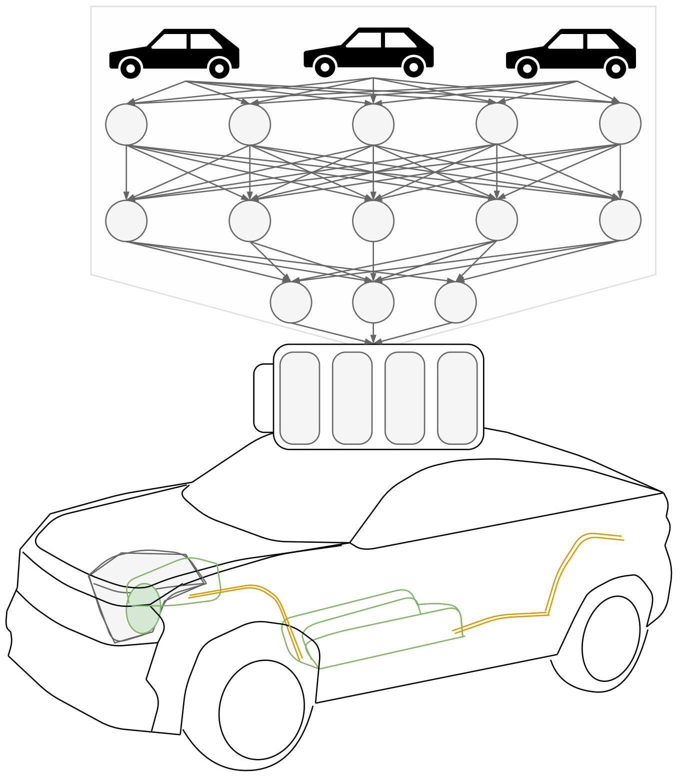
CHALMERS
UNIVERSITY OF TECHNOLOGY



Data-Driven Modelling for Health Estimation of High-Voltage Battery Systems

Master's Thesis in Systems, Control and Mechatronics

LUKAS RAUH

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Data-Driven Modelling for Health Estimation of High-Voltage Battery Systems

LUKAS RAUH

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden

Institute for System Dynamics
University of Stuttgart
Stuttgart, Germany

Data-Driven Modelling for Health Estimation of High-Voltage Battery Systems

LUKAS RAUH

Supervisor:	Sandeep David, China Euro Vehicle Technology (CEVT) AB
	Victor Judez, China Euro Vehicle Technology (CEVT) AB
	Yizhou Zhang, China Euro Vehicle Technology (CEVT) AB
	Patric Skalecki, Institute for System Dynamics, University of Stuttgart
Examiner:	Torsten Wik, Department of Electrical Engineering, Chalmers
	Cristina Tarín, Institute for System Dynamics, University of Stuttgart

Master's Thesis in cooperation of:

Department of Electrical Engineering	Institute for System Dynamics
Chalmers University of Technology	University of Stuttgart
SE-412 96 Gothenburg	DE-70563 Stuttgart
Telephone +46 31 772 1000	Telephone +49 711 68566302

Cover: Schematic illustration of a plug-in hybrid electric vehicle connected with fleet of vehicles through a multi layer perceptron network.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Data-Driven Modelling for Health Estimation of High-Voltage Battery Systems

LUKAS RAUH
Department of Electrical Engineering
Chalmers University of Technology

Abstract

With the introduction of electrified vehicles to the mass car market, battery development has gained momentum and nowadays appears as one of the major challenges of modern, future-oriented vehicle development. An accurate determination of battery health and life prediction is essential to ensure reliable, efficient and durable battery performance along the full lifetime of a vehicle. However, the currently most popular methods for ageing prognosis are either based on fast and imprecise or complex and slow battery models, which generally opens up the gap for data-driven modelling techniques as an efficient and accurate alternative.

This thesis builds on achievements with data-driven modelling to determine the behaviour of complex dynamical systems through machine learning techniques, to meet the demands of the battery lifetime estimation problem. A conducted survey over a wide range of model techniques, from standard baseline up to state-of-the-art approaches, indicates the power and flexibility of data-driven models in the directions of per vehicle and fleet use cases.

Therefore, a basic pipeline capable of handling big data as an input was implemented on diagnostic readout data from a fleet of plug-in hybrid electric vehicles (PHEV). The data was analysed with a feature selection filter to reduce redundancy in the data and improve the portability of the model. It was found that a trained model with sequential long short-term memory (LSTM) and static artificial neural network (ANN) modelling branches performed best for the available data. The mean absolute error (MAE) for a single value state of health (SOH), considered in the percentage scale for the typical SOH range between 100% and 80% SOH, was 0.68 [%-SOH] for the fleet dataset. For the population of data used, the comparison clearly identifies the best results when using mixed rather than pure sequential or static data.

Finally, with a simple usage propagation application, it has been shown that the featured model has potential for further investigation of lifetime prediction use-cases.

Keywords: Data-Driven Modelling, Machine Learning, State of Health Analysis, State of Health Prognosis.

Acknowledgments

This thesis work would not have been possible without the support of several key people. I appreciate their kind support and would therefore like to express my deepest gratitude to all of them at this place.

First, I want to thank my supervisors Sandeep David, Victor Judez and Yizhou Zhang at China Euro Vehicle Technology AB (CEVT) in cooperation with my supervisor and examiner Torsten Wik at Chalmers for the arrangement of this master thesis work opportunity. I would also like to extend my gratitude to my supvervisor Patric Skalecki and examiner Christina Tarín at the University of Stuttgart, as this thesis became a cooperation, due to my double degree masters program. All their support, trust and guidance brought me through this thesis work and helped me to gain new insights into research and practical application of battery technology and data-driven modeling.

I would also like to acknowledge the support and great love of my family, my girlfriend, my friends and all the great people at CEVT, who have not been mentioned so far. They have kept me going and made this thesis possible with their support.

Lukas Rauh, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1

Introduction

In the transport sector, the predicted demand for electrified mobility is enormous. Within the next 20 years significant changes in technology, long-term decarbonisation targets and the competition to build the next clusters of high-value future-oriented industry will reshape the worldwide automotive and freight market [1].

Although the electrified global market share is still small (about 2.5% of new car registrations in 2019), the annual growth has picked up speed, mainly driven by government support, fleet emission restrictions, infrastructure investments and increase in cost benefits. This leads to growing familiarity and willingness on the customer side to consider electrified vehicles in the purchasing decision [2, 3]. According to a study of Bloomberg New Energy Finance (BNEF) research organization this will lead to a fast growth of annual electrified vehicles (EV) sales in the next 20 years, as illustrated by the sharp increase of annual sales market share in Figure 1.1 [1].

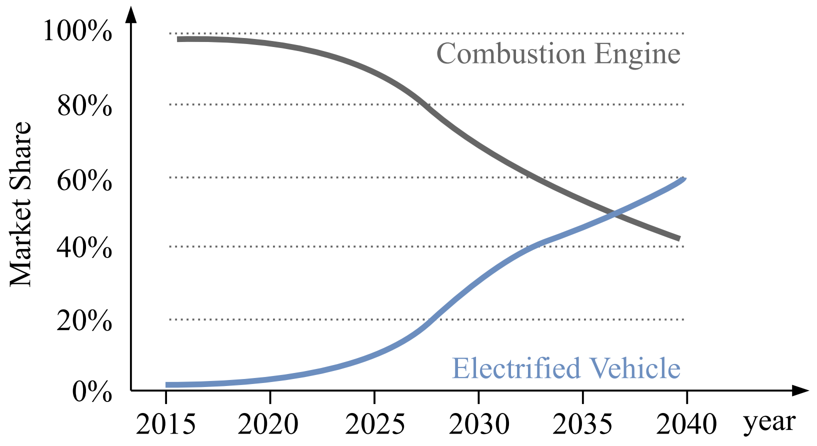


Figure 1.1: Annual market share forecast of combustion engine and electrified vehicles, adapted from [1].

The forecast expects a burgeoning market, creating enormous opportunities for the automotive and supplying industry. On the other hand, despite this large-scale growth there are uncertainties on how car manufacturers will face the fundamental problem of battery degradation towards sustainable lifetime performance. This major challenge consists of balancing the trade-off between usable energy, power, lifetime and cost. Currently, lithium-ion batteries are the main battery type for vehicle applications, thanks to their flexibility and performance in this trade-off. However, one main drawback relates to their degradation occurring during the lifetime, whether the battery is used or not, starting on the date it is produced. Due to the cross-dependence of degradation factors plus the dependency on the individual battery environment and use, the ageing phenomena are complicated to determine, characterize and optimize towards health-conscious future usage. Thus, much of the attention in battery technology research has been focused on health management and prognostics, to face the complexity of the degradation phenomena and provide insights into how the batteries actually should be used to achieve long lifetime without sacrificing performance [4, 5].

An integral part of battery health management and prognosis is the battery model. The currently most studied types of lithium-ion battery models in the literature for predicting battery characteristics are the equivalent circuit (ECM) and physics-based models (PBM). However, both models are limited, either in prediction accuracy or performance. For this reason, data-driven models (DDM) are a promising opportunity for efficient and accurate battery modelling, due to recent achievements in modelling complex dynamic systems. Figure 1.2 arranges the potential placing of DDMs through expected attributes compared to ECMs and PBMs [4, 6].

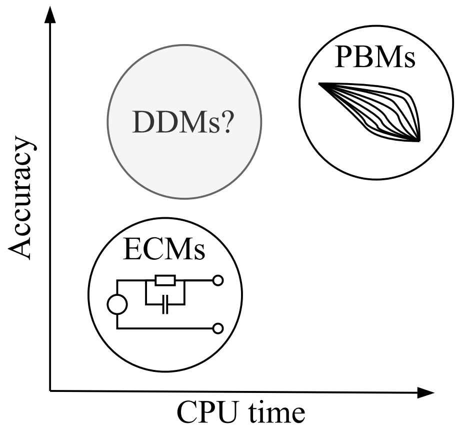


Figure 1.2: Illustration of model accuracy over CPU time for ECM, PBM and DDM models for predicting battery characteristics, adapted from [6].

ECMs are currently the major models used in vehicle Battery Management Systems (BMS), due to their computational efficiency which enables real time battery behaviour prediction even on performance-limited electrical control units (ECU) in the vehicles. The majority of ECM models are derived by empirical modelling and are represented as electronic circuit models consisting of a network of capacitors and resistors. Their required circuit model parameters are based on battery experiments in laboratory conditions, which is why attaining high accuracy in the individual vehicle remains a challenge, especially for predictive tasks [6]. In contrast, PBM has been developed to give detailed insights on the battery internal dynamics, such as lithium-ion diffusion or electro-chemical kinetics. However, this detail in the modelling is associated with a high computational effort, since PBMs are generally described by sets of partial differential equations (PDE). Hence, an implementation of the PBM for a real-time application is generally impossible due to the performance limitations in the BMS controller [6, 7].

1.1 Related Work

With the emerging success of DDMs in modelling dynamic systems, studies have been considered to facilitate the battery ageing problem under different aspects, i.e. considering varying time scales or data input. Most of these studies have in common that they build on the investigation of the State of Health (SOH) of a battery. The SOH is a figure of merit that rates the battery condition compared to an ideal, factor-new condition state. While there is no clear definition of how the SOH is determined, an assessment of capacity fluctuation or the increase in internal resistance has become widely established in science and industry.

SOH estimation and prognosis has been demonstrated with promising results in accuracy by using different machine learning algorithms. Most of the algorithms can be categorized into: rather simple and lightweight models as regression and their regularized extensions (as in Severson et al. [8]); ensemble decision tree methods as random forest regression (as in Mansouri et al. [9]); support vector machines (as in Nuhic et al. [10]) and neural networks (as in Ren et al. [11]), which are more expensive to train. Each of these algorithms has its own merits and challenges and a selection of the most appropriate approaches is a multi-faced problem, depending on the demands for the specific application. Some examples of factors to be taken into account are the type and amount of data available, the desired quality of results and the physical interpretability of the trained model. In this work the problem is addressed for the demands of an application in cooperation with China Euro Vehicle Technology AB (CEVT), as a company developing electrification of vehicles, by utilizing the available diagnostic readout data to train the data-driven models.

1.2 Goals

This thesis explores supervised learning approaches for battery SOH estimation and prediction, that can be used as an alternative to the ECM and PBM models. The goal is to define and build a modelling structure, that is capable of handling big data from a fleet as input and deliver the desired battery parameter value predictions as an output. Therefore, the thesis attempts to compare a wide range of modelling methods and identify the best suitable model that meets the demands of this application. Besides linear baseline models for the reference, machine learning model types were selected for a single value and sequential regression task. The considered model types are artificial neural networks (ANN), long short-term memory (LSTM) and temporal convolutional networks (TCN), as well as combined model architectures.

The work involves following key steps: As a starting point the available data is analysed to find an appropriate feature extraction that can be applied on the big data pool. With this data understanding, the feature extraction can then be implemented as the first step in a flexible tool chain, which allows the replacement of data preparation steps, modelling architectures or evaluation methods. For each model type the model architecture is implemented with a machine learning framework. This allows the optimization of model hyper-parameters with integrated tools to maximize the training metric performance per model. The final step is to find the most suitable model candidate by comparing the performance of the model metrics and visualizing the prediction errors in order to draw conclusions for further work.

1.3 Thesis Outline

The remaining content of this thesis is structured as follows: In Chapter 2, the theoretical concepts of battery health and data-driven models are described, from the underlying statistical framework up to state-of-the-art models. The data gathering and collection as starting point for this thesis are shown in Chapter 3. This includes the feature extraction of the raw data sources as well as the definition of the input and target for the modelling task. Results of the data exploration motivates the selection of modelling approach and implementations in Chapter 4. Results obtained for the SOH diagnosis, a further investigation of data reduction in the context of a performance-limited in-car application and an attempt to predict future SOH of a vehicle are presented and discussed in Chapter 5. Finally, Chapter 6 summaries the thesis findings, delivers final conclusions and provides an outlook on further research possibilities.

2

Theoretical Concepts

In this chapter theoretical concepts used in this thesis work are reviewed. This chapter intends to build a basic understanding of battery degradation and data-driven models, through a brief introduction from basic concepts up to state of the art models.

2.1 Battery Health Principles

This section gives a brief introduction on the field of battery lifetime research. First, different battery health condition indicators are listed and how they contribute to lifetime analysis. In the end, battery degradation mechanisms are reviewed to understand battery ageing.

2.1.1 Battery Health Condition Indicators

Throughout literature and industry, two standard indicators have been established to evaluate the health of a battery system, either on battery pack or on cell level.

The *State of Charge* (SOC) is an indicator of available energy in a battery system. It measures the remaining energy which is available for use, compared to the energy which was available after being fully charged. Therefore, SOC is defined as

$$\mathrm{SOC} = \frac{C_{\mathrm{curr}}}{C_{\mathrm{full}}} \times 100\%, \tag{2.1}$$

i.e. the relation between the currently available capacity C_{curr} and the capacity available when fully charged C_{full}, in percent, each for either pack or cell values. In this context, the fully available capacity is usually referred to as the capacity reached during the last completed charging of the battery. The SOC indicates how much energy can be used from the battery system until it needs to be recharged and therefore acts like a fuel level indicator in combustion drive systems [6, 12].

The *State of Health* (SOH) indicates the durability of a battery system, comparing the current battery system health to a factory-new battery state.

There is no fixed definition for determining SOH as an indicator of the gradual deterioration of battery performance due to irreversible chemical reactions. For this reason, each battery manufacturer can decide individually on the assessment when developing a battery management system. However, the determination of the SOH has been addressed in studies evaluating the significance of various performance parameters in changing with age. In a broad consensus, capacity decrease and internal resistance increase appear as qualified physical quantities for the EV application. Both quantities are used to address different long term objectives. In the case of resistance increase, the objective is power density. Hence it is used mainly to determine SOH in applications where delivering a specific power level is most important, e.g. for accelerating a high-performance EV or for the starter battery of a combustion engine. In contrast, for the determination in range or lifetime oriented applications the capacity decrease is used for the SOH assessment as

$$\mathrm{SOH} = \frac{C_{\mathrm{full}}}{C_{\mathrm{nom}}} \times 100\%, \tag{2.2}$$

between the current fully available capacity C_{full} and the factory-new nominal capacity C_{nom} in percent. This capacity based SOH determination is used in the thesis [6].

2.1.2 Battery Lifetime Indicators

Remaining Useful Life (RUL) is a second concept to describe the ageing of a battery system. It is closely related to the concept of SOH as battery system health indicator. RUL as concept has emerged from the paradigm of predictive maintenance and describes the value of remaining life, either in time or use-case specific domain (i.e. remaining battery cycles), until a failure event occurs. In the context of predictive maintenance this is mainly connected to the failure or breakdown of machine parts and thus followed by downtime of the defective machine. As a result of the failure occurred, the part often is broken or no longer usable, which is why this point in time is referred as End of Life (EOL), and the time remaining until as the remaining useful life. Compared to the predictive maintenance context, the EOL for a battery system is commonly referred to the moment where the battery system has lost 20% of the nominal total factory-new capacity, or in the context of SOH, where the SOH falls below a threshold value of 80%. Figure 2.1 illustrates this SOH and RUL relationship, for an arbitrary point in time t between the begin of battery live, referred as Begin of Life (BOL), and the EOL. The SOH curve in this figure only represents exemplary SOH course. However, the SOH most likely drops in the beginning of the battery life and gains negative momentum towards its end.

In predictive maintenance the result of a reliable RUL estimation is used to optimize maintenance schedules in order to reduce downtime and decrease maintenance cost. According to the same principle, a RUL and SOH estimation and prediction can help preventing battery malfunction or used as assessment tool to rate lifetime expectations of potential new battery generations during development.

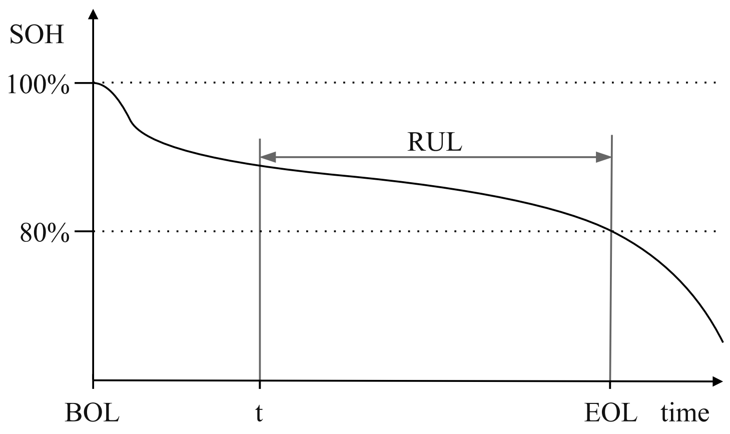


Figure 2.1: Illustration of the RUL lifetime indicator on the SOH health indicator for an arbitrary time t in the life of a battery.

2.1.3 Battery Ageing Degradation

Battery systems, due to the complex chemical composition and physical processes inside the battery cells, are complex non-linear dynamic systems. During usage many different ageing effects occur, interfere with each other and degrade the battery depending on the usage and surrounding environment [5]. Identifying the ageing mechanisms, understanding the causes and the optimizing battery technology or substance composition is the main task in battery cell development. Battery technology optimization is related to balance the trade-off between energy density, power density, lifetime and cost. Lithium-ion has stand out in this development as an efficient, high-power and durable choice among other cell chemistry composition types [13].

Nevertheless, the ageing process of lithium-ion batteries is complex and remains the major challenge for sustainable lifetime performance. In battery related studies, the ageing phenomenon is typically split into two types: *calendar* and *cycling ageing*. Both ageing types coexist and interact in the electrical vehicle application. Cycling ageing occurs when the battery is in active use, including charge and discharging. Moreover, calendar ageing occurs even when the battery is in rest, due to the innate chemical reaction between negative electrode and electrolyte. In other words, calendar ageing appears continuously in the battery, starting from the time it leaves the production.

The severity of the ageing is in the first instance dependent on the relationship between battery design and battery usage. In addition to a steady ageing caused by the design of the battery, there are additional factors that can dramatically accelerate the inevitable ageing of the battery. The so-called *battery stress factors* vary with the type of ageing.

In the case of calendar ageing, the main accelerating stress factors include:

- storage of the battery fully charged or discharged for long periods

- storage of the battery at high temperature.

On the other hand, for cycling ageing main stress factors include:

- high charging and discharging current rates
- charging the battery at low temperature
- over-charging or over-discharging beyond the capacity limits
- cycling the battery with large unloading volume, the so-called *depth of discharge* (DOD).

The fact that these factors amplify each other and apply to each cell individually makes the ageing process of a full battery pack considerably more complicated for the battery management system. To this end, a fine tuning of the battery operating conditions through adapting current regulations, charging speed and thermal management can contribute to extending the durability and ensuring safe operation [5, 14].

In summary, the ageing characteristic of lithium-ion batteries is complex and includes treatment and the environment in which it is operated. Correspondingly, the main ageing and stress factors are contained by an observation of overall use, charge levels, charging and discharging speed, depth of discharge and battery temperature. To this end, the objective for the feature extraction presented in Chapter 3 is to focus the extraction to ensure that these conditions are well reflected in the data used for modelling.

2.2 Statistical Learning Theory

Statistical learning theory provides the framework for machine learning. As a subcategory of artificial intelligence, machine learning refers to the method of analyzing data observations in order to automatically build a numerical model that generalizes learned knowledge from the data. For this purpose, machine learning algorithms are based on statistical principles and can therefore be seen as part of the statistical learning framework.

The theory of the framework draws from the field of statistics and functional analysis and dates back to the late 1960's. In the early days, until 1990, statistical learning was a purely theoretical analysis of function estimation for small scopes of data, since the computational effort was too heavy for the computers at that time. In the advent of new algorithms and more computational performance this changed and statistical learning became applicable to problems in practice. Popular examples for statistical learning in practice are image, speech or pattern recognition, fault or disease detection in medicine and parameter identification [15, 16, 17].

2.2.1 The Learning Framework

The main goal of the statistical learning framework is to find a mapping model constructed on observed data, which can then be used for unseen data to make predictions. The framework assumes the representation of such a mapping model as a function. Therefore, let $\boldsymbol{x} \in \mathbb{R}^d$ be a random number input vector of dimension d and $y \in \mathbb{R}$ a random output variable, jointly connected by the probability distribution $P(x, y)$. Then the task is to find the mapping function f that best predicts y for a given $\boldsymbol{x}$, in the form of

$$y = f(\boldsymbol{x}). \tag{2.3}$$

This search process to find the best possible function is called *training* or *learning.* In order to describe the process of learning and define an appropriate modelling problem, three components have to be considered: representation, evaluation and optimization. This division into three components also helps to find a suitable methods and algorithms for a given problem and guides the way through the related project [15, 18].

In the following sections, the three components are presented in detail.

2.2.2 Supervised Learning (Representation)

The representation of the modelling problem is the initial step in the statistical learning framework. It addresses the representation of the modelling problem, which forms the space of allowed models, referred to as *hypothesis space.* This hypothesis space shrinks the total number of available machine learning models to a smaller number of feasible models for the project under investigation. If a model type is not in the hypothesis space it is not considered in the further steps.

In this thesis the representation is a *regression* problem, mapping numerical inputs to numerical outputs. In contrast, *classification* is another common type of problem in machine learning, which refers to the mapping of numerical inputs to different classes. Regression is part of the *supervised* learning field. The goal in supervised learning is to predict the value of an output based on a number of input output measurement pairs. In contrast, in *unsupervised* learning, there are no output measurements. The goal is to detect associations or patterns among the set of input measurements and distinguish relevant information from structure-less noise in the input.

The theory of supervised regression learning defines the learning of a function

$$f : X \longrightarrow Y \tag{2.4}$$

as an element of the hypothesis space F, mapping the *feature space* X as input to the *target space* Y as output, based on the basic statistical learning equation (Equation 2.3). Therefore, the data is regarded as a number of l independent and identically distributed pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, so-called *samples*, such that each input feature vector $\boldsymbol{x}_i$ corresponds to the *target* vector $\boldsymbol{y}_i$. The total set of available data pairs $D = (x_1, y_1), ..., (x_l, y_l)$ is called *dataset.*

2.2.3 Objective Function (Evaluation)

A measurement of the accuracy is required, to find which function f out of the hypothesis space F best predicts the output vector $\boldsymbol{y}_i$ for a given input vector $\boldsymbol{x}_i$. The goal is to find the function obtaining the minimal error between the predicted value $\hat{y}_i$ of the true output $\boldsymbol{y}_i$ for the same $\boldsymbol{x}_i$. For this reason, an error measuring function is defined, called *objective* or *loss function.* In supervised learning the loss of a predicted value $\hat{y}_i$ of a function f for a given data sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is defined as

$$L(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = L\Big(\boldsymbol{y}_i, f(\boldsymbol{x}_i)\Big). \tag{2.5}$$

Accordingly, the best function f out of F is the function which obtains the minimal loss for all available data samples. For a number of l independent and identically distributed samples this extends Equation 2.5 to

$$L(f) = \frac{1}{l}\sum_{i=1}^{l}\Big(L(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)\Big), \tag{2.6}$$

by averaging the loss function over the data. In the literature, this loss $L(f)$ is often referred as *empirical risk* $R_{emp}(f)$ of the function f, following the idea that an actual performance of the function is uncertain in practice. It is only possible to evaluate the function on the observed data samples, leading to the name of empirical risk.

A common choice for a loss function in regression problems is the Mean Squared Error (MSE), which is defined as

$$MSE = \frac{1}{l}\sum_{i=1}^{l}\Big(||\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i||\Big)^2, \tag{2.7}$$

where $||\cdot||$ signifies the ℓ_2-norm between the true vector output $\boldsymbol{y}_i$ and prediction vector $\hat{\boldsymbol{y}}_i$ for each of l given data samples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ [19].

2.2.4 Gradient Descent (Optimization)

The last component of the learning framework and optimization is required to improve the result of the objective function. Therefore, the parameter vector $\boldsymbol{\theta}$ is introduced into the mapping function model f as

$$y = f(x|\boldsymbol{\theta}). \tag{2.8}$$

For instance, a simple mapping function model choice with parameters is the linear function

$$y = mx + b \tag{2.9}$$

where m is the slope and b marks the intercept, often referred as *bias* in the machine learning context. Hence, the parameter vector $\boldsymbol{\theta}$ contains the two parameters $\{m, b\}$.

