129

Karlsruher Schriftenreihe Fahrzeugsystemtechnik

Andreas M. Billert

# Predictive Battery Thermal Management of Electric Vehicles using Deep Learning

KIT Scientific Publishing

Andreas M. Billert

# Predictive Battery Thermal Management of Electric Vehicles using Deep Learning

**Karlsruher Schriftenreihe Fahrzeugsystemtechnik**
**Band 129**

Herausgeber
**FAST Institut für Fahrzeugsystemtechnik**
Prof. Dr.-Ing. Martin Cichon
Prof. Dr. rer. nat. Frank Gauterin
Prof. Dr.-Ing. Marcus Geimer
Prof. Dr.-Ing. Frank Henning
Prof. Dr.-Ing. Luise Kärger

Das Institut für Fahrzeugsystemtechnik besteht aus den Institutsteilen Bahnsystemtechnik, Fahrzeugtechnik, Leichtbau und Mobile Arbeitsmaschinen.

Eine Übersicht aller bisher in dieser Schriftenreihe erschienenen Bände finden Sie am Ende des Buchs.

# Predictive Battery Thermal Management of Electric Vehicles using Deep Learning

by
Andreas M. Billert

KIT Scientific Publishing

Karlsruher Institut für Technologie
Institut für Fahrzeugsystemtechnik

Predictive Battery Thermal Management of
Electric Vehicles using Deep Learning

Zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) von der KIT-Fakultät für Maschinenbau des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Andreas Martin Billert, M.Sc.

Tag der mündlichen Prüfung: 20. Dezember 2024
Hauptreferent: Prof. Dr. rer. nat. Frank Gauterin
Korreferent: Prof. Dr.-Ing. Andreas Jossen

**Impressum**

Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe



www.bibliothek.kit.edu/ksp.php | E-Mail: info@ksp.kit.edu | Shop: www.ksp.kit.edu



Print on Demand 2025 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1869-6058
ISBN 978-3-7315-1429-9
DOI 10.5445/KSP/1000180497

# Vorwort des Herausgebers

Die Fahrzeugtechnik ist kontinuierlich Veränderungen unterworfen. Klimawandel, die Verknappung einiger für Fahrzeugbau und -betrieb benötigter Rohstoffe, globaler Wettbewerb, gesellschaftlicher Wandel und das rapide Wachstum großer Städte erfordern neue Mobilitätslösungen, die vielfach eine Neudefinition des Fahrzeugs erforderlich machen. Die Forderungen nach Steigerung der Energieeffizienz, Emissionsreduktion, erhöhter Fahr- und Arbeitssicherheit, Benutzerfreundlichkeit und angemessenen Kosten sowie die Möglichkeiten der Digitalisierung und Vernetzung finden ihre Antworten nicht aus der singulären Verbesserung einzelner technischer Elemente, sondern benötigen Systemverständnis und eine domänenübergreifende Optimierung der Lösungen.

Hierzu will die Karlsruher Schriftenreihe für Fahrzeugsystemtechnik einen Beitrag leisten. Für die Fahrzeuggattungen Pkw, Nfz, Mobile Arbeitsmaschinen und Bahnfahrzeuge werden Forschungsarbeiten vorgestellt, die Fahrzeugsystemtechnik auf vier Ebenen beleuchten: das Fahrzeug als komplexes, digitalisiertes, mechatronisches System, die Mensch-Fahrzeug-Interaktion, das Fahrzeug in Verkehr und Infrastruktur sowie das Fahrzeug in Gesellschaft und Umwelt.

Eine hohe Lebensdauer der Traktionsbatterie von Elektrofahrzeugen, ihr energieeffizienter Betrieb und eine stets hohe verfügbare Batterieleistung steigern die Attraktivität von Elektrofahrzeugen und unterstützen damit eine stärkere Verbreitung im Markt. Da die Eigenschaften der Batterie und ihr Alterungsverhalten stark von ihrer Temperatur beeinflusst werden, kommt dem Thermomanagement hierbei eine besondere Bedeutung zu.

Die vorliegende Arbeit beschreibt eine Vorgehensweise zur Entwicklung quantiler modellprädiktiver Steuerungen mittels verschiedener neuronaler Netze auf

der Basis systematisch aufbereiteter Flotten- und Simulationsdaten. Sie zeigt weiterhin Wege auf, die Batterietemperatur auch dann über einen weiten Prädiktionshorizont vorhersagen zu können, wenn nur wenige Daten vorliegen oder die Rechenleistung im Fahrzeug begrenzt ist. Die vorgeschlagenen Methoden werden zur Optimierung der Kühlschwelle des Batterie-Thermomanagements angewandt und erreichen deutliche Verbesserungen gegenüber den bislang üblichen festen Schwellen.

*Frank Gauterin*

im Dezember 2024

# Kurzfassung

**Prädiktives Batteriethermomanagement von Elektrofahrzeugen mittels Deep Learning.**

Eine Verbesserung der Energieeffizienz von Batterieelektrofahrzeugen vergrößert ihre Reichweite und reduziert die (vom Strommix abhängigen) Emissionen. Ein effizientes Batteriethermomanagement reduziert den Energieverbrauch bei gleichzeitiger Berücksichtigung der temperaturabhängigen Batteriealterung und Leistungsverfügbarkeit. Zum Beispiel kann eine prädiktive Kühlstrategie den Energieverbrauch im Vergleich zu einer Kühlstrategie mit fixen Regelschwellen reduzieren, indem Informationen über die vorausliegende Strecke genutzt werden. Eine entsprechende prädiktive Regelung erfordert ein präzises Prädiktionsmodell.

Diese Arbeit präsentiert eine Methode zur Entwicklung von Prädiktionsmodellen für eine prädiktive Regelung, angewendet auf das Batteriethermomanagement. Deep Learning basierte Quantil Neuronale Netze (Q*NN) prädizieren Quantilssequenzen der Änderung der Batterietemperatur, unter Nutzung von Historien- und Vorausschaueingangsdaten. Im Rahmen der Datenverarbeitung werden Simulationsdaten und Flottendaten erhoben und mithilfe von Data Augmentation und Clustering vorverarbeitet. Die Q*NN liefern genauere Prädiktionen im Vergleich zu Referenzmodellen (u.a. Entscheidungsbäume). Bei begrenzter Datenverfügbarkeit können Prädiktionsmodelle mithilfe von Transfer Learning entwickelt werden.

Die Q*NN Prädiktionen werden in einer prädiktiven Regelung zur Anpassung der Batteriekühlschwellen verwendet. Die prädiktive Regelung minimiert eine Kostenfunktion aus Kühlenergieverbrauch, Alterung und Leistungsverfügbarkeit der Batterie. Im Vergleich zu einer Kühlung mit fixen Regelschwellen reduziert

die prädiktive Regelung mit Q*NN die Kühlkosten um 16 % bei vergleichbaren Alterungskosten und Leistungsverfügbarkeit. Die Ableitung von weniger rechenintensiven regelbasierten Modellen bietet einen Ausblick für reale Anwendungen in Fahrzeugen.

# Abstract

Improving the energy efficiency of battery electric vehicles increases their range and reduces well-to-wheel emissions. An efficient battery thermal management reduces the energy consumption while taking temperature-dependent battery ageing and power availability into account. For example, a predictive cooling strategy can reduce the energy consumption compared to a cooling strategy with fixed control thresholds, using information about the route ahead. Such a predictive control requires a precise prediction model.

This work presents a method for the development of prediction models for a predictive control, applied to the battery thermal management. Deep Learning based Quantile Neural Networks (Q*NN) predict quantile sequences of the change in battery temperature, using history input and foresight input data. Simulation data and fleet data are collected and further processed using data augmentation and clustering. The Q*NN provide more accurate predictions compared to reference models (i.a. decision trees). Transfer Learning enables the development of prediction models when only few data are available.

The Q*NN predictions are used in a predictive control to adapt the battery cooling thresholds. The predictive control minimizes a cost function of cooling energy consumption, ageing, and power availability of the battery. Compared to a cooling strategy with fixed control thresholds, the Q*NN predictive control reduces cooling costs by 16 % with comparable ageing costs and power availability. The derivation of computationally less expensive rule-based models provides an outlook for real-world application in vehicles.

# Preface

千里之行，始于足下 。– 老子 (公元前四零零年)
A journey of a thousand miles begins with a single step. – Laozi (400 BC)

The automotive industry currently experiences great disruption in electric mobility, autonomous driving, and digitalization. I am very grateful that I have been given the chance to take part and contribute to the development with this dissertation at Bayerische Motoren Werke (BMW) AG, in cooperation with the Institute of Vehicle System Technology at Karlsruhe Institute of Technology (KIT). I truly appreciate that I was given the freedom and support by both BMW AG and KIT to develop my ideas and to grow my skills and knowledge within the dissertation project.

Therefore, I would like to thank Prof. Frank Gauterin for his supervision and his trust in my ideas, as well as continuous feedback and support on the dissertation. I would further like to thank Prof. Andreas Jossen to take the role as second examiner, including his feedback on the dissertation. Many thanks go to Dr. Michael Frey for the technical discussions about the dissertation, as well as his valuable feedback, guidance, and continuous support.

Furthermore, I would like to thank Dr. Simone Fuchs and Dr. Stefan Erschen for the technical discussions, feedback, and guidance at BMW AG. They have been providing me great support during my three years of the dissertation project. I would like to additionally thank my direct managers (line, unit, and department) at BMW AG for their feedback, support, and trust in my work. Further thanks go to Marius Schmid, Runyao Yu, and Xin Huang for writing their Master's thesis at BMW AG. It was a pleasure to give them guidance and feedback, as well as discuss their methods and results with them.

The exchange with other doctoral students at BMW AG and KIT has been another valuable part of working on this dissertation. While I cannot list all names, I would like to thank Tobias Straub for his support especially in the beginning, since he has been a doctoral student in cooperation with both BMW AG and KIT as well. Additional thanks go to Simon Schramm with whom I had the pleasure to organize a doctoral research group about Machine Learning and Big Data at BMW AG. Moreover, I would like to thank Achim Winandi from KIT and all other participants of the coffee lectures he organized for their introductions to useful tools and best practices.

Additionally, I would like to express my deepest gratitude to my family who has been there for me in all good times and challenging times. This dissertation would not have been possible without my parents Hans-Joachim and Marita, who have given me all thinkable support, guidance, and encouragement during my life. They have achieved to let me personally grow and broaden my horizon without losing my roots. Many thanks go to my brother, Dr. Matthias Simon Billert, who has always been a great guide for me to go the next steps in life. Finally, I am very grateful to thank my girlfriend Ruiling Ding for her support and encouragement. She is a great source of inspiration for both my personal and professional life, and I am looking forward to our future.

Munich, 28th of August 2024 *Andreas M. Billert*

# Contents

# Acronyms and symbols

## Acronyms

| | |
|---|---|
| **AC** | Air-Conditioning |
| **ADAM** | Adaptive Moment Estimation |
| **ARIMA** | Auto-Regressive Integrated Moving Average |
| **BEV** | Battery Electric Vehicle |
| **BTMS** | Battery Thermal Management System |
| **CNN** | Convolutional Neural Network |
| **CORS** | Crossover Rate Score |
| **CRISP – DM** | Cross Industry Standard Process for Data Mining |
| **DBA** | Dynamic Time Warping based Barycentric Averaging |
| **DL** | Deep Learning |
| **DOE** | Design of Experiments |
| **DOD** | Depth of Discharge |
| **DP** | Dynamic Programming |
| **DWD** | German Weather Service |
| **ECU** | Electronic Control Unit |

| | |
|---|---|
| **ED** | Electrolyte Decomposition |
| **EDA** | Exploratory Data Analysis |
| **E/E** | Electrics and Electronics |
| **ETR** | Extra Trees Regressor |
| **FFT** | Fast Fourier Transformation |
| **GAN** | Generative Adversarial Network |
| **GPS** | Global Positioning System |
| **GRU** | Gated Recurrent Unit |
| **HEV** | Hybrid Electric Vehicle |
| **HPO** | Hyperparameter Optimization |
| **IBA** | Intersection-Based Assembly |
| **LD** | Less Dynamic |
| **LGBM** | Light Gradient Boosting Machine |
| **LIP** | Lithium-Plating |
| **LSTM** | Long Short-Term Memory |
| **MAE** | Mean Absolute Error |
| **MD** | More Dynamic |
| **MeD** | Mechanical Damage |
| **ML** | Machine Learning |
| **MPC** | Model Predictive Control |
| **MSE** | Mean Squared Error |
| **NARX** | Nonlinear Auto-Regressive with eXogeneous input |

| | |
|---|---|
| **NMPC** | Nonlinear Model Predictive Control |
| **NN** | Neural Network |
| **PCA** | Principle Component Analysis |
| **PHEV** | Plug-In Hybrid Electric Vehicle |
| **PI** | Proportional-integral |
| **PID** | Proportional-integral-derivative |
| **PINN** | Physics-informed Neural Networks |
| **Q*NN** | Quantile Neural Network based on a deep learning architecture (e.g. with convolutional or recurrent layers) |
| **QCNN** | Quantile Convolutional Neural Network |
| **QCNN22** | Quantile Convolutional Neural Network from [1] |
| **QETR** | Quantile Extra Trees Regressor |
| **QGRU** | Quantile Gated Recurrent Unit |
| **QL** | Qualifier |
| **QLSTM** | Quantile Long Short-Term Memory |
| **QRF** | Quantile Random Forest |
| $\mathbf{R^2}$ | Coefficient of Determination |
| **RAM** | Random Access Memory |
| **RF** | Random Forest |
| **RHC** | Receding Horizon Control |
| **RMSE** | Root Mean Squared Error |
| **RNN** | Recurrent Neural Network |

| | |
|---|---|
| **SDT** | Shallow Decision Tree |
| **SEI** | Solid Electrolyte Interface |
| **SMOTE** | Synthetic Minority Oversampling Technique |
| **SMPC** | Stochastic Model Predictive Control |
| **SOC** | State of Charge |
| **SOH** | State of Health |
| **SSD** | Sum of Squared Distance |
| **TL** | Transfer Learning |
| **t − SNE** | t-distributed Stochastic Neighbour Embedding |
| **UMAP** | Uniform Manifold Approximation and Projection |
| **VT** | Voting Threshold |
| **WLTC** | Worldwide harmonized Light vehicles Test Cycle |
| **WS** | Winkler Score |

## Constants

| | |
|---|---|
| $\varepsilon$ | Constant factor (to avoid division by zero): $10^{-7}$ |

## Latin symbols and variables

| | |
|---|---|
| $A$ | surface; sequence in Intersection-Based Assembly |
| $B$ | sequence in Intersection-Based Assembly |
| $c$ | cost function |
| $c_p$ | specific heat capacity at constant pressure |

| | |
|---|---|
| $D$ | disturbance |
| $d$ | distance |
| $dlat$ | corrected latitude |
| $dlong$ | corrected longitude |
| $F$ | force |
| $f$ | function |
| $g$ | (estimation) function |
| $H$ | heat transfer coefficient |
| $h$ | horizon |
| $I$ | current |
| $j$ | constraints |
| $L$ | loss function |
| $lat$ | latitude |
| $long$ | longitude |
| $m$ | mass |
| $N$ | number of layers as hyperparameter |
| $n$ | number (quantity) |
| $P$ | power |
| $p$ | occurrence |
| $Q$ | heat |
| $q$ | quantile |
| $R$ | inner resistance |

| | |
|---|---|
| $r$ | road height |
| $S$ | step |
| $s$ | segment |
| $T$ | temperature |
| $t$ | time |
| $U$ | voltage |
| $u$ | control value |
| $v$ | speed |
| $w$ | weighting factor |
| $X$ | inputs |
| $x$ | (placeholder) variable |
| $y$ | system output |

## Greek symbols and variables

| | |
|---|---|
| $\eta$ | efficiency |
| $\mu$ | mean value |
| $\mu_j$ | weight of quantile j |
| $\Omega$ | Ohmic resistance |
| $\psi$ | sharpness |
| $\rho$ | share |
| $\sigma_{q_j}$ | smoothness of quantile j |
| $\tau$ | trial |

| | |
|---|---|
| $\theta$ | slope |

## Operators and math symbols

| | |
|---|---|
| $\forall$ | for all |
| $\lvert x \rvert$ | absolute value of $x$ |
| $\bar{x}$, $\text{avg}(x)$, $\text{mean}(x)$ | average of $x$ |
| $\dot{x}$ | rate of $x$ (e.g. the first derivative over time) |
| $\cos(x)$ | cosinus of $x$ |
| $\Delta(x)$ | difference of consecutive values of $x$ |
| $\frac{\partial x}{\partial y}$ | partial derivative of x by y |
| $\exp(x)$ | exponential function of $x$ |
| $\max(x)$ | maximum of $x$ |
| $\min(x)$ | minimum of $x$ |
| $\sigma(x)$ | standard deviation of $x$ |
| $\text{var}(x)$ | variance of $x$ |

## General deep indexes

| | |
|---|---|
| $+$ | positive |
| $add$ | additional |
| $a$ | ageing |
| $amb$ | ambient |
| $avg$ | average |

| | |
|---|---|
| $b$ | battery |
| $b, cool, lower$ | lower battery cooling threshold |
| $b, cool, upper$ | upper battery cooling threshold |
| $c$ | cooling |
| $coolant$ | coolant |
| $cons$ | consumer |
| $ctrl$ | control |
| $cv$ | control values |
| $cwq$ | constrained weighted quantiles |
| $d$ | derating |
| $dr$ | driving |
| $d, abs$ | absolute difference |
| $em$ | electric machine |
| $end$ | end |
| $env$ | environment |
| $f$ | foresight |
| $gen$ | generated |
| $GPS$ | Global Positioning System |
| $h$ | history |
| $i$ | index i, related to horizon step |
| $init$ | initial |
| $inv$ | inverter |

| | |
|---|---|
| $j$ | index j, related to quantile |
| $max$ | maximum |
| $mse$ | mean squared error |
| $o$ | index o, related to observations |
| $\Omega$ | Ohmic resistance |
| $ocv$ | open circuit voltage |
| $p$ | prediction |
| $pa$ | parking |
| $pb$ | pinball |
| $q$ | quantile |
| $q_{0.5}$ | 0.5 quantile (median) |
| $rct$ | reaction |
| $res$ | resistance |
| $rev$ | reversible |
| $sc$ | scenario |
| $sensor$ | sensor |
| $shift$ | shift |
| $sig$ | signal |
| $t$ | threshold |
| $\tau$ | trial |
| $tc$ | temperature change |
| $tm$ | thermal management |

| | |
|---|---|
| $tr$ | transmission |
| $total$ | total |

# 1 Introduction

This chapter emphasizes the importance of an efficient battery thermal management for electric vehicles and the application of Machine Learning for an improved control in 1.1. The objective of this dissertation is further explained in 1.2, followed by an outline of its structure in 1.3. The dissertation extends on the prepublished articles [1, 2, 3], as explained in 1.4. Remarks on notation in this dissertation are given in 1.5.

## 1.1 Motivation

Climate change related regulations for greenhouse gas emissions push the growing sales of Battery Electric Vehicles (BEV) [8, 9, 10], [11, p. 66,72,78–79]. For example, more than 10 Million hybrid and battery electric vehicles have been sold in 2022, with a BEV share of 70 % [11, p. 8,14]. Strict vehicle emission policies support climate change mitigation as well as reduction in local air pollution and dependency on energy imports [8], [11, p. 76]. An energy consumption reduction of BEV further reduces well-to-wheel emissions (from electricity production) [9]. However, the thermal management of cabin and battery can increase the energy consumption of BEV (with consequently decreased range), dependent on the environmental conditions [12, 13, 14]. Thus, an efficient battery thermal management is necessary considering its impact on battery ageing, power derating, and energy consumption [13].

The energy consumption of a battery thermal management can be reduced by intelligent (predictive) control methods [15], [16, p. 123]. However, predictive

control requires system models which are both accurate and computationally efficient [17, 18]. Machine Learning (ML) methods can be used for a data-driven development of an adaptive control of the battery thermal management [15], [19, p. 298]. Knowledge can be extracted by combination of complementary data from different domains [20]. For instance, map-related energy consumption of BEV can be learned using vehicle fleet data [21]. A battery thermal management control can be improved with navigation information about the route ahead (during active guidance or by trip estimation), such as weather forecast [22] and speed prediction [23]. Deep Learning (DL), a subcategory of ML [24, p. 7], shows high performance in applications with large data sets [25, 26]. ML (and DL) can be used for forecasting of sequences (e.g. time-series) [24, p. 203ff], which can be extended by the prediction uncertainty for an improved practical value [27, 28]. Challenges of ML application include data collection, data processing, model evaluation and deployment [29, 30]. Furthermore, data availability of rare cases [31] and their evaluation can be problematic [32], as well as possible learning of unintended relations which lead to bad robustness and generalization of the models [33].

## 1.2 Objective

Accurate predictive control of the battery thermal management shows the potential to improve the energy consumption while additionally considering battery temperature requirements induced by battery ageing and power derating. In this dissertation, a method of Machine Learning based predictive control is developed and applied to the battery thermal management. The goal is to develop a Deep Learning prediction model, which uses fleet data and foresight input data (obtained from the route ahead) to accurately predict sequences with consideration of the prediction uncertainty. Such a prediction model is intended to be integrated in a predictive control scheme which minimizes application-dependent cost functions. The prediction model is applied to battery temperature prediction and the predictive control addresses the improvement of a battery thermal management

considering cooling energy consumption, battery ageing, and power derating. As a result, the predictive battery thermal management is intended to adapt the cooling strategy dependent on the scenario and the route ahead. Evaluation of the prediction model and predictive control includes a comparison with baseline models. While this work focuses on method development and testing, further requirements for real-world application are considered. For example, concepts are developed to target the issues of required computational resources for prediction model and predictive control, as well as limited data availability for new vehicle models.

The scope of this dissertation includes an improvement of battery thermal management for Battery Electric Vehicles (BEV), specifically the adaption of cooling strategies to routes ahead with accordingly large, distance-based horizons. Battery heating and the thermal management of the electric machine, power electronics, or cabin are not investigated. However, the method of ML based predictive control can be transferred to other applications. The prediction of routes [34], corresponding speed profiles [35], and weather [36] are not covered in this work, but assumed to be given. Moreover, performance evaluation of prediction model and predictive control is considered as an indicator, rather than a universal assessment and validation. Instead, performance needs to be individually evaluated for the respective application. Thus, this dissertation aims to provide a guide for suitable methods of development and evaluation, including an exemplary interpretation of results for an application to battery thermal management.

## 1.3 Structure

The structure of this dissertation is comparable to the V-model from systems engineering. The V-model is a general framework which specifies systems development including requirements, analysis and decomposition, design, implementation, integration, and validation [37, 38]. Translated into the dissertation structure, these elements correspond to the introduction, state of the art, method, application, discussion, and conclusion respectively, as shown in Fig. 1.1.

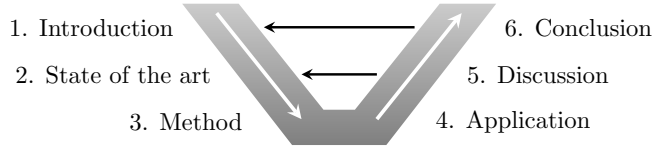


**Figure 1.1:** Structure of the dissertation as V-model, derived from [37, 38].

After this introduction, the state of the art is described in chapter 2, focusing on batteries for electric vehicles, predictive control, and Machine Learning for sequential data. The state of the art includes a research conclusion, pointing out identified gaps and resulting research questions. The method is described in chapter 3, with a pipeline from problem statement, prediction model, predictive control, and assessment of applicability. It is applied to the battery thermal management in chapter 4. The results are discussed according to the research questions and compared with the findings from the state of the art in chapter 5. Furthermore, the discussion includes advantages and disadvantages as well as potentials and limitations of this dissertation. A conclusion follows in chapter 6.

## 1.4 Citation of prepublications

The articles [1, 2, 3] have been prepublished as part of the dissertation project to receive early feedback from the scientific community. In this dissertation, these articles are put into a coherent context and extended with new results. Accordingly, the content of the articles is referenced and adapted. Namely, the dissertation combines the contents of data collection, data preprocessing, Machine Learning model building and training as well as evaluation from [1] and [3], and novel Transfer Learning with an additional data set. The application of Machine Learning models for a predictive control from [2] is extended by models from [3] as well as novel rule-based strategies. In very few cases, direct citations are used (e.g. for lists of parameter settings). These cases are indicated by the word “cited from” followed by the corresponding reference and (except for figures and tables) the usage of quotation marks. Details of the articles which are not relevant for the understanding of this dissertation are not included.

## 1.5 Notation

This section provides information about notation and conventions used in this dissertation in order to clarify their meaning. Further notation follows general conventions. The following expressions are used in this dissertation:

- Prediction and forecast: The two words are only distinguished in the description of model inputs and outputs. In this context, forecast specifically relates to input data that is assumed to be given, while prediction refers to the output from the developed prediction method.
- Sequence and time-series: Time-series is considered as a subgroup of sequences. Sequence is used to emphasize that data are not limited to be time-based but can be distance-based, which is the case in chapter 4.
- Horizon: Horizon relates to the length or end of a sequence.
- Segment and step: Segment describes an interval (of time or distance) with fixed length, for which quantities are sampled to a single value. Consecutive segments form a sequence. Step is used as a more general term when dealing with consecutive elements or actions.
- Quantile and percentile: A quantile describes the share of data below a value and a percentile is the corresponding percentage [39]. For example: 25 % of the values are below the 0.25 quantile or 25th percentile.
- Cost: In the context of predictive control, costs refer to optimization costs, rather than monetary costs.
- Speed and velocity: Speed can be considered a scalar value and velocity as a vector [40]. Considering the focus of this work, the term speed is used.
- Ambient and environment: Both words are used interchangeably for the surroundings of the vehicle.
- Mean and average: Both are used interchangeably.

- Quantity and feature: Quantity refers to a physical quantity or signal, while feature is used as another word for an input of a Machine Learning model.
- Prediction model and vehicle model: The term model is used to describe Machine Learning prediction models and vehicle models, which differ for example in generation, sub-components, or body (size, design).
- Parameter and hyperparameter: Parameters are trainable elements (weights) of a (Machine Learning) model, while hyperparameters are parameters on a higher level, defining the architecture and training process of the Machine Learning model.

In this dissertation, mathematical notation includes:

- $\hat{x}$ indicates that $x$ is a prediction variable.
- $\tilde{x}$ is used for a variable $x$ evaluated within a control loop.

# 2 State of the art

This chapter provides an overview of the state of the art about high voltage batteries for electric vehicles (2.1), predictive control (2.2) and Machine Learning methods for sequential data (2.3). Research gaps are pointed out in (2.4).

## 2.1 Batteries for electric vehicles

The high voltage battery is one of the most important components in battery electric vehicles, given its task to store and provide energy for the electric drivetrain and auxiliary consumers. Lithium-ion battery cells are the most popular due to their high energy density and cyclic lifetime compared to other currently available cells [41, p. 146], [19, p. 20]. Fundamentals of lithium-ion batteries (2.1.1) and the resulting requirements for a battery thermal management system (2.1.2) are further explained in the following, with focus on application in electric vehicles.

### 2.1.1 Fundamentals

The energy that can be stored in and provided by a battery is limited by the battery's capacity and voltage [41, p. 131–132]. The stored energy with respect to the maximum storable energy can be described as State of Charge (SOC, in %), and the amount of energy taken out of the battery as Depth of Discharge (DOD, in %) [41, p. 133]. The available capacity is reduced if only a fraction of the active material of a battery cell is used in the electro-chemical reactions [19, p. 25]. This can be caused by the temperature-dependent inner resistance and

ageing processes [41, p. 143], [19, p. 199]. As a consequence, the resulting range of an electric vehicle is further limited.

A battery management system is used for battery state estimation (stored energy, remaining lifetime, temperature) and battery control (power and thermal management) [41, p. 172], [19, p. 43]. Its tasks include keeping the battery in states with low calendaric and cyclic ageing. Temperature-related ageing effects can be reduced with a thermal management or with charging and discharging current limits (called derating) which reduce heat generation [42]. However, active battery cooling or heating results in higher energy consumption, and current limitation results in reduced available power. Thus, the considered dependencies include the battery cooling and heating, ageing as well as derating, as shown in Fig. 2.1. All three have a direct effect on the driving characteristic and range of battery electric vehicles (BEV). Further properties of BEV are safety, quality, and monetary costs [43, p. 401], which are not considered in this work. Despite ongoing research on improved lithium-ion battery cell materials [19, p. 20–24], [44], the effect of different battery cell materials is not investigated within this work.

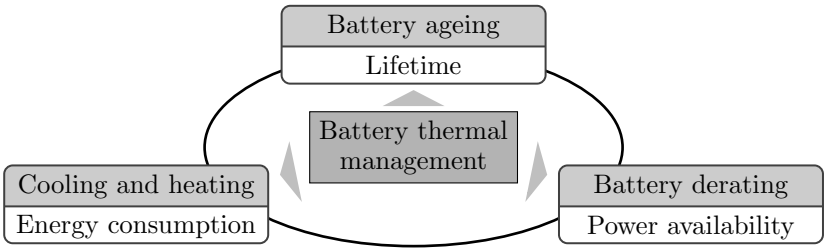


**Figure 2.1:** Battery thermal management related dependencies and their driver-perceivable effects.

#### 2.1.1.1 Battery ageing

Battery lifetime is limited by various ageing processes. Battery ageing can be categorized into cyclic ageing (due to charging or discharging) and calendaric ageing (over time) [44], [41, p. 139–140]. The battery ageing processes can

result in power reduction, due to an increased inner resistance [45], and capacity reduction, due to a loss of active material [44]. The capacity reduction can be described by the State of Health (SOH, in %) as ratio of the (reduced) capacity divided by the initial capacity [44]. Table 2.1 gives an overview of the key ageing processes with the corresponding category, conditions that enable or accelerate the ageing process, and if power or capacity are reduced as a result.

**Table 2.1:** Overview of key battery ageing processes solid electrolyte interface (SEI) formation, lithium-plating (LIP), electrolyte decomposition (ED), and mechanical damage (MeD), comparable to [46]. The processes are accelerated for comparably low (↓) or high (↑) battery temperature $T_b$, SOC, or battery current $I_b$.

| Process | Category | | Reduction in | | Reinforcing conditions | | |
|---|---|---|---|---|---|---|---|
| | Cyclic | Calendaric | Power | Capacity | $T_b$ | SOC | $I_b$ |
| SEI | X | | X | X | ↑ | ↑ | |
| LIP | X | | | X | ↓ | ↑ | ↑ |
| ED | | X | X | X | ↑ | ↑ | |
| MeD | X | | | X | | DOD↑ | ↑ |

The solid electrolyte interface (SEI) is a corrosion layer at the anode, which grows due to electrolyte decomposition, consuming lithium during cycling and increasing the inner resistance of the battery cell [47, p. 171,176–177], [44], [41, p. 158]. The SEI growth leads to a battery capacity and power loss, and is accelerated at high temperatures and high SOC [48, 49]. Lithium-plating (LIP) describes a lithium deposit, which causes capacity loss and dendrite formation [41, p. 157]. It occurs at the anode during slow intercalation of lithium, especially at high battery currents, high SOC, and low temperatures [41, p. 157], [47, p. 181]. Electrolyte decomposition (ED) occurs as electro-chemical side reaction and can lead to gas generation and SEI growth with the corresponding power and capacity loss [47, p. 175], [50], [51]. ED is accelerated with high temperatures and high SOC [50]. Mechanical damage (MeD) can be induced by volumetric stresses causing cracks and capacity loss [47, p. 189]. High battery currents and DOD accelerate the

process [47, p. 190]. As a result, battery ageing shows high dependency on the battery temperature [52, 44].

#### 2.1.1.2 Battery derating

The limitation of charging and discharging currents is called derating [42]. It leads to reduced power availability and, as a result, to reduced vehicle dynamics or increased charging times. The current limits depend on the battery temperature, SOC, and SOH [42, 53, 54, 55]. Additionally, derating can be adapted to the cell ambient temperature for a reduction of calendaric ageing [54]. An example of current limits dependent on the battery temperature is shown in Fig. 2.2. The limits can differ between charging and discharging [42]. Derating can be considered as an energy-efficient and robust alternative to active battery cooling or heating if the reduction in available power is acceptable [54].

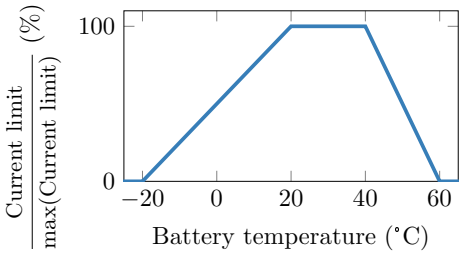


**Figure 2.2:** Example of temperature-dependent derating, based on [42], [19, p. 43], [43, p. 166].

#### 2.1.1.3 Battery cooling and heating

Active battery cooling or heating consumes energy. However, due to the 40 times lower energy density of batteries compared to gasoline and diesel, high energy efficiency of electric vehicles is required to achieve comparable ranges [41, p. 135]. An exemplary share of total energy consumption is shown in Fig. 2.3 for a Battery Electric Vehicle (BEV).

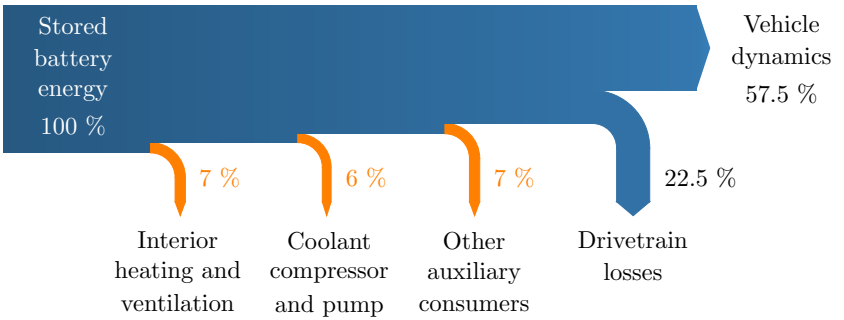


**Figure 2.3:** Share of battery energy consumption by auxiliary consumers (orange), drivetrain losses, and vehicle dynamics for a battery electric vehicle (BEV). The values are averages based on [41, p. 105], [14, p. 44], [56, p. 48], [57].

The energy consumption of the thermal management and other auxiliary consumers (orange in Fig. 2.3) takes a 1.5 to 5 times higher share than for combustion engine vehicles [41, p. 104–105]. Thus, the energy consumption of the corresponding components has a higher effect on the range of electric vehicles. This effect depends on the environmental conditions. Fig. 2.4 shows the auxiliary energy consumption share for different ambient temperatures. The share is lowest at an ambient temperature of 20 °C (8 %) and increases for both heating at lower temperatures (37 % at -10 °C) and cooling at higher temperatures (16 % at 30°C).

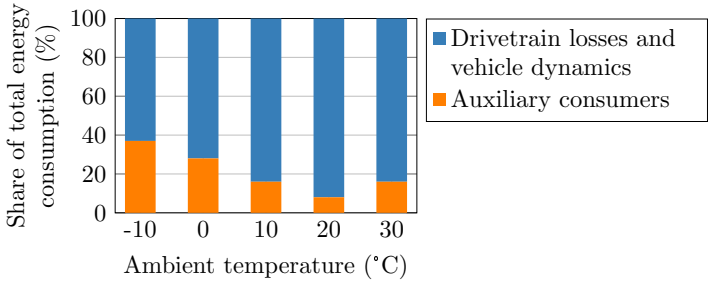


**Figure 2.4:** Share of auxiliary energy consumption for different ambient temperatures. The values are averages based on [14, p. 59–60].

### 2.1.2 Battery thermal management systems and control

The battery thermal management is responsible for battery cooling or heating in order to reduce temperature-dependent battery ageing and power limitation. The following sections present fundamentals of the battery thermal behavior, the components of a battery thermal management system (BTMS) and control types.

#### 2.1.2.1 Battery thermal behavior

Many battery properties are strongly affected by the battery temperature. Besides ageing and derating (see 2.1.1), various electro-chemical quantities and the resistance depend on the battery temperature [58]. For instance, the inner (ohmic) resistance, the SEI-related resistance, and the charge transfer resistance (related to lithium intercalation) decrease with increasing battery temperature [58]. However, increasing temperatures also accelerate battery ageing (see 2.1.1.1). Furthermore, large temperature differences between different cells result in limited usable capacity [19, p. 199]. The key processes causing a change in battery temperature are described in the following.

A physical equation of the change in battery temperature $T_b$ over time $t$ can be derived from energy conservation [1], [59, 60], [19, p. 209–210]. It can be formulated as in (2.1), considering self-generated heat $\dot{Q}_{b,gen}$, heat transfer with the environment $\dot{Q}_{env}$ and thermal management system $\dot{Q}_{tm}$, battery mass $m_b$, and battery heat capacity $c_{p,b}$ ($m_b \cdot c_{p,b}$ is also referred to as thermal mass).

$$\frac{\partial T_b(t)}{\partial t} = \frac{\dot{Q}_{b,gen} + \dot{Q}_{env} + \dot{Q}_{tm}}{m_b \cdot c_{p,b}} \tag{2.1}$$

Heat generation $\dot{Q}_{b,gen}$ can be calculated by an electro-chemical model (2.2) [1]. Three types of heat generation are included: heat generation due to ohmic resistance $\dot{Q}_{\Omega}$, reaction heat generation $\dot{Q}_{rct}$, and reversible heat generation from polarization $\dot{Q}_{rev}$ [60], [19, p. 207]. The heat generation equation depends on

the battery inner resistance $R_b$, battery cell current $I_b$ and voltage $U_b$, the open circuit voltage $U_{ocv}$, and the battery temperature $T_b$.

$$\dot{Q}_{b,gen} = \dot{Q}_{\Omega} + \dot{Q}_{rct} + \dot{Q}_{rev} = R_b I_b^2 + I_b(U_b - U_{ocv}) + I_b T_b \frac{\partial U_b}{\partial T_b} \tag{2.2}$$

Heat transferred to the environment of the battery $\dot{Q}_{env}$ is simplified with a combined heat transfer coefficient $H_{amb}$ for convection and radiation (2.3) [1], [61, p. 25–29]. It further depends on the surface $A_{amb}$, battery temperature $T_b$, and ambient temperature $T_{amb}$. Heat transfer with the thermal management system is described in the same way (2.4), with a heat transfer coefficient $H_{tm}$, intersection surface $A_{tm}$, and coolant temperature $T_{coolant}$.

$$\dot{Q}_{env} = H_{amb} A_{amb}(T_b - T_{amb}) \tag{2.3}$$

$$\dot{Q}_{tm} = H_{tm} A_{tm}(T_b - T_{coolant}) \tag{2.4}$$

Heat generation is mainly driven by the battery cell current $I_b$, which can be calculated by (2.5) with the power of the electric machine $P_{em}$ and of the consumers $P_{cons}$ [1]. The required power for the electric machine depends on the driving resistance force $F_{res}$, speed $v$, and efficiency of the electric machine $\eta_{em}$, transmission $\eta_{tr}$ and battery $\eta_b$ (2.6) [1], [41, p. 107–108]. The driving resistance force is composed of acceleration and climbing force, as well as aerodynamic and rolling friction [41, p. 99–103]. Amongst other parameters, it depends on speed $v$ and slope $\theta$.

$$I_b = \frac{P_b}{U_b} = \frac{P_{em} + P_{cons}}{U_b}, \tag{2.5}$$

$$P_{em} \approx \frac{F_{res}(v^2, \dot{v}, \theta, ...) \cdot v}{\eta_{em}\eta_{tr}\eta_b} \tag{2.6}$$

The presented equations are a simplified model of the battery thermal behavior, since many model parameters depend on the battery temperature, not all side reactions are considered, and heat transfer occurs in three dimensions. However, the model helps to understand the major dependencies of the battery temperature. More holistic models are more accurate, but might not be suitable for an

application in electric vehicles due to their high complexity [62]. Alternatively, equivalent circuit or data-driven models can be used as empirical gray-box or black-box models [63].

#### 2.1.2.2 Battery thermal management components

The thermal management of electric vehicles controls the temperature of the cabin, but also the temperature of drivetrain components such as battery, inverter, and electric machine. Fig. 2.5 shows an exemplary topology of a thermal management system, based on [64], [43, p. 169–172], [19, p. 48], including the cooling components (compressor, radiator, condenser), a chiller, two pumps, and two three-way valves. In this example, the battery can be cooled in two modes, dependent on the state of the three-way valves. At lower ambient temperatures, the battery is part of the cycle of the inverter and electric machine (blue), with the coolant getting cooled by the ambient air at the radiator. At higher ambient temperatures, the battery is cooled in a separate cycle (green) by heat removal at the chiller. The compressor and following condenser reduce the coolant temperature to supply the cabin air-conditioning (AC) and the chiller in this cooling mode (orange cycle), which increases the energy consumption [64]. Different architectures of the thermal management system are further analyzed in [65] considering their impact on electric vehicle range and including the usage of additional components such as heat pumps. The system architectures can differ in weight, costs, and energy consumption [43, p. 175]. A high energy consumption dependency on the cabin air-conditioning and battery thermal management is shown in [65, 12] for different environmental conditions.

