PADERBORN UNIVERSITY

FACULTY OF COMPUTER SCIENCE,
ELECTRICAL ENGINEERING AND
MATHEMATICS

Data-driven thermal modeling of a permanent magnet synchronous motor with machine learning

Von der Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Wilhelm Kirchgässner, M. Sc.

Erster Gutachter:	Prof. Dr.-Ing. Oliver Wallscheid
Zweiter Gutachter:	Prof. Dr.-Ing. Oliver Nelles
Dritter Gutachter:	Prof. Dr.-Ing. Joachim Böcker

Tag der mündlichen Prüfung: 25. Juli 2024

Paderborn 2024

Diss. EIM-E/378

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr.-Ing. Oliver Wallscheid, whose exceptional mentorship and continuous guidance played a pivotal role in the successful completion of this dissertation. His insightful feedback, patience, and dedication have been indispensable in refining my research methodology, enhancing the quality of my work, and expanding my understanding of the subject matter. I am particularly indebted to him for his unwavering belief in my potential and his continuous encouragement to push the boundaries of my research.

I would also like to extend my sincere appreciation to Prof. Dr.-Ing. Joachim Böcker, my esteemed professor, for his invaluable guidance, support, and encouragement throughout the journey of completing this dissertation. His expertise, profound knowledge, and commitment to academic excellence have been instrumental in shaping my research and broadening my horizons in the field of electrical engineering. I am thankful for the freedom I was granted in my research and for the engaging environment I could work in.

I am grateful to all the members of my dissertation committee – and particularly Prof. Dr.-Ing. Oliver Nelles – for their valuable insights, constructive feedback, and critical evaluation of my research.

I would like to thank my colleagues and fellow researchers for their stimulating discussions, collaboration, and support throughout this endeavor. Much appreciation in particular goes to Emebet Gedlu, Darius Jakobeit, Barnabas Haucke-Korber, and Maximilian Schenke for proofreading this work. Their expertise and scholarly contributions have greatly enriched the content of this dissertation. Moreover, I would like to thank Anian Brosch for being an incredible office colleague in the pre-pandemic times, providing a supportive and collaborative environment that fostered productivity and intellectual growth.

My heartfelt appreciation extends to my family and to a large degree to my wife for their unconditional love, encouragement, and understanding during this demanding academic pursuit. Their belief in my abilities and undeviating support have sustained me throughout the ups and downs of this journey.

Finally, I would like to express my gratitude to all the individuals, institutions, and organizations that have contributed to this research in any way. This includes the Deutsche Forschungsgemeinschaft (DFG) to a large extent for funding most of this work. Computational resources were provided by the Paderborn Center for Parallel Computing (PC^2).

Completing this dissertation would not have been possible without the contributions and support of all these individuals. Their collective efforts have shaped the outcome of this research and enriched my academic and personal growth. I am truly grateful for their invaluable contributions.

Abstract

The permanent magnet synchronous motor (PMSM) is a commonly used traction motor in automotive applications due to its high power and torque density with respect to volume and weight. Electric vehicle drive scenarios are characterized by dynamically changing operation points where long periods of steady load are intermitted by bursts of high power demand. This bears the consequence of an evolving thermal load on vital motor components such as the stator windings or the permanent magnets embedded in the rotor. The overheat thereof is harmful as winding insulations may melt and permanent magnets may irreversibly demagnetize. In order to prevent the motor's abrupt end of life, incorporating more material denotes a costly amendment with an increase in volume and weight, which is why monitoring-based power derating is the more desirable path to follow. A sensor-based temperature monitoring is not economically feasible at least for the rotor components due to the required high-quality, contactless, and EMI-hardened measurement instruments. On the other hand, estimation methods such as model-based approaches can alleviate the problem of missing thermal information at potentially no additionally required equipment.

This work collates a portfolio of data-driven thermal models from the domain of machine learning and investigates their feasibility for the task of accurate thermal modeling on the example of a PMSM data set recorded on a test bench. Aside from the average estimation error, the required amount of model parameters as an approximation for the computational demand dictates design decisions throughout. The whole process of designing a machine learning model is illuminated and carried out for varying linear models; tree-based models; feed-forward, recurrent, and convolutional neural networks, as well as various hybrid gray-box modeling approaches. Moreover, a hybrid modeling paradigm with thermal neural networks (TNNs) is highlighted, which was first introduced by this work's author. Eventually, an expert-designed, data-driven lumped-parameter thermal network (LPTN) is optimized under different algorithms in order to put machine learning models to the test against the state of the art of thermal modeling.

It is empirically found that tree-based models denote the lowest estimation performance to number of parameters ratio, followed by linear regression, feed-forward, and convolutional neural networks. In contrast, certain recurrent topologies such as gated recurrent units and TNNs achieve the highest estimation accuracy, while the latter also approaches the low number of parameters exhibited by LPTNs. It is concluded that the TNN inherits most advantages among all modeling paradigms including accurate estimation; low computational demand due to few model parameters; efficient identification on large data sets through automatic differentiation and on small data sets through a constrained search space; physical interpretability of model states; consistency with heat transfer principles; recovery from detuned initialization; scalable performance with model complexity up to some extent; and requiring neither material, geometry, nor power analyzer information.

Zusammenfassung

Der Permanentmagnet-Synchronmotor (PMSM) ist aufgrund seiner hohen Leistungs- und Drehmomentdichte bezogen auf Volumen und Gewicht ein häufig verwendeter Traktionsmotor in Automobilanwendungen. Das typisches Fahrprofil eines Elektrofahrzeugs wird durch dynamisch wechselnde Betriebspunkte gekennzeichnet, bei denen lange Perioden gleichmäßiger Belastung von Leistungsspitzen unterbrochen werden. Dies führt zu einer nicht unerheblichen thermischen Belastung der wichtigsten Motorkomponenten, wie z. B. der Statorwicklungen oder der im Rotor eingebetteten Permanentmagneten. Im Falle einer Überhitzung können die Wicklungsisolierungen schmelzen und die Permanentmagnete irreversibel entmagnetisiert werden. Um ein abruptes Ende der Motorlebensdauer durch Überhitzung zu vermeiden, wird daher das verbaute Material überdimensioniert. Dies stellt jedoch eine kostspielige Erhöhung des Volumens und Gewichts dar, weshalb ein überwachendes Leistungsderating der wünschenswertere Weg ist. Eine sensorgestützte Temperaturüberwachung ist jedoch insbesondere bei Rotorkomponenten nicht wirtschaftlich, da hierfür kontaktlose Messinstrumente mit hoher Qualität und EMV-Robustheit benötigt werden. Auf der anderen Seite sind Schätzverfahren wie modellbasierte Ansätze potentiell in der Lage, das Problem der fehlenden Temperaturinformation zu relativieren, ohne zusätzliche Geräte zu erfordern.

Diese Arbeit stellt ein Portfolio von thermischen Modellen aus dem Bereich des maschinellen Lernens zusammen. Die Untersuchung basiert auf einem PMSM-Datensatz, der auf einem Prüfstand aufgezeichnet wurde. Neben dem durchschnittlichen Schätzfehler diktiert die erforderliche Anzahl von Modellparametern zahlreiche Auslegungsentscheidungen. Der gesamte Entwurfsprozess eines Modells aus dem maschinellen Lernen wird beleuchtet und für verschiedene lineare, sowie baumbasierte Modelle; vorschiebende, rekurrente und faltende neuronale Netze als auch für verschiedene hybride Modellierungsansätze durchgeführt. Desweiteren wird der hybride Modellierungsansatz über thermische neuronale Netze (TNNs) besonders hervorgehoben. Sie setzen sich aus neuronalen Netzen und einem thermischen Ersatzschaltbild zusammen und wurden erstmals vom Autor dieser Arbeit veröffentlicht. Schließlich wird ein von Experten entworfenes, datengetriebenes thermisches Netz mit konzentrierten Parametern (LPTN) über verschiedene Algorithmen optimiert und als Stand der Technik herangezogen.

Es zeigt sich empirisch, dass baumbasierte Modelle die geringste Schätzgüte im Verhältnis zur Anzahl der Parameter aufweisen, gefolgt von linearer Regression, sowie vorschiebenden und faltenden neuronalen Netzen. Im Gegensatz dazu erreichen bestimmte rekurrente Topologien wie Gated Recurrent Units und TNNs die höchste Schätzgenauigkeit, während sich letztere auch der geringen Anzahl von Parametern bei LPTNs annähern. Es wird geschlussfolgert, dass das TNN die meisten Vorteile gegenüber allen anderen Modellierungsparadigmen aufweist. Es vereint eine genaue Schätzung mit geringem Rechenaufwand und effizienter Modellidentifikation bei großen als auch kleinen Datensätzen durch automatisches Differenzieren und durch einen eingeschränkten Suchraum. Ferner bietet es die physikalische Interpretierbarkeit von Modellzuständen, Konsistenz mit der Wärmelehre, Robustheit gegenüber Fehlinitialisierung, Skalierbarkeit der Schätzgüte über Modellkomplexität bis zu einem gewissen Grad und die Tatsache, dass weder Material-, Geometrie- noch Verlustleistungsvermessungen erforderlich sind.

Acronyms, symbols, and notation

Acronyms **Page**

Symbols

α	Coefficient or weight factor or $\mathcal{K}$-function (context-dependent)
β	Regression coefficient
γ	Thermal conductivity sub-ANN output
ϵ	Error threshold
ε	Rotation angle
$\boldsymbol{\zeta}$	Scheduling vector
ϑ	Temperature (Celsius)
θ	Free model or electrical parameter (context-dependent)
κ	Electric conductivity or thermal capacitance (context-dependent)
$\boldsymbol{\kappa}$	Thermal capacitances sub-ANN output
$\boldsymbol{\lambda}$	Directional thermal conductivity matrix
λ	Penalty factor
μ	Population mean
$\boldsymbol{\nu}$	Hyperparameter vector
ξ	Slack variable or additional observables vector or $\mathcal{KL}$-function (context-dependent)
π	Probability function or power losses sub-ANN output (context-dependent)
ρ	Material density coefficient or $\mathcal{K}$-function (context-dependent)
σ	Sigmoid activation function or population standard deviation (context-dependent)
τ_s	Variable of integration for the discrete-time input matrix
$\boldsymbol{\Phi}$	Discrete-time transition matrix
ϕ	Angular coordinate
χ	$\mathcal{K}$-function
$\boldsymbol{\psi}$	Magnetic flux linkage
ψ	Tree function

ψ_{PM}	Magnetic flux linkage caused by the permanent magnets
ω	Angular velocity
A	Area
$\boldsymbol{A}$	System matrix
$\boldsymbol{a}$	Adjoint (gradient of a loss function w.r.t. a system state vector)
$\boldsymbol{B}$	Magnetic flux density or input matrix (context-dependent)
$\mathcal{B}$	Mini-batch set
C	Scalar thermal capacitance
$\boldsymbol{C}$	Output matrix
c	Thermal capacitance function
$\mathcal{D}$	Data set
$\boldsymbol{D}$	Rotation Matrix for the PMSM voltage equation or displacement matrix (context-dependent)
d	Distance or drag (context-dependent)
$\boldsymbol{E}$	Electric field
E	Energy or noise random variable (context-dependent)
$\dot{E}_{\mathrm{mag}}$	Time-derivative of the stored magnetic energy
e	Scalar error
$\boldsymbol{f}$	Arbitrary function
$\mathcal{G}$	Generalization set
$\boldsymbol{G}$	Thermal conductance adjacency matrix
$\boldsymbol{g}$	Arbitrary function or thermal conductance (context-dependent)
H	Magnetic field strength or Heaviside function (context-dependent)
$\boldsymbol{H}$	Discrete-time input matrix
$\boldsymbol{h}$	Intermediate ANN layer output vector
$\boldsymbol{I}$	Identity matrix
I_{s}	Stator current magnitude
$i_{\mathrm{a,b,c}}$	Phase currents
J	Moment of inertia
j	Imaginary number
K	Data set size or total amount of time steps
$\boldsymbol{K}$	Linear feedback matrix
k	Tuning factor/ free model parameter or time step (context-dependent)
L	Inductance or length of measurement session (context-dependent)
$\mathcal{L}$	Cost function
l_h	Height
M	Number of measurement sessions
$\boldsymbol{m}$	Electrical parameter function
m	Number of nodes in a LPTN
$\mathcal{N}$	Gaussian distribution
n	Rotational motor speed or number of ancillary temperatures (context-dependent)
P	Thermal power loss or number TBDof independent variables (context-dependent)
P_{el}	Input electric power
$P_{\mathrm{l,\,CU}}$	Ohmic power loss
P_{me}	Mechanical Power
p	Pole pair number or thermal energy conversion rate from different sources (context-dependent)
$\boldsymbol{Q}$	Rotation matrix of the Park-Transform
Q	Number of target/dependent variables
q	Number of stator slots or number of output temperatures (context-dependent)
R	Scalar thermal resistance or random number (context-dependent)
R_{s}	Stator resistance

r	Radial coordinate or thermal resistance (context-dependent)
r_d^2	Coefficient of determination
S	Eddy current loss density or apparent power (context-dependent)
s	Span value in the context of EWMAs
$\boldsymbol{s}$	Switching states
$\mathcal{T}_{\text{e/r}}$	Test / training set
T	Air gap torque
T_L	Load torque
T_s	Sample time
$\boldsymbol{T}_{23}/\boldsymbol{T}_{32}$	Rotation matrices of the Clarke-Transform
t	Time
$\boldsymbol{U}$	Regressor matrix
$\boldsymbol{u}$	System input vector
u	System input quantity (element of system input vector)
$\mathcal{V}$	Validation set
V	Volume or ISS-Lyapunov function (context-dependent)
$v_{\text{a,b,c}}$	Phase voltage
$\boldsymbol{W}$	Weight matrix
w	Free model parameter
x	Spatial coordinate
$\boldsymbol{x}$	System state vector
$\boldsymbol{Y}$	Matrix of dependent variables
$\boldsymbol{y}$	System output vector
z	Axial coordinate
$\mathbb{1}_n$	Column vector with all entries being equal to one

Notation

x	Scalar quantity
X	Scalar quantity or random number
$\boldsymbol{x}$, $\boldsymbol{X}$	Vector, matrix
$\underline{x}$, $\underline{X}$	Complex quantity
$\boldsymbol{x}^\mathsf{T}$, $\boldsymbol{X}^\mathsf{T}$	Transposed vector/matrix
$\boldsymbol{x}_{\alpha\beta}$, $\boldsymbol{X}_{\alpha\beta}$	Transformed quantity in $\alpha\beta$ coordinate system (stator-fixed)
$\boldsymbol{x}_{\text{dq}}$, $\boldsymbol{X}_{\text{dq}}$	Transformed quantity in dq coordinate system (rotor-fixed)
$\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{X}}$	Estimated quantity
$\dot{x}$, $\dot{X}$	Time derivative of a quantity
$\|x\|_2$	Euclidean norm
$\|x\|_\infty$	Maximum norm

Contents

1 Introduction

The (battery) electric vehicle (EV) thrusts the internal combustion engine (ICE) aside and gains an increasing momentum in the automotive market thanks to the recent advancement in energy- and power densities of traction batteries as well as political backing. As a highly competitive market, the automotive sector features products where manufacturing and development costs allow for only thin margins. Accordingly, revenue has to be generated through mass production and high-volume sales. The keen competition fuels research effort in the steady improvement of EVs in terms of, e.g., power output, driving range, the quality and cost-efficient manufacturing of traction drive components but also more intelligent monitoring thereof.

Among the factors that limit an EV drive's power output, the thermal stress of temperature-sensitive components can be denoted as especially crucial. Due to the intricate and compact construction of an electric motor, those components' temperature is not measurable on a sensor basis within economically feasible constraints. In order to increase the thermal overload capabilities of an EV motor despite the lack of thermal state information, more material would be incorporated commonly, which, however, deteriorates production cost efficiency, and the additional weight taxes the driving range. For decades there exists, thus, an academic and industrial endeavor to estimate the thermal state of critical motor components.

One such approach is denoted by model-based estimation algorithms. The main challenge for these models is the accurate estimate, while being computationally lightweight at the same time. Latter requirement stems from the best-cost hardware (mostly microcontrollers) that is deployed as electronic control unit (ECU) in an EV in order to further reduce production cost. Reducing the amount of parameters in an approximating system model to a minimum virtually always leads to abstractions of the physical phenomena within the system. Deriving correct model parameters under severe abstractions from first principles alone becomes increasingly difficult, such that empirical (measurement) data is often considered to aid the parameter identification. The time and effort to obtain measurement data is not negligible in the automotive sector, but it can still be worthwhile when it enables a substantial improvement in thermal state estimation and, hence, material reduction potential.

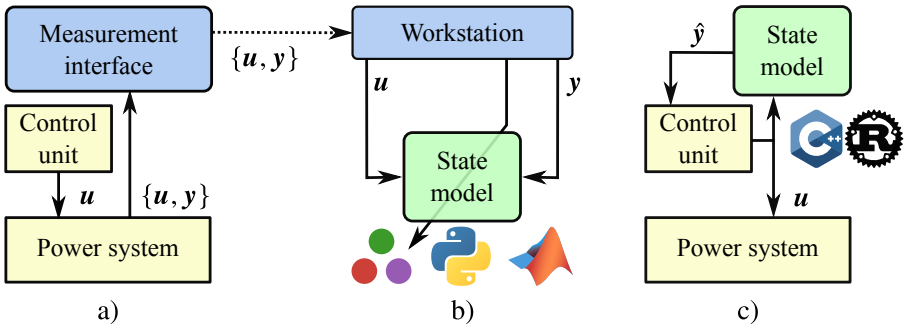


Figure 1.1: The identification process from (a) data acquisition over (b) model training to the subsequent (c) deployment in the field (series applications) is shown (cf. [KWB23]). $\boldsymbol{u}$ stands for the system control input, whereas $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ denote measured and estimated states, respectively.

The process of creating and employing a thermal system state model in a data-driven fashion is sketched in Fig. 1.1. With the aid of a test bench and a mounted prototype motor under test, sensor readings can be captured under varying operation conditions. Several thermo-element sensors are usually embedded into the

motor under test already during the manufacturing process to obtain a measurement-based ground truth, which denotes the best approximation of the real temperature development in a series production variant. With the information of what is available during field operation and what should be estimated in the form of a data set, the analysis and modeling process can start. Most of this thesis' content deals with this process. Typical programming languages for this phase are Julia, Python, R, or MATLAB, where rapid development cycles and quick prototyping feedback is desired. Eventually, when an appropriate thermal model was identified, it is to be deployed as a companion in the control unit informing the control algorithm about the current thermal situation in real time. In this phase, the model is to be converted into C/C++ or Rust code in order to be cross-compilable for use in a microcontroller. The computational requirements for the mere execution (inference) of an identified system model are, after all, less demanding in general than the identification of its parameters.

In this thesis, a few investigations into load cycle planning for the preparation of test bench measurements are carried out, which is followed by the separation of the resulting recordings into reasonable training and validation sets for the later cross-validation through unsupervised, data-driven clustering methods. The main part of this thesis, however, deals with the data-driven thermal modeling of a permanent magnet synchronous motor (PMSM) with machine learning, and especially with automatic differentiation, when all data and cross-validation schemes are in place. This includes not only the choice of the model category, but also the normalization of the data, its augmentation with additional features, the choice of hyperparameters that impact model topology and parameter optimization, and the trade-off between accuracy and model size measured in the number of model parameters required. In contrast to earlier work, this thesis delves into the thermal system identification not only through black-box machine learning models such as linear regression and neural networks, but additionally through hybrid gray-box models. These incorporate heat transfer principles directly into the model such that physical consistency is ensured, while all parameters remain end-to-end differentiable, and, thus, optimizable with efficient gradient descent optimization techniques. The comprehensive comparison of data-driven thermal models at varying degrees of domain knowledge use denotes the eventual conclusion of this work.

In Sec. 2, the societal and technical role of the PMSM and electric mobility in general is discussed together with the main challenges of its design, operation and monitoring. In Sec. 3, the formal description of the electromagnetic and thermal phenomena in a PMSM follows, which also illuminates the recent state of the art of thermal modeling. Then, the general state of machine learning in academia and industry alike as well as the most important black-box modeling techniques are outlined in Sec. 4. The model-agnostic preparation and analysis of the data set, which represents the fundamental base of all investigations in this work, is conducted in Sec. 5. The core experiments of this thesis can be found in Sec. 6 and Sec. 7 for pure black-box model and gray-box model approaches, respectively, before this thesis concludes in Sec. 8.

2 Automotive electric traction drives

In order to be able to discuss the relevance of this work properly, the role of thermal modeling in electric vehicles (EVs) among other challenges around their application has to be illuminated. Particularly, the state of public affairs; a systematic overview of the components involved in an electric drive train; prevalent control strategies; and the relevance of thermal stress and its monitoring are treated briefly in this chapter.

2.1 Market and societal significance

The market around commercial vehicles puts ever-increasing pressure on competitors, who seek to sustain their position by research and development investments and perpetually reducing production costs. The EV technology was historically a niche in the automotive market [Cha02]: As a direct competitor to the long-established internal combustion engine (ICE), it drew only marginal interest at the majority of original equipment manufacturers (OEMs). The promise of EVs for an emission-free and sustainable transportation faced the lack of meeting customer demand with regard to transportation range per charge, high initial purchasing costs, scarce charging facilities, and long recharge durations [Cha07]. These core challenges were still addressed just a decade ago in a Belgium-based survey [Leb+13], when consumers were asked for their expectations of feasible EVs.

Recently, however, EV technology emerged to inevitably become the prospective automotive market leader [Van+21]: with the aid of technological advances in energy sources and power electronics, energy densities of contemporary EV batteries have doubled with respect to a decade ago. It is expected to further increase at the same rate, while simultaneously battery prices plummeted by around 90 % due to accelerated mass production and careful choice of materials. Furthermore, the already high efficiency of inverters was further improved. The sum of these trends benefit the EV's driving range and charging times, and further relativizes the advantages of ICEs over EVs. Moreover, in an attempt to meet the decarbonization targets of increasingly strict European Union (EU) resolutions, many European countries offer more and more tax reliefs and monetary incentives for the purchase and operation of EVs [Tuc+19; Árp+21]. Even more so, the EU recently argued for a complete production stop of ICEs by 2035 and many OEMs share this objective already [Zei21]. However, the greenhouse-emission-free transportation over the vehicle's life cycle was shown to also depend on energy generation sources of the respective country [ZP21], which is unaffected by vehicle technology.

Irrespective of the mentioned technological advances and political steering, the EV comprises several inherent advantages over classical ICEs [Mal16; Van+21; SK21]:

- no CO_2 emissions during operation, helping to meet environmental goals such as reduction of air pollution,
- substantially higher tank-to-wheel and well-to-wheel efficiency,
- high acceleration ability since maximum torque is applicable during standstill,
- operations and maintenance cost savings due to fewer moving parts, and less brake attrition due to regenerative braking (recuperation),
- significant reduction of acoustic operation noise,
- the possibility to stabilize and supply the grid by having the EV participate as energy storage.