With this definition of the mapping function the function loss also becomes dependent on the parameters. A frequently used method to decrease the loss due to the choice of parameters is to update the parameters with *gradient descent*. Gradient descent is an update procedure to the parameters in the model to be changed with small incremental steps in the direction of the optimal parameter configuration. For this purpose, the loss function L must be differentiable. The update formula for the model parameters $\boldsymbol{\theta}^{(t)}$ at time t is given by

$$\boldsymbol{\theta}^{(t)} \longleftarrow \boldsymbol{\theta}^{(t-1)} - \lambda \nabla_{\boldsymbol{\theta}} \Big(L\Big(\boldsymbol{y}_i, f(\boldsymbol{x}_i | \boldsymbol{\theta}^{(t-1)})\Big)\Big), \tag{2.10}$$

where ∇_θ is the parameter gradient of the loss function L for the prediction of the model given the parameters $\boldsymbol{\theta}^{(t-1)}$ of the last time step. Therefore, the step size for each update step is scaled with the *learning rate* λ.

This procedure can be illustrated intuitively in a simple toy example with a single parameter θ_1, which is presented in Figure 2.2. The loss function depending on the parameter θ_1 is then given as curve $L(\theta_1)$. The parameter update of the parameter θ_1 at time t, represented as grey ball on the curve, corresponds to the movement of the ball to the optimal parameter value θ_1^* with the minimum achievable loss. For this reason, the ball moves in the negative direction of the gradient with a step size according to the learning rate λ.

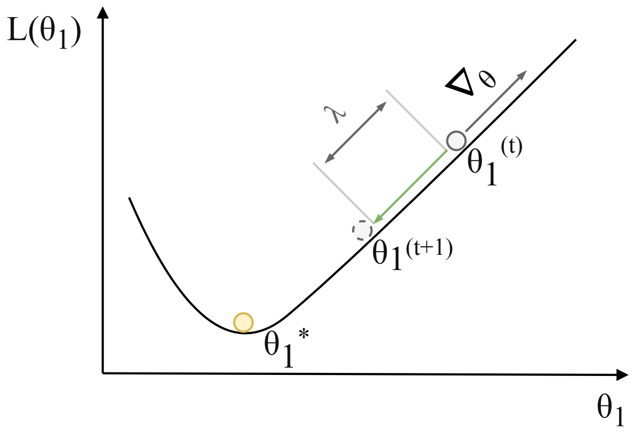


Figure 2.2: Illustration of a gradient descent parameter update.

This makes the learning rate an important hyper-parameter during the learning process. A small learning rate λ leads to a stable but slow learning process. A too high learning rate can lead to an overshooting update, such that the optimal parameter value can never be reached or the training becomes unstable. This is illustrated in Figure 2.3, where the optimization of the parameter θ_1 is visualized. The random initialized value for the parameter is marked with the grey dot and the optimal, minimal value is marked with the yellow dot. The blue arrows visualize a small learning rate. Hence the optimal parameter value can be found but it takes a lot of update steps. The red arrows visualize a too high learning rate. In this example the optimum can never be reached, because the gradient is overshooting around the optimum. The green arrows show a adaptive decreasing learning rate, which leads to optimum in fewer steps than the blue iterations.

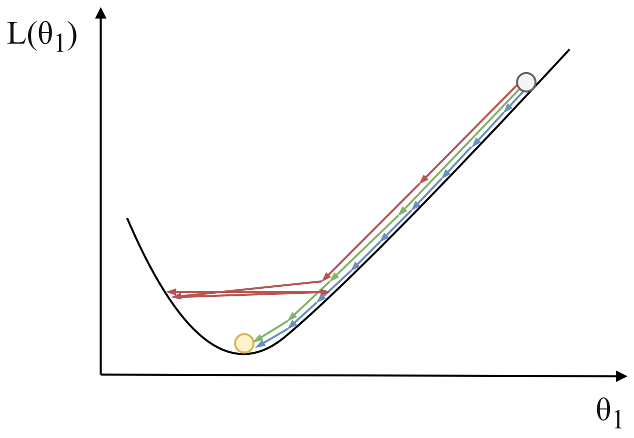


Figure 2.3: Illustration of a too high (red), too small (blue) and adaptive (green) learning rate for gradient descent parameter update.

Following this learning rate adaption, several learning rate optimization methods have been developed based on the gradient descent principle. Some common examples are RMSprop, ADAM, NADAM [20], whereby there is no ultimate best of the algorithms, hence the choice of the algorithm is usually made on a case-by-case basis or identified by hyper-parameter optimization.

2.2.5 Generalization

Generalization in statistical learning refers to the ability of a model to predict properly to new, previously unseen data. The goal is to get similar prediction performance, for example in the sense of prediction errors, for the unseen data as for the training data. If this is the case, then the model is said to have generalized well.

In order to evaluate the generalization of a model with a limited data set, usually the full available data is split into two subsets before training. One subset is used for the training of the model and one for the evaluation. Accordingly, the subsets are often referred as *train dataset* and *test dataset*. A common split rate is 80%/20%, where the former value is used for the training and the latter for the testing.

Splitting the data is usually done randomly to ensure that similar data is represented in both subsets. However, it is still possible that certain imbalances occur. Training on imbalanced datasets is one of the main reasons for *overfitted* models. A model is said to overfit, if it tends to predict significantly worse on unseen data than on data that it was trained on. This effect occurs when the model adapts explicitly on the training data and does not generalise sufficiently. An other reason for this can be that the model is overly complex. The model then rather learns input-output combinations than an underlying relation. In contrast to overfitting, *underfitting* occurs if the chosen model complexity is too low or was not trained enough, resulting in high errors with a low variance. The relation between prediction error and model complexity is illustrated in following Figure 2.4. This clearly indicates the sweet spot between underfitting and overfitting as the goal of the training process.

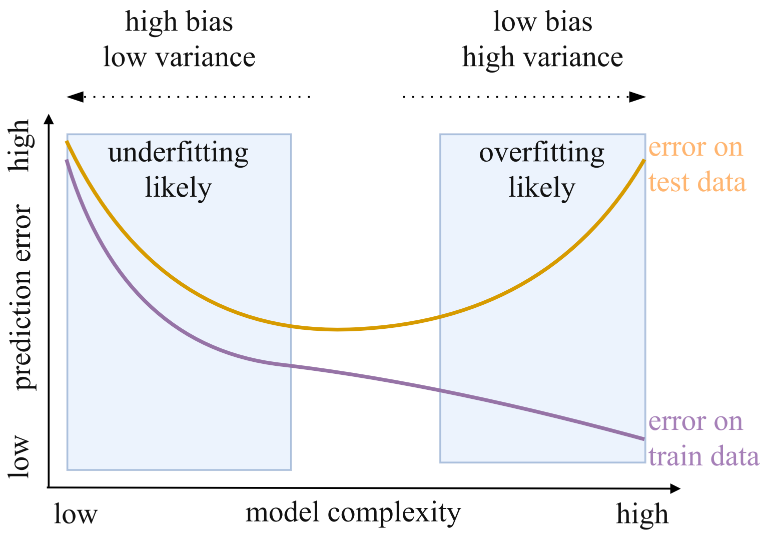


Figure 2.4: Illustration of model complexity and prediction error defining potential underfitting and overfitting ranges.

2.3 Data-Driven Modelling Theory

Following the framework of statistical learning from the previous section, this section introduces relevant theory for the specific network implementations used in this thesis. The section portrays theory and some background on how models evolved to their current technical state of the art.

2.3.1 Neural Network Fundamentals

One type of statistical learning model is *Artificial Neural Networks* (ANN), often just referred as a generic Neural Networks (NN) type. The model is inspired by the architecture of the human brain and is designed to solve a problem in the way the human brain would solve it. Therefore, an ANN is a collection of interconnected neurons, where each neuron is a simplified model of the neural cell in a biological brain.

Neuron

A *neuron* in an ANN has been conceived as a model of the neural cell in a human brain. Based on the biological archetype a neuron is triggered when signals above a certain threshold are applied to its receptors. The activated neuron then fires a signal via its output which in turn can activate other neurons.

An illustration of a neuron is given in Figure 2.5, where n inputs $x_1, ..., x_n$ are weighted by associated weights $w_1, ..., w_n$ for every input branch. In the neuron node the weighted inputs are summed up including the node bias b. This results in a single value, which is then feed into the activation function ϕ to get the output value y of the neuron.

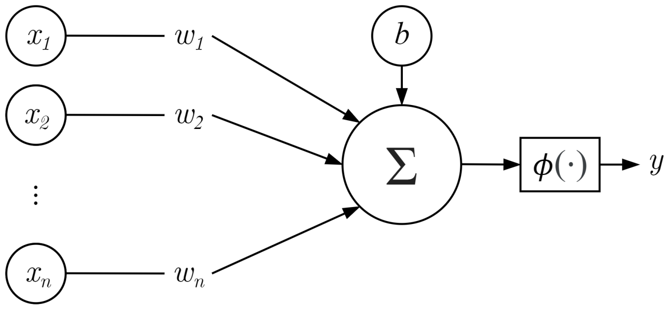


Figure 2.5: Illustration of the computational model of a neuron.

Following the illustration the output of a neuron is given by

$$y = \phi\Big(\sum_{i=0}^{n} w_i x_i + b\Big). \tag{2.11}$$

Activation Function

The activation function ϕ of a neuron acts as a mathematical gate in between the neuron state and its output to other neurons. As an enclosing function the activation function has an influence on the shape of the neuron output. For example, the basic Heaviside step function, defined as

$$\mathcal{H}(n) = \begin{cases} 0, & n < 0, \\ 1, & n \geq 0, \end{cases} \tag{2.12}$$

can be used to switch on and off the neuron output, depending on a rule or threshold. This is an abstraction of the activation potential firing between neural cells.

Besides the binary output of the Heaviside function a popular choice, especially for deep neural networks, is the rectified linear unit function (ReLU), which is defined as

$$\phi_{\text{ReLU}}(n) = max(0, n). \tag{2.13}$$

ReLU is a piece-wise linear function that will directly output a positive neuron result, otherwise, it will output zero. ReLU has become the standard activation function amongst others in many machine learning implementations, thanks to its simplicity, consistency and effectiveness. Alternatives have not managed to replace it, although extensions of ReLU such as the so-called leaky or gated variant may outperform the standard variant in some use cases. [21].

Multi Layer Perceptron
From Equation 2.11 it can be seen that a neuron is only capable of representing a linear relation between inputs and outputs. To extend this to the representation of non-linear relations, multiple neurons are stacked together in a so-called *layer*. According to the single neuron model each neuron in a layer receives the same inputs but has its individual parameters (weights and bias). The layer output is a concatenation of the neuron outputs, so a layer with n neurons will deliver an n-dimensional output. By stacking multiple layers sequentially, a so-called *Multi Layer Perceptron* (MLP), it is possible to train a network to perform much more complex function approximations than a single neuron perceptron.

In order to pass data through a multi layer network three types of layers are required: input, hidden and output layers. Naturally, input and output layers are required to put data in and take data out of the network. Hidden Layer refers to any layer between the input and output of a network. Figure 2.6 illustrates a simple MLP network with one hidden layer consisting three hidden neurons.

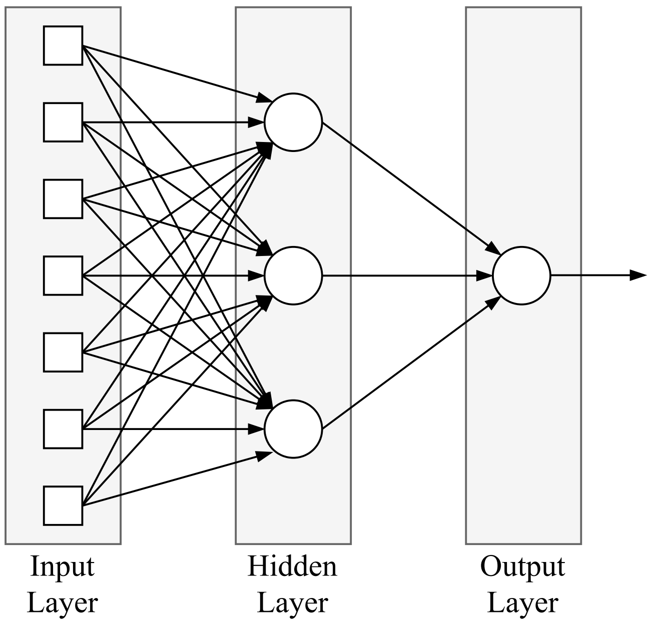


Figure 2.6: Illustration of feed-forward MLP consisting one hidden layer.

According to the *universal approximation theorem*, a network with one single hidden layer, referred as *shallow neural network*, with a large but finite number of neurons can approximate a wide range of continuous functions arbitrarily well. However, it has been shown that instead of a shallow network, a network with several hidden layers with fewer neurons is more effective, due to the fact that each layer can learn different abstractions of the transiting data. This type of network is called *deep neural network*. The term deep is not strictly defined, but to distinguish a deep network from a flat network, a network is referred to as deep if it contains at least three hidden layers [22, 23].

According to the architecture the data in a MLP follows the feed forward principle, flowing from the inputs through the hidden layers to the output of the network. Consequentially, the MLP is often referred as feed forward network. The parameters (weights and biases) affecting the flow of the data through the network to produce the output. According to Section 2.2.4, during training the parameters are optimized. Therefore the error gradient occurring on the output of the network flows backwards through the network, updating the parameters layer by layer on its way. So the update on a layer is defined through its subsequent layer. Due to this opposite direction of the flow to the feed forward pass, this method to construct the gradient is called *backpropagation.*

2.3.2 Recurrent Time-series Prediction Models

Another statistical learning model type is Recurrent Neural Network (RNN). RNN is a specific ANN type especially designed for modelling sequential data. Sequential data comes in many forms, i.e. audio as a numerical time-series, video as a sequence of picture frames or text as sequence of words.

To learn time-dependent information from data, RNN is based on a special kind of neuron. This neuron is inspired by the concept of sequential memory, which mimics the ability in the human brain to recognize sequences. A toy example is the sequence of all letters in the alphabet, which is recognizable by the brain. In a RNN this ability of recognizing a sequence and making predictions is enabled by changing the architecture of an ANN neuron slightly. Figure 2.7 illustrates both neuron architectures, also know as *neuron cell.* On the left side, the regular ANN cell is shown, passing the input through the unit to the output in a single pass. In contrast on the right, for the RNN cell a loop is added, which allows information to flow from one step in time to the next. This time-coded information is called the *hidden state* and contains information from previous cell inputs.

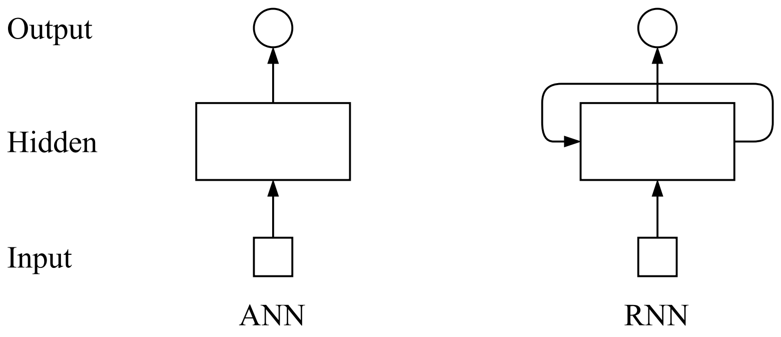


Figure 2.7: Illustrative comparison of an ANN neuron cell and RNN neuron cell.

A sequence of input data is considered as a series of input values in a RNN. The data for each step in this series is passed to the RNN cell iteratively and is combined in the cell with the hidden state. Consequentially, the hidden state of the current time-step includes recursively the output of all previous inputs. An illustration of this recursive data flow through a RNN cell is presented for an input data sequence $x_0, ..., x_n$ as unravelled architecture in Figure 2.8. Furthermore, the figure clearly shows the two output formats that RNN-based architectures can provide. The RNN cell can be used for either static (only the last step y_n) or sequential (all $y_0, ..., y_n$) output.

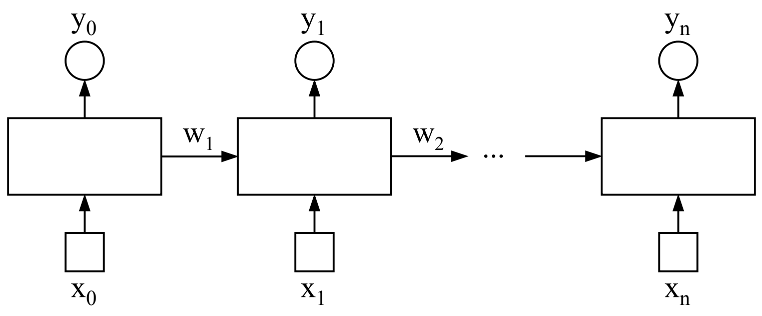


Figure 2.8: Illustration of an unravelled RNN cell architecture.

For each time step, a weight w_i is applied to the hidden state when it is passed through the sequence. This weighted looping through the sequential data creates the issue of short-term memory. This problem is also know as the *vanishing gradient problem*, which occurs on deep neural networks as well due to how they are trained with backpropagation. In RNN the vanishing gradient effect is caused by the recursive connection and thus depending on information processed before. Similar to a layer in a feed forward neural network during backpropagation each layer depends on the gradient of later layers in the network. Multiplying the gradient from layer to layer backwards exponentially shrinks the gradient. This causes the earlier layers to fail optimizing the parameters as they are barely adjusted. Following this issue in the recursive architecture on a RNN, this makes it hard to capture long-term dependencies.

A solution to face the vanishing gradient problem was presented by Hochreiter and Schmidhuber in 1997. They introduced an evolved RNN cell architecture using an internal cell memory and additional gates. Gates allow the RNN cell to regulate the flow of the information and thus which information to add to or remove from the cell memory. This enables the ability to learn long-term dependencies that can relate to events thousands of discrete time steps in the past, hence this architecture is called Long Short-Term Memory (LSTM) [24].

2.3.3 Convolutional Time-series Prediction Models

Until recently sequence modelling in machine learning was synonymous with recurrent networks. However, recent results indicate convolutional architectures as powerful alternative on sequence modelling, coming mainly from the real-time audio synthesis and machine translation [25, 26, 27]. In 2018, a generic family for convolutional sequence modelling architectures was introduced as Temporal Convolutional Networks (TCN). This generic TCN architecture combines the best practices of modern convolutional architectures of a wide range of research studies [28].

The main difference in TCN compared to recurrent networks is how the data is processed. As shown in the previous sections, recurrent networks transport important information as internal states through the sequence while processing. In contrast convolutional networks use filters to condense the important information from sequences to a reduced sequence representation, the so-called *latent space* representation.

The TCN architecture can be distinguished from the regular convolution used mainly for processing images, with one main concept, *causal convolution.* Causal convolution is a variant of ordinary convolution where the causal property is preserved for time-series data. A comparison of regular and causal convolution is illustrated in Figure 2.9. On the left side, the regular convolution is illustrated as a filter with a kernel size of three passed through the time-series sequence. In contrast to the causal kernel on the right this reveals that the regular convolution violates the causal property and x_{T+1} is involved in the receptive field of the kernel at time x_T to produce the output y_T.

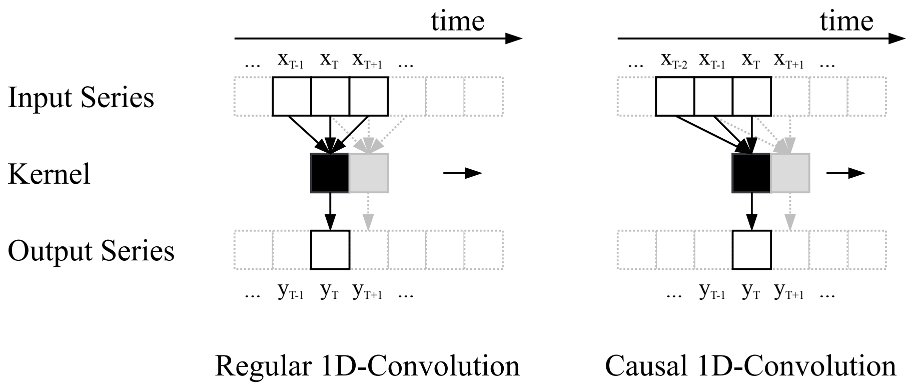

time
time
... x_{T-1} x_T x_{T+1} ...
... x_{T-2} x_{T-1} x_T x_{T+1} ...
Input Series
Kernel
Output Series
... y_{T-1} y_T y_{T+1} ...
... y_{T-1} y_T y_{T+1} ...
Regular 1D-Convolution
Causal 1D-Convolution

Figure 2.9: Illustrative comparison of regular and causal 1D convolution for time-series data.

This basic temporal convolution architecture can look back at previous time-steps with size linear in the depth of the network and the filter size, and thus capturing long term dependencies becomes challenging, i.e. the receptive field in the figure above shows a depth of one and a kernel size of three. To overcome this problem TCN benefits from the concept of *dilated convolution.* Dilated convolution allows to increase the receptive field width of the convolution kernel, without requiring an increase in the number of parameters. This is done by only considering every l_{th} element of a sequence in a l-dilated convolution with dilation factor l. The l-dilated convolution operation F for an element s of the input time-series $\boldsymbol{x}$ is defined as

$$F(s) = (\boldsymbol{x} *_l f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \boldsymbol{x}_{s-l\cdot i}, \tag{2.14}$$

where $f : \{0, ..., k-1\} \to \mathbb{R}$ is a filter of kernel size k and $s - l \cdot i$ accounts for the direction of the past. Hence the regular convolution is a dilated convolution with factor of one [28, 29].

This allows to increase the receptive field and thus the considered length of input sequence for the TCN. Figure 2.10 illustrates a comparison of dilation factor $l = 1$ and $l = 2$ for a causal convolution filter with kernel size $k = 3$. This already results in a receptive field of five with the same number of parameters.

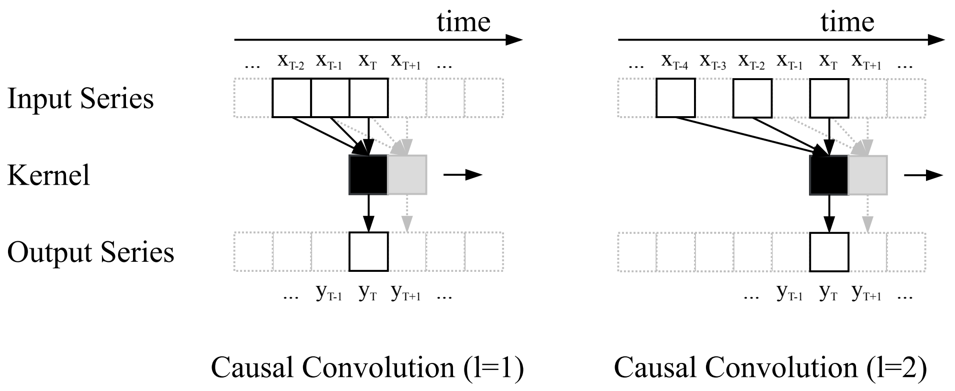

time
... x_{T-2} x_{T-1} x_T x_{T+1} ...
... x_{T-4} x_{T-3} x_{T-2} x_{T-1} x_T x_{T+1} ...
Input Series
Kernel
Output Series
... y_{T-1} y_T y_{T+1} ...
Causal Convolution (l=1)
Causal Convolution (l=2)

Figure 2.10: Illustrative comparison of causal 1D convolution for different dilation factors with the same kernel size on time-series data.

To further increase the receptive field, dilated convolution is used as a stack with varying dilation factors. Therefore, the dilation factor increases exponential with depth of the stack. This ensures that each time step of an input is captured by the filter, while at the same time allowing the capture of an extensive history. Following Figure 2.11 illustrates this with 3 layers of dilated convolution with increasing dilation factor $l = [1, 2, 4]$ and kernel size $k = 3$. This leads to a receptive field of width 15 compared to a non-dilated convolution width of 7 for the same kernel and depth. The illustrated convolution filter is in literature referred to as TCN block or stack. Stacking several of these blocks also increases the recording field by the factor of the number of stacked blocks. Hence, the number of stacked blocks is an additional hyper-parameter of the TCN architecture.

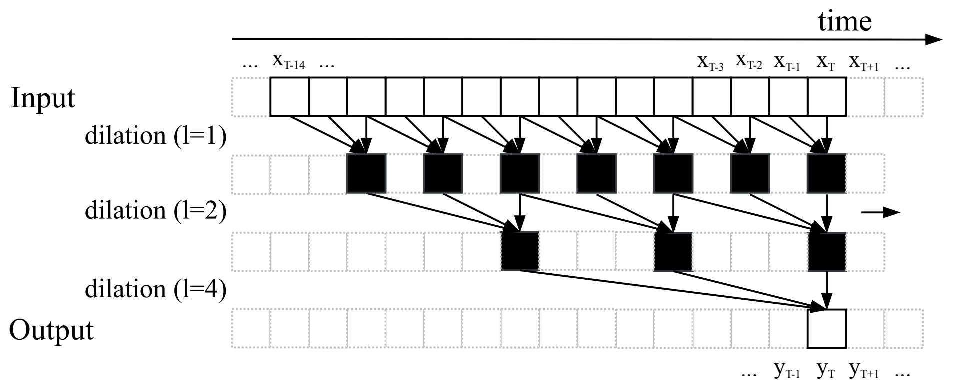

time
... x_{T-14} ...
x_{T-3} x_{T-2} x_{T-1} x_T x_{T+1} ...
Input
dilation (l=1)
dilation (l=2)
dilation (l=4)
Output
... y_{T-1} y_T y_{T+1} ...

Figure 2.11: Illustration of a 1D causal, dilated convolution filter with kernel size $k = 3$ and depth 3.

Since the size and architecture of the receptive field is fixed, to deliver an output y_0 for the first input sequence time-step x_0 there are no previous values available in order to fill the complete kernel while ensuring the causal property. For this reason, zeros are added before the first time-step in the input sequence, to ensure that input and output sequence are of the same length. This is called *input zero padding*.

As a conclusion, temporal convolutional networks seem to be a great alternative to recurrent networks. The different approach to handle sequential data has several advantages [28]:

- Convolution networks pair well with parallel matrix computations, i.e. on a GPU, because they do not require shared internal states.
- Convolution is flexible. Parameters such as filter size or depth and dilation factor easily increase the receptive field.
- Convolution layers are not affected by the vanishing gradient problem.
- Convolution is lighter on memory.