#### 2.1.2.3 Battery thermal management control

The goal of battery thermal management control strategies is to find the optimal balance between energy consumption and battery temperature requirements (e.g. related to ageing and power availability) [66, p. 380], [67, 23, 68]. Different control strategies can be used for the battery thermal management, including a fixed

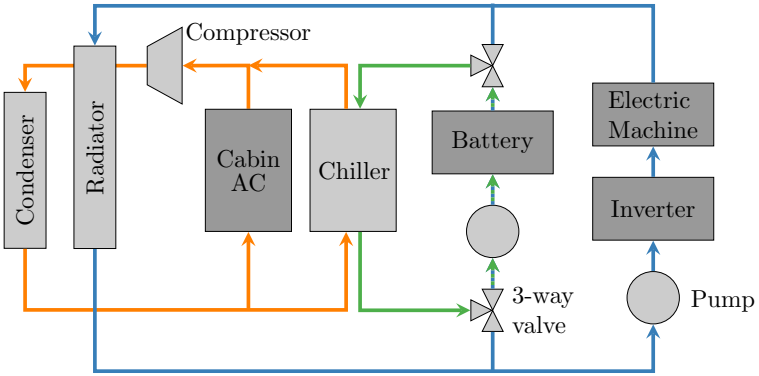


**Figure 2.5:** Example of a thermal management topology for electric vehicles with a focus on cooling components, based on [64], [43, p. 169–172] [19, p. 48].

threshold control, proportional-integral-derivative (PID) control, (fuzzy) rule-based control, observer-based control, and Model Predictive Control [15], [19, p. 211]. Complex algorithms and models for state estimation and control of a battery thermal management system can be integrated in a cloud-based system, which provides more computational resources and access to large fleet data for an improved control [19, p. 308–313], [69].

The fixed threshold control can be also called thermostat control [70] or setpoint control [71]. Battery cooling or heating is activated if a threshold temperature is exceeded, and deactivated according to a hysteresis to reduce the switch frequency [19, p. 211]. Systems with high switch frequency profit from a more stable control using PID [19, p. 211]. Better control performance can be achieved using (fuzzy) rule-based control [19, p. 211], [72, 73, 74, 75]. However, rule-based control is limited in flexibility and depends on domain knowledge and experience [15]. In [76], an observer is derived from a reduced order model and experimental data, providing an estimation of the battery cell core temperature as input for a fixed threshold control. Inhomogeneities of the battery temperature and required cooling flow are reduced with the observer-based control [76]. The battery thermal management control can be improved with a predictive control

using information about the route ahead from the navigation system and from weather forecasts [66, p. 380], [16, p. 107–108,116–117], [6].

## 2.2 Model Predictive Control

Model Predictive Control (MPC) uses a system model to optimize control values according to a cost function [77, p. 4], [78]. Fundamentals such as properties, parameters, and types of MPC are introduced in 2.2.1. Examples of MPC for a predictive battery thermal management are provided in 2.2.2.

### 2.2.1 Fundamentals

According to [77, p. 4], [78], and [79, p. 503–504], a system model predicts the system states $\hat{y}$ for different control values $\tilde{u}$ for a defined prediction horizon in each MPC control step. Fig. 2.6 shows a simplified MPC scheme. The predicted system states $\hat{y}$ are input to a cost function $c$ (also known as objective function) which MPC tries to minimize. The control values $u$ that lead to minimum costs and satisfy (optional) constraints $j$ are chosen until the next control step is reached after the control horizon $h_{ctrl}$. Due to the iterative sliding window approach, MPC is also known as Receding Horizon Control (RHC) [78]. Disturbances $D$ can impact the MPC performance on the observed system state $y$.

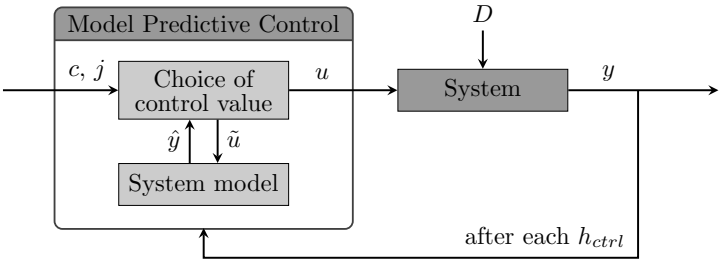


**Figure 2.6:** Example of a model predictive control scheme, based on [78], [79, p. 505].

The control horizon $h_{ctrl}$ may differ from the prediction horizon $h_p$ [77, p. 19], as shown in Fig. 2.7. The prediction horizon should capture the effect of the control variable $\tilde{u}$ on the predicted system behavior $\hat{y}$ [78]. Large prediction horizons or variable prediction horizon can be used for improved controller performance, with the downside of additional computation time [77, p. 281], [80, p. 325,330–331]. The best performance is achieved in [16, p. 118–119] with prediction horizons larger than the control horizon and below the real-time limit. Furthermore, the step size can be adapted according to the expected system dynamics, for example with larger steps for less dynamic situations or when the prediction accuracy is expected to be high [80, p. 353–356]. In this case, computation times can be reduced compared to an MPC with constant step size, which needs to be as small as necessary to cover all situations (including higher system dynamics). Additionally, computation times can be reduced by so-called move blocking of the control variable [78]. By freezing the control value within a blocking horizon $h_b$, move blocking reduces the degrees of freedom of the MPC and, as a result, the number of variants that need to be evaluated by the optimization algorithm [78]. The blocking horizon can be set to steps at the end of the prediction horizon, for which the prediction might be less accurate. However, move blocking can result in worse controller performance [78].

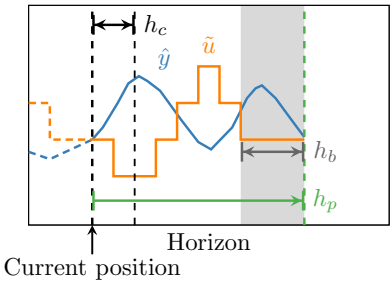


**Figure 2.7:** Example of model predictive control horizons, based on [78].

The cost function $c$ of an MPC describes unwanted system states or control values, for example if they lead to high energy consumption [80, p. 45–46]. The

MPC control law aims at minimizing the cost function by choosing corresponding control values [77, p. 21]. The cost function can be a weighted sum of individual cost functions, which express different, possibly contradicting optimization goals of the system and which is also known as multiobjective MPC [77, p. 213–214]. According to [77, p. 214], the individual cost functions can be considered as soft constraints, which means undesired values can be reached or exceeded, but result in higher costs. On the contrary, hard constraints such as safety or hardware limits have to be obeyed anytime. Hard constraints can be categorized into overshoot constraints, band constraints or terminal constraints, applied on control values (inputs) or system states (outputs) [77, p. 177–183]. Constraints can be used for optimization of operating conditions, transitions between operating points, or minimization of system errors [77, p. 187].

Three key properties of MPC are feasibility, stability, and robustness. A predictive control is not feasible if the optimizer cannot find a solution, for example when hard constraints cannot be obeyed due to sudden perturbation [78], [77, p. 206–207]. Possible solutions include relaxing the corresponding constraints to soft constraints, neglecting constraints in the close horizon steps, or using a back-up controller or control value [77, p. 207–208]. Stability is achieved if the MPC can make the system reach the desired state even under the influence of disturbances [78]. MPC robustness relates to the accuracy of system model predictions under system uncertainty (e.g. in disturbances, system models, boundary conditions, and measurement noise) [78, 81]. The corresponding uncertainties can be considered by a robust MPC scheme, for example with a prediction of a band of future system states which includes the real system behavior, and is narrow for low prediction uncertainty of good system models [77, p. 217–218]. A robust MPC can choose the control values by minimizing the cost function for the most expected future states or the worst states of a predicted set of probable system states [77, p. 224], [82].

According to [77, p. 13], the system model is the most important part of an MPC. It needs to be as complex and complete as necessary to achieve the required accuracy. On the other hand, an intuitive model understanding and efficient predictions of the future system states are required as well. An advantage of MPC is that many

types of system models can be used [78]. For example, dynamic, nonlinear processes require nonlinear models for a predictive control, which is also known as Nonlinear Model Predictive Control (NMPC) [77, p. 249]. According to [77, p. 252], NMPC system models can be:

- Fundamental models based on physical equations, e.g. an energy balance. If the process is close to linearity, the model can be linearized at different operating points and composed as piecewise affine system [83].
- Data-driven, empirical models, e.g. Neural Networks.
- Gray box models, which combine fundamental models with empirical methods, e.g. Physics-informed Neural Networks (PINN) [84].

NMPC model development and the optimization problem including NMPC stability and robustness are more complex and require more computational resources than linear MPC or classic controllers (e.g. PID) [77, p. 187,250,252]. Therefore, explicit MPC and rule-extraction from MPC are introduced as computationally less expensive and easier to integrate in a target system [78, 85]. In both approaches, the computationally expensive optimization is conducted offline for a representative set of scenarios and the resulting control strategies are translated into look-up tables or decision trees used in application [78, 85]. Since these models require simulations of a comprehensive set of scenarios, their most popular applications are binary controllers with short prediction horizons [78].

Robustness in MPC requires an uncertainty model for optimization with consideration of constraint satisfaction for mean expected or worst-case system states [78, 81]. In contrast to deterministic uncertainty models, Stochastic Model Predictive Control (SMPC) explicitly integrates system state probabilities, for example using Markov Chains or Gaussian distributions [82]. According to [81], SMPC control optimization can be based on the predicted mean and standard deviation. For example, the violation of state constraints can be checked for the predicted mean state, or the violation probability should stay below a threshold, which is also called chance constraint [81]. The threshold needs to be a compromise between the optimization costs and the satisfaction of the chance

constraints [81]. SMPC can be applied for both linear and nonlinear MPC [82]. Due to possible unknown system uncertainties, empirical stochastic models are developed based on data from the target domain with the expected probability distribution of the application [81]. Thus, challenges of SMPC are the stochastic model development for complex systems, computational requirements, and uncertainty propagation between the control steps [81].

### 2.2.2 Examples for battery thermal management

Model Predictive Control (MPC) can be applied for a predictive battery thermal management. Table 2.2 provides an overview of applications for each of the model types: fundamental, empirical, and gray box.

In the references from Table 2.2, fundamental models are simplified by discretization and finite sets, look-up tables, or model reductions. Empirical models are derived from simulation or vehicle data of drive cycles. Uncertainty is considered by an equation-based state observer in [89] or in an SMPC with a probability distribution in [70, 68], and with unequal discretization of the distribution for better computational performance in [18]. Reference models for performance comparison include fixed threshold, PID (also without the differential part as PI) and rule-based models. Dynamic Programming (DP) is used as baseline in [18]. DP is a global but computationally expensive optimization method, such that it is more advantageous in a development stage rather than for online optimization [56, p. 59–60], [18]. In [68], MPC computation times range from 0.23 s for a prediction horizon of 5 s to 2.7 s for a horizon of 60 s. In [17], the computation time is 1.82 s for a 50 s prediction horizon with 5 s sampling (i.e. 10 steps in the prediction horizon). Prediction horizons range from 30 s to 60 s in [89, 23]. The prediction horizon affects the temperature tracking error, energy consumption reduction, computation time, and the occurrence of disturbances in [92].

The electric power of the thermal management actuators is used as control value, additionally with the valve mode in [86, 93, 91]. Alternatively, the mass flow rate is used in [23] and battery heating and cooling thresholds are adapted in [16,

**Table 2.2:** Applications of Model Predictive Control for the battery thermal management.

| Type | Source | Model details | Reference model |
|---|---|---|---|
| Fundamental | [23] | Linearized model with discretization, Neural Network for speed prediction | Fixed threshold |
| | [16, p. 116ff] | Reduced system model | Fixed threshold |
| | [56, p. 109ff] | Reduced system model, with Pontryagin's Maximum Principle | Fixed threshold, DP |
| | [86] | Uses functional mock-up unit | PID |
| | [87, 88] | Includes look-up tables, finite sets | Rule-based |
| | [89] | Finite sets, state observer as uncertainty model | PID |
| Empirical | [17] | Parameter fitting, based on a fundamental simulation model | Fixed threshold, PI |
| | [90, 91] | Neural Network, based on a fundamental simulation model | Neural Network (without MPC) |
| Gray box | [70, 18, 68] | SMPC with probability distribution of heat generation from simulation data | Fixed threshold, MPC, DP |
| | [64] | Neural Networks, connected according to physical relations, trained with vehicle data | Rule-based |

p. 117–118]. The optimization goal is defined by a cost function of the battery temperature (e.g. considering ageing) and energy consumption. The battery temperature costs are included as difference from a setpoint (which can be derived from an operational window [86]) or as an empirical cost function [87, 88]. Costs for increased energy consumption are based on the electric power of the actuators, mass flow, or valve opening. Furthermore, a stable operation of thermal management components (low frequency of changing operation modes) and hardware constraints are included in [23, 86]. The difference between cabin temperature and cabin AC setpoint temperature is considered in [91, 17, 86].

The effect of predictive battery cooling and heating threshold adaption during the trip on energy consumption is analyzed in [16, p. 116–123]. The standard cooling strategy cools the battery down to 20 °C [16, p. 121]. In contrast, a higher battery cooling threshold reduces energy consumption, due to less cooling demand and higher efficiency of the battery at higher temperatures, with an acceptable increase in battery ageing [16, p. 120–122]. Large prediction horizons show greater reduction in energy consumption, especially for more dynamic drive profiles which lead to larger heat generation [16, p. 123], [92]. The largest energy consumption reduction of up to 9 % is achieved for simulated trips of 1 h with high ambient temperatures of 35 °C [16, p. 122–123]. Compared to a fixed threshold control, a predictive thermal management reduces the thermal management energy consumption of a hybrid electric vehicle (HEV) by 50 % on average for seven scenarios in [64]. Furthermore, predictive control with vehicle speed prediction reduces the energy consumption of the BTMS of a Plug-In HEV (PHEV) by up to 31 % in [23] without a negative ageing impact. In [17], battery cooling energy consumption is reduced by 11 % and total energy consumption by 3 % for simulations of a predictive battery thermal management in PHEV.

Further insights into battery thermal management strategies are provided by a system analysis and the Dynamic Programming (DP) baseline in [56, p. 43–46,68-80]. The investigation is decoupled from the control of the thermal management actuators, which can be optimized separately for shorter time horizons [56, p. 66]. According to the analysis, the battery thermal management can be improved with a predictive cooling during a downhill drive, if it is followed by an uphill drive [56, p. 45]. Another scenario is a cooling activation early in the trip for reduced battery ageing during drive, or no battery cooling at all if ageing is acceptable without [56, p. 78]. Furthermore, the importance of an accurate route estimation for a predictive battery thermal management is discussed in [56, p. 132-134].

MPC requires accurate system models, which can be improved with data-driven methods [15], [94, p. 10–11,58–59]. For example, Machine Learning can be used for fast and scalable model development without the need of model assumptions [94, p. 10]. However, in this case representative data are required and model uncertainty and complexity need to be considered for an application in MPC [94,

p. 11]. Additionally to the battery thermal management, MPC with Neural Networks shows good performance in other applications such as longitudinal vehicle control [95], energy management of hybrid vehicles [96], chemical processes [97], and thermal management of buildings [98, 99, 100]. Therefore, corresponding Machine Learning methods are presented in the following section.

## 2.3 Machine Learning for sequential data

According to [24, p. 27–28], Machine Learning (ML) is an automatic modeling of a task by fitting an algorithm to training data without explicit rule construction. ML tasks can be categorized into supervised learning (regression, classification) and unsupervised learning (clustering) [24, p. 55]. Supervised learning can be applied when the data consist of observations (data points) which include the target values (e.g. dependent variables) [24, p. 63]. An exploratory data analysis (EDA) and data processing are a crucial step before model training [24, p. 37,43], [101, p. 27–28]. Model hyperparameters are tuned using validation data with the same structure. Evaluation of model performance is based on test data which have not been part of the training and validation data. The model can be continuously improved with new data after its deployment to application. Fig. 2.8 provides an ML pipeline with the corresponding steps. The principles and steps are comparable for Deep Learning (DL) methods, which can be considered a subcategory of ML using complex (deep) Neural Networks [24, p. 7–8].

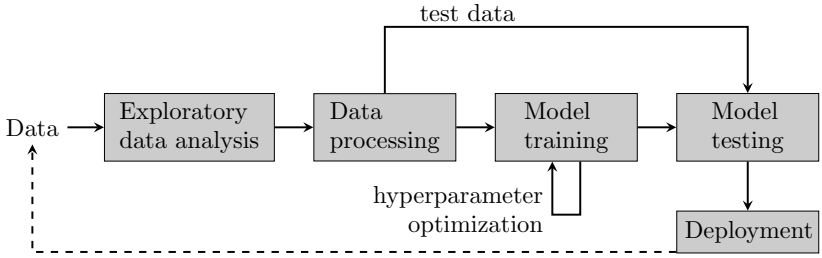


**Figure 2.8:** Example of a Machine Learning pipeline, based on [24, p. 28].

The advantages of ML include a fast, automated development of models and corresponding rule discovery with better generalization to unseen scenarios, compared to classic function development with hard coded rules [24, p. 27,30]. However, challenges of ML development are the required data management and processing, explainability, reproducibility, as well as finding the right trade-off between accuracy (overfitting) and generalization (underfitting) [24, p. 27,119–121], [101, p. 2–4]. Methods for evaluation of model performance depend on the tasks, data type, and domain requirements [24, p. 60]. A comprehensive evaluation of DL models is necessary due to their tendency to learn unintended shortcuts instead of the desired decision rules [33]. This work focuses on sequential data, also called time-series data, which is characterized by a meaningful order of elements in a sequence, as well as the frequency (or sampling) of the elements [24, p. 39,204], [101, p. 52]. The pipeline steps are further explained for sequential data in the following, including data analysis and processing, corresponding ML methods, and evaluation.

### 2.3.1 Data analysis and processing

The performance of Machine Learning models can be improved when large data sets are available, such that deeper Neural Networks can be developed [25, 102]. Data from different sources, also called cross-domain data, can be joined to provide more comprehensive information of the task to the ML model [20]. For example, informed ML combines measurement data with simulation data, which might be cheaper and safer to obtain than conducting experiments [103, 104]. Besides the opportunities, using big data for Machine Learning brings additional challenges to data analysis and processing considering data size, heterogeneity, possible noise, and redundancies [105, 106, 107]. Table 2.3 summarizes key steps of sequential data analysis and processing.

Exploratory Data Analysis (EDA) analyzes sequence characteristics such as level (mean value), trend (rate), stationary behavior (trend-stationary, difference-stationary), and seasonality (periodicity) [101, p. 52–55]. Resampling might

**Table 2.3:** Overview of sequential data analysis and processing steps, based on [101, p. 28].

| Focus | Processing step | Examples |
|---|---|---|
| Analysis | Sequence characteristics | Trend analysis, resampling |
| | Univariate or multivariate analysis | Occurrence of values, dependencies between features |
| | Correlation analysis | Pearson and cross correlation |
| | Clustering | Occurrence of characteristic patterns |
| Processing | Missing or faulty data | Interpolation, outlier removal |
| | Feature engineering | Feature construction, feature selection, dimension reduction |
| | Data balancing | Undersampling, oversampling |
| | Data augmentation | Mathematical transformation, generative models |
| | Range adjustment | Rescaling, normalization |

be required to achieve the same sampling frequency [24, p. 204], for example downsampling by aggregation or upsampling by interpolation [101, p. 52]. Univariate or multivariate analysis includes occurrence visualizations (e.g. boxplot, histogram, pair plot) and statistical values (e.g. mean, variance, skewness) [101, p. 31,36ff]. Pearson correlation helps to understand dependencies between features (redundancy, e.g. for correlation values greater than 0.9 [108]) and between a feature and the target variable (explainability) [101, p. 33], [24, p. 75–77]. Cross correlation checks for similar patterns between two observed sequences [109]. It is known as autocorrelation if patterns within the same sequence but different time-lags are analyzed [109], [101, p. 55]. Characteristic patterns of sequential data can be identified with sub-sequence clustering (e.g. using k-Means clustering) which groups similar sequences into the same cluster [109].

Missing and faulty data, also referred to as data noise, can be resolved by data imputation (e.g. with interpolation, forward fill, mean value, k-Nearest Neighbor) or data removal (e.g. with outlier analysis) [24, p. 42,68–69], [101, p. 52], [110,

106]. Outlier analysis can be used to identify and remove noisy data [111, p. 415], [112, 113]. Outlier types of sequential data include point, subsequence, and time-series outliers [113]. An exemplary method for outlier removal is isolation forest [114, 112], [111, p. 161–164]. Isolation forest is composed of isolation trees, which sort outliers in less deep branches, assuming the outliers to be located in sparse regions [111, p. 161–164]. Outlier detection is challenging for high-dimensional data, when less relevant and noisy features hide relevant outliers, which might be resolved with dimension reduction techniques [112]. Noise removal by smoothing and stationarity by differencing can improve the ML performance [115, 116].

Another step in data processing is feature engineering, which consists of the construction and selection of features that describe the system behavior as complete as possible for accurate predictions [24, p. 52–55]. This is especially required for classic ML models compared to Deep Learning methods [24, p. 142]. ML model performance can be improved with physics-informed feature engineering, which explicitly adds mathematical nonlinearities from the physical equations to the feature set [108, 117]. After feature construction, feature selection is conducted to reduce model complexity, noise, and redundancy, which reduces overfitting, the required computational resources and the required amount of training data [101, p. 66–67], [108]. The feature set can be reduced by feature selection methods (e.g. correlation or scoring methods) or dimension reduction [24, p. 77–81], [108]. Dimension reduction can be achieved with Principle Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP), or t-distributed Stochastic Neighbour Embedding (t-SNE) [118]. A reduction to two dimensions allows a visualization to check for overlap and relations between groups [101, p. 64–66].

Data imbalance describes a skewed ratio between the amount of data of minority classes (system behavior with low occurrence) and of majority classes (system behavior with high occurrence), starting with ratios of 1:100, which can lead to worse ML performance due to a prediction tendency to the majority classes [24, p. 73–75]. Data balancing methods include an undersampling of the majority

classes, oversampling of the minority classes, and adapted algorithms or metrics [24, p. 73–75]. For example, ML performance is improved in [119] with cluster identification using k-means clustering and oversampling using the Synthetic Minority Oversampling Technique (SMOTE). However, cluster overlap needs to be taken care of in clustering based data balancing [120]. Adjusting the amount of sequential data of a class can be done by windowing and window overlap (also known as stride) [24, p. 74], [121]. K-means clustering and simultaneous under- and oversampling of the respective classes increases the ML performance for sequential data in [122, 123]. Furthermore, better performance of ML models for sequential data is achieved with data balancing in [124, 31, 125, 126, 127]. Data of minority classes can be increased with data augmentation [121, 128].

Data augmentation can be used to synthetically generate new data (without the need to collect new measurements or conduct simulations) to improve the generalization ability of ML models and reduce data imbalance [128]. Sequential data can be generated from existing data by mathematical transformation (e.g. scaling, jittering), pattern mixing, and generative models (e.g. Generative Adversarial Networks, GAN) [128, 129, 130]. Augmentation of sequential data can improve the performance both for classification [128, 129, 131, 132, 133, 121] and forecasting [129, 134, 135]. However, the validity of the generated data (e.g. considering nonlinear physical dependencies and noise) as well as the development effort (e.g. for training of GANs) need to be taken care of [121, 131, 130].

Dependent on the ML method, an adjustment of the value ranges might be required, for example by min-max-scaling per feature, standardization (scaling with standard deviation), and normalization (considers distribution) [24, p. 70–72]. ML methods for sequential data are described in the following.

### 2.3.2 Forecasting of sequential data

Machine Learning models for sequential data forecasting can be categorized as follows, based on [95, 136, 137, 138]:

1. Single-step: single model which predicts a single value (at the horizon end).
2. Single-step multi-model: a group of single-step models which predict a single value each for a different horizon.
3. Multi-step recursive: single-step model which recursively predicts the following steps based on the prediction of the corresponding previous step.
4. Multi-step multi-horizon: multi-input multi-output model, also known as sequence-to-sequence [139], [24, p. 206], which predicts all steps at once.

Sequence forecasting models can be further distinguished by the number of input or output variables, as univariate (one quantity) or multivariate (multiple quantities) [138, 140, 141]. Classic ML regression models can be used for the single-step, single-step multi-model, and multi-step recursive [142, 143, 144]. In consideration of their limited complexity, feature reduction can be applied by converting the sequences into statistical values such as average and standard deviation [145, 146]. Multi-step multi-horizon models include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Transformer models [138, 147, 139, 148, 149]. Compared to the other three categories, the advantage of these multi-step multi-horizon models is their ability to explicitly represent nonlinear, multivariate sequence information [138, 147, 148, 150]. Additionally, a multi-step recursive prediction can lead to error accumulation over the horizon steps and requires all additional inputs for future states [137, 138, 148]. Thus, multi-step multi-horizon models show better performance in [136]. Furthermore, Neural Networks which are trained on a large data set of sequences are more accurate than locally trained, statistical methods such as Auto-Regressive Integrated Moving Average (ARIMA) [28]. A special type of Recurrent Neural Network is the Nonlinear Auto-Regressive with eXogeneous input (NARX) model, which extends a nonlinear auto-regressive model with additional input features [151, 152]. A NARX model takes input data from previous and current steps and can be used as multi-step recursive or auto-regressive model [152].

Convolutional Neural Networks (CNN) apply a convolution filter (also known as kernel) with trainable weights and tunable size over a vector (one dimensional

convolution) or matrix (two dimensional convolution), followed by a pooling layer (e.g. max pooling) for dimension reduction [24, p. 161–169,214–216], [148, 138, 28]. CNN can be used for both univariate models (with one dimensional convolution) and multivariate models (with parallel channels or two dimensional convolution, which requires a consistent sampling frequency) [138], [24, p. 215]. Deep CNN contain repeated convolution and pooling layers with increasing abstraction level from inputs to outputs [147, 138]. A decreasing number of nodes from inputs to outputs can reduce the training time and model size of deep architectures, while achieving comparable prediction accuracy [153]. Multi-scale channels can further increase the accuracy of CNN [93, 154]. Flatten layers and dense layers (fully connected with activation functions) can be used for dimension reduction to a specified number of output nodes [24, p. 161–169].

Recurrent Neural Networks (RNN) use memory nodes which take previous states into account [24, p. 205–207]. Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) can be considered as a special type of RNN [147]. GRU and LSTM use additional memory gates (input, forget, output) such that they do not suffer from exploding or vanishing gradients (constrained weight tuning during training [24, p. 158]) compared to classic RNN [138, 148, 28]. GRU are less complex than LSTM, since they substitute the input gate and forget gate by one combined gate [138]. Thus, using GRU can reduce the computation cost but might result in lower prediction accuracy [138, 147], [24, p. 213–214]. Further improvements can be achieved with bidirectional layers, including the attention method or a combination with CNN [138, 147]. As an alternative to RNN with attention, Transformer models include self-attention without recurrent layers, which can result in faster model training or higher accuracy [147, 149, 28, 155].

In order to avoid overfitting, regularization methods can be used, such as L1 and L2 weight regularization, dropout, and batch normalization [24, p. 160], [156]. Dropout multiplies a Bernoulli distributed random noise (with value 0 or 1) to each node of a hidden layer during training, such that nodes are randomly removed with a given probability for the value 0 (which is a hyperparameter of dropout) [156], [24, p. 160]. Batch normalization rescales the outputs of a hidden layer by mean and variance [156]. Both methods can be combined, with dropout

followed by batch normalization [156, 157]. Random weight initialization before model training can further improve the model performance [24, p. 159].

Sequence forecasting models can provide the prediction uncertainty with distribution functions (e.g. Gaussian) or quantiles [28]. Quantile prediction models are not limited to a distribution function and can be useful for decision makers and in optimization [27]. Corresponding methods are based on quantile regression using the pinball loss [28]. Table 2.4 provides an overview of Neural Network applications with quantile regression for sequence predictions. Quantile regression models can be improved with additional constraints and weights [158], generative quantile copula [159], or incremental spline quantile functions [160]. Reference models can be based on classic regression methods (e.g. Quantile Random Forest) [161, 162, 163].

**Table 2.4:** Neural Networks for probabilistic forecasting of sequences using quantile regression.

| Base model | Application | Source |
|---|---|---|
| Neural Network (NN) | Electric load, traffic, website traffic | [164, 161, 163, 160] |
| CNN | Retail, electric load, traffic | [27, 159, 165] |
| LSTM | Retail, electric load, traffic, website traffic | [27, 166, 167, 158] |
| Bi-directional LSTM | Electric load | [140] |
| LSTM with attention | Electric load | [162] |
| Ensemble (e.g. with NN, CNN, LSTM) | Wind power, electric load | [168, 169, 158, 170] |
| Transformer | Retail, electric load, traffic, finance, biogas production | [171, 172] |

Hyperparameter optimization (HPO) leads to better ML performance than using the default hyperparameters or manually adjusting the hyperparameters [173, 174]. The performance of different HPO algorithms depends on the domain and

data set [175]. For example, Bayesian optimization is used for HPO of Neural Networks for sequence forecasting in [176, 177, 178]. Bayesian optimization iteratively estimates and evaluates hyperparameters, based on previous trials and an acquisition function [173, 175]. According to [175, 179], the acquisition function balances exploration and exploitation of the search space, for example with estimation of the expected improvement. Compared to HPO with grid search or random search, Bayesian optimization is faster due to its guided search [173, 175]. HPO can be extended by data balancing parameters and architectural design choices [180, 181, 178]. In [175], Bayesian optimization is more robust because of model diversity, early-stopping, parallelization, and an adjustment of the optimization function. Furthermore, Bayesian optimization is faster when the computation time is included in the acquisition function [182], or when using data subsets for training and validation, in parallel or with growing size over the optimization trials [183, 178]. An analysis of hyperparameter importance can provide insights into explainability of HPO [184, 178]. According to [185, 173], Bayesian HPO can be improved in combination with the Hyperband optimization method [186]. Moreover, model optimization can be extended to simultaneously optimize data processing and model hyperparameters, for example using Genetic Algorithms for optimized LSTM models [187].

The prediction performance of sequence forecasting models can be evaluated using regression metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Coefficient of Determination ($R^2$) [188]. Probabilistic forecasts can be evaluated using the Winkler Score (WS), Crossover Rate Score (CORS), sharpness ($\psi$) [158], as well as calibration (share of true values within quantiles) [189]. Furthermore, an evaluation of probabilistic sequence forecasting models can include the prediction robustness towards perturbation and noise [190]. An analysis of feature importance provides insights into the contribution of each input variable to the prediction [162]. In [162], it is calculated as aggregation of the LSTM attention weights based on the training data, both for the whole sequence and dependent on the horizon step (for temporal importance evaluation). Besides the consideration of model weights, feature importance of Neural Networks can be analyzed with input permutation [191]. For example, permutation

feature importance can be obtained from the increase in prediction error after randomly shuffling the values of the corresponding feature in the test data [192].

### 2.3.3 Application-related topics

This section provides an overview of ML applications for electric vehicle batteries, as well as methods which support model deployment to vehicles. An overview of Neural Networks (NN) for battery temperature prediction is given in Table 2.5. NNs are used for state estimation (also known as soft sensors) to predict the current battery temperature, heat generation, or heat transfer (with the Nusselt number). The NNs are trained with simulations or measurements of battery cells, packs or vehicle batteries. Single-step NNs are used for single-step forecasting or recursive multi-step forecasting (e.g. for application in Model Predictive Control [90, 91, 64]). Single-step prediction of the change in battery temperature showed better accuracy and robustness with LSTM than with GRU in [193]. Furthermore, multi-horizon models are used for a direct multi-step forecasting (sequence to sequence). Multi-horizon LSTM performs better than statistical models as well as classic regression models (including Decision Tree and Random Forest) in [194]. Battery temperature prediction is more accurate with GRU than with LSTM and CNN in [195], and best performance is achieved with a combination of CNN and Transformer model (RMSE: 0.12 °C). However, the change in battery temperature stays within a range of 3 °C (and less for the test sequences).

Automotive software contains millions of conditional statements, functions, and function calls due to (growing) complexity [203]. Software complexity is increased by a high number of variants, for example for countries with different climate conditions, regulations, and customer preferences [203]. Software development for automotive ML applications requires additional (time-consuming) steps, such as data collection, data analysis and processing, as well as ML model training and evaluation (additionally to classic hardware and software testing) [204, 205]. Limited development times require a compromise on model accuracy [204]. Furthermore, ML models and the corresponding data need to be interpretable and

**Table 2.5:** Neural Networks for battery temperature prediction.

| Category | Type | Training data | Source |
|---|---|---|---|
| State estimation | NN | Pack measurement | [196] |
| | LSTM | Cell measurement | [197] |
| | NN (heat generation) | Cell measurement | [198] |
| | NN (Nusselt number) | Pack simulation | [199] |
| Single-step model | NN | Pack simulation | [90, 91] |
| | NARX | Vehicle measurement | [64, 200] |
| | NARX | Cell simulation | [201, 202] |
| | GRU, LSTM | Cell measurement | [193] |
| Multi-horizon model | LSTM | Cell measurement | [194] |
| | CNN, GRU, LSTM, Transformer | Cell measurement | [195] |

understandable for safety and quality assurance [206], which is difficult due to the probabilistic ML behavior [205]. Model deployment, system integration and monitoring further increase the technical complexity of ML application [207]. ML applications in automotive electrics and electronics (E/E) systems require large processing power, flash memory, and Random Access Memory (RAM) for data storage and processing [208]. Moreover, the communication systems need to handle the transmission of sensor and map data for distributed functions [208]. Applications of deep Neural Networks have high memory requirements considering their number of model parameters and mathematical operations, which motivates memory optimization with corresponding runtime scheduling [209, 210].

Considering the need of large data sets for ML training, Transfer Learning (TL) of an existing ML model can help when only few data of the target application in another domain or task are available [211, 212]. TL transfers the knowledge of low-level patterns and corresponding feature extraction from a pretrained Neural Network to the target domain [24, p. 169], [211]. Model-based TL methods

include fine-tuning a pretrained model, partial freezing of model layers or nodes, and architecture modifications (e.g. adding layers) [212]. The training procedure can be adjusted to keep the knowledge of the pretrained model, specifically for low-level layers, by reduction of number of epochs or learning rate (learning rate for NN layers increasing from input to output) [212]. In [213], Bayesian hyperparameter optimization is applied to Transfer Learning models, with better performance using the same set of hyperparameters for the pretrained model and fine-tuning, compared to a separate HPO. While the computation time of predictions with deep Neural Networks in applications is comparable to classic ML models, training can be more time-consuming [138]. However, Transfer Learning reduces the training time compared to training a model from scratch [211].

The requirement of interpretable ML can be addressed with feature importance analysis or Decision Trees extracted from Neural Networks (NN) [206]. Furthermore, the computational requirements of NNs can be reduced by quantization with look-up tables [214] or replacing convolutional layers by decision trees [215]. The replacement of Convolutional Neural Networks by Decision Tree Ensembles further reduces the computational complexity with comparable accuracy in [209]. Decision Trees can be extracted from trained NNs using genetic algorithms [216] or custom splitting methods with fidelity pruning [217]. Alternatively, interpretable fuzzy rules can be derived from NNs [218]. Additionally, Decision Trees can be used for extraction of interpretable and less computationally expensive rules from Model Predictive Control (MPC) [219, 220, 85]. For example, [85] employ time-series clustering of MPC simulations and develop Decision Trees with low complexity for classification of the clustered control strategies.

## 2.4 Research gaps

The literature review covers key properties of electric vehicle batteries and their thermal management, Model Predictive Control (MPC), as well as Machine Learning (ML) methods for sequential data. The energy consumption and range of electric vehicles depends on various consumers, including the battery thermal

management which is required to reduce temperature-related battery ageing and power limitation. MPC can be used for an optimized battery thermal management, with an explicit definition of cost functions that meet the corresponding requirements and with predictions from Neural Networks (NN). (Deep) Neural Networks can be built for multi-horizon sequence forecasting, but require large data sets and corresponding data analysis and processing methods. Sequence forecasting models can be extended by the prediction uncertainty. Additional challenges for the deployment of ML models include computational requirements and interpretability. Most battery temperature forecasting methods are based on single-step models and either simulations or measurements of battery cells or packs.

Despite the growing research on Machine Learning and its application to battery (thermal) management, several research gaps can be identified. The first gap consists of the combination of multi-horizon NNs for probabilistic sequence prediction with both history inputs and foresight inputs, as well as their training on balanced and augmented data from both simulations and fleet measurements. This combination is not covered in literature, although it potentially achieves prediction models with high accuracy, using information about the route ahead, and the ability to predict simulated behavior that is not present in measurements. Furthermore, the prediction of quantile sequences potentially leads to more flexibility and robustness for later application, in contrast to predictions of fixed distribution functions or without any uncertainty quantification, which is still more common in literature.

The second gap is identified as MPC using accurate sequence predictions and the prediction uncertainty from such a NN, with an explicit modelling of key system properties in the optimization cost functions and variation of novel strategies that the NN learned from simulations. The design of MPC with such a NN is not present in literature, despite its potentially improved control strategies with reduced optimization costs and high robustness against uncertainty.

As a third gap, Transfer Learning and rule extraction need to be investigated to address application requirements for deployment of such prediction models

and predictive control. Specifically, all three research gaps are present for their combined application for battery temperature prediction and predictive battery thermal management on vehicle level. In contrast, the reviewed literature targets single elements of these gaps and often focuses on battery cell or pack level. Thus, this dissertation aims at filling these three research gaps to improve the energy-efficiency of an electric vehicle battery thermal management. In summary, the following research questions are raised:

1. How can we use a Neural Network to predict battery temperature sequences with consideration of its prediction uncertainty and foresight inputs, based on large data sets from simulations and fleet measurements?
2. How can we use such a prediction model for a predictive control in order to improve the efficiency of the battery thermal management?
3. How can we extend the method of prediction model and predictive control to industrial application?

The research questions are addressed by the development of a method for sequence prediction with prediction uncertainty and a corresponding predictive control in chapter 3. The method is applied to the battery thermal management in chapter 4. The developed models are compared with reference models. Furthermore, the method potentials, limitations, and differences to existing research are discussed in chapter 5.

# 3 Method of predictive control with quantile prediction models

In this chapter, a method for developing a predictive control using Machine Learning methods is described. The chapter first introduces the problem statement and the development pipeline (3.1). The pipeline steps are further explained in the following sections, namely the prediction model (3.2), predictive control (3.3), and an assessment for applicability of the models (3.4).

## 3.1 Problem statement and pipeline

Predictive control assesses possible predicted future states in order to determine the best control strategy at that point. Machine Learning methods are able to model complex relations in a data-driven way and can provide the required prediction. The development of a prediction model and predictive control is specified in the problem statement and pipeline. The problem statement describes the goal and requirements of the desired models as well as their application in a target system (also referred to as target domain). Thus, it is comparable to the step of business understanding in the Cross Industry Standard Process for Data Mining (CRISP-DM) [221]. The problem statement can be derived from the research questions (see 2.4), i.e. how data sets and foresight data can be used for a predictive control using Machine Learning methods. Fig. 3.1 shows the development pipeline which defines the structure of this chapter. Different problem statements can lead to different pipelines with additional or removed steps.

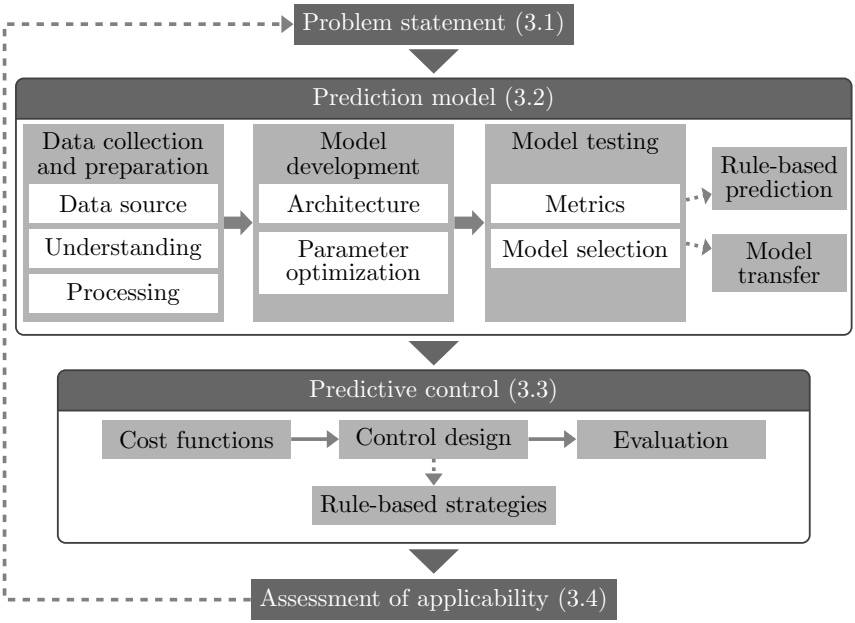


**Figure 3.1:** Development pipeline for the Machine Learning based predictive control with its steps (gray boxes). The dotted arrows indicate optional steps dependent on the goal and requirements from the problem statement.

The main steps of obtaining a prediction model include data collection and preparation, model development, and model testing. Data collection and preparation considers input data from different domains or sources. The properties of the data sets need to be investigated, which is called data understanding as in CRISP-DM [221]. Data processing is required for model development, which includes the architectural model design and optimization of parameters. The prediction model horizon and its sampling frequency are chosen according to the dynamics of the system behavior. Different models are developed and compared in order to select the best performing model, which is evaluated using according metrics in the step of model testing. Dependent on the application, a model transfer to the target domain might be needed. If applicable, simple rule-based prediction models can be trained.