Further, societal benefits were found to be fuel savings because of the lower cost per kilometer distance travelled with propulsion from electric energy sources; alleviated harm to public health due to reduced fine particle pollution; national security costs savings due to less dependencies on protected access to Middle East oil that is to be supported by military operations; and local economic development improvements that root in avoiding the local economy's low share in the current oil price, among others [Mal16]. Several future trends can be already identified that will leverage the emergence of EVs [Van+21]: More improvements in the domain of battery technology through solid-state electrolytes, in-cell sensors for improved state-of-health analysis, as well as expanding into self-healing properties is expected to enhance driving range, efficiency, and battery robustness. The powertrain design domain is experiencing a new enabling tool for optimizing efficiency and reliability with digital twins. Moreover, not only the powertrain's design but also its operation sees paradigm shifts through algorithmic applications due to advancements in embedded system development [SK08; Jou+18]. Last, power electronic materials of higher capabilities, classified as wide bandgap, find their ways into industry production, which can further reduce power inefficiencies in both, EV operation as well as charging. These future advances bear good prospects of further consolidating the EV's position and adoption in the market.

2.2 System overview of an electric drive train

In a typical drive setup, as seen in Fig. 2.1, the electric machine's main purpose is to convert electrical energy from a power source to mechanical energy and in certain cases also vice versa [DPV20]. The (embedded) controller is a platform for (nowadays digital) signal processing and control algorithms. The latter are dedicated to control the mechanical load exerted on the shaft through setting switching schemes in the converter (potentially via a modulator) and reacting on electrical and mechanical sensory feedback. The control algorithm's performance is usually evaluated on how accurate and fast the mechanical process is steered to the reference signals coming from a higher order system.

In EVs the power source is usually a high-voltage battery and the mechanical load mainly the resistive forces that apply at the tire-to-road contact. In automation applications the power source can also be rendered by a grid's power lines and the load by a mechanical step in an industrial process (e.g., positioning of tools or robot arms in a manufacturing process). It is characteristic of automotive applications that energy flow is occasionally reversed such that it is directed from the load to the power source, for instance during braking maneuvers, which is also termed as recuperation. The energy flow from power source over converter and electric motor to the mechanical shaft is, therefore, bi-directional and all involved mechanical and power electric components are to be designed correspondingly. Another noteworthy characteristic of automotive applications is the sporadic, sudden demand for high power (power spikes), which the drive components have to withstand. This leads to the economic challenge of determining an efficient dimensioning of all involved components under certain budget and safety constraints.

There are various drive train topologies that involve multiple electric motors or their combination with ICEs in a single hybrid electric vehicle (HEV) and balance advantages and disadvantages of the corresponding motor types through a more flexible per-machine operation [Cha02; San+12; Tra+20a]. Though the specific topology may have a great impact on overall EV efficiency, production cost, and drive behavior, it is irrelevant for this work's investigation, which focuses on operation-point-dependent thermal modeling of electric machines where the full operation domain is taken into account.

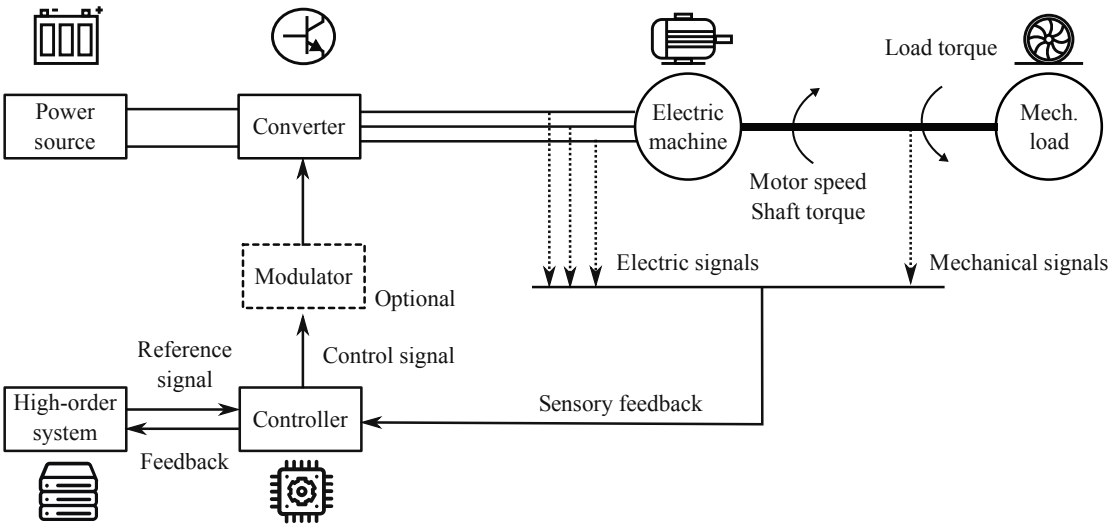


Figure 2.1: Simplified scheme of a drive train (derived from: [DPV20])

2.3 Requirements and challenges

A high-grade electric drive has to comply with a multitude of requirements in order to be eligible to enter series production for an application domain as demanding as the automotive sector. Only careful design and engineering work might lead to a successful drive train complying with at least all of the following [Cha02; Cha07; ZH07; Cha+17]:

- a high torque and power density with respect to mass and volume,
- high torque at low driving speeds, start, or up-hill drive, and high power during cruise operation,
- high velocities shall be covered by the motor's speed range in order to avoid an intricate manual gearbox,
- high-dynamic and precise torque control,
- high efficiency with low acoustic noise throughout most operation points, especially at base speed,
- reliability and robustness, overload-capability for short durations, but low maintenance,
- automatic traverse into safe-state in case of fault conditions (functional safety),
- low production cost.

Meeting these requirements does not cease to challenge industry and academia alike. Many diverse branches of research treat the peculiarities involved in designing suitable electric drive components, that, e.g., encompass control strategies and choice of material. Especially challenging is the fact that, during the design phase of an electric motor, not only many dynamic aspects are to be contemplated but also their interplay [Bil+19; San+12]. This makes research and development around electric drives a sophisticated, challenging, and typically time-consuming endeavor. The electromagnetic behavior, for instance, is determined by switching schemes dictated by elaborate control strategies acting on power electronic converters; the skin effect; spatial harmonics induced by slotting; and cross-coupling of different current pairs. The thermal dynamics become difficult to grasp due to sporadically spiking, distributed power losses that run across varying cooling conditions - all significantly dependent on the load condition that can vary in unpredictable ways. Mechanical stress is the result of torque, rotation speed, and electromagnetic forces. Related to this analysis, acoustic noise and vibration are further factors worth to be investigated to improve drivers' comfort. In order to get hold of these complexities, rigorous modeling concepts were developed in the past to a considerable technical depth.

In Sec. 3, the fundamentals of electromagnetic and thermal modeling are outlined. For an overview of current modeling paradigms for acoustics and mechanical stress in electric machines, see [Bil+19].

2.4 Electric motor types

Although this thesis deals with thermal modeling on the example of a permanent magnet synchronous motor (PMSM), most concepts that are presented can also be applied on other motor types (and even on other application domains, such as power electronic modules). Thus, a brief overview of different electric motor types is given in this chapter.

A qualitative comparison between brushless direct current (BLDC) machines, induction machines (IMs) / asynchronous motors (ASMs), and switched reluctance machines (SRMs) is given in [ZH07], while a cursory review of most contemporary electric motor types including the synchronous reluctance motor (SynRM) can be found in [Cha+17; RL19]. Fig. 2.2 shows a simplified sketch of the major electric motor types that are employed in the automotive industry. Most motor types are arranged with distributed windings due to lower losses of various kinds. The following motor types all work on the radial flux principle, however, it should not go unmentioned that there are also efforts to further cost savings with magnetless, axial-flux types [Lee+14].

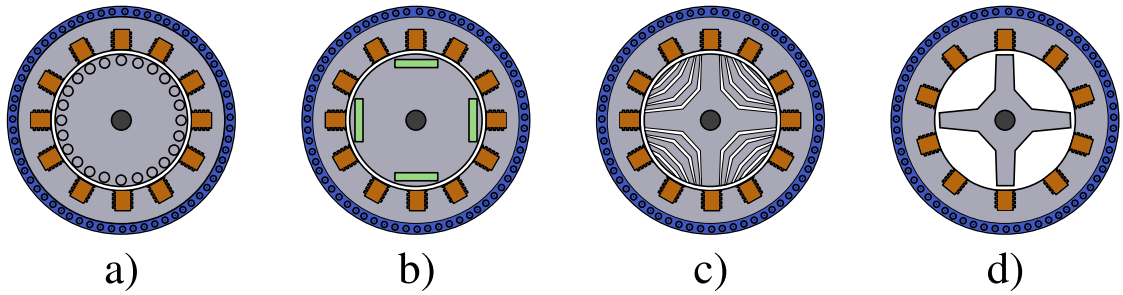


Figure 2.2: The cross-section of different electric motor types is shown schematically. a) The IM features several rotor bars around the rotor's circumference. b) The buried permanent magnets are evident in green for the two-pole PMSM. c) Rotor cavities following magnetic field lines characterize the SynRM. d) A SRM's rotor features salient poles and uneven rotor pole to stator pole ratios.

Advantages and disadvantages

An IM/ASM is the most reliable and robust type with well established manufacturing techniques at relatively low cost. The squirrel cage induction motor is an IM often used in automotive applications. It encompasses a rotor with steelbars in axial direction, and is simpler to control relative to other IM types. Through feeding the IM by an inverter, flux and torque can be controlled in a highly efficient manner, which can even consider acoustic noise attenuation. There is no dependency on rare-earth materials, and, thus, no significant heat susceptibility in the rotor part. However, the lack of permanent magnets (PMs) also leads to less torque density, and overall efficiency suffers from the fact that electromagnetic torque requires currents in the rotor bars to be induced.

A SRM is also positioned in the low cost region, but requires a small air gap and, therefore, lower production tolerances. The SRM's torque density is comparatively lower as it harnesses reluctance torque only. The largest constructional difference to other motor types is located at the SRM's rotor, which features radial, salient poles rather than a constant radial shape along the circumference. Moreover, the rotor consists of magnetic steel sheets exclusively. Hence, the rotor is also of robust nature, since no windings or magnets are involved, although rotor iron losses may be relatively high. The operation is based on the sequential excitation of diametrically opposite stator coils. This operating scheme requires more sophisticated rotor position and phase current readings and specialized power electronic units, especially for smooth low-speed operation. In addition, the operation's sequential character induces high torque ripple, vibration, and ultimately

acoustic noise. On the other hand, the SRM's low power factor is countered by its wide constant power range, high-speed capability, and its low susceptibility to temperature variations. Instances of this motor type for automotive applications can be found in [RS02; FES04; CK15].

The PMSM is a popular traction motor as it can satisfy most of the requirements stated in Sec. 2.3. In its prevalent form, PMs are embedded into the rotor, whose magnetic field interacts with the rotating magnetic field shaped by the stator windings. The PMSM exhibits excellent efficiency due to the inherent magnetic field of the PMs, which does not need to be generated beforehand as in IMs. In contrast to a BLDC type motor, the PMSM is easier to control with a relatively high maximum torque per current ratio even during the extended speed operation. The extended speed operation is enabled through flux-weakening of the PM's field, which is possible by high-frequency control with inverters [Jah87]. A downside of PMSMs is the dependency on rare-earth materials, which increases production costs. Also, a relatively high back electromotive force (EMF) in the windings especially at high speeds can cause issues when not mitigated by distributed windings. What is more, rare-earth materials are classified as depletable resource, similar to oil. Therefore, the cost is expected to not decrease in the long run, and alternative materials need to be searched for. The PMs' susceptibility to overheating is another downside, which will be treated more thoroughly later.

A SynRM utilizes reluctance torque only, similar to a SRM, by exhibiting magnetic steel sheets in the rotor exclusively. This eliminates the downsides of a PMSM that pertain to the rotor's PMs, but also the availability of electromagnetic torque. Similar to a PMSM and in contrast to a SRM, the SynRM utilizes three-phase currents fed through the stator windings, such that acoustic noise is less of an issue here. The constructional difference of the SynRM's rotor with respect to the SRM's is an axial lamination that tracks the stator magnetic field lines in a certain rotor position precisely. This maximizes the saliency ratio at the cost of higher manufacturing effort and counters the torque density decrease due to the lack of electromagnetic torque.

For the sake of completeness, it is to be mentioned that there are also unconventional motor types like the flux switching motor [Rib+16] and the axial flux motor [Lee+14; Tak+17]. The former has a lucrative high torque density at the cost of high cogging torque and more odd harmonics in the back EMF, while the latter comes with a higher power density but also higher manufacturing complexity.

Industry applications

Tesla's model S facilitates a three-phase IM with a copper squirrel-cage rotor structure [San+12]. A copper squirrel-cage is one of many constructional measures to minimize the leakage reactance, which leads to a wider speed range [ZH07]. Other electric vehicles that feature an IM are the 2018 and 2020 Audi e-tron (S) [Pin+18; Doe+18; Doe+20], and the 2021 VW ID CROZZ [Hel+20]. The Toyota Prius second and third generation (2003 and 2009, respectively) as well as the 2005 SUV model incorporate an interior-mounted permanent magnet synchronous motor (IPMSM) [CK15; Kam06]. The Porsche Taycan (2019) facilitates two PMSMs, one per drive axle for an improved dynamic control [Eng+19; Wie21]. A variety of motor types can be found in Tesla's model 3 and model Y: the rear-wheel axle, in particular, incorporates a mix of a PMSM and a SynRM. The composite motor is termed an interior permanent magnet synchronous reluctance machine, and it complements the efficient-at-high-speeds IM on the front-wheel axle [Ruo18; Lam18; Mar21].

An overview of other motor type instances among series vehicles can be found in [RL19] and together with their corresponding cooling methods in [GK21]. It is noteworthy though that no SRM has serious applications in the industry yet.

2.5 Power electronics

The drive train's power electronics encompass the battery as power source and the inverter (see Fig. 2.1). In the last decade, substantial improvements in the specific energy of batteries were accomplished [Van+21], coming from 110 W h/kg and achieving 250 W h/kg today, with an expected doubling in the following decade. At the same time, battery costs were cut by a factor of 10. Similarly, traction inverter power density was increased from 10 kW/L to 30 kW/L and is expected to also double in the next decade, depending on the DC-link voltage and the advent of new materials.

The dominant battery technology is the lithium-ion cell type due to its relatively high cell-level energy density. In order to reduce cable weight and copper losses, the battery's voltage supply level is set high – usually in the range of 200 V to 450 V, and recently also up to 800 V [Reb16; Agh+21]. The remarkable increase in specific energy is mainly due to successful research around suitable electrolytes for high-voltage cathodes, which could be further elevated in the future through the introduction of solid-state electrolytes [Van+21].

The battery's DC voltage supply has to be converted efficiently into three-phase sinusoidal waveforms before they are applied onto the motor's terminal for optimal energy conversion. For this task, three-phase inverters are employed, which convert the DC-link voltage in a frequency- and amplitude-variant manner. They are either directly connected to the battery or over another DC/DC boost converter for a stepped-up battery voltage. The prevalent topology in automotive applications is the two-level B6 inverter, which hosts six insulated-gate bipolar transistors (IGBTs) together with a smoothing DC-link capacitor input-side (see Fig. 2.3) [Rei+19b]: Being a simple single-stage-converting, well-studied topology, the B6 inverter is capable of torque-controlling the motor in both positive and negative speed direction such that all four operating quadrants are served reliably at low manufacturing costs. This capability is required for the energy-efficient operation of the vehicle in all operating conditions it can face including recuperation. Motor control is conducted through switching commands sent from a microcontroller or digital signal processing (DSP) to the inverter, potentially through a pulse width modulation (PWM), which is further discussed in Sec. 2.6.

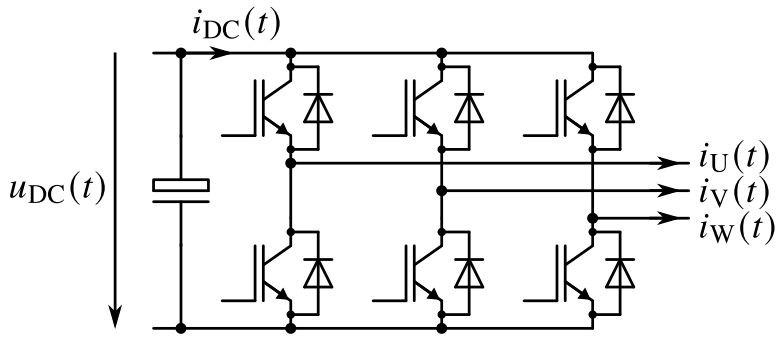


Figure 2.3: The schematic of an IGBT B6 two-level inverter

The IGBT two-level B6 inverter operates at up to 10 kHz with high efficiency, but above that, three-level topologies were shown to be superior with lower total harmonic distortions in the output voltage [SFK13; Rei+19b]. Although this increases the motor's efficiency, multi-level inverters are still not expected to enter series production in the long run because of higher capacitor bank volume, additional control requirements due to a higher amount of parts, and overall increased cost.

The choice of semiconductor materials for switching transistors in the automotive sector, on the other hand, trends from traditional silicon (Si) to wide band gap (WBG) materials such as silicon carbide (SiC) and gallium nitride (GaN). These new materials allow for higher switching frequencies, operation at higher temperatures, and higher breakdown voltages at lower losses and module sizes [Cha+19; Rei+19b]. More and more OEMs launch car models that facilitate SiC inverters, e.g., Tesla's model 3 and Y [BL18; Mar21] or Lucid Motor's

Lucid Air [Hah22]. However, research around these new materials is far from completed, and the lack of rigorous long-term studies dealing with overload capabilities and control at high thermal stress as well as high costs hinder sector-wide application yet [Van+21]. Future challenges around inverter design encompass mainly the application of WBG devices, advanced component packaging with more sophisticated thermal management, its integration into the system, and application of advanced manufacturing techniques.

2.6 Drive control

State-of-the-art drive control passed several milestones in the last 40 years and still experiences new paradigms thanks to the introduction of new power electronic devices, materials, high-speed digital devices such as continuously refined sensors and processors, and – equally important – more and more sophisticated control algorithms [DPV20]. Each technological leap in modern drive control enabled higher scalability to smaller and larger power devices, more robustness and reliability, shorter development cycles, and even the need for less sensors. The emergence of high-clocking microcontrollers and DSPs with fast floating-point arithmetic additionally enabled more advanced control algorithms that were not feasible before. Also, development tools of higher flexibility serving high-level programming languages such as C++ or model-based frameworks like MATLAB/SIMULINK for the design of ultimately embedded software are leveraged for the fast-paced progress in this field.

The prevalent control algorithm for three-phase AC drives is the field oriented control (FOC). Introduced 50 years ago by [Has69; Bla73], it is the standard among industry applications and academic lectures alike [Leo01; SB20]. It decouples control of flux linkages and torque by contemplating currents, voltages, and fluxes in a rotor-flux-fixed coordinate system [Par29]. In such a coordinate system, which requires knowledge of the rotor position, all alternating signals become of constant magnitude over time per operation point. This gives way for a multitude of control algorithms that stem from DC drive control. A central part of the FOC is a subsequent modulator which converts scalar duty cycles to applicable switching commands for the inverter. The duty cycles are provided by several proportional integral (PI) controllers acting on the control error (and its integral) with certain gain factors. Next to the classic PWM, the so-called space vector modulation is a recommended enhancement for digital implementations, where durations of active vectors are directly calculated given the sample interval and the specified average voltage reference vector [DPV20].

While the FOC is a linear control approach, the so-called direct torque control (DTC) is of nonlinear nature, enjoys almost equally high popularity, and foregoes the need for a modulator. Introduced by [Dep87; TN86], it avoids a cascaded control scheme and directly controls torque and flux, often incorporating a hysteresis control through on-off controllers and switching command look-up tables (LUTs). A prominent advantage of the DTC over FOC is it achieving higher torque dynamics by working on smaller time intervals due to a lower computational demand on DSPs [Mey10]. However, since the switching frequency is adaptive and depending on the actual torque and flux trajectory around the set reference trajectory bands, the electromagnetic compatibility becomes difficult to assess especially in automotive applications. Moreover, higher current- and torque ripple are common, which is generally undesired.

A more recently proposed control strategy is the model predictive control (MPC) of three-phase motors. First proposed in [Pro63], it gradually slips into industrial traction drive applications thanks to the rising performance of DSPs nowadays. A vast body of literature exists for this matter [Bol+08; Cor+08; Vaz+14; Kou+15; WB17; Han+19; Bro+20a]. The MPC is hallmarked by a relatively high computational demand due to iteratively solving a quadratic problem at run time. This optimal control problem is posed by a custom cost function containing arbitrary control goals over a fixed-size horizon and a mathematical model of the controlled system. In particular, the cost function is usually a variation of a distance metric between set reference trajectory and possible actual trajectories forecasted over the sliding time horizon with the aid of the model. The actuating variables are then set according to the cost's minimizer at the next sampling point where the full computation

would then be repeated. Due to the high computational complexity with an increasing horizon, only very few steps into the future are usually predicted, which simplifies the general optimal infinite-horizon control problem into a limited-horizon control problem. What is more, imprecise models tend to deteriorate the control performance significantly, and acquiring a model with sufficent accuracy is not trivial. On the other hand, the MPC denotes a control strategy with higher flexibility than the FOC or DTC and with an elegant embedding of system constraints because of an optimization according to an arbitrary cost function.

A relatively new control approach is presented by reinforcement learning (RL), which transfers the high computational burden into an offline training phase in exchange for a lean inference. In contrast to MPC, a RL controller can relinquish the incorporation of a system/plant model, and its cost function has no limited time horizon [Gör17; SB18]. The controller's parameters would be found during repeated exposure to measurement data and the usual gradient descent optimization onto a likewise flexible cost function (although, in the context of RL, it is usually the gradient ascent maximization of a reward function instead). Albeit less precise, (pre-)training can happen in a simulative environment [Tra+20b; Bal+21], and can aid in initial investigations before deploying this approach on a test bench or even series vehicles. Moreover, the training phase can even be quasi-online by decoupling parameter update calculations to edge computing while a mix of the so far identified controller and safety mechanisms operate on a test bench [Boo+21]. Although current research is in its infancy, first proof-of-concepts were highlighted [DR09; SKW20; SW21]. With the ceaseless advance of control hardware in form of microcontrollers, DSPs, field programmable gate arrays (FPGAs) [SD20; Lie+21], or even tensor processing units (TPUs) [Jou+18], it seems likely that intelligent control algorithms tuned in a data-driven fashion are increasingly adopted in the future.

