol. 82, pp. 103–121, Jul. 2023, ISSN: 2095-4956. DOI: 10.1016/j.jechem.2023.03.026.

[9] X. Ding, D. Zhang, J. Cheng, B. Wang, and P. C. K. Luk, “An improved Thevenin model of lithium-ion battery with high accuracy for electric vehicles,” en, *Applied Energy*, vol. 254, p. 113 615, Nov. 2019, ISSN: 03062619. DOI: 10.1016/j.apenergy.2019.113615.

[10] M. Doyle, T. F. Fuller, and J. Newman, “Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell,” en, *Journal of The Electrochemical Society*, vol. 140, no. 6, p. 1526, Jun. 1993, ISSN: 1945-7111. DOI: 10.1149/1.2221597.

[11] A. Paszke *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, 2019. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (visited on 03/04/2024).

[12] *Onnx/onnx*, Mar. 2024. [Online]. Available: https://github.com/onnx/onnx (visited on 03/04/2024).

[13] M. S. Whittingham, “The Origins of the Lithium Battery,” en, Dec. 2019.

[14] M. El Haj Assad, A. Khosravi, M. Malekan, M. A. Rosen, and M. A. Nazari, “Chapter 14 - Energy storage,” in *Design and Performance Optimization of Renewable Energy Systems*, M. E. H. Assad and M. A. Rosen, Eds., Academic Press, Jan. 2021, pp. 205–219, ISBN: 978-0-12-821602-6. DOI: 10.1016/B978-0-12-821602-6.00016-X.

[15] N. Nitta, F. Wu, J. T. Lee, and G. Yushin, “Li-ion battery materials: Present and future,” *Materials Today*, vol. 18, no. 5, pp. 252–264, Jun. 2015, ISSN: 1369-7021. DOI: 10.1016/j.mattod.2014.10.040.

[16] B. Chapman, *How does a lithium-Ion battery work? | Let’s Talk Science*, en, Sep. 2019. [Online]. Available: https://letstalkscience.ca/educational-resources/stem-in-context/how-does-a-lithium-ion-battery-work (visited on 02/26/2024).

[17] M. Bates, *MIT School of Engineering | » How does a battery work?* en-US, Hochreiter, May 2012. [Online]. Available: `https://engineering.mit.edu/engage/ask-an-engineer/how-does-a-battery-work/` (visited on 02/29/2024).

[18] A. Broatch, P. Olmeda, X. Margot, and L. Agizza, "A generalized methodology for lithium-ion cells characterization and lumped electro-thermal modelling," en, *Applied Thermal Engineering*, vol. 217, p. 119 174, Nov. 2022, ISSN: 13594311. DOI: `10.1016/j.applthermaleng.2022.119174`.

[19] R. R. Thakkar, "Electrical Equivalent Circuit Models of Lithium-ion Battery," en, in *Management and Applications of Energy Storage Devices*, IntechOpen, Sep. 2021, ISBN: 978-1-83969-645-9. DOI: `10.5772/intechopen.99851`.

[20] M.-K. Tran *et al.*, "A comprehensive equivalent circuit model for lithium-ion batteries, incorporating the effects of state of health, state of charge, and temperature on model parameters," *Journal of Energy Storage*, vol. 43, p. 103 252, Nov. 2021, ISSN: 2352-152X. DOI: `10.1016/j.est.2021.103252`.

[21] H. He, R. Xiong, and J. Fan, "Evaluation of Lithium-Ion Battery Equivalent Circuit Models for State of Charge Estimation by an Experimental Approach," en, *Energies*, vol. 4, no. 4, pp. 582–598, Apr. 2011, ISSN: 1996-1073. DOI: `10.3390/en4040582`.

[22] Y. Hu and S. Yurkovich, "Battery cell state-of-charge estimation using linear parameter varying system techniques," *Journal of Power Sources*, vol. 198, pp. 338–350, Jan. 2012, ISSN: 0378-7753. DOI: `10.1016/j.jpowsour.2011.09.058`.

[23] Rimsha *et al.*, "State of charge estimation and error analysis of lithium-ion batteries for electric vehicles using Kalman filter and deep neural network," *Journal of Energy Storage*, vol. 72, p. 108 039, Nov. 2023, ISSN: 2352-152X. DOI: `10.1016/j.est.2023.108039`.

[24] M. Danko, J. Adamec, M. Taraba, and P. Drgona, "Overview of batteries State of Charge estimation methods," *Transportation Research Procedia*, TRANSCOM 2019 13th International Scientific Conference on Sustainable, Modern and Safe Transport, vol. 40, pp. 186–192, Jan. 2019, ISSN: 2352-1465. DOI: `10.1016/j.trpro.2019.07.029`.

[25] A. G. Stefanopoulou and Y. Kim, “10 - System-level management of rechargeable lithium-ion batteries,” in *Rechargeable Lithium Batteries*, ser. Woodhead Publishing Series in Energy, A. A. Franco, Ed., Woodhead Publishing, Jan. 2015, pp. 281–302, ISBN: 978-1-78242-090-3. DOI: 10.1016/B978-1-78242-090-3.00010-9.

[26] D. Wang, Y. Yang, and T. Gu, “A hierarchical adaptive extended Kalman filter algorithm for lithium-ion battery state of charge estimation,” *Journal of Energy Storage*, vol. 62, p. 106 831, Jun. 2023, ISSN: 2352-152X. DOI: 10.1016/j.est.2023.106831.

[27] F. N. Dişçi, Y. El-Kahlout, and A. Balıkçı, “Li-ion battery modeling and SOC estimation using extended Kalman filter,” in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, Nov. 2017, pp. 166–169.