The predictive control integrates the prediction model in the control design. The control design considers cost functions that are application-specific and determine the chosen control value in each control step. The controller performance is evaluated for tuning the control parameters and for comparison of its potentials and limitations with other controller types. If the predictive controller and its prediction model are too complex, thus require too much computation power and resources in the application, simpler rule-based strategies can be derived from the control. The problem statement specifies the allowed complexity of the controller, as well as the choice of the control variable ($u$ in 2.2.1), its value range, and the target variable ($y$ in 2.2.1) for the predictions. An assessment of applicability concludes the development pipeline with a comparison with the goals and requirements from the problem statement.

## 3.2 Prediction model

The steps of developing prediction models are further explained in the following sections, including data collection and preparation (3.2.1), model development (3.2.2), model testing (3.2.3) as well as the optional steps of rule-based prediction and model transfer (3.2.4). The goal is to find a model with good prediction performance fulfilling the application requirements. Additionally, the steps of model development and testing include reference models for comparison.

### 3.2.1 Data collection and preparation

The step of data collection and preparation is important for the performance of Machine Learning models trained and tested on that data. The prediction capability is limited to the information contained in the training data and to the quality of the training data. Therefore, data collection or generation need to be suited to the desired information for later model application. This includes the choice of data source, data sampling (features, step size, and horizons), and data filtering (of faulty data or imbalanced data). It is based on a deeper understanding

of the problem statement, the underlying physical dependencies, and the properties of the collected data.

#### 3.2.1.1 Data source and understanding

According to the problem statement, mainly two data sources are chosen as in [1, 3]: simulation data which contain information of novel control strategies and a large measurement data set from a fleet of target systems (in the following called fleet data). Simulation data can be generated using a custom Design of Experiments (DOE), with a focus on the effect of different control values $u$ on system behavior $y$. While simulations can be time-consuming and the modeled physical dependencies may be simplified, fleet data can cover real scenarios from the target system with all physical dependencies. The different data sets are combined, which can be described as cross-domain data fusion as explained in [20]. Sequenced data is sampled to the same step size for all data sets. The step size is chosen as small as necessary to describe the system dynamics but as large as possible to reduce the data set size. The resulting steps of equal size are called segments in this work for a better distinction. In case of fleet data, inaccuracies of the corresponding sampling parameter (e.g. measured distances) are analyzed concerning their effect on data sampling.

Feature selection and feature engineering can be supported by an analysis of the physical dependencies and by redundancy identification using Pearson correlation, as done in [1]. Checking the range and monotony of features helps to identify gaps or errors in the data set, which can be the result of faulty simulation models or sensors [1]. Including the control variable of the predictive control as input is important for later application in a predictive control. Further data analysis can target the needed system behavior such as plausible effects of varied control values in the simulation data in [1].

An analysis of varied control values in the collected data set investigates how different control values change the system behavior. This is required for prediction models trained on that data to learn plausible effects between control value and

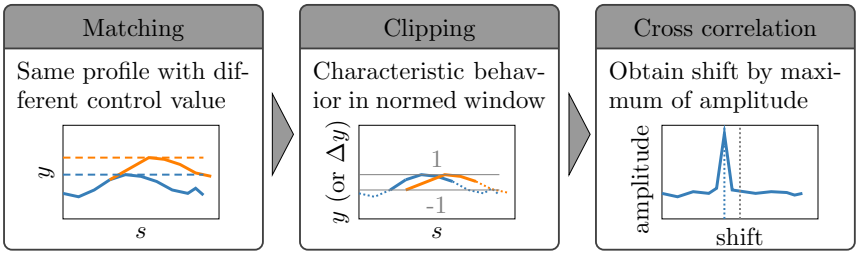


**Figure 3.2:** Processing steps for control value analysis using cross correlation.

system behavior. The effects can be analyzed by statistical analysis with visualizations of value distributions (e.g. box plots) and correlation analysis. The data set can be split into a subset for each control value or strategy. Then, distributions of key variables can be obtained for each subset, for instance the target variable or additional system quantities such as its energy consumption. A comparison of the subsets needs to show plausible differences in the distributions, for example a rising energy consumption for control values that increase the activity of energy consuming system states. Cross correlation can be used to analyze the effect of different control values on shifts of according quantities. For example, when comparing data of two thresholds with the same profile and boundary conditions, a shift in time can be identified by cross correlation of the same key quantity from the two data samples, as described in Fig. 3.2. Additionally, Pearson correlation can be used for an analysis of redundancies (correlation greater than 0.9 between two input features) and suitability of input features (correlation greater than 0.5 between an input feature and the target feature).

Sequence clustering is used to investigate characteristic system behavior and its occurrence in the data, as proposed by [3]. Sequence clustering can be a multivariate (with several input features) or univariate (only the target variable) clustering of extracted sequences, dependent on the scope of the analysis. Mini-batch k-means algorithm can be applied using sum of squared distances (the lower the better) and silhouette score (the higher the better) to determine the best suitable number of clusters. The clustering results are further investigated using Uniform Manifold Approximation and Projection (UMAP) for a reduction of the sequence

to two dimensions, followed by a two-dimensional visualization. A good clustering is characterized by no or only small overlap of the clusters. The typical cluster behavior of the sequence can be further analyzed by calculating the average sequence for each cluster, using Dynamic Time Warping based Barycentric Averaging (DBA). Together with the cluster occurrence, the average cluster curves provide an indicator about which system behavior occurs how often in the collected data set. Anomalies can be removed using outlier detection (e.g. with isolation forest), as done in [3]. Alternatively to clustering complete sequences of sequence data, the sequences can be resampled to single values for occurrence analysis. For instance, the difference between maximum and minimum value of a sequence can be considered to describe the system dynamics in a simplified way [1].

#### 3.2.1.2 Processing

Processing is required to obtain the data in the right shape for model training and testing. This includes the handling of faulty data and data gaps, for example by interpolation or using qualifier features [1]. The latter contain the information if the current value is obtained from the original data set or if it is a placeholder for a data gap. Further steps are joining additional data (e.g. weather data), data normalization, window slicing, data balancing, and data augmentation.

Window slicing considers different input and output horizons. Fig. 3.3 schematically shows the horizons considered in this work. A history input horizon $h_h$ contains recorded values $X_h$ from previous segments (equally-sized steps after sampling). A foresight input horizon $h_f$ includes values $X_f$ forecasted for the following segments, which can be obtained if future system states can be derived from external information, for example active guidance to a destination using a navigation system or a weather forecast. The prediction (output) horizon $h_p$ covers the target values $y$ which the model needs to predict. In this work, a fixed horizon size is used for each of the three horizons. At the beginning of a scenario (after the system started), the available number of previous segments is smaller than the history horizon. Close to the end of a scenario (the system is about to

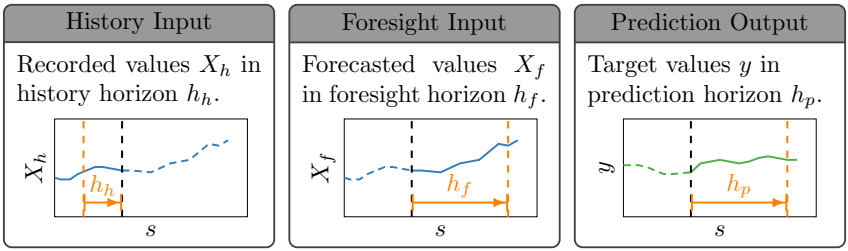


**Figure 3.3:** Window slicing and the resulting horizons for segments $s$.

be stopped), the number of foresight input segments and the number of predicted segments can be smaller than the according horizon sizes. For these cases, a qualifier feature is introduced which contains the information which segments are exceeding the start or end of a sequence profile, as proposed by [1]. Window slicing can be performed at each consecutive segment, or with a defined overlap between the resulting sequences. The overlap can be picked to achieve a balance between redundancy (large overlap, e.g. sequences of two consecutive segments only differ in one segment) and loss of information (small overlap, e.g. characteristic system behavior might not be covered by a sequence but is only partially included in neighboring sequences).

Data balancing targets a possibly imbalanced distribution of the system behavior. The distribution can be obtained by calculating the occurrence of the output variable range (difference between maximum and minimum within the prediction horizon) as in [1] or from a sequence clustering as previously described in this section and done by [3]. The data set is then balanced by reducing the amount of data of the majority classes, increasing the amount of data of the minority classes, or a combination of both. In [3], data balancing is conducted for the fleet data by sequence clustering, followed by random undersampling or random oversampling of the resulting classes. On the one hand, undersampling can result in information loss, dependent on the variance of data within each cluster. On the other hand, oversampling does not increase model knowledge about the respective class. Instead, data balancing helps the model to not (completely) neglect minority classes during training and evaluation.

Data imbalance can also exist between two different data sets, such as a smaller simulation data set compared with a larger fleet data set in [3]. In this case, data augmentation is proposed to increase the data set size by synthesizing new data based on the original data set. Besides reducing the data imbalance, data augmentation can bring additional variety into the simulation data set which reduces possible overfitting. The simulation data set is augmented using the newly introduced method of Intersection-Based Assembly (IBA) to increase its share compared to the fleet data [3]. The main steps of this novel method are shown in Fig. 3.4. In the original data set, segments of two different sequences $A$ and $B$ are matched according to matching criteria for a given feature set. For instance, the corresponding values are required to be equal within a matching tolerance. Two new sequences $A_1B_2$ and $B_1A_2$ are composed by combining the two parts (1 and 2) of the two original sequences ($A$ and $B$) around the matched segment. The new sequences are cut such that the intersection is contained in one of the horizons (from Fig. 3.3) in order to avoid data duplication. The larger the chosen matching tolerance, the more data can be generated, but the more inaccurate the transition at the intersection might be.

After the data preparation steps are complete, a training, validation, and testing data set are available for building, training, and evaluation of prediction models. Further processing depends on the model type and architecture and is covered in the following section.

### 3.2.2 Model development

The step of model development includes the selection of model type, design of model architecture, and hyperparameter optimization based on the validation data. In this work, the focus is on Deep Learning (DL) based Quantile Neural Networks (Q*NN). Single-step regression models are built for reference and comparison. Therefore, this section is divided into the development of Q*NN models (3.2.2.1) and reference models (3.2.2.2).

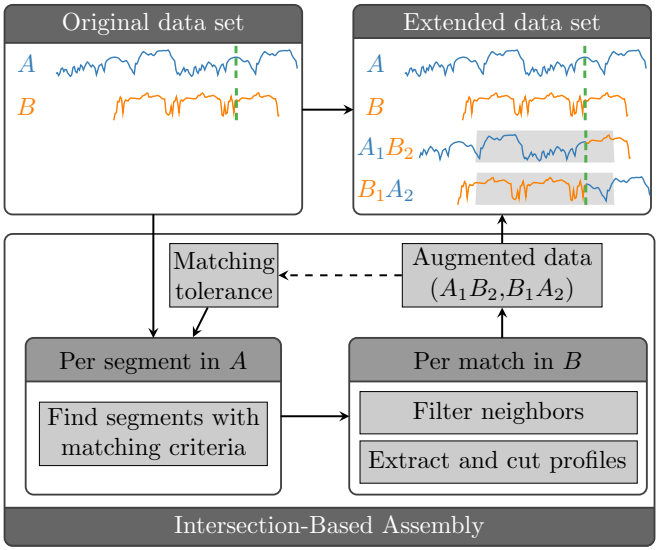


**Figure 3.4:** Intersection-based assembly, based on [3]. For example, newly composed sequences $A_1B_2$ and $B_1A_2$ are derived from sequence $A$ ("Worldwide harmonized Light vehicles Test Cycle" (WLTC) [222]) and sequence $B$ ("US06 cycle" [223]). The new sequences are cut to the gray area.

#### 3.2.2.1 Quantile Neural Networks (Q*NN)

The proposed prediction model is based on Neural Networks using convolutional or recurrent layers to handle the characteristics of time-series or sequences, as described in [1, 3]. It handles sequences from the two input horizons history and foresight (see subsection 3.2.1) and predicts sequences of the target variable. The model is designed to additionally provide information about the prediction uncertainty. Since the underlying distributions cannot be assumed to follow a specific distribution (e.g. Gaussian), the uncertainty is provided by a prediction of quantile sequences. One quantile sequence consists of one defined quantile for each consecutive segment in a sequence, for example the 0.5 quantile sequence is composed of the 0.5 quantiles of each segment. The model architecture aims at a decreasing size from inputs to outputs for faster training and lower computational

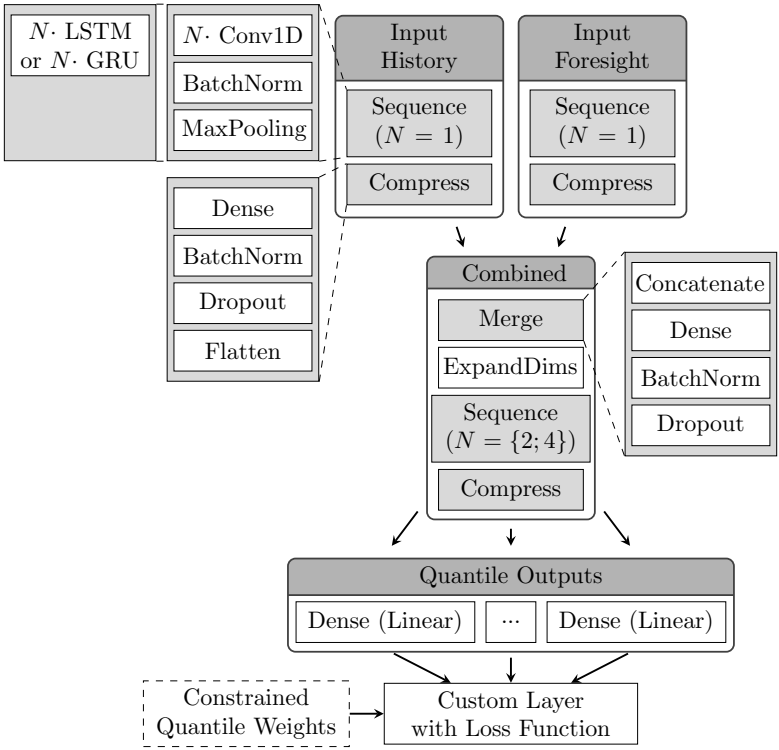


**Figure 3.5:** Q*NN model architecture based on [1].

load (see 2.3.2). The proposed model characteristics are considered in the Neural Network architecture, which is shown in Fig. 3.5 and follows the design choices from [1, 3].

In the first layers, the different input horizons are handled individually by two input channels, named input history and input foresight. Each of them consists of a group (called "Sequence") that handles the sequence characteristics and a group (called "Compress") that compresses the intermediate outputs for the following channel merge. The "Sequence" group contains either Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), or one-dimensional convolutional ("Conv1D") layers (as described in 2.3.2). The number of the recurrent or

convolutional layers is given by $N$. In this work, $N$ is set to one for all models for the sequence block in the input channels. In the case of convolutional layers, a layer for batch normalization ("BatchNorm") and a layer with maximum pooling ("MaxPooling") are added afterwards for regularization (see 2.3.2). The "Compress" group contains a dense layer, another "BatchNorm" layer, dropout, and a flatten layer. The additional "BatchNorm" layer and dropout layer are used for further regularization and are ordered as suggested by [156, 157, 224]. The "Compress" group reduces the number of trainable parameters to enable a following channel merge with comparable number of parameters per layer.

The output of both channels is merged in a combined channel using a concatenate layer. An "ExpandDims" layer expands the merged data by one dimension (with size one) for further sequence processing. Later hyperparameter optimization (HPO) considers an architecture with $N = 2$ and one with $N = 4$ (referred to as deeper network) for the sequence block in the combined channel. After another compression, the output is defined by one linear dense layer per quantile. The architecture is concluded with a custom layer that calculates the loss function $L_{total}$ using constrained quantile weights. Model training and HPO aim at minimizing the loss function. $L_{total}$ is calculated according to (3.1) by a weighted sum (with weight $w_{mse}$) of the Mean Squared Error (MSE) $L_{mse}$ (3.2) and constrained weighted quantile loss $L_{cwq}$ (3.3). The equations of $L_{total}$ and its components are based on [1], extending the loss function described in [158].

$$L_{total} = w_{mse} \cdot L_{mse} + (1 - w_{mse}) \cdot L_{cwq} \tag{3.1}$$

The MSE describes the difference between the true values $y$ and the median prediction (0.5 quantile) $\hat{y}_{q_{0.5}}$ over the prediction horizon with $n_p$ segments. $L_{cwq}$ sums up the pinball loss $L_{pb}$ with trainable but constrained weights $\mu_j$ per quantile over all $n_q$ quantiles. The pinball loss $L_{pb}$ (3.4) punishes the difference between true value and quantile prediction $\hat{y}_{q_j}$ differently, dependent on the quantile $q_j$, purposely allowing quantile-related over- or underestimation of the true value.

The quantile weights $\mu_j$ are constrained by a softmax function, symmetry around the median, fixed inital values, and fixed order, as described in [1].

$$L_{mse} = \frac{1}{n_p} \cdot \sum_{i=1}^{n_p} (y_i - \hat{y}_{i,q_{0.5}})^2 \tag{3.2}$$

$$L_{cwq} = \frac{1}{n_p(n_q+1)} \sum_{j=0}^{n_q} \sum_{i=1}^{n_p} \mu_j \cdot L_{pb}(i,j) \tag{3.3}$$

$$L_{pb}(i,j) = \max[(q_j - 1)(y_i - \hat{y}_{i,q_j}), q_j(y_i - \hat{y}_{i,q_j})] \tag{3.4}$$

The number of nodes per layer is decreasing from input to output layers. Thus, a higher number of nodes can capture complex patterns of the input sequences in the first layers and a reduced number of nodes for layers closer to the output reduces model size and improves the computation time [153]. A variable base number of nodes is used as hyperparameter, which is multiplied by a fixed factor for each layer ensuring the decreasing relation. The fixed factors and further layer parameters are the same than in [1] and listed in A.3.[1]

The Q*NN hyperparameters can be divided into two groups, related to model architecture or to model training. Architectural hyperparameters are the base number of layer nodes, dropout rate, loss weight $w_{mse}$, and the number of sequence processing layers $N$ in the combined channel. Training parameters include the learning rate and mini-batch size. The hyperparameters can be obtained by random grid search as in [1] or by an efficient Bayesian optimization as in [3]. For the latter, the first trials are run using a share of the training data to quickly obtain first indication of proper and non-proper hyperparameters and their combination. The share is increased over the number of trials, such that the last trials are run

[1] For the deeper network with two additional convolutional layers, a kernel size of 9 and 5 is chosen. For the LSTM and GRU layers, the number of units equals the base number of nodes.

with all training data for fine-tuning. The share $\rho(\tau)$ can be described by a formula such as (3.5) from [3], with trial number $\tau$ and the total number of trials $n_\tau$.

$$\rho(\tau) = \left\lceil 100 \cdot (0.1 + \frac{0.9}{1 + \exp(-12 * (\frac{\tau}{n_\tau} - 0.4))}) \right\rceil / 100 \tag{3.5}$$

For each trial of hyperparameters, at least two models are trained from scratch to consider possibly bad performance when training reaches and stays at a local minimum. For each model, the minimum validation loss is obtained over all training epochs. The trial is then evaluated using the average of the minimum validation losses. Three new models are trained using the hyperparameters of the trial with lowest loss and all training data. The model with lowest validation loss is used for testing and for a comparison with reference models.

#### 3.2.2.2 Reference models

Reference models are developed using standard Machine Learning methods to provide a baseline for the evaluation of the Q*NN models. Regression and quantile regression models are built to predict a single-step in the prediction horizon in contrast to a sequence. Models with different horizons can be combined to obtain multi-step predictions. While (deep) Neural Networks can handle large numbers of input features, the dimension and number of input features need to be reduced for classic regression models. This can be achieved by additional feature processing, such as the calculation of average, standard deviation, maximum, or minimum of the feature over the horizon. Moreover, this processing can be applied to the differences of subsequent values of a feature (comparable with the first derivative).

Classic regression models can be built using automated Machine Learning tools (e.g. the Pycaret Python package [225]). This enables fast training and comparison of different regression methods, such as Random Forest (RF), Light Gradient Boosting Machine (LGBM), and Extra Trees Regressor (ETR). For each considered horizon, the model with lowest validation loss is picked as reference for

the following testing and comparison with the Q*NN models. The validation loss can be based on various regression metrics such as the Root Mean Squared Error (RMSE) or Coefficient of Determination ($R^2$). In contrast to classic regression methods, quantile regression can provide single-step predictions for different quantiles. For example, Quantile Random Forest (QRF) or Quantile Extra Trees Regressor (QETR) are corresponding methods offered by the Python package scikit-garden [226]. Another considered reference model is a Shallow Decision Tree (SDT) [3]. It is also a one-step regression model built for different horizons. Its hyperparameters of maximum depth and maximum number of leaves are optimized using grid search. Their ranges are limited according to the requirements of maximum allowed complexity for rule-based prediction in 3.2.4. The hyperparameter set with lowest validation loss is proposed as best suitable SDT model.

### 3.2.3 Model testing

The performance of the trained Q*NN and reference models is evaluated and compared based on the test data. Different metrics and diagrams are used for an evaluation of the point-prediction performance and the quantile-related performance, which are explained in the following sections. Further analysis includes prediction examples and feature importance derived from model weights. The goal of this step is to identify potentials and limitations of the different model types and to choose the best performing model for a predictive control.

#### 3.2.3.1 Point-prediction performance

Point-prediction performance refers to the evaluation of only one predicted segment or sequence compared with the true values, in contrast to an evaluation of several quantile sequences and their quantile-related properties. Regression models only predict one point, such that they only offer one predicted sequence (when

models with different horizons are combined). For quantile-based models, point-prediction performance is evaluated for the 0.5 quantile (median) prediction. The evaluation of point-prediction performance is based on classic regression metrics and visualizations. These metrics include the MSE, RMSE, $R^2$, and Mean Absolute Error (MAE).

The $R^2$ is calculated according to (3.6), with the true values $y$, their average $\bar{y}$, the predicted value $\hat{y}_{q_j}$ per segment, $n_p$ segments in the prediction horizon, $n_o$ observations in the test data. The numerator is the sum of the squared prediction errors ($y_i - \hat{y}_{q_j}$). The denominator contains the sum of squares of the deviation of the data from their mean value and a constant $\varepsilon$ to avoid possible division by zero. The values of $R^2$ are lower or equal than 1, with 1 being the best, and possible negative values in non-linear regression [227, 228, 229, 230]. Negative values occur if the prediction errors (numerator) are significantly larger than the deviation of the data from the average (denominator). Thus, a model with $R^2$ smaller than zero performs worse than a model which always predicts the average value, independent of the model inputs. Besides zero, another $R^2$ significance level can be defined to evaluate if model accuracy is not good enough for the given application, even if it is better than predicting the average value. This threshold depends on the application, for example in social sciences it can be 0.51 or 0.1 dependent on further statistical properties [230]. In this work, it is set to 0.51, but the choice of the right threshold is recommended for future research.

$$R^2(q_j) = 1 - \frac{\sum_{o=1}^{n_o} \sum_{i=1}^{n_p} (y_i - \hat{y}_{q_j})^2}{\sum_{o=1}^{n_o} \sum_{i=1}^{n_p} (y_i - \bar{y})^2 + \varepsilon}, \text{ with } \varepsilon = 10^{-7} \tag{3.6}$$

The smoothness $\sigma_{q_j}$ of the predicted quantile $q_j$ is an additional metric used in [1]. It is defined by (3.7) with the standard deviation $\sigma$ of the first derivative of a predicted sequence, approximated by the difference of the predicted value $\hat{y}_{q_j}$ between two consecutive segments $s_1$ and $s_2$, divided by the difference of the segments (which is a fixed value, according to the definition of the segment, e.g. distance or time), with $n_p$ segments in the prediction horizon. Lower values

indicate smoother sequences, which is desired for systems with normal dynamics (contrary to systems with high frequencies of large changes of the system states).

$$Smoothness(q_j) = \sigma_{q_j} = \sqrt{\frac{\sum_{i=1}^{n_p}(x_i - \bar{x})^2}{n_p}}, \text{ with } x = \frac{\hat{y}_{q_j}(s_2) - \hat{y}_{q_j}(s_1)}{s_2 - s_1} \tag{3.7}$$

Visualizations include regression plots and error histograms. All metrics and visualizations can be investigated for different horizons and for different values of the control variable used by the predictive control. Regression plots show the predicted values over the true (target) values. Thus, the prediction performance can be analyzed dependent on the true value, for example to specifically check prediction accuracy for large true values (i.e. dynamic system behavior). Error histograms give insights into the occurrence of (larger) errors and over- or underestimation of the true values. Additional metrics can be derived to describe the maximum error for a given data share, for example a maximum RMSE for 90 % of the test data. These metrics and visualizations can be beneficial for the evaluation of how well the model serves the given problem statement.

#### 3.2.3.2 Quantile-related performance

Quantile-related performance is evaluated with metrics and visualizations of quantile predictions. The metrics consider the occurrence of true values within quantiles ($p(q_j)$) and quantile intervals ($p(q_j, 1-q_j)$) as included in [1]. This concept is also known as calibration [189]. For instance, a predicted 0.9 quantile is accurate according to its definition if 90 % of the true values are below the predicted value. Quantile intervals are predicted well if, for example, 80 % of the true values are between the predicted 0.9 and 0.1 quantile. An evaluation can focus on the absolute differences between the actual occurrence and the definition given by the quantiles (the lower the better), indicated by the subscript $d, abs$ in this work. The advantage of both metrics is an intuitive understanding. However, a comprehensive evaluation of the quantile-related performance additionally requires the following metrics.

An important quantile property is the occurrence of unwanted quantile crossing. This can be addressed by the metric Crossover Rate Score (CORS) as defined in [158]. The lower the value, the less quantile crossing occurs. The Winkler Score (WS) is used by [158] to describe the occurrence of true values within quantile intervals, punishing differences for outer intervals stronger than for inner intervals. Another metric from [158] is called sharpness ($\psi$), which calculates the width of the quantile intervals. A bigger width can cover more true values, but also reduces the practical value if the interval is much wider than the underlying range of the target variable. For sequenced predictions, the average or maximum sharpness can be considered as metrics, with smaller values indicating better performance.

Visualizations of quantile-related performance can include a heat map of the metrics, emphasizing which model performs best for each metric. A histogram of the share of true values within quantile intervals can provide further insights into the reliability of the predicted uncertainties. Further possible visualizations are presented in [231].

#### 3.2.3.3 Exemplary analysis

The described point-prediction and quantile-related metrics and visualizations enable an evaluation of the prediction performance over the whole test data set. An additional exemplary analysis targets a more tangible understanding of the provided information and limitation for specific scenarios [1, 3]. At first, relevant scenarios need to be defined and identified in the test data set. For example, one data point from each cluster (see subsection 3.2.1) can be picked. If the data set contains scenarios with different values of the control variable but same boundary and initial conditions, an exemplary analysis can be done to compare the influence of the control variable on the prediction. For each scenario, the predicted quantiles and the true values are plotted over the prediction horizon. Sections of changing system states can be marked and characteristic behavior can be connected to the observed metrics. The sharpness of the quantile sequences can be compared

for different predictions, which shows if the prediction uncertainty is adapted to different scenarios.

#### 3.2.3.4 Feature importance

Further model analysis can include feature importance. This helps to identify relevant features, for better understanding of the model behavior and the input data. The identification of less relevant features provides information about which features could be discarded in a later iteration of building a prediction model. This can be useful in case the model complexity needs to be reduced due to limited Random Access Memory (RAM) and disk storage of the target system. High feature importance means the prediction model relies strongly on the corresponding feature, such that the performance will significantly decrease if the according input is inaccurate.

For the reference (quantile) regression models, the impurity-based Gini feature importance can be calculated, which provides good results if all input features are of the same type (in this work: numerical) [232]. Alternatively, the permutation feature importance is calculated by comparison of the prediction performance on the test data after shuffling the according feature within the test data set. This method is also applied for the Q*NN models, individually for the history input features and the foresight input features. The according feature data are shuffled between the different test data, but not for the horizon. For the reference models, feature importance can be analyzed for each model of different horizon or as average over all horizon models.

### 3.2.4 Rule-based prediction and model transfer

An application of prediction models raises various requirements and limitations dependent on the target system. For example, a target system might require fast models (e.g. with real-time capability), models with low RAM and disk storage consumption, or explainable ML models. In these cases Neural Networks or large

regression models might not be suitable. Furthermore, only few data are available during system development, when the system is not yet released to customers or other users. For both aspects, a possible solution is proposed in this section.

Decision Trees are models which can be limited in size and offer explicit, interpretable rules. In this work, a Shallow Decision Tree (SDT) model is built with limited depth and maximum number of leaves for application in systems with small RAM and disk storage. It is developed according to the reference models in 3.2.2.2, with the same data set and data processing for training, validation and testing. The SDT can be visualized with its nodes and leaves. The conditional rules of the tree can be translated into a function with a look-up table which simplifies the integration into the target system (e.g. an Electronic Control Unit (ECU)). A comparison of the number of parameters between the SDT and the Q*NN models gives insights into the required computational resources.

In case of limited available data, Transfer Learning (TL) can be used to train a model with few data of the target system, based on a model trained on a large data set of a similar, existing system. It is assumed that the underlying (physical) relations remain the same and the model weights are fine-tuned to the new target system. In this work, the following methods of Transfer Learning are investigated to assess their applicability for the developed Q*NN:

- No adjustments, i.e. continue model fitting with the new data.
- Adjusted learning rate, i.e. a lower learning rate for the first layers and a higher learning rate for the last layers.
- Resetting layer weights of the last layers.
- Freezing layers except for the last layers.
- Combinations of the above methods.

Each method takes the best performing Q*NN as base model and is applied three times to cope with the non-deterministic training process. The model with lowest validation loss is picked for comparison with a reference model. The reference model is trained from scratch, only with the (small) data set from the target system.

In this work, large data sets from two similar systems are available. This allows to investigate Transfer Learning with both a large data set and reduced data sets (by random subsampling) of the target system. Reduced data sets with different sizes are considered to investigate the effect of data set size on the prediction performance for the according application. Such a comparison helps to identify the required amount of data that meets the required accuracy. The point-prediction and quantile-related metrics from 3.2.3 are applied for the evaluation.

## 3.3 Predictive control

In this chapter, a novel predictive control is proposed using the prediction model (from 3.2). The steps consist of the control design (3.3.1), with cost functions (3.3.2) to determine the control value, and an evaluation of its performance (3.3.3). Additionally, rule-based strategies are derived from the predictive control as a novel approach to cope with computational requirements (3.3.4).

### 3.3.1 Control design

The proposed control design varies the control value based on the Q*NN predictions in order to minimize the cost function. The predictive control can be integrated into a simulation framework as shown in Fig. 3.6 and proposed by [2]. In application, the simulation model of the system model block is replaced by the target system. The simulation framework supports the design process of the predictive control and allows an early evaluation before testing it in a real environment. The control design and simulation framework are further explained in the following.

A simulation of the predictive control is conducted until the (integer) simulation step $S$ reaches the end ($S_{end}$) of a given profile. Data processing is needed during the whole simulation to obtain the inputs for the predictive control and to log the simulation results according to the desired output. In the first simulation

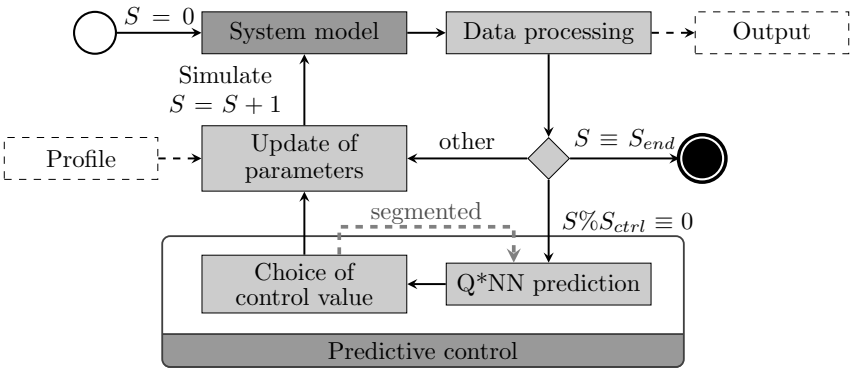


**Figure 3.6:** Activity diagram of a simulation framework using the proposed predictive control, based on [2].

step ($S = 0$), the model is initialized, including the choice of the initial control value by the predictive control. While the profile is simulated, each time the number of simulation steps reaches a parameter value $S_{ctrl}$, the predictive control updates the control value. In this work, $S_{ctrl}$ is called control step size. The value for $S_{ctrl}$ is defined according to the expected dynamics of the system behavior and computational efforts. On the one hand, low values are desired such that the predictive control can quickly react to changing foresight input data and boundary conditions. On the other hand, high values reduce the requirements towards real-time capability. In each predictive control step, the Q*NN is used to predict quantile sequences of the target variable for cost calculation of each control value. The control value with lowest total costs is chosen until the next predictive control step is reached. The number of possible control values needs to be limited according to the allowed computation time. The ratio between $S_{ctrl}$ and the Q*NN prediction horizon (i.e. length of the predicted quantile sequences) is not fixed. However, the predictive control cannot evaluate the effect of the control value on the system behavior after the prediction horizon, such that $S_{ctrl}$ should not be larger than the prediction horizon. Thus, the Q*NN prediction horizon needs to be large enough to cover $S_{ctrl}$ and, furthermore, to provide enough information for the predictive control.

Before the cost functions are introduced in 3.3.2, an extended control design is presented as alternative with more degrees of freedom for the control values. It includes a segmented prediction as indicated by a gray, dashed arrow in Fig. 3.6. In this case, the selection of possible history input features of the Q*NN prediction model is limited to the available foresight input features (including the control variable) and the prediction variable. The segmented prediction is similar to a recursive prediction: the predicted output for a given control value is used as input for a prediction shifted by an adjustable number of segments (shift size). The shift size should be a multiple of the control step size $S_{ctrl}$ to consider the effect of possible changes of the control value in the prediction. New predictions are conducted for all quantile predictions of each control value and the according shift. This step can be repeated for any number of shifts within the foresight input horizon. However, each additional shift increases the number of predictions and thus the needed computation time. The number of predictions $n_p$ for $n_q$ quantile predictions of $n_{cv}$ control values and $n_{shift}$ shifts can be calculated by (3.8).

$$n_p = n_{cv} \cdot (n_q \cdot n_{cv})^{n_{shift}} \tag{3.8}$$

Fig. 3.7 depicts an example of segmented predictions of the target variable $y$ with one shift. The prediction model considers three quantiles and two control values $u_1$ and $u_2$. The first predictions without a shift are only included for control value $u_1$ and cut off after the shift horizon $h_{shift}$ (shift 0). These predictions are based on the history data of $y$ and available foresight data. Shifted predictions after the shift horizon $h_{shift}$ are shown for the upper quantile prediction of control value $u_1$ of shift 0 (blue-green dashed line) for both control values $u_1$ and $u_2$ for shift 1. They are cut off after the total prediction horizon $h_p$. The predictions of shift 1 use the corresponding quantile sequence from shift 0 (in this case the upper quantile, blue-green dashed line) and foresight data from shift 0 as history input, instead of recorded history data. The segmented predictive control calculates costs for each of the predicted sequences over the prediction horizon, including all combinations of quantiles for all shifts and all control values. The costs are averaged over all quantile combinations for each control value combination (in

this example $u_1u_1$, $u_1u_2$, $u_2u_1$, and $u_2u_2$). The control value combination with lowest costs is chosen for the current control step.

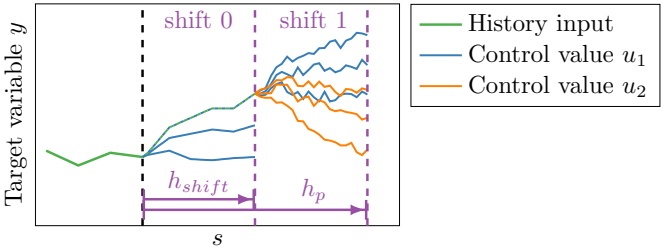


**Figure 3.7:** Example of segmented predictions with one shift, three quantiles, and two control values $u_1$ and $u_2$ over segments $s$ for a shift horizon $h_{shift}$ and prediction horizon $h_p$. Predictions for $u_2$ at shift 0 are not shown to keep the diagram easier to read. The shown predictions of shift 1 use the previous prediction of the upper quantile of shift 0 with $u_1$ (blue-green dashed line) as history input.

### 3.3.2 Cost functions

The total costs are calculated as weighted sum of individual cost functions. The cost functions represent different system properties (e.g. energy consumption, lifetime, power availability) that can lead to contradicting optimization goals for the controller. An adjustment of the weights allows fine-tuning the controller according to the required balance between the contradicting goals, which is further explained in 3.3.3.

Each considered property is represented by an according cost function $c$ with quantile predictions $\hat{y}_q$ as input. The cost functions can be distinguished in the way they are calculated:

- Direct computation $c(\hat{y}_q)$ for all predicted segments.
- Direct computation $c_{end}(\hat{y}_q)$ for the last segment of a profile (similar to terminal constraints in [80, p. 54-55]).

- Indirect computation $c(g(\hat{y}_q))$, where $c$ describes costs that need an estimation $g$ of another future system state that depends on $\hat{y}_q$.

The cost functions can be derived by empirical or mathematical relations between the predicted quantity and the system property. Additionally, foresight input features can be included in the cost functions. The cost functions are normalized and centered to a range between 0 and 1 according to the expected minimum and maximum values. They are calculated for each quantile sequence provided by the prediction model. For each control value, the average of each cost part over all quantiles is calculated. The total costs are calculated as weighted sum and the control value with lowest total costs is chosen.

### 3.3.3 Evaluation

The predictive control can be evaluated based on simulations (in this work) or measurements of test scenarios. The scenarios are defined in a Design of Experiments (DOE), for example by time-profiles of input data and boundary conditions. The evaluation covers the adaptability, tunability, and robustness of the predictive control as proposed in [2] and shown in Fig. 3.8. Further analysis assesses the resulting computation time.

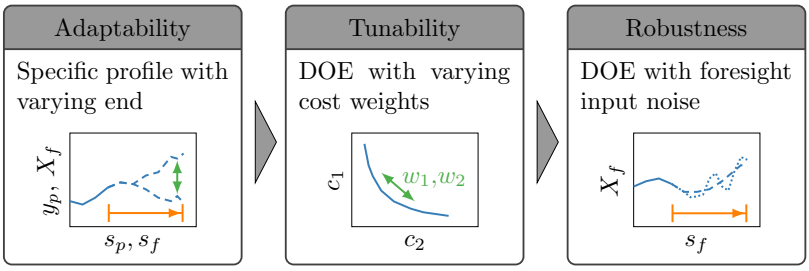


**Figure 3.8:** Evaluation categories of the predictive control, based on [2].

#### 3.3.3.1 Adaptability

Adaptability describes the controller's ability to adapt its strategies according to different foresight input data $X_f$ of foresight segments $s_f$ and thus different predictions $\hat{y}_q$ for the predicted segments $s_p$. An analysis requires at least two simulations or measurements with the same initial and boundary conditions, and a profile that is the same in the beginning, but differs at the end. For the varying end, a significant difference in $X_f$ is required, such that the predictive control is expected to adapt the choice of its control values accordingly, before reaching the changed end. The analysis includes the calculated costs for each control value and Q*NN predictions of control steps that lead to a change in the chosen control value and related system state. In this work, the focus is on an exemplary analysis of two simulations with varying end. However, the adaptability can be evaluated systematically when a large set of according simulations or measurements is available that fulfill the evaluation requirements. In case the Q*NN predictive control shows plausible adaption of the control values to different foresight input data, its tunability can be evaluated next.

#### 3.3.3.2 Tunability

Tunability refers to the ability to fine-tune the cost weights in order to achieve a desired balance of contradicting controller goals. A tunability evaluation requires simulations (or measurements) of a DOE with profiles that are repeated with the same initial and boundary conditions but varied cost weights.

In the following, simulations with variation of two weights $w_1$ and $w_2$ for the respective cost functions $c_1$ and $c_2$ are considered. The resulting costs of $c_1$ and $c_2$ are visualized as Pareto front (see Fig. 3.8), normalized by the maximum occurring values of each cost part. The Pareto front represents the Pareto optimal performance of the respective controller since it represents the points at which any reduction of one cost part results in an increase of the other cost part [233]. The weights with lowest total costs can be identified as the weights with resulting costs closest to the theoretical optimum of zero costs (the origin in the diagram).

Furthermore, a characteristic property of the Pareto front is the point at which the slope is closest to -1. At this point, the reduction in one cost part equals to the increase in another cost part, such that the point describes a balance of the two. The Pareto fronts of different controllers can be used to compare their performance. The better performing controller is the one with a Pareto front closer to zero costs. The comparison can be continued by identifying a weight set which results in the same value for cost function $c_1$ for both controllers. The difference in the corresponding value for cost function $c_2$ describes the improvement by the better controller with unchanged value for the other cost part. Furthermore, related system quantities (e.g. total energy consumption) can be compared both as average over all scenarios and individually for each scenario. The latter indicates if the predictive control performs well over all scenarios or if the improvements depend on the scenario. Additionally, system states resulting from the control value can be evaluated considering their starting point, duration, and number of events. Such an analysis provides further insights into the control behavior and rules of thumb for good strategies.

#### 3.3.3.3 Robustness

The proposed predictive control uses a prediction model that considers foresight input data $X_f$ from foresight segments $s_f$ for its predictions. Since the foresight input data are derived from assumptions about future system states, they can be noisy and inaccurate. Thus, the effect of input noise on the control performance is evaluated with simulations, which is called robustness in this work. Noise on history input data (e.g. due to inaccurate sensor measurements) is not focus of this work, but recommended for further research.