2.7 Relevance of thermal stress and its monitoring

A precise, real-time-capable temperature estimating model would serve a multitude of purposes in an electric drive application: Foremost, the motor can be protected from overheating harm by applying a power derating before temperatures reach critical values; the motor's overload performance can be sustained for longer periods as no substantial safety margins have to be complied with; less costly and heavy material could be incorporated into the motor design, that would serve as additional heat sinks otherwise; and, eventually, the knowledge about the thermal conditions can aid in secondary applications, like state-of-health assessment, more efficient drive control with well-informed flux observers, or conforming to functional safety regulations.

Most OEMs have high ambitions to meet the market's growing demand for ever-increasing power densities and peak-power capabilities. This lets increasingly sophisticated cooling mechanisms sprout beyond mere cooling jackets, e.g., to spray cooling and high-speed rotor shaft cooling. A comprehensive review for state-of-the-art cooling systems in highly utilized traction motors can be found in [GK21]. However, as long as no thermal monitoring is set into place the root problem of avoiding overheating through safety margins and additional materials will not be resolved but rather pushed to higher power ratings.

The components most susceptible to damage from overheating in a PMSM are the stator winding insulations and the PMs in the rotor [Jun+11]. The winding heads experience high ohmic power losses, especially in the prevalent distributed winding scheme. Heat transfer from there into the stator core and an adjacent cooling jacket is often denoted by a short conductive heat path, but direct liquid-cooling is also common. In contrast, the PMs' internal location makes them more challenging to cool down, which is usually conducted through heat convection by the moving air fluid in the air gap between stator and rotor nowadays.

Overheating in the stator windings may lead to premature aging or even damaging of the insulation. Such conditions worsen electrical stress through voltage surges, which is a crucial aging aspect in EVs next to mechanical fatigue of the bearings [RH11; HG14; Mad+19]. A possible fault condition due to insulation

melting and the subsequent short-circuited copper litz wire is a degraded motor performance from shortened effective wire lengths and asymmetrical load, or even an erupting fire hazard and an abrupt end of lifetime.

On the other hand, PMs tend to slowly demagnetize over their lifetime [AH19], which directly deteriorates the flux and, thus, torque output. The demagnetization is mainly driven by high loads in the flux-weakening region (that is, at high speeds) and by thermal stress. Overheating can even lead to irreversible demagnetization up to the extent of total motor failure. The threshold of overheating depends on the materials used in the PMs [Mal+14; Wal17]. Among possible materials, neodymium iron boron (NdFeB) is most often used due to its highest energy density and coercivity as well as remanence [RL19], but at the same time it exhibits the lowest maximum allowed operating temperatures at as low as 150 °C [Spe14; Wal17]. In order to alleviate this constraint, rare-earth materials like dysprosium and terbium are often added to the PM composite (see [RL19] for a comprehensive review on employed materials in electric traction motors). Yet this material in particular is of relatively high cost and worsens the price pressure for PMSMs drastically although it makes less than 10 % of the final PM's composition [Mal+14]. This circumstance of high-price materials is tolerated by manufacturers due to the safety aspect against overheating and looming maintenance and repair costs otherwise. All the more, a thermal modeling-based monitoring could unsnarl this entanglement by allowing for smaller safety margins and for utilization of overload capacities in materials. This, in turn, allows for reduced incorporated mass at even safer operation.

What is putting more weight into the importance of model-based temperature estimation is the fact that real-time PM temperature monitoring through sensors is not technically and economically feasible yet. Prevailing arrangements encompass thermography cameras [GKK10] and slip-ring- [Mej+08] or telemetry-based signal transmission [Sti+12]. The former can capture heat development on the surface of rotor-surface-mounted magnets in the air gap, but not on embedded magnets within the rotor, which renders the method of thermography camera recording not only costly but also infeasible for many PMSM variants. The alternative of telemetry-based sensors employ classic thermocouples, but these have to be constructively incorporated into the rotor build, which is involved, error-prone, and of high cost as well. The high vibration strain and centrifugal forces on the thermocouples and their adhesive in the rotor makes for rather unreliable signal sources, which is merely feasible for laboratory equipment installations [Wal+16].

Another topic that increases the relevance of thermal monitoring is the tightening strict regulation for functional safety in road vehicles as stated in ISO 26262 [ISO18]. This norm was launched due to the increasingly complex electronic ecosystem in automotive vehicles, which makes for more hidden hazards for human life in case of hardware faults or misimplementation of software logic. The norm separates hardware and software (modules) into different levels of criticality depending on their potential impact on safety with higher levels being classified into automotive safety integrity levels (ASILs) A to D (highest). While features like vehicle climate control do not qualify for any particular redundancy measures, the – for instance – parking brake display icon would require more thorough monitoring (classified as quality managed, or QM) to not mislead the driver when leaving the vehicle. The emergency brake at high velocities would require even higher risk assessment and redundancy paths from sensor to actuator across hardware and software abstraction layers being located in ASIL. With regard to temperature sensor monitoring, there is usually one single temperature sensor embedded in the stator denoting a negative temperature coefficient (NTC) thermistor, which might degrade over the motor's lifetime with no plausibility or redundancy checks in place yet. It is to be expected that this sensor will also be subject to higher ASILs in the future due to the overheating hazard and its importance for economic operation of the motor, which might only be met with model-based monitoring. Especially for machine learning applications, though, the norm is not detailed enough [SQC17], and it will require more attention and work to make way for this emerging category of algorithms, despite its importance in advanced driver assistance systems and in autonomous driving even today [Spa+12].

Precise thermal information can be leveraged for more efficient motor control. Since most motors are torque-controlled, and the PM magnetic flux, as direct contributor to the torque output, is (approximately) linearly temperature-dependent, the adaptive response to temperature feedback can rigorously mitigate torque

deviations (also called torque ripple) [WB17]. Failing to accommodate measures against this source of torque ripple would otherwise lead to inferior performance also for higher cascaded control loops that aim for, e.g., efficient interplay of an ICE and electric motor(s) in a HEV [Tra+20a], or vibration suppression in the PMSM [DZ19]. The ohmic resistance of the stator windings is also susceptible to temperature variations, which has a large impact on current control loop performance [Pet14]. What is more, as will be further discussed in Sec. 3.1.4, efficient operation points are dependent on the actual temperature distribution in the motor because power losses develop differently according to actual material temperatures [WB17].

For the sake of completeness, it should be noted that next to motor component temperatures, the heat development in power electronic modules and the traction battery are also of concern in the traction drive power train, albeit these come with substantially smaller time constants. Examples of high interest are, for instance, the junction temperature in semiconductors [ADK12], or the operation temperature of the traction drive battery [Lin+12; KOL19]. Contemplating thermal monitoring in the full drive train enables optimal energy management in HEVs [Tra+20a] and derating control mechanisms involving regulation of switching frequency and current control limits that opt for safe maximum attainable drive train performance [LVD14; WB17]. Moreover, intelligent thermal management might reduce the load on certain heated power train components with almost no reduction in power output by variation of operation parameters, aiding in a more homogeneous wear and tear and reducing the time in overload conditions.

Although this work restricts its investigations to the thermal monitoring of a PMSM, the proposed methods and algorithms are developed with a focus on general applicability with minimal domain knowledge involved, often to the extent that they are also readily applicable to the domain of power electronics (power modules, inverters, batteries, etc.).

3 Dynamic modeling of a permanent magnet synchronous motor

The system identification or modeling procedure, as formally defined in [IM11], encompasses the derivation of a mathematical description for the temporal behavior of a process or technical system. During this procedure several simplifications are commonly applied to the resulting model to accommodate limited computational capacity present for both hardware platforms, workstations that identify the model and (potentially embedded) systems that are supposed to work with the identified model in the context of, e.g., a control scheme. These simplifications comprise the conversion of a distributed system described by partial differential equations (PDEs) to a lumped-parameter system represented by ordinary differential equations (ODEs); model order reductions that shrink the state space dimensionality; and decreasing the amount of model parameters, which might necessitate the transition from nonparametric to parametric models. Furthermore, on an orthogonal axis, the degree between theoretical (white-box) modeling and experimental (black-box) modeling, which relies completely on measurements and the observed input-output behavior of the system thereof, is often steered by attainable accuracy from computational constraints (see Fig. 3.1). The farthest end of the spectrum into the black-box direction is characterized by statistical models that are commonly associated with machine learning (ML). In order to avoid the identification of two models, a mixture of both, white- and black-box approaches, governs the state of the art of system identification (gray-box modeling). In such a hybrid scheme, a differential equation from literature is usually drawn on and furnished with trainable parameters.

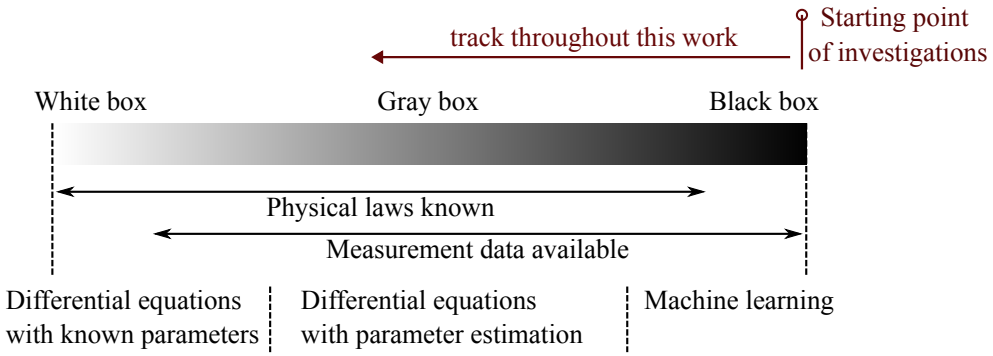


Figure 3.1: Different modeling categories that gradually transition to each other (derived from: [IM11])

This work distinguishes itself from previous work by approaching thermal modeling from the far black-box side of the spectrum, and then gradually working through to grey-box methods. More specifically, this thesis illustrates not only the chances of black-box thermal modeling for a PMSM, but also the benefits of employing an innovative mixture of classicly pure-black-box considered neural networks (NNs) with common lumped-parameter thermal networks (LPTNs) stemming from the ODE representation of a permanent magnet synchronous motor (PMSM). The mathematical representation of the electromagnetic and thermal behavior are, hence, fundamental to the later presented modeling techniques. Thus, this chapter deals with the state of the art of PMSM electromagnetic modeling, its fundamental mathematic derivation, the cause of power losses, and how sophisticated thermal modeling meshes with the increasing complexity of competitive electric vehicle design.

3.1 Electromagnetic modeling

A three-phase synchronous motor is characterized by the electromagnetic interaction between the three-phase current-induced coils in its fixed stator and its shaft-mounted rotor. In particular, the rotating magnetic field (RMF), as excited from the stator coils, rotates with a speed being an integer multiple of the rotor's mechanical speed, which coins the synchronous specification. A PMSM, then, is marked by incorporating permanent magnets into or onto the rotor, as the magnetic flux cause from the rotor side, in contrast to other synchronous machine designs that would rely on external excitation of the rotor flux. Moreover, it is to be distinguished whether the permanent magnets (PMs) are surface-mounted – resulting in a surface-mounted permanent magnet synchronous motor (SPMSM) – or embedded, which would be called an interior-mounted permanent magnet synchronous motor (IPMSM). See Fig. 3.2 for a sketched overview.

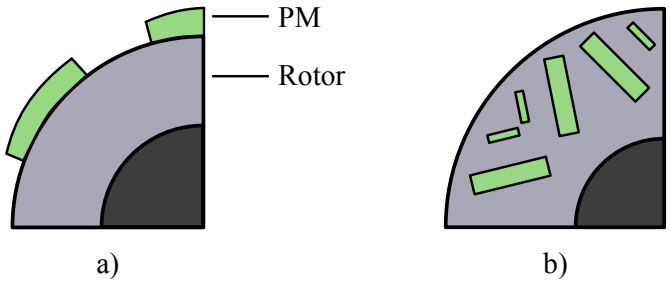


Figure 3.2: Quarter cross-section of two different PMSM rotor designs. a) surface-mounted PMs, b) embedded PMs in a stacked V-shape (derived from: [Eng+19])

The angle of the PMs relative to the circumferential surface and their thickness is not only a constructive property but also determines the torque yield significantly. Since the PM's permeability is approximately equal to the permeability of air ($\mu_r \approx 1$) and, hence, substantially lower than that of the iron sheets around it, the stator inductance along the IPMSM's rotor circumference is varying. This entails another class of torque – the reluctance torque – next to the electromagnetic torque, which is the only utilizable torque in a SPMSM. Consequently, IPMSMs are generally of higher energy and torque density, and are preferred in automotive applications [Mey10]. The double-layered and V-shaped PM embedding shown in Fig. 3.2 is a constructional optimization targeted on further increasing the stator inductance difference around the rotor's circumference [Hon+97; Kam06; Liu+16]. More embedding configurations can be found in [Gie08]. In this work, a PMSM with embedded PMs is investigated.

3.1.1 Coordinate systems

It is common practice to view all relevant electromagnetic quantities not in terms of the actual three symmetric phases abc, but rather in an equivalent, two-dimensional coordinate system, the so-called $\alpha\beta$ coordinate system. The original three signals are projected onto the 2D system taking advantage of Kirchhoff's current law [SB20]. Assuming the lack of a neutral conductor due to balanced phases, the transform (3.1a) and inverse transform (3.1b) described by Clarke [DSC51] can be simplified to

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \boldsymbol{T}_{23} \begin{bmatrix} i_\mathrm{a} \\ i_\mathrm{b} \\ i_\mathrm{c} \end{bmatrix} \qquad \text{with} \qquad \boldsymbol{T}_{23} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}, \tag{3.1a}$$

$$\begin{bmatrix} i_\mathrm{a} \\ i_\mathrm{b} \\ i_\mathrm{c} \end{bmatrix} = \boldsymbol{T}_{32} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \qquad \text{with} \qquad \boldsymbol{T}_{32} = \frac{3}{2} \begin{bmatrix} \frac{2}{3} & 0 \\ -\frac{1}{3} & \frac{\sqrt{3}}{3} \\ -\frac{1}{3} & -\frac{\sqrt{3}}{3} \end{bmatrix}. \tag{3.1b}$$

This is the amplitude-invariant version of the transform on the example of the stator currents i. Accordingly, power terms in the $\alpha\beta$ system have to be multiplied by $3/2$ to account for the third phase. Since the $\alpha\beta$ reference frame is stator-oriented, all electromagnetic quantities will be sinusoidal in this frame for a non-zero motor speed during steady-state operation. Having knowledge of the actual rotor position, one can further transform all signal values into a rotor-oriented reference frame where all alternating quantities become direct. This enables many DC machine control algorithms for application to three-phase machines as well [SB20]. This transform is known as dq transformation for the direct and quadrature axes that rotate with the stator RMF (see Fig. 3.3), and – in combination with the Clarke transform – was proposed by Park [Par29]. The dq transformation is described with the electrical rotor angle ε_el between d-axis and the stator by

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \boldsymbol{Q}(\varepsilon_\mathrm{el}) \begin{bmatrix} i_\mathrm{d} \\ i_\mathrm{q} \end{bmatrix} \qquad \text{with} \qquad \boldsymbol{Q}(\varepsilon_\mathrm{el}) = \begin{bmatrix} \cos(\varepsilon_\mathrm{el}) & -\sin(\varepsilon_\mathrm{el}) \\ \sin(\varepsilon_\mathrm{el}) & \cos(\varepsilon_\mathrm{el}) \end{bmatrix}, \tag{3.2a}$$

$$\begin{bmatrix} i_\mathrm{d} \\ i_\mathrm{q} \end{bmatrix} = \boldsymbol{Q}^{-1}(\varepsilon_\mathrm{el}) \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \qquad \text{with} \qquad \boldsymbol{Q}^{-1}(\varepsilon_\mathrm{el}) = \begin{bmatrix} \cos(\varepsilon_\mathrm{el}) & \sin(\varepsilon_\mathrm{el}) \\ -\sin(\varepsilon_\mathrm{el}) & \cos(\varepsilon_\mathrm{el}) \end{bmatrix}. \tag{3.2b}$$

For a PMSM it is conventional to align the d-axis with the PM flux, leading to a higher inductance in q direction (for embedded PMs). Conversely, for a SRM or SynRM the d-axis is aligned with a radial rotor direction where no cavities are pierced and, hence, the highest inductance exists [Jah87; IC15]. Note that the electrical angle ε_el scales with the mechanical angle ε_me through the pole pair number p:

$$\varepsilon = \varepsilon_\mathrm{el} = p\varepsilon_\mathrm{me}$$

In the following, ε and the angular velocity $\omega = \omega_\mathrm{el} = p\omega_\mathrm{me}$ denote the electrical quantities.

3.1.2 The fundamental wave model

The general voltage equation for three-phase stator windings in electric machines is given by

$$\boldsymbol{u}(t) = R_\mathrm{s}\boldsymbol{i}(t) + \frac{\mathrm{d}\boldsymbol{\psi}(t)}{\mathrm{d}t}, \tag{3.3}$$

where small bold letters $\boldsymbol{x} = \begin{bmatrix} x_\mathrm{a} & x_\mathrm{b} & x_\mathrm{c} \end{bmatrix}^\mathsf{T}$ denote column vectors, $\boldsymbol{u}$ the terminal voltage, R_s the stator resistance (assumed to be equal in all phases), $\boldsymbol{i}$ the phase currents, and $\boldsymbol{\psi}$ the magnetic flux linkage as seen by the stator windings. The time derivative of $\boldsymbol{\psi}$ represents the induced voltage. In the following, the time dependence is dropped in the notation in order to reduce clutter and for brevity. Applying (3.1a) and (3.2b) on

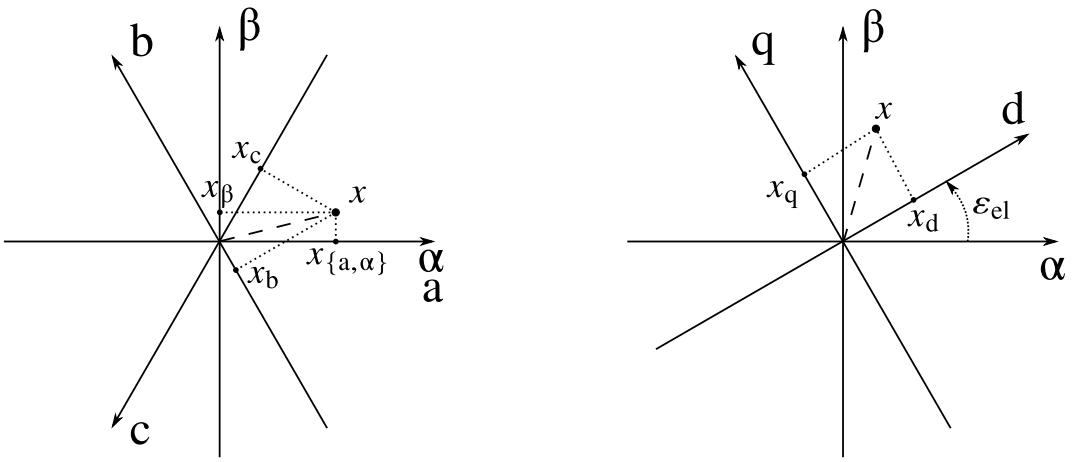


Figure 3.3: The stator-oriented αβ and the rotor-oriented dq reference frames (cf. [SB20])

(3.3) gives the PMSM voltage equations in dq coordinates:

$$\boldsymbol{v}_{dq} = R_s \boldsymbol{i}_{dq} + \frac{d}{dt}\boldsymbol{\psi}_{dq} + \omega \boldsymbol{D} \boldsymbol{\psi}_{dq} \qquad \text{with} \qquad \boldsymbol{D} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{3.4}$$

The last summand stems from the product rule and from $\boldsymbol{Q}(\varepsilon)$ being time-dependent.

Proceeding from here, the torque equation can be derived. Two causes of torque can be classified in a PMSM with embedded PMs: electromagnetic torque that originates from the Lorentz force resulting from the interaction between PM flux and stator currents, as well as reluctance torque, which is induced by varying circumferential reluctances and consequently also inductances [SB20]. Expanding (3.4) with $\frac{3}{2}\boldsymbol{i}_{dq}^{\mathsf{T}}$ by left-multiplication yields the power balance

$$\underbrace{\frac{3}{2}\boldsymbol{i}_{dq}^{\mathsf{T}}\boldsymbol{v}_{dq}}_{P_{el}} = \underbrace{\frac{3}{2}R_s\boldsymbol{i}_{dq}^{\mathsf{T}}\boldsymbol{i}_{dq}}_{P_{l,\,Cu}} + \underbrace{\frac{3}{2}\boldsymbol{i}_{dq}^{\mathsf{T}}\frac{d}{dt}\boldsymbol{\psi}_{dq}}_{\dot{E}_{mag}} + \underbrace{\frac{3}{2}\omega\boldsymbol{i}_{dq}^{\mathsf{T}}\boldsymbol{D}\boldsymbol{\psi}_{dq}}_{P_{me}}. \tag{3.5}$$

All terms can be interpreted as P_{el} for the input electric power applied on the motor terminal, $P_{l,Cu}$ for the ohmic power losses, $\dot{E}_{mag}$ for the time derivative of stored magnetic energy within the motor, and, consequently, P_{me} for the mechanical power applied on the shaft. The air gap torque T, then, can be obtained after rearranging

$$P_{me} = \frac{3}{2}\omega\boldsymbol{i}_{dq}^{\mathsf{T}}\boldsymbol{D}\boldsymbol{\psi}_{dq} \overset{!}{=} T\omega_{me} \tag{3.6}$$

$$\Leftrightarrow T = \frac{3}{2}p\boldsymbol{i}_{dq}^{\mathsf{T}}\boldsymbol{D}\boldsymbol{\psi}_{dq} = \frac{3}{2}p(\psi_d i_q - \psi_q i_d). \tag{3.7}$$

The air gap torque relation to the mechanical angular velocity ω_{me} can be given with

$$\omega_{me} J = T - T_L, \tag{3.8}$$

where J denotes the moment of inertia and T_L the load torque. The latter sums up all resistive forces that act against the air gap torque, such as bearing friction, gas friction, and, most notably, resistive torque at the wheel-to-road contact point. The latter summarizes resistive forces such as rolling friction as well as grade and wind resistive forces.