3 Data Exploration

This chapter portrays the available data in the project and how it was used as dataset, since data understanding is an essential step in data-driven modelling projects. All data used in this thesis project was provided by CEVT AB and is not available in public. No other data was used, neither additional nor generated datasets.

3.1 Data Gathering

The provided data was collected from a fleet of Plug-in Hybrid vehicles (PHEV). As a part of the after market tools at CEVT, the data is gathered from the vehicles during workshop visits. Accordingly, the data collection is a continuous process where data is added to the available data pool every day through new workshop visits. This is typical for data collection in the Internet of Things (IOT) era, where data is collected everywhere even without plans to use it directly, but to keep for later use. This was the case for the data used in this thesis. The data was originally not intended to be used for research project like this thesis project. Hence, it was impossible to change contained data fields, shape, frequency of logging, precision of logging etc.

The data gathering process from the fleet to the cloud, is illustrated in Figure 3.1, starting with the gathering in the fleet. In a vehicle, multiple electronic control units (ECU) continuously monitor sensors, control the vehicle and track measurement data. Plug-In and fully electrified vehicles usually have a separate ECU module responsible for battery functions, known as the Battery Energy Control Module (BECM). This module is responsible for taking readings of battery internal values (i.e current, voltage or temperature), ensuring safety functions, computing battery states (i.e. SOC or SOH) and reporting the state of the battery to other ECUs in the vehicle. Besides an internal memory for the battery state functionality the BECM also has a diagnostic data memory. In general, the diagnostic data is saved in this memory to be able to screen the battery status and history during a workshop stop in case that there is a problem with the battery. Therefore, the diagnostic data is regularly stored even if the vehicle is not driven, for example if it charges during parking or when it wakes up during parking to ensure that the battery is in a stable condition and within the temperature limits. Because of this design for diagnostic purpose, the data is only read out during workshop stops. At the stop, a worker connects the vehicle and a software tool to read out the diagnostic data, which is then stored in a cloud database.

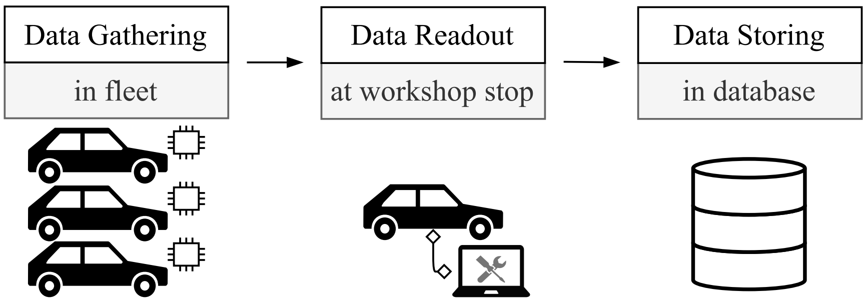


Figure 3.1: Illustration of the data gathering process from fleet to database.

Due to the mentioned continuous growth of the big data pool, differently sized data collections were used during the thesis. The dimensions of the final data collection, which was used to evaluate the different models, are given in Table 3.1. Only complete readouts were considered. As missing data is a common effect in many real life data sources, this reduced the total number of usable vehicles to 3 150, due to problems with the data readout specifications. To only consider most recent data per vehicle, only the last available readout for each of these vehicles was considered.

Table 3.1: Dimensions of the final data collection

Property	Value
Number of vehicles	3 150
Number of readouts	3 150
Data file size	≈ 2 GB

3.2 Data Type Exploration

Before any closer analysis of the data it is necessary to familiarise with the overall structure and content of the readout data. In order to reduce the diagnostic data memory in the vehicle, the data gathered over lifetime is partly converted into histograms. Besides the gathering of periodically saved values with accumulating time or distance, data is added to the histogram bins with the occurrence of certain events, for example when starting the vehicle or battery limits are violated. Consequently, multiple data types are collected in the diagnostic memory. An overview of the available data types per readout is given in Figure 3.2.

The data can be split into two main data types, static and sequential data. Owing to the vastly different data types, the sequential and static data are explored separately in following Section 3.2.1 and 3.2.2.

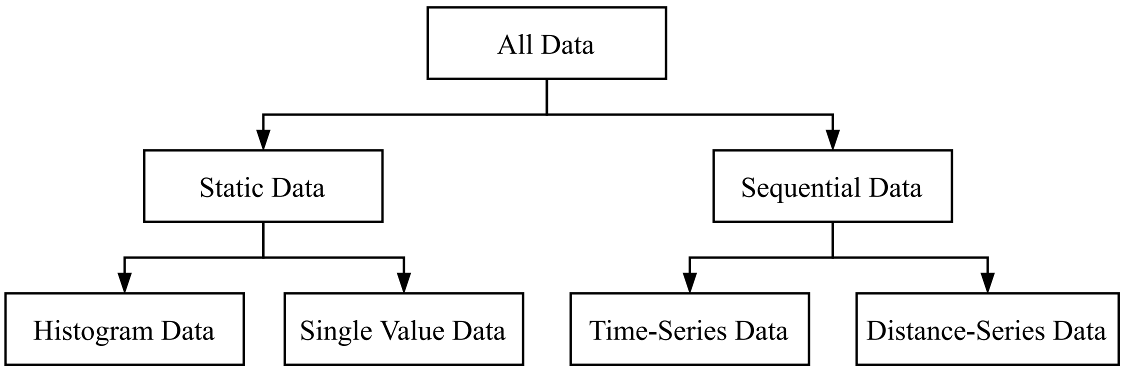


Figure 3.2: Illustration of included data content in each readout from a workshop stop.

3.2.1 Sequential Data

The sequential data gathered from the vehicle fleet contains time-dependent and distance-dependent variables. The sequential variables are computed continuously in the vehicles, but are only stored in the diagnostic memory at certain frequencies or events, depending on the type of series. For time-dependent data the saving frequency is three months for the first two years and decreases afterwards down to a frequency of only one saving per year. The distance-dependent data is in the beginning stored every 1 000 kilometres, up to every 20 000 kilometres beyond a certain mileage.

This design was chosen during the battery software design phase to ensure that data can be collected over a long lifetime without requiring extensive diagnostic data memory in the vehicle. For a data-driven modelling process, this brings advantages and disadvantages. On the first hand a rare storage of data reduces the overall data volume. Hence, the whole data gathering process including readout or uploading to the cloud is faster. On the other hand side, data-driven models benefit from large training data volume and the accuracy of a model usually improves as the available data increases.

A complete list of the sequential diagnostic data gathered per vehicle can be found in Table 3.2. Each listed variable is recorded over time and distance. Only in special cases, e.g. when a vehicle is not moved for a long time, the data between the time and distance series for one variable are non-redundant. Hence, only the time-dependent type of series was considered during this thesis to avoid redundancy in the variables. From the dimension column in the table, it can be seen that almost every variable gathered can be further divided into multiple single time-series, i.e. the variable "Accumulated Energy Throughput" is gathered as three individual series, one series per drive mode. Thus 16 recorded time series per vehicle are obtained for the variables listed in the table below. This includes the capacity SOH series, which was used in this thesis as target for the sequential modelling tasks. Given the fact that input and target are gathered in parallel is an advantage for the modelling problem, because it ensures that the data is collected at the same frequency and is consistent in length. A potential drawback of this could be that if the SOH data is inaccurate, a model trained on it will not produce more accurate results. However, this reliance on the quality of the data applies to all supervised learning problems.

Additionally to the capacity SOH, the BMS stores the intermediate products for the calculations as variables "0C/1C Capacity" and "Available Energy" in the diagnostic memory. To avoid that a model only learns the relation between intermediate variables and the SOH as a target, these variables have been excluded from the training data in this thesis.

Since all the sequential data is numeric, there is no need to format or encode the data to be used in the data-driven models.

Table 3.2: List of all gathered sequential diagnostic time-series

Variable Name	Dimension	Description
Charge Resistance	(2 x steps)	Charge resistance for different SOC levels (70 and 50 %)
Discharge Resistance	(2 x steps)	Discharge resistance for different SOC levels (50 and 20 %)
Available Energy	(2 x steps)	Available energy (kWh) for charge and discharge
0C Capacity	(2 x steps)	The min. and max. 0C cell capacity in the battery
1C Capacity	(2 x steps)	The min. and max. 1C cell capacity in the battery
2s Resistance	(2 x steps)	The min. and max. 2s cell resistance in the battery
SOH capacity	(1 x steps)	The SOH capacity of the full battery **(target)**
Acc. Energy throughput	(3 x steps)	Accumulated energy throughput for three drive modes

3.2.2 Static Data

Similar to the sequential diagnostic data, the corresponding source variables for the static diagnostic data are continuously calculated in the BMS. However, in contrast to the sequential data, the time or distance axis is not taken into account for storing and thus the data is only stored as absolute values or as part of a histogram. As a result, the data of one readout only represent the state of the vehicle on the readout day, similar to a snapshot of the vehicle state. For this reason, a lot of potentially relevant information is already lost in the data gathering process.

In total around 30 histograms are included in a readout. A selection has been carried out to exclude histograms from the thesis that seem defective or irrelevant for the battery condition. The selection was based on domain knowledge provided by the supervisors at CEVT and the known battery degradation factors from Section 2.1.3. This has led to a reduction in the number of histograms from 30 to 16.

Additionally, the selected histograms were grouped based on the battery degradation factor they belong to. This has been done to determine the most relevant histograms within the defined degradation factor groups. The resulting groups together with the number and dimensions of included histograms are listed below in Table 3.3. The third column in the table lists the applied feature extraction calculation performed on the histograms per feature group. This has been applied to histograms in order to extract their main information and reduce the data per vehicle significantly, intended to be informative and non-redundant.

For instance, features from the histograms in the depth of discharge group are extracted with the statistical properties of total sum, mean, median, standard deviation, skewness, kurtosis and cycles. Cycles in this case is a rough approximation of cycles through the division of the total sum of discharged SOC with 100%. Skewness is a statistical measure that describes the nature and strength of the asymmetry of a distribution. It indicates whether and to what extent the distribution is inclined to one side. Kurtosis is a measure for the steepness or "relative peakedness" of a distribution function. Since all the extracted feature data is numeric, there was no reason for further formatting or encoding of the data to be used in the models.

For the static target, two features have been extracted from the corresponding histogram. Under the assumption that the minimum SOH value is obtained at the last measured SOH value, both features describe the same variance. That this assumption holds will be shown in Section 5.2. Hence, only the minimum SOH value was selected as static target feature, since it appears to be more intuitive to understand.

Table 3.3: List of grouped histograms, their dimensions and feature extraction calculation properties

Feature group	Type of Histograms	Feature calculations
Energy Throughput	4 x 3D	sum
Depth of Discharge	1 x 2D	sum, mean, median, std., skewness, kurtosis, cycles
Current	1 x 2D and 2 x 3D	sum, mean, median, std.
Temperature	5 x 3D	mean, median, std., min, max
State of Charge	5 x 3D	mean, median, std.
Over-charge/discharge	2 x 3D	count, mean, median, std.
Age	1 x 1D	total
SOH (Target)	1 x 1D	min, difference (between initial and last value)

4 Methodology

This chapter serves the methodology used during the thesis work. As a higher-level methodology, a widely used data-mining standard process has been used to bring structure into the thesis work. This procedure is presented in Section 4.1 in order to explain covered major steps, which are described in detail in the remaining Sections, 4.2 to 4.5, of this chapter.

4.1 The CRISP-DM Methodology

Considering the strong relation of this thesis work to data-mining methods, the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology was followed to a large extent as basis for the work process.

The CRISP-DM process is a standard cross-sector procedure promoted by the European Union for data-driven modelling development. The methodology was conceived in 1996 as a step-by-step guide for data-driven projects by a core consortium of companies from the data mining, data warehouse, insurance and automotive sector. In total, the process methodology covers the full development phase of a data-driven project with six major phases, from an understanding part over the model engineering part to a deployment part. The major phases in the process are not strictly set and iterating back and forth is part of the process. In Figure 4.1 the most important and frequent transitions between the major phases are marked with arrows [30].

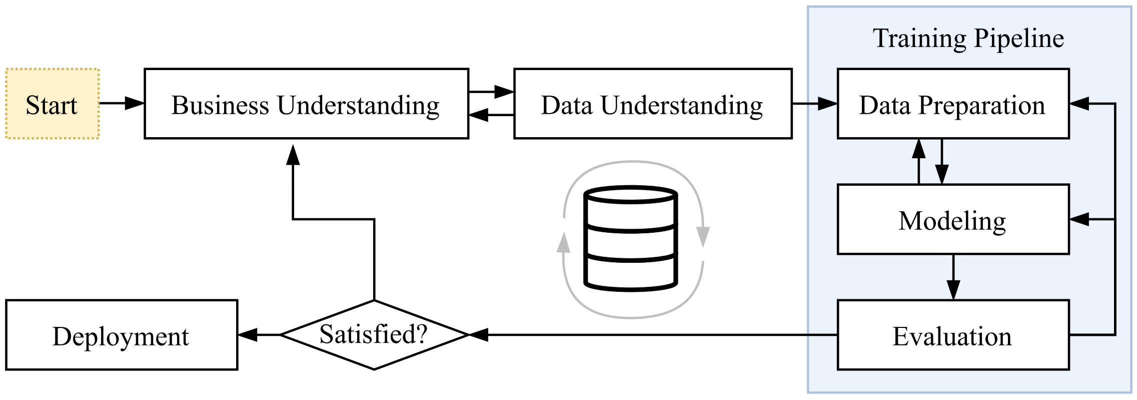


Figure 4.1: Illustration of the CRISP-DM standard process.

According to the CRISP-DM methodology the modelling phase, as an iterative process involving data preparation and evaluation, benefits from the use of a pipeline. This allows flexibility for varying data, preparation steps and model architectures.

During this thesis project such a pipeline was implemented, to act as a flexible backbone for improvement iterations during modelling. The associated or required methods for each major step of the methodology are listed in the next sections. Starting with the preparation of the available data into a dataset in Section 4.2. The modelling problem description is presented in Section 4.3, including the selection of promising model architecture. Thereafter, the training procedure for selected models is described in Section 4.4 and finally Section 4.5 is devoted to evaluation of the model, in order to benchmark the investigated models.

4.2 Dataset Creation

The first major step in the CRISP-DM process attempts the dataset creation. The goal of this step is to use domain knowledge combined with best practice tools to create a comprehensive, condensed dataset from the raw data extracted features discussed in Chapter 3.

It is important to mention in advance that the methodology for reducing the data volume described in Section 4.2.1 was not a permanent component of the tool chain, and will instead be examined separately in Section 5.2 of the results chapter.

4.2.1 Feature Selection Procedure

Feature Subset Selection (FSS), or *feature selection* in short, denotes the process of selecting a subset of features from the total number of available features before training a learning algorithm. The central goal of using feature selection techniques is to remove either redundant or irrelevant features before model training, without incurring much loss of information. Redundant and irrelevant are two distinct notions, since two relevant features may be redundant if they are strongly pairwise correlated. The objectives to be achieved with features selection promise three advantages: improving the prediction accuracy, improving training efficiency and providing better understanding of the data generating process [31].

In this thesis, a simple correlation-based filtering method is used to select the features. Therefore, the *correlation* is calculated between each feature and the target feature as a dimensionless correlation coefficient that measures strength and direction of association between two continuous variables. The coefficient is a varying value between +1 and -1, where a coefficient close to ± 1 indicates strong positive or negative correlation between the two variables. Hence, the best features have been selected by picking the features with the top absolute coefficient value between feature and target, to achieve a strong correlation to the target and discard irrelevant features.

This correlation-based filtering method is quickly calculable, can be applied without any usage of a model and is intuitively interpretable. However, the sole consideration of correlation with respect to the target can leave redundant features in the data and can not take into account an impact of the learning algorithm. To prevent the former, the correlation analysis has been combined with a *cross-correlation* analysis and *principle component analysis* (PCA), to reduce redundant features.

The cross-correlation analysis is a pairwise correlation analysis among the features without the target. Features with a high cross-correlation are more linearly dependent and hence have almost the same effect with respect to the target feature. So, when two features have high cross-correlation, one feature can be dropped to reduce redundancy.

In addition, PCA was applied on the input features to further reduce the feature space. The main objective of PCA is to find a transformation from the feature data space onto a component data space, where dimensions are orthogonal and span into largest variance directions. As a result, PCA returns the principal components ordered by their largest explained variance. This is helpful to reduce the dimension of the feature data by discarding the least informative principal components, hence dropping less relevant features.

The full feature selection procedure is illustrated in Figure 4.2. In the beginning, all input and target features are given. In a first step, the features are assigned to feature groups. This has been done according to the battery degradation factors, leading to the same groups as in Section 3.2.2. Each features group then has been analysed individually with a correlation analysis with respect to the SOH target and pairwise cross-correlation analysis among the features per group, followed by PCA. As a final result, the feature selection procedure delivers a subset of high target correlating and redundancy-free features.

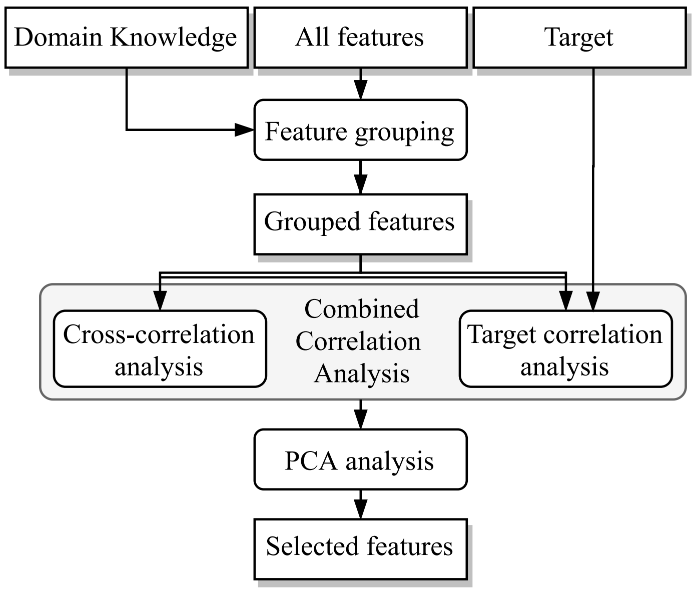


Figure 4.2: Illustration of the complete feature selection process.

4.2.2 Data Standardization

Input data scaling is a well-established technique for improving the convergence characteristics of a network. Scaling inputs does not affect the network accuracy performance, due to the compensation from the network weights, i.e. largely scaled inputs are compensated with largely scaled weights. However, scaling affects the training performance, i.e. networks with large weights are more sensitive to the input, which may lead to unstable training performance and lower generalization. For this reason, scaling is an essential pre-processing step and was applied on the data in this project.

There are different scalers available to transform input data. Probably the most used scaler is the *standardization scaler*. It scales the data per feature to the statistical properties of a standard Gaussian distribution by removing the mean feature mean and scaling to unit variance. This leads to an unbiased, standardized input data. However, standardization can be affected by outlier values in the data. In order to get around this problem another common choices is the so-called *robust scaler*. This scaler uses the interquartile range to identify the scaling parameters, and is thus unaffected by outliers in the data, allowing the main portion of the dataset to be more spread out.

4.2.3 Sliding Window Transformation

The *sliding window* transformation was used in this thesis to transform sequential to static data. This is a data preparation step that is specifically used for the step model architecture in the SOH estimation model comparison and, hence, is not used as a permanent preparation for all input datasets.

An illustration of this transformation for an uni-variate time-series is given in Figure 4.3 with window size $w = 3$. The transformation cuts uni-variate or multi-variate time-series data into time-series parts equal to the window size w. In the case of the step model the data of each window was subsequently transformed to static data. Therefore the sum for each time-series per window was calculated and thus a $1 \times n_{features}$ shaped vector was obtained for each window. Furthermore, the target SOH time-series was split with the same window size. The first element of each target window was additionally added to the transformed input windowed data vector and the last element as static target for the window.

The sliding window transformation has only been applied on sequential data in this thesis. This is due to the readout dependency of the static feature data. Since only one readout per vehicle is used, all static features must be interpolatable for the transformation. However, this does not apply to all available static features. For example, the static value of battery age can be interpolated over the elapsed lifetime by default, whereas a temperature value cannot. The temperature history, as an example, is additionally dependent on the temperature variation through the seasons since the date of production. Therefore, a vehicle manufactured in winter and checked in summer cannot be interpolated in the same way as a vehicle

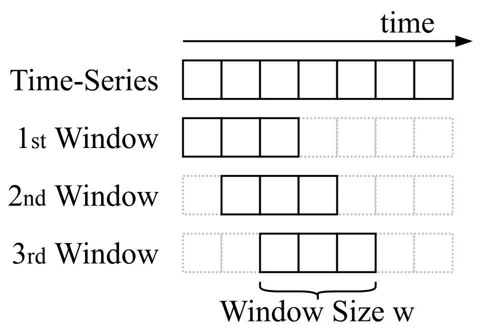


Figure 4.3: Illustration of sliding window transformation with window size $w = 3$.

manufactured in winter and checked in the succeeding winter. To this end, the static feature data has been excluded in the sliding window transformation.

4.2.4 Data Arrangement

As a final step in the dataset creation the full input data was split into training and evaluation data subsets. This has been done over the total number of all remaining vehicles. The data was randomly split according to an 80/20 split rate, such that the data of 80% of the total vehicles is used for training and the remaining 20% for evaluation.

In addition, the data was arranged to supply the correct data format for each model in the model comparison. This results in four different datasets: pure static, pure sequential, mixed and windowed sequential. In summary, Table 4.1 lists all considered datasets used for generating the results, based on the data from Chapter 3.

Table 4.1: List of all considered datasets

Dataset Name	Included Data Types	Split
static dataset	static	80/20
sequential dataset	sequential	80/20
mixed dataset	static & sequential	80/20
windowed dataset	window transformed sequential	80/20

4.3 Model Selection

A model comparison was conducted, since the available data with several data types allow the application of different prognosis approaches. The goal of the comparison was therefore to find a suitable prognosis approach and a matching model architecture to solve the modelling problem.

The modelling problem and the prognosis approach that solves it depend on the available data. With a finalized dataset the input and output data shape for the modelling problem is given and the prognosis approach can be established. Hence, based on the datasets introduced in the previous Section 4.2.4, in Section 4.3.1 an overview on the considered modelling problem formulations is given. Afterwards the setup for the SOH estimation model comparison is presented in Section 4.3.2 and the prognosis setup in Section 4.3.3.

4.3.1 Prognosis Approach Selection

Following the general statistical learning idea, a model representation as an entry point helps to reduce the hypothesis space to find a suitable and appropriate model. For this purpose, the representation is based on the available data, more precisely the shape and type of the input and output data. Consequentially, the following model representation rely on the presented dataset options from Table 4.1 in Section 4.2.4.

Stateless function approximator
The intuitive approach to use the model to approximate a sole input output mapping leads to the simplest approach in this comparison. In this case the model can be seen as a stateless function approximator, where

$$\mathrm{SOH}(t_k) = \mathrm{ModelNetwork}\Big(\mathrm{usage}(t_k - t_0)\Big) \tag{4.1}$$

gives the relation between input and output data. The machine learning model ModelNetwork just takes the complete usage history of one vehicle, from the date of production t_0 until the arbitrary day of readout t_k, and returns the estimated absolute SOH value at the readout day $\mathrm{SOH}(t_k)$ as output. Correspondingly, the same stateless function aproximator concept can be applied for sequential data and thus the sequential dataset by extending the return output to a sequence format. This extends Equation 4.1 to

$$\mathrm{SOH}(t_k - t_0) = \mathrm{ModelNetwork}\Big(\mathrm{usage}(t_k - t_0)\Big), \tag{4.2}$$

where sequential SOH output is given for an equal sequence length as the input sequence length $t_k - t_0$.

In both approaches 4.1 and 4.2 the definition of usage is explicitly chosen flexible and not restricted to a specific dataset type. Hence, both approaches can be directly applied to the static, sequential and mixed datasets presented.

Stateful function approximator
In a stateful approach the model attempts to predict the relative change of the SOH rather than the total value. Therefore the SOH is also included in the input of the model, so the model gains an update characteristics. For this case, the output of the model is the relative SOH change $\delta_{\mathrm{SOH}}(t_{k+1})$ given by

$$\delta_{\mathrm{SOH}}(t_{k+1}) = \mathrm{ModelNetwork}\Big(\mathrm{usage}(t_{k+1} - t_k), \mathrm{SOH}(t_k)\Big). \tag{4.3a}$$

In order to obtain the updated absolute value $\mathrm{SOH}(t_{k+1})$, the relative SOH update $\delta_{\mathrm{SOH}}(t_{k+1})$ then has to be applied to the previous state $\mathrm{SOH}(t_k)$ in an additional step by

$$\mathrm{SOH}(t_{k+1}) = \mathrm{SOH}(t_k) + \delta_{\mathrm{SOH}}(t_{k+1}). \tag{4.3b}$$

Instead of only learning the update difference to perform the update manually, the absolute SOH value can also be learned directly by

$$\mathrm{SOH}(t_{k+1}) = \mathrm{ModelNetwork}\Big(\mathrm{usage}(t_{k+1} - t_k), \mathrm{SOH}(t_k)\Big). \tag{4.4}$$

Since the approaches in Equation 4.3b and 4.4 aim for the identical result, only the second variant was further considered due to the resulting lower implementation effort. The approach can be directly applied on the window transformed sequential dataset.

4.3.2 SOH Estimation Setup

According to the first goal of this thesis, to investigate different modelling approaches in order to find a suitable model being capable of handling big data as input and delivering battery SOH as an output, this section presents a selection of models based on the modelling approaches of the previous section.

As aforementioned, the stateless approach for static (Equation 4.1) and sequential (Equation 4.2) SOH output are able to be applied to static, sequential and mixed input data. In general, this leads to a total combination of six possible input-output combinations. However, the static input to sequential output combination is known as impractical, due to the missing time-dependency in the input. For this reason, it was excluded from the considered combinations.

For the remaining combinations the following model architectures have been selected to predict the SOH based on the usage history.