Two categories of foresight input noise are considered: underestimation and overestimation of the dynamics of $X_f$. Underestimation is analyzed by applying a moving average to the foresight input data before they are used by the prediction model. Overestimation is considered by adding Gaussian noise to the foresight input data. The window size of the moving average as well as the mean value and standard deviation of the Gaussian noise are set corresponding to the expected

noise of the individual input features. The predictive control is then simulated with the DOE and weights from the tunability evaluation. Robustness is achieved when the resulting costs are in a similar range than for the simulations without foresight input noise. Additionally, the influence of the control step size $S_{ctrl}$ (see 3.3.1) is analyzed by comparing the resulting costs for different step sizes.

#### 3.3.3.4 Computation time

For applications with highly dynamic systems, low computation times need to be achieved such that the predictive control can adapt the control values accordingly. The computation times of the predictive control are assessed with (3.9) and (3.10). The average and maximum times for prediction (including required data processing) $t_p$ and for the time of control value choice $t_{ctrl}$ are calculated over all control steps $n_{ctrl}$ and scenarios $n_{sc}$. For the proposed control, higher computational efforts are caused by the prediction model than by the following choice of the control value. If the predictive control takes longer than the system needs to reach the next control step $S_{ctrl}$, it is not able to operate in real-time. In this case, the previous predictions and resulting control value choice might be outdated which leads to bad performance. This limits the applicability of the predictive control variant with several time-consuming steps of segmented predictions (see 3.3.1).

$$t_{avg} = \frac{1}{n_{sc} + n_{ctrl}} \sum_{i=1}^{n_{sc}} \sum_{j=1}^{n_{ctrl}} (t_{p,i,j} + t_{ctrl,i,j}) \tag{3.9}$$

$$t_{max} = \max_{i \in n_{sc}} (\max_{j \in n_{ctrl}} (t_{p,i,j})) + \max_{i \in n_{sc}} (\max_{j \in n_{ctrl}} (t_{ctrl,i,j})) \tag{3.10}$$

### 3.3.4 Rule-based strategies

The novel predictive control uses Q*NN prediction models, which require sufficient disk storage and Random Access Memory (RAM) of the target system. Alternatively, rule-based strategies can be derived from simulations or measurements of the predictive control. The derivation is conducted in a back-end, and

the resulting rules can be integrated in the target system similar to look-up tables. The idea is to develop a simple rule-based model that describes characteristic patterns of the Q*NN predictive control.

This work proposes a method for the development of rule-based models from simulations of the Q*NN predictive control. A Design of Experiments (DOE) can be used to generate the scenarios. The simulation data are reprocessed as sequences with fixed horizons. The considered input and output features are transformed into single values (e.g. mean, variance, current value of the sequence), as shown in Fig. 3.9. A Shallow Decision Tree (SDT) classifier is trained for different horizons at the same control step (e.g. M1 to M3 in Fig. 3.9), such that both long-term trends and dynamic short-term changes can be considered. As in 3.2.4, each decision tree is limited in depth and maximum number of leaves to keep its complexity low. Each SDT classifier predicts the desired control value, based on the simulation results of the Q*NN predictive control. The training data are balanced if the occurrence of control values or system states is imbalanced. A voting can be performed on the classification results to use the SDT classifiers with different horizons as ensemble model with an adaptable voting threshold. In case the end of a profile is close, only the SDT classifiers are considered whose horizons end before the end of the profile. Thus, the SDT classifier ensemble can be used even when its large horizons are exceeding the profile.

An evaluation of the SDT ensemble can be done by analysis of the confusion matrix which provides information about the number of correct and false classifications. The F1-score and the accuracy can be used as metrics, based on the confusion matrix. Additionally, the distributions of predicted classes and true classes can be compared. Even if wrong classifications occur, the model performance might be acceptable if it results in a similar class distribution than for the Q*NN predictive control. The results can be obtained for different horizons and different voting thresholds. Further analysis can include feature importance of the SDT models, which indicates which input parameters have the strongest effect on the choice of the control value. The resulting costs of the SDT models for different horizons and the SDT ensemble can be visualized in the Pareto diagram

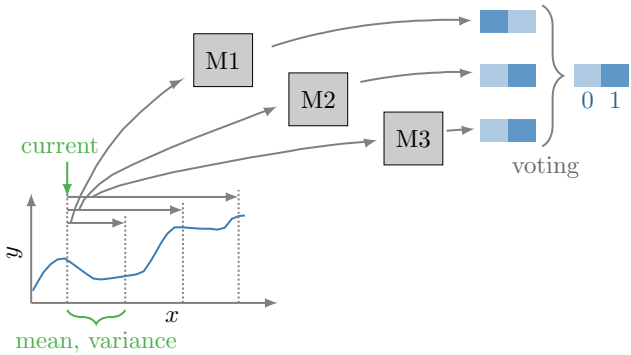


**Figure 3.9:** Ensemble Shallow Decision Tree (SDT) Classifier for a rule-based predictive control. For example, processed features (current value, mean, variance) of three horizons are input of three SDTs (M1, M2, M3) at the same control step. A voting of the three predictions obtains the final result.

of the tunability analysis from 3.3.3.2. Tunability of the SDT ensemble can be analyzed by adapting the voting threshold.

The trained SDT classifier ensemble can be translated into if-else-statements with the according thresholds and the voting threshold of the SDT ensemble. Thus, the rule-based strategies can be integrated into the target system with low computational requirements. The number of required parameters can be obtained by the number of if-conditions and tree leaves (which need to contain the corresponding output value). A comparison of the number of parameters, number of input features, and computation times with the Q*NN models shows the improvement in needed computational resources.

## 3.4 Assessment of applicability

The key goal of the developed predictive control methods is to improve the overall system behavior, for example by improving its energy efficiency. Furthermore, an assessment of the predictive control for real-world application includes requirements towards computational resources and system architecture. In the following,

different possibilities for an integration of the predictive control are suggested. The different variants are shown in Fig. 3.10, sorted according to their complexity levels. The highest adaptability, tunability, and accuracy can be achieved by using the Q*NN prediction model in the predictive control with custom cost functions. Adaptability refers to the direct consideration of all information of the foresight input data as well as the most individual consideration of different scenarios. High tunability provides various possibilities of parameter tuning (e.g. weights) of the prediction model and the predictive control (e.g. its cost functions). On the one hand, the most complex of the considered prediction models presumably result in the highest prediction accuracy. On the other hand, complex models require more computational resources which limits their applicability.

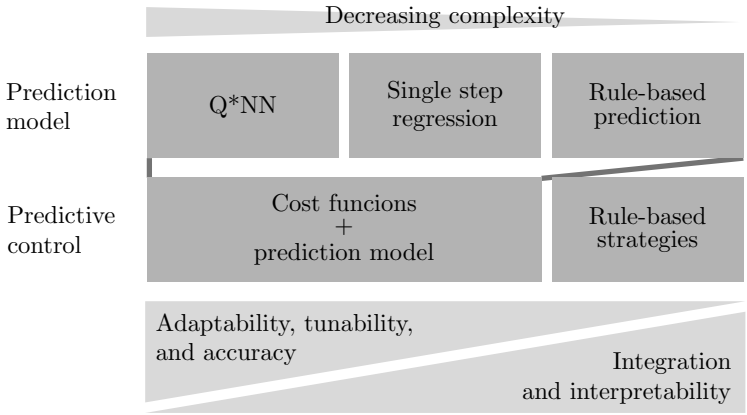


**Figure 3.10:** Complexity levels of the predicion model and predictive control.

A reduction of the computational requirements can be achieved with simpler prediction models (e.g. single step regression). However, their predictions provide less information and might be less accurate. This also holds for rule-based prediction models that are based on Shallow Decision Trees. They require the least computational resources compared to the other considered models, but show the lowest accuracy. Rule-based strategies can be derived based on simulations

of the Q*NN predictive control. In this case, Shallow Decision Trees are used to directly output the control value or desired system state. Look-up tables derived from these trees can offer the easiest integration in the target system. They allow the highest interpretability of results because of their direct, rule-based mapping of inputs to outputs. This is advantageous for transparency and explainability of the predictive control strategies. On the contrary, the performance of the rule-based control is limited by its shallow tree architecture and the scenarios from which it is derived. It is important to note that the development of such a rule-based control still requires the previous steps of developing and optimizing a prediction model (e.g. Q*NN) and predictive controller.

Another aspect is the availability of input data for the development pipeline as introduced in section 3.1. During the development of a real-world system, only few simulation data and few measurement data are available before the system is handed over to customers. Therefore, Transfer Learning of a model trained on large data sets from previous systems to the new system might be necessary for the application of the predictive control (see 3.2.4). Further tuning of the predictive control parameters (e.g. weights of the cost functions) might be necessary as well.

# 4 Application for battery thermal management of battery electric vehicles

In this chapter, the method of predictive control using Deep Learning based Quantile Neural Networks (Q*NN) is applied for the Battery Thermal Management System (BTMS) of Battery Electric Vehicles (BEV). The structure of this chapter mirrors the structure of the previous chapter. In 4.1, the problem statement and development pipeline are described for a battery thermal management. The development of battery temperature prediction models is explained in 4.2. The prediction models are used in section 4.3 for a predictive battery thermal management. The applicability of the prediction models and predictive control is assessed in 4.4.

## 4.1 Problem statement and pipeline

Battery thermal management of BEV often uses a fixed threshold for simple and robust control of battery cooling and heating (see 2.1.2.3). In contrast, predictive strategies adapt to different driving scenarios and information about the route ahead. For example, if the route ahead leads to a low increase in battery temperature, it can be tolerated when the battery temperature exceeds the fixed cooling threshold without cooling activation within an acceptable range and time. This benefits in a reduction in energy consumption, while battery derating and ageing are within tolerable ranges. However, a predictive control requires accurate prediction models, for example using data-driven prediction models based on fleet

data. Thus, a problem statement can be defined by the question: How can fleet data and foresight data be used to optimize the cooling strategies of a battery thermal management?

The developed method from chapter 3 is employed to answer this research question. Cross-domain data of a BEV model are collected and processed for the development of battery temperature prediction models. The data need to include both novel cooling strategies and a large amount of realistic profiles under real driving conditions. For the application of battery temperature prediction, foresight input data are expected to be provided by the navigation system, in particular for trips with active route guidance. Navigation data are location-based, such that the input and output sequences of the prediction model are sampled to fixed distances in this work. A fixed sampling distance of 250 m segments is chosen because of two reasons: Firstly, the low dynamics of the battery thermal behavior do not require smaller segments. Secondly, the limited spatial resolution of navigation data often cannot provide foresight input data for smaller segments.

The collected and processed data are used for training and evaluation of Q*NN and reference prediction models. A rule-based prediction model based on Shallow Decission Trees (SDT) is considered as additional reference with lower computational requirements (see 3.2.2.2). Transfer learning is investigated considering possible model transfer to other vehicle models for which only few data are available. The best performing prediction model is used for a predictive battery thermal management to adapt the battery cooling threshold. Cost functions are related to the key battery properties: energy consumption, ageing and power derating. The predictive control is evaluated for different scenarios and a rule-based model is derived as alternative with lower computational requirements.

The applicability of the resulting models is assessed with a focus on an onboard integration into new battery electric vehicles. On the one hand, such an integration can pose strict requirements concerning resource availability, for example limited storage and Random Access Memory (RAM) of the Electronic Control Unit (ECU). On the other hand, the models need to be able to deal with the complexity of the underlying physical relations and assumptions about the route

ahead. Thus, information about model uncertainty is beneficial. Furthermore, the assessment considers data availability and tunability of model parameters.

## 4.2 Model for battery temperature prediction

Battery temperature prediction models are developed as described by the method in 3.2 and in [1, 3]. The model output is the change in battery temperature with respect to the current battery temperature. The steps include data collection and preparation (4.2.1), model development (4.2.2) and model testing (4.2.3). Rule-based prediction and model transfer (4.2.4) are investigated to increase the applicability of the method.

### 4.2.1 Data collection and preparation

In order to train and evaluate battery temperature prediction models, suitable data need to be collected and prepared. Battery data are collected from different sources and analyzed if they provide plausible information about battery thermal behavior. Processing involves filtering and resampling the data as well as reshaping the data for model training and testing.

#### 4.2.1.1 Data source and understanding

Two main data sources provide the input for the development of battery temperature prediction models: simulations and a vehicle fleet. Same as in [1, 3], the simulations are calculated with new values of battery cooling thresholds compared to the fleet data. However, the fleet data cover a large variety of real drive profiles, without simplifications that are present in a simulation model. The data sets used in the publications [1, 3] and in this work are shown in Fig. 4.1. A Quantile Convolutional Neural Network (referred to as QCNN22) is trained on simulation data and small vehicle fleet data combined with weather data in [1]. In [3], improved

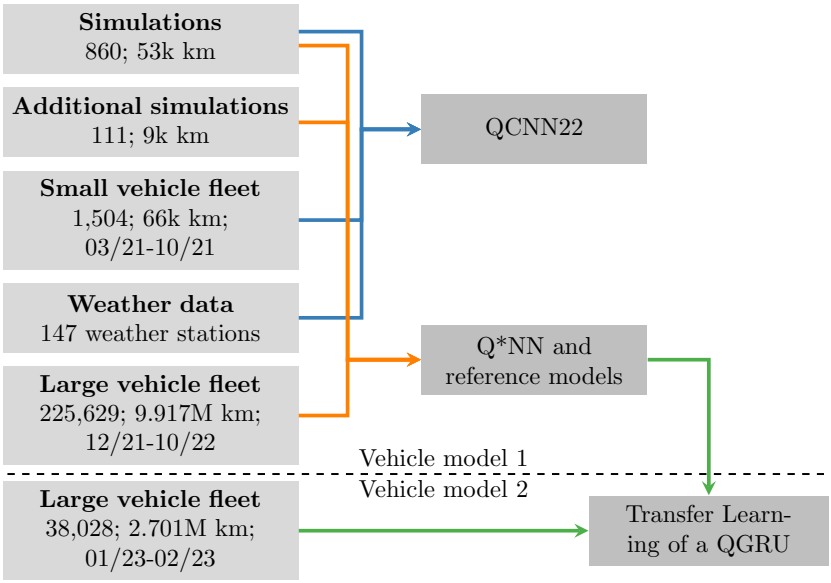


**Figure 4.1:** Overview of the collected and processed cross-domain data used in this work. QCNN22 refers to [1], Q*NN and reference models to [3] and Transfer Learning to 4.2.4.

Q*NN models and reference models are trained on an extended simulation data set and vehicle fleet data that contains 150 times more trips (225,629) than the small vehicle fleet (1,504). However, the large vehicle fleet data from that vehicle model do not contain road height data, in contrast to the small vehicle fleet data and the data from vehicle model 2. Transfer Learning is investigated as an approach to adapt a Q*NN for application in another vehicle model with corresponding fleet data (vehicle model 2). An analysis of the data sets is provided in the following.

The simulation data are generated with a given, validated vehicle simulation model in Dymola. A Design of Experiments (DOE) is used to obtain information about the battery thermal behavior for five battery cooling thresholds (25 °C, 30 °C, 35 °C, 40 °C, 45 °C) with a 2 °C hysteresis (which leads to cooling deactivation when the temperature is lower than 2 °C below the threshold). The initial battery temperature is chosen such that the cooling thresholds are reached or exceeded in the simulations. Furthermore, the ambient temperature is varied in

ranges corresponding to the initial battery temperature. The simulation data used in [1] are extended by simulations with additional profiles in [3]. The simulation data used for model testing are the same (i.e. they are not extended in [3]). In [1], a simulation data analysis indicates a plausible effect of the thresholds on battery energy consumption and battery temperature. The same analysis is done for the extended simulation data (including all 971 simulations except for the test data). The resulting distributions are shown in Fig. 4.2 with the same plausible effects of the battery cooling thresholds. Additionally, the effect on the average power limit is included (c), which correctly shows reduced maximum available power for higher cooling thresholds. Furthermore, the cross correlation as described in 3.2.1.1 shows the expected shift to later battery cooling activation for higher thresholds [1]. For the next steps, the relevant physical quantities and simulation parameters are selected according to the physical equations from section 2.1.2.1 [1]. Furthermore, pairwise Pearson correlation helps to eliminate redundant features [1]. A list of the selected features as prediction model input is provided in A.3.

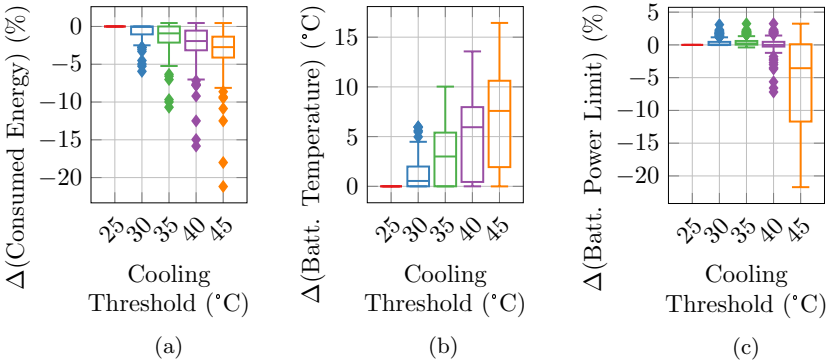


**Figure 4.2:** Simulation results with different battery cooling thresholds. The changes in total energy consumption (a, based on [1]), average battery temperature (b, based on [1]) and maximum available power (c, as average sum of the absolute power limit for charging and discharging) are shown with a 25 °C threshold as baseline.

The vehicle fleet data are sampled to segments with a fixed distance of 250 m by calculation of the average value of each signal for the segment. Same than for the simulation data, the signals are selected based on the physical equations and Pearson correlation [1]. If a single signal value is missing, the gap is replaced by the previous value or using linear interpolation. Signals with discrete values are smoothed by a moving average of five segments, for example the ambient temperature, battery temperature, and the State of Charge (SOC). An analysis in [1] shows acceptable inaccuracies of the distance-based sampling on the simulation data. Speed, sensor resolution, and sensor inaccuracy can lead to an additionally driven distance of the fleet data segments, such that the segments are not all equally covering 250 m. It occurs when the distance sensor value (that triggers the sampling) is received late after the 250 m sampling distance has been passed. The additional distance $d_{add}$ is calculated in a simplified way by (4.1), with signal resolution $t_{sig}$, constant speed $v$, sensor inaccuracy $d_{sensor}$, and sampling distance $d_{sample}$ (250 m). Fig. 4.3 shows that the effect is negligible for a resolution of 1 s, even with additional 10 m sensor inaccuracy. For comparison, a lower resolution of 5 s is included, which results in higher maximum inaccuracy for higher velocities.

$$d_{add} = t_{sig} \cdot v \cdot \left(1 - \frac{d_{sensor}}{d_{sample}}\right) \cdot \left\lceil \frac{d_{sample}}{t_{sig} \cdot v \cdot (1 - \frac{d_{sensor}}{d_{sample}})} \right\rceil - d_{sample} \quad (4.1)$$

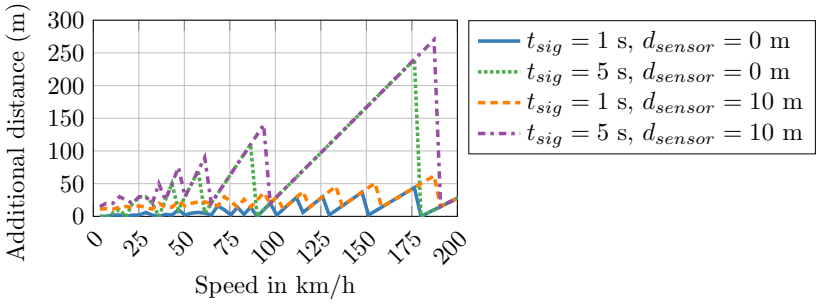


**Figure 4.3:** Additional driven distance of 250 m segments caused by inaccurate distance measurement with signal resolution $t_{sig}$ and sensor inaccuracy $d_{sensor}$.

The fleet data are filtered for trips with more than 5 km in [1] because the change in battery temperature is limited for very short distances due to the battery thermal capacity. The large fleet data are filtered for trips with more than 20 km for sequence clustering [3]. While the small vehicle fleet data is collected in Germany from Spring to Autumn, the large fleet data is collected for countries from different climate zones and including the winter months of the northern hemisphere. Therefore, the large fleet data are additionally filtered for trips without active battery heating. The number of trips and total driven distance in Fig. 4.1 are obtained from the according data sets after the filtering. The validity and occurrence of ambient temperatures is analyzed in [1] by joining weather data from the German weather service. The method of joining and comparing the weather data is further described in the appendix A.1. The analysis confirms that the ambient temperature measured by the vehicle fleet is comparable to the weather data.

An important aspect of the vehicle fleet data analysis is the occurrence of battery temperature changes over segments. This helps to define the horizon of the sequence predictions of the prediction model and to identify possible data imbalance. According to an investigation in [1], a prediction horizon of 20 km is chosen for the prediction model. The small vehicle fleet data (of vehicle model 1) are further analyzed concerning the occurrence of the difference between maximum and minimum battery temperature over 20 km, as shown in Fig. 4.4. In more than 50 % of the collected segments, the maximum change in battery temperature is less than 1 °C. The distributions for both large vehicle fleet data sets have been added. The large vehicle fleet data of vehicle model 1 shows even higher shares of small battery temperature changes (more than 80 % with less than 1 °C difference). The large differences in occurrence between smaller and larger battery temperature changes show the need for data balancing before using the data for training and testing of predictions models, as described in 2.3.1.

A sequence clustering of the battery temperature change is used in [3] for further analysis of the occurrence of different behavior in the large vehicle fleet data. While the previous analysis focused on the maximum difference, sequence clustering can take more information such as the derivative and curvature of the battery

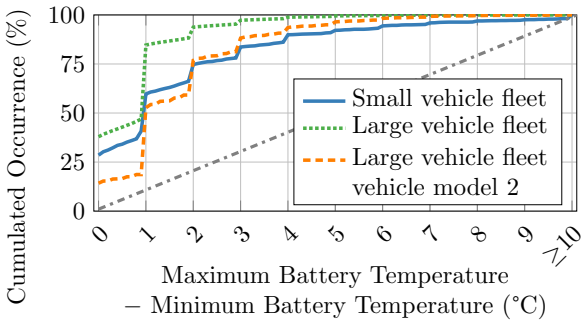


**Figure 4.4:** Occurrence of maximum battery temperature change over 20 km in the small vehicle fleet data. In contrast to the occurrence diagram in [1], the shown occurrences are accumulated and contain the whole vehicle fleet data set, without simulation data. The data are more imbalanced the larger the distance between the cumulated occurrence and the linear line.

temperature change into consideration. Sequences are picked with an overlap of 25 % between two consecutive sequences, which reduces redundancy in the data compared to keeping each sequence shifted by one segment (see 3.2.1.2). The mini-batch k-means algorithm is applied for all cluster numbers between 2 (data should be split into at least two clusters) and 29 (picked large enough for further analysis, but limited by computation time). The resulting normalized silhouette score and sum of squared distance (SSD) are shown for both large fleet data sets in Fig. 4.5. Seven clusters are chosen to achieve both a low SSD and a high silhouette score at the same time. Furthermore, a higher number of clusters allows better consideration of rare battery temperature behavior during clustering-based data balancing. For example, if too few clusters are chosen, important battery temperature behavior can be lost by random undersampling.

The clusters can be visualized using Uniform Manifold Approximation and Projection (UMAP) for dimension reduction as done in [3] and included in Fig. 4.6. It shows a good clustering result since there is only few overlap between the clusters for both of the large vehicle fleet data sets (a and c). Outliers of each cluster are removed using the isolation forest algorithm with 1 % outlier removal rate (also called contamination). The averaged curves of each of the seven clusters are obtained by Dynamic Time Warping based Barycentric Averaging (DBA) and

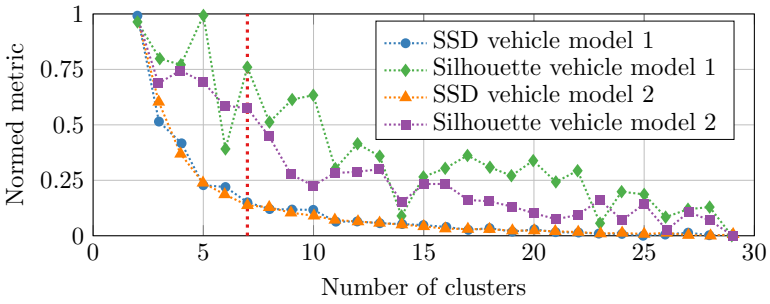


**Figure 4.5:** Sum of squared distance (SSD) and silhouette score of mini-batch k-means clustering for both large vehicle fleet data sets. All four curves are normalized by their maximum value. A vertical line indicates the selected number of clusters (7).

shown in Fig. 4.6 (after outlier removal), together with the occurrence of each of the seven clusters. The cluster occurrence confirms the imbalance of both large fleet data sets, for example with 53 % (b) and 34 % (d) of the data with no change in battery temperature (cluster 4). In comparison, the data set of vehicle model 2 (d) shows more dynamic battery temperature behavior. The seven clusters are additionally shown in appendix A.2 with a random subset and the corresponding DBA for both large vehicle fleet data.

#### 4.2.1.2 Processing

The vehicle fleet data are processed for training, validation, and testing of battery temperature prediction models. Missing sensor values are replaced by linear interpolation as described in 4.2.1.1. Otherwise the gaps are replaced by zero and an additional qualifier feature indicates if it is a replaced value. A list of all qualifier features is included in A.3. The input data are reshaped into windows of history and foresight input as described in 3.2.1.2. The history horizon $h_h$ is chosen to 5 km and the foresight horizon $h_f$ to 20 km. The prediction output is the change in battery temperature with respect to the current temperature (at the time of prediction). The prediction horizon $h_p$ is set to 20 km, since smaller horizons result in less significant change in battery temperature as shown in [1].

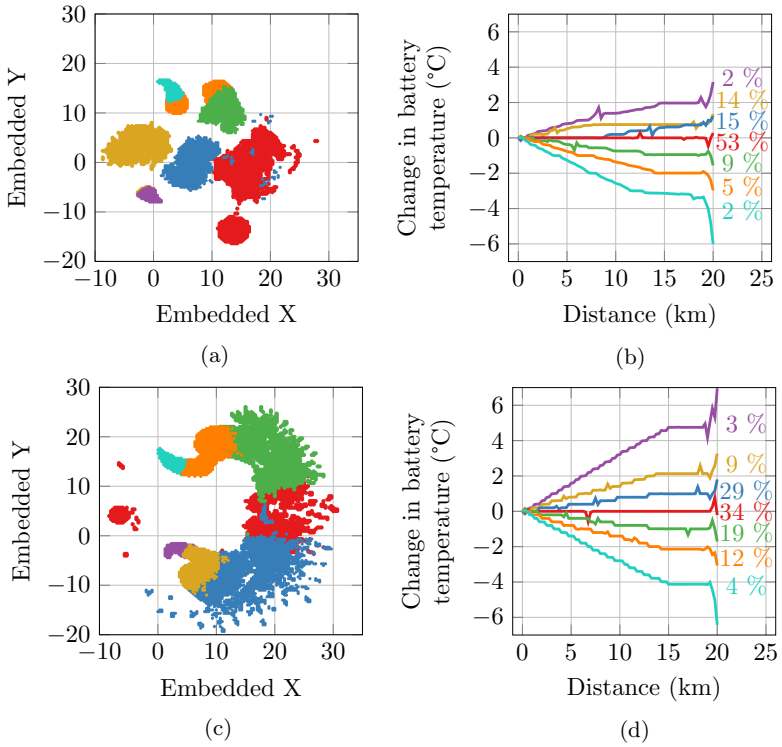


**Figure 4.6:** UMAP (a) and DBAs (b) of the clusters 1 to 7 (order from top to bottom in b) for the large vehicle fleet data from [3]. Accordingly, the UMAP (c) and DBAs (d) for the large vehicle fleet data from vehicle model 2 are included.

The dependency of foresight and prediction horizon on prediction performance can be further investigated during model evaluation and comparison (also see [3]). If the start or end of a trip are within either of the horizons, the exceeding segments of the fixed horizons are filled with zeros. An additional feature indicates which values are exceeding the start or end of the trip.

The vehicle fleet data are balanced according to the previous analysis of occurrence of changes in battery temperature. In [1], the small vehicle fleet data are balanced using a fitting curve for the occurrence in difference between maximum and

minimum battery temperature. The large vehicle fleet data are balanced based on the occurrence of the sequence clusters, as in [3]. At first, the classes are undersampled such that they contain maximum 20,000 samples each. The fleet data are split into training, validation, and testing with a fixed share of 70:15:15 respectively. For each of the three sets, random oversampling is applied such that the number of samples of each class equals the number of samples of the largest class of the set. For the split into training, validation, and testing sets, data selection is conducted by whole trips and not segments to avoid data leakage between the training, validation, and testing sets.

Data imbalance is also given by the different size of the simulation data set compared to the large vehicle fleet data. Therefore, the IBA data augmentation method (3.2.1.2) is applied to enlarge the training and validation simulation data without the need to conduct additional time-consuming simulations. In [3], a tolerance value of 3 is chosen such that the resulting augmentation factor leads to a simulation data set size close to the size of one large vehicle fleet cluster after random undersampling. The simulation test data are not augmented such that the prediction model evaluation is not influenced by possible inaccuracies introduced by the IBA tolerance value. In order to remove redundancy in the simulation data, an overlap is applied for the training and validation data set in [3]. The overlap is set to 75 %, compared to 25 % for the large vehicle fleet data, to consider the smaller simulation data size. The simulation test data set is used completely without overlap filtering due to its size which is not increased by data augmentation. Clustering and data balancing within the simulation data are not conducted due to the limited size and the DOE, which puts focus on rare cases with higher battery temperature change.

Final data processing includes centering and normalization of each feature to the range -1 to 1. The training, validation, and testing data are composed of both the vehicle fleet data and the simulation data. The training data are shuffled to a random order of the samples. Table 4.1 provides an overview of the data set sizes after processing, used for model development and testing.

**Table 4.1:** Data set size for training, validation and testing after data processing, based on [3].

| Data set | Training | Validation | Test |
|---|---|---|---|
| Vehicle fleet | 86.5% | 80.4% | 85.7% |
| Simulation | 13.5% | 19.6% | 14.3% |
| Sum | 142,809 | 13,634 | 13,107 |

### 4.2.2 Model development

The processed training and validation data are used for building different types and architectures of battery temperature prediction models. The validation loss is used to evaluate Q*NN parameters (4.2.2.1) and regression model architectures (4.2.2.2). The best performing models are tested and compared in 4.2.3.

#### 4.2.2.1 Quantile Neural Networks (Q*NN)

Q*NN battery temperature prediction models are developed in [1, 3] using different data sets (see Fig. 4.1). QCNN are developed based on simulation data and small vehicle fleet data with additional weather data [1]. In this work, the resulting model from [1] is referred to as QCNN22. Different Q*NN architectures (QCNN, QGRU, QLSTM) are built based on extended simulation data and large vehicle fleet data [3]. As described in 3.2.2.1, the choice of fixed and varied hyperparameters is the same for all Q*NN models, except for the number of convolutional or recurrent layers $N$ in the combined channel.

The varied hyperparameters are optimized using random grid search (QCNN22, [1]) or efficient Bayesian optimization (QCNN, QGRU, and QLSTM, [3]). The hyperparameter ranges are obtained from an exploratory search and provided in Table 4.3. In random grid search for QCNN22, one out of three values is randomly selected for each hyperparameter. For each set of hyperparameters, three models are trained to cope with random initialization of the Neural Network

weights. 50 unique hyperparameter sets are considered (which covers 21 % of all possible combinations), such that in total 150 models are trained. The efficient Bayesian optimization of the Q*NN models can select any value within the chosen hyperparameter range, with additional constraints for the step size and sampling as specified in Table 4.3. 20 trials with different hyperparameters are evaluated, with three models (QCNN, QGRU) or two models (QLSTM, due to longer training time of LSTM layers) per trial. The amount of training data used per trial is given by (3.5) in 3.2.2.1.

**Table 4.3:** Considered ranges of hyperparameter optimization. For the random grid search of QCNN22, one of three values is selected each. The Bayesian optimization selects any value in the range between the given bounds. If a fixed step size or logarithmic sampling are used, the according value is included between the ":".

| Hyperparameter | QCNN22 [1] | Q*NN [3] |
|---|---|---|
| Base number of layer nodes | [128, 192, 256] | [128 : 32 : 256] |
| Dropout rate | [0.2, 0.3, 0.4] | [0.2, 0.4] |
| Loss weight $w_{mse}$ | [0.01, 0.03, 0.1] | [0.01, 0.1] |
| Learning rate | [$10^{-3}$, $10^{-4}$, $10^{-5}$] | [$10^{-3}$ : log : $10^{-5}$] |
| Mini-batch size | [32, 64, 128] | [32 : 16 : 128] |
| Number of sequential layers $N$ | fixed | 2 or 4 (combined channel) |

The average validation loss for each trial and model type is given in Fig. 4.7, scaled to a range between 0 and 1 for each model. The efficient Bayesian optimization results in higher validation loss for the first trials of QGRU and QLSTM. For QCNN, the efficient Bayesian optimization does not show a clear trend, but the highest loss occurs within the first 30 % of the trials. The validation loss of QCNN22 does not show an increasing or decreasing trend, as is expected for a random grid search. For each of the best Q*NN hyperparameters, three models are trained out of which the best performing model is used in model testing.

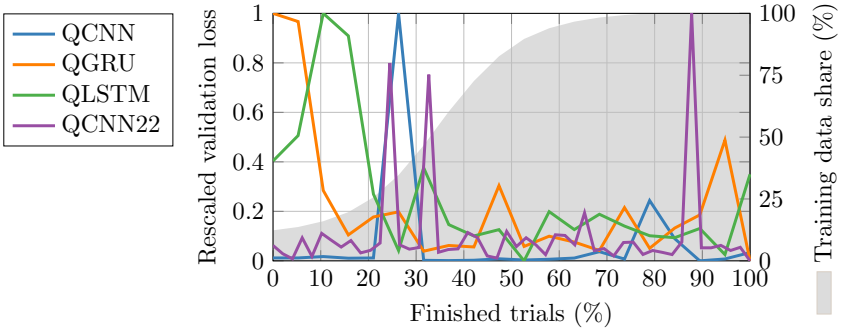


**Figure 4.7:** Validation loss for each optimization trial, scaled to a range between 0 and 1 for each model type. The gray area indicates the training data share for QCNN, QGRU and QLSTM.

The efficient Bayesian optimization reduces the computation time for QGRU by 64 % from 16 days to 6 days compared to a Bayesian optimization with all training data from the beginning. Furthermore, the test metrics of the best performing QGRU from the efficient optimization (RMSE: 0.61, $R^2$: 0.86, WS: 1.13) are not worse (compared to RMSE: 0.68, $R^2$: 0.83, WS: 1.26). The number of parameters of each model is as follows, sorted from least to most:

- QGRU: 6,395,924 parameters
- QCNN22: 7,128,228 parameters
- QCNN: 7,959,604 parameters
- QLSTM: 20,001,244 parameters

Table 4.5 provides the number of input features, multiplied by the number of segments per horizon and the size of the training data set. The numbers are given for independent input features and for all input features including calculated or QL features which are based on the independent input features. The resulting number of input values is between 6 and 41 times larger than the number of parameters of the Q*NN models. Thus, the Q*NN models are not expected to overfit the data, additionally taking their regularization layers ("BatchNorm" and dropout) into account.

**Table 4.5:** Number of input features (based on A.3) and resulting number of input values of the Q*NN.

| | Independent | Including calculated and QL |
|---|---|---|
| History features | 19 | 39 |
| · history segments (20) | 380 | 780 |
| Foresight features | 6 | 13 |
| · foresight segments (80) | 480 | 1040 |
| Sum (history and foresight) | 860 | 1820 |
| · training data size (142,809) | 122,815,740 | 259,912,380 |

#### 4.2.2.2 Reference models

The reference regression and quantile regression models are trained using the same large vehicle fleet data after outlier removal and data balancing than the Q*NN models, as proposed in [3]. Since the models are less complex than Deep Learning models, the input features need to be further reduced. Only key physical quantities are used, selected with consideration of domain knowledge about the battery thermal behavior and the results from 4.2.1.1. Furthermore, features with missing data (and thus require QL features) are not included, e.g. road height differences which are not available for the large vehicle fleet data from vehicle model 1 (see 4.2.1.1). The sequences of the features are sampled into single values per sequence. As a result, the following input features are sampled for a fixed horizon, as cited from [3] (speed difference refers to the difference between consecutive segments):

- “Battery temperature: Current value.
- Ambient temperature: Current value.
- Upper battery cooling threshold: Current value.
- Lower battery cooling threshold: Current value.

- Speed: Maximum, minimum, mean, standard deviation.
- Speed difference: Maximum, minimum, mean, standard deviation."

A regression or quantile regression model is trained to predict the change in battery temperature as a single-step prediction. In order to obtain the battery temperature change as sequence over a horizon of 20 km, four models are individually trained with a different horizon each (5 km, 10 km, 15 km, 20 km) and their single-step predictions (one value for the respective horizon) are combined to a sequence. Three categories of reference models are built: the best performing regression model from an automated Machine Learning toolbox called Pycaret [225], a quantile regression model using the Python package scikit-garden [226], and a Shallow Decision Tree (SDT) using the Python package scikit-learn [234]. The SDT is further used as rule-based prediction model in 4.2.4. For better interpretability of the tree nodes, the data are not normalized for training and testing of the SDT.

The results of the Pycaret regression model comparison for the validation data show lowest Root Mean Squared Error (RMSE) of the Light Gradient Boosting Machine (LGBM) for the 5 km horizon and of the Extra Trees Regressor for the remaining horizons [3]. Hence, the reference model using these four regression models is called LGBM-ETR, as done in [3]. Due to the good performance of ETR in the Pycaret comparison, Quantile Extra Trees Regressor (QETR) is used as quantile reference model. The default hyperparameters are used for both LGBM-ETR and QETR, resulting in 100 estimators (boosting stages in LGBM and trees in ETR and QETR). The maximum tree depth and maximum number of leaves are limited for the Shallow Decision Tree (SDT) to keep its complexity low [3]. These two hyperparameters are optimized by grid search. The following values are investigated, with the hyperparameters resulting in lowest validation loss (RMSE) marked in bold:

- Maximum tree depth: 4, **5** (5 km), 6, **7** (all other horizons), 8, 9
- Maximum number of leaves: 9, 11, 13, 15, **17** (all horizons)

### 4.2.3 Model testing

Testing of the prediction models includes the evaluation of point-prediction performance, quantile-related performance, an exemplary analysis, and further investigation about feature importance. The metrics and visualizations are compared for both the Deep Learning based Quantile Neural Networks (Q*NN) and the reference models. Therefore, only the prediction values for the four horizons (5 km, 10 km, 15 km, 20 km) are used from the Q*NN predictions, unless otherwise stated that the whole sequence over 20 km is considered. All battery temperature predictions are denormalized, such that the evaluation can be done in °C.

#### 4.2.3.1 Point-prediction performance

Point-prediction performance compares the (median) prediction with the true values. Fig. 4.8 shows the RMSE and $R^2$ of each model for the different horizons (a, c), taken from [3]. The proposed Q*NN models perform better in both metrics than the reference models and the QCNN22 model that is trained on the small vehicle fleet. The QGRU is the best model with an RMSE between 0.44 °C (5 km) and 0.86 °C (20 km), with an average of 0.66 °C (over the four horizons), and an $R^2$ between 0.75 (5 km) and 0.89 (20 km), with an average of 0.84 (over the four horizons). The RMSE and $R^2$ are increasing with larger horizons, which is in accordance with higher occurrence of larger battery temperature changes for larger horizons. In Fig. 4.8 (c), the $R^2$ is above the significance level of zero for all models and for all four horizons. However, it is above 0.51 only for the Q*NN for all four horizons (see 3.2.3.1 for the definition of $R^2$ and the significance level).

Furthermore, the metrics are distinguished between the battery cooling thresholds in Fig. 4.8 (b, d), for which the metrics are averaged over the horizons. Consequently, the metrics are split into simulation and vehicle fleet data since only the simulation data contain a novel variation of the battery temperature thresholds. The reference regression models (e.g. LGBM-ETR) show different performance dependent on the threshold and if it is simulation or vehicle fleet data. Moreover,

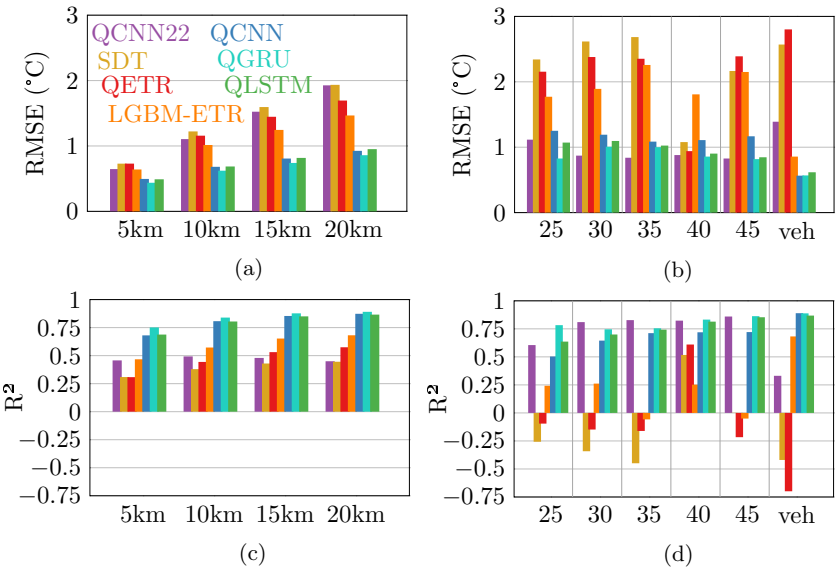


**Figure 4.8:** Metrics of the (median) prediction for different horizons (a,c), taken from [3], and different cooling thresholds in °C (b,d) ("veh" means vehicle data). The models are listed in (a), the labels from top left to bottom right map to the bars from left to right.

the $R^2$ of the reference regression models is negative for several cooling thresholds (d), indicating large prediction errors (see 3.2.3.1). The performance of a predictive control using such models is limited due to the imprecise predictions that are used for choosing the cooling threshold. In contrast, the Q*NN prediction models show comparable performance for the thresholds and vehicle fleet data, which indicates their suitability for application in a predictive control. The QCNN22 performance in (b, d) is comparable to the the Q*NN models for the simulation data, but worse for the vehicle fleet data. Only the $R^2$ of the QGRU and QLSTM models is above the significance value of 0.51 for all five cooling thresholds and the vehicle data, which underlines their good prediction accuracy.