[28] C. Li, N. Cui, C. Wang, and C. Zhang, “Simplified electrochemical lithium-ion battery model with variable solid-phase diffusion and parameter identification over wide temperature range,” *Journal of Power Sources*, vol. 497, p. 229 900, Jun. 2021, ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2021.229900.

[29] B. Ng, P. T. Coman, W. E. Mustain, and R. E. White, “Non-destructive parameter extraction for a reduced order lumped electrochemical-thermal model for simulating Li-ion full-cells,” *Journal of Power Sources*, vol. 445, p. 227 296, Jan. 2020, ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2019.227296.

[30] T. M. Mitchell, *Machine learning* (McGraw-Hill series in Computer Science), en, Nachdr. New York: McGraw-Hill, 2013, ISBN: 978-0-07-115467-3.

[31] K. A. Severson *et al.*, “Data-driven prediction of battery cycle life before capacity degradation,” en, *Nature Energy*, vol. 4, no. 5, pp. 383–391, May 2019, ISSN: 2058-7546. DOI: 10.1038/s41560-019-0356-8.

[32] A. Geslin *et al.*, “Selecting the appropriate features in battery lifetime predictions,” *Joule*, vol. 7, no. 9, pp. 1956–1965, Sep. 2023, ISSN: 2542-4351. DOI: 10.1016/j.joule.2023.07.021.

[33] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[34] R. C. Staudemeyer and E. R. Morris, *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*, Sep. 2019. [Online]. Available: `http://arxiv.org/abs/1909.09586` (visited on 02/21/2024).

[35] Z. Kong, Y. Cui, Z. Xia, and H. Lv, "Convolution and Long Short-Term Memory Hybrid Deep Neural Networks for Remaining Useful Life Prognostics," *Applied Sciences*, vol. 9, p. 4156, Oct. 2019. DOI: `10.3390/app9194156`.

[36] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*, Oct. 2014. [Online]. Available: `http://arxiv.org/abs/1409.1259` (visited on 03/13/2024).

[37] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*, Dec. 2014. [Online]. Available: `http://arxiv.org/abs/1412.3555` (visited on 03/13/2024).

[38] Y. Chen, L. Li, W. Li, Q. Guo, Z. Du, and Z. Xu, "Chapter 3 - Deep learning," in *AI Computing Systems*, Y. Chen, L. Li, W. Li, Q. Guo, Z. Du, and Z. Xu, Eds., Morgan Kaufmann, Jan. 2024, pp. 53–121, ISBN: 978-0-323-95399-3. DOI: `10.1016/B978-0-32-395399-3.00009-3`.

[39] S. Heath, "What is an embedded system?" en, in *Embedded Systems Design*, Elsevier, 2002, pp. 1–14, ISBN: 978-0-7506-5546-0. DOI: `10.1016/B978-075065546-0/50002-5`.

[40] R. Zalman, "Chapter 8 - Rugged autonomous vehicles," in *Rugged Embedded Systems*, A. Vega, P. Bose, and A. Buyuktosunoglu, Eds., Boston: Morgan Kaufmann, Jan. 2017, pp. 237–266, ISBN: 978-0-12-802459-1. DOI: `10.1016/B978-0-12-802459-1.00008-7`.

[41] S. Heath, *Embedded systems design*, eng, 2nd ed. Oxford Boston: Newnes, 2003, ISBN: 978-0-7506-5546-0.

[42] J. Henriksson, M. Borg, and C. Englund, "Automotive safety and machine learning: Initial results from a study on how to adapt the ISO 26262 safety standard," en, in *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, vol. May 2018, 2018, pp. 47–49. DOI: `10.1145/3194085.3194090`.

[43] R. Salay, R. Queiroz, and K. Czarnecki, *An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software*, Sep. 2017. DOI: 10.48550/arXiv.1709.02435. [Online]. Available: http://arxiv.org/abs/1709.02435 (visited on 03/08/2024).

[44] *Product Selector*. [Online]. Available: https://www.nxp.com/products/product-selector:PRODUCT-SELECTOR (visited on 03/08/2024).

[45] K. Barrera Llanga, A. Sapena-Bañó, J. Martinez-Roman, and R. Puche-Panadero, "Implementing Deep Learning Models in Embedded Systems for Diagnosis Induction Machine," *International Journal of Electrical and Computer Engineering Research*, vol. 3, pp. 7–12, Mar. 2023. DOI: 10.53375/ijecer.2023.319.

[46] S. Shah, *Microsoft and Facebook's open AI ecosystem gains more support*, en-US, Oct. 2017. [Online]. Available: https://www.engadget.com/2017-10-11-microsoft-facebooks-ai-onxx-partners.html (visited on 03/04/2024).

[47] *C*, en-US. [Online]. Available: https://onnxruntime.ai/docs/get-started/with-c.html (visited on 03/08/2024).

[48] *Kraiskil/onnx2c: Open Neural Network Exchange to C compiler*. [Online]. Available: https://github.com/kraiskil/onnx2c (visited on 03/08/2024).

[49] Y. Choi, S. Ryu, K. Park, and H. Kim, "Machine Learning-Based Lithium-Ion Battery Capacity Estimation Exploiting Multi-Channel Charging Profiles," *IEEE Access*, vol. 7, pp. 75 143–75 152, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2920932.

[50] Y. Zheng and Z. Wu, "Physics-Informed Online Machine Learning and Predictive Control of Nonlinear Processes with Parameter Uncertainty," *Industrial & Engineering Chemistry Research*, vol. 62, no. 6, pp. 2804–2818, Feb. 2023, ISSN: 0888-5885. DOI: 10.1021/acs.iecr.2c03691.