In order to end up with discriminable terms for both torque classes, the terms that compose the magnetic flux have to be further elaborated. As of here, the equations become motor-type-specific for a PMSM. Several levels of simplification exist for assessing the flux components among which the simplest is the fundamental wave model, in which no magnetic saturation exists, i.e., flux scales linearly with current:

$$\boldsymbol{\psi}_{\mathrm{dq}} = \boldsymbol{\psi}_{\mathrm{PM}} + \begin{bmatrix} L_{\mathrm{d}} & 0 \\ 0 & L_{\mathrm{q}} \end{bmatrix} \boldsymbol{i}_{\mathrm{dq}} \quad \text{with} \quad \boldsymbol{\psi}_{\mathrm{PM}} = \begin{bmatrix} \psi_{\mathrm{PM}} \\ 0 \end{bmatrix}. \tag{3.9}$$

Here, the flux linkage consists only of the PM flux ψ_{PM} (by definition only in d direction) and the constant stator self-inductances L_{d} and L_{q}, with a saliency ratio $L_{\mathrm{q}}/L_{\mathrm{d}}$ larger than 1 for an IPMSM and equal to 1 for an SPMSM. Applying (3.9) to (3.4) gives

$$\boldsymbol{v}_{\mathrm{dq}} = R_{\mathrm{s}}\boldsymbol{i}_{\mathrm{dq}} + \begin{bmatrix} L_{\mathrm{d}} & 0 \\ 0 & L_{\mathrm{q}} \end{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{i}_{\mathrm{dq}} + \omega \boldsymbol{D}\left(\boldsymbol{\psi}_{\mathrm{PM}} + \begin{bmatrix} L_{\mathrm{d}} & 0 \\ 0 & L_{\mathrm{q}} \end{bmatrix} \boldsymbol{i}_{\mathrm{dq}}\right), \tag{3.10}$$

and the corresponding equivalent electric circuit is illustrated in Fig. 3.4.

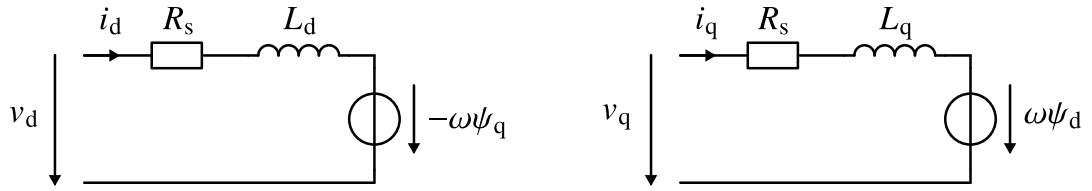


Figure 3.4: The equivalent electric circuit of the linear PMSM model in dq coordinates (cf. [SB20])

It follows for the air gap torque (3.7) that

$$T = \underbrace{\frac{3}{2}p\psi_{\mathrm{PM}}i_{\mathrm{q}}}_{\text{el.-magn. torque}} + \underbrace{\frac{3}{2}p(L_{\mathrm{d}} - L_{\mathrm{q}})i_{\mathrm{d}}i_{\mathrm{q}}}_{\text{reluctance torque}}. \tag{3.11}$$

Now the separation of torque classes is evident as well as the condition for reluctance torque to be $L_{\mathrm{d}} \neq L_{\mathrm{q}}$.

3.1.3 Nonlinear effects on the flux linkage

The flux linkage in real automotive applications is subject to several nonlinearities that are not covered by the linear fundamental wave model. Among others, (cross-)saturation effects as well as rotor angle and rotor speed dependencies are to be expected [Wal17]. The flux linkage is thus denoted by

$$\boldsymbol{\psi}_{\mathrm{dq}} = \boldsymbol{f}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) = \begin{bmatrix} f_{\mathrm{d}}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) \\ f_{\mathrm{q}}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) \end{bmatrix}. \tag{3.12}$$

Especially in highly utilized traction motors, nonlinearities due to the magnetic cross-saturation are induced by the inductive coupling between the stator phase windings, which will lead to a nonlinear (damped) scaling between flux linkage and currents. Since (3.12) is not separable into PM and stator components anymore without strong assumptions, inductances will be defined in terms of the Jacobian of the dq flux linkage with respect to the dq currents more appropriately (see Fig. 3.5 for the differential inductances in different operation

points of an exemplary motor [Bro+20a]):

$$\boldsymbol{L}_{\mathrm{dq}} = \frac{\partial \boldsymbol{\psi}_{\mathrm{dq}}}{\partial \boldsymbol{i}_{\mathrm{dq}}} = \begin{bmatrix} L_{\mathrm{dd}}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) & L_{\mathrm{dq}}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) \\ L_{\mathrm{qd}}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) & L_{\mathrm{qq}}(\boldsymbol{i}_{\mathrm{dq}}, \varepsilon, \omega) \end{bmatrix}. \tag{3.13}$$

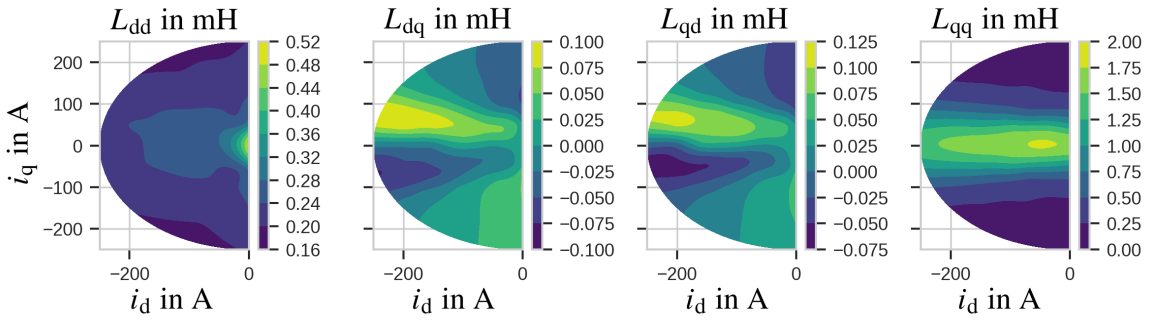


Figure 3.5: Differential inductances of an exemplary PMSM for traction applications (derived from: [Bro+20a])

The flux linkage's dependence on the rotor position is explicable by means of the motor geometry: the discrete distribution of PMs in the rotor and the winding scheme result in nonideal sinusoidal flux linkage distributions along the rotor's circumference. Moreover, the stator teeth and slots lead to a varying air gap width that has implications for the reluctance. Consequently, these irregularities will lead to high-order harmonic distortions in the torque yield even during stationary operation ($\mathrm{d}\boldsymbol{i}_{\mathrm{dq}}/\mathrm{d}t = 0$), which may result in additional strain on the mechanical components as well as noise pollution. However, these can be suppressed by a targeted current control [Lee+08].

Furthermore, the flux linkage's dependence on the motor speed is rooted in additional iron losses (especially eddy current losses) and frequency-dependent measuring errors in the sensors [Ram+13]. Similar to the magnetic saturation, the iron losses and sensor influence can be experimentally identified and being accounted for in the electromagnetic model, and will lead to torque deviations if neglected otherwise.

Table 3.1: Major power loss causes in inverter-fed PMSMs (cf. [Bin17; Wal17])

current-independent	• cyclic magnetization losses due to pulsing magnetic flux density variation • Joule heating due to pulsing current ripples • bearing friction losses, and windage friction losses in the air gap
current-dependent	• Joule heating in the stator winding • Joule heating in the supply cables • Additional ohmic losses due to the asymmetric distribution of the alternating current over the conductor's cross-section (skin-effect) • Iron losses due to the flux fundamental wave • Iron losses due to flux harmonic waves • Iron losses in the permanent magnets due to flux harmonic waves • Iron losses in core end plates and miscellaneous attachments due to stray fields

3.1.4 Power losses

Power losses impact heat generation in power systems fundamentally and, thus, often play an important role in several aspects. Knowledge of generated losses can be utilized for efficient motor control during operation [PWB12; WMB15], for motor control with derating measures in overload conditions [LVD14; Qi+15], or for optimal motor design [YI10; Dor+11]. Most importantly, power loss maps, either measured or calculated, often represent prerequisites for thermal models, e.g., as in [Bog+09; WB16a; Wöc+20]. Losses are usually assessed either analytically, experimentally on test benches with a power meter [WB16a], or through simulations, e.g., with a comprehensive, multi-physics, finite element analysis (FEA) [MSB03; YI10] or with computational fluid dynamics (CFD) [Bog+09; Nat+13]. An overview on contemporary loss modeling techniques and analyses is given in [Wro+16; Bil+19].

In this thesis, however, power losses are not derived a priori and, therefore, are not fed into the presented thermal models, but are rather modeled implicitly as unknown components of heat generation that are to be identified (especially for the gray-box approaches). Despite the fact that this work's thermal models make no strong assumptions over the physical reasons behind power losses – except for being assessable from patterns in the typical sensor readings – they are, nonetheless, rooted in thermoelectromagnetic and mechanic processes that will be elucidated in the following.

Losses can be categorized into copper, iron, and mechanical losses, which can further be subdivided into classes with varying dependencies. In Tab. 3.1, an overview of the major power loss causes in inverter-fed PMSMs is given [Bin17; Wal17]. It is to be noted, however, that losses do not only depend on current, but also many more factors like component temperatures, RMF frequency, inverter switching frequency, or control efficiency. A brief summary of the most important losses and their modeling follows.

Copper losses

Copper losses, or equivalently Joule heating, occur in the stator windings and represent a major loss source in electric machines. Their cause and, therefore, calculation differs according to whether AC or DC runs through the wire. While for DC a linear stator winding resistance $R_{\mathrm{S,DC}}$, which scales positively with the winding temperature ϑ_{S}, is a widely accepted first-order approximation, the frequency-dependent resistance computation for AC is a more complex matter [Wro+16]. The resistance varies according to the skin, proximity,

slot, and rotor reaction effect, whose efficiency-derogating contributions superimpose in the windings and varies the current density over the conductors' cross-sections. Additionally, the PWM switching frequency and circulating currents due to the PM flux complicate the assessment.

The skin effect describes the displacement of alternating current in a conductor to its surface boundary with higher frequencies, leading to a higher current density at the outer cross-section (shaping a "skin"). The current density decreases exponentially in radial direction toward the conductor core. In contrast to the other effects, the skin effect also occurs for a single levitating, straight conductor. The resulting skin depth leads to a smaller effective conducting cross-section and, hence, a higher effective ohmic resistance and higher ohmic losses eventually. The effect originates from the law of induction, or more specifically, the Maxwell-Faraday equation [Hei16]

$$\nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \tag{3.14}$$

which states that each time-varying magnetic field is accompanied by a spatially varying and possibly time-varying electric field – the eddy currents. These currents superimpose the magnetic-field-generating AC in the conductor, canceling it in the core region and intensifying it at the surface. The negative time derivative of the magnetic flux is also known as electromotive force (EMF), which is the work done on a unit of charge when having passed through the entire conductor loop due to the time-varying magnetic field.

When multiple conducting wires come together, e.g., to form the motor's winding, an additional effect appears – the proximity effect. It increases the ohmic resistance with higher frequencies through a smaller effective cross-section for the current flow similar to the skin effect. Unlike the skin effect, the proximity effect is caused by the conductor-permeating, time-varying magnetic field generated by the neighboring conducting wires. Due to the origin of the electromagnetic field wave being outside the considered conductor, the current density is not pushed evenly over the surface but rather to the side that is farthest away from the neighboring conductor carrying AC with the same phase. In electric machines specifically, the net result of the proximity effect leads to the current density tending to concentrate near the slot opening because of the uneven arrangement of copper wires in radial direction within slots, which is termed the slot effect [Bil+19]. In addition, the excitation field from the rotor PMs also contribute to the total eddy currents in the stator winding, denoting the rotor reaction effect [Wro+16].

Measures against these effects usually denote a trade-off with the slot fill factor [VPF14]: each conduction phase is split into several strands that make a bundle of thin wires to reduce the resistance increase. Such measures require twisting and displacing the strands to avoid too high ohmic losses from circulating currents and imbalanced per strand current share though, which reduces the slot fill factor, that is, the ratio of copper wire area per total slot area.

To summarize, all copper losses can be described as follows [Wro+16]:

$$P_{\mathrm{l,\,Cu}}(I_\mathrm{s}, \omega, \vartheta_\mathrm{S}) = P_{\mathrm{l,\,Cu,\,DC}}(I_\mathrm{s}, \vartheta_\mathrm{S}) + P_{\mathrm{l,\,Cu,\,AC}}(I_\mathrm{s}, \omega, \vartheta_\mathrm{S}), \tag{3.15}$$

where $P_{\mathrm{l,\,Cu,\,DC}}$ represents copper losses due to the ohmic DC resistance (as mentioned, usually increasing linearly with winding temperature), and $P_{\mathrm{l,\,Cu,\,AC}}$ denotes the frequency-dependent ohmic loss due to the AC in the stator winding. It should be noted that the net correlation of the resistance with the stator temperature is not necessarily positive, due to the power losses that originate from AC tending to decrease with higher temperatures. Starting from (3.15), more detailed models or further approximations can be elaborated, e.g., as in [Wro+13] with

$$P_{\mathrm{l,\,Cu,\,AC}}(\omega, \vartheta_\mathrm{S}) = P_{\mathrm{l,\,Cu,\,DC}}(\vartheta_{\mathrm{S},0}) \left(1 + w_1 (\vartheta_\mathrm{S} - \vartheta_{\mathrm{S},0}) + \frac{\frac{R_{\mathrm{S,AC}}}{R_{\mathrm{S,DC}}}(\vartheta_{\mathrm{S},0}, \omega) - 1}{(1 + w_1 (\vartheta_\mathrm{S} - \vartheta_{\mathrm{S},0}))^{w_2}} \right), \tag{3.16}$$

where $R_{\mathrm{S,AC}}/R_{\mathrm{S,DC}}$ is the AC to DC stator resistance ratio that is assumed to be inferrable from a LUT depending on ω and a reference stator temperature, whereas w_1 and w_2 denote free parameters for curve fitting on data from experiments or FEA.

While the dependence on stator temperature ϑ_{S}, stator current magnitude I_{s}, and RMF frequency ω in (3.15) is usually sufficient for an accurate modeling, there are many more factors coming into play in principle, like PM temperature, excitation current angle, or the PWM switching frequency, among others [Wro+16]. Especially the influence of the PWM switching scheme and the resulting current ripple on the proximity and skin effect is found to be not negligible [Iwa+08; VPF14; SK15]. However, with the emerging WBG materials in voltage-source inverters, current ripples, and, thus, copper losses due to the PWM control can be mitigated [Cha+20]. Like most of the other loss terms, the impact of the PWM switching scheme is commonly investigated with FEA, if at all, due to the intricate interaction between many components and effects.

Iron losses

The sum of both, eddy currents and hysteresis losses in the machine lamination, commonly denote iron losses. As the name says, they are generated in the iron assembly of the stator and rotor including magnetic steel sheets and in the PMs. They also include losses in the motor housing due to leakage flux. Iron losses are also commonly referred to as core losses or magnetic losses, and they have their cause in the alternating magnetic field that permeates the motor components [Kri14].

Hysteresis losses incur due to the additional energy needed to demagnetize ferromagnetic substances after they are fully magnetized by an external magnetic field, and even more so for reversing their magnetization with a subsequent reversed external field. The so-called Weiß domains in iron tend to align with an external magnetic field nonlinearly, and they retain their alignment partially even when no external field exists anymore. The remaining magnetization is called retentivity or remanence – it is desired to be high for PMs and low for electromagnets. The magnetization strength of a magnetic field in opposite direction that is required to fully cancel the material's magnetization is called coercive force. This nonlinear and biased alignment is obvervable for both polarizations in ferromagnetic materials, such that the magnetization flux density over field strength curve $B(H)$ represents a hysteresis loop. The area enclosed by this hysteresis loop denotes the total hysteresis specific energy in $\mathrm{J/m^3}$ needed for a ferromagnetic component to traverse a full cycle:

$$E_{\mathrm{hyst}} = \oint B(H)\mathrm{d}H. \tag{3.17}$$

If the hysteresis losses in the PMs are to be calculated, for instance, and the hysteresis curves are known, the equation

$$P_{\mathrm{hyst,\,PM}} = 2E_{\mathrm{hyst,\,PM}}V_{\mathrm{PM}}qpn \tag{3.18}$$

can be used [Pet+17], where V_{PM} describes the volume of one PM, q denotes the number of stator slots, p the pole pair number, and n the rotational motor speed. This equation neglects small-scale variations from the PWM switching scheme and from flux harmonics that produce sub-cycles of the hysteresis, though. A more thorough investigation is required to account for these as well.

Similar to eddy currents that contribute to copper losses, there are also eddy currents that develop in iron-like materials with a finite electric conductivity when permeated with an alternating magnetic field. These currents result in heat that impairs the system's efficiency. In order to assess them, first, the Maxwell-Faraday equation (3.14) is rearranged into integral form

$$\oint \boldsymbol{E}\mathrm{d}\boldsymbol{s} = -\int \frac{\partial \boldsymbol{B}}{\partial t}\mathrm{d}\boldsymbol{A} = -\frac{\mathrm{d}}{\mathrm{d}t}\int \boldsymbol{B}\mathrm{d}\boldsymbol{A}. \tag{3.19}$$

Now, assuming for the sake of simplicity a box-shaped steel sheet with thin thickness d and height l_h ($d << l_\mathrm{h}$) that is permeated by an alternating magnetic field, (3.19) gives

$$E(x)2l_\mathrm{h} = \frac{\mathrm{d}B}{\mathrm{d}t}x2l_\mathrm{h} \tag{3.20}$$

in which x denotes the spatial coordinate across the magnetic field. From here, the eddy current loss density follows by introducing complex notation

$$\underline{S}(x) = \kappa\underline{E}(x) = -j\omega\kappa x\underline{B} \quad \text{where} \quad \hat{S}(x) = \omega\kappa x\hat{B}. \tag{3.21}$$

Given the material density coefficient ρ and electric conductivity κ, the specific eddy current loss can be formulated as [Wal17]:

$$P_\mathrm{eddy} = \frac{1}{\rho\kappa d}\int_0^{d/2}\hat{S}^2(x)\mathrm{d}x = \frac{1}{24}\frac{\kappa}{\rho}d^2\omega^2\hat{B}^2. \tag{3.22}$$

It can be seen that the eddy current losses scale quadratically with the frequency and amplitude of the external magnetic field, but also with the sheet's thickness. That is why the iron assembly in electric machines is usually sheeted, and why PMs are outlined in segments for each pole: Eddy current propagation is effectively impeded. The analytical treatment of eddy currents in PMSMs is elaborate and can be found, e.g., in [Ata+99; Wan+10; PB15]. Nonetheless, it is noteworthy that iron losses are traditionally difficult to asses despite their long history of research: They cannot be measured directly, and efforts through, e.g., thermal or calorimetric measurements, hall sensors, or loss separation schemes, are only indirect measurement techniques [Kri14].

Moreover, it is to be noted that for both, iron and copper losses, rather significant thermal dependencies exist [SH13]. When the component temperature rises, copper windings experience an increase in their electrical resistance, which directly leads to higher copper losses, see (3.15). Iron losses, on the other hand, tend to decrease with higher temperatures: The Weiß domains require less polarization energy because of more atomic motion, and the hysteresis loop contracts with a reduced coercive force and remanence. Furthermore, eddy currents lessen in intensity due to a decreasing electric resistance of the ferromagnetic sheets. Eventually, the PM flux with its quadratic contribution to eddy currents (see (3.22)) is reduced significantly due to a decreased remanence of up to 10 %/100 K in ferromagnetic material. A rigorous loss model, irrespective of the approach, would have to consider these interactions, which necessitates a multi-physics investigation in which electromagnetic models influence thermal models and vice-versa.

Mechanical losses

Power losses due to mechanical friction represent the last and smallest class of losses in an electric motor. Although they are not of electromagnetic nature, they shall be outlined for the sake of completeness. Their relative contribution to the total losses is expected to increase based on the trend to smaller motors with higher rotational speed. Mechanical friction losses can be identified to come from bearing friction and air fluid friction in the air gap.

Several variants of bearings were developed in the last decades from contactless bearings such as oil, air, or magnetic bearings to classic lubricated – and potentially cooled – (ceramic) ball bearings [SQW18]. The latter is the most prevalent, and leads to friction losses based on rolling and sliding and pressure forces between the lubricated ball and its enclosing races [Ian+20]. Alternatively, a simple approximation is given in [ĆOP12] with

$$P_\mathrm{bearing} = k_\mathrm{bearing}\sqrt{\frac{n}{n_\mathrm{max}}}, \tag{3.23}$$

in which k_bearing is a tunable parameter and can be empirically identified with the retardation method (letting the motor coast to standstill from a pre-defined rotor speed).

Losses from fluid friction, also called windage loss, occurs through laminar and turbulent flow of the air fluid in the gap between rotor and stator, as well as on the front plates, which increases with the rotor speed, too. Since this class of friction losses has the smallest contribution to total losses among mechanical losses, which are themselves often neglected, windage losses are often ignored in loss analyses. However, comprehensive analytical methods with various heat transfer coefficients for thermal convection and conduction in the air gap depending on rotor surfaces and cooling systems exist for traction motors and are reviewed in [GK21]. Empirical approaches can be found, e.g., in [Chi+18; Par+21], where the sum of bearing and windage losses are determined according to the process of elimination:

$$P_{\text{mech}} = T_{\text{no-load}}\omega_{\text{mech}} - P_{\text{Fe, FEA}} \tag{3.24}$$

where P_{mech} is the mechanical loss, $T_{\text{no-load}}$ is the required torque for an external load machine to keep the specimen motor at angular velocity ω_{mech}, and $P_{\text{Fe, FEA}}$ are the FEA-calculated iron losses in the complete assembly including PMs and conductors. This investigation would then be repeated for different angular speeds. Moreover, in case of oil-cooled electric machines, the fluid distribution of the cooling oil over the rotor surface is important, and is often assessed with CFD [PPP12].

3.2 Thermal modeling

Electric machines for traction applications experience highly dynamic operation conditions with a nonlinear thermal response such that simple thermal maps as LUTs or polynomial modeling with no further processing of sensor readings are far from feasible. Consequently, the economic request for precise thermal models (see Sec. 2.7) has let several different elaborate approaches emerge over the last decades. All thermal models, independent of the paradigm and that are to be employed in electric vehicles, are subject to the same set of requirements [Wal17]:

- High estimation accuracy for temperatures in vital components according to common regression metrics (e.g., mean squared or absolute error, coefficient of determination, etc.),
- low performance variance with changing operation points (all-time high accuracy desired for all motor speeds, coolant and ambient temperatures, current excitations, etc.),
- no additional measurement equipment shall be required – estimation shall work upon signals that are readily available through ECUs,
- low computational demand since thermal models are usually implemented on best-cost hardware,
- independence of potentially existing temperature sensors for increased redundancy according to functional safety.

Spanning the black- to white-box scale introduced in Fig. 3.1 to another orthogonal dimension – the computational demand – one obtains Fig. 3.6, which gives a broad overview of modeling paradigms for the temperature monitoring task.

It becomes clear that comprehensive spatial analyses like CFD and FEA simulations are beyond the scope of lean thermal models that might be employed on embedded systems, such that they are restricted to the design phase of a motor's development, as in [MSB03; PPP12; Nat+13; IC15; PB15; SK15; Pet+17; Wu+19; Nat+19; Cha+20]. It is characteristic of white-box models that they accommodate the heat transfer PDE by exclusively harnessing geometry and material specifications, often as provided by the manufacturer – if at all. For precise thermal analyses with the highest requirements for spatial resolution, however, white-box models are invaluable, yet still not free of approximation errors.