The error distribution of the Q*NN models and the reference models is analyzed in the following. Regression plots of the prediction models are provided in Fig. 4.9, taken from [3]. The regression plots confirm the superior performance of the

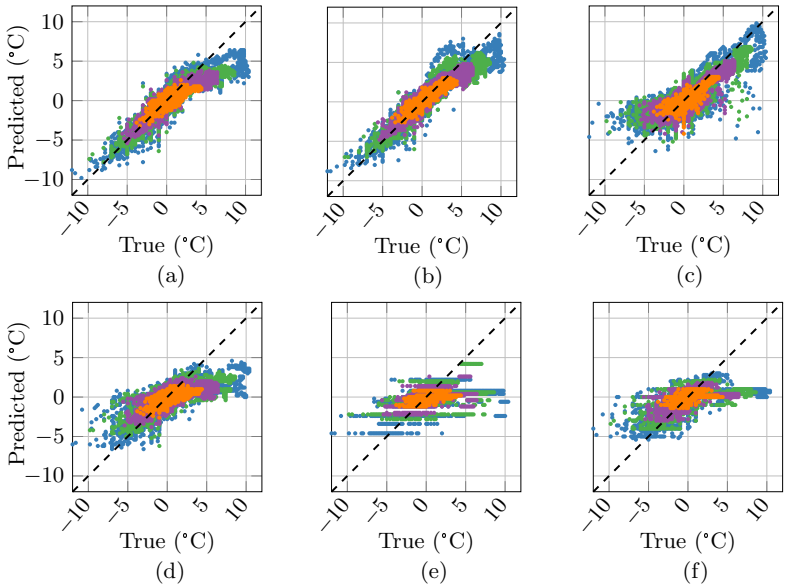


**Figure 4.9:** Regression plots taken from [3], "for QCNN (a), QGRU (b), QCNN22 (c), LGBM-ETR (d), SDT (e), QETR (f)". Larger changes occur with larger horizons, from 5 km (orange, most inner), 10 km (purple), 15 km (green) to 20 km (blue, most outer).

Q*NN models (a, b) to well predict the battery temperature change compared to the QCNN22 (c) and regression models (d to f). The QCNN22 and regression models show large underestimation of larger battery temperature changes (both for large positive and negative values). The QGRU (b) shows least underestimation, also compared with the QCNN (a), but slightly more overestimation for the changes between 0 and 5 °C. The QGRU prediction behavior is favorable since an overestimation of the battery temperature change would result in more conservative cooling strategies in a predictive control, whereas an underestimation could increase battery aging and derating. Good predictions for larger horizons are also favorable for a predictive control to better plan long-term cooling strategies. An alternative visualization of the regression plot can consider the occurrence of the prediction errors instead of different horizons, as done in [1].

While the regression plots from Fig. 4.9 quantify the prediction error, the occurrence of prediction errors can be further investigated with Fig. 4.10. Better performance is achieved when the cumulated occurrence is closer to a step function with a step from zero to one at 0 °C difference between the predicted and true battery temperature change. The order of the model performance is comparable to the metrics from Fig. 4.8. The QGRU and QLSTM show least underestimation and more overestimation of the battery temperature change, since their cumulated occurrences are lowest until approximately 0.5 °C difference (see left and right magnifier window). The QGRU model shows a prediction error between -1.3 °C and 1.3 °C in 95 % of the test data, which is visualized by an arrow in Fig. 4.10. 94 % of the QLSTM and the QCNN predictions and 69 % of the QCNN22 predictions are in the same range. The regression models achieve 87 % (LGBM-ETR), 86 % (QETR) and 77 % (SDT).

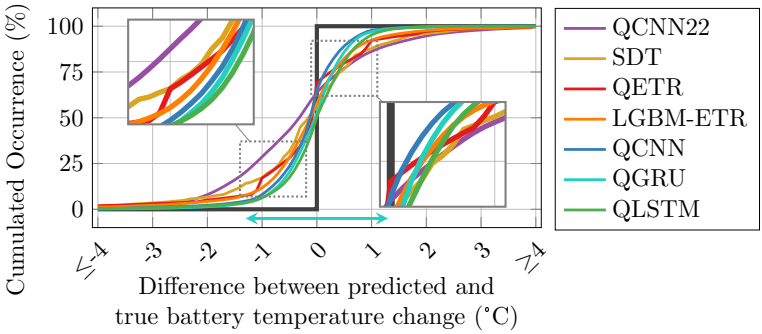


**Figure 4.10:** Cumulated occurrence of the error between the (median) prediction and the true values of the test data. The arrow covers 95 % of the QGRU predictions.

#### 4.2.3.2 Quantile-related performance

Quantile-related performance can be evaluated with various metrics as introduced in 3.2.3.2. The metrics of the Q*NN, QCNN22, and QETR are calculated for the predictions on the test data as done in [3]. The metrics are shown in Table 4.6

**Table 4.6:** Quantile-related metrics for the test data, averaged over the four horizons, as given in [3]. The color coding is for each row from best (green, bold, lowest value or closest to quantile), the median value (white) to worst (red, highest value or farthest from quantile).

| Metric | QCNN22 | QETR | QCNN | QGRU | QLSTM |
|---|---|---|---|---|---|
| $p(q_j)_{d,abs}$ | 0.20 | 0.05 | **0.03** | 0.04 | 0.05 |
| $p(q_j, 1-q_j)_{d,abs}$ | 0.44 | 0.06 | 0.05 | 0.06 | **0.02** |
| WS | 16.81 | 1.54 | 2.42 | **1.27** | 1.39 |
| $\psi_{avg}$ (°C) | **1.03** | 2.67 | 1.37 | 1.54 | 1.86 |
| $\psi_{max}$ (°C) | 6.10 | 7.67 | 4.51 | **4.11** | 5.37 |
| CORS in $10^{-3}$ | 1.83 | **0.00** | 12.09 | 0.02 | 1.70 |
| 0.50 quantile (%) | 41.27 | 40.40 | **46.35** | 56.19 | 60.10 |
| 0.99 quantile (%) | 68.95 | 98.29 | 95.99 | **98.87** | 98.13 |

as average over the quantiles and horizons. The LGBM-ETR and SDT models are not included, because they do not provide quantile predictions. The QCNN22 and QETR model show worse performance on average compared to the Q*NN models. However, their average sharpness $\psi_{avg}$ and CORS are similar or even better than for the Q*NN models. Overall, the QGRU performs best, with the best values for Winkler Score (WS) and maximum sharpness $\psi_{max}$, and no metric with very bad performance in comparison. Moreover, its share of true values below the 0.99 quantile interval is 98.87 % (closest to 99 %).

Section 3.2.3.2 introduces the share of true values within quantiles ($p(q_j)$) and quantile intervals ($p(q_j, 1-q_j)$) as calibration metric. In a quantile histogram in Fig. 4.11, the achieved occurrences are compared with the occurrence per quantile definition (wide, gray bar in the background). For example, 15 % of the true values are expected to be between the quantiles 0.1 and 0.25. In comparison, the range between the predicted quantiles 0.1 and 0.25 from the QGRU covers 16 % of the true values, the QLSTM predictions cover 19 % (too high), and the QCNN22 only 6 % (too low). Considering all intervals, the QCNN22 shows the worst performance with high occurrence of true values in the intervals with

supposedly small occurrence ($\leq$0.01, >0.99). The QETR shows comparably good performance, while the Q*NN models achieve the best match of true occurrence related to the definition of the quantile intervals.

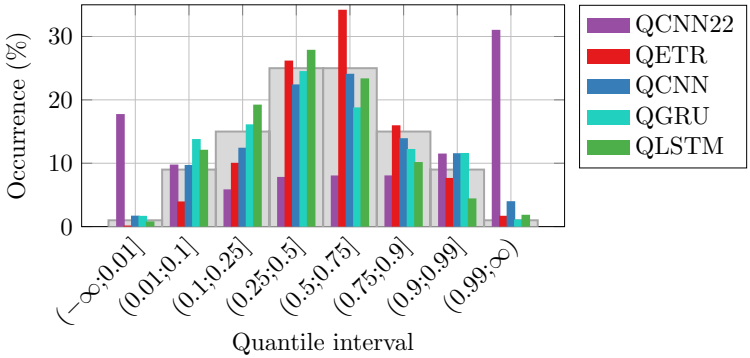


**Figure 4.11:** Occurrence of true values between the predicted quantiles compared to their definition (gray bars in the background). The upper bound values are included (marked with "]").

The evaluation of point-prediction and quantile-related performance indicated that the QGRU model performs best in comparison with the other Q*NN, QCNN22, and reference regression models, even though it has less parameters than the other Q*NN and the QCNN22. While the evaluation was focused on the predicted values for the four horizons 5 km, 10 km, 15 km, and 20 km, Table 4.7 includes the QGRU metrics on the test data for the whole predicted sequence of the 20 km horizon, same than the evaluation in [1]. The 0.5 quantile (median) prediction shows a Mean Absolute Error (MAE) of 0.42 °C and an $R^2$ of 0.86. The median prediction covers 53 % of the true values and the 0.99 quantile prediction covers 98 % of the true values (see $p(q_j)$). This table indicates a good QGRU performance not only for the four horizons but also for the whole sequence prediction.

**Table 4.7:** Point-prediction metrics (first group) and quantile-related metrics (second group) of the QGRU. The metrics are calculated per quantile (columns), with the median shaded in gray.

| quantile | 0.01 | 0.10 | 0.25 | 0.50 | 0.75 | 0.90 | 0.99 |
|---|---|---|---|---|---|---|---|
| MSE ((°C)$^2$) | 2.10 | 0.78 | 0.49 | 0.38 | 0.46 | 0.69 | 1.92 |
| MAE (°C) | 1.18 | 0.64 | 0.48 | 0.42 | 0.49 | 0.64 | 1.19 |
| RMSE (°C) | 1.45 | 0.88 | 0.70 | 0.61 | 0.67 | 0.83 | 1.39 |
| $R^2$ | 0.24 | 0.72 | 0.82 | 0.86 | 0.84 | 0.75 | 0.30 |
| $\sigma_{q_j}$ (°C/km) | 0.61 | 0.18 | 0.15 | 0.15 | 0.15 | 0.17 | 0.55 |
| $p(q_j)$ | 0.01 | 0.15 | 0.31 | 0.53 | 0.72 | 0.87 | 0.98 |
| $p(q_j, 1-q_j)$ | 0.97 | 0.72 | 0.41 | n/a | 0.41 | 0.72 | 0.97 |
| WS | 1.09 | 1.17 | 1.12 | n/a | 1.12 | 1.17 | 1.09 |
| CORS in $10^{-3}$ | 0.46 | 12.51 | 29.32 | 36.66 | 10.36 | 0.25 | 0.25 |
| $\psi_{avg}$ (°C) | 2.34 | 1.11 | 0.57 | n/a | 0.57 | 1.11 | 2.34 |
| $\psi_{max}$ (°C) | 9.93 | 4.45 | 2.25 | n/a | 2.25 | 4.45 | 9.93 |

#### 4.2.3.3 Exemplary analysis

In contrast to the quantitative performance evaluation of the previous sections, the exemplary analysis aims at a qualitative comparison for a more tangible understanding of the prediction model performance for a later application. The analysis focuses on scenarios with specific system behavior, such as larger changes in battery temperature, battery temperatures reaching the battery cooling threshold and thus activating cooling, and in comparison the same scenario with higher thresholds without battery cooling activation. These scenarios are included in the evaluation of the QCNN22 in [1] and provided in Fig. 4.12.

The examples show that the model adapts its predictions to different battery cooling thresholds (comparing (a) to (b)). The model can predict the cooling behavior more accurately for positions that are close to the current position and provides predictions even when the end of the profile is reached within the prediction horizon (c). The model can provide correct predictions for the small vehicle fleet

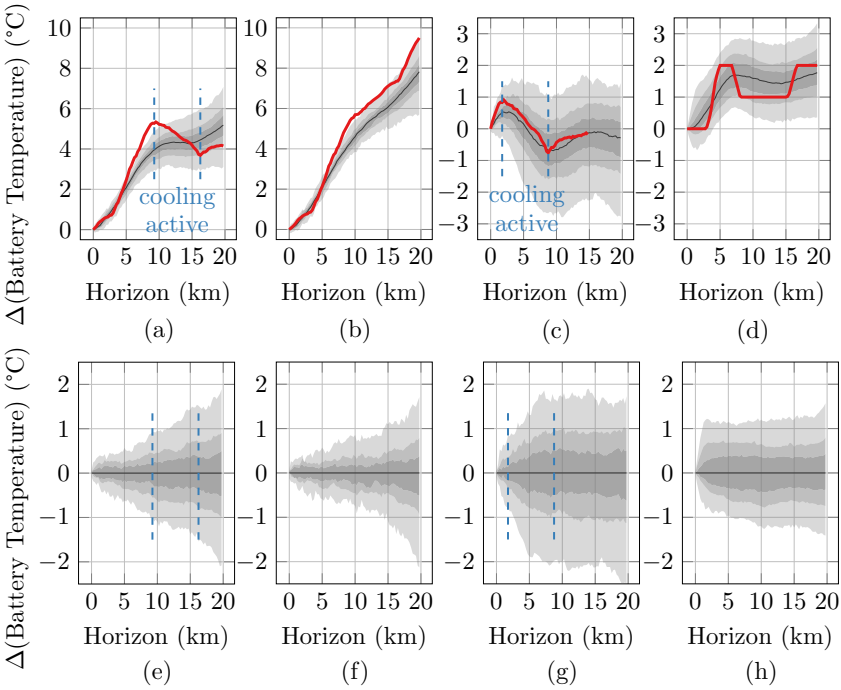


**Figure 4.12:** Example predictions of the QCNN22 from [1], with the true values as thick red line, the median as thin black line and the quantile intervals as gray area. (a) and (b) show the same scenario with an initial battery temperature of 30 °C but cooling thresholds 35 °C and 40 °C respectively. (c) is the scenario from (a) but 7.5 km later. (d) represents an example from the small vehicle fleet data. (e) to (h) visualize the same predicted quantile intervals from (a) to (d), but with respect to the median prediction (i.e. the median prediction coincides with the x-axis).

data despite the step-like behavior of the true values due to sensor discretization. Furthermore, (e) to (h) show the adaption of the quantile intervals (i.e. the prediction uncertainty over the horizon) corresponding to the scenario. For example, the uncertainty is larger for later points in the horizon (e, f) and when cooling is active (g). For vehicle fleet data, the uncertainty is larger (e.g. with the 98 % quantile interval between -1 °C and 1 °C) starting from the first horizon segments, but does not increase significantly over the horizon. This behavior might be caused by

the discrete sensor values with steps that occur at early segments in the horizon. Further example predictions of the QCNN22 as well as predictions of the QGRU, QCNN, and the reference regression models are included in appendix A.4, taken from [3].

#### 4.2.3.4 Feature importance

Permutation feature importance is calculated for the Q*NN models for better understanding of their dependency on the input features. For the reference regression models, both impurity based feature importance and permutation feature importance can be evaluated. Impurity based feature importance is analyzed in [3] for the LGBM-ETR and QETR for each horizon. For better comparison with the QGRU, the permutation feature importance of the reference regression models is calculated and averaged over the four horizons. Fig. 4.13 shows the permutation feature importance for the LGBM-ETR, QETR, and SDT (a) and the importance of the 15 highest ranked features of the QGRU, distinguished by history and foresight input sequences (b). For each model, the feature importances are normalized such that their sum equals 100 % (for QGRU the sum of history and foresight features together).

The battery temperature $T_b$ and ambient temperature $T_{amb}$ are high ranked features for both the reference regression models and the QGRU. The speed $v$ of the foresight horizon (QGRU) and maximum speed (reference regression models) are ranked high as well. The upper or lower battery cooling threshold $T_{b,cool,upper/lower}$ is also present in the higher ranked features of all models. The second highest ranked QGRU feature is the (within) horizon feature of the foresight horizon, which indicates which horizon segments are exceeding the end of the trip. Additionally, the QGRU shows high feature importance for electric machine temperature ($T_{em}$), the inverter temperature ($T_{inv}$), and positive changes in road height $r$ ($\Delta r_+$). 33 more QGRU history features have a total importance of 15 % and 6 more foresight features of 4 %. In comparison, the permutation feature importance is more equally distributed between the QGRU features than

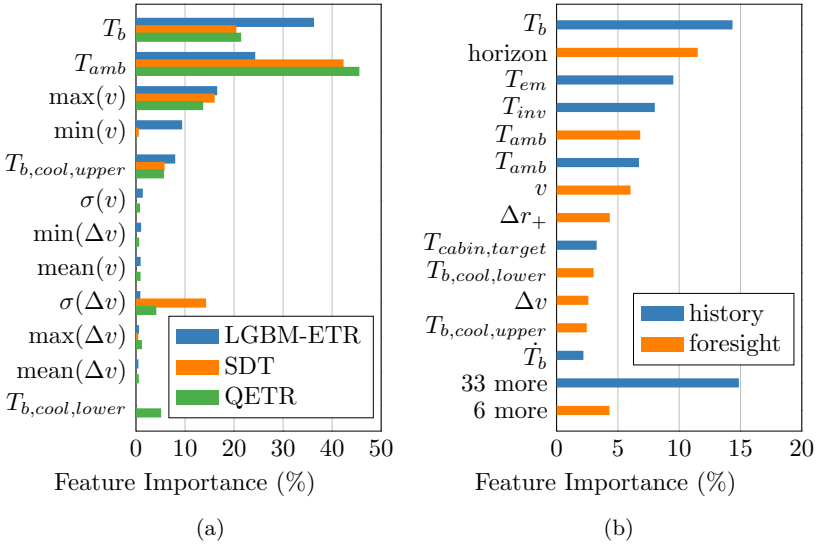


**Figure 4.13:** Permutation feature importance of the reference regression models (a) and the 13 highest ranked features of the QGRU (b). In (b), the features are sequences and distinguished between history input and foresight input.

for the reference regression models. For example, the three highest ranked features of the reference regression models take a share of more than 75 %, while this share is achieved with the 11 highest ranked features for the QGRU.

### 4.2.4 Rule-based prediction and model transfer

Requirements considering the available hardware can limit the allowed complexity of a prediction model. Therefore, the Shallow Decision Tree (SDT) is included in the previous steps of model development and testing. The SDT is limited in depth and number of leaves, such that it does not need large disk storage and RAM. The four models for the different prediction horizons 5 km, 10 km, 15 km, and 20 km each consist of 16 if-conditions and 17 leaves (i.e. different outputs), which results in 33 parameters per model and 132 parameters in total. In contrast to that, for example, the QGRU model has 6,395,924 parameters and requires more

complex functions (e.g. GRU layers), while the if-conditions of the SDT can be translated into any programming language without the need of specific libraries. Thus, the integration of an SDT into the system hardware is easier and requires less storage than the Q*NN. However, the SDT predictions are less accurate, such that the balance between complexity and accuracy need to be further investigated (e.g. using larger SDTs) in future works.

The potential of Transfer Learning (TL) is evaluated in case a prediction model is needed for a novel vehicle model for which a large data set is not available, yet. TL is analyzed using the QGRU as base model and a large vehicle fleet data set of a second vehicle model representing the novel vehicle model (31k trips after data balancing). The vehicle fleet data set has been analyzed and balanced as introduced in 4.2.1.1. Subsets with different size are used to investigate the dependency of the data set size on model performance after TL. Different TL methods are suggested in 3.2.4. Table 4.8 shows the resulting total loss $L_{total}$, RMSE, $R^2$, and Winkler Score (WS) for TL with varying learning rate and different choice of reinitialized layers using a data subset of approximately 500 trips. For each method (row), three models are trained and the results of the best performing model are shown for the validation data. TL using an adapted learning rate but without resetting layer weights shows the best performance. The learning rate is adapted to $10^{-6}$ for all layers from input until (excluding) the sequence layers in the combined channel. Beginning with these sequence layers, a learning rate of $10^{-5}$ is chosen for all remaining layers to the output. For comparison, models are trained without TL (last two rows) by resetting all layer weights, i.e. with the same architecture but without the pretrained weights from the QGRU. The results of TL with freezing layers are not included in the table, because the training and validation loss did not converge to a minimum during training. The table shows metrics for the validation data, while another comparison between data set size and again with the models after resetting all weights follows using the testing data.

Transfer Learning with the adapted learning rate is applied to training subsets with approximately 250 trips, 500 trips, 1000 trips, and using all training data of the vehicle fleet data set after balancing (approximately 25k trips). The RMSE,

**Table 4.8:** Metrics of the best model out of three on validation data for different Transfer Learning (TL) methods and the reference without TL. The table is sorted from lowest to highest total loss $L_{total}$ to identify the best model with and without TL. The best metric is marked in bold each.

| Learning rate | Weights reset | Loss in $10^{-3}$ | RMSE | $R^2$ | WS |
|---|---|---|---|---|---|
| Adapted | None | **0.56** | 1.87 | **0.75** | **1.35** |
| Original | None | 0.57 | 1.68 | 0.73 | 1.99 |
| Adapted | Last | 0.91 | **1.55** | 0.72 | 4.10 |
| Original | Last | 1.19 | 1.64 | 0.55 | 8.66 |
| Original | All (no TL) | 1.25 | 1.83 | 0.49 | 8.60 |
| Adapted | All (no TL) | 2.10 | 2.01 | -0.01 | 52.87 |

$R^2$, and Winkler Score of the best out of three trained models each are shown in Fig. 4.14 for the same test data (approximately 3k trips). For each subset, a comparison is included with a model trained from scratch after resetting all weights (without an adapted learning rate). The results show a plausible effect of increasing performance with larger data size. The smaller the subset the larger the improvement in model performance using TL compared to the corresponding model trained from scratch. As a result, even with the smallest subset of around 250 trips a RMSE of 1.31 °C and a $R^2$ of 0.63 are achieved with the help of TL. A large effect is also shown for the Winkler Score, which describes the quantile prediction performance. However, even when using the whole fleet data set, TL improves the shown metrics. Therefore, the application of Transfer Learning can extend the applicability of the proposed method of battery temperature predictions, even when only few data are available from the target vehicle model.

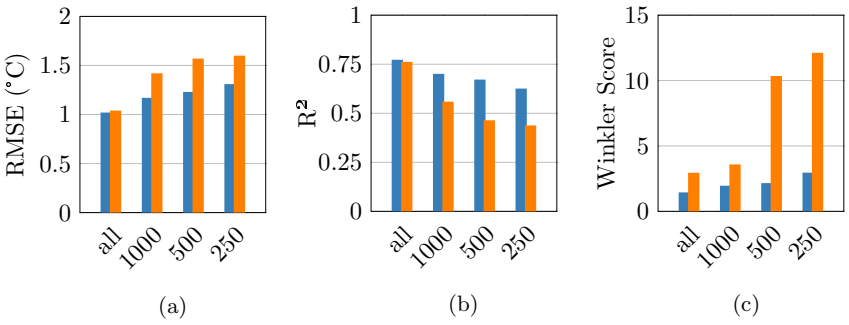


**Figure 4.14:** RMSE (a), $R^2$ (b), and Winkler Score (c) of Transfer Learning (TL) models using subsets with different number of trips. The results are shown for TL using an adaptive learning rate (left, blue bar) and for a model with all weights reinitialized (right, orange bar).

## 4.3 Predictive battery thermal management

A predictive battery thermal management can be implemented using a battery temperature prediction model from the previous step. The control design, cost functions, and evaluation of such a predictive control are described in the following. The potential of deriving rule-based strategies is investigated for real-world applications with reduced computational effort.

### 4.3.1 Control design

The control design of the predictive battery thermal management follows the activity diagram introduced in 3.3.1, and as described in [2]. A vehicle simulation model is calculated for different drive profiles. After each control distance $S_{ctrl}$, a Q*NN model predicts the battery temperature for five different upper battery cooling thresholds (25 °C, 30 °C, 35 °C, 40 °C, 45 °C), which represent the control value. Battery cooling is activated when the battery temperature exceeds the upper battery cooling threshold $T_{b,cool,upper}$ and deactivated when it passes the lower battery cooling threshold $T_{b,cool,lower}$ afterwards, i.e. it is a two-point control with varying thresholds. The hysteresis between $T_{b,cool,upper}$ and $T_{b,cool,lower}$ is fixed to 2 °C for both the predictive control and the reference cooling strategy with

fixed control thresholds. The predictive control is simulated using the QCNN22 (as in [2]) and the QGRU for comparison. Optimization costs are calculated for each threshold, such that the cooling threshold with lowest costs is chosen by the predictive control. The simulation continues with the new cooling threshold until the control distance is reached again (or until the end of the profile is reached).

The control distance is set to 2.5 km, which shows to be a good compromise between the RMSE and the quantile metrics as described in [2]. Furthermore, on the one hand, the distance needs to be set small enough to react on novel foresight data and changing boundary conditions. On the other hand, larger distances reduce the total simulation time since less predictions are required, and the battery temperature is not expected to dynamically change within short distances, as the data analysis shows in 4.2.1.1, [1], and [3].

Predictive control with segmented predictions is conducted using one shift of 2.5 km. This allows the consideration of an adaption of the cooling threshold within the prediction horizon, with only little increase in computational time compared to the required number of predictions with more shifts (see 3.3.1). The shift distance is set to 2.5 km to be equal to the control horizon. Future works need to investigate the effect of different numbers of shifts and shift distances. Features of the history horizon that are not present in the foresight horizons cannot be considered by the prediction model for segmented predictions, since the according data are not available for future states. Therefore, another QCNN is built only using the history features that are also available in the foresight channel. It is trained on the same data set than QCNN22 and with 30 trials of the same hyperparameter optimization than in [1]. For comparison, the total loss $L_{total}$ of the resulting model on the test data (extended simulation and large fleet data as in 4.2.3) is 16 % higher ($1.07 \cdot 10^{-3}$) than for the QCNN22 ($0.92 \cdot 10^{-3}$). The predictive control selects the combination of battery cooling thresholds with lowest costs, using the same cost functions than the predictive control without segmented predictions. The comparison of the predictive control with and without segmented predictions is part of the controller evaluation in 4.3.3.

### 4.3.2 Cost functions

The predictive control chooses the battery cooling threshold in order to minimize a total cost function. The calculation of the cost functions is equivalent to [2]. An overview of the calculation steps is given in Fig. 4.15 as in [2]. Costs $c_{q,t}$ are calculated as weighted sum of normalized cooling costs $c_c$, ageing costs $c_a$, and derating costs $c_d$ for each predicted quantile sequence and each battery cooling threshold (4.2). The weights enable the tunability of the control, with their sum defined to equal one. The resulting costs are averaged over all quantiles to $c_t$, before the battery cooling threshold with minimum total costs is chosen. In case of segmented prediction, every combination of battery cooling thresholds and according quantiles is considered for cost calculation and the threshold combination with lowest total costs is chosen. The calculation of each cost part uses the battery temperature as input sequence. Since the ML models predict the difference in battery temperature with respect to the battery temperature at the current position, the inputs for the cost parts are calculated as the sum of the (measured) battery temperature at the current position and the predicted sequence. The cost parts are further explained in the following.

$$c_{q,t} = w_c \cdot c_c + w_a \cdot c_a + w_d \cdot c_d \tag{4.2}$$

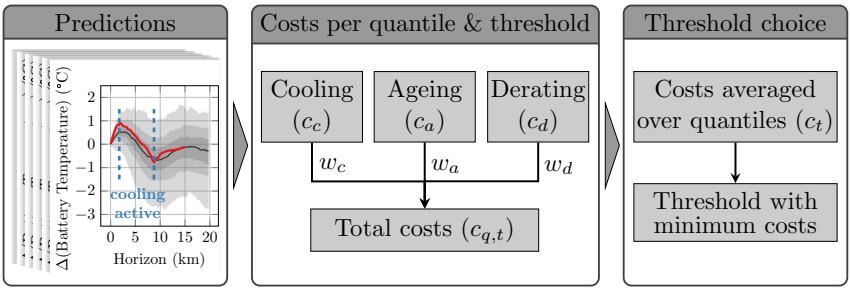


**Figure 4.15:** Calculation of total optimization costs based on three parts, taken from [2].

#### 4.3.2.1 Cooling costs

Cooling costs $c_c$ describe the energy consumption of active battery cooling. Since the prediction model does not directly predict cooling activity, the activity needs to be derived by checking at which segments the predicted battery temperature passes the corresponding thresholds. A tolerance of 0.1 °C takes inaccurate predictions into account. An example is shown in Fig. 4.16: at the second local maximum the model predicts active cooling behavior (by a predicted decrease in battery temperature) but before the actual threshold $T_{b,cool,upper}$ is reached. However, battery cooling activation is considered when the battery temperature reaches the threshold within the tolerance (at $T_{b,cool,upper} - 0.1$ °C). The resulting segments with expected active cooling are marked with a gray background.

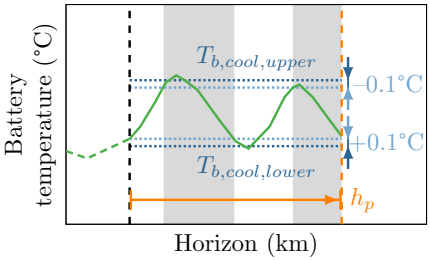


**Figure 4.16:** Example of how cooling activity (gray) is derived from a battery temperature prediction for a prediction horizon $h_p$. A tolerance of 0.1 °C is subtracted from the upper cooling threshold $T_{b,cool,upper}$ and added to the lower cooling threshold $T_{b,cool,lower}$.

The cooling costs are the sum of all segments with active cooling, divided by the number of segments within the prediction horizon and not exceeding the end of the trip. Thus, the according cost function represents an indirect computation (see 3.3.2) based on an estimation of the cooling activity. Since each cost part is calculated individually for each predicted quantile sequence and then averaged over all quantiles, the cooling costs are higher the more predicted quantile sequences reach the upper cooling threshold (within the tolerance), which means the model is more certain that battery cooling will be active.

#### 4.3.2.2 Ageing costs

Battery ageing is considered by two parts in this work: costs for the battery temperature effect on cyclic and calendaric ageing during the drive $c_{a,dr}$ and costs for the temperature effect on calendaric ageing when the vehicle is parked $c_{a,pa}$ after the end of the trip. Both cost parts only model the temperature dependency of battery ageing and do not consider the charge throughput and (drive and parking) time as input. The total ageing costs are calculated by (4.3), using weight $w_{a,pa}$ (in this work fixed to 0.2).

$$c_a = (1 - w_{a,pa}) \cdot c_{a,dr} + w_{a,pa} \cdot c_{a,pa} \tag{4.3}$$

Costs for ageing during drive $c_{a,dr}$ are calculated by a cost function directly dependent on the predicted battery temperature (4.4). The cost function is based on [88] and normalized to 1 at a battery temperature $T_b$ of 45 °C, as done in [2] and indicated by a dashed line in Fig. 4.17 (a).

$$c_{a,dr} = f_a(T_b) = \Big(0.00006121 * T_b^4 - 0.002717 * T_b^3 + 0.03095 * T_b^2 - 0.6697 * T_b + 17.57\Big)/53.52 \tag{4.4}$$

The cost function during parking considers the effect of high battery temperatures on calendaric ageing. In cases of lower ambient temperatures, the battery is expected to cool down naturally according to the temperature difference. This reduces calendaric ageing, and consequently the ageing costs are assumed to be lower. According to [2], the battery temperature change is approximated by a linear function $f_{tc}(T_b, T_{amb})$ with constant $\kappa$ (4.5). $\kappa = -0.1163$ is determined by curve fitting based on the small vehicle fleet data [2].

$$f_{tc}(T_b, T_{amb}) = \frac{\partial T_b(t)}{\partial t} = \kappa \cdot (T_b - T_{amb}) \tag{4.5}$$

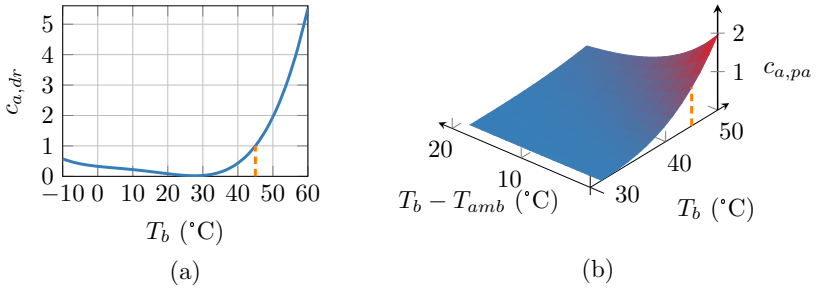


**Figure 4.17:** Cost functions for ageing during drive $c_{a,dr}$ (a), taken from [2], and during parking $c_{a,pa}$ (b), with battery temperature $T_b$ and ambient temperature $T_{amb}$. Both cost functions are normalized to 1 for $T_b$ at 45 °C.

The costs $c_{a,pa}$ are calculated only for the last segment of a trip, if it is within the prediction horizon, using the last predicted battery temperature $T_{b,pa}$ and the ambient temperature $T_{amb,pa}$ before parking (4.6). Thus, the ageing costs during parking are considered by costs of the last drive segment, representing terminal constraints (see 3.3.2). The parking time itself is not considered because it requires a prediction of the following departure time, which is not addressed by this work. $f_a$ is multiplied with an exponential function of $f_{tc}$, such that the costs are reduced for increasing $T_{b,pa} - T_{amb,pa}$ (which leads to faster cooling during parking), but stay positive (4.6). Due to the multiplication and the exponential function, the costs are normalized to 1 at a battery temperature of 45 °C and $T_{b,pa} - T_{amb,pa} = 0$, as indicated by a dashed line in Fig. 4.17 (b). If the battery temperature is lower than the ambient temperature, the costs are set to zero.

$$c_{a,pa} = \begin{cases} f_a(T_{b,pa}) \cdot \exp(f_{tc}(T_{b,pa}, T_{amb,pa})) & \text{if } T_{b,pa} - T_{amb,pa} \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

#### 4.3.2.3 Derating costs

Battery temperature based derating of the maximum available power is taken into account by derating costs. As in [2], the derating costs $c_d$ are directly calculated by (4.7) as a linear function of the battery temperature for temperatures above 40 °C. The derating costs are normalized to 1 at a battery temperature of 50 °C, as shown in Fig. 4.18 with a dashed line.

$$c_d = \frac{1}{n_p} \sum_{i=1}^{n_p} \left( \frac{T_{b,i} - 40\ °\text{C}}{10\ °\text{C}} \text{ if } T_{b,i} \geq 40\ °\text{C}, 0 \text{ otherwise} \right) \tag{4.7}$$

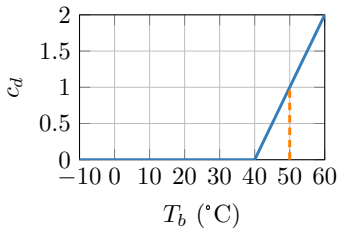


**Figure 4.18:** Cost function for derating during drive $c_d$ with battery temperature $T_b$, normalized to 1 for $T_b$ at 50 °C.

In future research, the derating costs could be extended by derating costs for cold temperatures, relevant for charging and recuperation and resulting battery heating strategies. Furthermore, an extension in future works can focus on the end of a trip, when battery charging is expected to be limited by low or high battery temperatures. Furthermore, a prediction of the battery power request could be considered to evaluate if the power limits are presumably reached [2].

### 4.3.3 Evaluation

The predictive control is evaluated according to its adaptability, tunability, robustness, and computation time by simulations as described in 3.3.3 and done in [2]. The evaluation is based on the Design of Experiments (DOE) from [2], which is included in appendix A.5. It specifies 18 scenarios with different profiles, ambient temperatures, and initial battery temperatures. The scenarios are based on the drive profiles US06 (D) [223], Artemis (Highway H, Rural R, and Urban U) [235], and Tzirakis *et al.* (T) [236]. The evaluation of the predictive control using QCNN22 from [2] serves as a base for additional comparison with predictive control using the best performing QGRU and a segmented predictive control using the QCNN22 with reduced number of history input features.

#### 4.3.3.1 Adaptability

Adaptability is evaluated for example scenarios of the predictive control with the QCNN22 from [1] with the cooling, ageing, and derating weights set to 0.08, 0.72, 0.2 respectively. Two profiles HDDUU and HDDD are simulated with the resulting temperatures shown in Fig. 4.19, which is based on [2]. The profiles only differ in the last approximately 10 km at which either an urban area without any slope is reached (HDDUU, a) or another dynamic uphill drive (HDDD, b).

For both profiles, the predictive control adapts the cooling threshold after 17.5 km from 35 °C to 40 °C to avoid battery cooling which would be otherwise activated when the battery temperature reaches 35 °C after 33 km (c and d). Consequently, the total costs for a 40 °C threshold are lower than for 35 °C at this point (e and f). After 47.5 km of the HDDD profile, the predictive control chooses a lower cooling threshold which activates battery cooling (d). The predictions of the 40 °C cooling threshold (h) show an increase in battery temperature compared to the HDDUU profile with less dynamics in the last 10 km (g). This results in higher costs for predictions with 40 °C cooling threshold than for predictions with 35 °C and 25 °C at 47.5 km (f), also when compared with the HDDUU costs (e) (shown as dotted line in f). Due to the following battery cooling and resulting decrease

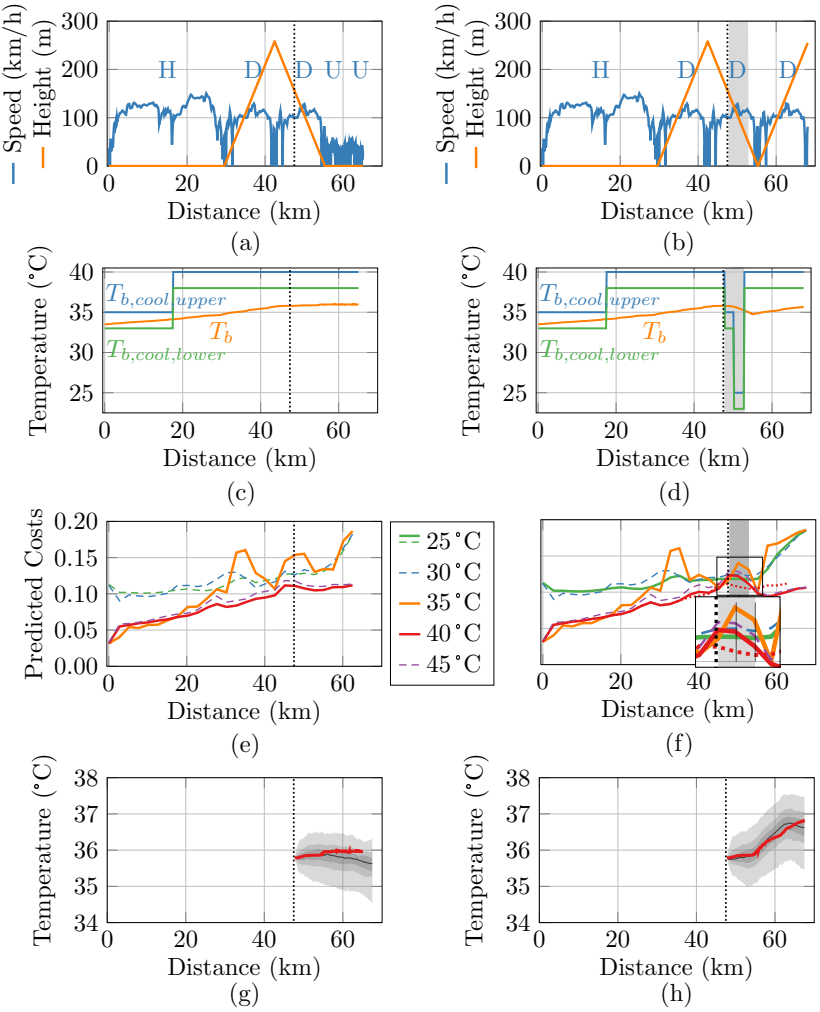


**Figure 4.19:** Simulated profiles HDDUU and HDDD as specified in appendix A.5 with an ambient temperature of 30 °C and an inital battery temperature of 33.5 °C, based on [2]. Speed and road height are shown in (a) and (b), the battery temperature $T_b$ with the upper and lower cooling thresholds $T_{b,cool,upper/lower}$ in (c) and (d). Cooling is active in the gray shaded areas. The total costs per threshold are given in (e) and (f), with the same y-axis and the costs for 40 °C from HDDUU in (f) as dotted line. (g) and (h) show the QCNN22 predictions of the control step at 47.5 km (vertical dotted line) for a cooling threshold of 40 °C, with quantiles (gray shaded) and true values from simulation (red).

in battery temperature, the costs of the predictions with 40 °C cooling threshold become the lowest again, which results in the deactivation of battery cooling at 52.5 km. A comparison with the true values in (g) and (h) shows good predictions of the QCNN22 at 47.5 km, which adapt to the different profiles at the end. The true values of (h) are taken from a simulation of the HDDD profile with a fixed threshold of 40 °C, since the cooling threshold is adapted again in the simulation of the predictive control and thus does not provide a reference.