[51] V. Safavi, N. Bazmohammadi, J. C. Vasquez, and J. M. Guerrero, "Battery State-of-Health Estimation: A Step towards Battery Digital Twins," en, *Electronics*, vol. 13, no. 3, p. 587, Jan. 2024, ISSN: 2079-9292. DOI: 10.3390/electronics13030587.

[52] M.-K. Tran *et al.*, “Python-based scikit-learn machine learning models for thermal and electrical performance prediction of high-capacity lithium-ion battery,” en, *International Journal of Energy Research*, vol. 46, no. 2, pp. 786–794, 2022, ISSN: 1099-114X. DOI: 10.1002/er.7202.

[53] X. Sui, S. He, A. Gismero, R. Teodorescu, and D.-I. Stroe, “Robust Fuzzy Entropy-Based SOH Estimation for Different Lithium-Ion Battery Chemistries,” in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, Oct. 2022, pp. 1–8. DOI: 10.1109/ECCE50734.2022.9947792.

[54] P. Venugopal *et al.*, “Analysis of Optimal Machine Learning Approach for Battery Life Estimation of Li-Ion Cell,” *IEEE Access*, vol. 9, pp. 159 616–159 626, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3130994.

[55] L. Zhang, K. Li, D. Du, Y. Guo, M. Fei, and Z. Yang, “A Sparse Learning Machine for Real-Time SOC Estimation of Li-ion Batteries,” *IEEE Access*, vol. 8, pp. 156 165–156 176, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3017774.

[56] S. S. S. Narayanan and S. Thangavel, “Terminal voltage prediction of Li-Ion batteries using Combined Neural Network and Teaching Learning Based Optimization algorithm,” *Applied Soft Computing*, vol. 133, p. 109 954, Jan. 2023, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2022.109954.

[57] M. A. Hannan *et al.*, “Toward Enhanced State of Charge Estimation of Lithium-ion Batteries Using Optimized Machine Learning Techniques,” en, *Scientific Reports*, vol. 10, no. 1, p. 4687, Mar. 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-61464-7.

[58] A. Q. Tameemi, J. Kanesan, and A. S. M. Khairuddin, “Model-based impending lithium battery terminal voltage collapse detection via data-driven and machine learning approaches,” *Journal of Energy Storage*, vol. 86, p. 111 279, May 2024, ISSN: 2352-152X. DOI: 10.1016/j.est.2024.111279.

[59] *Pandas.DataFrame.ewm — pandas 2.2.1 documentation*. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.ewm.html (visited on 03/28/2024).

[60] *INCA Software Products*, en. [Online]. Available: https://www.etas.com/en/products/inca_software_products.php (visited on 05/03/2024).

[61] J. Daoud, "Multicollinearity and Regression Analysis," *Journal of Physics: Conference Series*, vol. 949, p. 012 009, Dec. 2017. DOI: 10.1088/1742-6596/949/1/012009.

[62] S. Singh, Y. E. Ebongue, S. Rezaei, and K. P. Birke, "Hybrid Modeling of Lithium-Ion Battery: Physics-Informed Neural Network for Battery State Estimation," en, *Batteries*, vol. 9, no. 6, p. 301, Jun. 2023, ISSN: 2313-0105. DOI: 10.3390/batteries9060301.

[63] M. Wang, S. Rasoulinezhad, P. H. W. Leong, and H. K. H. So, "NITI: Training Integer Neural Networks Using Integer-only Arithmetic," en, *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3249–3261, Nov. 2022, ISSN: 1045-9219, 1558-2183, 2161-9883. DOI: 10.1109/TPDS.2022.3149787.

[64] *Responsible Raw materials management*, en-SE. [Online]. Available: https://www.scania.com/group/en/home/sustainable-transport/sustainability-at-scania/responsible-business/sustainable-supply-chain/responsible-raw-materials-management.html (visited on 03/07/2024).

[65] C. f. Development, *Cobalt for Development (C4D) - Towards responsible artisanal cobalt mining in the DR Congo*, English. [Online]. Available: https://cobalt4development.com (visited on 03/07/2024).

[66] *Northvolt deepens engagement in the DRC*, en, Jan. 2024. [Online]. Available: https://northvolt.com/articles/northvolt-joins-fca/ (visited on 03/07/2024).

[67] *Goal 7 | Department of Economic and Social Affairs*. [Online]. Available: https://sdgs.un.org/goals/goal7#targets_and_indicators (visited on 06/26/2024).

[68] *Goal 9 | Department of Economic and Social Affairs*. [Online]. Available: https://sdgs.un.org/goals/goal9#targets_and_indicators (visited on 06/26/2024).

[69] *Goal 11 | Department of Economic and Social Affairs*. [Online]. Available: https://sdgs.un.org/goals/goal11#targets_and_indicators (visited on 06/26/2024).

[70] *Goal 12 | Department of Economic and Social Affairs*. [Online]. Available: https://sdgs.un.org/goals/goal12#targets_and_indicators (visited on 06/26/2024).

[71] B. K. Lavine and T. R. Blank, “3.18 - Feed-Forward Neural Networks,” in *Comprehensive Chemometrics*, S. D. Brown, R. Tauler, and B. Walczak, Eds., Oxford: Elsevier, Jan. 2009, pp. 571–586, ISBN: 978-0-444-52701-1. DOI: 10.1016/B978-044452701-1.00026-0.