Static input models
The restriction that purely static input data can only be considered for the static output data leads to the static to static architecture shown in Figure 4.4. Being probably the most popular machine learning algorithm for static output data, the artificial neural network type (ANN) was taken into account in a comparison for this modelling approach. However, depending on the actual selected hyper-parameters, ANN networks can require a lot of parameters to train. In order to evaluate if more resource-efficient algorithms can already provide a suitable solution, three less complex models were added to the comparison:

- *Linear Regression*, as the name indicates, provides a linear least square fitted model. It is considered to be the most basic machine learning model, based on the closed-form ordinary least square solution.
- *Elastic Net* is an extension of linear regression with additional ℓ_1 and ℓ_2 regularization terms.
- *Random Forest* for regression is a so-called ensemble model, aggregating the results of multiple decision trees, where each decision tree provides a prediction output. It is known for efficient processing of large data volume and can handle non-linear data relations.

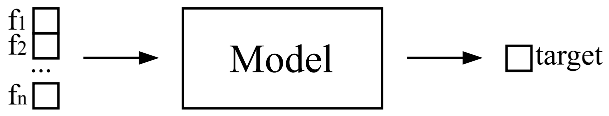


Figure 4.4: Illustration of static features to static target model.

Sequential input models
To additionally incorporate the available sequential data, networks with long short-term memory (LSTM) cells or temporal convolutional networks (TCN) cells were considered in the comparison. The models can be used either for sequential or static output format, as illustrated in Figure 4.5. In a direct comparison, LSTM has been the default model choice for sequential data, where TCN recently appeared as an powerful alternative. Hence, the idea was to considered both allowing a side-by-side comparison.

For this reason, the cell types of the model are capable of handling sequential input data and can return sequential or static output data, by returning either the full length of the sequence output or just the last value of the sequence. The LSTM or TCN cells are used in stacks in the models similar to neurons in a layer. Thus, their output format is concatenated either as 1D vector for the static output or 2D matrix for the sequential output format, where the first dimension refers to the time-steps of the sequential output. Hence, the dimensions have to be reduced to match the available target format, either single value or single time-series.

In the case of the vector cell output, a ANN-like architecture can be used to bring bring the vector to the single value output format for the complete model. On the other hand for the multi-variate 2D cell output the reduction to a single time-series can be achieved with time-distributed wrapped neuron layers. A time-distributed wrapper around a regular neuron layer architecture allows to apply the layer transformation to every time-step in the sequence, so the time dimension is not reduced in the transformation. So if the neuronal layers return a single value as a result, the expected uni-variate time series is obtained due to the preservation of the time dimension.

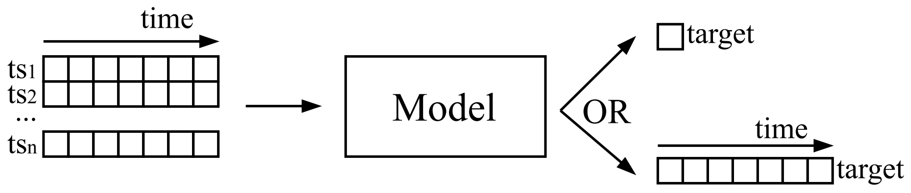


Figure 4.5: Illustration of sequential feature input models with varying output format.

Mixed input models

As investigation if combining both available feature input data types, and thus more data input volume, leads to another results than utilizing them individually, a mixed input architecture was added to the comparison. Therefore the static ANN model was combined with either a LSTM or TCN model, to further benchmark the cell types with each other. An illustration of a mixed input model architecture is presented in Figure 4.6, combining the static ANN model branch with a sequential LSTM/TCN branch into a concatenated branch. The concatenated branch then can be used to either return static or sequential output.

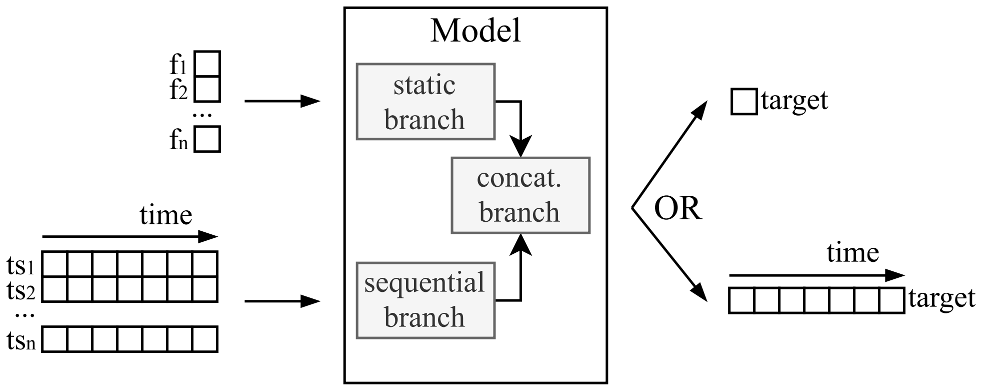


Figure 4.6: Illustration of mixed input features to either static or sequential output format model.

The concatenation of the static and sequential branch in the model depends on the expected output format of the network. For a static output format, the last layer of neurons in the static and sequential branch only contain a fixed number of output neurons. This is illustrated in Figure 4.7. For example, if both branches contain four neurons in the last layer, these eight real value outputs can be stacked in one vector with the length of eight, by concatenating one output after the other. Hence, the following concatenated branch has a static input and behaves in this case like the previous static branch, with additional layers and the single static output value format.

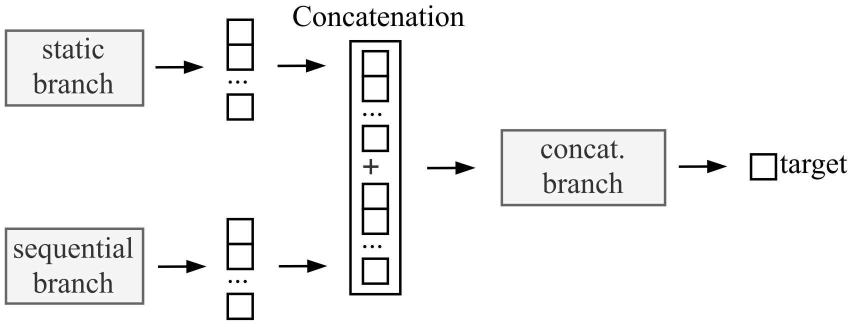


Figure 4.7: Illustration of the concatenation of static and sequential branch for a static model output format.

For a sequential output format, Figure 4.8 illustrates the procedure. The LSTM or TCN cell return full sequential output and thus the output of the sequential branch is a 2D matrix for each sample, referring to the number of time-steps in the sequence times the number of neurons per time-step. This number of neurons initially was equal to the number of sequential features in the input. However, with the LSTM or TCN cell and subsequent layers this can be different from the initial feature number. In order to enable the concatenation of the static and sequential branch, the 1D output of the static branch has to be modified. In this thesis, the 1D output is just repeated for the number of time-steps in the sequential branch. This leads to a 2D branch output for the static branch, which enables the concatenation of both branch outputs over the axis of time-steps. Hence, in this case the concatenation branch handles 2D data and has to reduce the second dimension to only one value at the end. Thus the total output of the model equals the SOH sequence target format, with number of time-steps times one value.

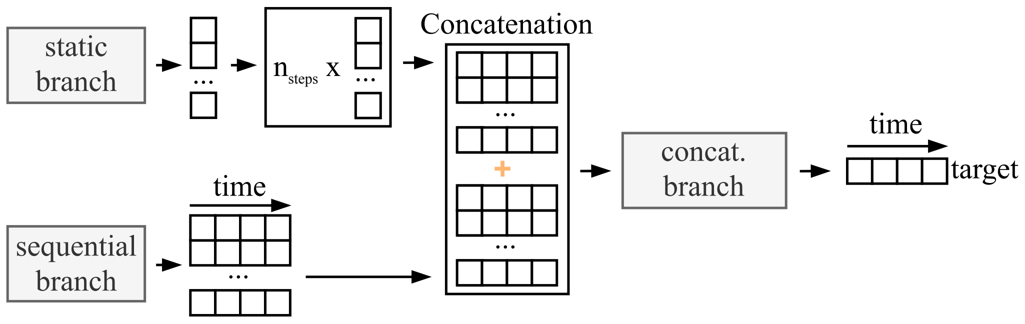


Figure 4.8: Illustration of the concatenation of static and sequential branch for a sequential model output format.

Windowed input step model

Finally, the windowed input model was added to the comparison to investigate the stateful modelling approach from Equation 4.4. This is illustrated in Figure 4.9. As previously discussed, only sequential data is used here for the step approach, due to the missing interpolability of some static features.

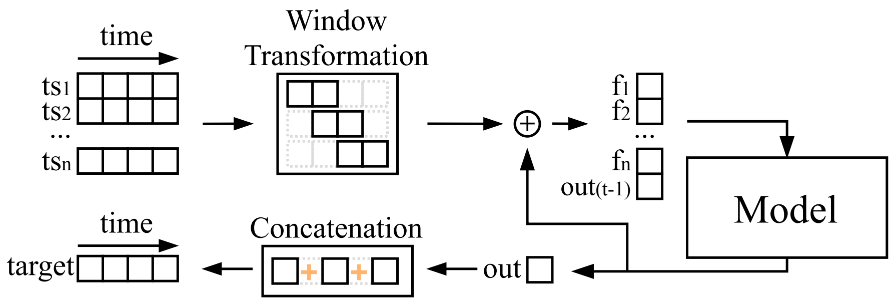


Figure 4.9: Illustration of the step model as windowed input stateful modelling approach.

The sequential input data is transformed into static data with the introduced window transformation in Section 4.2.3, for each sample in the dataset. Depending on the input sequence length of the sample and window size of the transformation this delivers a number of static feature sets, one set per window. The sets are passed through the model iteratively, where for each set the SOH is added as additional feature before feeding the data into the model. For the first set. the SOH feature is initialized with a value of 100 %-SOH. For every subsequent set, the SOH output of the model from the previous window is used as SOH input for the set.

Since the model considers a static input to static output modelling problem, the ANN model was chosen in this approach. To obtain a sequential output format, each model output is stored and concatenated to produce an output of the same length as the sequential input data. Thus the concatenated prediction can be evaluated with respect to the available sequential target.

4.3.3 SOH Prediction Application Setup

As an application showcase, the SOH estimation models from the setup introduced in Section 4.3.2 were tested on a simplified prognosis scenario. The goal was to show chances and limitations of using the investigated estimation models in a prediction setup, where they are tested to predict the future SOH of an individual vehicle based on its history.

Therefore, the data for training and testing was different. The considered application concerns a training on full length sequential data samples of a training dataset, where the full sequence output of the network is compared to the full target SOH sequence to calculate the prediction error, build the gradient and optimize the weights.

The testing is performed on modified samples from the test dataset. The test samples are modified by truncating the sequential input data at half of the sequence and padding it back to the full length with the average of the last three values per input data time-series. This was done to allow a full sequence to sequence prediction. This truncated and padded input sequence is referred to as "half" sequence input in this thesis. In a comparison, the predicted output for the half sequence input is compared with the target of the original test sample and the prediction output for the unmodified sequence input. An illustrative comparison between the full and half sequence input is displayed in Figure 4.10.

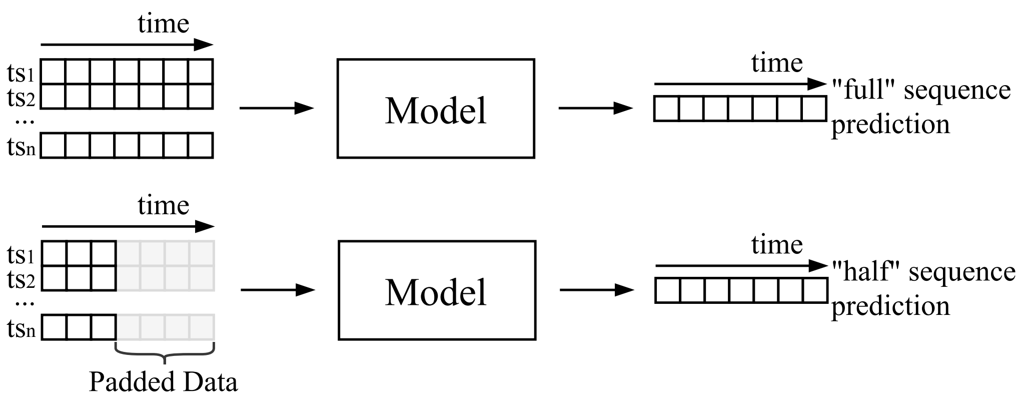


Figure 4.10: Illustration of the comparison between full and half sequential input data handling.

This scenario corresponds to a prediction of the SOH at a readout date at half of the sample forward, using forward propagation of the historic usage into the future, to predict the future SOH. Naturally, the informative value of this predictive application is limited, since this model design with linear usage dispersion does not take into account dynamic usage prediction. Hence, it was expected that if the assumption of consistent future usage forward starting at the half of the sample does not hold, then the prediction will clearly show an increasing bias over time.

4.4 Model Training

This section portrays the considered methods during model training. This includes the used training loss function, hyper-parameter optimization and cross-validation with early stopping.

4.4.1 Loss Function

According to the introduction of statistical learning in Section 2.2, machine learning models use the loss or objective function during training to evaluate the prediction in comparison to the expected output. Therefore, the model is trained with the data using multiple of so-called *epochs*. A complete run through of all input data is called one epoch. The network parameters are updated after each epoch, based on the calculated loss, to improve the prediction in the next epoch iteration.

In this project, the prediction error between target output data and the prediction output was calculated by the mean squared error (MSE), based on the introduced error metric version in Equation 2.7. However, unlike the definition of MSE as evaluation metric, being calculated once per epoch, MSE as loss function during training is calculated for multiple batches of the total dataset. For this reason, the entire dataset is split randomly per epoch into the so-called *mini-batches*, containing a number of samples determined by the hyper-parameter *batch-size*. Applying the

MSE error per mini-batch instead of the entire dataset allows more frequent, smaller updating of the weights during training. This mimics the effect of a small learning-rate and improves convergence robustness.

4.4.2 Hyper-parameter Optimization

In machine learning, hyper-parameter optimization concerns the problem of choosing the optimal set of hyper-parameters for the training of a learning algorithm. The goal is to find the set of parameters that achieve the best possible evaluation score.

A hyper-parameter can be a parameter controlling the model architecture or the training of the model. For example for the former, common hyper-parameters concern the dimension of an ANN network by using the number of layers and the number of neurons per layer as hyper-parameters. For the training usually the batch-size, learning-rate or learning-rate optimizer algorithm are used as hyper-parameters.

In order to find the optimal value for each considered hyper-parameter the range to be investigated has to be defined for each parameter. In this thesis a *grid-search* algorithm was used to search for the tuple of optimal parameter values, by iterating through the grid of given parameter ranges.

4.4.3 Early Stopping

Early stopping is used during the training of a model to reduce training duration and avoid overfitting to the training data. Often, in a long training procedure of a data-driven model, at some point the model does not improve further in the sense of minimizing the loss error, and tends to overfit to the data instead. To avoid this effect a monitoring of the loss error along the trained epochs can be utilized to stop on critical points in time rather than forcing the training for the pre-defined number of epochs. For this purpose, the MSE loss for the available training data set is monitored. As the objective during training is to minimize the loss, the training is stopped as soon as the monitored variable falls beneath a level of change for a selected number of epochs. In this thesis, the level was an absolute total loss error difference of 0.001 for the last 10 past epochs.

4.4.4 Cross-validation training

Cross-validation can be used during training to validate the performance and to allow an assessment of the generalization of a parameterized model. The cross-validation setup is a statistical technique illustrated in Figure 4.11. For the setup, the available training dataset is further divided into several subsets, determined by the cross-validation factor K. According to the number of subsets a number of K models is trained. Hence, for a $K = 5$ cross-validation setup, the available data is split into five subsets of equal size and five different models are trained. For the training of each model, one of the subsets of data is omitted. This remaining subset is used as a so-called *validation* data subset for this specific model. So in total K models are trained for K variations of the training and validation data.

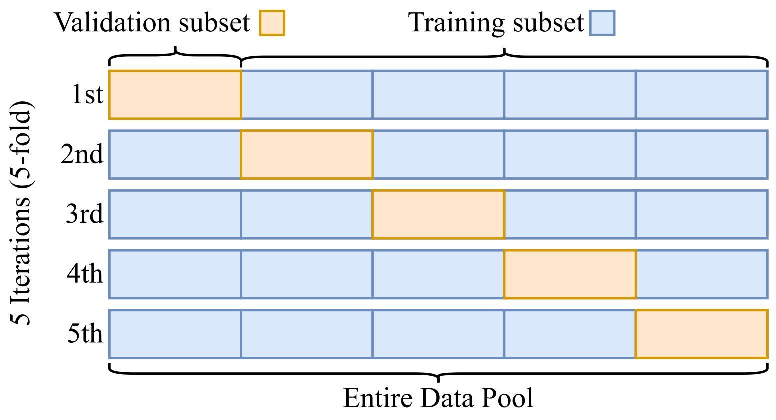


Figure 4.11: Illustration of a five fold ($K = 5$) cross-validation schema.

After each epoch during the training of a model, the validation data is used to calculate the validation loss. This is done similarly to the training loss, with the calculation of MSE as an error metric between the true output and the predicted output of the model for the samples of the validation data subset. Hence, after each trained epoch the training and validation loss is available. According to the discussion on generalization in Section 2.2.5 this is helpful in combination with early stopping, since the training can be stopped either when the validation error seems to stagnate at a certain loss value or is even increasing while the training error still decreases. For the latter this leads to an increase of the difference between training and validation error and thus overfitting is more likely to occur.

For this reason, cross-validation is often used during model selection as well to identify model architectures that are likely to overfit and therefore do not generalize well. For example, cross-validation can be used during the hyper-parameter optimization. However, this would require a significantly longer training period, if each considered configuration in the hyper-parameter grid is trained and evaluated for a number of K modelling folds.

4.5 Model Evaluation and Benchmarking

In the CRISP-DM procedure model evaluation is described as key step to consider how well the trained model generalizes to unseen data. This is important to rate if the trained model acts as expected and consequently if the prediction can be trusted.

How well a model performs is on the first hand measured with mean error performance metrics. However, taking the mean of errors over a large set of samples vanishes the weight on each sample error. Consequently, a small mean error is not reliable enough to measure the performance of a model. Other relevant information about the error is missing in that consideration, i.e. how is the error distributed, is it evenly distributed or with a shift, are there outliers?

The following model evaluation methods were used to provide insights on the model performance and enable the rating of different approaches to provide a solid basis for a discussion and conclusions.

4.5.1 Performance Metrics

Besides the MSE metric, the Mean Absolute Error (MAE) and the Coefficient of Determination (R^2) scores are additionally used as performance metrics during evaluation. Therefore, the trained models are evaluated with the performance metrics over the entire test dataset.

In difference to the MSE metric, the MAE returns the mean over samples of the absolute prediction error. This indicates a scale-dependent measure on the deviation between prediction and target, which can be more intuitive than the squared equivalent MSE. Nevertheless, the MSE is considered in the evaluation as well because it helps to rate the deviation for errors with larger magnitude, as a result of the consideration of the squared prediction error. In addition, the regression specific R^2 score metric, pronounced "R squared", is used to rate the fitting of the models. The R^2 coefficient is a statistical measure how well the regression predictions of the model approximate the true values. A higher score signifies that a network does a better job of predicting expected true values correctly. The best and highest possible R^2 score is 1.0, indicating a 100% fit between predictions and true values. The smaller the value, the worse the prediction. As a drawback of this metric, the score is not well-defined and also depends on the characteristics of the true values. With values that are less scattered being easier to explain the score becomes less important when comparing models with different target values.

4.5.2 Cross-validation Scoring

Aside from the importance of cross-validation for model selection and training, the method is popular to use for evaluation of trained models on limited-sized datasets. According to Section 4.4.4 the K-fold approach helps to identify over-fitting architectures, if they show inconsistent performance end scores when iterating through the modelling folds and the subsets of data, respectively.

For K different folds K different models of the same architecture are trained and evaluated. A consideration of the minimum, maximum and mean value of the performance results thus allows an assessment about the consistency of the achieved scores and therefore about the generalization of the architecture. This helps to rate an architecture amongst other architectures to find the most suitable.

4.5.3 Error Distribution

For a detailed examination of the prediction errors, the model evaluation is supported visually and in statistical terms with the consideration of the prediction error distribution.

In the case of visual support, a common approach is to take the prediction error for each sample in the test dataset and to illustrate them in the form of a histogram along with a fitted kernel density estimation (KDE) function. The presentation as a histogram gives an overview of the frequency distribution of the prediction errors and the KDE function portrays a smoothed curve of the distribution. This combined illustration allows a quantitative view on the prediction errors and thus can reveal the occurrence of outliers in the total amount of errors, which would be undetected if only a mean error value were taken into account.

In addition, the consideration of the statistical properties of the prediction errors extends this into further details to a qualitative view. This enables a detailed side-by-side comparison between model results. Table 4.2 presents the list of evaluated statistical properties and how they contribute to a better understanding and comparison.

Table 4.2: List of statistical properties applied on the error distributions

Statistical Property	Contribution
Mean, deviation, variance	Evaluates the spread of prediction errors
Minimum, maximum values	Reveals extreme outliers, that are vanished in a mean score treatment
Skewness	Measures the symmetry of the errors, hence if the model is more likely over- or underestimating
Kurtosis	Is a measure of the steepness of a (single-peak) probability function, hence describes the ratio of outliers in the distribution

4.5.4 Residual Plot

One of the most used evaluation metrics for machine learning classification problems is the *confusion matrix*. It is used to compare the true output class with the predicted output class of the model for every sample in the test dataset. However, due to the continuous value predicted in regression problems a confusion matrix is not appropriate for regression problems. As an alternative, a *residual plot* is commonly used. Therefore, instead of distinct classes the real value of observation and prediction is compared.

The *residual* value is calculated for each sample in the test dataset as prediction error following the equation $residual_{value} = value_{true} - value_{prediction}$. In the residual plot, the test dataset residuals are plotted in relation to the true target value range as scatter point, where: $(x, y) = (value_{true}, (value_{true} - value_{prediction}))$. For an ideal accurate prediction, with residuals of zero, this would lead to an constant straight line at $y = 0$.

To this end, the residual plot allows an detailed analysis of the prediction errors: How much is the spread of the errors? Is the spread equal over the true value range? Are there outliers in a specific range?

5
Results and Discussion

This chapter presents the results found during different stages of the thesis work. Therefore, the chapter is split into three main sections. Section 5.1 provides the comparison over a range of models to approach the predictive task of SOH estimation for a given vehicle usage. The models are grouped into sets by their delivered output to compare varying data inputs for multiple outputs. In order to achieve unbiased results the models in this section were trained with the full available data. In Section 5.2, a detailed investigation of the impact of data condensation with the feature selection approach is performed. Finally in this chapter, Section 5.3 attempts the prediction of the SOH for a simple prognosis scenario implemented with the models validated in Section 5.1.

All results have been obtained in Python 3.6.8 on an Intel i5 CPU with 8 GB RAM. Used software libraries were, among others, Scikit-learn for the baseline machine learning models and Keras with a Google Tensorflow backend for the neural network models [32, 33].

5.1 SOH Estimation Performance

This section presents the performance of the implemented models to rate and compare their suitability for the predictive SOH estimation task. To receive reliable results all models were trained on the same final dataset (description in Table 3.1) with an 80/20 train and test data split and additionally five-fold cross-validation setup on the retaining 80% training data.

After this introduction part the results of the performance comparison are split into three sections, in order to benchmark models with the same predictive approach respectively the same input and output data. Therefore, models with a static output are compared in Section 5.1.1, models with sequential output in Section 5.1.2 and the step approach, as a mixture of both, in Section 5.1.3. In each of these sections, the model hyper-parameters are presented before the performance metric results. The error distribution examination and residual plots are presented for a detailed understanding, according to the introduction in Section 4.5. Finally, Section 5.1.4 summarizes the findings per approach and evaluates the best per category among the three predictive approaches.

According to the model selection in Section 4.3.2 four different subsets (static, sequence, mixed and windowed) of the final dataset were used as input and two different subsets (static, sequence) for the SOH target output. Each investigated input and output combination can be found in Table 5.1 below.

The baseline models Regression, Elastic Net, Random Forest and plain ANN are not able to handle sequence or mixed data input. Hence theses models could only be trained on static input and deliver static output. Furthermore, LSTM and TCN models require sequential data as input. Hence, for these models pure static input has not been considered. In addition, to the variant with static output for comparison against the purely static models, a sequential output was also evaluated for the sequential models. As a transition between static and sequential data the step model, according to Section 4.3.2, handles the windowed sequential dataset.

Table 5.1: List of evaluated models for the SOH estimation predictive task

Model Type	ID	Model	Input	Output	See Section
Baseline	B1	Linear Regression	static	static	5.1.1
	B2	Elastic Net	static	static	5.1.1
	B3	Random Forest	static	static	5.1.1
	B4	ANN	static	static	5.1.1
Sequential (Recurrent)	R1	LSTM	sequence	static	5.1.1
	R2	LSTM	sequence	sequence	5.1.2
	R3	LSTM with ANN	mixed	static	5.1.1
	R4	LSTM with ANN	mixed	sequence	5.1.2
Sequential (Convolution)	C1	TCN	sequence	static	5.1.1
	C2	TCN	sequence	sequence	5.1.2
	C3	TCN with ANN	mixed	static	5.1.1
	C4	TCN with ANN	mixed	sequence	5.1.2
Sequential (Step)	S1	ANN step model	sequence (windowed)	sequence (windowed)	5.1.3

5.1.1 Varying Data Input for Static Estimation Output

This section presents the performance results for the part of the model comparison that have a static output format. The considered input choices are static, sequence and mixed (combination of static and sequence) datasets for the same static output data format. First, a description of the models used in this section follows, divided according to the combination of input and output data types.

Baseline Models (static input, static output)
The four baseline models were implemented in different software libraries. The first three models, Linear Regression (B1), Elastic Net (B2) and Random Forest (B3) were implemented with the SciKit-Learn package [32]. The package provides a grid-search hyper-parameter optimization which was applied. However, the Linear Regression implementation in the Scikit-learn package does not require hyper-parameter optimization since the implementation is based on the closed-math ordinary least square function. Therefore, the model type was then trained without grid-search.

For the Elastic Net the resulting hyper-parameters and their provided range for the optimization are presented in Table 5.2. The grid-search optimum value for each parameter is listed in the last column.