Further analysis of the results is included in [2]. For example, the predictive control achieves a 65 % reduced cooling time and a 1.02 % reduced total energy consumption, compared to a fixed threshold of 35 °C. The adaptability analysis indicates in this example that the QCNN22 and the predictive control are able to adapt their predictions and resulting cooling threshold choice to different routes ahead.

#### 4.3.3.2 Tunability and robustness

Tunability evaluation considers the effect of a variation of the cost weights on the resulting cost parts. This work focuses on a variation of the cooling weight $w_c$ and ageing weight $w_a$ and the according cooling costs $c_c$ and ageing costs $c_a$. The results of the DOE with weight variation are shown in Fig. 4.20, based on [2]. A variation of cost weights of the predictive control with QCNN22 (blue) and QGRU (green) results in plausible change of the resulting costs in the shape of a Pareto front, see Fig. 4.20 (a). For example, an increase in the cooling weight $w_c$ (i.e. higher consideration of cooling energy consumption) results in reduced cooling costs but increased ageing costs. Compared with the costs for fixed thresholds, both Pareto fronts of the predictive control are closer to the theoretical optimum of zero costs.

The QGRU, which also performs better than the QCNN22 in the prediction model comparison, further improves the predictive control performance. For example, the QCNN22 predictive control reduces the cooling costs by 9 % with unchanged ageing costs compared to the 35 °C fixed threshold (Fig. 4.20 (b)), as also analyzed

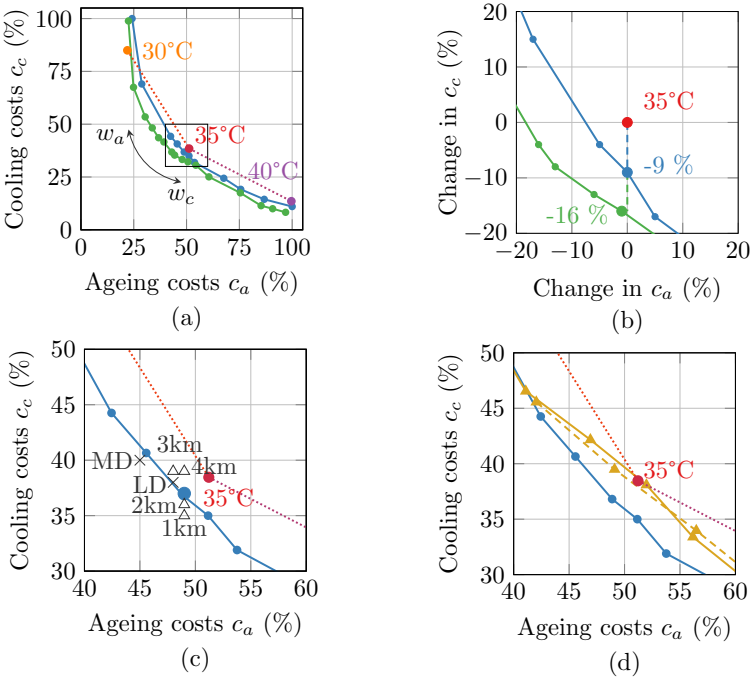


**Figure 4.20:** Normalized cooling and ageing costs for fixed thresholds (30 °C, 35 °C, 40 °C) and predictive control with QCNN22 (blue) from [2] and with QGRU (green). The arrow indicates an increased ageing weight $w_a$ or cooling weight $w_c$. (b) shows the cost difference compared to a 35 °C fixed threshold. (c) and (d) focus on the range of the rectangle in (a). (c) shows the QCNN22 predictive control with a varied control step (1 km to 4 km) and added input noise for overestimation (MD) and underestimation (LD) of the profile dynamics, with the same weights than the larger blue dot, taken from [2]. (d) includes results of a segmented predictive control using the QCNN22 with less history input features and one prediction shift (solid gold line with triangles) and the same model without shifts (dashed).

in [2]. The QGRU further reduces the cooling costs to 16 % less than for the 35 °C fixed threshold. The total energy consumption is reduced by 0.1 % with the QCNN22 predictive control and by 0.2 % with the QGRU predictive control. For both models, derating costs are the same than for the 35 °C fixed threshold.

Robustness of the QCNN22 predictive control is investigated as in [2]. The effect of a varied control step size and input foresight noise on the costs is included in Fig. 4.20 (c). The control step size is varied between 1 km and 4 km, compared to the chosen 2.5 km. Input noise is applied using Gaussian noise on the speed profile and height profile for a more dynamic input than actual (MD). A moving average of the foresight speed profile and the same Gaussian noise of the height profile are applied for a less dynamic input than actual (LD). The parameters of the applied Gaussian noise and moving average are included in appendix A.5. For all variations, the resulting costs are closer to the theoretical optimum of zero costs than for the fixed 35 °C cooling threshold. In this analysis, smaller control step sizes decrease the cooling costs, all other variations result in higher cooling costs, but also lower ageing costs.

Results with segmented predictive control using the QCNN22 with less history input features are included in Fig. 4.20 (d). For comparison, the same prediction model (with less history input features) is used with the predictive control without segmented predictions. In both cases, the Pareto front is worse than for the QCNN22 predictive control. In this example, the removal of the history input features decreased the performance (dashed line with triangles), which could not be improved again by segmented predictive control (solid line with triangles).

The effect of the QCNN22 and QGRU predictive control (without segmented predictions) on the total energy consumption is further analyzed by Fig. 4.21. It shows the change in total energy consumption for each of the 18 DOE scenarios for the QGRU and QCNN22. While the energy consumption is reduced on average by 0.2 % with the QGRU predictive control, the change in energy consumption differs between -0.9 % and 1.0 % over the scenarios. In comparison, the QCNN22 predictive control results in less change in energy consumption.

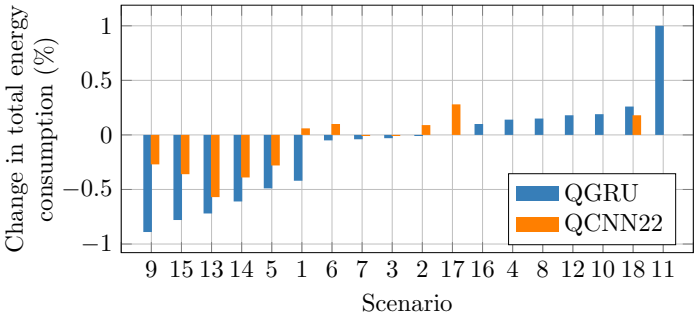


**Figure 4.21:** Relative change in total energy consumption of the QGRU and QCNN22 predictive control compared to a fixed threshold of 35 °C, sorted by the results with the QGRU. The scenario numbers are mapped to Table A.2 in appendix A.5.

Amongst the scenarios with larger reduction are the profiles (13-15) with large vehicle dynamics and elevation. The scenario with largest reduction (9) and with largest increase (11) with the QGRU predictive control are both the profile without elevation and few dynamics (HRUU), only differing in the ambient temperature. The energy consumption is also increased for the other two scenarios with the HRUU profile (10, 12). The difference of the predictive control between scenario 9 and 11 is an unnecessary cooling activation at the beginning of the drive for scenario 11. Thus, further optimization could target the case of prediction and control at the beginning of a trip when no history input data is available. Except for scenario 11, the other six cases with higher energy consumption exhibit only small increases of up to 0.26 %, while the best six cases show a reduction of more than 0.40 %. Overall, the analysis shows that the total energy consumption reduction depends on the scenario. Therefore, an evaluation of the predictive control performance in later application requires a representative DOE, taking (regional-dependent) environmental conditions and driver behavior into account.

In an additional analysis, the time until first active battery cooling and the total cooling duration are compared between the 35 °C fixed threshold and the predictive control with QGRU and QCNN22. Fig. 4.22 depicts the differences between the predictive control and the 35 °C fixed threshold control for all 18 scenarios.

According to Fig. 4.22 (a), the QGRU predictive control activates cooling earlier by at least 1000 s in six scenarios, and at maximum 250 s later in four scenarios. In comparison, the QCNN22 predictive control activates cooling less early in the same six scenarios than the QGRU. Cooling duration (Fig. 4.22 (b)) is closer to the 35 °C fixed threshold with the QGRU than with the QCNN22. The QCNN22 predictive control shows greater variation with scenarios of both shorter and longer battery cooling. On average over all scenarios, the cooling duration is shorter for both the QGRU (by 87 s) and the QCNN22 (by 49 s), resulting in lower cooling costs and energy consumption. Fig. 4.22 (c) shows a difference in the number of cooling events between the QCNN22 predictive control and the fixed 35 °C threshold in only two scenarios, while the QGRU predictive control leads to more (but shorter) cooling events. However, dependent on the system properties, a more frequent cooling activation might cause more stress for the thermal management components.

The cooling analysis shows that the QGRU predictive control results in earlier but shorter cooling than the 35 °C fixed threshold control. Early cooling activation reduces the battery temperature early in the drive, such that cooling duration can be reduced in order to achieve the same ageing costs. In this way, battery cooling costs are reduced and thus the total energy consumption. However, if the weight for ageing costs during parking is increased, later cooling might be more favorable to achieve a lower battery temperature at the end of the drive, in case the temperature is expected to rise significantly during the drive. Considering the dependency of the results on the weights and scenarios, this analysis needs to be conducted with a representative DOE and the corresponding optimal weights for application. Furthermore, simple rules (for the given example an early cooling activation) can be derived from the analysis.

#### 4.3.3.3 Computation time

Computation time for the predictive control step is calculated based on the DOE simulations with the set of weights which leads to costs closest to the theoretical optimum of zero costs. The computation times of prediction and threshold choice

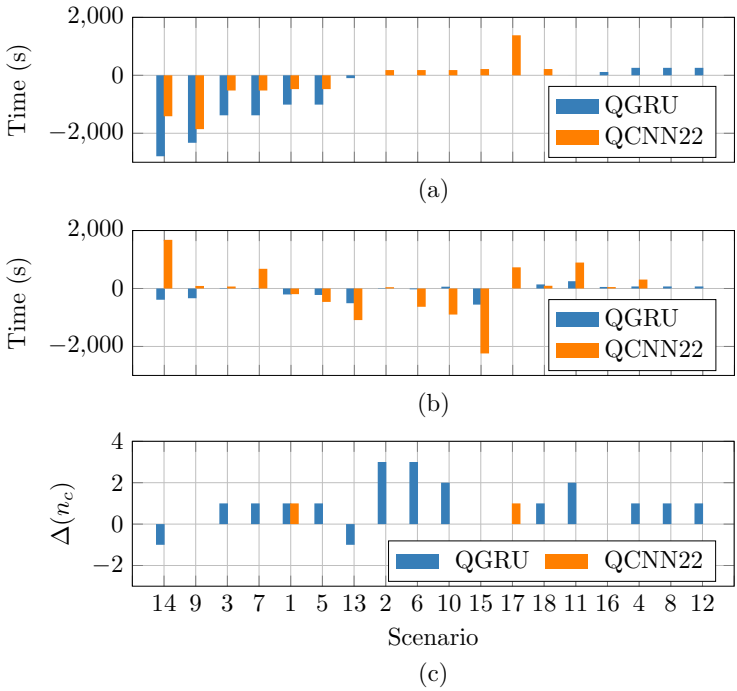


**Figure 4.22:** Difference in time until first cooling event (a), total cooling duration (b), and difference in number of cooling events $n_c$ (c) of the QGRU and QCNN22 predictive control compared to a fixed threshold of 35 °C. The scenario numbers are mapped to Table A.2 in appendix A.5 and are the same for (a), (b), and (c), sorted by the time difference of the QGRU in (a).

are compared in Table 4.9. The simulations are conducted with the same hardware than in [2][1]. The comparison includes the QCNN22 predictive control from [2], a predictive control with the QCNN22 with reduced history input features both with and without segmented prediction as well as the QGRU predictive control. The shown times are the sum of average prediction time and average time for threshold choice (average time) as well as the sum of maximum prediction time and maximum time for threshold choice (maximum time). The segmented predictive

[1] Intel i7 3.0 GHz processor, 16 GB RAM, Python 3.7.5, TensorFlow/Keras 2.5.0

control takes around 17 times longer for the average time (29.38 s) and 6 times longer for the maximum time (34.36 s) of predictive control. This can be converted into the speed in km/h which would be needed to drive 250 m within that time. Accordingly, if a vehicle is driving faster than 31 km/h (maximum time: 26 km/h), the segmented control step takes longer than the time needed to pass 250 m (which is one step of the horizon), resulting in a delayed threshold update. The computation times of the other three predictive control models are in a similar range, for which delayed threshold updates only occur in rare cases of high speeds and maximum predictive control time.

**Table 4.9:** Time needed for prediction and threshold choice, and speed in km/h which would be needed to drive 250 m within the according time. Predictive control is simulated for the DOE using the QCNN22, QCNN22 with reduced number of history features (with and without segmented predictions) and the QGRU, with the weights resulting in the cooling and ageing costs closest to the 35 °C fixed threshold (for comparable cooling strategies in this analysis).

| Model | Average Time (s) | Maximum Time (s) | Speed for 250 m (Average) (km/h) | Speed for 250 m (Maximum) (km/h) |
|---|---|---|---|---|
| QCNN22 | 1.59 | 7.23 | 567 | 125 |
| QCNN22 reduced | 1.83 | 4.19 | 491 | 215 |
| QCNN22 reduced segmented | 29.38 | 34.36 | 31 | 26 |
| QGRU | 1.87 | 6.27 | 480 | 143 |

#### 4.3.3.4 Example of control with real trip data

The QGRU predictive control is simulated with recorded data from a trip in Naples (Italy), taken from [237], to analyze the control behavior for real trip data. The ambient temperature and initial battery temperature are set to 34 °C, which

corresponds to a hot summer day in Naples[2]. Fig. 4.23 shows the results for simulations with a fixed 35 °C cooling threshold and with the QGRU predictive control. The trip starts with a drive up and down a hill (with a height difference of 425 m), followed by a rural and highway section, the latter with speeds up to 123 km/h. The total trip distance is 66.5 km.

During the uphill and highway parts, the battery temperature increases. When it reaches 35 °C after 53.1 km, battery cooling is activated with the fixed 35 °C cooling threshold, and deactivated 456 s later when it reaches 33 °C at the end of the trip (Fig. 4.23 (c)). In comparison, the QGRU predictive control activates cooling after 2.7 km, at the beginning of the uphill drive (Fig. 4.23 (d)). Furthermore, the cooling duration is reduced by 45 s.

The early cooling activation of the QGRU predictive control reduces the battery temperature from the beginning, such that it stays at a lower level for most of the trip compared to the fixed 35 °C threshold cooling strategy. Thus, the battery temperature is on average 0.9 °C lower and the maximum temperature is 1.7 °C lower for the QGRU predictive control. As a consequence, the ageing costs are 32 % lower. However, at the end of the trip, the battery temperature is 0.5 °C higher. In case of a higher weight on ageing during parking, ageing costs might be similar or worse with an early cooling activation.

Due to the shorter cooling, the QGRU predictive control reduces the cooling costs by 10 % and the total energy consumption by 0.12 %. As result, the QGRU predictive control strategy of early but shorter battery cooling corresponds to the cooling analysis of the 18 DOE scenarios in 4.3.3.2. However, energy consumption might be increased compared to the fixed 35 °C threshold if the trip ends earlier than 53.1 km. The end of the trip is considered by the predictive control if it is within the prediction horizon, which is 20 km for the developed models. A possible solution is a larger QGRU horizon or a predictive control with segmented predictions (which can evaluate larger horizons as well).

[2] The maximum ambient temperature in Naples can reach 35.9 °C and the average daily maximum 27.5° C, according to measurements from 2016 [238].

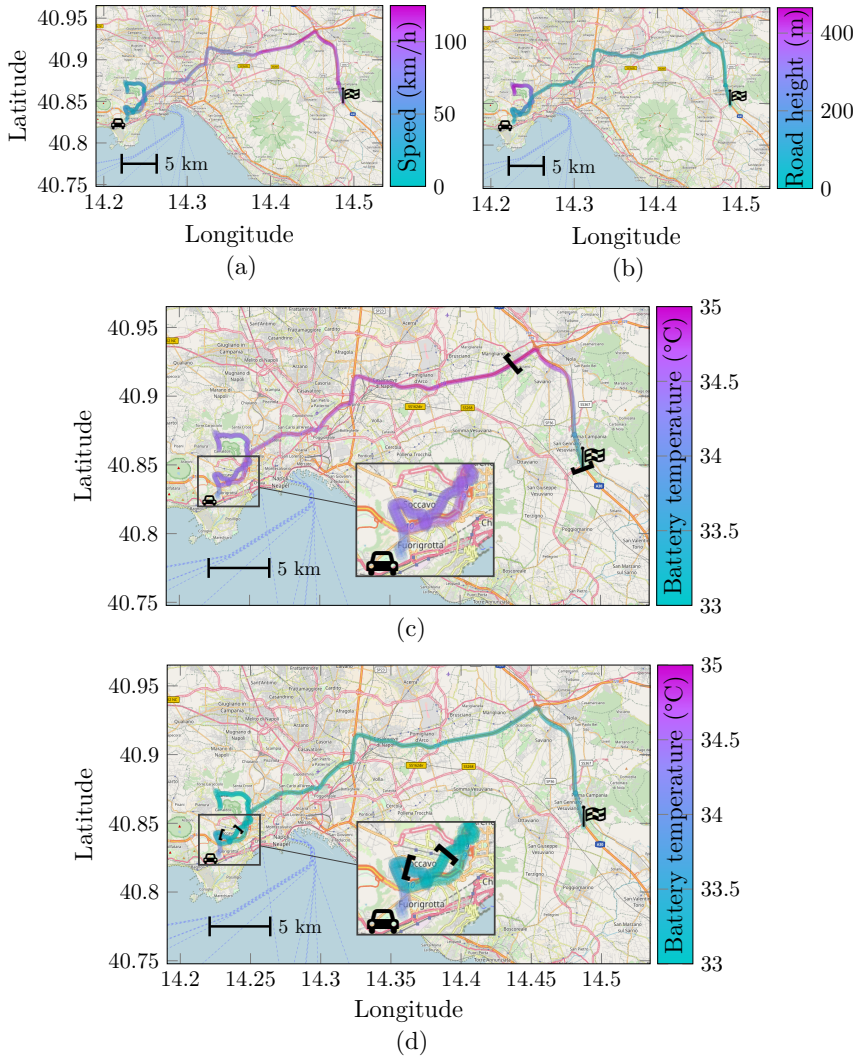


**Figure 4.23:** Speed in km/h (a) and altitude in m (b), as well as battery temperature in °C and battery cooling activity (active between brackets) for a 35 °C fixed threshold (c) and QGRU predictive control (d). The simulated profile starts at the car icon and consists of the hill, rural, and highway profile (in that order) in Naples from [237]. Map: ©OpenStreetMap contributors.

### 4.3.4 Rule-based strategies

Based on the Q*NN predictive control, rule-based strategies are developed as an alternative control with fewer computational effort in real-world application. The rule-based models are trained with simulation results of a predictive control using Q*NN prediction models. The DOE is extended to in total 100 simulations with additional combinations of the same cycles and more boundary conditions than in appendix A.5. As a result, the simulations cover a total distance of 6,715 km, with ambient temperatures ranging between 15 °C and 35 °C, the initial battery temperature between 20 °C and 40 °C, and the absolute slope $|\theta|$ of segments with elevation between 20 and 40 m/km. The simulations are conducted with the QCNN22 predictive control with cooling weight 0.04 and ageing weight 0.76 (the big blue dot in Fig. 4.20 (c)).

The rule-based strategies are derived as Shallow Decision Tree (SDT) classifier. In total, eight input features are used: mean value and variance of speed, mean value and variance of speed difference (between consecutive segments), mean value and variance of height difference (between consecutive segments), as well as the current ambient temperature and current battery temperature. The output is a binary classification for cooling off (0) or on (1). The input features are processed for four foresight horizons (5 km, 10 km, 15 km, 20 km), the output (and training label obtained by the Q*NN predictive control) is the activation or deactivation of battery cooling at the current position. All data are split into training and testing data with a ratio of 80:20, with the same share of cooling on (15 %) and cooling off (85 %) in both data sets. Afterwards, the training data are further split into training and validation data in a ratio of 80:20. In order to handle data imbalance during training, random undersampling is applied on the remaining training data, such that the share of the minority class (cooling on) is increased to 25 % (and the majority class with cooling off reduced to 75 %). The validation and testing data do not need to be balanced, because model evaluation includes the F1-score, which considers imbalanced data.

The SDT is limited in maximum tree depth and number of leaves to ensure models with low complexity. Optimization of these two hyperparameters is done

by grid search individually for each of the four horizons and with the same ranges than for the SDT battery temperature prediction model (see 4.2.2.2). The hyperparameters with highest accuracy on the validation data are a maximum depth of 5 and maximum number of leaves of 17 (5 km), 8 and 17 (10 km), 7 and 17 (15 km), 5 and 9 (20 km).

Impurity feature importance of the four SDT models is shown in Fig. 4.24. The current battery temperature $T_b$ has the highest importance for all four models. The importance of the other features differs between the horizons and does not exceed 10 %. On average, the variance in speed changes $\mathrm{var}(\Delta v)$ has the second highest importance and the variance in height difference $\mathrm{var}(\Delta h)$ has the lowest importance. The high feature importance for $T_b$ is plausible considering the impact of the current battery temperature on the optimization costs. However, the imbalance in feature importance can be also related to the low complexity of the SDT models, which limits their ability to model the patterns of the predictive control. Thus, further improvements could address the imbalanced feature importance by different model design.

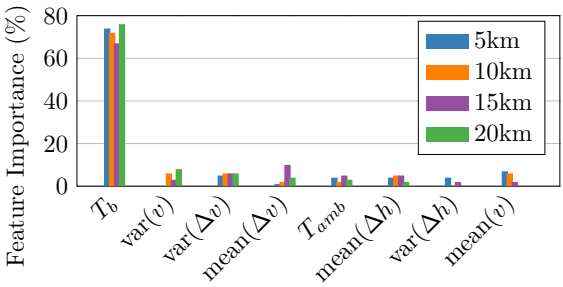


**Figure 4.24:** Impurity feature importance of the Shallow Decision Trees (SDT) for for rule-based strategies, sorted by importances from the SDT for 20 km.

The SDT models with input features of four different horizons are composed to an ensemble SDT classifier with an adjustable voting threshold, as described in 3.3.4. Four different thresholds are considered, such that battery cooling is

**Table 4.10:** Metrics of the Shallow Decision Trees (SDT) for each horizon and the ensemble SDT with different voting thresholds. The test data has a cooling share of 15 %.

| Model | 5 km | 10 km | 15 km | 20 km | 1/4 | 2/4 | 3/4 | 4/4 |
|---|---|---|---|---|---|---|---|---|
| F1-Score | 0.44 | 0.54 | 0.51 | 0.52 | 0.49 | 0.49 | 0.48 | 0.42 |
| Accuracy | 0.88 | 0.85 | 0.84 | 0.87 | 0.79 | 0.82 | 0.85 | 0.87 |
| Cooling share | 7 % | 18 % | 17 % | 13 % | 26 % | 20 % | 13 % | 8 % |

activated if at least one, two, three, or all four SDT classifiers predict cooling on. The classification performance is evaluated by F1-score, accuracy, and the share of cooling off to cooling on for the testing data. Table 4.10 shows the results for each of the single SDT classifiers and the ensemble SDT with each of the voting thresholds. Accuracy is comparably good for all models (between 0.79 and 0.88), while the F1-score (between 0.42 and 0.54) indicates bad performance, which considers the data imbalance. The share between cooling off and on indicates if, for example, a model results in a more conservative cooling strategy with more cooling thus less ageing costs. For the ensemble SDT, a lower voting threshold (1 of 4) results in more cooling than a higher voting threshold (4 of 4), as long as the classification performance is limited for all models. Thus, the voting threshold can be used for tunability of the rule-based control, similar to the cooling and ageing weights from Q*NN predictive control.

Predictive control is simulated for the DOE from 4.3.3.2 using the SDT classifiers of the four horizons and the ensemble SDT classifier with four voting thresholds to determine battery cooling activation. The resulting cooling and ageing costs are shown in Fig. 4.25, together with the costs for the QCNN22 and QGRU predictive control. The costs are normalized by the same maximum values from the tunability evaluation (4.3.3.2) of the Q*NN predictive control for better comparison. Tunability of the ensemble SDT is given by the voting threshold (VT), which decreases ageing costs but increases cooling costs for a lower VT. The ensemble SDT performs well for a VT with cooling activated if at least one of four models votes for active cooling. The resulting costs are below the Pareto front of the

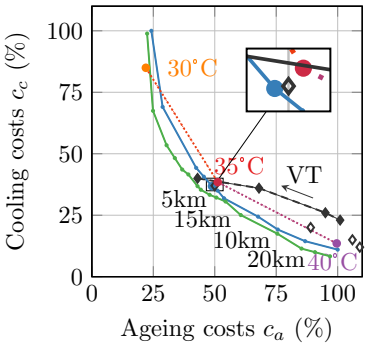


**Figure 4.25:** Normalized costs from predictive control using QCNN22 (blue), QGRU (green), fixed thresholds, the rule-based control with the SDT models for different horizons (single, hollow diamonds), and with the SDT ensemble for different voting thresholds VT (connected, filled diamonds). The arrow points into the direction of a decreasing VT (the most left diamond is VT 1 of 4). The magnifier is centered at the 5 km SDT control (hollow diamond), with the costs from the 35 °C fixed threshold (red) and the QCNN22 predictive control that the SDT models are derived from (big blue dot) next to it.

QCNN22 predictive control (see the filled diamond in Fig. 4.25, not the hollow diamond in the magnifier). Compared to the 35 °C fixed threshold strategy, cooling costs are increased by 4 % but ageing costs are reduced by 16 %. However, the costs also differ from the QCNN22 costs with the weight set that the rule-based models are derived from (big blue dot in magnifier in Fig. 4.25). In contrast, the single SDT classifier with 5 km horizon results in closer costs (hollow diamond in the magnifier in Fig. 4.25). Its cooling and ageing costs are each 1 % higher than for the respective QCNN22 predictive control, but the SDT still reduces the costs by 3 % each compared to the 35 °C fixed threshold control.

The number of parameters can be determined as done for the rule-based predictions. The SDT models for the different prediction horizons 5 km, 10 km and 15 km each consist of 16 if-conditions (20 km: 8) and 17 leaves (i.e. different outputs, 20 km: 9), which results in 33 parameters per model (20 km: 17) and, together with the voting threshold, in 117 parameters for the ensemble SDT. In comparison, the QCNN22 predictive control needs 7,128,228 parameters for the prediction model (QGRU: 6,396,924; see both numbers in 4.2.2.1), additionally

to 12 parameters for cost calculation and the 4 cost weights. Furthermore, the SDT classifiers only need up to 8 input features each (28 for the SDT ensemble), while the Q*NN models need 52 (QCNN22: 60). The ensemble SDT with a voting threshold 1 of 4 needs 1.15 s (Average Time) and 3.63 s (Maximum Time) in total for the control step. The single SDT classifier for 5 km needs 1.06 s (Average Time) and 3.30 s (Maximum Time). Thus, the rule-based models are approximately 40 % faster than the Q*NN and QCNN22 predictive control. Consequently, the single SDT and the ensemble SDT are a suitable alternative for integration in the target hardware of real-world applications with stricter computational requirements (e.g. the ECU of a vehicle).

## 4.4 Assessment of applicability

The method from chapter 3 for a predictive control has been applied to a battery thermal management of BEV. The results show how fleet data and foresight data can be used to optimize battery cooling strategies, as specified in the problem statement. For both the prediction models and the predictive control, simple models are developed as reference and for real-world application with less computational effort. The developed models and controllers differ in complexity, accuracy, and information (prediction uncertainty, tunable cost weights), as described in the following. For real-world application, the most suitable model and controller must be selected according to the computational effort and desired accuracy.

Q*NN models provide accurate battery temperature predictions for the given test data. The predicted quantile sequences additionally contain information about the prediction uncertainty. The Q*NN models are the best choice out of all the models developed in this work, if a later application requires precise predictions and the prediction uncertainty, and if the computational requirements allow the usage of large prediction models. The prediction of quantile sequences provides the following advantages:

1. The Q*NN can model complex situations with large uncertainty by providing a quantile corridor in which the true values most likely will be.
2. The development of Q*NN is independent from assumptions about the underlying distribution, since quantile sequences are not limited to a specific probability distribution (e.g. the normal distribution).
3. Information about the prediction uncertainty can be used to increase awareness for inaccurate predictions in complex situations (e.g. larger differences between quantiles means larger model uncertainty), which can increase the overall trust in the prediction model.
4. The quantile sequences can be evaluated independently in a predictive control, which improves flexibility and robustness in the cost calculation for choosing the battery cooling strategy.

In contrast to that, simpler prediction models can be selected if the computational requirements are strict, if quantile sequence predictions are not needed, and if the reduction in prediction accuracy is acceptable. However, the experiments showed a significant loss in prediction performance compared to the Q*NN models. Furthermore, Transfer Learning of Q*NN models enables the application of battery temperature prediction models for vehicle models for which only few data are available. An analysis of Transfer Learning with different amounts of data from a second vehicle model provides information about the required data set size to achieve the needed prediction accuracy.

The developed predictive control adapts the battery cooling threshold based on the Q*NN predictions for different thresholds and resulting optimization costs. Battery specific cost functions and cost weights allow an interpretable tuning of the predictive control dependent on the target application. For instance, if battery ageing needs to be reduced and an increase in cooling energy consumption is accepted, the corresponding cooling and ageing weights can be adjusted to shift the costs along the Pareto front. Robustness to input noise during application is analyzed for cases of inaccurate foresight input data from the navigation system. If the

computation resources of the target system (ECU in the vehicle) are limited, alternative rule-based strategies can be implemented as Shallow Decision Tree (SDT) classifiers or an SDT ensemble classifier with a voting threshold. These rule-based models consist of significantly less parameters than the Q*NN predictive control and can be translated into simple functions and look-up tables. However, their tunability is limited and their performance can be insufficient regarding the resulting costs. Furthermore, the rule-based models are derived from simulations of the Q*NN predictive control for a specific Design of Experiments (DOE), such that the DOE needs to be representative for the scenarios of the application, i.e. the typical driving scenarios and behavior.

Overall, the application of the proposed method provides models with different levels of complexity, suiting different requirements for application in a battery thermal management system. The applicability of each prediction model and predictive control depends on these requirements and can be evaluated with this assessment.

# 5 Discussion

The developed method is able to answer the research questions raised in 2.4, according to the results from chapter 4. Deep Learning based Quantile Neural Networks (Q*NN) predict battery temperature sequences and provide information about the prediction uncertainty. The prediction uncertainty is given by quantile predictions, such that it is not limited to a specific probability distribution function. Data processing includes data balancing with sequence clustering of large fleet data and data augmentation of simulation data. The Q*NN are trained with both the simulation and fleet data, using history and foresight input data. The Q*NN provide better prediction performance than the reference prediction models. Transfer Learning enables Q*NN training with good prediction accuracy for a new vehicle model even when limited data are available. The novel predictive control of a battery thermal management uses the Q*NN and cost functions that represent battery-related properties (cooling energy consumption, battery ageing, and power limitation). The Q*NN predictive control results in lower Pareto costs than a fixed 35 °C cooling threshold, with a reduction of cooling costs (by on average 16 % over 18 scenarios) and total energy consumption (0.2 %) for unchanged ageing costs. Rule-based strategies can be derived using Shallow Decision Trees, resulting in less energy reduction but computationally less expensive integration in the vehicle. Finally, an assessment of applicability provides an approach for choosing the best suitable prediction model and predictive control according to real-world requirements.

The results are further discussed with comparison to the state of the art in 5.1 and concerning method limitations with possible adjustments in 5.2.

## 5.1 Comparison with state of the art

The results of the Q*NN for battery temperature prediction and Q*NN predictive battery thermal management are compared with the state of the art described in chapter 2. The battery temperature prediction results are discussed in 5.1.1, followed by the predictive control in 5.1.2.

### 5.1.1 Prediction model

According to the results, the method can be applied to the development of accurate prediction models for quantile sequences of battery temperatures. A comparison with single-step reference models indicates the superior prediction performance of the multi-horizon Q*NN models, in accordance with the results in [194]. Similar to [195], the GRU-based architecture performed better than the CNN and LSTM. The RMSE is higher for the Q*NN models compared to [195], which may be related to more dynamic battery temperature changes in the test data of this work. Furthermore, both [194] and [195] focus on training and test data of battery cell measurements, while the Q*NN models are trained and tested on a combined data set of vehicle simulations and a large vehicle fleet. In contrast to the methods in 2.3.1, data augmentation with Intersection-based Assembly (IBA) does not distort data or generate completely synthetic data, but combines existing data taking physical properties into account.

The Q*NN prediction performance is improved with larger data size in combination with improved data processing (balancing and augmentation, see the comparisons with QCNN22). The positive correlation of available data size and prediction performance is further shown in the Transfer Learning investigations (4.2.4), which corresponds to the findings of [25, 102]. However, in contrast to [102], the deeper Q*NN architecture has not been picked by the efficient Bayesian hyperparameter optimization. This is in accordance with [24, p. 215–216], who states that time-series forecasting CNNs with one dimensional convolutions usually have fewer hidden layers than CNNs with two or three dimensional convolutions.

Moreover, the Q*NN architecture distinguishes from [194, 195] considering the additional input channel for foresight input data, the output channels for quantile sequences, and the custom loss function.

### 5.1.2 Predictive control

The developed predictive battery thermal management can successfully use the quantile predictions of the Q*NN models to improve the Pareto front of cooling and ageing costs compared to a fixed threshold control. In the references in 2.2.2, predictive control leads to better performance than a fixed threshold control. Results in [67, 23] show a Pareto front between cooling energy and battery ageing. Differences in the achieved energy consumption reduction can be based on different driving scenarios, vehicle types ((P)HEV instead of BEV) [64, 23, 17], and reference cooling thresholds [16, p. 121]. Furthermore, in contrast to [87, 88, 64], battery cooling is considered as a binary state (cooling on or off) in Q*NN predictive control, without an adjustable cooling power. Thus, more energy consumption reduction might be possible when including an adjustable cooling power, which requires according simulation models and fleet data collection. For example, an adjustable instead of constant compressor speed reduces the compressor energy consumption by 11 % in [67].

As an extension to the references in 2.2.2, the cost functions of the Q*NN predictive control additionally include power derating costs and calendaric ageing costs during parking (as terminal costs). Moreover, the Q*NN predictive control evaluates the costs based on distance-based, multi-horizon quantile predictions for five different thresholds. The maximum prediction horizon of the references in 2.2.2 is 60 s [89, 23, 68, 17], which can be converted to 1.7 km for a constant speed of 100 km/h. In comparison, the prediction horizon of the Q*NN predictive control is larger with 20 km (80 steps of 250 m). Thus, the Q*NN predictive control can adapt the battery cooling strategy to information from larger distances of the route ahead, for example, if an urban area or a dynamic uphill drive are expected

after a highway section. Despite the larger prediction horizon, the computation times of the Q*NN predictive control are comparable to the references in 2.2.2.

## 5.2 Limitations

The performance of the Q*NN prediction model and predictive control are linked to the given vehicle model, collected simulation and fleet data, and the scenarios of the Design of Experiments (DOE). In this context, predictive control with the more accurate QGRU improves the Pareto front compared to the QCNN22, which points out the benefit of the increased prediction model accuracy. However, the average reduction in total energy consumption (0.2 %) remains low compared to the 16 % cooling energy reduction. It depends on the drive scenarios and the according energy consumption of the drivetrain as well as the share of the corresponding thermal management components. For example, 16 % cooling energy reduction results in a reduction of total energy consumption by 1 % when a share of 6 % is assumed, as in 2.1.1.3. Additionally, the derivation of rule-based strategies in 4.3.4 highly depends on the DOE. Furthermore, the predictive control does not operate when the vehicle is stopped (due to distance-based sampling) [2]. While shorter stops in traffic jams are not expected to significantly change the battery temperature, thermal management during fast charging is an important use-case which is not covered by the predictive control, yet. Another aspect is the dependency on information about the route ahead. However, the predictive control shows good robustness to foresight input noise in 4.3.3.2. Moreover, the method steps need to be individually evaluated concerning their effect on the results, for example how the novel data augmentation approach improves the prediction model performance.

Besides these limitations, the developed method could be extended in order to improve the performance and applicability of the prediction model and the predictive control. For example, the Q*NN prediction accuracy is expected to improve when using simulation or fleet data with battery cooling thresholds changing during the same trip. Future Q*NN architectures might use more advanced sequence layers,

such as bi-directional RNN, with self-attention, or Transformer models. Bayesian optimization could be extended to larger ranges of hyperparameters, architecture modifications, and data processing. Furthermore, the prediction models might be tested live in a vehicle, with speed prediction models from the navigation system as foresight input data. Tests on the vehicle hardware assess the potentials and limitations of the corresponding implementation and integration. Additionally, continuous training of the Neural Networks could be investigated in order to fine-tune the models to the individual driver behavior or environmental conditions [1]. However, in this case, either higher on-board computational power or cloud-computing are required for model training. Further research could address model explainability. Investigations of Transfer Learning need to include additional simulation data of the second vehicle model for fine-tuning and evaluation considering different battery cooling thresholds.

The Q*NN predictive control design could be extended by adjustable prediction and control horizons, according to the drive scenario (e.g. longer horizons for highways, shorter horizons for cities) [2]. Moreover, an adjustable, continuous cooling power could be used for control output and cooling cost calculation. Battery ageing and derating might be more accurately represented by the cost functions if the SOC, SOH, as well as drive and parking time are included. Battery derating costs could be extended by a prediction of actual power demand [2]. Furthermore, the Q*NN predictive control could be compared with other MPC implementations and with global optimization methods (e.g. Dynamic Programming). Similar to the discussed extensions for the prediction model, the predictive control might be integrated and tested in the vehicle. The cost weights could be continuously tuned during application according to the driver behavior and environmental conditions [2], similar to Reinforcement Learning methods as proposed in [7]. The development of rule-based models might be improved, using larger data sets of the best performing Q*NN predictive control and further optimization of model type, hyperparameters, and input features.

# 6 Conclusion

This chapter concludes the dissertation with a summary of the method and its key results and findings in 6.1. An outlook provides further remarks for future research and potential implications in 6.2.

## 6.1 Summary

The automotive industry enters a new era with the rise of Battery Electric Vehicles (BEV) in order to reduce emissions. An energy consumption reduction of BEV contributes to well-to-wheel emission reduction as well as increased driving ranges. However, electric vehicle batteries require an energy-consuming thermal management due to their temperature dependent properties such as battery ageing and power derating. Thus, an energy-efficient battery thermal management control is developed in this dissertation.

The developed method includes a battery temperature prediction model and a predictive control which uses the model for an adaption of battery cooling thresholds. The prediction model is a Deep Learning based Quantile Neural Network (Q*NN). Its architecture considers both history inputs and foresight inputs to predict quantile sequences of the battery temperature change. The data-driven development processes simulation data (971 simulations) and fleet data (225,629 trips). The simulation data are augmented with Intersection-Based Assembly (IBA). Furthermore, the fleet data are balanced based on a sequence clustering. The Q*NN is trained and evaluated with different sequence architectures such as Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), and Long Short-Term

Memory (LSTM). Moreover, the resulting Q*NNs are compared with reference regression models including Shallow Decision Tree (SDT), Light Gradient Boosting Machine and Extra Trees Regressor (LGBM-ETR), as well as Quantile Extra Trees Regressor (QETR). The Q*NN with GRU (QGRU) achieves best performance for point-prediction (median) metrics (RMSE: 0.66 °C, $R^2$: 0.84, both as average of the four evaluated horizons) and quantile-related metrics (e.g. with 98.87 % of the true values below the 0.99 quantile). Furthermore, Transfer Learning of the QGRU to another vehicle model provides accurate prediction performance even when only few data are available and reduces training time for large datasets.

The best performing prediction model (QGRU) is used in a predictive control of the battery thermal management. The battery cooling thresholds are adapted according to a cost function considering battery cooling, ageing, and derating with tunable cost weights and the predicted quantile sequences as input. Thus, the QGRU predictive control can adapt the cooling strategies according to the route ahead. Simulations of 18 scenarios show a reduction in cooling energy consumption on average by 16 % with corresponding ageing costs compared to a fixed 35 °C threshold control. Simpler rule-based models are derived from simulations of a Q*NN predictive control and achieve comparable performance with lower computational requirements.

In a nutshell, the main contributions of this dissertation are:

- A method for the development of Deep Learning based quantile sequence prediction models with history and foresight inputs and using simulation and fleet data, as well as their integration in a predictive control.
- The application of the developed method for an improved battery thermal management of Battery Electric Vehicles (BEV).
- Assessment of real-world requirements with further investigations of Transfer Learning of the prediction model and the derivation of rule-based models of the predictive control.