A LPTN, on the other hand, represents an approximated ODE for the heat transfer in a given system and resembles electrical equivalent circuits. The electrical circuit parameters can be substituted with thermal properties while Kirchhoff's laws still apply [Ber+07], see Sec. 3.2.2. The LPTN approach can cover a wide

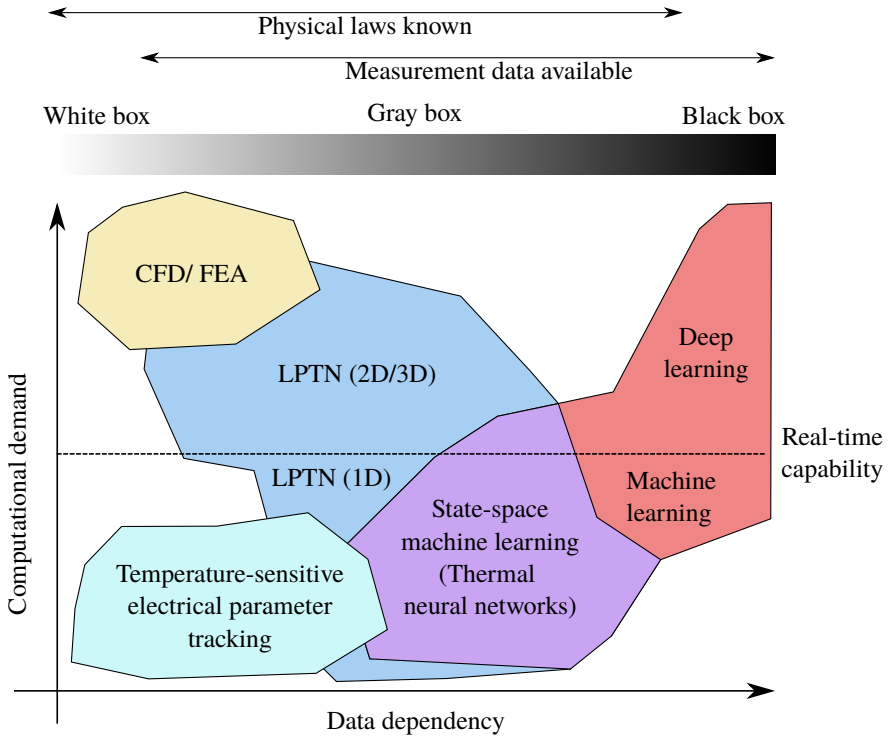


Figure 3.6: The position of different thermal modeling approaches in a landscape spanned by computational demand (on a runtime platform) and data dependency (during the model design phase) is shown (derived from: [Wal21]). The less empirical data is available the more an engineer has to rely on analytical coefficients derived from geometrical approximations or on simulations that might miss to capture parasitic effects.

range on the white- to black-box spectrum with many or few system states considered, and represents the prevalent approach to thermal modeling of electric machines [Bra+12; WB15; Bog+16; WB16a; Wal17; Chi+18; ZF18; Nat+19; GWB20; Wöc+20; Wal21]. Usually, the lower the order of the LPTN, i.e., the fewer states considered the more abstract the design and the more empirical data is necessary to achieve a desirable estimation accuracy. Moreover, with each state removed in an LPTN in order to achieve real-time capability, fewer actual physical locations in a system are representable. LPTNs with a very low order (under ten) denote heat transfer in a low spatial resolution most of the time, depending on the system complexity. With a similar computational burden as FEA/CFD, LPTNs with a high spatial resolution in 2D or 3D can aid during the design phase. Yet more and more efforts lead to computationally lighter forms due to the relevance of hot spots at runtime in contrast to the average component temperatures that a low-order LPTN would provide [QSD14; Bah+16; Lia+21].

Among actually employable methods, thermal models based on (potentially invasive) monitoring of temperature-sensitive electrical motor parameters are also to be mentioned. Noninvasive observers rely on a sufficient temperature-dependent variation in electrical parameters as, e.g., winding resistances, PM flux linkage, or stator inductance, during the nominal motor operation [SWB14; WSB17; Kat+20; Jun+21]. The invasive variant denotes different forms of fundamental- and high-frequency current or voltage injection [Rei+10; Gan+11; Rei+15; Fer+19; Rei+19a; Zap+21], and is often necessary if the electromagnetic fundamental wave model does not exhibit enough variation, for example due to insufficient system excitation at low motor speeds [Wal21].

The newest paradigm evolves from ML-based methods and mixed forms with LPTNs, which are this thesis'

core contribution. Black-box approaches with ML, where no domain knowledge from heat transfer theory is incorporated, are demonstrated in [WKB17; KWB19a; KWB19b; KWB20; LH20; KWB21a; CGK21], while gray-box developments where LPTNs are combined with neural networks can be found in [GWB21; KWB22; Kir+22; KWB23]. Most of the mentioned references are contributed by this thesis' author and denote the preliminary investigation to this work. A certain implementation of gray-box modeling is coined thermal neural networks (TNNs) [KWB23], and has been shown to facilitate most of the benefits from both, ML and LPTN thermal modeling, see Sec. 7.3.

Thermal modeling through observation of electrical parameters and LPTNs are briefly reviewed in this section, before machine learning as a system identification tool, and then specifically for thermal modeling, is treated in Sec. 4 and Sec. 6, respectively. For a more comprehensive overview of classic thermal modeling, the reader is referred to [Wal+16; Wal21; MZ22].

3.2.1 Real-time observers for temperature-sensitive electrical parameters

Instead of directly estimating component temperatures from typical ECU sensor readings, it is possible, under certain circumstances, to indirectly derive them from estimated electric parameters associated with the electromagnetic motor model [Wal21]. These electric parameters could be the winding resistance, or, specifically for PMSMs, the PM flux linkage or stator inductance. Commonly, the readily available FOC quantities currents $\boldsymbol{i}$, voltages $\boldsymbol{u}$, motor angular velocity ω, and rotor position ε are utilized for the estimation of a certain set of electrical parameters $\boldsymbol{\theta}_{\mathrm{el}}$ with a function $\boldsymbol{m}$ as in

$$\hat{\boldsymbol{\theta}}_{\mathrm{el}} = \boldsymbol{m}(\boldsymbol{i}, \boldsymbol{u}, \omega, \varepsilon) + \underbrace{\boldsymbol{K}(\boldsymbol{y} - \hat{\boldsymbol{y}})}_{\text{optional}} . \tag{3.25}$$

The optional second term indicates that besides an open-loop estimation also closed-loop observers with full motor models for the estimation of additional states, e.g., stator currents $\boldsymbol{y}$ with a linear feedback matrix $\boldsymbol{K}$ are feasible. Note the hat notation for estimates. Assuming a (often sufficient) linear relationship between electrical parameters and the corresponding temperature ϑ through a temperature coefficient α as in

$$\theta_{\mathrm{el}}(\vartheta) = \big(1 + \alpha(\vartheta - \vartheta_0)\big)\theta_{\mathrm{el}}(\vartheta_0), \tag{3.26}$$

in which ϑ_0 is a reference temperature, a rearrangement would give

$$\hat{\vartheta} = \vartheta_0 + \frac{1}{\alpha}\left(\frac{\hat{\theta}_{\mathrm{el}}}{\theta_{\mathrm{el}}(\vartheta_0)} - 1\right). \tag{3.27}$$

In view of (3.26), it can be noted that this equation assumes an effective average temperature over a component's spatial dimension, similar to low-order LPTNs. In a particular case, however, the spatial PM temperature distribution might be deduced from back EMF harmonics [Rei+16]. Notwithstanding the spatial resolution, there exists a substantial accuracy challenge with electrical parameter tracking, as can be anticipated from (3.27): Any electrical parameter estimation error will inflict the same error on the temperature estimation but scaled by $1/\alpha$. Typical values for α are in the range of $[-0.2 \ldots 0.4]\,\mathrm{K}^{-1}$ depending on the material and electrical parameter to estimate, which can result in an error propagation from ± 1 % in electrical parameters to ± 10 K in temperature estimates in the worst case [Wal21].

Further limitations of this method surface when considering the observability of the corresponding electrical parameters and the enforcement hereof through signal injection [Wal21]. First and foremost, signal injection usually causes additional losses, torque harmonics, and potentially acoustic noise. Furthermore, any invasive measure requires that there is a voltage and current reserve in the control loop, which contradicts the endeavor

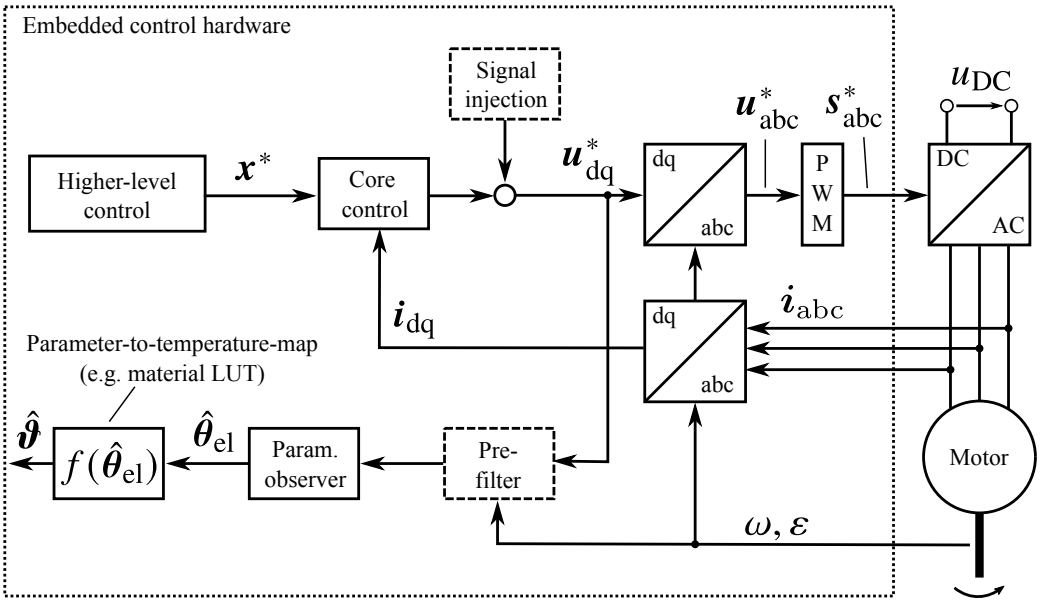


Figure 3.7: Sketched control block diagram that facilitates temperature estimation through electrical parameter tracking on an example that uses dq-transformed quantities (source: [Wal21]). Blocks with a dashed line are optional.

of temperature estimation for maximizing a drive's utilization. Applicability is, thus, restricted to small to medium motor speeds. This is the speed range where indirect estimation techniques without signal injection would lack voltage estimation accuracy due to a marginal back EMF and a low phase voltage. Therefore, signal injection at low speeds can become necessary to complement the indirect noninvasive estimation at higher speeds for a sufficient temperature estimation accuracy. What is more, if the natural harmonics caused by the inverter switching scheme are utilized, the control hardware would require additional computation through additional current samplings that were to be compared with the known inverter modulation sequence.

Moreover, it is characteristic of real-time observers for temperature-sensitive electric parameters to be tightly integrated into the control framework, as sketched in Fig. 3.7. This circumstance forces the parameter estimation routine to be run in the same task context as the motor control routine, i.e., typically in the micro- to millisecond range. In contrast, detached direct temperature estimation procedures could be implemented in a time scale more appropriate for thermal trajectories in the half second to a couple of seconds range.

Due to the mentioned drawbacks of this method, this thesis focuses entirely on direct temperature estimation models. Among these, thermal models based on heat transfer theory denote a potential base, and are outlined in the following.

3.2.2 Real-time estimation with lumped-parameter thermal models

Based on heat transfer theory, lumped-parameter thermal networks (LPTNs) are designed to estimate specific, spatially separated, system component temperatures by representing heat flows through an equivalent circuit diagram. Each parameter stands for a certain thermal characteristic. An appropriate LPTN structure is usually derived from the heat diffusion equation [Ber+07]:

$$\rho c_\mathrm{p} \frac{\partial \vartheta}{\partial t} = p + \nabla \cdot (\boldsymbol{\lambda} \nabla \vartheta), \tag{3.28}$$

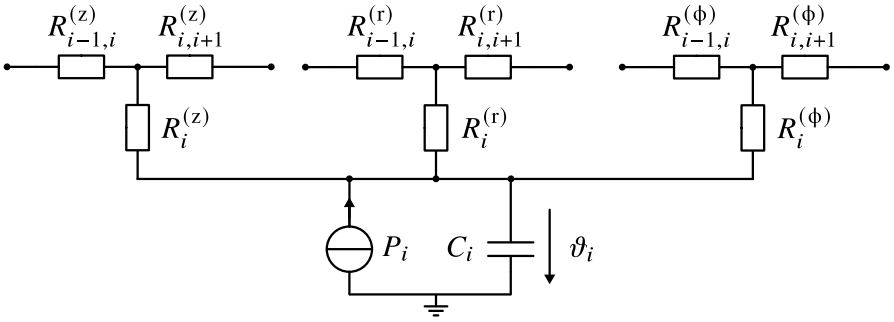


Figure 3.8: A LPTN T-type node in cylindrical coordinates (derived from: [Wal21])

where ρ denotes the mass density, c_p stands for the volume's specific heat capacity at constant pressure, ϑ is the scalar temperature field, p being the thermal energy generation from other energy sources (such as electric or chemical energy), ∇ being the del operator across spatial dimensions, and $\boldsymbol{\lambda}$ representing the potentially nonconstant, directional thermal conductivity matrix in $\mathrm{K} \cdot \mathrm{W/m}$ with

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_\mathrm{r} & 0 & 0 \\ 0 & \lambda_\phi & 0 \\ 0 & 0 & \lambda_\mathrm{z} \end{bmatrix}. \tag{3.29}$$

Due to the similarity to many motor components, the cylindrical coordinate system is used as can be inferred from (3.29). Rearranging (3.28) yields

$$\rho c_\mathrm{p} \frac{\partial \vartheta}{\partial t} = p + \lambda_\mathrm{r} \left(\frac{\partial^2 \vartheta}{\partial r^2} + \frac{1}{r} \frac{\partial \vartheta}{\partial r} \right) + \frac{\lambda_\phi}{r^2} \frac{\partial^2 \vartheta}{\partial \phi^2} + \lambda_\mathrm{z} \frac{\partial^2 \vartheta}{\partial z^2}. \tag{3.30}$$

For an approximated low-order LPTN, the PDE in (3.30) has to be mapped into an ODE. A popular form is the conjunction of T-type LPTN nodes, as depicted in Fig. 3.8. An alternative, widely used type is a simpler structure with only two resistors leading off the temperature node, which tends to overestimate the average component temperature but does not need to employ negative thermal resistances [QSD14]. The parameters of a T-type LPTN can be obtained by solving the Neumann and Dirichlet boundary conditions for each space coordinate [Ber+07; Wal21]. The Dirichlet condition is known as the first kind of boundary conditions where a solid body's surface is kept at a fixed temperature. The Neumann condition, on the other hand, is known as a boundary condition of the second kind, in which a fixed or constant heat flux p is maintained at the surface, e.g., at the axial end plates with

$$p = -\lambda_\mathrm{z} \left. \frac{\partial \vartheta}{\partial z} \right|_{z=0}. \tag{3.31}$$

Having identified the approximate thermal parameters for an m-th order LPTN according to the already approximated geometry and – if available – material properties, under given boundary conditions, further network reductions through simplifications of parallel and series resistances will result in a system of ODEs

$$C_i\big(\boldsymbol{\zeta}(t)\big) \frac{\mathrm{d}\vartheta_i}{\mathrm{d}t} = P_i\big(\boldsymbol{\zeta}(t)\big) + \sum_{j \in \mathcal{M} \setminus i} \frac{\vartheta_j - \vartheta_i}{R_{i,j}\big(\boldsymbol{\zeta}(t)\big)} + \sum_{j=1}^{n} \frac{\tilde{\vartheta}_j - \vartheta_i}{R_{i,j}\big(\boldsymbol{\zeta}(t)\big)}, \tag{3.32}$$

where C denotes the thermal capacitance, P the power loss, and $R_{i,j}$ the thermal resistance between the temperatures at node i and j, with $\mathcal{M} = \{1, \ldots, m\}$. Moreover, temperatures that can be measured during operation (typically ambient and coolant temperatures) are incorporated as further n ancillary nodes merely consisting of an equivalent thermal source $\tilde{\vartheta}$. Note that due to the simplification from a PDE into a system of

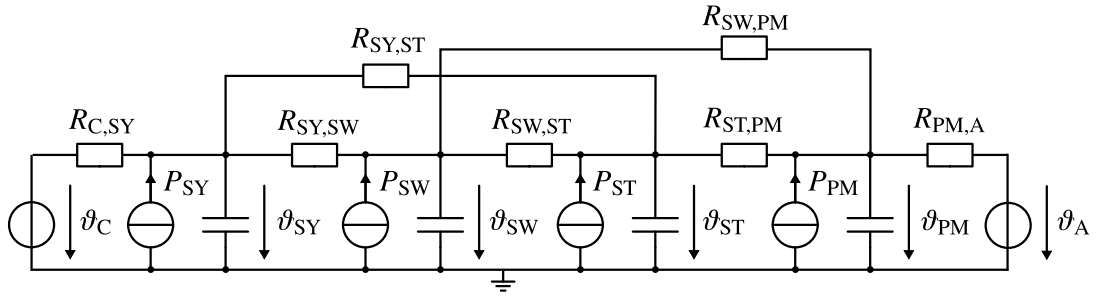


Figure 3.9: A low-order LPTN as synthesized for a traction PMSM in [WB16a] for four targeted temperature nodes representing the stator yoke (SY), stator teeth (ST), stator winding (SW), and permanent magnets (PM), as well as two ancillary temperatures with the ambient (A) and coolant (C) sensor readings (derived from: [WB16a])

ODEs, the thermal parameters in (3.32) are commonly operation-point-dependent rather than being constant for most applications. Moreover, especially convective and radiative heat transfer processes inherit highly nonlinear thermal characteristics making the transient definition of thermal parameters a challenging endeavor [Bog+06; HCH12]. Thus, thermal parameters are usually denoted in dependence on a scheduling vector $\boldsymbol{\zeta}(t)$. This vector is determined not only by sensor readings that are available during operation but also by all thermal states themselves. An example of a low-order LPTN is shown in Fig. 3.9, where a PMSM with a certain thermal node array is thermally modeled that is also central to this thesis.

One can bring (3.32) also into a state-space representation that unifies the system of ODEs into a linear parameter-varying (LPV) system with

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} &= \boldsymbol{A}\big(\boldsymbol{\zeta}(t)\big)\boldsymbol{x} + \boldsymbol{B}\big(\boldsymbol{\zeta}(t)\big)\boldsymbol{u}\big(\boldsymbol{\zeta}(t)\big), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t), \end{aligned} \tag{3.33}$$

in which

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{\vartheta} = [\vartheta_1 \dots \vartheta_m]^\mathsf{T}, \\ \boldsymbol{u} &= [P_1\big(\boldsymbol{\zeta}(t)\big) \dots P_m\big(\boldsymbol{\zeta}(t)\big) \quad \tilde{\vartheta}_1 \dots \tilde{\vartheta}_n]^\mathsf{T}, \\ \boldsymbol{A} &\in \mathbb{R}^{m\times m}, \quad \boldsymbol{B} \in \mathbb{R}^{m\times(m+n)}, \quad \boldsymbol{C} \in \mathbb{R}^{q\times m}, \end{aligned}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are the system, input, and output matrix, respectively. The output matrix $\boldsymbol{C}$ is typically a masking operation allowing the system output $\boldsymbol{y}$ to encompass fewer temperatures than are being modeled with $q \leq m$.

When implemented in software, the LPTN state-space formulation (3.33) is to be time-discretized with

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{\Phi}\big(\boldsymbol{\zeta}[k]\big)\boldsymbol{x}[k] + \boldsymbol{H}\big(\boldsymbol{\zeta}[k]\big)\boldsymbol{u}\big(\boldsymbol{\zeta}[k]\big), \\ \boldsymbol{y}[k] &= \boldsymbol{C}\boldsymbol{x}[k], \end{aligned} \tag{3.34}$$

in which it is assumed that the input and scheduling variables are constant during one sampling interval $t_k = k \cdot T_\mathrm{s} \quad \forall \quad k \in \{1, \dots, K\}$ (zero-order hold) [WB16a]. Here, the discrete-time transition and input

matrices

$$\begin{aligned}\boldsymbol{\Phi}\big(\boldsymbol{\zeta}[k]\big) &= e^{\boldsymbol{A}(\boldsymbol{\zeta}[k])T_\mathrm{s}},\\ \boldsymbol{H}\big(\boldsymbol{\zeta}[k]\big) &= \int_{t_k}^{t_{k+1}} e^{\boldsymbol{A}(\boldsymbol{\zeta}[k])\tau_\mathrm{s}}\mathrm{d}\tau_\mathrm{s}\boldsymbol{B}\big(\boldsymbol{\zeta}[k]\big),\end{aligned} \tag{3.35}$$

respectively, with T_s representing the sample time and k the sample index, have to be recomputed for each scheduling variable change, which is computationally taxing. A common first-order approximation is, therefore, denoted by the explicit Euler method

$$\begin{aligned}\boldsymbol{\Phi}\big(\boldsymbol{\zeta}[k]\big) &= \boldsymbol{I} + T_\mathrm{s}\boldsymbol{A}\big(\boldsymbol{\zeta}[k]\big),\\ \boldsymbol{H}\big(\boldsymbol{\zeta}[k]\big) &= T_\mathrm{s}\boldsymbol{B}\big(\boldsymbol{\zeta}[k]\big),\end{aligned} \tag{3.36}$$

with $\boldsymbol{I}$ being the identity matrix.

Node association of losses

In (3.33), it is assumed that the losses generated in each temperature node are known through the input vector $\boldsymbol{u}(t)$. Then, only the identification of $\boldsymbol{A}\big(\boldsymbol{\zeta}(t)\big)$ and $\boldsymbol{B}\big(\boldsymbol{\zeta}(t)\big)$ would be central to thermal modeling with LPTN structures. This denotes a common assumption in thermal modeling literature. However, depending on the abstraction level of the LPTN, associating losses to each node is a challenging task in itself on top of assessing them in the motor in the first place. The problem of locally mapping losses arises for both situations, when they are estimated through the fundamental wave model (see Sec. 3.1.2), which gives a mere allocation of the total losses into copper and iron losses, or through measurement-based models. Moreover, potential estimation accuracy penalties may emerge even when a proper allocation is found, e.g., through expert knowledge. Inaccurate loss estimates may stem from approximations in the (analytic or numeric) loss model or the measurement equipment's capability. This factor is further complicated by the input space dimension, which spans a large excitation space to be covered that influences the motor's operation, and eventually lets loss LUTs soar in their memory demand [Wal21].