[72] F. Marini, “3.14 - Neural Networks,” in *Comprehensive Chemometrics*, S. D. Brown, R. Tauler, and B. Walczak, Eds., Oxford: Elsevier, Jan. 2009, pp. 477–505, ISBN: 978-0-444-52701-1. DOI: 10.1016/B978-044452701-1.00128-9.

[73] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” en, *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, ISSN: 0932-4194, 1435-568X. DOI: 10.1007/BF02551274.

[74] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” in *Readings in Cognitive Science*, A. Collins and E. E. Smith, Eds., Morgan Kaufmann, Jan. 1988, pp. 399–421, ISBN: 978-1-4832-1446-7. DOI: 10.1016/B978-1-4832-1446-7.50035-2.

[75] C.-F. Wang, *The Vanishing Gradient Problem*, en, Jan. 2019. [Online]. Available: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484 (visited on 02/28/2024).

Appendix A

Machine learning

A.1 Perceptron

The most basic neural network starts with a single unit called a *perceptron*. The perceptron has a number of inputs, computes the weighted sum of these inputs, applies a *transfer function*, and then outputs the result. The weights ensure that the most salient inputs have a greater contribution to the result. The process of altering the weights to match the desired output more closely is known as *weight training* [71].

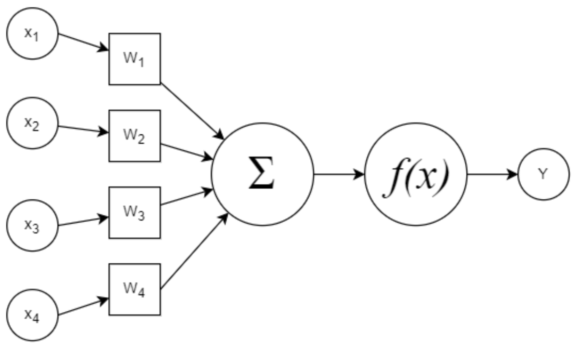


Figure A.1: Perceptron with inputs, weights, transfer function and output.

$$f(\sum_{i=0}^{n} w_i x_i) = f(\hat{w}^T \hat{x}) \qquad \text{(A.1)}$$

In Equation A.1, f represents the transfer function, $\hat{x}$ is the vector of input values, and x_i is the i^{th} value in $\hat{x}$. Likewise, $\hat{w}$ is the vector of weights, and

w_i is the i^{th} weight. The sum of the pairwise multiplication of $\hat{x}$ and $\hat{w}$ is given to the transfer function, which returns the output of the entire node. Some common transfer functions are sigmoid, hyperbolic tangent, linear, and decreasing exponential [72]. There is usually a concept of bias, which is a value derived from the weight that is subtracted from the value before it is entered into the transfer function, but that is omitted for now [72].

The perceptron is limited to the classification of linearly separable datasets. To allow for a model to perform more complex classification and regression, inserting a *hidden layer* of perceptrons between the input and output creates what is called a feedforward neural network (FFNN) [71]. In 1989, it was proven that a network of perceptrons with a single hidden layer is capable of modelling an arbitrary continuous decision boundary [73].

A.2 Feedforward neural network

In a feedforward neural network, the task is to optimise the weights between each layer to achieve the lowest loss when measuring the performance of the model. The output of the first layer serves as the input of the next until the final output layer is reached [72]. A feedforward network can have multiple hidden layers.

A.2.1 Back propagation

Backpropagation is an algorithm to train FFNN first introduced by Rumelhart et al. in 1988 [74]. This approach aims to minimise the value of a loss function when the model's output is compared to the measured voltage. Taking the derivative of the loss function with respect to the model's weights gives the direction of decreasing loss. The weights of all the layers are then nudged in a direction following the gradient of the loss function, a process known as gradient descent [71]. If the derivative of the loss function can be expressed in terms of the loss function itself, it makes computations more efficient.

In general, with the increase of layers and distance between the first layer and the output, the concept of *vanishing gradient* comes into play, where the further away a node is from the output, the less impact it has on the input, and thus the gradient at that point is much smaller [75].

A.3 Recurrent neural network

FFNNs are limited to data being transferred in one direction, akin to a directed acyclic graph. When originally posed by David Rumelhart in 1986, he coined the term recurrent neural network (RNN). An RNN contains connections that can propagate data from earlier events and times to the current processing step. As a result, RNNs have the concept of memory. This allows the network to more easily adapt to time series data since the model itself contains an internal *state* [34].

Appendix B

Detailed feature selection

The feature selection process involved many iteration of training different models in order to determine which features gave the most performance.

From Figure B.1, it is evident that many values have significant correlation, with their coefficient being close to 100. Expectedly, the moving averages of a feature are correlated with each other. Another correlation is between the $dsoc$ and the current i, which is explained by the fact that the SOC is the integral of current with respect to time, and thus the derivative of the SOC should be the current. As a result all features that are derivatives of SOC are not useful. Notably, there seems to be much smaller correlation between temperatures with exponential decay with different values of α compared to other features where the same feature with different values of α have a lot of correlation with each other. Given these results, all features with a correlation greater than 0.85 were removed.