In conclusion, this dissertation describes the data-driven development of a predictive control method, which improves the energy efficiency of a battery thermal management of BEV.

## 6.2 Outlook

The developed method could be improved and extended, addressing the limitations discussed in chapter 5. For example, further improvements in data collection and processing might provide a more representative, balanced, and clean data set for training and evaluation. Optimization of model architectures could be investigated for better performance concerning accuracy and computational requirements. Furthermore, future research could aim at more adaptability of control values, cost functions, and horizons of the predictive control. The evaluation of the predictive control might be based on more (representative) driving scenarios. Additionally, alternative (global) optimization methods could be investigated for further reference. Both the prediction model and predictive control need to be tested in real vehicles, with real driving conditions and estimated foresight input data (rather than recorded data). Research about explainability is required for a better understanding of the capabilities and limitations of the prediction model and predictive control.

Besides method modifications, future research might improve other use-cases with a data-driven prediction model of quantile sequences or a predictive control using such models. For example, the method could be applied to a predictive control of battery heating or other components of the electric drivetrain. Additionally, an application in other industries could be assessed, such as the thermal management of buildings, predictive control in aviation, or financial forecasting. Overall, the developed method provides a large base for further research about prediction models and predictive control, in order to develop more energy-efficient control systems.

# A Appendix

## A.1 Weather data analysis

If a vehicle fleet data set contains the vehicle location, weather data from nearby weather stations can be additionally analyzed and taken as input for the prediction model [1]. The data processing steps conducted in this work for an according analysis are shown in Fig. A.1. Weather data from the German Weather Service (DWD) [239] are used for the analysis. The vehicle location is provided in longitude ($long$) and latitude ($lat$) coordinates from the Global Positioning System (GPS).

At first, faulty or plausible data are filtered. The first and last six segments are cut off, since the vehicle could be located in a parking garage such that its ambient temperature measurements are not comparable with weather data. The plausibility of the vehicle position is investigated by comparing the GPS distance and the measured distance of a trip. The GPS distance $d_{GPS}$ is calculated in a simplified way by equations (A.1) to (A.4) according to [240], with corrected latitude $dlat$ and corrected longitude $dlong$. The curvature of earth is approximated by considering different meridian distances along the latitude.

$$d_{GPS} = \sqrt{(dlat)^2 + (dlong)^2} \tag{A.1}$$

$$dlat = 111.3km/^\circ \cdot \Delta lat \tag{A.2}$$

$$dlong = 111.3km/^\circ \cdot \cos(\overline{lat}) \cdot \Delta long \tag{A.3}$$

$$\overline{lat} = \frac{lat_1 + lat_2}{2} \tag{A.4}$$

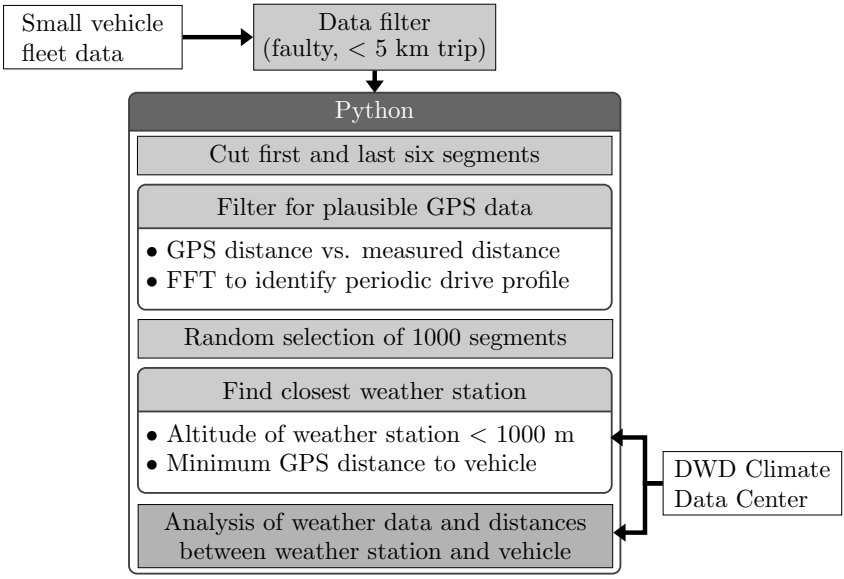


**Figure A.1:** Steps of data processing for weather data analysis as used in [1] and 4.2.1.

A trip is neglected if the distances between weather station and vehicle differ by more than 10 km or 5 %. Drives on a test rig inside a testing facility are identified if the drive follows a periodic profile. A Fast Fourier Transformation (FFT) of the speed provides information about the periodicity. A trip is neglected if the maximum absolute amplitude is larger than 100 and shifted by at least two steps in the frequency domain. The analysis is reduced to 1000 segments.

A list of the weather stations is queried from the DWD server to find the closest weather station. As additional, simplified requirement, only weather stations located at 1000 m altitude or lower are considered, to discard weather stations located at mountain peaks [1]. For each sample, the weather data of the matching weather station, date, and time are queried. An analysis can target the differences in measured ambient temperature combined with the distance between vehicle and weather station. According results are included in [1] and show a good fit between the measured ambient temperature of the vehicle and the weather stations. The difference is between -2.1 °C and 3.5 °C in 95 % of the data.

## A.2 Fleet data clusters

Random samples of the time-series cluster results from the large vehicle fleet data are shown in Fig. A.2 and for the second vehicle model in Fig. A.3. The shown Dynamic Time Warping based Barycentric Averaging (DBA) curves are the same than in Fig. 4.6 in section 4.2.1.1. The results are shown after outlier removal with 1 % for each cluster.

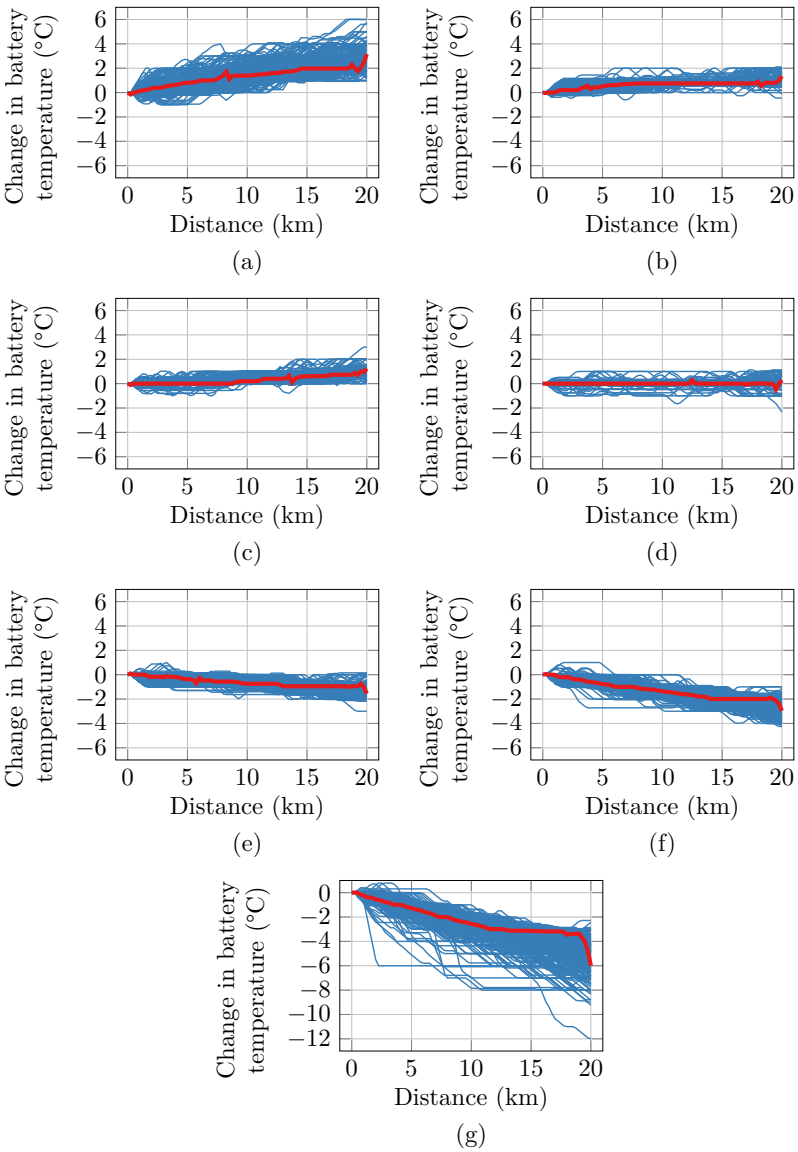


**Figure A.2:** A random sample of 500 curves for each cluster 1 (a) to 7 (g) and the DBA (red) from time-series clustering of the large vehicle fleet data.

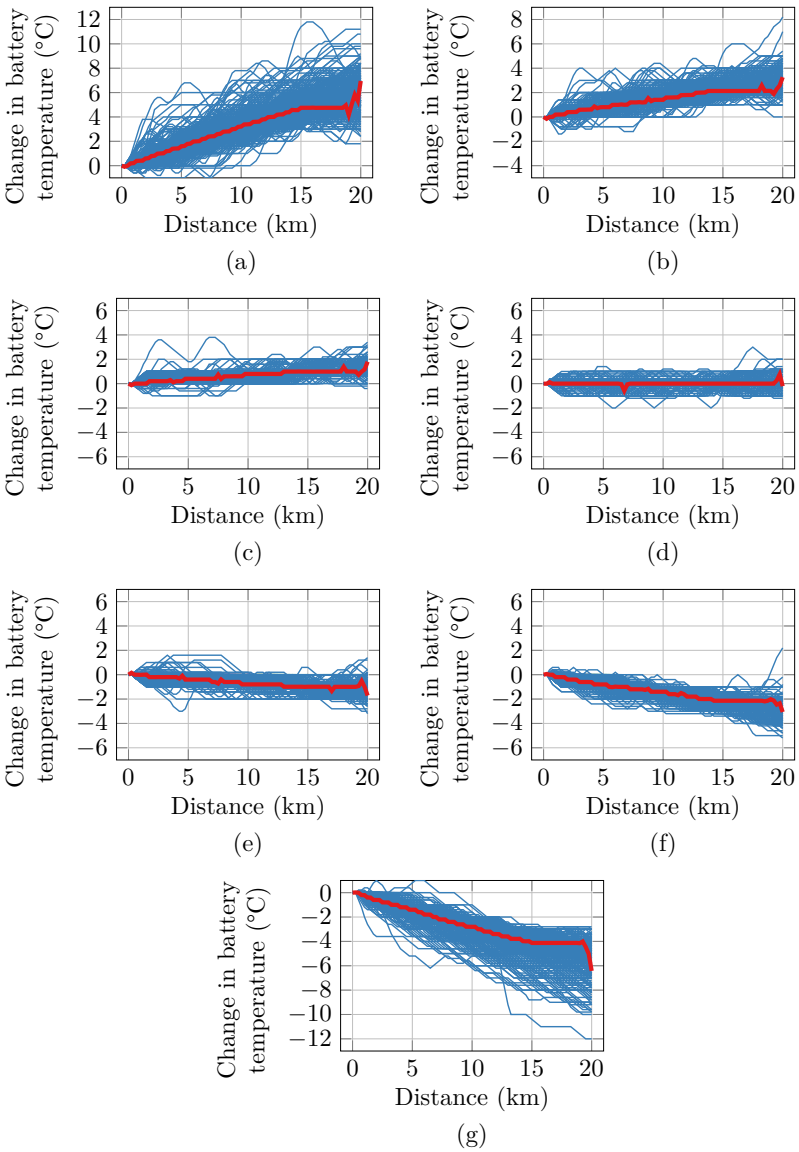


**Figure A.3:** A random sample of 500 curves for each cluster 1 (a) to 7 (g) and the DBA (red) from time-series clustering of the large vehicle fleet data of a second vehicle model.

## A.3 Q*NN input features and fixed parameters

The choice of input features of the Deep Learning based Quantile Neural Networks (Q*NN) is based on the data collection and processing step (see 4.2.1). The used features and qualifiers (QL) are marked with 1 for the each horizon in Table A.1, taken from [1]. Additional weather data (marked with *) are only used with the small vehicle fleet data for the development of the QCNN22 from [1].

**Table A.1:** Input features and qualifiers of both input channels, cited from [1].

| Parameter | History Channel | Foresight Channel |
|---|---|---|
| Acceleration pedal | 1 | 0 |
| Ambient temperature | 1 | 1 |
| Battery cooling threshold end | 1 | 1 |
| Battery cooling threshold start | 1 | 1 |
| Battery SOC rate | 1 | 0 |
| Battery SOC squared | 1 | 0 |
| Battery State of Charge (SOC) | 1 | 0 |
| Battery temperature | 1 | 0 |
| Battery temperature - Ambient temperature | 1 | 0 |
| Battery temperature rate | 1 | 0 |
| Cabin target temperature | 1 | 0 |
| Cabin target temperature QL | 1 | 0 |
| Cabin temperature | 1 | 0 |
| Cabin temperature QL | 1 | 0 |
| DPMA acceleration propulsion | 1 | 0 |
| DPMA acceleration propulsion QL | 1 | 0 |

(Continues on next page)

| | | |
|---|---|---|
| DPMA acceleration recuperation | 1 | 0 |
| DPMA acceleration recuperation QL | 1 | 0 |
| DPMA speed propulsion | 1 | 0 |
| DPMA speed propulsion QL | 1 | 0 |
| DPMA speed recuperation | 1 | 0 |
| DPMA speed recuperation QL | 1 | 0 |
| Electric machine stator temperature | 1 | 0 |
| Electric machine stator temperature QL | 1 | 0 |
| Global radiation* | 1 | 1 |
| Global radiation QL* | 1 | 1 |
| Inverter temperature | 1 | 0 |
| Inverter temperature QL | 1 | 0 |
| Relative humidity* | 1 | 1 |
| Relative humidity QL* | 1 | 1 |
| Road height difference negative sum | 1 | 1 |
| Road height difference negative sum QL | 1 | 1 |
| Road height difference positive sum | 1 | 1 |
| Road height difference positive sum QL | 1 | 1 |
| Speed | 1 | 1 |
| Speed difference | 1 | 1 |
| Speed difference squared | 1 | 1 |
| Speed inverted | 1 | 1 |
| Speed to the power of five | 1 | 1 |
| Temperature after heat pump | 1 | 0 |
| Temperature after heat pump QL | 1 | 0 |
| Vehicle torque | 1 | 0 |
| Within horizon QL | 1 | 1 |

Settings of fixed parameters of the Deep Learning based Quantile Neural Networks (Q*NN), taken and cited from [1]:

- "Optimizer: Adaptive Moment Estimation (ADAM)
- Maximum number of epochs: 200
- Early stopping patience: 20
- Quantiles: [0.01, 0.1, 0.25, 0.5, 0.75, 0.9, 0.99]
- Initial quantile weights (after constraints and softmax): [0.024, 0.064, 0.175, 0.475, 0.175, 0.064, 0.024]
- Horizon lengths: 20 (history), 80 (foresight and prediction)
- Kernel and bias constraint: maxnorm with weight 2
- Activation function: "relu" ("linear" in final dense layers)
- Conv1D kernel size: 3 (history), 15 (foresight), 25 and 15 (combined)
- MaxPooling poolsize: 2 (history), 3 (foresight and combined)
- Scaling factor for number of nodes: 2 (Conv1D) and 0.5 (Dense) for history and foresight, 1.75 (Dense in Concatenate) and 0.25 (Dense in ChannelOut) for combined. The number of nodes of the quantile output layers equals the fixed horizon length of 80. All other layers have a scaling factor of 1."

## A.4 Further example predictions

Additionally to 4.2.3.3, further example predictions are shown in Fig. A.4 for the QCNN (a), QGRU (b), QCNN22 (c) and the reference regression models (d), as provided by [3]. In this example, all models predict the correct trend of battery temperature change. However, the QCNN and QGRU predictions are more accurate, and their quantile intervals are more narrow than for the QETR.

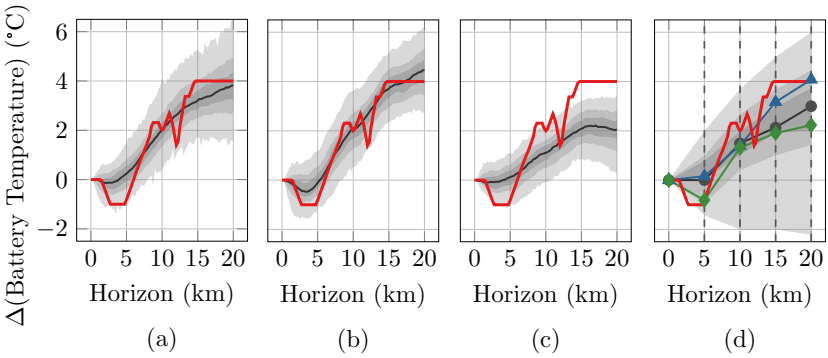


**Figure A.4:** Example predictions of the QCNN (a), QGRU (b), QCNN22 (c) and the reference regression models (d) from [3]. The true values are represented by a thick red line, the median by a thin black line and the quantile intervals are the gray areas. (d) includes the LGBM-ETR (triangles), SDT (diamonds) and QETR (circles and gray areas of quantile intervals).

## A.5 Design of experiments for predictive control evaluation

The Design of Experiments (DOE) used for predictive control evaluation is taken from [2] and shown in Table A.2. As described in 4.3.3, the DOE is based on the drive profiles US06 (D) [223], Artemis (Highway H, Rural R and Urban U) [235] and Tzirakis *et al.* (T) [236]. The noise on foresight input data for robustness analysis is specified by Table A.3.

**Table A.2:** Design of experiments for tunability evaluation of the predictive control, with initial battery temperature $T_{b,init}$, ambient temperature $T_{amb}$, and slope $\theta$, cited from [2].

| Scenario | Profile | Elevation | $T_{amb}$ (°C) | $T_{b,init}$ (°C) | $\lvert\theta\rvert\ \forall(\uparrow,\downarrow)$ (m/km) |
|---|---|---|---|---|---|
| 1/2/3/4 | HDDD | _ ↑ ↓ ↑ | 30/30/25/30 | 33.5/38/33.5/30 | 20 |
| 5/6/7/8 | HDDUU | _ ↑ ↓ _ _ | 30/30/25/30 | 33.5/38/33.5/30 | 20 |
| 9/10/11/12 | HRUU | _ _ _ _ | 30/30/25/30 | 33.5/38/33.5/30 | 20 |
| 13/14/15 | 12T | 3↓ 3↑ 3↓ 3↑ | 30/20/30 | 30/25/38 | 50 |
| 16/17/18 | 9TUU | 3↓ 3↑ 3↓ _ _ | 30/20/30 | 30/25/38 | 50 |

**Table A.3:** Foresight input noise, cited from [2]. "Gaussian noise is defined by mean value $\mu$ and standard deviation $\sigma$, centered moving average by window size."

| Noise type | Speed | Elevation |
|---|---|---|
| More Dynamic (MD) | Gaussian noise ($\mu = 0$ km/h, $\sigma = 20$ km/h) | Gaussian noise ($\mu = 0$ m, $\sigma = 4$ m) |
| Less Dynamic (LD) | Centered moving average (window size 9) | Gaussian noise ($\mu = 0$ m, $\sigma = 4$ m) |

# List of Figures

# List of Tables

# List of Publications

## Journal articles

[1] A. M. Billert, M. Frey, and F. Gauterin, "A Method of Developing Quantile Convolutional Neural Networks for Electric Vehicle Battery Temperature Prediction Trained on Cross-Domain Data," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 411–425, 2022.

[2] A. M. Billert, S. Erschen, M. Frey, and F. Gauterin, "Predictive battery thermal management using quantile convolutional neural networks," *Transportation Engineering*, vol. 10, p. 100150, 2022.

[3] A. M. Billert, R. Yu, S. Erschen, M. Frey, and F. Gauterin, "Improved Quantile Convolutional and Recurrent Neural Networks for Electric Vehicle Battery Temperature Prediction," *Big Data Mining and Analytics*, vol. 7, no. 2, pp. 512–530, 2024.

Individual contributions to the published articles:

- Article [1]: Main idea, concept, implementation and writing by Andreas M. Billert, feedback by Michael Frey, supervision by Frank Gauterin. Presentation of the published article at 14th International Joint Conference on Computational Intelligence (IJCCI) 2022 conference by Andreas M. Billert.
- Article [2]: Main idea, concept, implementation and writing by Andreas M. Billert, feedback by Stefan Erschen and Michael Frey, supervision by Frank Gauterin.

- Article [3]: Support on time-series clustering methods and their implementation by Runyao Yu. Implementation of all other parts as well as the main idea, concept and writing by Andreas M. Billert. Feedback by Stefan Erschen and Michael Frey, supervision by Frank Gauterin.

## Supervised master theses

[4] R. Yu, "Data-Driven Understanding and Efficient Prediction of Thermal Behavior for Electric Vehicle Battery using Artificial Intelligence," Master thesis, Technical University of Munich, Munich, 29.09.2022.

[5] X. Huang, "Extracting rule-based decisions of a predictive control of the battery thermal management system using machine learning," Master thesis, Leibniz University Hannover, Hannover, 31.05.2023.

Contributions of the supervised master theses to the dissertation:

- Master thesis [4]: Applicability of state-of-the-art time-series clustering methods, which are adapted and suited by Andreas M. Billert for application in this dissertation. Furthermore, the proposal of the topic is from Andreas M. Billert, including the idea of battery temperature sequence clustering.
- Master thesis [5]: Applicability of further state-of-the-art classification methods for rule-based predictive control, which however could not significantly improve the rule-based control. Additionally, the proposal of the topic is from Andreas M. Billert, including the idea of derivation of rule-based methods from a predictive battery thermal management control.

For both master theses, the implementations and choice of methods have been discussed as part of the feedback and supervision by Andreas M. Billert.

# Patents

[6] A. M. Billert and S. Fuchs, “Temperature-Control System and Method for the Temperature Control of an Electrified Motor Vehicle,” Patent Application Publication WO 00 2022 156 968 A1, 2022. [Online]. Available: https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=WO002022156968A1

[7] A. M. Billert, E. Alberts, and S. Fuchs, “Heat Management System for an Electrified Motor Vehicle,” Patent Application Publication WO 00 2022 233 524 A1, 2022. [Online]. Available: https://depatisnet.dpma.de/DepatisNet/depatisnet?action=bibdat&docid=WO002022233524A1

Individual contributions to the patent application publications:

- Patent application publication [6]: Main idea and concept by both Andreas M. Billert and Simone Fuchs. Diagrams from Andreas M. Billert with feedback from Simone Fuchs.
- Patent application publication [7]: Main idea and concept by Andreas M. Billert, Esther Alberts and Simone Fuchs. Focus of Andreas M. Billert was on the prediction model part, focus of Esther Alberts on the Reinforcement Learning part, and focus of Simone Fuchs on the battery thermal management system.

# Bibliography

[8] W. Choi, E. Yoo, E. Seol, M. Kim, and H. H. Song, "Greenhouse gas emissions of conventional and alternative vehicles: Predictions based on energy policy analysis in South Korea," *Applied Energy*, vol. 265, p. 114754, 2020.

[9] Y. Zheng, X. He, H. Wang, M. Wang, S. Zhang, D. Ma, B. Wang, and Y. Wu, "Well-to-wheels greenhouse gas and air pollutant emissions from battery electric vehicles in China," *Mitigation and Adaptation Strategies for Global Change*, vol. 25, no. 3, pp. 355–370, 2020.

[10] G. Conway, A. Joshi, F. Leach, A. García, and P. K. Senecal, "A review of current and future powertrain technologies and trends in 2020," *Transportation Engineering*, vol. 5, p. 100080, 2021.

[11] International Energy Agency, "Global EV Outlook 2023: Catching up with climate ambitions," Paris, 2023, Accessed May 28, 2023. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2023

[12] A. Enthaler, T. Weustenfeld, F. Gauterin, and J. Koehler, "Thermal management consumption and its effect on remaining range estimation of electric vehicles," in *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. Vienna, Austria: IEEE, 2014, pp. 170–177.

[13] J. Neubauer and E. Wood, "Thru-life impacts of driver aggression, climate, cabin thermal management, and battery thermal management on battery electric vehicle utility," *Journal of Power Sources*, vol. 259, pp. 262–275, 2014.

[14] H. Helms, B. Bruch, D. Räder, S. Hausberger, S. Lipp, and C. Matzer, "Energieverbrauch von Elektroautos (BEV)," Dessau-Roßlau, 2022, Accessed April 16, 2023. [Online]. Available: https://www.umweltbundesamt.de/publikationen/energieverbrauch-von-elektroautos

[15] L. He, Z. Gu, Y. Zhang, H. Jing, and P. Li, "Review on Thermal Management of Lithium-Ion Batteries for Electric Vehicles: Advances, Challenges, and Outlook," *Energy & Fuels*, vol. 37, no. 7, pp. 4835–4857, 2023.

[16] M. Auer, *Ein Beitrag zur Erhöhung der Reichweite eines batterieelektrischen Fahrzeugs durch prädiktives Thermomanagement*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016.

[17] S. Zhao, A. M. Reza, S. Jing, and C. C. Mi, "A Two-Layer Real-Time Optimization Control Strategy for Integrated Battery Thermal Management and HVAC System in Connected and Automated HEVs," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 6567–6576, 2021.

[18] S. Park and C. Ahn, "Computationally Efficient Stochastic Model Predictive Controller for Battery Thermal Management of Electric Vehicle," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8407–8419, 2020.

[19] S. Yang, X. Liu, S. Li, and C. Zhang, *Advanced Battery Management System for Electric Vehicles*, ser. Key Technologies on New Energy Vehicles. Singapore: Springer Nature Singapore, 2022.

[20] Y. Zheng, "Methodologies for Cross-Domain Data Fusion: An Overview," *IEEE Transactions on Big Data*, vol. 1, no. 1, pp. 16–34, 2015.

[21] T. Straub, M. Frey, and F. Gauterin, "Learning From the Fleet: Map Attributes for Energetic Representation of Driving Profiles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 471–482, 2022.

[22] P. Engel, S. Lempp, A. Rausch, and W. Tegethoff, "Improving Thermal Management of Electric Vehicles by Prediction of Thermal Disturbance

Variables," in *ADAPTIVE 2018*, A. Rausch, C. Knieke, and M. Schranz, Eds. Barcelona, Spain: IARIA, 2018, pp. 75–83.

[23] Y. Xie, C. Wang, X. Hu, X. Lin, Y. Zhang, and W. Li, "An MPC-Based Control Strategy for Electric Vehicle Battery Cooling Considering Energy Saving and Battery Lifespan," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 657–14 673, 2020.

[24] P. Fergus and C. Chalmers, *Applied Deep Learning: Tools, Techniques, and Implementation*, ser. Computational Intelligence Methods and Applications. Springer International Publishing, 2022.

[25] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era," in *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy: IEEE, 2017, pp. 843–852.

[26] T. J. Sejnowski, "The unreasonable effectiveness of deep learning in artificial intelligence," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 033–30 038, 2020.

[27] R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka, "A Multi-Horizon Quantile Recurrent Forecaster," in *31st Annual Conference on Neural Information Processing Systems (NIPS): Time Series Workshop*, Long Beach, CA, USA, 2017.

[28] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, F.-X. Aubet, L. Callot, and T. Januschowski, "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey," *ACM Comput. Surv.*, vol. 55, no. 6, 2022.

[29] A. Arpteg, B. Brinne, L. Crnkovic-Friis, and J. Bosch, "Software Engineering Challenges of Deep Learning," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Prague, Czech Republic: IEEE, 2018, pp. 50–59.

[30] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation," in *Agile Processes in Software Engineering and Extreme Programming*, P. Kruchten, S. Fraser, and F. Coallier, Eds. Cham: Springer International Publishing, 2019, pp. 227–243.

[31] N. Moniz, P. Branco, and L. Torgo, "Resampling strategies for imbalanced time series forecasting," *International Journal of Data Science and Analytics*, vol. 3, no. 3, pp. 161–181, 2017.

[32] S. Lerch, T. L. Thorarinsdottir, F. Ravazzolo, and T. Gneiting, "Forecaster's Dilemma: Extreme Events and Forecast Evaluation," *Statistical Science*, vol. 32, no. 1, pp. 106–127, 2017.

[33] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020.

[34] J. Krumm, R. Gruen, and D. Delling, "From destination prediction to route prediction," *Journal of Location Based Services*, vol. 7, no. 2, pp. 98–120, 2013.

[35] Z. Zhou, Z. Yang, Y. Zhang, Y. Huang, H. Chen, and Z. Yu, "A comprehensive study of speed prediction in transportation system: From vehicle to traffic," *iScience*, vol. 25, no. 3, p. 103909, 2022.

[36] S. Rasp and S. Lerch, "Neural Networks for Postprocessing Ensemble Weather Forecasts," *Monthly Weather Review*, vol. 146, no. 11, pp. 3885–3900, 2018.

[37] I. Gräßler, J. Hentze, and T. Bruckmann, "V-Models for Interdisciplinary Systems Engineering," in *Proceedings of the DESIGN 2018 15th International Design Conference*, D. Marjanovic, M. Storga, N. Pavkovic, N. Bojcetic, and S. Skec, Eds., Dubrovnik, Croatia, 2018, pp. 747–756.

[38] I. Graessler and J. Hentze, "The new V-Model of VDI 2206 and its validation," *at - Automatisierungstechnik*, vol. 68, no. 5, pp. 312–324, 2020.

[39] P. C. Austin and M. J. Schull, "Quantile Regression: A Statistical Tool for Out-of-hospital Research," *Academic Emergency Medicine*, vol. 10, no. 7, pp. 789–797, 2003.

[40] K. Rogers, "What's the Difference Between Speed and Velocity?" 2016, Accessed June 3, 2023. [Online]. Available: https://www.britannica.com/story/whats-the-difference-between-speed-and-velocity

[41] M. Doppelbauer, *Grundlagen der Elektromobilität: Technik, Praxis, Energie und Umwelt*. Wiesbaden: Springer Vieweg, 2020.

[42] J. V. Barreras, T. Raj, and D. A. Howey, "Derating Strategies for Lithium-Ion Batteries in Electric Vehicles," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. Washington, DC, USA: IEEE, 2018, pp. 4956–4961.

[43] R. Korthauer, *Handbuch Lithium-Ionen-Batterien*. Berlin: Springer Vieweg, 2013.

[44] L. Spitthoff, P. R. Shearing, and O. S. Burheim, "Temperature, Ageing and Thermal Management of Lithium-Ion Batteries," *Energies*, vol. 14, no. 5, p. 1248, 2021.

[45] J. R. Belt, C. D. Ho, C. G. Motloch, T. J. Miller, and T. Q. Duong, "A capacity and power fade study of Li-ion cells during life cycle testing," *Journal of Power Sources*, vol. 123, no. 2, pp. 241–246, 2003.

[46] J. Vetter, P. Novák, M. Wagner, C. Veit, K.-C. Möller, J. Besenhard, M. Winter, M. Wohlfahrt-Mehrens, C. Vogler, and A. Hammouche, "Ageing mechanisms in lithium-ion batteries," *Journal of Power Sources*, vol. 147, no. 1, pp. 269–281, 2005.

[47] K. Kleiner and H. Ehrenberg, "Challenges Considering the Degradation of Cell Components in Commercial Lithium-Ion Cells: A Review and Evaluation of Present Systems," in *Electrochemical Energy Storage: Next Generation Battery Concepts*, R.-A. Eichel, Ed. Cham: Springer International Publishing, 2019, pp. 169–213.

[48] E. Peled and S. Menkin, "Review—SEI: Past, Present and Future," *Journal of The Electrochemical Society*, vol. 164, no. 7, p. A1703, 2017.

[49] C. R. Birkl, M. R. Roberts, E. McTurk, P. G. Bruce, and D. A. Howey, "Degradation diagnostics for lithium ion cells," *Journal of Power Sources*, vol. 341, pp. 373–386, 2017.

[50] S. Zhang, "Insight into the Gassing Problem of Li-ion Battery," *Frontiers in Energy Research*, vol. 2, 2014.

[51] R. Stockhausen, L. Gehrlein, M. Müller, T. Bergfeldt, A. Hofmann, F. J. Müller, J. Maibach, H. Ehrenberg, and A. Smith, "Investigating the dominant decomposition mechanisms in lithium-ion battery cells responsible for capacity loss in different stages of electrochemical aging," *Journal of Power Sources*, vol. 543, p. 231842, 2022.

[52] M. Schimpe, M. E. von Kuepach, M. Naumann, H. C. Hesse, K. Smith, and A. Jossen, "Comprehensive Modeling of Temperature-Dependent Degradation Mechanisms in Lithium Iron Phosphate Batteries," *Journal of The Electrochemical Society*, vol. 165, no. 2, p. A181, 2018.

[53] Y. Sun, S. Saxena, and M. Pecht, "Derating Guidelines for Lithium-Ion Batteries," *Energies*, vol. 11, no. 12, p. 3295, 2018.

[54] M. Schimpe, J. V. Barreras, B. Wu, and G. J. Offer, "Battery Degradation-Aware Current Derating: An Effective Method to Prolong Lifetime and Ease Thermal Management," *Journal of The Electrochemical Society*, vol. 168, no. 6, p. 060506, 2021.

[55] H. Ruan, J. V. Barreras, T. Engstrom, Y. Merla, R. Millar, and B. Wu, "Lithium-ion battery lifetime extension: A review of derating methods," *Journal of Power Sources*, vol. 563, p. 232805, 2023.

[56] A. Suchaneck, "Energiemanagement-Strategien für batterieelektrische Fahrzeuge," Dissertation, Karlsruher Institut für Technologie (KIT), Karlsruhe, 2018.

[57] K. Kruppok, B. Jäger, and R. Kriesten, "Auswirkung der Elektrifizierung von Nebenverbrauchern auf das Energiemanagement im Kraftfahrzeug," *Forschung aktuell / Hochschule Karlsruhe, Technik und Wirtschaft*, vol. 2016, pp. 47–49, 2016.

[58] M. Alipour, C. Ziebert, F. V. Conte, and R. Kizilel, "A Review on Temperature-Dependent Electrochemical Properties, Aging, and Performance of Lithium-Ion Cells," *Batteries*, vol. 6, no. 3, 2020.

[59] B. Wu, V. Yufit, M. Marinescu, G. J. Offer, R. F. Martinez-Botas, and N. P. Brandon, "Coupled thermal–electrochemical modelling of uneven heat generation in lithium-ion battery packs," *Journal of Power Sources*, vol. 243, pp. 544–554, 2013.

[60] T. R. Ashwin, Y. M. Chung, and J. Wang, "Capacity fade modelling of lithium-ion battery under cyclic loading conditions," *Journal of Power Sources*, vol. 328, pp. 586–598, 2016.

[61] Y. A. Cengel, *Heat Transfer: A practical approach*, 2nd ed. New York, NY, USA: McGraw-Hill, 2002.

[62] A. Fotouhi, D. J. Auger, K. Propp, S. Longo, and M. Wild, "A review on electric vehicle battery modelling: From Lithium-ion toward Lithium–Sulphur," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 1008–1021, 2016.

[63] W. Zhou, Y. Zheng, Z. Pan, and Q. Lu, "Review on the Battery Model and SOC Estimation Method," *Processes*, vol. 9, no. 9, 2021.

[64] J. Park and Y. Kim, "Supervised-Learning-Based Optimal Thermal Management in an Electric Vehicle," *IEEE Access*, vol. 8, pp. 1290–1302, 2020.

[65] T. J. Shelly, J. A. Weibel, D. Ziviani, and E. A. Groll, "Comparative analysis of battery electric vehicle thermal management systems under long-range drive cycles," *Applied Thermal Engineering*, vol. 198, p. 117506, 2021.

[66] A. Kampker, D. Vallée, and A. Schnettler, *Elektromobilität: Grundlagen einer Zukunftstechnologie*. Berlin: Springer Vieweg, 2013.

[67] X. Kuang, K. Li, Y. Xie, C. Wu, P. Wang, X. Wang, and C. Fu, "Research on Control Strategy for a Battery Thermal Management System for Electric Vehicles Based on Secondary Loop Cooling," *IEEE Access*, vol. 8, pp. 73 475–73 493, 2020.

[68] S. Park and C. Ahn, "Model Predictive Control With Stochastically Approximated Cost-to-Go for Battery Cooling System of Electric Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4312–4323, 2021.

[69] M.-K. Tran, S. Panchal, T. D. Khang, K. Panchal, R. Fraser, and M. Fowler, "Concept Review of a Cloud-Based Smart Battery Management System for Lithium-Ion Batteries: Feasibility, Logistics, and Functionality," *Batteries*, vol. 8, no. 2, 2022.

[70] S. Park and C. Ahn, "Stochastic Model Predictive Controller for Battery Thermal Management of Electric Vehicles," in *2019 IEEE Vehicle Power and Propulsion Conference (VPPC)*. Hanoi, Vietnam: IEEE, 2019, pp. 1–5.

[71] T. J. Shelly, J. A. Weibel, D. Ziviani, and E. A. Groll, "A Dynamic Simulation Framework for the Analysis of Battery Electric Vehicle Thermal Management Systems," in *Proceedings of the Nineteenth InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. Orlando, FL, USA: IEEE, 2020, pp. 538–547.

[72] A. Rahman, F. Nur, M. Hawlader, and R. Afroz, "Fuzzy controlled evaporative battery thermal management system for EV/HEV," *International Journal of Electric and Hybrid Vehicles*, vol. 7, no. 1, pp. 22–39, 2015.

[73] M. Umair Ali, S. Hussain Nengroo, M. Adil Khan, K. Zeb, M. Ahmad Kamran, and H.-J. Kim, "A Real-Time Simulink Interfaced Fast-Charging Methodology of Lithium-Ion Batteries under Temperature Feedback with Fuzzy Logic Control," *Energies*, vol. 11, no. 5, 2018.

[74] Y.-H. Hung, Y.-F. Lue, and H.-J. Gu, "Development of a Thermal Management System for Energy Sources of an Electric Vehicle," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 402–411, 2016.

[75] M. Scholl, K. Minnerup, C. Reiter, B. Bernhardt, E. Weisbrodt, and S. Newiger, "Optimization of a Thermal Management System for Battery Electric Vehicles," in *2019 Fourteenth International Conference on Ecological Vehicles and Renewable Energies (EVER, REAL)*, Monte-Carlo, Monaco, 2019, pp. 1–10.

[76] F. He and L. Ma, "Thermal management of batteries employing active temperature control and reciprocating cooling flow," *International Journal of Heat and Mass Transfer*, vol. 83, pp. 164–172, 2015.

[77] E. F. Camacho and C. B. Alba, *Model Predictive Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer London, 2013.

[78] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021.

[79] J. Adamy, *Nonlinear Systems and Controls*. Berlin: Springer Vieweg, 2022.

[80] L. Grüne and J. Pannek, "Nonlinear Model Predictive Control," in *Nonlinear Model Predictive Control: Theory and Algorithms*. Cham: Springer International Publishing, 2017.

[81] T. A. N. Heirung, J. A. Paulson, J. O'Leary, and A. Mesbah, "Stochastic model predictive control — how does it work?" *Computers & Chemical Engineering*, vol. 114, pp. 158–170, 2018.

[82] A. Mesbah, "Stochastic Model Predictive Control: An Overview and Perspectives for Future Research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[83] M. Torchio, L. Magni, R. D. Braatz, and D. M. Raimondo, "Design of Piecewise Affine and Linear Time-Varying Model Predictive Control Strategies for Advanced Battery Management Systems," *Journal of The Electrochemical Society*, vol. 164, no. 4, p. A949, 2017.

[84] J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators," *IFAC-PapersOnLine*, vol. 55, no. 20, pp. 331–336, 2022.

[85] M. G. Yu and G. S. Pavlak, "Extracting interpretable building control rules from multi-objective model predictive control data sets," *Energy*, vol. 240, p. 122691, 2022.

[86] T. Fischer, "Modellprädiktive Regelung eines innovativen Thermomanagement-Systems für batterieelektrische Fahrzeuge," Dissertation, Karlsruher Institut für Technologie (KIT), Karlsruhe, 2022.

[87] J. Lopez-Sanz, C. Ocampo-Martinez, J. Alvarez-Florez, M. Moreno-Eguilaz, R. Ruiz-Mansilla, J. Kalmus, M. Gräeber, and G. Lux, "Nonlinear Model Predictive Control for Thermal Management in Plug-in Hybrid Electric Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 3632–3644, 2017.

[88] J. Lopez-Sanz, C. Ocampo-Martinez, J. Alvarez-Florez, M. Moreno-Eguilaz, R. Ruiz-Mansilla, J. Kalmus, M. Graeber, and G. Lux, "Thermal Management in Plug-In Hybrid Electric Vehicles: A Real-Time Nonlinear Model Predictive Control Implementation," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 7751–7760, 2017.

[89] C. Zhu, F. Lu, H. Zhang, and C. C. Mi, "Robust Predictive Battery Thermal Management Strategy for Connected and Automated Hybrid Electric Vehicles Based on Thermoelectric Parameter Uncertainty," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 6, no. 4, pp. 1796–1805, 2018.

[90] Y. Liu and J. Zhang, "Self-Adapting Intelligent Battery Thermal Management System via Artificial Neural Network Based Model Predictive Control," in *Volume 2A: 45th Design Automation Conference*. Anaheim, California, USA: American Society of Mechanical Engineers, 2019, p. V02AT03A026.