A viable approach is to include $P(t)$ in the identification task, such that its per-node trajectory is estimated with similar methods as those for the component temperatures. This could be done for example with black-box approaches like polynomials and a least-squares fit, or in conjunction with analytical iron and copper loss models [GWB20]. The advantage is a reduced memory footprint since no LUTs are required.

Data-driven LPTN topology synthesis

As can be seen in Fig. 3.6, the LPTN approach covers a wide range in the white- to gray-box spectrum. On the far white-box side, a LPTN can be designed from analytical formulae and heuristic heat coefficients alone, which are compiled from tables that give recommendations based on simplified geometry and material information (for which a comprehensive survey is given in [GK21]). While this allows for an early investigation in the design phase of an electric motor because no test bench is required, the implementation of white-box LPTNs is not recommended for real-time temperature monitoring [Wal21] due to

- an increased model order necessity in order to cover hot spots within components to the detriment of the computational load [Nat+13; QSD14; BRG20],
- parameter uncertainty as no application is perfectly reflected in literature tables, and many coefficients, especially those associated with convection and radiation, exhibit high sensitivity to the current operation point [Bog+19],

- the paramount importance of accurate loss information, which is challenging to obtain as mentioned in the previous section.

Gray-Box LPTNs, on the other hand, are common for the temperature monitoring task, because many of their flavors can satisfy the requirements outlined in the beginning of Sec. 3.2. This modeling technique amends the disadvantages of white-box LPTNs by utilizing empirical measurement data for fine-tuning initially determined model parameters. Through this adjustment process, the inadequateness of highly abstracted, low-order LPTNs designed from simplified geometry and material information can be overcome, and detecting hot spots in the system becomes feasible again.

A low-order structure can be synthesized either from iterative reduction of a given full-order, white-box LPTN, as shown in [Qi+16], or by direct synthesis from preliminary investigation of the motor's geometry, assuming the most relevant heat sources and paths [WB16a]. Having determined a network structure, the coefficients, i.e., thermal properties can be identified next. The dependence of the ODE coefficients in (3.33) on a scheduling variable makes it evident that the low-order LPTN ODE describes a (quasi-)LPV system. Identification of these coefficients is, thus, not trivially completed by formulating and solving a quadratic program as it would be possible for linear time-invariant (LTI) systems. However, a larger repertoire of identification tools exists due to the LPV characteristic than it would do for an entirely nonlinear, time-varying system.

Three identification methods could be outlined in [Wal21]: The local, global, and glocal approach. The local approach follows the paradigm of the so-called gain scheduling [HW15], where several local models are interpolated, and in which each model is associated with a certain operation region. For this it is assumed that a reasonable linearization in each operation region is possible, in which $\boldsymbol{\zeta}(t)$ can be rendered constant. In the global approach, a single LPTN model is identified where its coefficients can be composed of arbitrary nonlinear (lifting) functions of the observable system states and inputs. Lifting functions introduce additional coefficients for the sake of capturing the thermal properties' time-dependency. The design of lifting functions usually lends from heat transfer theory and their coefficients may get value boundaries imposed that are informed from literature. Moreover, such a problem formulation represents a global, constrained, multi-dimensional optimization problem, and the global optimal coefficient set is not guaranteed to be traceable even with contemporary tools in finite time for none but the simplest systems. However, this is the situation for almost all data-driven identification methods applied on non-LTI systems, and a sufficiently accurate local optimum denotes the conventional objective. Eventually, the glocal approach [GWB17] tries to be a compromise between the local and global approach. Here, a newly initialized global model with lifting functions is fitted on a set of locally applicable models. While a superior estimation performance might result from a glocal approach, it must be noted that its design also requires the full effort of both paradigms, local and global identification, with no guarantee for improved performance.

4 Machine learning and its applications in the engineering field

The term machine learning (ML) generally describes the process in which a program with some degrees of freedom is adapted according to a predetermined metric and certain optimization rules, while given a data set or a stream of data. The degrees of freedom can be floating-point model coefficients, model structures, the amount of sub-models that form an ensemble, a certain selection of features to include, or other higher-order hyperparameters. Metrics or cost functions are chosen in an attempt to adequately measure the goodness of fit of a model with respect to the observational data, which is commonly a distance function between model output (estimates or predictions) and the targeted, dependent variable (usually a difficult-to-obtain value). Common for all ML approaches is the iterative process of learning, in which the degrees of freedom are adjusted repeatedly by scanning through the data either randomly and repetitively, if it is a completed set, or continuously in case of a data stream. This process is commonly called *training* of a model or algorithm. Since the facilitated paradigm of learning relies more often than not on statistical assumptions, it is fundamental to work with large data sets in order to properly utilize the smoothing properties of the law of large numbers. Moreover, the data used for the learning process should be collected from a representative source in order to ensure a similar data distribution between the set used for model training and whatever the model has to deal with in production afterwards (during *inference*). Any machine learning application can be classified into three distinct tasks: supervised learning, unsupervised learning, and reinforcement learning (see Fig. 4.1).

In supervised learning applications, several independent variables and one or more dependent variables within a data set are declared beforehand. Then, the goal is to find a mapping function between these two groups of variables (equivalently, *features*) with function approximators. This task corresponds to system identification in an engineering context, where supervised learning techniques are especially applicable to nonlinear dynamic systems [Nel01].

Unsupervised learning, on the other hand, deals with the structuring, aggregation and filtering of a data set with no declaration of in-/dependent variables at any time. Common metrics for the goodness of fit in unsupervised learning are based on similarity between samples. The general intention is to detect outliers, identify operation modes, find personas in case of user data, or to find a meaningful orientation for dimensionality reduction techniques, among other goals. Useful applications in engineering would be, e.g., fault detection, condition monitoring, and predictive maintenance.

In reinforcement learning (RL), eventually, algorithms act on the premise that previously chosen actions or actuating values will affect future states observed in a system or environment. Training of RL algorithms is based on maximizing a reward signal, which either stems naturally from the environment (e.g., highscore in video games, winning a board game), or can be engineered and customized to a certain task (e.g., minimize control costs or control errors, maximize trading return, or maximize the traffic flow). Obviously, the connection to the field of engineering can be found in optimal control, where usually a cost function is minimized. Note that minimizing a function is equivalent to maximizing the same function with a negative sign.

These three ML disciplines are not necessarily mutually exclusive: a RL agent might construct through supervised learning an internal model of the environment it interacts with to predict the environment's dynamics. A condition monitoring application might put its focus on outlier detection by observing system states that are estimated through a previously offline-trained model by means of supervised learning. A RL

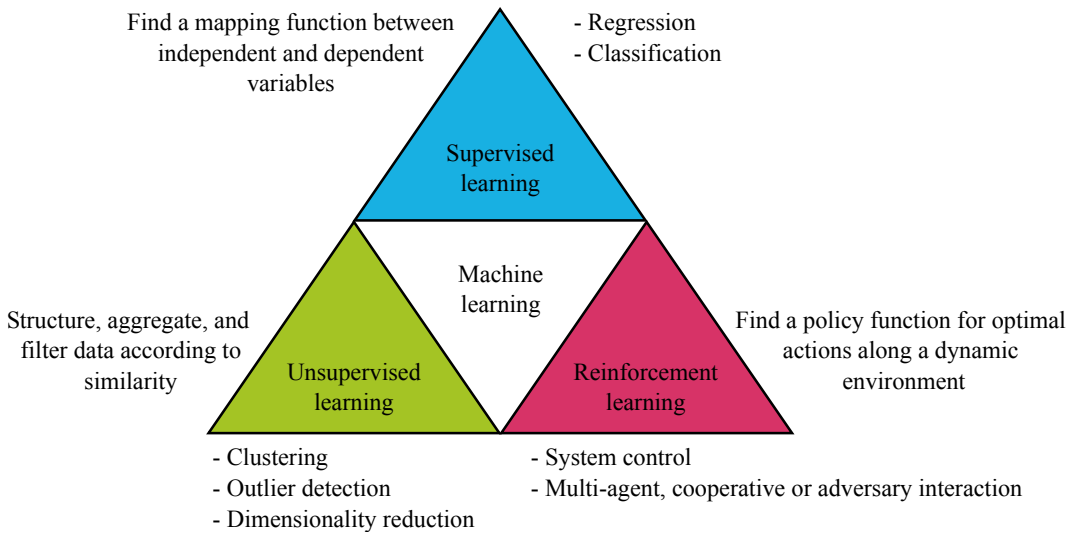


Figure 4.1: The three major categories of ML (derived from: [Wal+])

control algorithm might be fed with an observation vector containing not only environmental states, but also engineered signals indicating the current distance from predetermined cluster centers that were obtained from unsupervised learning.

4.1 Current and past trends

New achievements of ML applications make headlines regularly in news articles and magazines nowadays. These are often attributed to the field of computer science, and not infrequently enabled by novel, accelerated hardware. Instances of such applications with notable public attention are not only recommender systems in general [PAC18] – for which examples are ad placement [Lac+06; Ana+09], web search [BFJ96; Hua+13] and spam filters [GC09] – but also the field of natural language processing (NLP) [Wol+20; AM20; Bro+20b] and computer vision [KSH12; He+15a; Xu+15a; Min+21; WD21] provide an inexhaustible source of public-relations-effective innovation. These have a rather direct impact on consumer-facing products and services, such as digital assistants, chatbots, translators, phone use and its security (face recognition) or social networks. Likewise, generative modeling for perceptible signals (sound, images, and video) enjoy high public interest for its promise to take some, if not most, of its users' creative burden. Notable mentions are DALL-E 2 [Ram+22] and Stable Diffusion [Rom+22] for image generation from text, Google's DeepDream (originally Inception [Sze+15]) for image style altering, or MuseNet [PSA20] for music composition. Recently, chat bots such as ChatGPT and its successor GPT-4 caused a big stir in media, industry, and academia [Ope23]. Not of less public interest is super-human, competitive board and video game play, e.g., in Go [Sil+16] or Starcraft II [Vin+19].

Under the surface of the public, nonacademic awareness, however, there is an upward trend of research articles published on the use of ML for solving complex problems in all research domains including computer science. For instance, ML approaches find themselves more and more in the pharmacological domain, e.g., for drug design and discovery [Lip+19; Iru+20]. Medical imaging benefits from advances in image processing with ML [SWS17]. Fintech applications such as credit default risk analysis [AGH18], fraud detection [WB16b] and stock trading [DD16; Wu+20] have seen a surge of ML applications recently. Within engineering fields – into which this work is also to be classified – notable achievements in complex problem solving with ML can be found in

- automatic speech recognition [Hae+21],
- nonlinear dynamic system identification [Nel01], e.g., for
  - heating, ventilation, and air conditioning (HVAC) [Xu+15b; Laz+18],
  - internal system component temperature estimation [Zha+18; KWB20],
  - state-of-charge estimation in batteries [Che+18],
  - energy demand forecasting [Gha+17],
  - viscosity estimation in a polymer reactor [Mün+20],
- control tasks, e.g., in
  - robotics and motion [KBP13],
  - high-speed and efficient control of electric traction motors [Boo+21; SW21],
  - magnetic control of a nuclear fusion reactor [Deg+22],

just to name a few. Even in the relatively small share of academic literature denoted by the field around electric drives there have been hundreds of ML applications identified in [ZWP23; BMH22].

Yet, ML-based algorithms went a long way to accomplish today's methodological heights, which were often driven by research efforts in the control theory domain including its subfields, that are, system identification and optimization. In fact, artificial intelligence (AI)-based innovations emerged in waves over the past 60 years [GBC16; Fra20]: One of the earliest methods is Frank Rosenblatt's perceptron [Ros58] developed in the 1950s, which denotes the base of nowadays' artificial neural networks (ANNs), and was inspired by neuro science and the concept of neural links in mammal brains. Moreover, the fundamentals of support vector machines (SVMs) were proposed in the 60s already [Vap63], which would gain much attention later in the 90s together with ANNs [CV95; Bis06; HTF09]. Before that, however, plenty of research funding in the ML domain had declined, mostly due to the conclusion in [MP69] stating that perceptrons – which are, in essence, linear models – were inherently incapable of estimating exclusive OR (XOR) or negated XOR functional relationships.

Then, another soar in research interest in learning with ANNs was initiated with the general function approximation theorem in the late 80s [HSW89]. It states that ANNs with a single hidden layer and sufficiently many neurons can approximate arbitrary nonlinear functions from and onto finite multidimensional spaces arbitrarily precise. Increased research effort during that time led to milestones such as the error backpropagation algorithm [RHW86], and improvements thereof [Cha88; SA90], which made the compute-efficient retrieval of gradients for ANN weights and biases more tractable. Beyond that, recurrent neural networks (RNNs) [Lin+96; SP97], their training with truncated backpropagation through time (TBPTT) [Pin89; Wer90; WP90], memory-enhanced RNNs [HS97; GC99], and convolutional neural networks [LeC+89; LeC+98] were introduced during that time. All these innovations would gain again a lot of attention and use in applications only much later in the 2010s [Gu+18; Yu+19]. The popularity in ANN-based models would decline again in the late ninetees, for the time being, as ambitious goals of AI-based ventures would not meet the unrealistic expectations they had set out.

During the late ninetees and the beginning of the next millenium, statistical modeling with support vector machines (SVMs) and linear regression had their peak [Bis06; HTF09]. With the turn of the millenium, however, hardware resources began to improve in performance significantly with a drop in costs at the same time, enabling the emergance of *big data*. Innovative infrastructure was developed to accommodate unprecedented amounts of structured and unstructured data in data warehouses with technologies such as MapReduce [DG08] and Hadoop [Shv+10]. Costs for disk space and memory plummeted while graphic processing units (GPUs), especially by NVIDIA, proved to accelerate not only graphic processing but also large matrix multiplications, which are at the heart of ANN training [Owe+08]. This development has set the foundation for another wave of research interest into ANN methods that would last to date, and goes under the term *deep learning* [GBC16]. It comprises all efforts of training ANNs with many hidden layers, i.e., from approximately seven up to hundreds of layers. Their training was demonstrated to be feasible the first time in deep-belief networks and a so-called greedy layer-wise pretraining [HOT06], and successful end-to-end

training of many layers followed soon after through diverse regularization techniques [IS15; SK16; LKH16; Hof+19]. The popularity of deep learning holds to the time of this writing not least due to its unprecedented achieved accuracy in many classification, regression, as well as optimal sequential decision tasks.

Other recent leaps in ML research advancement are certainly denoted by the so-called attention mechanism for sequence modeling (e.g., language modeling) and by the transformer architectures derived from that [Cho+15; Vas+17; Wol+20]. Not only do they represent a paradigm shift for tasks as language understanding and language generation away from bi-directional RNNs and convolutional neural networks, but also started an arguable scaling race in ever increasing accuracy by likewise increasing model sizes. At the time of this writing, the most accurate language models are, among others, BERT [Dev+18] with its larger variant holding 340 million model parameters, and GPT-3 [Bro+20b] with 175 billion model parameters. GPT-4 is rumoured to exhibit even a manifold of the amount of GPT-3. Although transformers in NLP are amenable to pretraining, which fosters large-scale distribution of a once trained model, the magnitude of the amount of necessary trainable parameters pushes research in this field into dimensions where it becomes exclusive to institutes that can afford the necessary accelerators and infrastructure. This is not a trend without concern [TY23].

In the classical control engineering domain, however, many of the parameter-intensive trends are only slowly adopted, if at all. An obvious reason for hesitation is the stricter constraint on computational complexity as many control applications are still limited to best-cost hardware (as already mentioned in previous chapters). These application-specific challenges ask for especially lean and fast methods with low model parameter counts, and explain the unaligned focus between applications in control engineering and computer science.

Despite these circumstances, both fields of research tend to draw closer to each other methodically and start to appreciate the other's established milestones such that interesting methods at the intersection arise, e.g., the combination of differential equations with ANNs (see Sec. 4.4). More innovations in this intersection are to be expected in the future, which places the demand for multidisciplinary expertise on future researching and practicing engineers.

4.2 Supervised learning

The task of supervised learning is most comparable to the engineering discipline of system identification, although for supervised learning there is technically not necessarily a system to be identified but rather a functional map from independent variables to one or more dependent ones. These variables might represent very abstract quantities ranging from sales volume, ad click-through rates, housing properties such as size and perceived neighborhood quality, up to molecular magnetic coupling strengths.

Independent of those features available, it is characteristic of supervised learning to work with a finite labeled data set $\langle \boldsymbol{u}_k, \boldsymbol{y}_k \rangle \in \mathcal{D}$ with $k \in [0, K-1]$ and K being the amount of available samples. Note that $\boldsymbol{u}$ is used as notation for the set of independent variables instead of the more common $\boldsymbol{x}$ in computer science, since $\boldsymbol{x}$ will be reserved for a dynamic system's state where $\boldsymbol{u}$ is the system's input as is common in control theory. The goal of supervised learning is then to find a parameterizable approximation function $\boldsymbol{f}_{\boldsymbol{w}} : \boldsymbol{u}_k \rightarrow \hat{\boldsymbol{y}}_k$ with a finite set of parameters $\boldsymbol{w}$ for the perfect map $\boldsymbol{f}^* : \boldsymbol{u}_k \rightarrow \boldsymbol{y}_k$, in the hope that new unlabeled instances of the set of independent variables can be transformed to the corresponding dependent variable sufficiently accurately. Note the hat notation for estimated variables. All samples of a single input feature will be denoted by the column vector $\boldsymbol{u}_p \in \mathbb{R}^K$ with $p \in [0, P-1]$, assuming there are P independent variables, while a dependent variable vector for all data set instances will be $\boldsymbol{y}_q \in \mathbb{R}^K$ for Q targets each in likewise fashion or just $\boldsymbol{y}$ without a subindex if there is only one target variable $Q = 1$. The entire regressor matrix comprising all regressors across all samples will be $\boldsymbol{U} \in \mathbb{R}^{K \times P}$ and all target variable instances denote the matrix $\boldsymbol{Y} \in \mathbb{R}^{K \times Q}$. Random variables will be capital and in non-bold font like in R. The approximation error is made accessible by a predetermined cost function $\mathcal{L}(\hat{\boldsymbol{y}}_k, \boldsymbol{y}_k) : \mathbb{R}^Q \times \mathbb{R}^Q \rightarrow \mathbb{R}$ indicating the adequateness of the approximation.

The dependent variables $\boldsymbol{y}$ might either be real-valued continuous quantities or of discrete nature, e.g., integer enumerated or boolean flags, which is also called *one-hot* encoding. In case of the former, the supervised learning task is specified as regression task, which is dominant in classical control engineering problems, whereas in case of the latter it is called a classification task. For the remainder of this chapter it is assumed that a regression problem is at hand since it corresponds to this work's investigation of thermal modeling.

4.2.1 Decomposition of the generalization error

The motivation behind drawing on supervised learning methods is not to just observe the learning process itself but rather to have a sufficiently well approximated function as a result, which can be applied on unseen samples, that is, instances $\boldsymbol{u}_k$ that might not have been seen during training and for which no labels exist. Therefore, the overall goal is to train a function approximator in such a way that its generalization capability is maximized.

The most prevalent cost function for regression tasks is the mean squared error (MSE), and it is the cost of choice also throughout this thesis due to its desirable attribute of penalizing large deviations significantly stronger than small deviations. The MSE describes the expected quadratic residuals of the difference between two random variables R_1 and R_2

$$\mathcal{L}_{\mathrm{MSE}}(R_1, R_2) = \mathbb{E}\left[\left(R_1 - R_2\right)^2\right], \tag{4.1}$$

or, equivalently for our finite data set $\mathcal{D}$ with labels $\boldsymbol{y}$ and estimates $\hat{\boldsymbol{y}}$ – assuming one dependent variable – the average over the residual sum of squares (RSS) of their difference

$$\mathcal{L}_{\mathrm{MSE};\mathcal{D}} = \frac{1}{K}\sum_{k=0}^{K-1}(y_k - \hat{y}_k)^2 = \frac{1}{K}\sum_{k=0}^{K-1}(y_k - f_{\boldsymbol{w}}(\boldsymbol{u}_k))^2. \tag{4.2}$$

Note that $\mathbb{E}$ is the expectation operator. In case of multiple targets, the MSE for each target is averaged across to a scalar as well. Therefore, the MSE cost is applicable to single samples of a data set, the whole data set, or grouped samples thereof which are labeled, e.g., as training and test set in order to perform cross-validation (CV), which emulates the generalization scenario (more to that in Sec. 4.2.2).

Several factors add to the degradation of a model's generalization capability, which can be classified into two categories: those pertaining to flaws in the experimental setup and the others in the design of the model architecture and its optimization. Measurement noise, missing correlation, and the difference in data distribution correspond to the former, while errors due to early stopping in optimization, convergence to local minima, and over- or underfitting correspond to the latter, compare Tab. 4.1. These error components are further illustrated in Fig. 4.2 and illuminated in the following.

Model design	Experimental setup
Under- and overfitting	Measurement noise
Early stopping	Missing correlation
Local minima	Data distribution differences

Table 4.1: The generalization error is composed of different factors that can be classified into two categories.

The early stopping error is closely related to the local minimum error, which is immanent to any iterative optimization technique for non-convex cost functions [NW06]. Since most of the useful regression algorithms start with many randomly initialized coefficients, which are then to be optimized iteratively if there is no closed-form solution, and most cost functions exhibit a highly non-convex and high-dimensional error landscape riddled with local minima, it is likely that the optimization ends up in such a local minimum after a certain amount of optimization steps. Stopping the optimization too early is also probable, if the learning rate is or

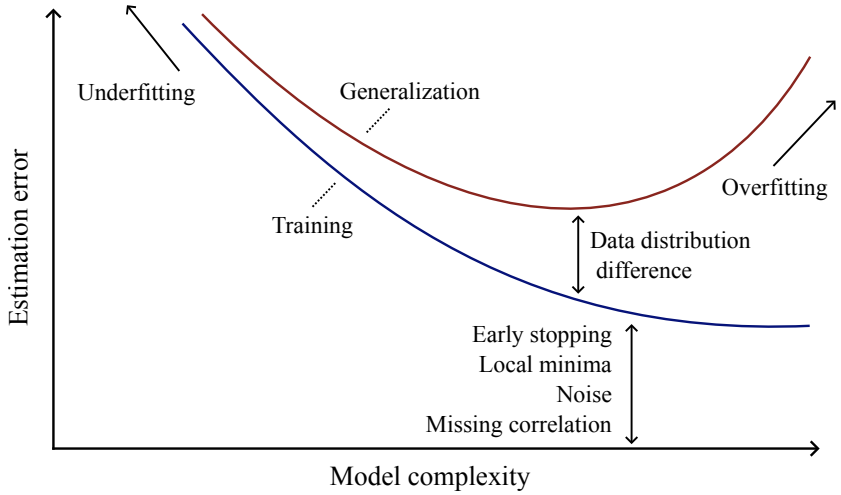


Figure 4.2: Decomposition of the generalization and training error (derived from: [HTF09])

becomes too small over the course of optimization in order to escape local minima or error plateaus with no substantial negative gradient into any direction. The only practical remedy is to either repeat the optimization from different starting points and pick the best model, or to improve the optimizing algorithm, e.g., with momentum terms [Sut+13; KB17]. However, the fundamental problem cannot be avoided.