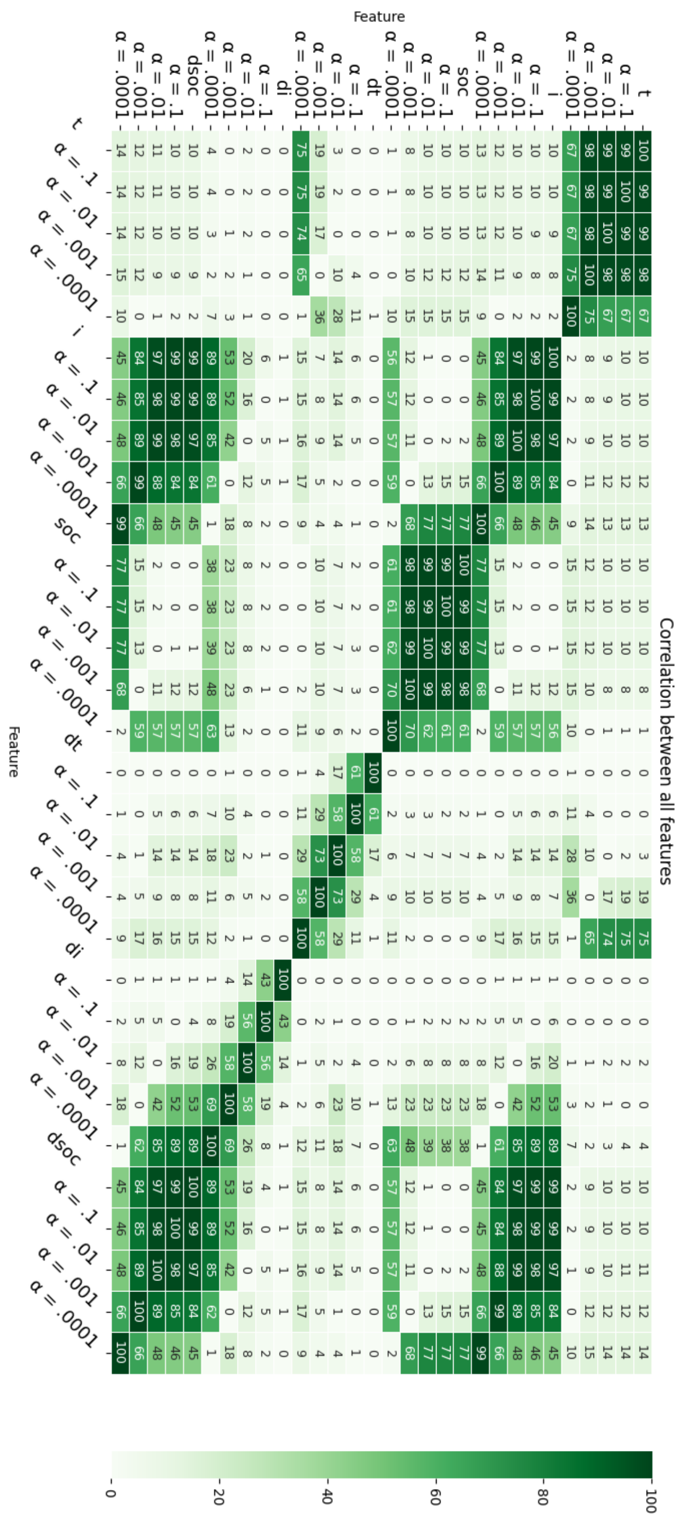


Figure B.1: Heatmap of coefficient of correlation between all features. All coefficients are multiplied by 100 and lies in the range 100-0 instead of 1-0.

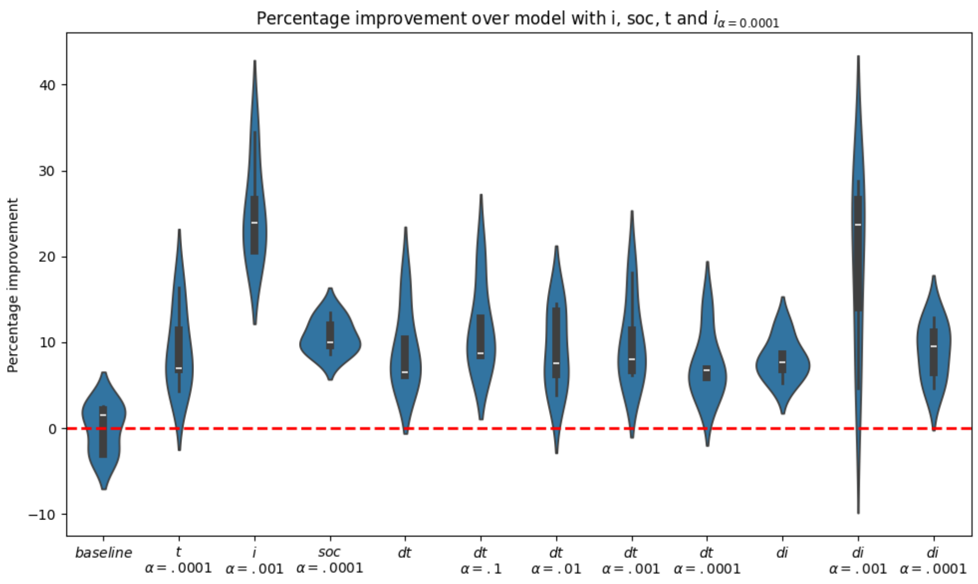


Figure B.2: Violin-plot of features with their corresponding improvement with base features i, t, soc and $i_{\alpha=0.0001}$

After $i_{\alpha=0.0001}$ was added as a feature, each subsequent feature that was added provided a more significant improvement in performance compared to the previous run. The most significant improvements were seen with $i_{\alpha=0.001}$ and $\partial i_{\alpha=0.0001}$, both resulting in a 24% improvement. I chose $i_{\alpha=0.001}$ as a feature since the spread of the results, as visible in Figure B.2, was smaller than for $\partial i_{\alpha=0.0001}$. $i_{\alpha=0.001}$ will provide the model with more information about the medium-term changes in current, compared to $i_{\alpha=0.0001}$. The results after adding $i_{\alpha=0.001}$ as a base feature are shown in Figure B.3.