[91] Y. Liu and J. Zhang, "Electric Vehicle Battery Thermal and Cabin Climate Management Based on Model Predictive Control," *Journal of Mechanical Design*, vol. 143, no. 3, p. 031705, 2020.

[92] H. Zomorodi, D. Yoon, and B. Ayalew, "Use of predictive information for Battery pack Thermal Management," in *2017 American Control Conference (ACC)*. Seattle, WA, USA: IEEE, 2017, pp. 5020–5025.

[93] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time Series Classification With Multivariate Convolutional Neural Network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, 2019.

[94] A. Jain, "Methods for Data-driven Model Predictive Control," Dissertation, University of Pennsylvania, Philadelphia, 2020. [Online]. Available: https://repository.upenn.edu/edissertations/3904/

[95] A. Kempf, E. Weber, and S. Müller, "A Multi Model Neural Network Approach for Longitudinal Model Predictive Control of a Passenger Vehicle," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, 2020, pp. 1–6.

[96] F. Deufel, M. Gießler, and F. Gauterin, "A Generic Prediction Approach for Optimal Control of Electrified Vehicles Using Artificial Intelligence," *Vehicles*, vol. 4, no. 1, pp. 182–198, 2022.

[97] H. Hassanpour, B. Corbett, and P. Mhaskar, "Artificial Neural Network-Based Model Predictive Control Using Correlated Data," *Industrial & Engineering Chemistry Research*, vol. 61, no. 8, pp. 3075–3090, 2022.

[98] J. W. Moon and J.-J. Kim, "ANN-based thermal control models for residential buildings," *Building and Environment*, vol. 45, no. 7, pp. 1612–1625, 2010.

[99] A. Afram, F. Janabi-Sharifi, A. S. Fung, and K. Raahemifar, "Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system," *Energy and Buildings*, vol. 141, pp. 96–113, 2017.

[100] A. Jain, F. Smarra, E. Reticcioli, A. D'Innocenzo, and M. Morari, "NeurOpt: Neural network based optimization for building energy management and climate control," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., vol. 120. Online: PMLR, 2020, pp. 445–454.

[101] U. Kamath and J. Liu, *Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning*. Cham: Springer International Publishing, 2021.

[102] D. Jha, V. Gupta, L. Ward, Z. Yang, C. Wolverton, I. Foster, W.-k. Liao, A. Choudhary, and A. Agrawal, "Enabling deeper learning on big data for materials informatics applications," *Scientific Reports*, vol. 11, no. 1, p. 4244, 2021.

[103] L. von Rueden, S. Mayer, R. Sifa, C. Bauckhage, and J. Garcke, "Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions," in *Advances in Intelligent Data Analysis XVIII*, M. R. Berthold, A. Feelders, and G. Krempl, Eds. Springer International Publishing, 2020, pp. 548–560.

[104] L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, M. Walczak, J. Pfrommer, A. Pick, R. Ramamurthy, J. Garcke,

C. Bauckhage, and J. Schuecker, "Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2021.

[105] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 1, 2015.

[106] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, "Machine learning on big data: Opportunities and challenges," *Neurocomputing*, vol. 237, pp. 350–361, 2017.

[107] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine Learning With Big Data: Challenges and Approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[108] S. Liu, B. B. Kappes, B. Amin-ahmadi, O. Benafan, X. Zhang, and A. P. Stebner, "Physics-informed machine learning for composition – process – property design: Shape memory alloy demonstration," *Applied Materials Today*, vol. 22, p. 100898, 2021.

[109] M. Ali, A. Alqahtani, M. W. Jones, and X. Xie, "Clustering and Classification for Time Series Data in Visual Analytics: A Survey," *IEEE Access*, vol. 7, pp. 181 314–181 338, 2019.

[110] J. Fan, F. Han, and H. Liu, "Challenges of Big Data analysis," *National Science Review*, vol. 1, no. 2, pp. 293–314, 2014.

[111] C. C. Aggarwal, *Outlier Analysis*, 2nd ed., ser. Springer eBook Collection Computer Science. Cham: Springer, 2017.

[112] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier Detection: Methods, Models, and Classification," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 55:1–55:37, 2020.

[113] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A Review on Outlier/Anomaly Detection in Time Series Data," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 56:1–56:33, 2021.

[114] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*. Pisa, Italy: IEEE, 2008, pp. 413–422.

[115] I. E. Livieris, S. Stavroyiannis, L. Iliadis, and P. Pintelas, "Smoothing and stationarity enforcement framework for deep learning time-series forecasting," *Neural Computing and Applications*, vol. 33, no. 20, pp. 14 021–14 035, 2021.

[116] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, "An Empirical Comparison of Machine Learning Models for Time Series Forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[117] N. Zobeiry and K. D. Humfeld, "A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications," *Engineering Applications of Artificial Intelligence*, vol. 101, p. 104232, 2021.

[118] C. Weaver, A. C. Fortuin, A. Vladyka, and T. Albrecht, "Unsupervised classification of voltammetric data beyond principal component analysis," *Chem. Commun.*, vol. 58, no. 73, pp. 10 170–10 173, 2022.

[119] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Information Sciences*, vol. 465, pp. 1–20, 2018.

[120] P. Vuttipittayamongkol and E. Elyan, "Neighbourhood-based undersampling approach for handling imbalanced and overlapped data," *Information Sciences*, vol. 509, pp. 47–70, 2020.

[121] C. Li, L. Minati, K. K. Tokgoz, M. Fukawa, J. Bartels, A. Sihan, K.-I. Takeda, and H. Ito, "Integrated Data Augmentation for Accelerometer Time Series in Behavior Recognition: Roles of Sampling, Balancing, and Fourier Surrogates," *IEEE Sensors Journal*, vol. 22, no. 24, pp. 24 230–24 241, 2022.

[122] S. Jia, H. Xianglin, Q. Sijun, and S. Qing, "A bi-directional sampling based on K-means method for imbalance text classification," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. Okayama, Japan: IEEE, 2016, pp. 1–5.

[123] C. Liu, J. Tan, H. Shi, and X. Wang, "Lithium-Ion Cell Screening With Convolutional Neural Networks Based on Two-Step Time-Series Clustering and Hybrid Resampling for Imbalanced Data," *IEEE Access*, vol. 6, pp. 59 001–59 014, 2018.

[124] N. Moniz, P. Branco, and L. Torgo, "Resampling Strategies for Imbalanced Time Series," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Montreal, QC, Canada: IEEE, 2016, pp. 282–291.

[125] M. Saripuddin, A. Suliman, S. Syarmila Sameon, and B. N. Jorgensen, "Random Undersampling on Imbalance Time Series Data for Anomaly Detection," in *MLMI 2021 The 4th International Conference on Machine Learning and Machine Intelligence*, ser. MLMI'21. Hangzhou, China: Association for Computing Machinery, 2021, pp. 151–156.

[126] L. P. Silvestrin, L. Pantiskas, and M. Hoogendoorn, "A Framework for Imbalanced Time-Series Forecasting," in *Machine Learning, Optimization, and Data Science*, G. Nicosia, V. Ojha, E. La Malfa, G. La Malfa, G. Jansen, P. M. Pardalos, G. Giuffrida, and R. Umeton, Eds. Springer International Publishing, 2022, pp. 250–264.

[127] X. Chen, L. Gupta, and S. Tragoudas, "Improving the Forecasting and Classification of Extreme Events in Imbalanced Time Series Through Block Resampling in the Joint Predictor-Forecast Space," *IEEE Access*, vol. 10, pp. 121 048–121 079, 2022.

[128] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PloS one*, vol. 16, no. 7, pp. 1–32, 2021.

[129] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time Series Data Augmentation for Deep Learning: A Survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4653–4660.

[130] G. Iglesias, E. Talavera, Á. González-Prieto, A. Mozo, and S. Gómez-Canaval, "Data Augmentation techniques in time series domain: a survey and taxonomy," *Neural Computing and Applications*, vol. 35, no. 14, pp. 10 123–10 145, 2023.

[131] B. Fu, F. Kirchbuchner, and A. Kuijper, "Data Augmentation for Time Series: Traditional vs Generative Models on Capacitive Proximity Time Series," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '20. Corfu, Greece: Association for Computing Machinery, 2020, pp. 107–116.

[132] B. Liu, Z. Zhang, and R. Cui, "Efficient Time Series Augmentation Methods," in *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. Chengdu, China: IEEE, 2020, pp. 1004–1009.

[133] B. K. Iwana and S. Uchida, "Time Series Data Augmentation for Neural Networks by Time Warping with a Discriminative Teacher," in *2020 25th International Conference on Pattern Recognition (ICPR)*. Milan, Italy: IEEE, 2021, pp. 3558–3565.

[134] S. Demir, K. Mincev, K. Kok, and N. G. Paterakis, "Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting," *Applied Energy*, vol. 304, p. 117695, 2021.

[135] S. Kim, N. H. Kim, and J.-H. Choi, "Prediction of remaining useful life by data augmentation technique based on dynamic time warping," *Mechanical Systems and Signal Processing*, vol. 136, p. 106486, 2020.

[136] S. B. Taieb and A. F. Atiya, "A Bias and Variance Analysis for Multistep-Ahead Time Series Forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 1, pp. 62–76, 2016.

[137] D. Sahoo, N. Sood, U. Rani, G. Abraham, V. Dutt, and A. D. Dileep, "Comparative Analysis of Multi-Step Time-Series Forecasting for Network Load Dataset," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Kharagpur, India: IEEE, 2020, pp. 1–7.

[138] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A Review of Deep Learning Models for Time Series Prediction," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 7833–7848, 2021.

[139] L. Sehovac and K. Grolinger, "Deep Learning for Load Forecasting: Sequence to Sequence Recurrent Neural Networks With Attention," *IEEE Access*, vol. 8, pp. 36 411–36 426, 2020.

[140] J.-F. Toubeau, J. Bottieau, F. Vallée, and Z. de Grève, "Deep Learning-Based Multivariate Probabilistic Forecasting for Short-Term Scheduling in Power Markets," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1203–1215, 2019.

[141] R. Sen, H.-F. Yu, and I. S. Dhillon, "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting," in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Vancouver, Canada: Curran Associates, Inc, 2019.

[142] A. M. Pirbazari, E. Sharma, A. Chakravorty, W. Elmenreich, and C. Rong, "An Ensemble Approach for Multi-Step Ahead Energy Forecasting of Household Communities," *IEEE Access*, vol. 9, pp. 36 218–36 240, 2021.

[143] P.-P. Phyo, Y.-C. Byun, and N. Park, "Short-Term Energy Forecasting Using Machine-Learning-Based Ensemble Voting Regression," *Symmetry*, vol. 14, no. 1, p. 160, 2022.

[144] C. M. Liapis, A. Karanikola, and S. Kotsiantis, "Energy Load Forecasting: Investigating Mid-Term Predictions with Ensemble Learners," in *Artificial Intelligence Applications and Innovations*, I. Maglogiannis, L. Iliadis, J. Macintyre, and P. Cortez, Eds. Cham: Springer International Publishing, 2022, pp. 343–355.

[145] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013.

[146] Y. Liu, J. Wang, P. Zhao, D. Qin, and Z. Chen, "Research on Classification and Recognition of Driving Styles Based on Feature Engineering," *IEEE Access*, vol. 7, pp. 89 245–89 255, 2019.

[147] J.-S. Ang, K.-W. Ng, and F.-F. Chua, "Modeling Time Series Data with Deep Learning: A Review, Analysis, Evaluation and Future Trend," in *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*. Selangor, Malaysia: IEEE, 2020, pp. 32–37.

[148] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2194, p. 20200209, 2021.

[149] Z. Liu, Z. Zhu, J. Gao, and C. Xu, "Forecast Methods for Time Series Data: A Survey," *IEEE Access*, vol. 9, pp. 91 896–91 912, 2021.

[150] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[151] A. Wunsch, T. Liesch, and S. Broda, "Groundwater level forecasting with artificial neural networks: a comparison of long short-term memory (LSTM), convolutional neural networks (CNNs), and non-linear autoregressive networks with exogenous input (NARX)," *Hydrology and Earth System Sciences*, vol. 25, no. 3, pp. 1671–1687, 2021.

[152] G. Hayder, M. Iwan Solihin, and M. R. N. Najwa, "Multi-step-ahead prediction of river flow using NARX neural networks and deep learning LSTM," *H2Open Journal*, vol. 5, no. 1, pp. 43–60, 2022.

[153] S. Zhang, Y. Bao, P. Zhou, H. Jiang, and L. Dai, "Improving deep neural networks for LVCSR using dropout and shrinking structure," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Florence, Italy: IEEE, 2014, pp. 6849–6853.

[154] B. Qian, Y. Xiao, Z. Zheng, M. Zhou, W. Zhuang, S. Li, and Q. Ma, "Dynamic Multi-Scale Convolutional Neural Network for Time Series Classification," *IEEE Access*, vol. 8, pp. 109 732–109 746, 2020.

[155] Y. Wang, Z. Wang, X. Kang, and Y. Luo, "A novel interpretable model ensemble multivariate fast iterative filtering and temporal fusion transform for carbon price forecasting," *Energy Science & Engineering*, vol. 11, no. 3, pp. 1148–1179, 2023.

[156] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the Disharmony Between Dropout and Batch Normalization by Variance Shift," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, 2019, pp. 2677–2685.

[157] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards Understanding Regularization in Batch Normalization," in *Seventh International Conference on Learning Representations ICLR 2019*, New Orleans, LA, USA, 2019.

[158] M. Lopez-Martin, A. Sanchez-Esguevillas, L. Hernandez-Callejo, J. I. Arribas, and B. Carro, "Additive Ensemble Neural Network with Constrained Weighted Quantile Loss for Probabilistic Electric-Load Forecasting," *Sensors*, vol. 21, no. 9, p. 2979, 2021.

[159] R. Wen and K. Torkkola, "Deep Generative Quantile-Copula Models for Probabilistic Forecasting," in *Proceedings of the 36th International Conference on Machine Learning Time Series Workshop*, Long Beach, CA, USA, 2019.

[160] Y. Park, D. Maddix, F.-X. Aubet, K. Kan, J. Gasthaus, and Y. Wang, "Learning Quantile Functions without Quantile Crossing for Distribution-free Time Series Forecasting," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., vol. 151. Virtual: PMLR, 2022, pp. 8127–8150.

[161] W. Zhang, H. Quan, and D. Srinivasan, "An Improved Quantile Regression Neural Network for Probabilistic Load Forecasting," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 4425–4434, 2019.

[162] C. Xu, C. Li, and X. Zhou, "Interpretable LSTM Based on Mixture Attention Mechanism for Multi-Step Residential Load Forecasting," *Electronics*, vol. 11, no. 14, p. 2189, 2022.

[163] W. Zhang, H. Quan, O. Gandhi, R. Rajagopal, C.-W. Tan, and D. Srinivasan, "Improving Probabilistic Load Forecasting Using Quantile Regression NN With Skip Connections," *IEEE Transactions on Smart Grid*, vol. 11, no. 6, pp. 5442–5450, 2020.

[164] D. Gan, Y. Wang, S. Yang, and C. Kang, "Embedding based quantile regression neural network for probabilistic load forecasting," *Journal of Modern Power Systems and Clean Energy*, vol. 6, no. 2, pp. 244–254, 2018.

[165] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic forecasting with temporal convolutional neural network," *Neurocomputing*, vol. 399, pp. 491–501, 2020.

[166] J. Gasthaus, K. Benidis, Y. Wang, S. S. Rangapuram, D. Salinas, V. Flunkert, and T. Januschowski, "Probabilistic Forecasting with Spline Quantile Function RNNs," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. Naha, Okinawa, Japan: PMLR, 2019, pp. 1901–1910.

[167] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[168] Y. Wang, N. Zhang, Y. Tan, T. Hong, D. S. Kirschen, and C. Kang, "Combining Probabilistic Load Forecasts," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3664–3674, 2019.

[169] Y. Yu, X. Han, M. Yang, and J. Yang, "Probabilistic Prediction of Regional Wind Power Based on Spatiotemporal Quantile Regression," *IEEE Transactions on Industry Applications*, vol. 56, no. 6, pp. 6117–6127, 2020.

[170] W. Cui, C. Wan, and Y. Song, "Ensemble Deep Learning-Based Non-Crossing Quantile Regression for Nonparametric Probabilistic Forecasting of Wind Power Generation," *IEEE Transactions on Power Systems*, pp. 1–16, 2022.

[171] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[172] J. Sappl, M. Harders, and W. Rauch, "Machine learning for quantile regression of biogas production rates in anaerobic digesters," *Science of The Total Environment*, vol. 872, p. 161923, 2023.

[173] M. Feurer and F. Hutter, "Hyperparameter Optimization," in *Automated Machine Learning: Methods, Systems, Challenges*, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham: Springer International Publishing, 2019, pp. 3–33.

[174] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, Eds., vol. 25. Lake Tahoe, Nevada, USA: Curran Associates, Inc, 2012.

[175] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic Enhancement Strategies When Using Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks," *IEEE Access*, vol. 8, pp. 52 588–52 608, 2020.

[176] X.-B. Jin, W.-Z. Zheng, J.-L. Kong, X.-Y. Wang, Y.-T. Bai, T.-L. Su, and S. Lin, "Deep-Learning Forecasting Method for Electric Power Load via Attention-Based Encoder-Decoder with Bayesian Optimization," *Energies*, vol. 14, no. 6, 2021.

[177] K. Miao, Q. Hua, and H. Shi, "Short-Term Load Forecasting Based on CNN-BiLSTM with Bayesian Optimization and Attention Mechanism," in *Parallel and Distributed Computing, Applications and Technologies*, Y. Zhang, Y. Xu, and H. Tian, Eds. Cham: Springer International Publishing, 2021, pp. 116–128.

[178] D. Deng, F. Karl, F. Hutter, B. Bischl, and M. Lindauer, "Efficient Automated Deep Learning for Time Series Forecasting," in *Machine Learning and Knowledge Discovery in Databases*, M.-R. Amini, S. Canu, A. Fischer, T. Guns, P. Kralj Novak, and G. Tsoumakas, Eds. Grenoble, France: Springer Nature Switzerland, 2022, pp. 664–680.

[179] G. de Ath, R. M. Everson, A. A. M. Rahat, and J. E. Fieldsend, "Greed is Good: Exploration and Exploitation Trade-Offs in Bayesian Optimisation," *ACM Trans. Evol. Learn. Optim.*, vol. 1, no. 1, 2021.

[180] J. Kong, W. Kowalczyk, D. A. Nguyen, T. Bäck, and S. Menzel, "Hyperparameter Optimisation for Improving Classification under Class Imbalance," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. Xiamen, China: IEEE, 2019, pp. 3072–3078.

[181] D. A. Nguyen, A. V. Kononova, S. Menzel, B. Sendhoff, and T. Bäck, "An Efficient Contesting Procedure for AutoML Optimization," *IEEE Access*, vol. 10, pp. 75 754–75 771, 2022.

[182] Z. Wang, M. Agung, R. Egawa, R. Suda, and H. Takizawa, "Automatic Hyperparameter Tuning of Machine Learning Models under Time Constraints," in *2018 IEEE International Conference on Big Data (Big Data)*. Seattle, WA, USA: IEEE, 2018, pp. 4967–4973.

[183] L. Wang, M. Feng, B. Zhou, B. Xiang, and S. Mahadevan, "Efficient hyper-parameter optimization for NLP applications," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 2112–2117.

[184] F. Hutter, H. Hoos, and K. Leyton-Brown, "An Efficient Approach for Assessing Hyperparameter Importance," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32. Bejing, China: PMLR, 2014, pp. 754–762.

[185] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and Efficient Hyperparameter Optimization at Scale," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholm, Sweden: PMLR, 2018, pp. 1437–1446.

[186] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[187] A. Meola, M. Winkler, and S. Weinrich, "Metaheuristic optimization of data preparation and machine learning hyperparameters for prediction of dynamic methane production," *Bioresource Technology*, vol. 372, p. 128604, 2023.

[188] C. M. Liapis, A. Karanikola, and S. Kotsiantis, "A Multi-Method Survey on the Use of Sentiment Analysis in Multivariate Financial Time Series Forecasting," *Entropy*, vol. 23, no. 12, 2021.

[189] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate Uncertainties for Deep Learning Using Calibrated Regression," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholm, Sweden: PMLR, 2018, pp. 2796–2804.

[190] T. Yoon, Y. Park, E. K. Ryu, and Y. Wang, "Robust Probabilistic Time Series Forecasting," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., vol. 151. PMLR, 2022, pp. 1336–1358.

[191] J. D. Olden, M. K. Joy, and R. G. Death, "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data," *Ecological Modelling*, vol. 178, no. 3, pp. 389–397, 2004.

[192] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[193] Y. Jiang, Y. Yu, J. Huang, W. Cai, and J. Marco, "Li-ion battery temperature estimation based on recurrent neural networks," *Science China Technological Sciences*, vol. 64, no. 6, pp. 1335–1344, 2021.

[194] S. Zhu, C. He, N. Zhao, and J. Sha, "Data-driven analysis on thermal effects and temperature changes of lithium-ion battery," *Journal of Power Sources*, vol. 482, p. 228983, 2021.

[195] Z. Wan, Y. Kang, R. Ou, S. Xue, D. Xu, and X. Luo, "Multi-step time series forecasting on the temperature of lithium-ion batteries," *Journal of Energy Storage*, vol. 64, p. 107092, 2023.

[196] T. J. Kim, B. D. Youn, and H. J. Kim, "Battery pack temperature estimation model for evs and its semi-transient case study," *Chemical Engineering Transactions*, vol. 33, pp. 955–960, 2013.

[197] M. Kopp, M. Ströbel, A. Fill, J. Pross-Brakhage, and K. P. Birke, "Artificial Feature Extraction for Estimating State-of-Temperature in Lithium-Ion-Cells Using Various Long Short-Term Memory Architectures," *Batteries*, vol. 8, no. 4, 2022.

[198] S. Arora, W. Shen, and A. Kapoor, "Neural network based computational model for estimation of heat generation in LiFePO4 pouch cells of different nominal capacities," *Computers & Chemical Engineering*, vol. 101, pp. 81–94, 2017.

[199] A. Afzal, J. K. Bhutto, A. Alrobaian, A. Razak Kaladgi, and S. A. Khan, "Modelling and Computational Experiment to Obtain Optimized Neural Network for Battery Thermal Management Data," *Energies*, vol. 14, no. 21, 2021.

[200] T. Wang, X. Liu, D. Qin, and Y. Duan, "Thermal Modeling and Prediction of The Lithium-ion Battery Based on Driving Behavior," *Energies*, vol. 15, no. 23, 2022.

[201] J. Kleiner, M. Stuckenberger, L. Komsiyska, and C. Endisch, "Advanced Monitoring and Prediction of the Thermal State of Intelligent Battery Cells in Electric Vehicles by Physics-Based and Data-Driven Modeling," *Batteries*, vol. 7, no. 2, 2021.

[202] J. Kleiner, M. Stuckenberger, L. Komsiyska, and C. Endisch, "Real-time core temperature prediction of prismatic automotive lithium-ion battery cells based on artificial neural networks," *Journal of Energy Storage*, vol. 39, p. 102588, 2021.

[203] V. Antinyan, "Revealing the Complexity of Automotive Software," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. Virtual: Association for Computing Machinery, 2020, pp. 1525–1528.

[204] F. Falcini, G. Lami, and A. M. Costanza, "Deep Learning in Automotive Software," *IEEE Software*, vol. 34, no. 3, pp. 56–63, 2017.

[205] M. Staron, "Machine Learning in Automotive Software," in *Automotive Software Architectures: An Introduction*, M. Staron, Ed. Cham: Springer International Publishing, 2021, pp. 171–188.

[206] M. Rahimi, J. L. Guo, S. Kokaly, and M. Chechik, "Toward Requirements Specification for Machine-Learned Components," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. Jeju, Korea (South): IEEE, 2019, pp. 241–244.

[207] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden Technical Debt in Machine Learning Systems," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Montreal, Canada: Curran Associates, Inc, 2015.

[208] P. Obergfell, S. Kugele, and E. Sax, "Model-Based Resource Analysis and Synthesis of Service-Oriented Automotive Software Architectures," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Munich, Germany: IEEE, 2019, pp. 128–138.

[209] M. Taghavi and M. Shoaran, "Hardware Complexity Analysis of Deep Neural Networks and Decision Tree Ensembles for Real-time Neural Data Classification," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. San Francisco, CA, USA: IEEE, 2019, pp. 407–410.

[210] S. Bateni, Z. Wang, Y. Zhu, Y. Hu, and C. Liu, "Co-Optimizing Performance and Memory Footprint Via Integrated CPU/GPU Memory Management, an Implementation on Autonomous Driving Platform," in *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Sydney, NSW, Australia: IEEE, 2020, pp. 310–323.

[211] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and

I. Maglogiannis, Eds. Rhodes, Greece: Springer International Publishing, 2018, pp. 270–279.

[212] M. Weber, M. Auch, C. Doblander, P. Mandl, and H.-A. Jacobsen, "Transfer Learning With Time Series Data: A Systematic Mapping Study," *IEEE Access*, vol. 9, pp. 165 409–165 432, 2021.

[213] R. J. Borgli, S. H. Kvale, M. A. Riegler, and P. Halvorsen, "Automatic Hyperparameter Optimization for Transfer Learning on Medical Image Datasets Using Bayesian Optimization," in *2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*. Oslo, Norway: IEEE, 2019, pp. 1–6.

[214] L. Wang, X. Dong, Y. Wang, L. Liu, W. An, and Y. Guo, "Learnable Lookup Table for Neural Network Quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, Louisiana, USA, 2022, pp. 12 423–12 433.

[215] W. Fuhl, G. Kasneci, W. Rosenstiel, and E. Kasneci, "Training Decision Trees as Replacement for Convolution Layers," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3882–3889, 2020.

[216] R. Krishnan, G. Sivakumar, and P. Bhattacharya, "Extracting decision trees from trained neural networks," *Pattern Recognition*, vol. 32, no. 12, pp. 1999–2009, 1999.

[217] O. Boz, "Extracting Decision Trees from Trained Neural Networks," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. Edmonton, Alberta, Canada: Association for Computing Machinery, 2002, pp. 456–461.

[218] A. Jamali, S. J. Motevalli, and N. Nariman-zadeh, "Extracting fuzzy rules for modeling of complex processes using neural networks," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 227, no. 12, pp. 2861–2869, 2013.

[219] P. T. May-Ostendorp, G. P. Henze, B. Rajagopalan, and C. D. Corbin, "Extraction of supervisory building control rules from model predictive control of windows in a mixed mode building," *Journal of Building Performance Simulation*, vol. 6, no. 3, pp. 199–219, 2013.

[220] A. Domahidi, F. Ullmann, M. Morari, and C. N. Jones, "Learning near-optimal decision rules for energy efficient building control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. Maui, HI, USA: IEEE, 2012, pp. 7571–7576.

[221] C. Schröer, F. Kruse, and J. M. Gómez, "A Systematic Literature Review on Applying CRISP-DM Process Model," *Procedia Computer Science*, vol. 181, pp. 526–534, 2021.

[222] M. Tutuianu, P. Bonnel, B. Ciuffo, T. Haniu, N. Ichikawa, A. Marotta, J. Pavlovic, and H. Steven, "Development of the World-wide harmonized Light duty Test Cycle (WLTC) and a possible pathway for its introduction in the European legislation," *Transportation Research Part D: Transport and Environment*, vol. 40, pp. 61–75, 2015.

[223] United States Environmental Protection Agency, "Dynamometer Drive Schedules," Washington, D.C., United States of America, 2021, Accessed June 2, 2022. [Online]. Available: https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules

[224] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, vol. 79, no. 19, pp. 12 777–12 815, 2020.

[225] M. Ali, "PyCaret: An open source, low-code machine learning library in Python," 2020, Python package. Version 1.0. Accessed January 3, 2023. [Online]. Available: https://www.pycaret.org

[226] M. Kumar, "Scikit-Garden: Scikit-Garden: A Garden for Scikit-Learn Compatible Trees," 2017, Python package. Version 0.1. Accessed January 7, 2023. [Online]. Available: https://github.com/scikit-garden/scikit-garden

[227] A. Colin Cameron and F. A. Windmeijer, "An R-squared measure of goodness of fit for some common nonlinear regression models," *Journal of Econometrics*, vol. 77, no. 2, pp. 329–342, 1997.

[228] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, Jul. 2021. [Online]. Available: https://doi.org/10.7717/peerj-cs.623

[229] V. Plevris, G. Solorzano, N. P. Bakas, and M. E. A. Ben Seghier, "Investigation of performance metrics in regression analysis and machine learning-based prediction models," in *8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2022)*. Oslo, Norway: European Community on Computational Methods in Applied Sciences, 2022.

[230] P. K. Ozili, "The acceptable R-square in empirical modelling for social science research," in *Social research methodology and publishing results: A guide to non-native english speakers*, C. Saliya, Ed. Hershey, PA: IGI global, 2023, pp. 134–143.

[231] Y. Chung, I. Char, H. Guo, J. Schneider, and W. Neiswanger, "Uncertainty Toolbox: an Open-Source Library for Assessing, Visualizing, and Improving Uncertainty Quantification," 2021, Accessed February 20, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2109.10254

[232] C. Aldrich, "Process Variable Importance Analysis by Use of Random Forests in a Shapley Regression Framework," *Minerals*, vol. 10, no. 5, 2020.

[233] P. N. Ngatchou, A. Zarei, and M. A. El-Sharkawi, "Pareto Multi Objective Optimization," in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, Arlington, VA, USA, 2005, pp. 84–91.

[234] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-Learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[235] M. André, "The ARTEMIS European driving cycles for measuring car pollutant emissions," *Science of The Total Environment*, vol. 334-335, pp. 73–84, 2004.

[236] E. Tzirakis, F. Zannikos, and S. Stournas, "Impact of driving style on fuel consumption and exhaust emissions: defensive and aggressive driving style," in *Proceedings of the 10th international conference on environmental science and technology*, Kos, Greece, 2007, pp. 1497–1504.

[237] M. V. Prati, M. A. Costagliola, R. Giuzio, C. Corsetti, and C. Beatrice, "Emissions and energy consumption of a plug-in hybrid passenger car in Real Driving Emission (RDE) test," *Transportation Engineering*, vol. 4, p. 100069, 2021.

[238] N. Scafetta, A. Fortelli, and A. Mazzarella, "Meteo-climatic characterization of Naples and its heating-cooling degree day areal distribution," *International Journal of Heat and Technology*, vol. 35, no. Special Issue1, pp. S137–S144, 2017.

[239] Deutscher Wetterdienst, "Climate Data Center (CDC)," Offenbach, 2021, Accessed November 27, 2021. [Online]. Available: https://opendata.dwd.de/climate_environment/CDC/

[240] M. Kompf, "Distance calculation," 2023, Accessed March 15, 2023. [Online]. Available: https://en.kompf.de/gps/distcalc.html

## Karlsruher Schriftenreihe Fahrzeugsystemtechnik FAST Institut für Fahrzeugsystemtechnik (ISSN 1869-6058)

Eine vollständige Übersicht der Bände finden Sie im Verlagsshop

**Band 76** Kai-Lukas Bauer
**Echtzeit-Strategieplanung für vorausschauendes automatisiertes Fahren**
ISBN 978-3-7315-0949-3

**Band 77** Thomas Schirle
**Systementwurf eines elektromechanischen Fahrwerks für Megacitymobilität**
ISBN 978-3-7315-0995-0

**Band 78** Dominik Dörr
**Simulation of the thermoforming process of UD fiber-reinforced thermoplastic tape laminates**
ISBN 978-3-7315-0998-1

**Band 79** Dominik Robert Naake
**Simulation of damage mechanisms in weave reinforced materials based on multiscale modeling**
ISBN 978-3-7315-1005-5

**Band 80** Martin Hohberg
**Experimental investigation and process simulation of the compression molding process of Sheet Molding Compound (SMC) with local reinforcements**
ISBN 978-3-7315-1007-9

**Band 81** Benedikt Fengler
**Manufacturing-constrained multi-objective optimization of local patch reinforcements for discontinuous fiber reinforced composite parts**
ISBN 978-3-7315-1006-2

**Band 82** Johannes Masino
**Road Condition Estimation with Data Mining Methods using Vehicle Based Sensors**
ISBN 978-3-7315-1004-8

**Band 83** **11. Kolloquium Mobilhydraulik**
**10. September 2020, Karlsruhe**
ISBN 978-3-7315-1036-9

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

**Band 84** Felix Weber
**Beitrag zur Entwicklung von Konstantflusspumpen für Frischbeton unter genauerer Betrachtung der Dickstoffventile**
ISBN 978-3-7315-1037-6

**Band 85** 8. Fachtagung
**Hybride und energieeffiziente Antriebe für mobile Arbeitsmaschinen. 23. Februar 2021, Karlsruhe**
ISBN 978-3-7315-1071-0

**Band 86** Sebastian Fünfgeld
**Vorausschauende Regelung von Fahrzeugsystemen durch stochastische Vorhersage der Fahrzeugdynamik**
ISBN 978-3-7315-1060-4

**Band 87** Isabelle Charlotte Ays
**Development of a CO2e quantification method and of solutions for reducing the greenhouse gas emissions of construction machines = Entwicklung einer CO2e Quantifizierungsmethode und von Lösungen zur Reduzierung von Treibhausgasemissionen in Baumaschinen**
ISBN 978-3-7315-1033-8

**Band 88** Alexander Bernath
**Numerical prediction of curing and process-induced distortion of composite structures**
ISBN 978-3-7315-1063-5

**Band 89** Nils Bulthaupt
**Objektivierung des Schwingungskomforts schwerer Nutzfahrzeuge**
ISBN 978-3-7315-1075-8

**Band 90** Lars Brinkschulte
**Assistenzsysteme zur Reduktion des Schädigungsverhaltens von Komponenten einer mobilen Arbeitsmaschine**
ISBN 978-3-7315-1089-5

**Band 91** Dominik Dörr
**Adaptive Fahrhinweise für ein längsdynamisches Fahrerassistenzsystem zur Steigerung der Energieeffizienz**
ISBN 978-3-7315-1090-1

**Band 92** Jürgen Römer
**Steuerung und Regelung des Lenkradmoments durch Nutzung radselektiver Frontantriebe**
ISBN 978-3-7315-1104-5

---

**Band 93** Christian Riese
**Werkzeuge und Konzepte für die Untersuchung und Entwicklung zukünftiger Kfz-Bremssysteme**
ISBN 978-3-7315-1125-0

**Band 94** Yaoqun Zhou
**Dynamisches Bremsverhalten des Reifen-Fahrwerk-Systems**
ISBN 978-3-7315-1156-4

**Band 95** Stefan Haug
**Ganzheitliche Optimierung einer Axialkolbenpumpe durch bedarfsangepasste Entlastung tribologischer Kontakte**
ISBN 978-3-7315-1150-2

**Band 96** Stefan Scheubner
**Stochastic Range Estimation Algorithms for Electric Vehicles using Data-Driven Learning Models**
ISBN 978-3-7315-1166-3

**Band 97** Yusheng Xiang
**AI and IoT Meet Mobile Machines: Towards a Smart Working Site**
ISBN 978-3-7315-1165-6

**Band 98** Nils Meyer
**Mesoscale simulation of the mold filling process of Sheet Molding Compound**
ISBN 978-3-7315-1173-1

**Band 99** Christian Timo Poppe
**Process simulation of wet compression moulding for continuous fibre-reinforced polymers**
ISBN 978-3-7315-1190-8

**Band 100** Torben Fischer
**Modellprädiktive Regelung eines innovativen Thermomanagement-Systems für batterieelektrische Fahrzeuge**
ISBN 978-3-7315-1199-1

**Band 101** Florian Wittemann
**Fiber-dependent injection molding simulation of discontinuous reinforced polymers**
ISBN 978-3-7315-1217-2

**Band 102** Sebastian Watzl
**Experimentelle und numerische Analyse des Körperschallübertragungsverhaltens von Aggregatelagerelementen im akustisch relevanten Frequenzbereich**
ISBN 978-3-7315-1226-4

---

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

**Band 103** Dominik Stretz
**Vibroakustische Analyse eines elektrischen Radnabenmotors und Optimierung durch geeignete Steuerungsansätze**
ISBN 978-3-7315-1245-5

**Band 104** Mohamed Elgharbawy
**Measurable Safety of Automated Driving Functions in Commercial Motor Vehicles - Technological and Methodical Approaches**
ISBN 978-3-7315-1254-7

**Band 105** Bernhard Schmiedel
**Indirekte Schätzung des Fahrbahnnässegrads zur Detektion von gefährlichen Fahrzuständen**
ISBN 978-3-7315-1258-5

**Band 106** 9. Fachtagung
**Hybride und energieeffiziente Antriebe für mobile Arbeitsmaschinen. 28. Februar 2023, Karlsruhe**
ISBN 978-3-7315-1260-8

**Band 107** Patrick Riehm
**Zur Wechselwirkung zwischen Fahrbahntextur und Laufstreifenmischung von Pkw-Reifen**
ISBN 978-3-7315-1268-4

**Band 108** Markus Tesar
**Deep Reinforcement Learning zur Steigerung von Energieeffizienz und Pünktlichkeit von Straßenbahnen**
ISBN 978-3-7315-1277-6

**Band 109** Michael Mürken
**Methode zur Bewertung der Zuverlässigkeit der elektrischen Energieversorgung in der automobilen Vorentwicklung**
ISBN 978-3-7315-1298-1

**Band 110** Julien Pinay
**Experimental investigation of relevant road surface descriptors for tire-road noise measurements on low-absorbing road surfaces**
ISBN 978-3-7315-1328-5

**Band 111** Adrian Strigel
**Methode zur Ermittlung optimaler Rad- und Reifendimensionen in der frühen Entwicklungsphase von Personenkraftwagen**
ISBN 978-3-7315-1321-6

**Band 112** Jens Jauch
**Trajectory optimization based on recursive B-spline approximation for automated longitudinal control of a battery electric vehicle**
ISBN 978-3-7315-1332-2

**Band 113** Nicolas Fraikin
**Methodik zur effizienten Applikation automatisierter Fahrfunktionen**
ISBN 978-3-7315-1339-1

**Band 114** Jan Siebert
**Effizienzoptimierung mobilhydraulischer Load-Sensing-Systeme durch Reduzierung systembedingter Druckverluste am Beispiel eines Hydraulikbaggers**
ISBN 978-3-7315-1343-8

**Band 115** Tobias Sebastian Straub
**Flottendatenbasierte physikalische Routenenergiebedarfsprognose**
ISBN 978-3-7315-1348-3

**Band 116** Michael Herrmann
**Eine Methodik zur Definition von Zielkriterien am Beispiel des tieffrequenten Geräuschkomforts eines Fahrzeugs**
ISBN 978-3-7315-1370-4

**Band 117** Adam Thor Thorgeirsson
**Probabilistic Prediction of Energy Demand and Driving Range for Electric Vehicles with Federated Learning**
ISBN 978-3-7315-1371-1

**Band 118** Alexander Jackstadt
**Constrained-layer damping in hybrid fibre metal elastomer laminates and its tolerance to damage**
ISBN 978-3-7315-1376-6

**Band 119** **13. Kolloquium Mobilhydraulik**
**8./9. Oktober 2024, Karlsruhe**
ISBN 978-3-7315-1381-0

**Band 120** Toni Wilhelm
**Querdynamik von Velomobilen unter Berücksichtigung der Reifeneigenschaften**
ISBN 978-3-7315-1382-7

**Band 121** Alexander Brunker
**Hochgenaue und robuste odometriebasierte Lokalisierung in einem Parkvorgang**
ISBN 978-3-7315-1383-4

**Band 122** Daniel Förster
**Systemauslegung autarker Hybridantriebe unter Berücksichtigung kundenspezifischer Randbedingungen**
ISBN 978-3-7315-1384-1

**Band 123** Maurizio Mauro Festa
**Objektive Fahrkomfortbewertung des Gesamtfahrzeugs unter Berücksichtigung menschlicher Wahrnehmungsaspekte**
ISBN 978-3-7315-1392-6

**Band 124** 10. Fachtagung
**Hybride und energieeffiziente Antriebe für mobile Arbeitsmaschinen. 19. Februar 2025, Karlsruhe**
ISBN 978-3-7315-1403-9

**Band 125** 8. Fachtagung
**MOBILE MACHINES – Sicherheit und Fahrerassistenz für Arbeitsmaschinen. 18. Februar 2025, Karlsruhe**
ISBN 978-3-7315-1404-6

**Band 126** Raphael Mieth
**Kundenbedarfsgerechte Auslegung elektrischer Pkw-Antriebssysteme zur Steigerung der Energieeffizienz**
ISBN 978-3-7315-1416-9

**Band 127** Lukas Michiels
**A simulation-based approach to the fluid-structure interaction inside fatigue cracks in hydraulic components**
ISBN 978-3-7315-1424-4

**Band 128** Felix Deufel
**Optimales Energiemanagement mild elektrifizierter Antriebe unter realen Betriebsbedingungen mittels Prädiktionsalgorithmen aus dem Bereich des Maschinellen Lernens**
ISBN 978-3-7315-1426-8

**Band 129** Andreas M. Billert
**Predictive Battery Thermal Management of Electric Vehicles using Deep Learning**
ISBN 978-3-7315-1429-9

Karlsruher Schriftenreihe
**Fahrzeugsystemtechnik**

FAST
*Institut für Fahrzeugsystemtechnik*

ISSN 1869-6058
ISBN 978-3-7315-1429-9