It is very well possible for any given parameterizable function to *overfit* the data set $\mathcal{D}$ during training, such that any instance within the data set is learned as much by heart as possible by the function structure but any unseen sample will be estimated poorly [Dom12]. Such a case is highly undesirable and might even mislead the applying researcher about the model's estimation accuracy when not taken care of. On the other hand, too few of degrees of freedom in an approximating function might also lead to a poor estimation, and is termed *underfitting*. Instances of such are, e.g., too few hidden layers or neurons in a neural network or the application of a linear model to a nonlinear relationship between system input and output. Formally [HTF09], the over- and underfitting problem can be decomposed into a bias-variance-tradeoff if one uses the MSE cost and assumes the ground truth labels to stem from an additive, statistical model $g(\cdot)$ with the measured output denoting the random variable $Y = g(U) + E$, where E with $\mathbb{E}[E] = 0$ stands for all other parasitic effects, such as measurement noise. Based on such assumptions one can show that

$$\begin{aligned}\mathcal{L}_{\mathrm{MSE}}\big(Y, f_{\boldsymbol{w}}(\boldsymbol{U})\big) &= \mathbb{E}\Big[\big(g(\boldsymbol{U}) + E - f_{\boldsymbol{w}}(\boldsymbol{U})\big)^2\Big] \\ &= \underbrace{\Big(g(\boldsymbol{U}) - \mathbb{E}\big[f_{\boldsymbol{w}}(\boldsymbol{U})\big]\Big)^2}_{\text{Squared bias of } f_{\boldsymbol{w}}} + \underbrace{\mathbb{E}\Big[\Big(\mathbb{E}\big[f_{\boldsymbol{w}}(\boldsymbol{U})\big] - f_{\boldsymbol{w}}(\boldsymbol{U})\Big)^2\Big]}_{\text{Variance of } f_{\boldsymbol{w}}} + \underbrace{\mathbb{E}\big[E^2\big]}_{\text{Variance of noise}} .\end{aligned}$$

The approximated model's estimates' contribution to the MSE cost are, thus, decomposable into a bias and variance term, which are usually to be traded off through the model's topological complexity or the choice of modeling algorithm altogether. The particular choice of modeling algorithm is said to span an already constrained hypothesis space, which might not include the true mapping [Dom12]. High bias and low variance is generally associated with underfitting, whereas low bias and high variance with overfitting, accordingly.

In real-world systems, however, it is seldomly the case that the observed sensory samples stem from a bias-free random distribution, and further effects add up to approximation errors. One of the most humbling risks is the potential independence between the set of available sensors for the (random) model input $\boldsymbol{U}$ and the sensor reading of interest Y, that is, the joint cumulative probability distribution function would be denoted by

$$\mathrm{Pr}_{\boldsymbol{U},Y}(u, y) = \mathrm{Pr}_{\boldsymbol{U}}(u)\mathrm{Pr}_{Y}(y) \quad \forall \quad u, y. \tag{4.3}$$

From an empirical point of view, this corresponds to sensor data representing the dependent variable not correlating neither linearly nor nonlinearly with sensor data that denotes the input to a model, not to mention any causal relationship. Approximation errors due to such a lack of correlation are summarized as missing correlation errors. Also, non-detectable events that have a substantial impact on the target fall into this category.

Another error component corresponding to the experimental setup is the potential deviation between the data distribution as measured on a test bench or similar in the laboratory and the distribution of samples observed "on the field", e.g., in a series vehicle driving on a road. A mismatch between the so obtained training set and the set of data that is eventually to be estimated on is sometimes called out-of-distribution estimation, and corresponds to an ML model to be forced to extrapolate rather than interpolate from seen data during training. ML models are known to be good at interpolating but deficient at extrapolating [Dom12], which is again related to overfitting. This even becomes evident when cross-validation is performed with data sets from the same test bench but different environmental situations that influence the data distribution between the two sets. Hence, in contrast to the classic field of optimization, the task of ML is actually to optimize a function which is not known (the estimation error on true observables later on the field), and the error on the training set is only a surrogate. This fact might then again mitigate the local minima problem, since it is not necessary to find the very global optimum but more likely the right point in the vicinity.

4.2.2 Cross-validation

In order to not fall for over- or underfitting, several techniques for the evaluation of any ML method became best practices and can be summarized under the term cross-validation (CV) [Koh95; HTF09]. The most standard form of cross-validation is coined *k-fold CV*, which describes the separation of all data set samples into k groups or folds, that are utilized as test or validation set sequentially, while all other k-1 folds will represent the training set during each fold. This will produce k different models with usually different performances. Typically, the test set performance will be worse since samples thereof were not seen during training and are generalized upon. The gap between test and training set performance indicates the severity of overfitting. The reason behind a k-fold repetition is to alleviate the risk to misinterpret a model's performance due to relatively many quirks in the data of one fold. A ML engineer would have to average these performances to get the overall performance or further analyse the deviation dependent on the constructed fold. Note that a performance deviation between folds might also stem from the random coefficient initalization and the subsequently different local minima into which training has converged (in case of iterative optimization methods). Thus, averaging performances during cross-validation should happen for the best models out of many that were differently seeded in order to mitigate the local minima optimization problem in the equation.

Another extension is the grouped k-fold CV, which is advisable when the engineer has realized that the data set at hand features different modes, that is, environmental factors, e.g., a specific measurement day or a certain system among a set of systems that influence the data distribution significantly. Grouped k-fold CV will sort the data such that each group will appear evenly in each fold, in order to avoid splits that enforce out-of-distribution predictions and where the model would be evaluated on its extrapolating capability. This is to be avoided when it can be assumed that the data set will feature all modes that are to be expected during the deployment of the ML model. Otherwise, when this question cannot be answered certainly, falling back to standard k-fold CV can give a better indication of how the model will deal with unseen environmental variables.

The mentioned CV strategies work especially well for independent and identically distributed (i.i.d.) data, where each row in a data table (i.e., sample) can be exchanged with any other row without breaking causality assumptions. Static models assume this property as well. However, for time series and dynamic data this method is inadequate as this might break the coherence in the time domain. For forecasting tasks, there is a convention to design the cross-validation folds to be fixed, bound to contiguous sample series, and facilitating

a strictly increasing time index for the relationship between training and test set. In such CV schemes a ML model would always estimate future samples from past observations. This is to accommodate the fact that during deployment the ML model would also be to predict samples in the future without ever having to estimate past observations.

In this work, however, we deal with sequence-to-sequence modeling, where an identified model is evaluated for each instance in time consecutively in real-time. ECU sensor information that are available in series vehicles will inform each current temperature estimate. The CV scheme has to take this into account, and will be outlined in Sec. 6.2.

4.3 Black-box modeling

Black-box models can be characterized by their general-purpose applicability, e.g., their use merely requires the declaration of input and output vectors in a given data set $\langle \boldsymbol{u}_k, \boldsymbol{y}_k \rangle \in \mathcal{D}$. No domain knowledge has to be applied in order to conduct training and testing (see Fig. 3.1), which contributes to their popularity. On the other hand, high estimation accuracies are not guaranteed due to a large nonquadratic search space spanned by many model coefficients, potentially application-inappropriate model structures, and possibly insufficient volumes of observational data, for which a higher amount is necessary.

4.3.1 Linear regression

Among the various modeling techniques the ML domain has to offer, linear regression forms one of the simplest, yet still powerful approach. It pertains to a model family of linear approximators that assume a linear relationship between real-valued input vectors $\boldsymbol{U}$ and the real-valued (possibly multi-dimensional) output vector $\boldsymbol{y}$, whereas multiple dependent variables correspond to as many linear models with each their own set of model coefficients. The trainable coefficients $\boldsymbol{\beta} \in \mathbb{R}^{P+1}$ for one certain linear model describe the output function $f_{\text{linear}} : \mathbb{R}^P \rightarrow \mathbb{R}, \boldsymbol{u}_k \rightarrow \hat{y}_k$ applied on observation k [HTF09]:

$$f_{\text{linear}}(\boldsymbol{u}_k) = \beta_0 + \sum_{p=1}^{P} u_{k,p}\beta_p = \beta_0 + \boldsymbol{u}_k^{\mathsf{T}}\boldsymbol{\beta}_{[1:P]} = \tilde{\boldsymbol{u}}_k^{\mathsf{T}}\boldsymbol{\beta}, \tag{4.4}$$

with $\tilde{\boldsymbol{u}}_k = [1, u_{k,0}, u_{k,1}, \ldots, u_{k,P-1}]$ denoting the expanded input vector to accommodate the bias coefficient β_0. Estimating $\boldsymbol{\beta}$ from K samples in the training data, one can choose from several methods, whereas the most popular one is the least squares method, which minimizes the residual sum of squares (RSS):

$$\mathcal{L}_{\text{RSS}}(\boldsymbol{\beta}) = \sum_{k=1}^{K}(y_k - f_{\text{linear}}(\boldsymbol{u}_k))^2, \tag{4.5}$$

$$= (\boldsymbol{y} - \tilde{\boldsymbol{U}}\boldsymbol{\beta})^{\mathsf{T}}(\boldsymbol{y} - \tilde{\boldsymbol{U}}\boldsymbol{\beta}). \tag{4.6}$$

Setting the gradient of (4.6) with respect to $\boldsymbol{\beta}$ equal to zero, while assuming $\tilde{\boldsymbol{U}}^{\mathsf{T}}\tilde{\boldsymbol{U}}$ to be positive definite, yields the following analytically-closed solution form for estimations $\hat{\boldsymbol{y}}$:

$$\hat{\boldsymbol{y}} = \tilde{\boldsymbol{U}}\hat{\boldsymbol{\beta}} = \tilde{\boldsymbol{U}}(\tilde{\boldsymbol{U}}^{\mathsf{T}}\tilde{\boldsymbol{U}})^{-1}\tilde{\boldsymbol{U}}^{\mathsf{T}}\boldsymbol{y}. \tag{4.7}$$

This modeling approach is time-invariant yet not scale-invariant, and standardizing each feature vector $\boldsymbol{u}_p$ is beneficial.

The least squares approach (4.6) or, equivalently, ordinary least squares (OLS) denotes two drawbacks [HTF09] that become more severe as soon as the feature vectors are not orthonormal (multicollinearity), which is the

case for most real-world data. In such scenarios, coefficient estimation accuracy suffers from high variance, and, consequently, the absolute coefficient values are not as interpretable anymore. Adding an additional penalty term to (4.6) imposes shrinkage gradually stricter to those coefficients that adhere to features with significant multicollinearity. This lowers variance and increases bias in model coefficient determination, albeit it may lead to more solid interpretation of each explanatory variable when evaluating their regression coefficient [HTF09]. In literature, one often finds the ℓ_1 and ℓ_2 norm, which are also referred to as least absolute shrinkage and selection operator (LASSO) [Tib96] and RIDGE [HK70], as possible penalty choices:

$$\mathcal{L}_{\text{RSS; Lasso}}(\boldsymbol{\beta}) = (\boldsymbol{y} - \tilde{\boldsymbol{U}}\boldsymbol{\beta})^{\mathsf{T}}(\boldsymbol{y} - \tilde{\boldsymbol{U}}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_1, \tag{4.8}$$

$$\mathcal{L}_{\text{RSS; Ridge}}(\boldsymbol{\beta}) = (\boldsymbol{y} - \tilde{\boldsymbol{U}}\boldsymbol{\beta})^{\mathsf{T}}(\boldsymbol{y} - \tilde{\boldsymbol{U}}\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_2^2, \tag{4.9}$$

where λ controls the shrinkage penalty weight. In machine learning literature, the ℓ_2 penalty is also coined as *weight decay*. Independent of a cost penalty, the model size complexity of linear models is $\mathcal{O}(Q \cdot (P+1))$.

4.3.2 Artificial neural networks

Artificial neural networks (ANNs), or short, neural networks (NNs), gained high popularity during the last decade (again, see Sec. 4.1), especially due to their exceptional estimation performance in tasks within the field of image and natural language processing, or related time-series modeling. They are not the all-in-one solution for any problem – notably tabular i.i.d. data with static relationships between independent and dependent variables are dominated by tree-based approaches. These will be treated briefly in the next section (Sec. 4.3.3). However, since most control engineering applications and especially temperature monitoring deal with time series data, ANNs play a pivotal role for this work.

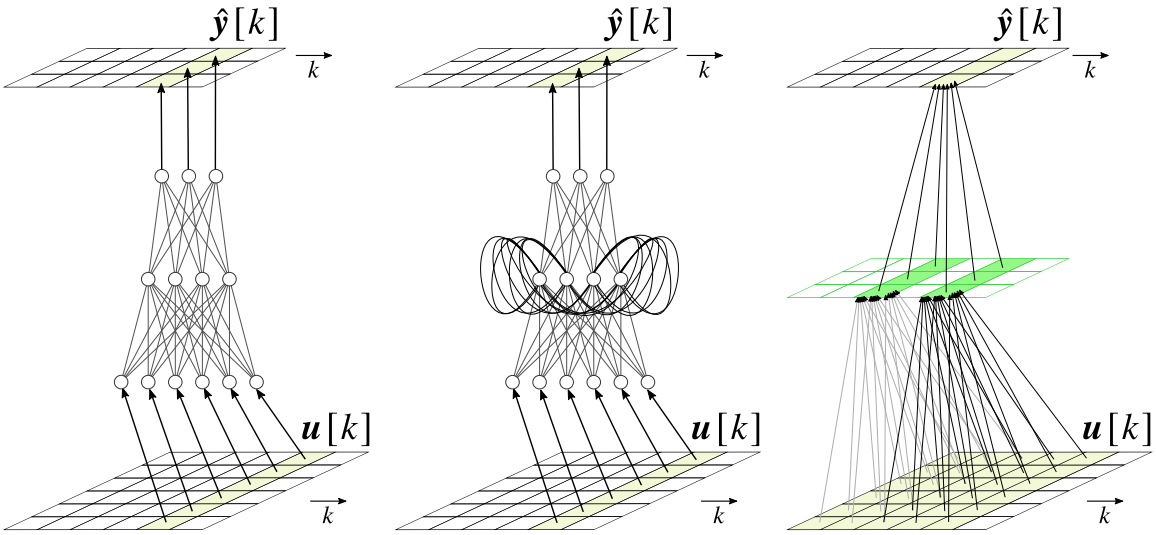
Architectures

Due to ANN's repeatedly emerging attention within academia, plenty of new architectures are proposed regularly. The most common topologies are that of the multi-layer perceptron (MLP), the recurrent neural network (RNN) – often with memory cells such as the long short-term memory (LSTM) or gated recurrent unit (GRU) – and convolutional neural networks (CNNs) [GBC16]. An overview is given in Fig. 4.3. These architectures are evaluated in this work, and, hence, briefly discussed in the following.

It is fundamental to ANN topologies that they are modularly decomposable into arbitrarily many layers, which would be stacked onto each other. Each layer describes a differentiable nonlinear transformation of the previous layer's feature set onto a new output feature set. This transformation commonly includes differently weighted sums of the input tensor (i.e., the multi-dimensional array) that are often represented as *neurons*. Sometimes the label ANN is extended to the application of automatic differentiation frameworks that merely require an end-to-end differentiable algorithm, such that the boundaries of what is understood as ANN can blur. Howsoever, the MLP algorithm is unambiguously defined for a certain layer l, where the previous layer's activation $\boldsymbol{h}^{(l-1)} \in \mathbb{R}^m$ is mapped to the layer's output feature vector $\boldsymbol{h}^{(l)} \in \mathbb{R}^n$ at time k, to be

$$\boldsymbol{h}^{(l)}[k] = \boldsymbol{g}\Big(\boldsymbol{W}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{w}_0\Big) \tag{4.10}$$

where $\boldsymbol{g}(\cdot)$ is an element-wise-applied nonlinear function, called the *activation function*, e.g., the Sigmoid σ function or the hyperbolic tangent (tanh), and $\boldsymbol{W} \in \mathbb{R}^{n \times m}$ is the trainable weight matrix, that is, the set of model coefficients that can be identified according to the data set $\mathcal{D}$, and $\boldsymbol{w}_0 \in \mathbb{R}^n$ is the likewise trainable bias term. There is plenty of research published around finding more appropriate activation functions for various applications, such as the rectified linear unit (ReLU) [NH10], the exponential linear unit (ELU) [CUH15], sinusoidal functions [Sit+20], or leaky rectified linear unit (ReLU) [He+15b], among others.



a) Multilayer perceptron b) Recurrent NN c) Convolutional NN

Figure 4.3: Different ANN topologies (bias terms omitted to reduce clutter), a) MLP – one hidden and one output layer transform the input feature vector nonlinearly; b) RNN – a recurrent layer with feedback connections is followed by a feed-forward layer; c) CNN – 1-D and causal, where three filters with a kernel size of three are slided through a window of size five, followed by three filters with a dilated kernel of size two (only weighted connections of one filter per layer are shown to ease clutter).

The MLP is a static model, and, hence, does not exploit the strong correlation in the time domain between consecutive samples, which is inherent in measurement sequences especially from physical systems that obey causality and finite dynamics. The RNNs were invented to utilize this property by facilitating feedback loops from neuron outputs to their corresponding input

$$\boldsymbol{h}^{(l)}[k] = \boldsymbol{f}_{\boldsymbol{\theta}}\left(\boldsymbol{u}[k], \boldsymbol{h}^{(l)}[k-1]\right) = \boldsymbol{g}\left(\boldsymbol{W}_{\mathrm{r}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{W}_{\mathrm{f}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{w}_0\right), \tag{4.11}$$

with $\boldsymbol{W}_{\mathrm{r}} \in \mathbb{R}^{n\times n}$ being the weight matrix corresponding to the recurrent connection and $\boldsymbol{W}_{\mathrm{f}} \in \mathbb{R}^{n\times m}$ to the forward connections, and $\boldsymbol{\theta}$ comprising all trainable parameters. Training of recurrent structures typically involves not only backpropagation (see next section) but rather an extension called truncated backpropagation through time (TBPTT) [Wer90]. When computing the gradient of a cost function with respect to $\boldsymbol{\theta}$, not only is the contribution of $\boldsymbol{W}_{\mathrm{r}}$ at time k involved but also all other instances in time before that, up to a truncation time t_{trunc}. If truncation was to be omitted, this recursion would lead to a severe case of the vanishing or exploding gradient problem [BSF94; PMB13]. It still exists also with truncation though, and worsens the further into the past backpropagation descends, such that further measures usually need to be applied. One of those measures is a substantial expansion of recurrent neurons to multi-gated memory blocks, for which the LSTM is very

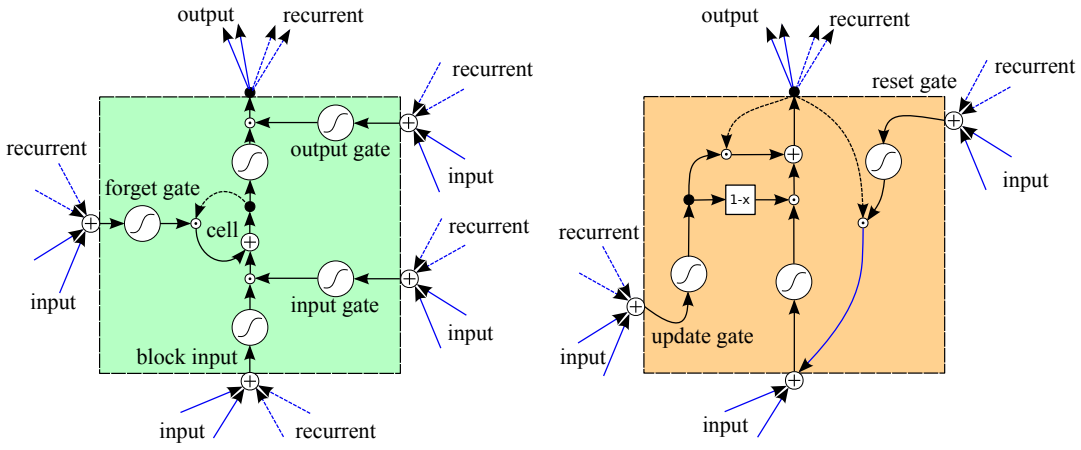
popular [HS97; GC99] and is described by

$$
\begin{aligned}
\boldsymbol{h}^{(l)}_{\text{input}}[k] &= \boldsymbol{\sigma}\Big(\boldsymbol{W}_{\text{f,input}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,input}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{w}_{0,\text{input}}\Big) \\
\boldsymbol{h}^{(l)}_{\text{forget}}[k] &= \boldsymbol{\sigma}\Big(\boldsymbol{W}_{\text{f,forget}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,forget}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{w}_{0,\text{forget}}\Big) \\
\boldsymbol{h}^{(l)}_{\text{output}}[k] &= \boldsymbol{\sigma}\Big(\boldsymbol{W}_{\text{f,output}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,output}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{w}_{0,\text{output}}\Big) \\
\boldsymbol{h}^{(l)}_{\text{gate}}[k] &= \tanh\Big(\boldsymbol{W}_{\text{f,gate}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,gate}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{w}_{0,\text{gate}}\Big) \\
\boldsymbol{h}^{(l)}_{\text{cell}}[k] &= \boldsymbol{h}^{(l)}_{\text{forget}}[k] \circ \boldsymbol{h}^{(l)}_{\text{cell}}[k-1] + \boldsymbol{h}^{(l)}_{\text{input}}[k] \circ \boldsymbol{h}^{(l)}_{\text{gate}}[k] \\
\boldsymbol{h}^{(l)}[k] &= \boldsymbol{h}^{(l)}_{\text{output}}[k] \circ \tanh\Big(\boldsymbol{h}^{(l)}_{\text{cell}}[k]\Big),
\end{aligned}
\tag{4.12}
$$

with $\circ$ representing element-wise multiplication. Having (learnable) gated entries controlling the information flow into the memory block for forward and recurrent connections effectively disentangles the recursion problem in the gradient calculation. Long-term events that are located further in the past of a sequence can be learned more effectively. The GRU architecture is a slight simplification compared to the LSTM topology [Cho+14], where the input and forget gates are coupled into an update gate and the output gate is renamed to the reset gate, which only scales the recurrent connection to the block input rather than the block output. The GRU update equation reads

$$
\begin{aligned}
\boldsymbol{h}^{(l)}_{\text{reset}}[k] &= \boldsymbol{\sigma}\Big(\boldsymbol{W}_{\text{f,reset}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,reset}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{w}_{0,\text{reset}}\Big) \\
\boldsymbol{h}^{(l)}_{\text{update}}[k] &= \boldsymbol{\sigma}\Big(\boldsymbol{W}_{\text{f,update}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,update}}\boldsymbol{h}^{(l)}[k-1] + \boldsymbol{w}_{0,\text{update}}\Big) \\
\boldsymbol{h}^{(l)}_{\text{cell}}[k] &= \tanh\Big(\boldsymbol{W}_{\text{f,cell}}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{W}_{\text{r,cell}}\big(\boldsymbol{h}^{(l)}_{\text{reset}}[k] \circ \boldsymbol{h}^{(l)}[k-1]\big) + \boldsymbol{w}_{0,\text{cell}}\Big) \\
\boldsymbol{h}^{(l)}[k] &= \boldsymbol{h}^{(l)}_{\text{update}}[k] \circ \boldsymbol{h}^{(l)}[k-1] + \big(\mathbb{1}_n - \boldsymbol{h}^{(l)}_{\text{update}}[k]\big) \circ \boldsymbol{h}^{(l)}_{\text{cell}}[k],
\end{aligned}
\tag{4.13}
$$

with $\mathbb{1}_n$ being a column vector of n ones. Both memory cell types are displayed in Fig. 4.4. Some research investigated the appropriateness of these ad-hoc topologies empirically [Chu+14], even with systematic searches for topological variations [JZS15; Gre+16]. However, no topology was found to be particularly better across many applications than the LSTM or the GRU.



a) LSTM b) GRU

Figure 4.4: Two different memory blocks that would replace each RNN neuron (source: [Gre+16]). Blue connections are weighted, and dashed connections are time-delayed by one sample.