Table 5.2: Elastic Net (B2) SOH estimation hyper-parameters

Parameter	Values	Optimum
Alpha	[0.0001, ..., 5]	1.25
L1 Ratio	[0.01, 0.25, 0.5, 0.75, 1.]	0.01

Similarly, the grid-search optimization result for the parameters of the random forest is presented in Table 5.3.

Table 5.3: Random Forest (B3) SOH estimation hyper-parameters

Parameter	Values	Optimum
Max Depth	[2, 3]	3
Estimators	[10, 100, 1000]	100

The fourth baseline model, ANN (B4), was implemented with the Keras package [33], which allows a detailed network modelling. Hence, more parameters were given for the optimization. Grid-search was applied and the hyper-parameters that resulted in the lowest MSE validation error are presented in Table 5.4.

Table 5.4: ANN (B4) SOH estimation hyper-parameters

Parameter	Values	Optimum
Max. Number of Epochs	[1000]	1000
Batch Size	[4, ..., 128]	16
Number of Layers	[1, ..., 10]	2
Number of Neurons	[16, ..., 256]	128
Dropout Rate	[0, ..., 0.5]	0.2
Learning Rate	[1e-1, ..., 1e-6]	5e-3
Optimizer Algorithm	[RMSprop, Adam]	Adam

Recurrent Sequence Models (sequence or mixed input, static output)
The group of recurrent sequence models was implemented with the Keras package [33], which enables native LSTM cell support for recurrent architectures. The main difference between the models in this group is the input used. A mixed input requires a static branch in addition to the sequence branch to be able to process the complete data. Due to time limitations in this thesis, the static branch was adapted from the best found static ANN (B4) model and no further optimisation was conducted. It only retains hyper-parameters of the sequence branch, the concatenated branch and the overall training hyper-parameters to optimize with the grid-search. Table 5.5 presents the best found hyper-parameters.

Table 5.5: LSTM (R1, R3) SOH estimation hyper-parameters

Parameter	Values	Optimum
Max. Number of Epochs	[1000]	1000
Batch Size	[4, ..., 64]	16
Number of LSTM Blocks	[1, 2, 3]	2
Number of Hidden Neurons	[16, 32, 64, 128]	64
Number of Neurons (Branch Output)	[1, 16, 32]	16
Number of Neurons (Concatenated)	[16, 32, 64, 128]	32
Dropout Rate	[0, ..., 0.5]	0.2
Learning Rate	[1e-1, ..., 1e-6]	5e-3
Optimizer Algorithm	[RMSprop, Adam]	Adam

Convolutional Sequence Models (sequence or mixed input, static output)
The third group related to convolutional sequence models was implemented with the Keras package [33], where the implementation of the TCN cell was taken over from Philippe Rémy [34], who adapted the implementation from the original paper for the used Keras package. Similar to recurrent sequential modelling, the models in this TCN group differ in the input data used. The additional static branch and concatenated branch for mixed input was adapted from the LSTM model with mixed input. This allows to compare the convolutional and recurrent approaches side-by-side. Consequently, only TCN-related hyper-parameters were optimized. Table 5.6 presents the best found hyper-parameters.

Table 5.6: TCN (C1, C3) SOH estimation hyper-parameters

Parameter	Values	Optimum
Max. Number of Epochs	[1000]	1000
Batch Size	[4, ..., 64]	16
Number of TCN Filters	[16, 32, 64]	64
Number of TCN Stacks	[1, 2, 4]	2
TCN maximum Dilation Factor	[4, 8]	4
TCN Kernel Size	[1, 2, 4, 8]	2
Number of Neurons (Branch Output)	[1, 16, 32]	16
Number of Neurons (Concatenated)	[16, 32, 64, 128]	32
Dropout Rate	[0, ..., 0.5]	0.0
Learning Rate	[1e-1, ..., 1e-6]	5e-3
Optimizer Algorithm	[RMSprop, Adam]	Adam

According to this hyper-parameter selection the receptive field for each TCN filter was

$$\text{Dilation} \times \text{Kernel Size} \times \text{TCN Stacks} = 4 \times 2 \times 2 = 16. \tag{5.1}$$

Metric Performance

The performance has been determined with five fold cross-validation on the remaining 80% train dataset, to further retain a rate of 80/20 between actual training data and validation data for the training procedure. To this end, five models are trained on the training dataset of all eight models in this section. Table 5.7 summarizes the results collected for each model fold iteration by selecting the best score achieved per model. For MSE and MAE, the smaller the score, the better the result and thus only the minimum score for the five trained models is presented. In contrast, for the R^2 metric, the best result is the highest score. For each metric the best score over all models is marked by blue font in the table.

Three assertions are apparent from Table 5.7. At first, the pure static linear baseline models (B1, B2) perform worst over all metrics in this comparison. The switch to non-linear pure static models (B3, B4) shows a clear improvement, which supports the assumption of a non-linear association of the modelling problem for the available data. Secondly, it is apparent that the best performing model is the mixed input LSTM model (R3), with the exception on the MSE metric, where the mixed input TCN model (C3) performs best. This leads to the third assertion, that mixed input models (R3, C3) outperform their pure sequential (R1, C1) or static (B4) equivalents.

Table 5.7: Comparison of static SOH estimation scores for varying models

Model	ID	Input	MSE (min.)	MAE (min.)	R^2 (max.)
Linear Regression	B1	static	4.1863	1.4971	0.7400
Elastic Net	B2	static	4.4476	1.5402	0.7238
Random Forest	B3	static	2.7149	1.1953	0.8314
ANN	B4	static	1.3232	0.7779	0.9178
LTSM	R1	sequence	1.4392	0.7996	0.9106
LTSM + ANN	R3	mixed	1.0032	0.6825	0.9377
TCN	C1	sequence	3.9953	1.4157	0.75190
TCN + ANN	C3	mixed	0.9624	0.7751	0.9123

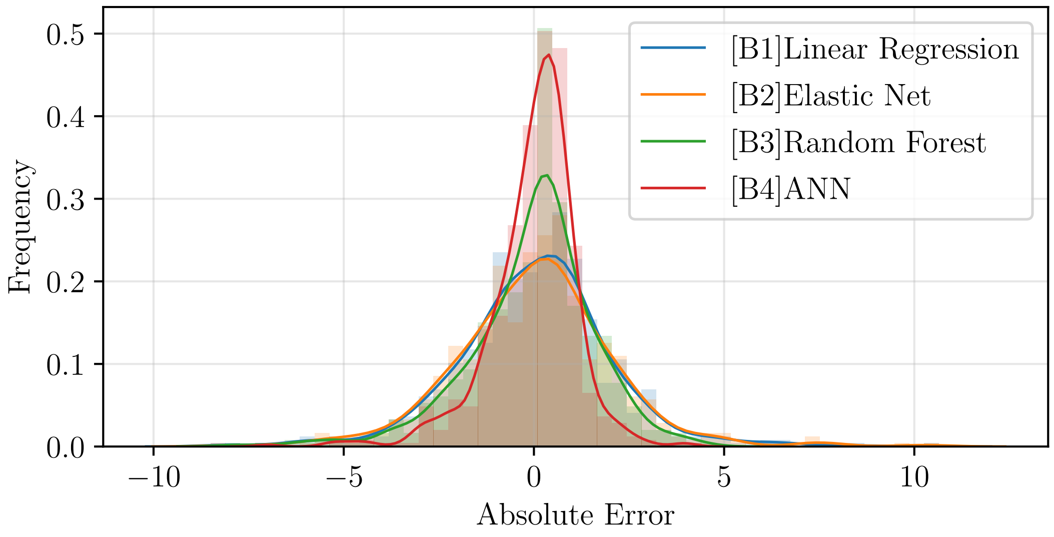
Prediction Error Consideration

Regarding the prediction error distribution, Figure 5.1 serves a comparison across the different models. An overview of the prediction error on each sample of the test dataset is given in form of a histogram with additional kernel density estimation, as introduced in Section 4.5.3. In order to enable intuitive analysis and present differences and similarities, the results are divided into two groups, according to the difference in the input data whether sequential data was considered or not. This results in two groups with four models each. In addition to the visual impression of the distributions, the statistical properties for the distribution per model have been calculated and can be found in Table A.1 in Appendix A.

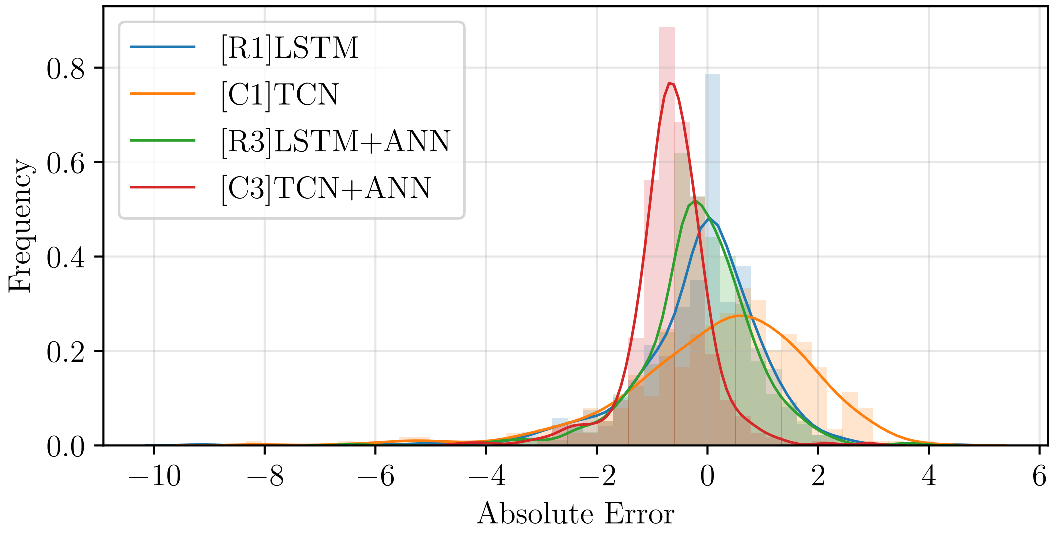
The error distributions confirm the results from the metric evaluation of Table 5.7. Among the baseline models, Figure 5.1a confirms the ANN (B4) model as the best model, with a slim (low variance, high kurtosis), centered (zero mean) peak in the prediction error distribution. In addition, the Random Forest model (B3) results in only a slight increase in the variance compared to the ANN (B4). Both linear models (B1, B2) show higher variance and thus the worst results in this comparison. This supports the assumption, that linear models cannot deal correctly with the modelling problem dynamics. Surprisingly, it is noticeable that all four models show a centered peak in the distribution.

This does not hold for all models in Figure 5.1b, investigating sequential data input. Both TCN models (C1, C3) are shifted to the side, whereby a shift to the right corresponds to an overly degraded estimate, according to the calculation of $y - \hat{y}$ in the MSE. Hence, the left shifted peak of C1 indicates an underestimation and C3 an overestimation of the ageing. The LSTM models (R1, R3) in this second comparison show a more centered (zero mean) and equally shaped distribution peak, which aligns with the overall better performance metric results of the LSTM models (R1, R3) compared to the TCN models (C1, C3) in Table 5.7.

Moreover, the explanation for the lowest MSE value of the mixed input TCN (C3) in the previous Table 5.7 can be derived from Figure 5.1b. In a comparison with the otherwise best mixed input LSTM (R3) model a narrower distribution in deviation, variance, minimum and maximum value reveals the better MSE metric score. However, the shifted mean of this mixed input TCN (C3) model explaining the worse performance in MAE and R^2 score.



(a) Linear Regression (B1), Elastic Net (B2), Random Forest (B3) and ANN (B4).



(b) Recurrent (R1, R3) and convolutional (C1, C3) sequential models.

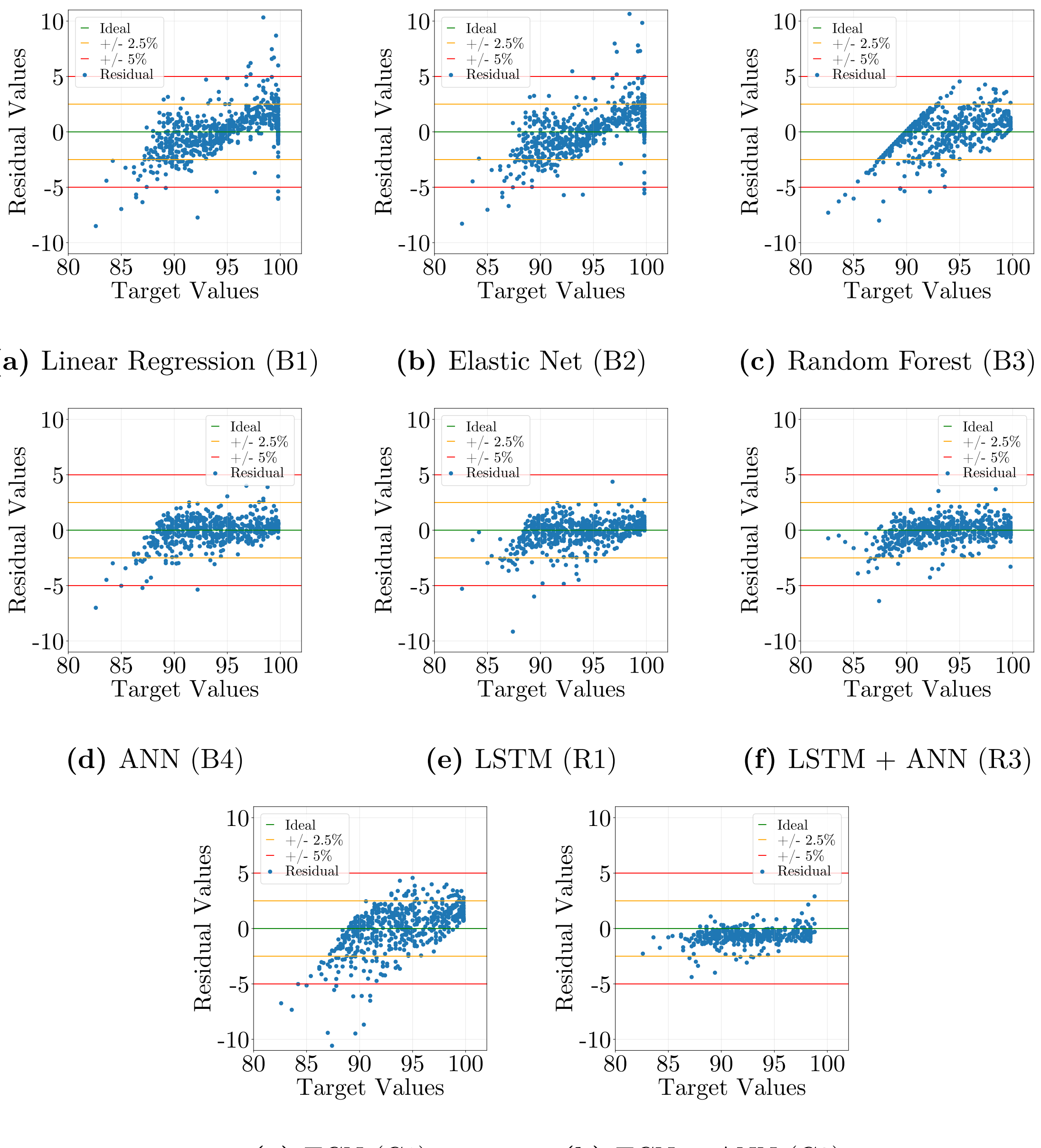
Figure 5.1: Error distribution comparison over different models for varying input and static output on the test dataset.

Residual Inspection
The residual inspection per model in the comparison is presented below in Figure 5.2, as introduced in Section 4.5.4. Therefore, the actual model prediction results are compared to the target SOH in form of a residual plot. These plots illustrate the spread and shape of the predictions, along the range of target values, which is applied on the horizontal x-axis in its original percentage SOH scale. Each point in the plot marks the residual value over the target value on the x-axis for each sample in the test dataset. An ideal result would be a match of true target and model prediction output resulting in a straight constant line at $y = 0$, representing a total positive correlation of 1. This straight line is marked in the figure as green line. Additionally, a $\pm 2.5\%$ and $\pm 5\%$ deviation interval is marked as parallel lines to the ideal line, to allow rating of the results visually.

The plots are consistent with the previously described results. In the first row, the less accurate prediction of the three baseline models (B1, B2, B3) can be seen. Linear Regression (B1) and Elastic Net (B2) show similar performance, both having a wide spread of the residuals and a wave is recognizable around the ideal line. It can also be seen that for a target of 100, there is a wide dispersion from of $\pm 5\%$ for the residuals. The Random Forest model (B3) performs better in this high target value range, with a spread almost constantly below 2.5%. However, it is not able to predict values below 90, which causes the straight diagonal line in the residual plot, crossing the ideal line at the target value of 90. A possible explanation for this could be the overall imbalance in the amount of data over the range of the target value. This is indicated by the same imbalance in the shown test dataset. If there is less training data in the lower range, models can tend to predict wrongly. Especially the baseline models are affected by an unbalance in the data, due to their less complex architecture and thus lower ability to learn complex mappings on unbalanced data.

The ANN (B4), visually outperforms the other baseline models, with significantly less noticeable spread. Most residual spread is below 2.5%. Only a few outliers reach the 5% mark, in the lower half of the target value range. Additionally, the shape of the prediction points is more evenly distributed around the ideal line. In the same row, both LSTM models (R1, R3) can further reduce the dispersion visually. However, some outliers still reach the 5% mark. Outstanding is the accurate prediction of both models in the below SOH 85% target value range, where both models compared to all models mentioned before.

The TCN models complete the comparison and repeatedly confirm the metric and error distribution results. The mixed input TCN (C3) model clearly has the best visual results, with the lowest spread and good performance in the low SOH target range. However, the shift to underestimated ageing can be seen from the shift of residuals to the negative side.



(a) Linear Regression (B1) **(b)** Elastic Net (B2) **(c)** Random Forest (B3)

(d) ANN (B4) **(e)** LSTM (R1) **(f)** LSTM + ANN (R3)

(g) TCN (C1) **(h)** TCN + ANN (C3)

Figure 5.2: Residual comparison for static data on the test dataset.

In summary, in terms of accurate predictions the models with added sequential input data outperform the pure static models in all evaluated metrics. This emphasizes the utilization of available sequential data for the task. However, only using sequential data appears generally worse than using the mixed data variants. The linear baseline models (B1, B2) in this comparison perform worse, which aligns with the assumption that the underlying mapping function between input and output data is non-linear. In addition, all models except the mixed LSTM and TCN models (R3, C3) perform poor in the lower SOH target range, most probably due to the imbalance of data amount over the target value range.

5.1.2 Varying Data Input for Sequence Estimation Output

In addition to the considered static output in the previous section, this chapter presents a comparison of different input formats for sequential data as the target output format. The considered choices for the input format are pure sequential (in R2, C2) and mixed data (in R4, C4). Pure static input cannot be used as sole input to predict sequence data, hence the ANN and baseline models and are excluded from the compared models. The following two paragraphs introduce the compared model architectures.

Sequence input architecture (sequence input, sequence output)
The sequential models from the previous section have been adopted to predict sequential instead of a static output. Therefore, the LSTM (in R2) and TCN (in C2) cells have been setup according to the sequential to sequential model setup from Figure 4.5 in Section 4.3.2.

Mixed input architecture (mixed input, sequence output)
To additionally handle the static data input, the mixed input architecture from the previous section was adopted for sequential instead of static output, based on the architecture introduced in Figure 4.8 of Section 4.3.2. Therefore, the output format of a static branch must be changed so that the actual output is repeated as often as required to achieve the same sequence length as the multi-variate sequential branch output. After this transformation, the sequence branch output and the repeated static output can be concatenated over the axis of time.

Metric Performance
The metric performance per model is displayed in Table 5.8. Both TCN models (C2, C4) perform worse than the LSTM models (R2, R4) per input to output data combination. The best model in this comparison is the mixed input LSTM model (C4) with the best results in all metrics, marked with the blue font in the table.

In addition, it is apparent from this table that the R^2 score metric is less meaningful for sequential data output. The score is only slightly lower than the perfect score for all models, which is due to the fact that the variance for target and forecast values is almost the same. However, this does not consider the chronological order of variance in the sequence and therefore less significance for the comparison. For the LSTM models (R2, R4) the table finally continues the relationship from the previous section for adding different data type input, such that a combination as mixed data brings better performance than pure sequential input. Surprisingly, this does not hold for the TCN models (C2, C4). Most probably this is due to insufficient hyper-parameter optimization of the mixed input TCN model (C4) and thus worse scores than the pure sequential input model (C2).

Table 5.8: Comparison of sequence SOH estimation scores for varying models

Model	ID	Input	MSE (min.)	MAE (min.)	R^2 (max.)
LTSM	R2	sequence	1.4794	0.6453	0.9993
LTSM + ANN	R4	mixed	0.9927	0.5409	0.9995
TCN	C2	sequence	1.8786	0.9182	0.9962
TCN + ANN	C4	mixed	2.2646	1.1423	0.9969

Prediction Error Consideration

Regarding the prediction error, Figure 5.3 gives a comparison across the different sequential output models. The corresponding detailed statistical properties for each error distribution is appended in Table A.2 in Appendix A. The error is calculated as the sum over the full length of the sequence of the absolute error per time step. Hence, each sequence sample is evaluated as one prediction error value. Accordingly the distribution of all samples can be represented in form of a histogram.

For all distributions no strong shift is visible. In addition to the performance metric results before, the figure shows the more accurate performance of the LSTM models (R2, R4) compared to the TCN models (C2, C4) recognizable through their smaller spread in deviation, larger minimum values and higher Kurtosis values, representing a smaller prediction error. Accordingly, the LSTM models perform better in task of sequential SOH estimation.

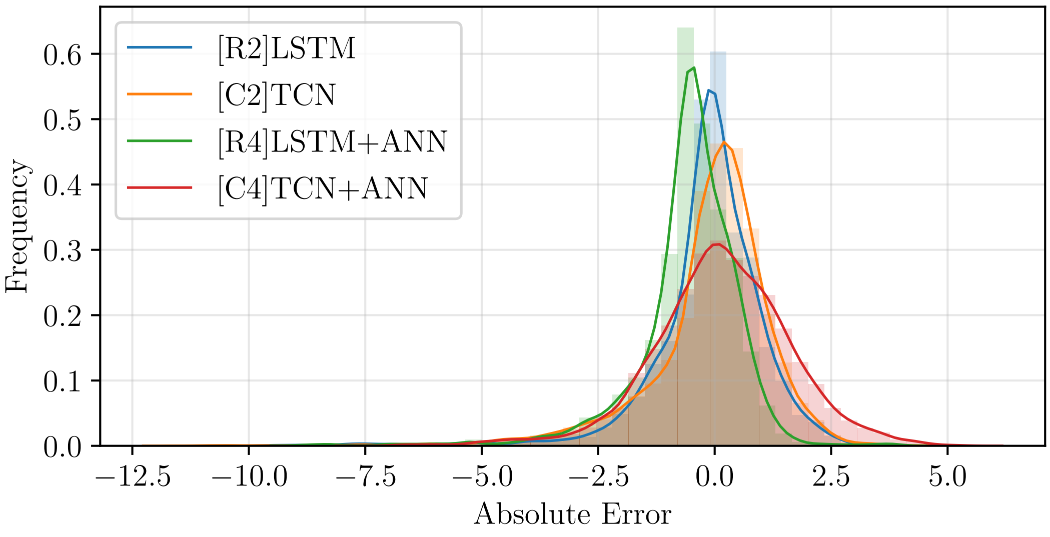


Figure 5.3: Error distribution comparison for recurrent (R2, R4) and convolutional (C2, C4) models on varying input for sequential output.

Sequence Residual Inspection

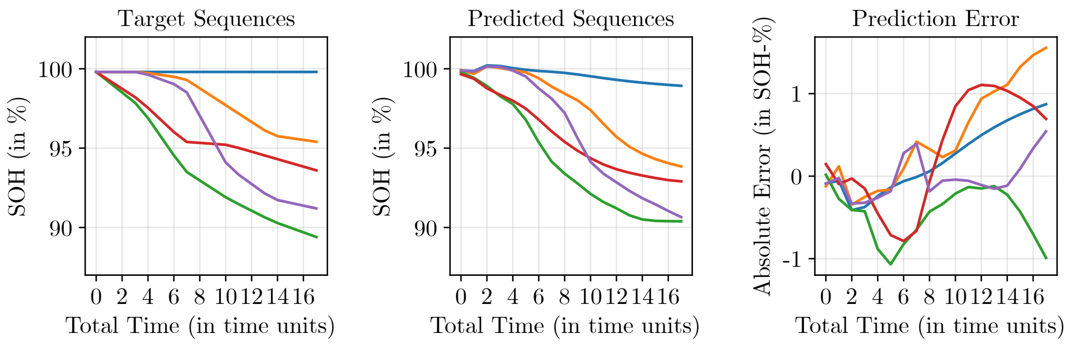
As equivalent to the residual plot for static output values, sequence residuals can be represented as error sequences, where the error is evaluated per time-step. In addition, a visualization of predicted sequence output helps to compare the performance visually.

Since it would not be an illustrative comparison to display all sequences at once, five sequences were selected semi-randomly from the test set. A random selection of five sample was repeated until these illustrated a broad spectrum of sequence diagrams, in order to be able to view the sequence diagram for as different time-series as possible. To allow a side-by-side comparison of all four models, the selected sequences were steady for all models. The color of the individual sequences remains constant over all sub-figures. The three column plots show the original five sequences on the left, the predicted sequence model output in the middle and the error per time-step on the right. The smaller the error per time-step, the better the prediction.

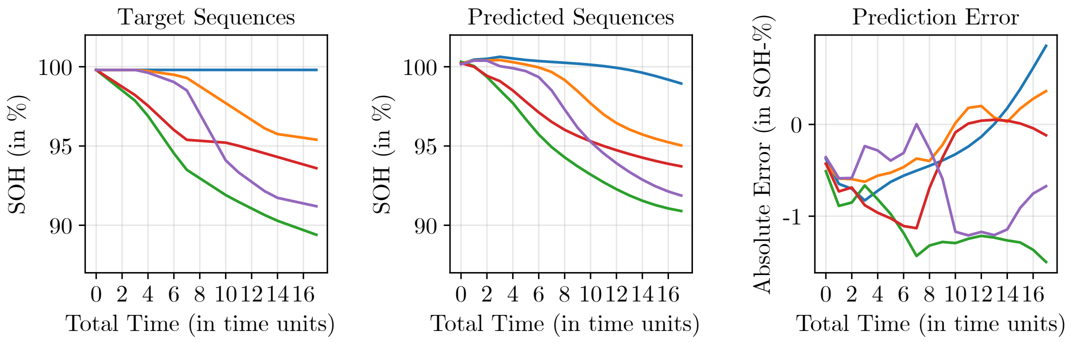
The first two rows in Figure 5.4 represent the results from the LSTM models (R2, R4). For both models, the predicted sequences show smooth graphs, without noticeable corners. Overall, the predictions approximate the targets closely, although the errors tend to increase over the length of the sequences. However, the prediction in the first four time-units is noticeable negatively for those target sequences that show no change in SOH during this period. For this period, the models predict an increase in SOH above a value of 100 %-SOH. This is not consistent with the target.