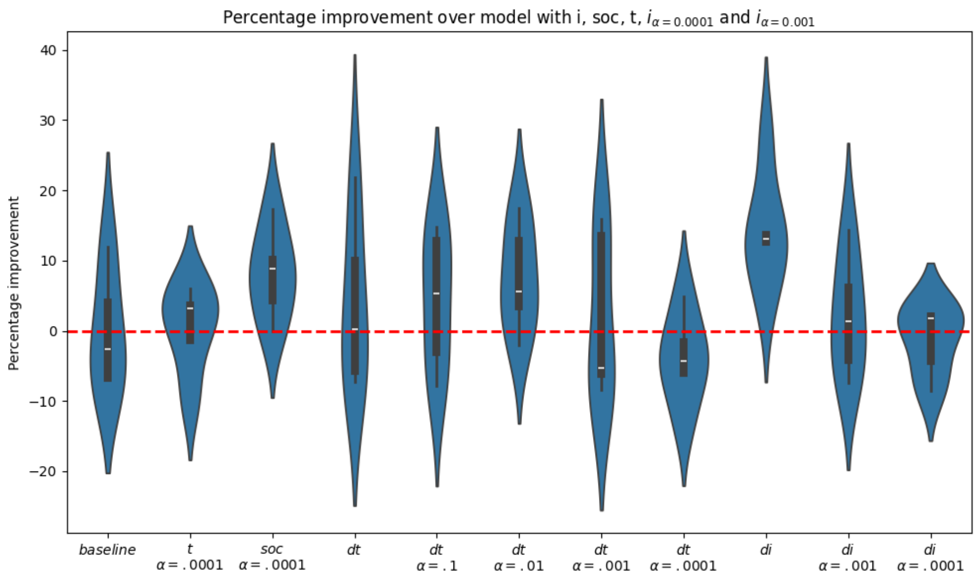


Figure B.3: Violin-plot of features with their corresponding improvement with base features $i, t, soc, i_{\alpha=0.0001}$ and $i_{\alpha=0.001}$

As shown in Figure B.3, the relative improvements with each added feature have begun to decline, with the maximum improvement being only 13%. Adding di will provide the model with information about immediate changes in current draw.

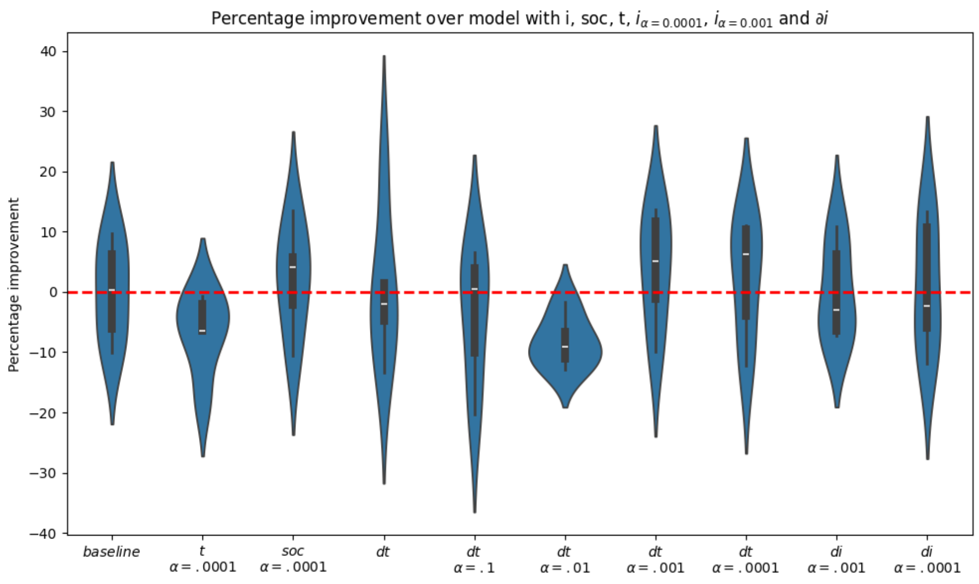


Figure B.4: Heat-map of features with their corresponding improvement with base features $i, t, soc, i_{\alpha=0.0001}, i_{\alpha=0.001}$ and di

After adding di as a feature as shown in Figure B.4, it is visible in the violin plot that the distribution of improvements has an interquartile range that includes zero, suggesting that these improvements lack the statistical significance to be relevant. Thus, we conclude the search for new features. The summary of improvements for each added feature is visible in Figure 4.3.

Appendix C

Model source code

```
import torch.nn as nn
class FFNN3(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(FFNN3, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, output_size)
    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x

class LSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTM, self).__init__()
        self.hidden_size = hidden_size
        self.lstm = nn.LSTM(input_size, hidden_size)
        self.linear = nn.Linear(hidden_size, output_size)
    def forward(self, x, hidden=None):
        if hidden is None:
            h0 = torch.zeros(1, x.size(0), self.hidden_size)
            c0 = torch.zeros(1, x.size(0), self.hidden_size)
            hidden = (h0, c0)

        lstm_out, hidden = self.lstm(x, hidden)
        out = self.linear(lstm_out)
        return out, hidden
```