In contrast to the smooth predictions from the LSTM models, the predictions of the TCN models (C2, C4) appear trembling and oscillating. The sequences are recognizable as erratic. Especially the mixed input TCN model (C4) shows an erroneous trend for high SOH values over the first time-units, i.e. the blue curve already start with an SOH value over 100. However, both models show better results than the LSTM models for the constant blue curve, resulting in a zero error for the rest of the sequence.

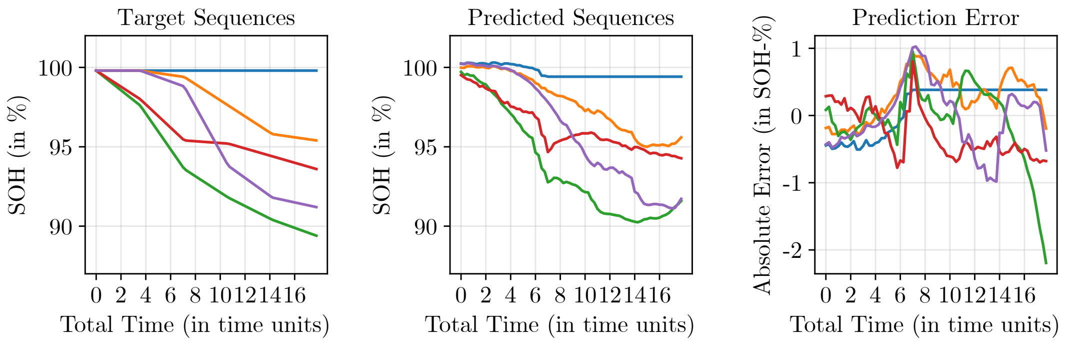
For all models in this comparison it is noticeable that the prediction for extreme cases, as represented in the examples by the blue and green curve, usually runs towards the mean of the data, thus creating the error for those examples. For instance, the green sequence predictions tend to slow down the decrease or even increase the value within the last time-steps in each model. This might be explained by the unbalanced amount of data, where extreme cases are less present in the data population. Hence, the models tend to predict in the direction where more data is available. Here, this direction is the rather moderate ageing, which could be the reason why the extreme cases tend to converge toward the center.



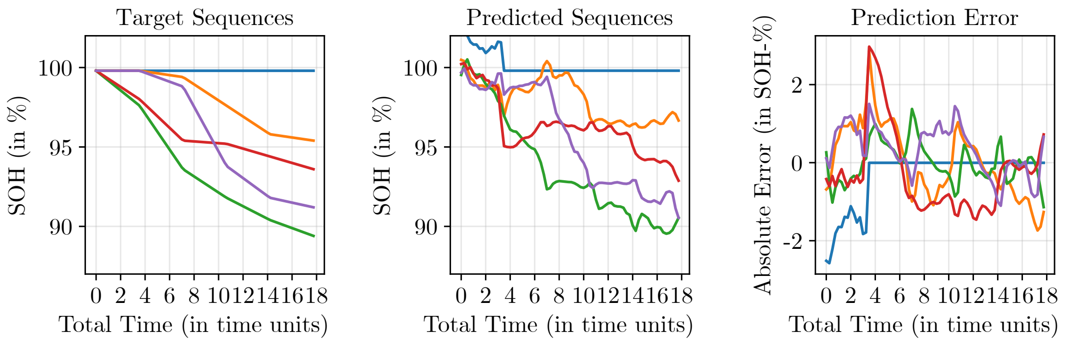
(a) Pure sequential input LSTM (R2)



(b) Mixed sequential and static input LSTM+ANN (R4)



(c) Pure sequential input TCN (C2)



(d) Mixed sequential and static input TCN+ANN (C4)

Figure 5.4: Comparison of target and predicted sequences, including the prediction error, for five sequence samples of the test dataset.

In summary, the presented sequence prediction models show promising results in performance metrics and prediction error distributions. However, as already expected, the actual resulting predicted sequences show significant differences in terms of smoothness and accuracy in extreme cases. Based on the results, the mixed input LSTM model (R4) appears as the best model for this sequence output comparison.

5.1.3 Recursive Step Estimation

The recursive model stands out among the previously considered estimation models, because it requires more complex data preparation and post-processing. The model follows the stateful windowed step model approach from Figure 4.9 in Section 4.3.2. The sequential input data is pre-processed with the sliding window transformation, resulting in the set of static input data vectors per window for each sample. Hence, a static input to static output model was required and the ANN model (B4) has been selected as the best performing model evaluated in Section 5.1.1. No further hyper-parameter optimisation has been conducted.

Metric Performance
During training, the step model results in metric scores on windowed data samples from the test dataset with significantly low error over all metrics (see Table 5.9).

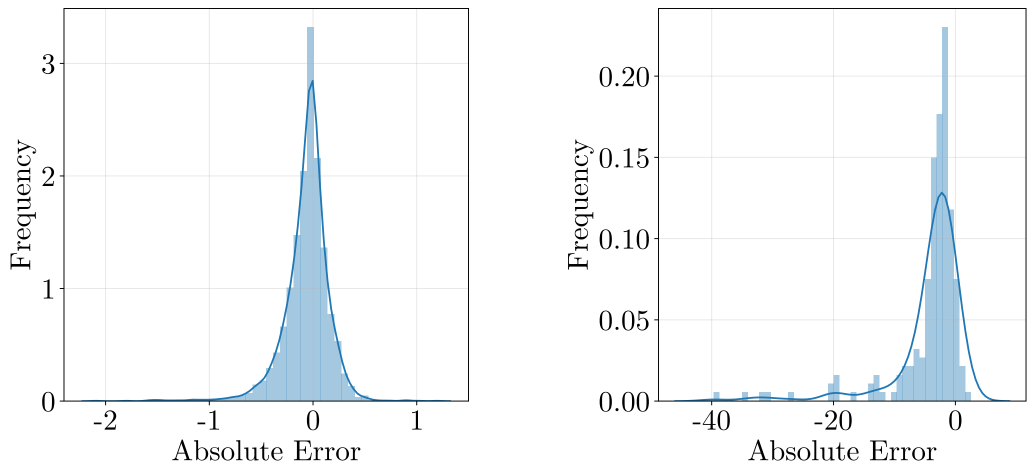
Table 5.9: Training metric performance on static SOH estimation scores for the step model

Model	ID	Input	MSE (min.)	MAE (min.)	R^2 (max.)
Step model	S1	sequence (windowed)	0.0531	0.1512	0.9903

However, this training performance does not apply to a similar sequence length of consideration as in the previous section, due to the training on the windowed data in contrast to the full length sequences. To compare with the sequence models, the input data windows for each sample of the test dataset are used in a specific evaluation method. In this method, the input is initialised with a 100 %-SOH value in the first window of the sequence. For each following input data window the SOH prediction of the previous window is added to the input data and the full prediction length is obtained by iterating over all windows of the sequence. Naturally, due to potential error propagation, this does not lead to similar performance results for the entire sequence length as the per step perspective from Table 5.9.

A comparison of the both perspectives is shown in Figure 5.5. Supporting statistical properties for the prediction error are appended in Table A.3 in Appendix A. The left plot in Figure 5.5 shows the per step prediction error on the windowed test dataset. The errors appear mostly below an absolute value of 1 but tend to be negative. Hence, the model is underestimating the ageing to some extent. In the right figure, the full sequence prediction error on the test dataset shows a significantly larger error, occurring almost completely on the negative side. Naturally, the error for the entire sequence is higher, since the error per step is propagated when the faulty output is used as input for the subsequent time-step.

The figure indicates an error increment rate of around 10 between the left and right plot, with the exception of error outliers up to the value of −40 %-SOH. The rate factor can be explained by the average evaluated sequence length of 10 to 20 steps. Hence, in the worst case an error is propagated up to 20 times. However, for the outlying extreme values of more than 10 %-SOH prediction error, it must be noted that the complete SOH in the data only has an absolute value variance of 16. The outliers are predicted completely wrong, most probably due to the error propagation.



(a) Per step error

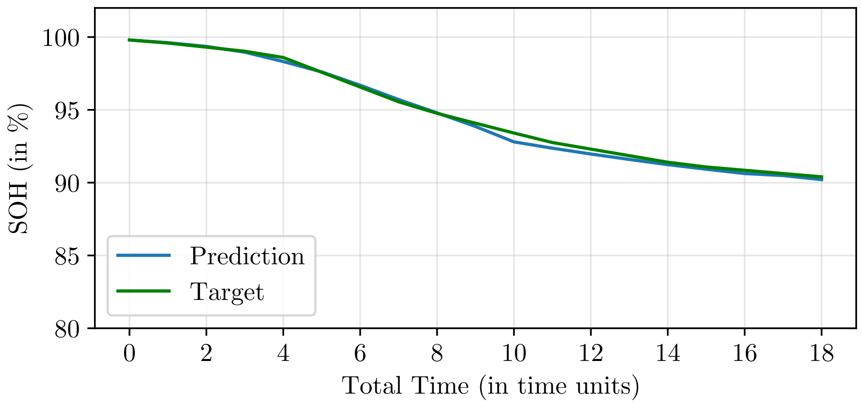
(b) Per sequence error

Figure 5.5: Comparison of prediction error for sequence data in step model.

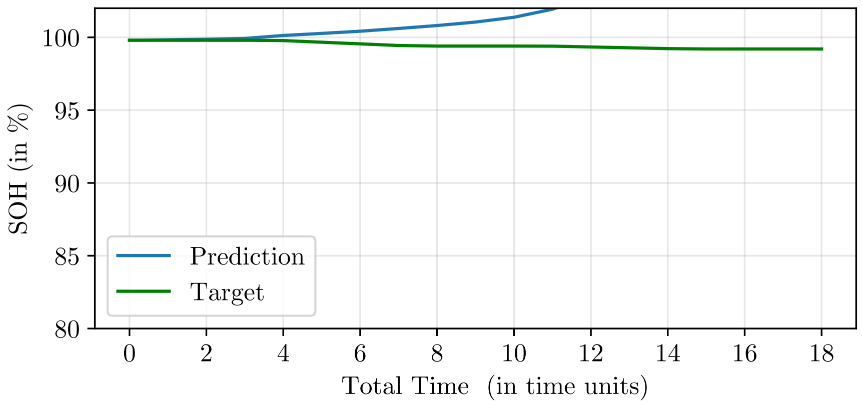
Full sequence residuals

According to the sequence residual showcase in the previous chapter, Figure 5.6 presents two predicted sequences from the step model compared to the target sequence. The selected samples represent the lowest and highest prediction error for all samples in the test dataset to allow a broad review of the results.

The top sample shows an accurate prediction over the full sequence. There is hardly any visually discernible difference between target and prediction visual. However, the sequence with the worst prediction error indicates a clear incorrect and unusable sequence prediction. Presumably this behaviour results from error propagation, i.e., if an extremely high error is propagated a few steps further and thus the input leaves the range of learned input values. This indicates a poor generalization of the model on the data.



(a) Prediction sample with low prediction error.



(b) Prediction sample with highest prediction error, unstable behaviour.

Figure 5.6: Comparison of prediction error for sequence data in step model.

5.1.4 Summary of Estimation Findings

As a conclusion, the results from the last three sections are summarized. For each considered input output combination approach, only the best model is presented in Table 5.10. The models were selected according to the number of leading scores in each metric, the more leading scores the better. All other models and their results are skipped in this comparison.

It is important to mention, that a presence of a low error value is not sufficient to select the best approach and can be deceptive if error distributions and residuals are not considered. Each approach and corresponding models has its own merits and challenges, which must be weighed up depending on the application in order to make the selection.

On the assumption that the data available is representative, Table 5.10 indicates a clear association that mixed data input outperforms pure static or sequential input. For both output formats, the models with mixed data input have the lowest MSE and MAE error and the highest R^2 score. In addition, to the examination of the metric performance, the examination of error diagrams and residual plots in the previous sections indicates the mixed input LSTM and ANN model as most suitable and reliable candidate for the task of predictive SOH estimation, both for a static and sequential prediction output format.

Table 5.10: Comparison of static SOH estimation scores for varying models

Model	ID	Input	Output	MSE (min.)	MAE (min.)	R^2 (max.)
ANN	B4	static	static	1.3232	0.7779	0.9178
LTSM	R1	sequence	static	1.4392	0.7996	0.9106
LTSM + ANN	R3	mixed	static	1.0032	0.6825	0.9377
LTSM	R2	sequence	sequence	1.4794	0.6453	0.9993
LTSM + ANN	R4	mixed	sequence	0.9927	0.5409	0.9995

5.2 Data Reduction Analysis

The procedure of feature selection as described in Section 4.2.1 is an important step to improve the training efficiency regarding training time and inference time. In addition it may also improve the model performance in evaluation metrics if relevant data can be extracted and irrelevant, redundant or even miss-leading data can be excluded.

However, the implementation of a selection will always shrink available input data and thus can unintentionally exclude relevant information for the model to identify the underlying system dynamics mapping function. For instance, if the goal is to learn a complex non-linear function and available training data is reduced, this reduced data might just lead to a simplified, even linear version of the complex dynamics, hence the approximation would be unsuitable. This is the reason why the models in the last section have been trained and evaluated on the full available data to evaluate the existence and approximation with a model in a first stage. Due to the satisfying results from the previous section, as a second stage, this section evaluates what performance can be achieved when data is reduced.

The results of the filter selection procedure are presented in the following sections. Due to the mixed data types available in the data basis, the feature selection has been applied to the time-series and feature-based data separately, divided in Section 5.2.1 and 5.2.2 respectively. Moreover, in Section 5.2.3 a model was trained on multiple subsets according to the selection to evaluate the performance of reduced feature subsets as results of the selection procedure. For this evaluation, the mixed data LSTM (R2) was selected, being the best model from the previous section.

According to the method described in Section 4.2.1, , the Spearman correlation coefficient and regular PCA was used for the procedure.

5.2.1 Time-series Data Reduction

As the first step of the selection procedure, the correlation coefficients were computed pairwise for each time-series data feature. Therefore, the time-series for each feature were concatenated for all vehicles to receive one long column per feature. Time-series which are internally used in the BMS to directly calculate the SOH are excluded, in this case the available energy, 0C and 1C capacity. No further feature grouping was applied, since only 9 features remained.

Figure 5.7 shows the Spearman correlation result represented as combined correlation matrix plot. On the left side the cross-correlation matrix is plotted for all selected time-series data features. Since the correlation is computed pairwise the computed coefficient of one feature pair is shown twice in the cross-correlation matrix plot, above and below the diagonal entries, which represent the identity (1.00) between the same feature. The right side of the figure shows the correlation of all time-series data features with respect to the target time-series, the capacity SOH time-series. Both plots share the same color bar scale. Deep blue cells correspond to a strong positive correlation (≈ 1) and deep red to a strong negative correlation (≈ -1). The lighter the color, the lower the correlation, down to correlation-free, which is recognizable as whitish.

On the cross-correlation side, a high positive correlation (> 0.82) is presented between the different Energy Throughput features, representing the accumulated throughput while charging, regeneration and driving. Additionally these features correlate weakly negative with the minimum and maximum 2s Resistance features. Another relevant cross-correlation cluster can be seen between charge and discharge resistance ($\approx |0.3|$), where the 20% SOC Resistance among the others shows no correlation. All other cross-correlation coefficients show no relevant values.

For the correlation with respect to the target time-series, the SOH capacity series, the correlation column on the right of the figure indicates a strong positive correlation ($0.64 - 0.82$) between the energy throughput features and the SOH capacity time-series. Furthermore, a weaker negative correlation (-0.21) is indicated between the maximum 2s Resistance and the target time-series. In summary, this confirms the theoretical background that resistance and capacity change in opposite directions, the former increasing and the latter decreasing with increasing age. Hence, the available data population represents the effects of chemical and physical ageing processes from Section 2.1.3.

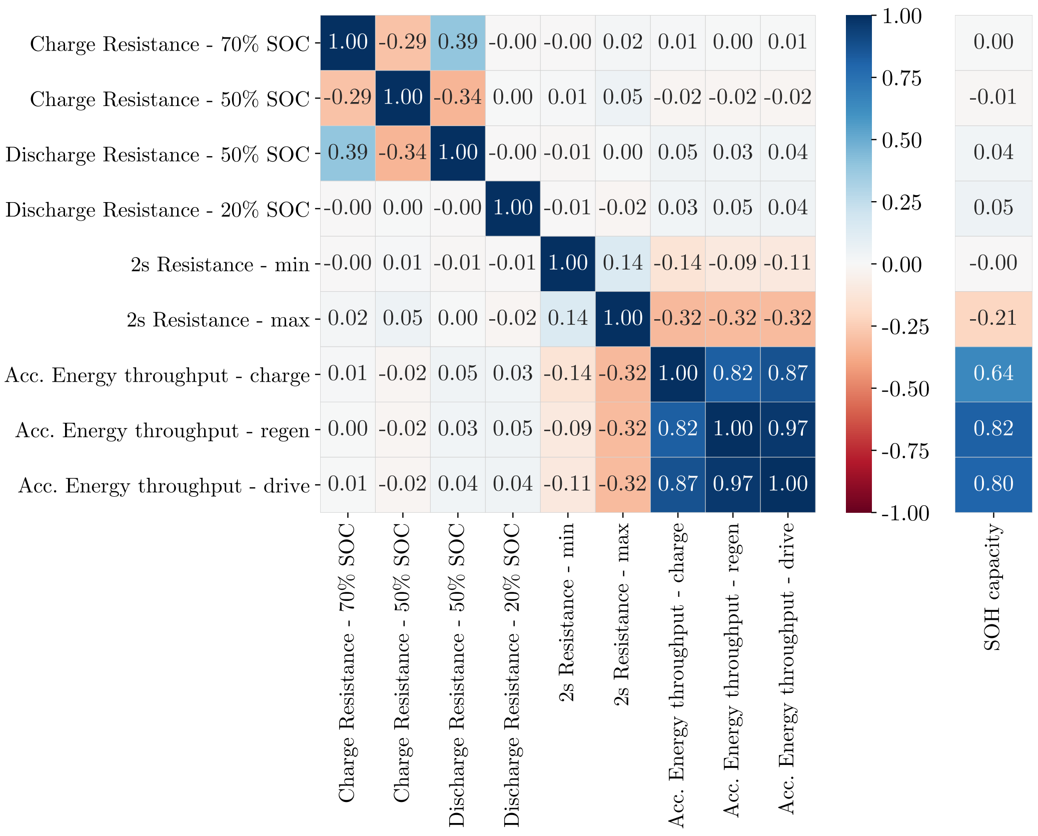


Figure 5.7: Combined Spearman correlation matrix plot for time-series data.

Following the feature selection procedure, PCA was subsequently applied to reduce the dimension of input data. Therefore, a common threshold of 90% explained variance of the data was used for the reduction. PCA delivered a total number of 3 required components to retain an explained variance of 92.25%, while reducing the dimension from 9 to 3 dimensions. Following this reduction decision, PCA was used to identify the features that contribute most to the selected number of three components. Table 5.11 lists the most important features given by the PCA. Only accumulated energy throughput time-series remain. However, when looking at the correlation in Figure 5.7, it is clear that the remaining three features are at least partially redundant and a prior selection of non-redundant features could prevent this.

Table 5.11: Most contributing time-series features for a remaining variance over 90%

Features
Accumulated Energy throughput over time - during charging
Accumulated Energy throughput over time - during regeneration
Accumulated Energy throughput over time - during driving

5.2.2 Static Data Reduction

The reduction analysis for static data was performed on the grouped static features generated and grouped as described in Section 3.2.2.

To avoid the correlation analysis between all 80 generated features, and therefore a 80×80 sized matrix, a pre-analysis has been done on each feature group to already exclude irrelevant features, before analysing the correlation between remaining features of all groups. For this purpose, two strongly correlated features with respect to the target features were selected from each group, while considering that the cross-correlation between them is low, to keep redundancy low between the input features from the same group. The correlation matrix plots for all groups, used for the pre-analysis, are given in Appendix B.

Figure 5.8 shows the resulting combined correlation plot for a maximum of two generated features per feature group. On the right part of the figure, the correlation with respect to the target is shown. Each feature is named following the convention "*name of histogram - extracted statistical feature*", for example "Depth of Discharge - sum" is the extracted sum of the depth of discharge histogram for each vehicle.

Due to the pre-selection of highly correlated features from the groups, the plot indicates high correlation on almost every feature, whereby the cross-correlation is in general lower than the correlation with respect to the targets. Outstanding is the cross-correlation of minimum temperature during charging, which is the only negative correlation to all other features. In addition, it is apparent that both depth of discharge features, sum and cycles, share the exact same correlations and their cross-correlation is 1. Hence, these features are redundant and one feature can be removed.

With respect to the target, on the right side of the figure, the high correlation in every feature is noticeable. No correlation between feature and the targets is below |0.5|. Additionally, it can be directly seen that the extracted two target features (SOH minimum and difference) correlate in the completely opposite direction. This holds for the SOH as long as the minimum SOH is one of the last values in the time-series, so both variables describe the same variance bidirectionally. This confirms the expectation, that only one target feature is sufficient as target.

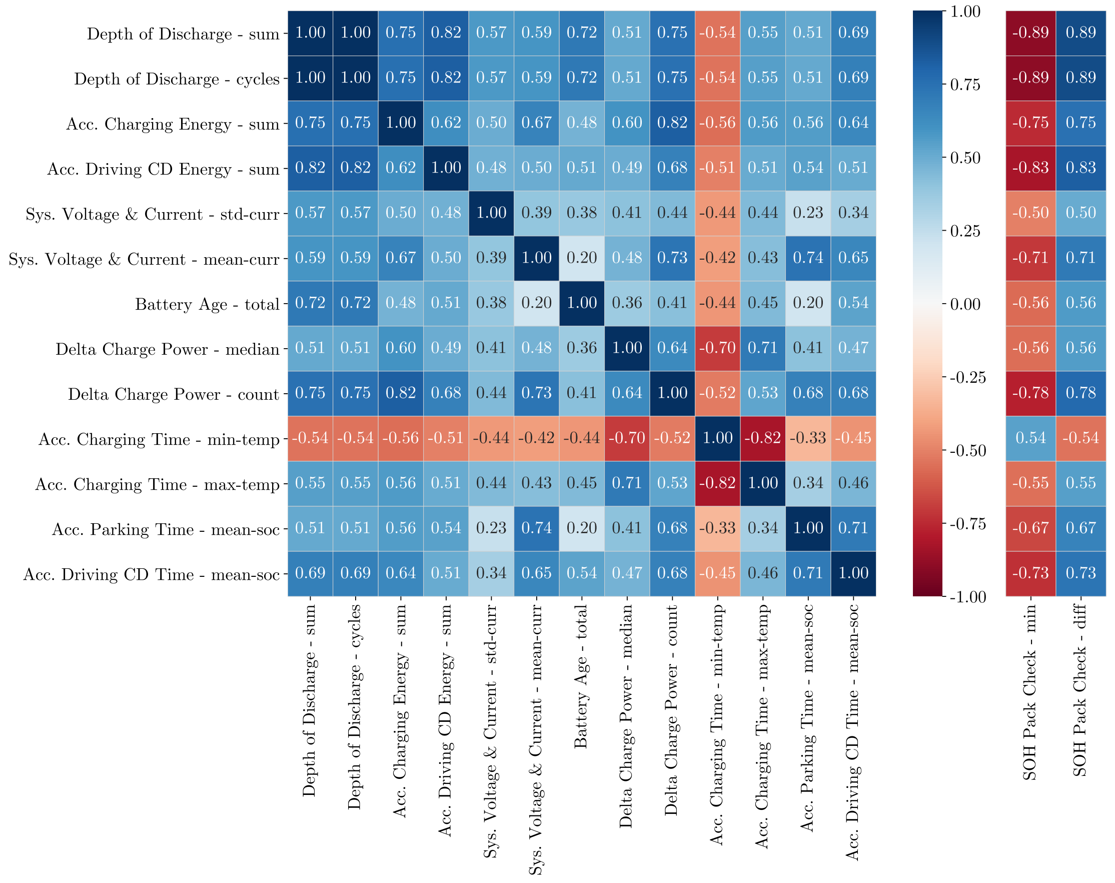


Figure 5.8: Combined Spearman correlation matrix plot for generated feature data.

Following the feature selection procedure, PCA was applied on the remaining feature subset of 13 features, following the pre-selection with two top, redundancy-free correlated features per group. As comparison PCA was additionally applied on the full feature set of 80 features. Table 5.12 shows a summary of the PCA component analysis for the two feature sets. The total number of required components per analysis was chosen to retain an explained variance over 90% per subset. The resulting explained variance for the selected number of components is listed as well. However, this comparison must be considered with caution, since the explained variance only applies on the number of total components of the selected subset. Hence, an evaluation of the effect of feature reduction on the training performance is presented in the next section.

Table 5.12: PCA component result for histogram feature subsets

Feature subset	Total components	Needed components for variance > 90%	Resulting explained variance
All generated features	80	21	0.9034
All pre-selected features	13	5	0.9905

Finally, PCA was additionally used to identify the most important features through their contribution on a selected number of components. Table 5.13 lists the most important static features given by the PCA contributing to the required 5 components. The remaining features consider the system current, charging power, charging temperature and battery age. On a wide extent this corresponds to the known chemical and physical ageing factors presented previously in Section 2.1.3, as this selection contains information about the important degradation mechanisms of over-charging, battery temperature, current and battery age.

Table 5.13: Most contributing histogram features

Features	
Histogram Name	Statistical Property
System Voltage and Current	std. deviation of current
Delta Charge Power	limit violation count
Battery Age	total number of months
Acc. Charging Time over SOC and temperature	maximum temperature
Acc. Charging Time over SOC and temperature	minimum temperature

5.2.3 Feature Reduction Analysis

The general filter approach for the feature selection, as performed in the previous two sections, is a quick and intuitive procedure. However, due to the fact that it is only performed on the data, dependencies on the learning algorithm, or the model approach, are not considered. To investigate the impact on the network training and resulting performance, this section presents a comparison of different data subsets.