Listing C.1: Python code of the three layer FFNN and LSTM pytorch classes

TRITA – EECS-EX 2024:0000
Stockholm, Sweden 2024

www.kth.se

€€€€ For DIVA €€€€

{
”Author1”: { ”Last name”: ”Nyberg”,
”First name”: ”Isak”,
”Local User Id”: ”u1zl4y8k”,
”E-mail”: ”isaknyb@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
}
},
”Cycle”: ”2”,
”Course code”: ”DA231X”,
”Credits”: ”30.0”,
”Degree1”: {”Educational program”: ”Degree Programme in Information and Communication Technology”
,”programcode”: ”CINTE”
,”Degree”: ”Degree of Master of Science in Engineering”
,”subjectArea”: ”Computer Science and Engineering”
},
”Title”: {
”Main title”: ”Optimising Battery Cell Dynamics in Electric Vehicles with Embedded Machine Learning”,
”Subtitle”: ”Implementing Real-Time Voltage Modelling in Electric Vehicle Subsystems”,
”Language”: ”eng” },
”Alternative title”: {
”Main title”: ”Optimering av battericelldynamik i elfordon med inbyggd maskininlärning”,
”Subtitle”: ”Implementering av realtidsmodellering av batterispänning i eldrivna fordon”,
”Language”: ”swe”
},
”Supervisor2”: { ”Last name”: ”Bökelund”,
”First name”: ”Björn”,
”E-mail”: ”bjorn.bokelund@scania.com”,
”Other organisation”: ”Scania CV AB”
},
”Supervisor3”: { ”Last name”: ”Lundström”,
”First name”: ”Johan”,
”E-mail”: ”johan.x.lundstrom@scania.com”,
”Other organisation”: ”Scania CV AB”
},
”Examiner1”: { ”Last name”: ”Nordahl”,
”First name”: ”Mats”,
”Local User Id”: ”u1d13i??”,
”E-mail”: ”mnordahl@kth.se”,
”organisation”: {”L1”: ”School of Electrical Engineering and Computer Science”,
”L2”: ”Computer Science” }
},
”Cooperation”: { ”Partner_name”: ”Scania CV AB”},
”National Subject Categories”: ”10201, 10206”,
”Other information”: {”Year”: ”2024”, ”Number of pages”: ”xxiii,91”},
”Copyrightleft”: ”copyright”,
”Series”: { ”Title of series”: ”TRITA – EECS-EX” , ”No. in series”: ”2024:0000” },
”Opponents”: { ”Name”: ”Hugo Dettner Källander”},
”Presentation”: { ”Date”: ”2024-06-25 11:00”
,”Language”:”eng”
,”Room”: ”via Zoom https://kth-se.zoom.us/j/6454199194”
,”Address”: ”Isafjordsgatan 22 (Kistagången 16)”
,”City”: ”Stockholm” },
”Number of lang instances”: ”3”,
”Abstract[eng]”: €€€€

This thesis explores the application of machine learning for modelling battery cell dynamics in battery electric vehicles. The primary objective is to develop and implement machine learning based models, which accurately estimate the terminal voltage of lithium-ion battery cells and are able to run inference in real-time on embedded systems present in battery electric vehicles. Conventional methods, such as the equivalent circuit model, have limitations in handling the complex and dynamic environments encountered in battery electric vehicles. This thesis aims to improve upon these methods by leveraging the capabilities of machine learning in an embedded setting.

The research was conducted in collaboration with Scania CV AB, utilising data from their battery labs and electric trucks. The study involved preprocessing and feature engineering on the data, followed by training various machine learning models, including feedforward neural networks and long short-term memory networks. These models underwent training and evaluation based on their efficacy in interpreting data derived from battery tests conducted in a laboratory setting. The trained machine learning models were then adapted to run on the embedded systems within electric trucks, while considering the limited computational power and memory resources.

Both models were evaluated in a real-world electric truck during driving, charging, and idling scenarios. The long short-term memory network exhibited better performance when driving and idling while, the feedforward neural network performed better during the charging scenario. These findings are valuable as they demonstrate that machine learning models are the feasible for real-time applications in battery electric vehicles. It also highlights a promising area of further research,

particularly for battery chemistries that are not easily modelled by the equivalent circuit model, paving the way for more intelligent, safe, and efficient battery management solutions in electric vehicles.

€€€€,
"Keywords[eng]": €€€€
Battery Electric Vehicles, Embedded Machine Learning, Lithium-Ion Battery Cell Modelling, Equivalent Circuit Model, Real-Time Voltage Modelling €€€€,
"Abstract[swe]": €€€€

Denna avhandling utforskar tillämpningen av maskininlärning för modellering av battericellsdynamik i batteridrivna elfordon. Det primära målet är att utveckla och implementera maskininlärningsbaserade modeller, som uppskattar terminalspänningen i litiumjonbattericeller i realtid på inbyggda system som finns i batteridrivna fordon. Konventionella metoder, såsom ekvivalentkretsmodellen, har begränsningar när det gäller att hantera de komplexa och dynamiska miljöer som finns i batteridrivna fordon. Den här avhandlingen syftar till att förbättra dessa metoder genom att utnyttja möjligheterna med maskininlärning i en inbäddad miljö.

Forskningen genomfördes i samarbete med Scania CV AB och använde data från deras batterilaboratorier och elektriska lastbilar. Studien omfattade förbearbetning och så kallad \textit{feature engineering} av data, följt av träning av olika maskininlärningsmodeller, inklusive feedforward neuronnät och long short-term memory modeller. Dessa modeller genomgick träning och utvärdering baserat på data från batteritester som utförts i laboratoriemiljö. De tränade maskininlärningsmodellerna anpassades sedan för att köras på de inbyggda systemen i eldrivna lastbilar, med hänsyn tagen till den begränsade beräkningskraften och minnesresurserna.

Båda modellerna utvärderades i en eldriven lastbil under körning, laddning och tomgångskörning. Long short-term memory nätverket hade en bättre prestanda vid körning och tomgångskörning, medan feedforward nätverket presterade bättre under laddningsscenariot. Dessa resultat är värdefulla eftersom de visar att maskininlärningsmodeller är användbara för realtidsapplikationer i batteridrivna elfordon. De visar också på ett lovande område för vidare forskning, särskilt för batterikemikalier som inte enkelt kan modelleras med den ekvivalenta kretsmodellen, vilket banar väg för mer intelligenta, säkra och effektiva batterihanteringslösningar i elfordon.

€€€€,
"Keywords[swe]": €€€€
Elektriska Fordon, Inbyggd Maskininlärning, Litiumjonbattericeller Modellering, Ekvivalenta kretsmodell, Realtidsmodellering av Spänning €€€€,
"Abstract[ger]": €€€€

In dieser Arbeit wird die Anwendung des maschinellen Lernens zur Modellierung der Dynamik von Batteriezellen in batteriebetriebenen Elektrofahrzeugen untersucht. Das Hauptziel ist die Entwicklung und Implementierung von auf maschinellem Lernen basierenden Modellen, die die Klemmenspannung von Lithium-Ionen-Batteriezellen in Echtzeit auf eingebetteten Systemen in batteriebetriebenen Elektrofahrzeugen abbilden. Herkömmliche Methoden, wie das Ersatzschaltbildmodell, sind für die komplexen und dynamischen Vorgänge in batteriebetriebenen Fahrzeugen nur bedingt geeignet. Diese Arbeit zielt darauf ab, diese Methoden zu verbessern, indem die Möglichkeiten des maschinellen Lernens in einer eingebetteten Umgebung genutzt werden.

Die Forschung wurde in Zusammenarbeit mit Scania CV AB durchgeführt, wobei Daten aus deren Batterielabors und Elektro-LKWs verwendet wurden. Die Studie umfasste die Vorverarbeitung und das Feature-Engineering der Daten, gefolgt vom Training verschiedener maschineller Lernmodelle, einschließlich neuronaler Feedforward-Netzwerke und Long short-term Memory-Netzwerke. Diese Modelle wurden anhand ihrer Effizienz bei der Interpretation von Daten aus Batterietests in einer Laborumgebung trainiert und bewertet. Die trainierten maschinellen Lernmodelle wurden dann so angepasst, dass sie auf den eingebetteten Systemen in Elektro-LKWs laufen, wobei die begrenzte Rechenleistung und die Speicherressourcen berücksichtigt wurden.

Beide Modelle wurden in einem realen Elektro-Lkw während der Fahrt, beim Aufladen und im Leerlauf getestet. Das Long short-term Memory-Netzwerke zeigte eine bessere Leistung während der Fahrt und im Leerlauf, während das neuronale Feedforward-Netzwerk im Ladeszenario besser abschnitt. Diese Ergebnisse sind wertvoll, da sie zeigen, dass Modelle des maschinellen Lernens für Echtzeitanwendungen in batteriebetriebenen Elektrofahrzeugen geeignet sind. Darüber hinaus wird ein vielversprechender Bereich für weitere Forschungen aufgezeigt, insbesondere für Batteriechemien, die nicht ohne weiteres durch das Ersatzschaltbildmodell modelliert werden können, wodurch der Weg für intelligentere, sicherere und effizientere Batteriemanagementlösungen in Elektrofahrzeugen geebnet wird.

€€€€,
"Keywords[ger]": €€€€
Elektrofahrzeuge, Eingebettetes maschinelles Lernen, Modellierung von Lithium-Ionen-Batteriezellen, Ersatzschaltungsmodell, Echtzeit-Spannungsmodellierung €€€€,
}

acronyms.tex

```
%%% Local Variables:
%%% mode: latex
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
\setabbreviationstyle[acronym]{long-short}
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
%                                       or \newacronym[options]{label}{acronym}{phrase}
% see ”User Manual for glossaries.sty” for the  details about the options, one example is shown below
% note the specification of the long form plural in the line below

\newacronym{KTH}{KTH}{KTH Royal Institute of Technology}
\newacronym{SDG}{SDG}{Sustainable Development Goal}
\newacronym{LSTM}{LSTM}{long short-term memory}
\newacronym{GRU}{GRU}{gated recurrent unit}
\newacronym{ML}{ML}{machine learning}
\newacronym{ECM}{ECM}{equivalent circuit model}
\newacronym{FFNN}{FFNN}{feedforward neural network}
\newacronym{SOC}{SOC}{state of charge}
\newacronym{SOH}{SOH}{state of health}
\newacronym{CPU}{CPU}{central processing unit}
\newacronym{GPU}{GPU}{graphics processing unit}
\newacronym{MSE}{MSE}{mean squared error}
\newacronym{MAE}{MAE}{mean absolute error}
\newacronym{ReLU}{ReLU}{rectified linear unit}
\newacronym{BEV}{BEV}{battery electric vehicle}
\newacronym{BMU}{BMU}{battery management unit}
\newacronym{ONNX}{ONNX}{Open Neural Network Exchange}
\newacronym{AMD}{AMD}{Advanced Micro Devices}
\newacronym{OCV}{OCV}{open circuit voltage}
\newacronym{RNN}{RNN}{recurrent neural network}
\newacronym{INCA}{INCA}{Integrated Calibration and Application Tool}
\newacronym{ISO}{ISO}{International Organisation for Standardisation}
\newacronym{IBM}{IBM}{International Business Machines Corporation}
\newacronym{RAM}{RAM}{random access memory}
\newacronym{ADAM}{ADAM}{Adaptive Moment Estimation}
\newacronym{MDA}{MDA}{Measure Data Analyser}
```