The evaluated subsets are based on the selection results from the previous sections. A list of all evaluated subsets including the number of features is presented in Table 5.14. Besides the full available data and the full reduced selection result (combined sequential and static from the previous two sections), intermediate selection results only from the correlation analysis and only from the PCA were added to the comparison. Furthermore, the use of pure sequential and static features has been evaluated as well.

Table 5.14: List of evaluated feature selections subsets

Subset name	Sequential feature count	Static feature count
All features available	9	80
Full selection result	3	5
Correlation result all features	3	13
PCA results all features	3	21
Only sequential features	9	0
Only static features	0	80

For the comparison, the overall best model with mixed input opportunity has been selected based on the comparison in Section 5.1.4. The mixed input LSTM model (R3) was selected for the evaluation, on a five-fold cross-validation setup. As metric for the comparison MSE has been selected, because it is used during training for the optimization.

A summary of the results is presented in Table 5.15. In terms of metric results, the model trained with full available data performs best. This is followed by the models trained with pure static and pure sequential data. The remaining three models, trained with mixed reduced data with varying reduction, performed worse with respect to the minimum MSE. In addition, they show unstable training results over multiple cross-validation models per data subset. Thus, the effect of a high MSE score for one of the five trained models per data subset can be identified with the significantly higher mean MSE score.

There are two possible explanations for this: first and most likely the remaining training data is insufficient for the model complexity or the model architecture is unsuitable for training problem. Secondly, the random initialisation of weights and biases may have lead to a local loss minimum and the optimization fails to reach the global optimum. This is also facilitated by an inappropriate model architecture.

Table 5.15: Error and duration results comparison for the evaluated feature selection subsets

Name	MSE (min.)	MSE (mean)	Training duration
All features	1.0032	1.5082	5.9min
Full selection result	3.0527	3566.8720	6.89min
Correlation result all features	2.0649	1784.4278	9.79min
PCA results all features	2.0681	3565.5066	8.49min
Only sequential features	1.4794	1.8577	8.5min
Only static features	1.3231	1.4438	4.2min

The training duration consideration does not indicate a clear result. Presumably, this is caused by the early stopping method, leading to an inconsistent training duration over the cross-validation fold. Furthermore, an additional hyper-parameter optimization could change the training duration times significantly. Finally, the random initialization for the weights in the model can lead to getting stuck in a local minimum of the objective function during training. This is more likely to occur when data is reduced, since over-fitting often occurs when the model complexity is inappropriate high for the given amount of training data. In total, the significance of the duration consideration is limited.

5.2.4 Selection Analysis Summary

In conclusion, for the available data the feature filter approach delivers a reduction from 9 to 3 time-series features. According to Figure 5.7 and Table 5.11 the most important features are the three accumulated energy throughput over time features. This corresponds to the theoretical considerations of chemical and physical ageing factors from Section 2.1.3, so the data represents an adequate population for these factors.

For the static features the selection result is not unique and depends on the selection parameters, i.e. the number of high-correlated features chosen. Therefore, different selections have been evaluated. The most extensive reduction was achieved by using correlation-based filtering in combination with subsequent principle component analysis. This reduced the total number of 80 static features to 5. According to Table 5.13 the most important features derived from the data, are deviation of system current, number of charge power limit violations, battery age, minimum and maximum temperature during charging. This repeatedly correlates with the chemical and physical ageing factors.

In contrary to expectations, the reduction has failed to achieve effective results in the sense of training efficiency and consistent performance metric results. According to Table 5.15 the reduction analysis on the selected subset of data revealed uncertainties in generalization of the model. No consistent error could be obtained for the five trained cross-validation models per data subset. All three filtered subsets had minimum one failed model training, where the model showed insufficient learning.

There are several possible explanations for this result. First, it may be explained by the fact that the selection approach was inappropriate or has been applied in an over-reducing manner. In addition, the total amount of available data might be sufficient for training when all data is considered, but is insufficient if only a subset is considered. Finally, improper hyper-parameters of the used model may have caused this issue and further optimization might have been necessary.

In summary, for a final assessment further investigations are required. However, the achieved performance values for a low reduction and the partially successful performance for an extensive reduction indicate that a reduction is feasible.

5.3 SOH Prediction Application Performance

This section attempts the predictive application showcase, as introduced in Section 4.3.3, by utilizing the in Section 5.1 evaluated models to predict the future SOH of an individual vehicle based on its history.

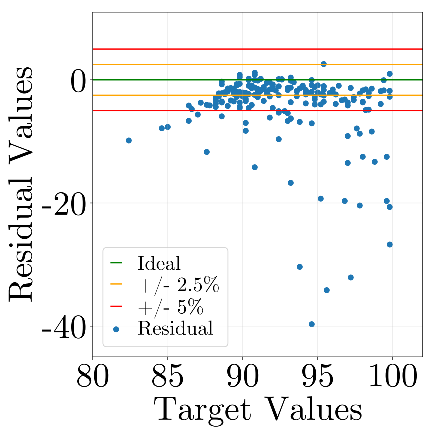
According to the introduced showcase, only sequential data was considered. However, the step model with windowed data input allows a transformation from sequential to static data values. Hence, for the showcase two models have been evaluated: the step model and the sequence to sequence LSTM model, presented in the following sections 5.3.1 and 5.3.2, respectively.

5.3.1 Iterative-Step Sequence Prediction

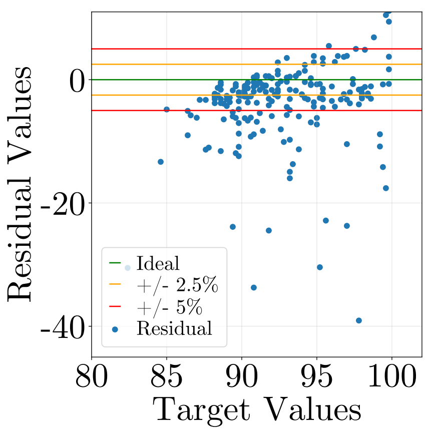
First, the results for iterative sequence prediction with the step model approach are presented. For this purpose, the implemented step model (S1) from Section 5.1.3 was used, following the windowed data transformation of sequential data to static values. According to the difference equation (Equation 4.4) the output of each prediction is reused as input for the following time-step. Thus the sequential output is given as the concatenation of the network output per step.

Last Time-step Residual Inspection
The prediction application is considered as comparison between the full sequence input and the half sequence input with modified second half input, according to the introduction of the showcase in Section 4.3.3. Hence, two different input datasets are given for the comparison. Although the step model additionally allows a consideration of the per step residual, only the residual of the last time-step of each sequence was taken into account for this comparison. This has been done to be able to compare the model results with the LSTM model, which only allows the last time-step comparison.

Consequently, Figure 5.9 compares the last time-step residual plot for both input variations. The comparison shows a significant difference in the residual values. The half sequence input shows a generally wider spread. This is most probably due to the fact, that the considered dataset does not fulfill the constant usage assumption and thus a simple usage propagation as for the half sequence is more likely to not be predicted correctly. Besides this difference, between both input variations, the residual plots show high residuals with extreme values of -40%-SOH. This indicates a poor performance of the model, similar to the performance of the step model in the previously comparison in Section 5.1.3. Due to the fact that the same model from that comparison is reused in this section, the large residuals most likely occur due to the error propagation problem.



(a) Full sequence input



(b) Half sequence input

Figure 5.9: Last time-step residual comparison for the step prediction model.

Prediction Error Consideration

For a more detailed look on the prediction difference between full and half modified sequence input, Figure 5.10 presents the distribution of absolute error between the predictions for both data input variations, for an average sequence length of 16 time-units. The difference is defined as the half sequence input prediction subtracted from the full sequence input prediction $\hat{y}_{full} - \hat{y}_{half}$ for each sample in the test dataset. Surprisingly, the distribution is more equally distributed on positive and negative error direction. An explanation for this might be that for a rather same amount of sample in the test dataset, the prediction for the full sequence length worked well, while the prediction for half the sequence length failed, and vice versa. However, the absolute error value up to ±30 %-SOH confirms the previously observed overall poor performance of the step model.

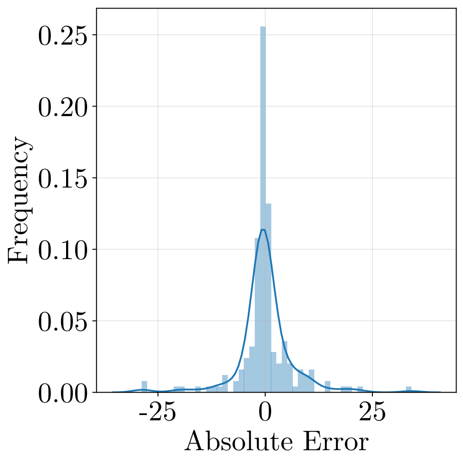
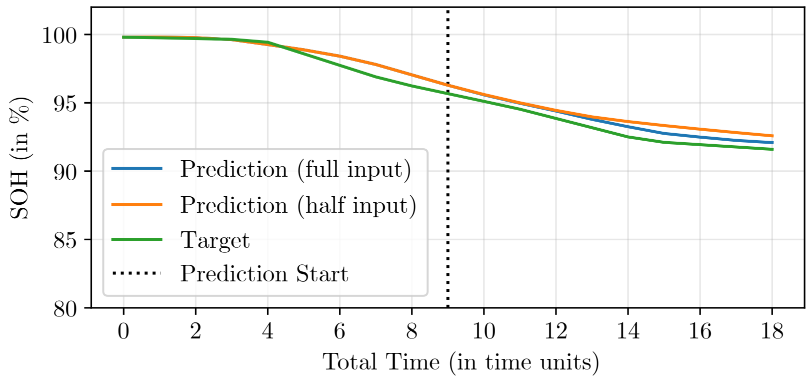


Figure 5.10: Last time-step deviation error distribution between full- and half-length sequence prediction.

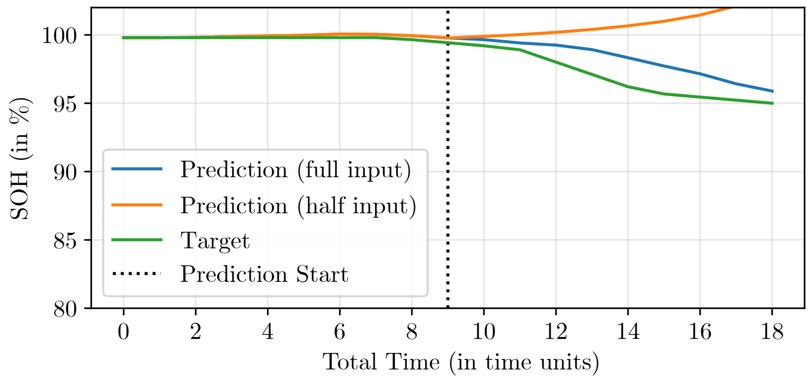
Sequence Inspection

In order to allow a visual interpretation of the achieved residual performance, the following Figure 5.11 presents the prediction results of the step model for two samples from the test dataset. The samples are picked by hand to present a positive and a negative performance example from the step model for the SOH prediction showcase.

In the upper figure the positive performance sample is presented. Both input variations only show a low deviation from the target sequence. However, with increasing time the difference between the input variations increases as well. Unsurprisingly, the prediction for both input variations before the prediction start are the same, since the input data only differs beyond this point in time. In contrast, the bottom sample shows a negative performance for the considered half sequence input prediction. The step model shows an incorrect prediction for the modified half sequence input, presumably due to the poor generalization and the error propagation issue. Surprisingly, the prediction with full length unmodified sequential input shows a rather correct prediction, although the prediction error is large between time-unit 12 and 16.



(a) Prediction sample with low prediction error.



(b) Prediction sample with highest half sequence input prediction error.

Figure 5.11: Comparison of sequence predictions from step model for half and full sequence input.

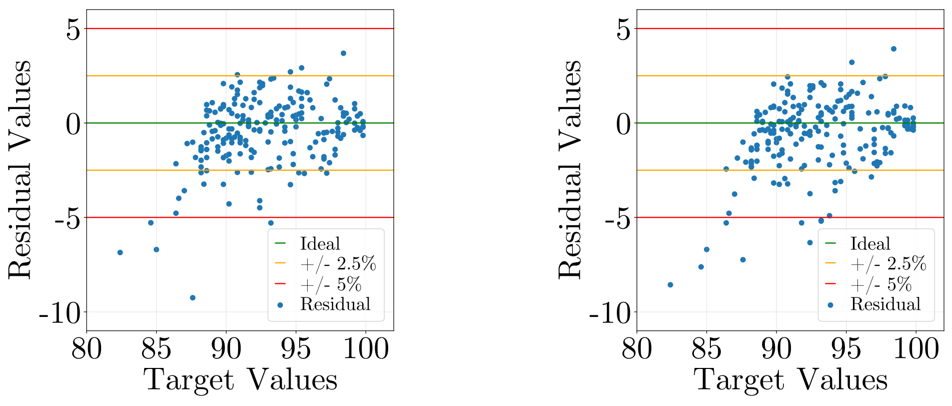
In summary, this section showed that the step model (S1) in the implemented configuration for the available data is not a suitable solution. As expected, the error that already occurred in the estimation of the SOH value with full available data has further deteriorated in the evaluation of the half sequence input for on the blind usage propagation showcase. Further detailed examination of wrong predicted samples could provide an insight on whether the constant usage assumption is the decisive factor.

5.3.2 Multi-Step Sequence Prediction

As second prediction approach the full sequence to sequence prediction has been evaluated. Therefore the sequential input LSTM model (R2) has been selected from the evaluation in Section 5.1.2. As with the step model, the input data was modified for the prediction showcase. In contrast to the step model, this full and half sequence input data has been used directly in the model, without a window transformation into static values.

Last Time-step Residual Inspection

Following the evaluation procedure of the previous section, the prediction residuals are first given in Figure 5.12. In contrast to the results of the step model in Figure 5.9 of the previous section, the residuals are significantly smaller. The majority of the residuals appears smaller than ±3 %-SOH. Only a few outliers have an absolute residual value above 5 %-SOH, especially in the lower target SOH range. In addition, the residuals are more equally distributed around the ideal residual value straight line at $y = 0$.



(a) Full sequence input **(b)** Half sequence input

Figure 5.12: Last time-step residual comparison for the sequence prediction model.

Prediction Error Consideration

Taken together, the difference between residuals on full and half-length sequence data is illustrated by Figure 5.13, where the error is defined as difference of the half sequence input prediction subtracted from the full sequence input prediction $\hat{y}_{full} - \hat{y}_{half}$.

The significant difference to the results of the step model in Figure 5.10 of the previous section, the error between the different data considered is substantially smaller. Besides the peak at zero, the figure shows only an absolute error of up to ± 2.5. This small difference for the varying data, can potentially be explained in two different ways. On the one hand side, due to the fact that the data in the test dataset meets the assumption that vehicles are used consistently heavy or that the main variance in the usage of the vehicles only occurs during the first few time-steps and the shrinks afterwards. Hence, a propagation of the input data from a certain time-step can achieve these rather accurate results for a blind prediction. On the other hand, the underlying supervised learning approach always prones to over-fit to the training data, due to the teacher learning environment. Hence, if training data and test data are rather similar, then the prediction error on the test set should appear low.

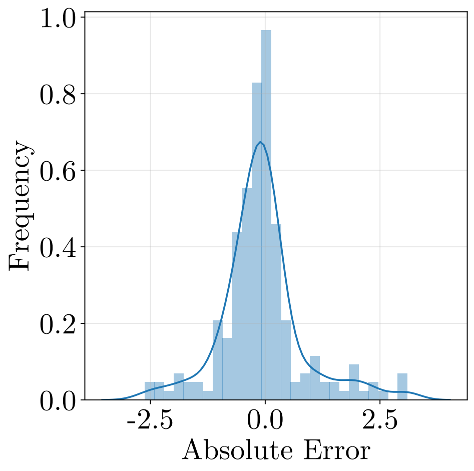
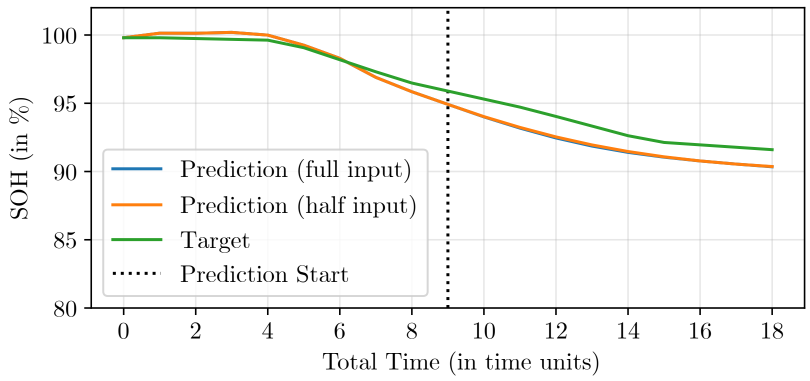


Figure 5.13: Last time-step deviation error distribution between full- and half-length sequence prediction.

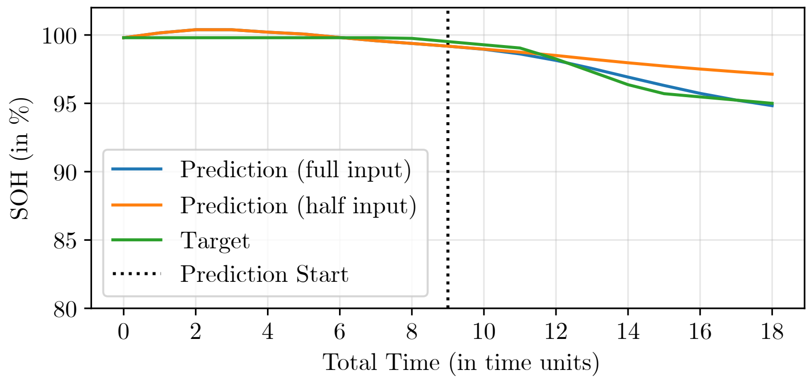
Sequence Inspection

For the visualization of the predicted output sequences the same vehicle samples were selected as for the step model. The result is presented in Figure 5.14.

The top figure repeatedly indicates an accurate prediction. Besides the expected same prediction before the prediction start time-step, there is hardly any noticeable difference in the prediction on full and modified half sequence input. In contrast, the bottom figure shows a noticeable difference between full and modified input. However, this difference seems reasonable if the available data for the prediction is considered. It seems like the usage has been consistently low in the first half of the sequence. It is assumed that this changed after roughly 10 time-units, since the SOH target decreased beyond this time. Unsurprisingly, the prediction with the full sequence input can capture the change and continuously predict the SOH accurately. In contrast, the blind usage propagation can naturally not predict this change and, hence, is unable to predict accurately.



(a) Prediction sample with low prediction error.



(b) Prediction sample with high prediction error.

Figure 5.14: Comparison of prediction error for the sequence model.

Compared to the step model in the section before, the sequence to sequence LSTM model showed significantly improved results for the considered prediction showcase. Nevertheless, for instance the prediction sample in Figure 5.14b reveals the limitations of the presented prediction application, due to fact that it was not designed to include the prediction of the future driver behaviour and assumes it to be constant. This will always remain a crucial disadvantage of the model architectures presented, since it is unlikely that linear propagation works well in a prediction task.

5.3.3 Summary Prediction Findings

In summary, in this section two models from Section 5.1 evaluated SOH estimation task have been selected to investigate a basic SOH prediction application, based on usage propagation of historical usage per vehicle.

On the basis that a SOH prediction is generally demanding because assumptions about the future use have to be made, the models evaluated have shown simple results. If these assumptions do not hold, because the vehicle is moved in a fundamentally inconsistent way beyond a prediction start, the models naturally show a deviation, i.e. as shown in Figure 5.14b. However, if the assumption holds and the vehicle is used consistently, for example as a commuter or taxi vehicle, then the models presented can offer an appropriate prediction, as indicated in Figure 5.14a.

There is one more drawback of this method for SOH prediction, besides the constant usage assumption. Data of the entire life cycle must be available for the training, in order to allow the training with sequential data in the supervised learning framework. Although, this data might be collected from fast ageing laboratory tests or prototyping drive test, the method seems difficult to realize. For instance, if the software is developed together with the hardware, the required test data would not be available during development. Consequently, either the development process or the method have to be modified. This clearly reveals the limitations of pure supervised learning methods for the predictive ageing task and opens the scope to investigate unsupervised or hybrid learning methods.

With an intensified focus on the predictive usage propagation, rather than assuming consistent behaviour, i.e. in a combination with a driving behaviour prediction model, this model could produce interesting findings in further research.

6 Conclusion

In this thesis, an appropriate tool chain has been established, that is capable of handling big data as input and delivers the desired state of health (SOH) predictions as output for vehicles in a fleet. The constructed pipeline has been used to investigate the feasibility, effectiveness and accuracy over a wide range of data-driven modelling approaches and model architectures.

For this purpose, CEVT AB provided diagnostic battery readout data of an entire plug-in hybrid vehicle fleet, including static and sequential data types. This gathered data was analyzed intensively and features have been extracted in order to serve the data in a compact representation. It has been found that the extracted feature data confirms the theoretical background of the chemical and physical internal battery processes for ageing. Hence, the provided features appear as an appropriate population for the investigation of the battery ageing problem. In addition to this feature analysis, a selection filter approach was applied on the features to separate relevant from redundant or irrelevant data. This would improve portability due to the reduced data volume, in the context of an in-car application. However, the selection approach was found to be inadequate, since the training on the resulting selection became unstable. A possible explanation for this could be that the remaining reduced amount of data was insufficient for the training.

The conducted model comparison revealed significant deviations between the considered model architectures. The pure linear models, closed-form linear regression and elastic net, have not been able to provide accurate results. Hence, it was assumed that models have to be capable of non-linearity for the battery ageing problem. To this end, artificial neural networks (ANN), long short-term memory (LSTM) and temporal convolutional networks (TCN) have been identified as suitable candidates.

Through the possibility of utilizing static and sequential data as input and output format, multiple format combinations could be investigated. The comparison has clearly identified the best results when using a mixed input format consisting of static and sequential data instead of using them individually. To this end, among all models considered, the combination of LSTM and ANN branches in a model to allow mixed input data, appeared as the most accurate model, for both the static and the sequential SOH output format. The static output SOH determination based on the historical usage of the vehicles achieved a mean absolute error (MAE) of

0.68%-SOH. For the sequential output format, the MAE was calculated as single value for the full output SOH sequence. In this case, the LSTM and ANN mixed input model achieved a MAE of 0.54%-SOH for the fleet dataset. This evaluation indicates the utilization of usage history fleet data for the SOH determination per vehicle is feasible and reliable.

Finally, a selection from the evaluated models for SOH determination was applied in a basic application for SOH prediction. Therefore, the average usage was linearly propagated per vehicle from half of the sequence and the resulting SOH output was compared to the full length available target. For this simple test case, under the assumption of constant use, the models evaluated showed a clear positive correlation to the correct results. However, this assumption of constant use will not be valid in all real use cases.

6.1 Further Research and Improvement

Reason for this thesis has mainly been an exploratory endeavour, seeking to investigate new usage of already collected diagnostic readout data. Although the data that could actually be used is only a fraction of the available data, the work has still given rise to topics for further work on utilizing field big data. During the project, three main areas for further research have been identified.

First, the established basic pipeline including feature selection, can be improved. In this thesis, only statistical properties of the diagnostic histogram data have been investigated. However, histograms contain more information, due to their condensing information gathering. Further investigation could analyse the effectiveness when introducing 2D convolution architectures into the model to capture differences on the raw histogram rather than on calculated statistical properties of it. However, this would first require an investigation of how to link irregularly taken snapshots of the histograms in form of successive readouts with battery ageing.

In addition, a comprehensive model architecture design and broad hyper-parameter optimization have not been taken into account in this thesis. The training duration and results are likely to improve if the model parameters are further optimised. Naturally, optimizing could be more detailed if it is limited to one specific model type. An extensive analysis of the best found model could be applied to optimize training and output accuracy.

Finally, it remains to be evaluated whether a comparison with other training approaches, such as unsupervised or hybrid learning, could produce more robust results for the case in question. Robust algorithms for prediction have to be found at least for a SOH prognosis application since the availability of clean validation data limits today's supervised methods.

Bibliography

[1] Bloomberg New Energy Finance. *Electric Vehicle Outlook 2020.* URL: `https://about.bnef.com/electric-vehicle-outlook/` (visited on 05/25/2020).

[2] Amsterdam Roundtable Foundation and McKinsey & Company. *Evolution. Electric vehicles in Europe: Gearing up for a new phase?* 2014. URL: `https://www.mckinsey.com/featured-insights/europe/electric-vehicles-in-europe-gearing-up-for-a-new-phase` (visited on 05/25/2020).

[3] Roland Irle. *Global BEV & PHEV Sales for 2019.* 2020. URL: `https://www.ev-volumes.com/` (visited on 05/25/2020).

[4] Anthony Barré, Benjamin Deguilhem, Sébastien Grolleau, Mathias Gérard, Frédéric Suard, and Delphine Riu. "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications". In: *Journal of Power Sources* 241 (2013), pp. 680–689. DOI: `10.1016/j.jpowsour.2013.05.040`.

[5] Christoph R. Birkl, Matthew R. Roberts, Euan McTurk, Peter G. Bruce, and David A. Howey. "Degradation diagnostics for lithium ion cells". In: *Journal of Power Sources* 341 (2017), pp. 373–386. DOI: `10.1016/j.jpowsour.2016.12.011`.

[6] Man-Fai Ng, Jin Zhao, Qingyu Yan, Gareth J. Conduit, and Zhi Wei Seh. "Predicting the state of charge and health of batteries using data-driven machine learning". In: *Nature Machine Intelligence* 114 (2020), p. 11414. DOI: `10.1038/s42256-020-0156-7`.

[7] Ali Jokar, Barzin Rajabloo, Martin Désilets, and Marcel Lacroix. "Review of simplified Pseudo-two-Dimensional models of lithium-ion batteries". In: *Journal of Power Sources* 327 (2016), pp. 44–55. DOI: `10.1016/j.jpowsour.2016.07.036`.

[8] Kristen A. Severson et al. "Data-driven prediction of battery cycle life before capacity degradation". In: *Nature Energy* 4.5 (2019), pp. 383–391. DOI: `10.1038/s41560-019-0356-8`.

[9] Sina Sharif Mansouri, Petros Karvelis, George Georgoulas, and George Nikolakopoulos. "Remaining Useful Battery Life Prediction for UAVs based on Machine Learning". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 4727–4732. DOI: `10.1016/j.ifacol.2017.08.863`.

[10] Adnan Nuhic, Tarik Terzimehic, Thomas Soczka-Guth, Michael Buchholz, and Klaus Dietmayer. "Health diagnosis and remaining useful life prognos-

tics of lithium-ion batteries using data-driven methods”. In: *Journal of Power Sources* 239 (2013), pp. 680–688. DOI: 10.1016/j.jpowsour.2012.11.146.

[11] Lei Ren, Li Zhao, Sheng Hong, Shiqiang Zhao, Hao Wang, and Lin Zhang. “Remaining Useful Life Prediction for Lithium-Ion Battery: A Deep Learning Approach”. In: *IEEE Access* 6 (2018), pp. 50587–50598. DOI: 10.1109/ACCESS.2018.2858856.

[12] Seyed Mohammad Rezvanizaniani, Zongchang Liu, Yan Chen, and Jay Lee. “Review and recent advances in battery health monitoring and prognostics technologies for electric vehicle (EV) safety and mobility”. In: *Journal of Power Sources* 256 (2014), pp. 110–124. DOI: 10.1016/j.jpowsour.2014.01.085.

[13] Arthur K. Barnes, Juan Carlos Balda, Scott O. Geurin, and Andres Escobar-Mejia. “Optimal battery chemistry, capacity selection, charge/discharge schedule, and lifetime of energy storage under time-of-use pricing”. In: *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*. 2011, pp. 1–7. DOI: 10.1109/ISGTEurope.2011.6162702.

[14] Yi Li, Kailong Liu, Aoife M. Foley, Alana Zülke, Maitane Berecibar, Elise Nanini-Maury, Joeri van Mierlo, and Harry E. Hoster. “Data-driven health estimation and lifetime prediction of lithium-ion batteries: A review”. In: *Renewable and Sustainable Energy Reviews* 113 (2019), p. 109254. DOI: 10.1016/j.rser.2019.109254.

[15] Vladimir Vapnik. “An overview of statistical learning theory”. In: *IEEE Transactions on Neural Networks* 10.5 (1999), pp. 988–999. DOI: 10.1109/72.788640.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. New York, NY: Springer New York, 2009. DOI: 10.1007/978-0-387-84858-7.

[17] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. “Introduction to Statistical Learning Theory”. In: *Advanced Lectures on Machine Learning*. Vol. 3176. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 169–207. DOI: 10.1007/978-3-540-28650-9-8.

[18] Pedro Domingos. “A few useful things to know about machine learning”. In: *Communications of the ACM* 55.10 (2012), pp. 78–87. DOI: 10.1145/2347736.2347755.

[19] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. Second edition. O’Reilly UK Ltd., 2019. ISBN: 9781492032649.

[20] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. URL: https://arxiv.org/abs/1609.04747.

[21] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. URL: https://arxiv.org/abs/1710.05941.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[23] Jürgen Schmidhuber. “Deep learning in neural networks: an overview”. In: *Neural networks : the official journal of the International Neural Network Society* 61 (2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

[24] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`.

[25] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. *WaveNet: A Generative Model for Raw Audio*. 2016. URL: `https://arxiv.org/abs/1609.03499`.

[26] Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. "A Convolutional Encoder Model for Neural Machine Translation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 123–135. DOI: `10.18653/v1/P17-1012`.

[27] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. "Convolutional Sequence to Sequence Learning". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. JMLR.org, 2017, pp. 1243–1252.

[28] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. URL: `https://arxiv.org/abs/1803.01271`.

[29] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016.

[30] Umair Shafique and Haseeb Qaiser. "A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA)". In: *International Journal of Innovation and Scientific Research* 12.1 (2014).

[31] Isabelle Guyon and André Elisseeff. ": An introduction to variable and feature selection". In: *Journal of machine learning research* 3 (2003), pp. 1157–1182.

[32] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[33] François Chollet et al. *Keras: The Python deep learning API*. 2015. URL: `https://keras.io`.

[34] Philippe Rémy. *Temporal Convolutional Network implementation for Keras*. 2020. URL: `https://github.com/philipperemy/keras-tcn`.

A

Error Distribution Properties

This appendix section presents all detailed statistical error distribution properties considered in the result section 5.1. They are listed in the order of their corresponding sub-section 5.1.1, 5.1.2 and 5.1.3, according to the different investigated data input output combinations pure static, pure sequential and windowed data respectively.

All statistical properties have been calculated on the set of prediction errors for all samples in the test dataset, following the 80/20 split rate introduced in section 4.2.4.

Table A.1: Statistical properties of the error distributions for the **static output format** SOH determination models. Calculated per model for the comparison in section 5.1.1

	ID	Mean	Standard Deviation	Variance	Minimum	Maximum	Skewness	Kurtosis
Linear Reg.	B1	0.046	2.047	4.191	-8.504	10.326	0.102	2.840
Elastic Net	B2	0.037	2.110	4.453	-8.291	10.647	0.333	2.899
Rand. Forest	B3	-0.034	1.649	2.718	-8.008	4.547	-0.880	2.458
ANN	B4	-0.011	1.151	1.325	-7.004	4.008	-1.249	4.889
LSTM	R1	-0.173	1.188	1.411	-9.157	4.374	-1.543	7.312
LSTM+ANN	R3	-0.155	0.990	0.981	-6.396	3.701	-0.799	4.163
TCN	C1	0.123	1.997	3.986	-10.579	4.571	-1.370	3.658
TCN+ANN	C3	-0.670	0.717	0.515	-4.381	2.904	-0.569	5.475

Table A.2: Statistical properties of the error distributions for the **sequential output format** SOH determination models. Calculated per model for the comparison in section 5.1.2

	ID	Mean	Standard Deviation	Variance	Minimum	Maximum	Skewness	Kurtosis
LSTM	R2	-0.053	1.215	1.477	-8.969	5.856	-1.053	10.092
LSTM+ANN	R4	0.235	0.968	0.938	-7.748	5.201	-0.703	10.110
TCN	C2	-0.083	1.368	1.872	-11.666	4.514	-1.863	7.735
TCN+ANN	C4	0.135	1.499	2.247	-7.468	5.512	-0.414	1.457

Table A.3: Statistical properties of the error distribution for the **step** SOH determination model. Calculated per step and per sequence for the comparison in section 5.1.3

	ID	Mean	Standard Deviation	Variance	Minimum	Maximum	Skewness	Kurtosis
Step model (per step)	S1	-0.062	0.222	1.193	-2.095	1.193	-1.595	9.525
Step model (per sequence)	S1	-4.258	6.060	36.729	-39.693	2.562	-3.310	12.712

B

Feature Correlation Matrices

This appendix section presents all correlation matrices considered in the pre-selection analysis of features per feature group in section 5.2.2. They are listed in the following order, where each list item contains the features of the corresponding group:

- **B.1**: Temperature
- **B.2**: State of Charge
- **B.3**: Over charge and discharge
- **B.4**: Age
- **B.5**: Current
- **B.6**: Energy Throughput
- **B.7**: Depth of discharge

For each feature group the cross-correlation matrix between all features of the group and a correlation matrix with respect to the target are combined in one figure as combined correlation plot. The Spearman correlation coefficient was used to calculate the correlation coefficients.

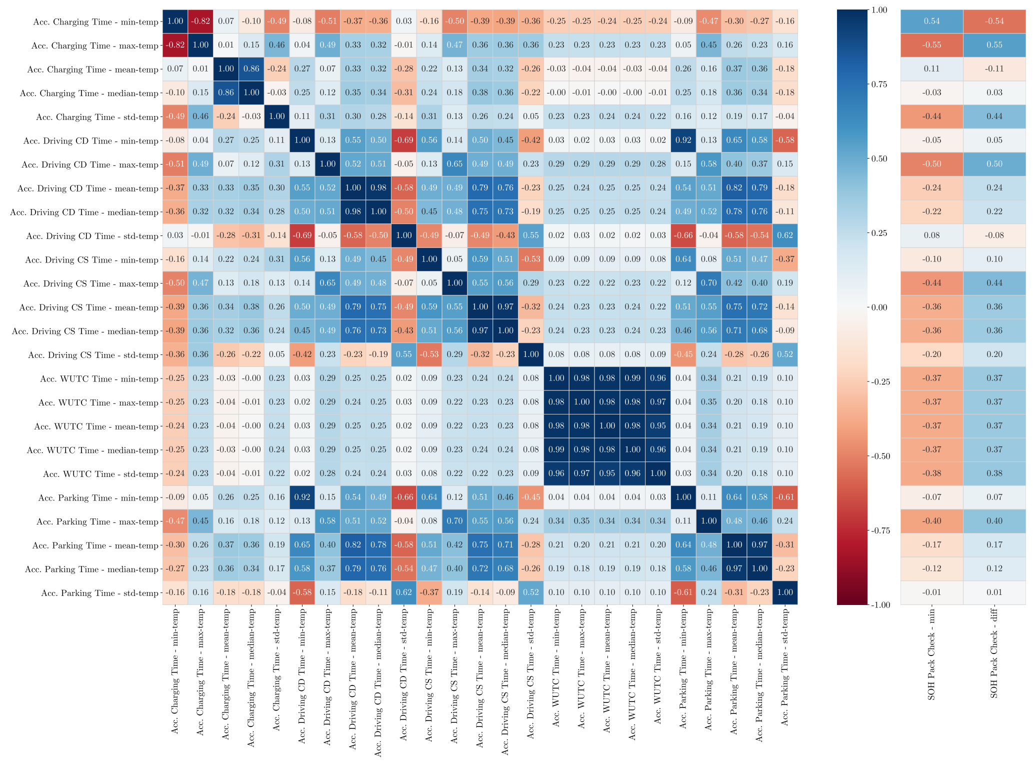


Figure B.1: Spearman correlation matrices for the **temperature** feature group.

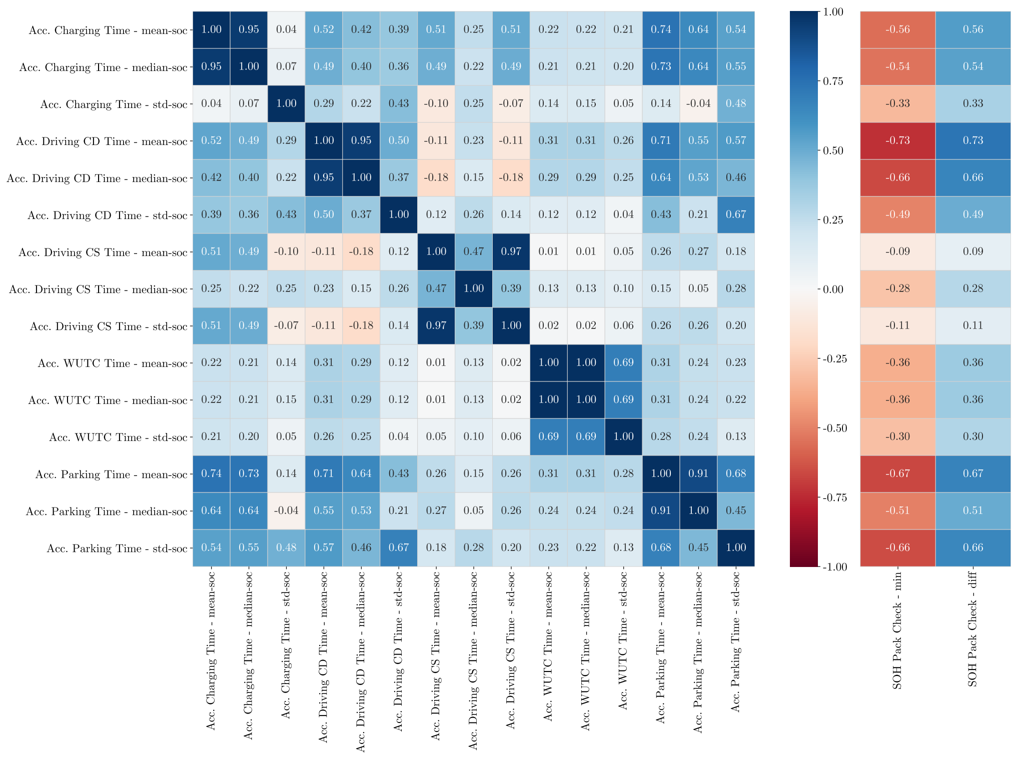


Figure B.2: Spearman correlation matrices for the **SOC** feature group.

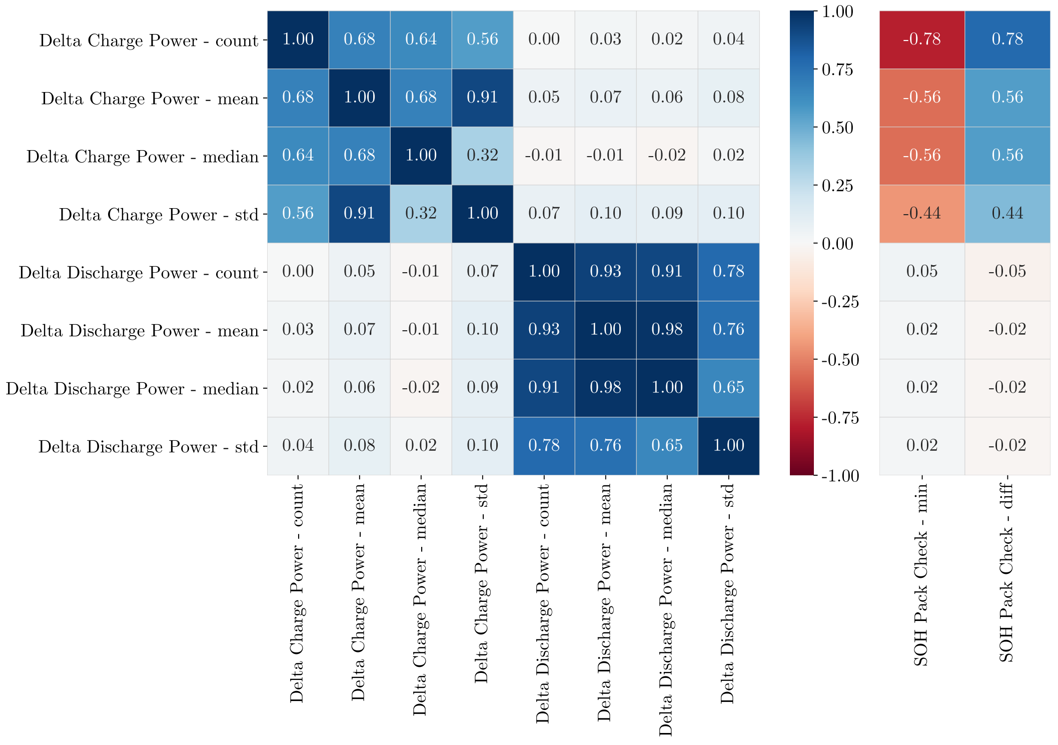


Figure B.3: Spearman correlation matrices for the **over charge/discharge** feature group.

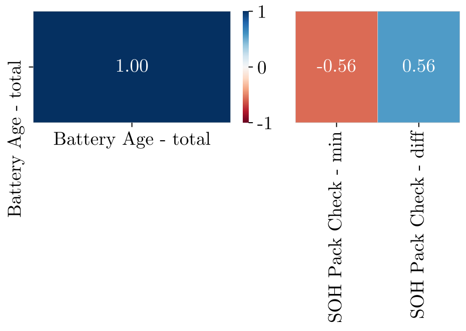
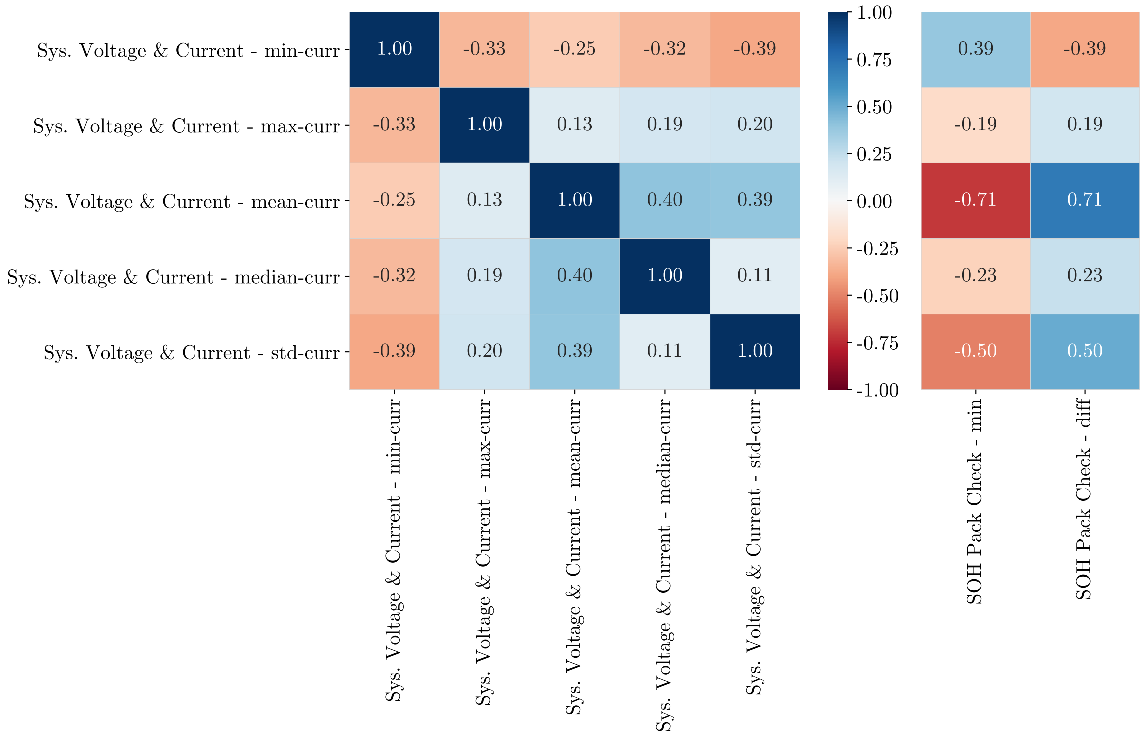
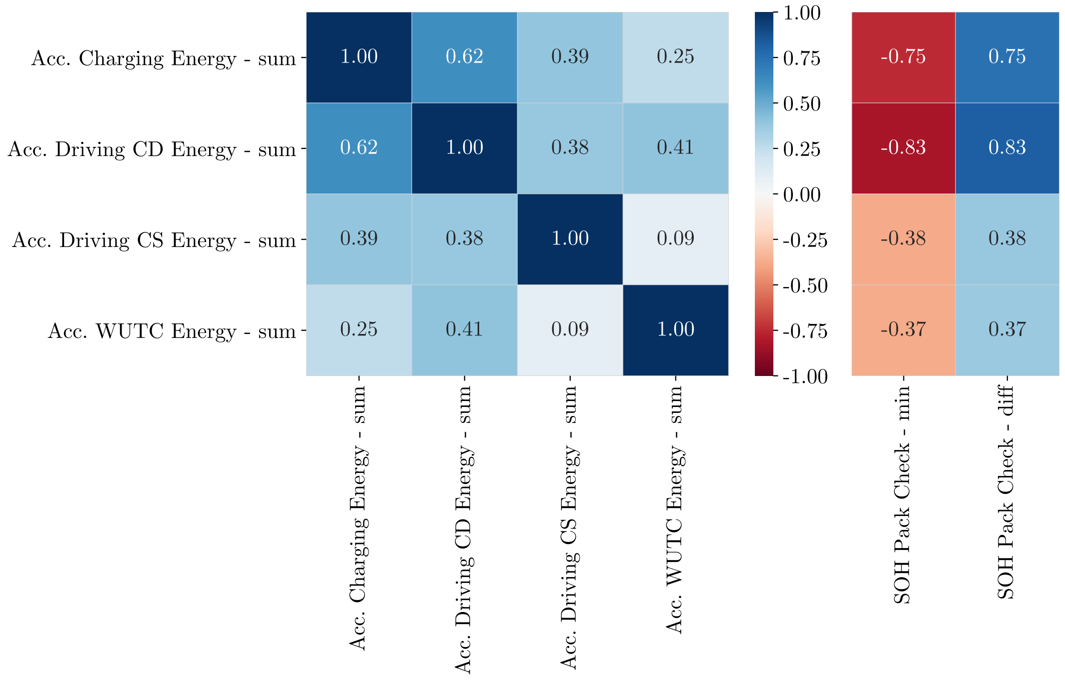


Figure B.4: Spearman correlation matrices for the **age** feature group.



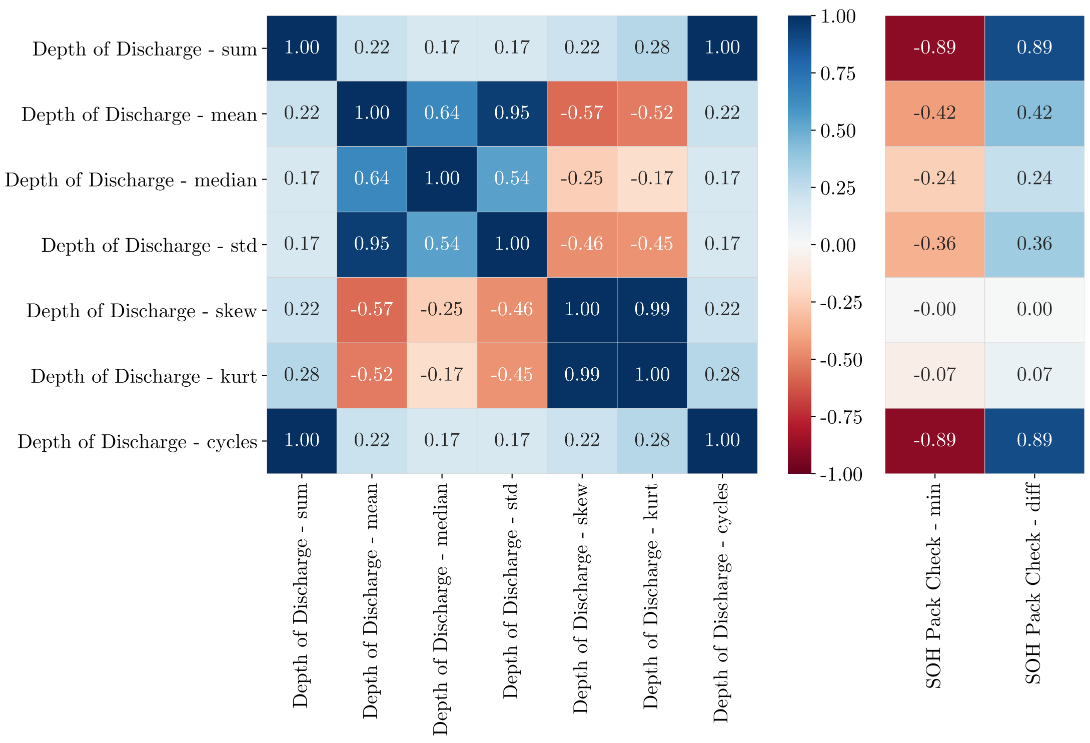
	Sys. Voltage & Current - min-curr	Sys. Voltage & Current - max-curr	Sys. Voltage & Current - mean-curr	Sys. Voltage & Current - median-curr	Sys. Voltage & Current - std-curr	SOH Pack Check - min	SOH Pack Check - diff
Sys. Voltage & Current - min-curr	1.00	-0.33	-0.25	-0.32	-0.39	0.39	-0.39
Sys. Voltage & Current - max-curr	-0.33	1.00	0.13	0.19	0.20	-0.19	0.19
Sys. Voltage & Current - mean-curr	-0.25	0.13	1.00	0.40	0.39	-0.71	0.71
Sys. Voltage & Current - median-curr	-0.32	0.19	0.40	1.00	0.11	-0.23	0.23
Sys. Voltage & Current - std-curr	-0.39	0.20	0.39	0.11	1.00	-0.50	0.50

Figure B.5: Spearman correlation matrices for the **current** feature group.



	Acc. Charging Energy - sum	Acc. Driving CD Energy - sum	Acc. Driving CS Energy - sum	Acc. WUTC Energy - sum	SOH Pack Check - min	SOH Pack Check - diff
Acc. Charging Energy - sum	1.00	0.62	0.39	0.25	-0.75	0.75
Acc. Driving CD Energy - sum	0.62	1.00	0.38	0.41	-0.83	0.83
Acc. Driving CS Energy - sum	0.39	0.38	1.00	0.09	-0.38	0.38
Acc. WUTC Energy - sum	0.25	0.41	0.09	1.00	-0.37	0.37

Figure B.6: Spearman correlation matrices for the **energy throughput** feature group.



	Depth of Discharge - sum	Depth of Discharge - mean	Depth of Discharge - median	Depth of Discharge - std	Depth of Discharge - skew	Depth of Discharge - kurt	Depth of Discharge - cycles
Depth of Discharge - sum	1.00	0.22	0.17	0.17	0.22	0.28	1.00
Depth of Discharge - mean	0.22	1.00	0.64	0.95	-0.57	-0.52	0.22
Depth of Discharge - median	0.17	0.64	1.00	0.54	-0.25	-0.17	0.17
Depth of Discharge - std	0.17	0.95	0.54	1.00	-0.46	-0.45	0.17
Depth of Discharge - skew	0.22	-0.57	-0.25	-0.46	1.00	0.99	0.22
Depth of Discharge - kurt	0.28	-0.52	-0.17	-0.45	0.99	1.00	0.28
Depth of Discharge - cycles	1.00	0.22	0.17	0.17	0.22	0.28	1.00

	SOH Pack Check - min	SOH Pack Check - diff
Depth of Discharge - sum	-0.89	0.89
Depth of Discharge - mean	-0.42	0.42
Depth of Discharge - median	-0.24	0.24
Depth of Discharge - std	-0.36	0.36
Depth of Discharge - skew	-0.00	0.00
Depth of Discharge - kurt	-0.07	0.07
Depth of Discharge - cycles	-0.89	0.89

Figure B.7: Spearman correlation matrices for the **depth of discharge** feature group.