Feeding an ANN not only with the current observation $\boldsymbol{u}_k$ but also with a set of previous observations $(\boldsymbol{u}_{k-1}, \boldsymbol{u}_{k-2}, \dots)$ as well suggests itself to incorporate past information into the current estimate. Some works consider this as an architectural extension, e.g., as in time-delay NNs [LWH90], or, if feeding also past estimates or available ground truth targets into the model, nonlinear autoregressive exogenous (NARX) models [Lin+96] or Elman NNs [Elm90]. However, others would interpret this as a form of feature engineering with the model architecture remaining the same. Such features would be denoted as *lag* or time-delayed input features and introduce substantially more model parameters in the first hidden layer the more past instances are considered.

A more parameter-efficient way of considering past information is described by the use of causal and temporal convolutional neural networks (TCNs) [BBO17; BKK18]. Convolutional neural networks (CNNs) – originally introduced in the field of image processing for utilizing local neighboring pixels within a sliding kernel [KSH12] – work in their 1-D temporal and causal version on the same principle by utilizing neighboring samples in a time sequence without looking ahead into the future. Instead of neurons, one would speak of filters, which inherit kernels of a certain size, which in turn determines the amount of past information considered into a certain intermediate filter activation. The first hidden layer's activations denote the convolution (more precisely, the cross-correlation) between learnable kernels and past time series data, i.e., an arbitrary amount of filters is slided through a time window of a certain length that produce a feature map of the same length. Upon this map, further kernels of the next hidden layer are applied in the same fashion up to the output activation, where a reduction operation can return the desired output dimensionality. Since kernels are effectively shared across instances within the windowed sequence, a considerable amount of parameters is saved compared to a likewise built time-delay NN. The so-called receptive field is further increased by dilatation, that is, kernels are not applied on strictly adjacent samples in later layers, but on those with an increasing distance to each other. In this memoryless architecture, zero-padding ensures that subsequent activation maps are of equal size, and no past estimates are utilized in contrast to NARX models or Elman nets. The dilatated, causal and temporal CNN update equation for the i-th filter's activation $h_i^{(l)}[k]$ at time k and layer l with the learnable coefficients $\boldsymbol{W}_i \in \mathbb{R}^{H \times \varsigma}$ applied on H previous layer's filters, a kernel size of ς, and the dilatation factor δ reads

$$h_i^{(l)}[k] = \sum_{p=0}^{H-1} \sum_{j=0}^{\varsigma-1} \boldsymbol{W}_{i;(p,j)} \cdot h_p^{(l-1)}[k - j\delta]. \tag{4.14}$$

Note that there are also other architectures one often comes across in literature, which are less applicable for the sequence-to-sequence modeling task at hand. For instance, there are liquid state machines [MNM02], extreme learning machines [HZS06], echo state networks [Jae01] and spiking NNs [Maa97] that all incorporate random sparsity in their partly recurrent topology by eliminating arbitrary connections between neurons. Imposing random structure into a NN, however, is less an attempt to approximate the physical behavior of an application and more one to reduce the model size at almost equal accuracies. In the case of the extreme learning machine, hidden layer connections are even not trained but assigned randomly, which practically renders this topology a linear model with randomly affine-transformed, higher-dimensional features. This (feature engineering) approach was even formalized into a framework called reservoir computing [SVV07]. Since none of these variations were shown to consistently outperform the base topologies mentioned above in a variety of relevant applications, they can be considered incremental, and are not further investigated in this work.

Then there are topologies for generative modeling like the generative adversarial network [Goo+14], which is only remotely interesting for control engineering applications, e.g., for design-of-experiments investigations or synthetic measurement data at most (for which simpler methods seem to be more constructive). Moreover, the variational auto-encoder (VAE) topology is also common in literature [TBL18; KW19]. However, the auto encoder's main goal is to find a lower-dimensional encoding for a set of input features, such that its application can be sorted to the field of representation learning, whereas the VAE is also often proposed for synthesizing observational data. The recent workhorse of natural language processing – transformers, which work with the

so-called attention mechanism [Vas+17] – fundamentally require full sequences to be available beforehand, which is not in accordance with real-time temperature monitoring. A more comprehensive overview of different topologies is elaborated in [BK22] and specifically of the transformer architecture in [Wol+20].

Optimization via gradient descent and automatic differentiation for compute-efficient gradient retrieval

ANNs are infamous for learning from vast volumes of data, that is, many samples and many features, that occupy terabytes of storage, which would need to be loaded into random access memory (RAM) chunk by chunk. Intriguingly, large ANNs tend to learn better approximations with more data. The combination of huge sets of model coefficients and observational data was not feasible up until recently with increased hardware performance and the emergence of automatic differentiation [Bay+18], for which error backpropagation is a special case.

The prevalent optimization scheme for training ANNs is (stochastic) gradient descent. Recall that an application-specific loss function as in (4.2) has to be chosen beforehand. Since big data sets will not fit into a computer's RAM at once, and matrix multiplication computation time does not scale linearly with the matrix dimension in contemporary hardware accelerators such as GPUs, a subset of the whole data set $\mathcal{B} \subset \mathcal{D}$, which is called the (mini-)batch, is commonly used for gradient descent updates

$$\frac{\partial \mathcal{L}_{\mathcal{D}}}{\partial \boldsymbol{\theta}} \approx \frac{\partial \mathcal{L}_{\mathcal{B}}}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{B}|} \sum_{\langle \boldsymbol{u}_k, \boldsymbol{y}_k \rangle \in \mathcal{B}} \frac{\partial \mathcal{L}\big(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{u}_k), \boldsymbol{y}_k\big)}{\partial \boldsymbol{\theta}}, \tag{4.15}$$

in whose direction the search space is traversed for a certain step size η, also called learning rate (LR), with

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial \mathcal{L}_{\mathcal{B}}}{\partial \boldsymbol{\theta}}. \tag{4.16}$$

An optimization scheme where updates would be conducted according to a single observation at time k is called stochastic gradient descent (SGD). Such stochastic updates are less accurate than with batch gradient descent (BGD), but usually still converge in practice, see Fig. 4.5. For the theoretical convergence properties of gradient descent among various other optimization methods, refer to [NW06].

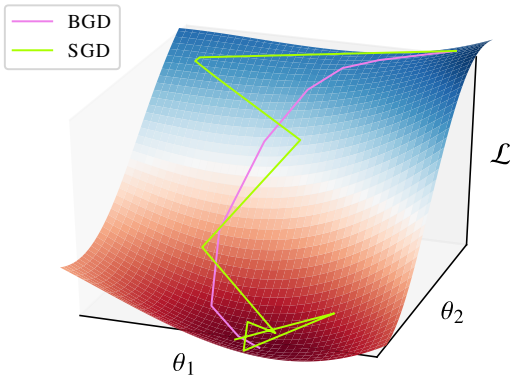


Figure 4.5: Exemplified stochastic vs. batch gradient descent for a cost function with two parameters (source: [Wal+]).

What makes SGD optimization on many parameters feasible nowadays compared to the past is the computation of derivatives not by pure numerical finite differences, but with automatic differentiation (AD) – also often algorithmic differentiation – instead [Bay+18]. It is both, precise and efficient, compared to pure numerical

differentiation, which is imprecise, and pure symbolic differentiation, which is inefficient. In the AD paradigm, the scalar cost function $\mathcal{L}$ is represented by a computational graph in which each node denotes an elementary algorithmic operation together with its derivative function. Linking all nodes to the cost function according to how the parameters $\boldsymbol{\theta}$ are engaged with the function arguments $\boldsymbol{u}$ provides two ways of traversing this graph: forward passes yield the operations' result (cost), and a backward (reverse mode) or another forward pass (forward mode) will yield the gradient through repeated application of the chain rule, that is, propagating derivatives along the graph links. The two modes denote different efficiencies according to the problem setup. The forward mode is efficient when few independent variables are transformed into many dependent variables, where one forward pass per independent variable is necessary only. For the reverse mode it is vice-versa – one backward pass per output variable suffices for the complete (numeric) gradient of each output variable with respect to all model parameters. Since ML often deals with a scalar cost's derivative with respect to many model coefficients, reverse mode is virtually always implemented and used in programming frameworks for ML and especially deep learning. Reverse mode AD in the context of ANNs is also known as error backpropagation. For example, in the trivial scalar case, let $y = f_3(f_2(f_1(u))) = f_3(f_2(h_1)) = f_3(h_2) = h_3$, then the chain rule is computed in reverse mode according to

$$\frac{\mathrm{d}y}{\mathrm{d}u} = \left(\left(\frac{\mathrm{d}y}{\mathrm{d}h_2}\right) \cdot \frac{\mathrm{d}h_2}{\mathrm{d}h_1}\right) \cdot \frac{\mathrm{d}h_1}{\mathrm{d}u}. \tag{4.17}$$

AD is symbolic in so far that an AD framework would have derivative primitives implemented for any mathematical operation that can occur in a program. AD is numeric in so far that intermediate values are stored in the computational graph during the first forward pass, that are reused during the backward pass when the gradient of the scalar loss with respect to the model parameters is computed. AD is rigorously defined not only for scalars but also for multi-dimensional tensors. The computational demand increases linearly with each further output variable (reverse mode) or input variable (forward mode). However, in case of (transposed) Jacobian-vector products, the computational demand can be reduced again to constant complexity by seeding the computational graph with the vector. For more details to AD, see [Bay+18].

4.3.3 Other models from the ML domain

Next to OLS and ANNs, other algorithms had their peak time in the ML community, too. Although they slowly fade into the background, and are less dealt with in publications nowadays due to the prevalent popularity of ANN-based approaches, some of them will be evaluated in this work for the sake of comprehensiveness. Alternative black-box models are, thus, briefly introduced in the following. Note that lazy algorithms, such as k-nearest neighbors, are deliberately ignored as these have to store most observations in memory for any prediction and are, therefore, hardly real-time capable or feasible for embedded systems.

The ϵ-support vector regression (SVR) algorithm is the support vector machine (SVM) version for regression tasks. It constructs a regularized linear model with coefficients $\boldsymbol{\beta}$ that are applied on higher-dimensional feature sets as transformed through a kernel function $\boldsymbol{f}_{\text{kernel}}(\boldsymbol{u}, \boldsymbol{u'}) = \langle \boldsymbol{g}(\boldsymbol{u}), \boldsymbol{g}(\boldsymbol{u'}) \rangle$ [HTF09]. More specifically, the linear cost function is ϵ-insensitive, i.e., cost is accumulated only if the prediction error exceeds a threshold ϵ. Those observations with errors beyond ϵ are the support vectors. Support vectors deviate, therefore, from the ϵ-band and are penalized through non-negative slack-variables $\boldsymbol{\xi}$ and $\boldsymbol{\xi'}$ (depending on the error sign) in the minimization formulation:

$$\arg\min_{\boldsymbol{\beta}} \left(\frac{\boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{\beta}}{2} + \lambda_C \sum_{k=0}^{K-1} (\xi_k + \xi'_k) \right), \tag{4.18}$$

$$\begin{aligned} \text{s.t.} \quad & y_k - \hat{y}_k \leq \epsilon + \xi'_k, \\ & \hat{y}_k - y_k \leq \epsilon + \xi_k, \\ & \xi_k, \xi'_k \geq 0 \quad \forall \quad k \in \{0 \,..\, K-1\}, \end{aligned}$$

where λ_C balances coefficient regularization and deviation from the ϵ-band. Solving the dual problem (which is omitted for brevity but yields memory-efficiency benefits for abstract kernels by resorting to vector-vector products) gives the following approximate solution:

$$\hat{\boldsymbol{y}}_k = \sum_{i=1}^{M} (\alpha_i - \alpha'_i) \boldsymbol{f}_{\text{kernel}}(\boldsymbol{u}_i, \boldsymbol{u}_k) \quad \text{s.t.} \quad 0 \leq \alpha'_i \leq \lambda_C, 0 \leq \alpha_i \leq \lambda_C,$$

with $M < K$ and M being the number of support vectors, and $(\alpha_i - \alpha'_i)$ denoting the weight for support vector i.

A completely different approach to modeling is denoted by decision trees [Myl+04]. These represent hierarchically applied if-else decisions, where the conditionals are the trainable coefficients, also called nodes. Nodes at the end of a decision sequence are denoted leafs, which are not conditionals anymore but actual estimates, while the rest are branch nodes. A decision tree effectively partitions the input feature space into subspaces representing the model estimate. Training a decision tree (growing it) encompasses the maximization of a score, usually the information gain or the Gini index, for a certain additional partition (node) within a certain regressor.

Since decision trees tend to grow overly complex and overfit quickly, ensembling methods are commonly used instead. Two of such methods are denoted by random forests (RFs) [Bre01] and extremely randomized trees (ETs) [GEW06]. Here, an ensemble of decision trees is fit on random subsets of the given observations with replacement (bootstrapping) and random subsets of features. Both methods lead to diversely grown trees that are partially decorrelated from each other, such that averaging across them reduces variance significantly. ETs differ from RFs in so far that, during training, ETs draw random thresholds in each feature out of the considered set in order to determine the next split, whereas RFs search for the most discriminative thresholds.

Albeit ETs and RFs show superior accuracy compared to decision trees, their performance is still inferior with respect to gradient boosting machines (GBMs), the third ensemble method for decision trees, among other potential *base learners*. In fact, during ML competitions that provide tabular data sets, i.e., static systems/environments to be identified excluding images, time series, and audio data, GBMs are the most popular modeling algorithm due to their unprecedented accuracy [Nie16]. In contrast to ETs and RFs, base learners are not grown in parallel but rather sequentially: Each subsequent tree $\psi : \mathbb{R}^P \rightarrow \mathbb{R}$ is fit to reduce the additive error of the previous ensemble of base learners $\mathcal{L}(y, \hat{y}_{m-1})$. Thus, by adding the m-th base learner the loss $\mathcal{L}(y, \hat{y}_m)$ is to be minimized where

$$\hat{y}_m = \hat{y}_{m-1} + \psi_m(\boldsymbol{U}) = \sum_{i=1}^{m} \psi_i(\boldsymbol{U}). \tag{4.19}$$

During boosting, base learners are kept simple, i.e., they are not grown out to large complexity, and the optimization does not evaluate a multidimensional parameter search space but rather an additive function space. The objective $\mathcal{L}(y, \hat{y}_m)$ at the m-th boosting round can be Taylor-expanded and truncated after the second-order derivative

$$\mathcal{L}(y, \hat{y}_m) = \mathcal{L}(y, \hat{y}_{m-1} + \psi(\boldsymbol{U})) \approx \mathcal{L}(y, \hat{y}_{m-1}) + (\tau')\psi_m(\boldsymbol{U}) + \frac{1}{2}(\tau'')\psi_m^2(\boldsymbol{U}), \tag{4.20}$$

with $\tau' = \partial_{\hat{y}_{m-1}} \mathcal{L}(y, \hat{y}_{m-1})$ and $\tau'' = \partial^2_{\hat{y}_{m-1}} \mathcal{L}(y, \hat{y}_{m-1})$. Ignoring the constant first term yields a simplified

objective

$$\tilde{\mathcal{L}}_m = (\tau')\psi_m(\boldsymbol{U}) + \frac{1}{2}(\tau'')\psi_m{}^2(\boldsymbol{U}) = \sum_{k=0}^{K-1}[\tau'_k\psi_m(\boldsymbol{u}_k) + \frac{1}{2}\tau''_k\psi^2_m(\boldsymbol{u}_k)]. \tag{4.21}$$

Further regrouping this loss with respect to each leaf node and introducing leaf scores will yield a sum of single-variabled quadratic functions [CG16], from which optimal splits can be read off directly for any given tree structure. However, since there are infinitely many possible tree structures, a greedy search for the optimal structure is usually conducted. As mentioned, the search is commonly based on the information gain or Gini index, but applied on the leaf scores that consist of the first and second-order derivatives τ' and τ''. Having found a good candidate, its estimate will contribute additively to the ensemble as it becomes part of it. The next boosting round would begin until user-defined termination.

The tree-based GBM approach became renowned by fast, scalable, cache-aware, and sparsity-aware SW implementations such as XGBoost [CG16] or Microsoft's LightGBM [Ke+17]. As is typical in ML, regularization is also part of the optimization (tree structure constraints, weight decay, random sub-sampling, adaptive learning rates, etc.) but was omitted here for conciseness.

4.4 Neural networks in state-space representations

Early on, disadvantages of ANNs' black-box character, e.g., lack of interpretability, has led to research efforts that would attempt a combination of ANNs with consolidated knowledge from classical physical theory. A selection of older and newer approaches into this paradigm are illuminated in the following. Newer methods, such as physics-informed NNs or neural ODEs seem to gain more momentum than their predecessors, which could be attributed to the emergence of AD and the improved performance of contemporary computing equipment. Common for all methods is the embedding of ANNs into state-space formulations. A comprehensive review on the combination of black-box neural networks with physics-driven or knowledge-based constraints is given in [Far+23].

4.4.1 State-space and structured neural networks

The state-space neural networks (SSNNs) are among the first attempts [RP96] to mount state-space representations on ANNs in order to approximate nonlinear dynamical systems according to the difference equation

$$\begin{aligned} \hat{\boldsymbol{x}}[k+1] &= \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}, \boldsymbol{u}, k), \\ \hat{\boldsymbol{y}}[k] &= \boldsymbol{f}_{\boldsymbol{w}}(\hat{\boldsymbol{x}}, \boldsymbol{u}, k), \end{aligned} \tag{4.22}$$

where $\boldsymbol{f}_{\boldsymbol{\theta}}$ and $\boldsymbol{f}_{\boldsymbol{w}}$ are nonlinear functions represented by potentially the same ANN with parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$ at discrete time k. Most contributions deriving from SSNNs employ some form of RNNs according to (4.11) for the nonlinear function approximations [ZV98; AW08]. Except for the state-space form in (4.22), no further algorithmic structure is incorporated, and accuracy is relied upon the general approximation theorem of ANNs completely [HSW89]. SSNNs can be seen as a special case of the more general structured NNs [FFG88]. However, the similarity of structured NNs to ANNs, especially as how they are declared nowadays, is sometimes so high that they become not discriminable. A structured NN can, e.g., resort to mere matrix multiplications with constant entries and gradient descent for adaptive control [WW01]. Other instances introduce network sparsity for more interpretation at the expense of accuracy [SR95]. In more recent publications, the term *structured NN* is also used as the superordinate concept for combining physical ODEs with ANNs. This would

encompass the concepts in the next sections as well, e.g., as in control of AC machines [Gar+07], lithium-ion battery state of health estimation [And+13], or for predicting contact dynamics [Hoc+21].

4.4.2 Physics-informed neural networks

The physics-informed neural networks (PINNs) were introduced in [RPK19] and are defined by appending additional loss terms to the total loss function additively, which penalizes the deviation of N_F state estimates from the ODE/PDE and from N_B initial/boundary conditions within the contemplated system

$$\begin{aligned}\mathcal{L}_\mathrm{total} &= \lambda_\mathcal{D}\mathcal{L}_\mathcal{D} + \lambda_{C_\mathrm{B}}\mathcal{L}_{C_\mathrm{B}} + \lambda_{C_\mathrm{F}}\mathcal{L}_{C_\mathrm{F}}, \\ &= \frac{\lambda_\mathcal{D}}{K}\sum_{k=0}^{K-1}\big(f_{\boldsymbol{\theta}}(\boldsymbol{u}_k) - y_k\big)^2 + \frac{\lambda_{C_\mathrm{B}}}{N_\mathrm{B}}\sum_{n=0}^{N_\mathrm{B}-1} g_\mathrm{B}\big(f_{\boldsymbol{\theta}}(\boldsymbol{u}_n)\big)^2 + \frac{\lambda_{C_\mathrm{F}}}{N_\mathrm{F}}\sum_{i=0}^{N_\mathrm{B}-1} g_\mathrm{F}\big(f_{\boldsymbol{\theta}}(\boldsymbol{u}_n)\big)^2,\end{aligned} \tag{4.23}$$

with g being a distance metric for the model's estimate at the collocation points on the boundary or within the investigated system, and λ representing weighting of the loss terms. Other than the loss function, training through error backpropagation remains as usual. It becomes obvious that PINNs could also be used as white-box models when the data-set-related loss term $\mathcal{L}_\mathcal{D}$ is eliminated, i.e., as a faster, mesh-free, and approximating alternative to FEA or CFD. In fact, PINNs are often facilitated for solving PDEs, fractional PDEs, integro-differential equations, and stochastic differential equations, and enjoy growing popularity [Cuo+22].

Nonetheless, since a PINN essentially represents a black-box ANN after training, several disadvantages are inherited from the black-box ML domain:

- no physically interpretable states or parameters in the model,
- initial estimates cannot be set when stateless MLPs are employed,
- model estimates can actually deviate from an ODE or PDE due to the soft contribution to the loss.

In addition, due to the multi-criteria cost function, the question for finding a constructive weighting among the loss terms ensues, which is of course application-specific. During the course of training a PINN, it cannot be ruled out that tradeoffs between estimation accuracy and physical correctness could hamper successful system identification.

4.4.3 Neural ordinary differential equations

Another interesting movement in gray-box modeling with ANNs is constituted by neural ordinary differential equations (NODEs) [Che+19]. Similar to SSNNs, the state-space representation of the system is at the core, yet in its continuous, differential form with time t:

$$\frac{\mathrm{d}\hat{\boldsymbol{x}}}{\mathrm{d}t} = \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}, \boldsymbol{u}, t) \quad \text{and} \quad \hat{\boldsymbol{x}}(t=0) = \boldsymbol{x}(t=0) = \boldsymbol{x}_0. \tag{4.24}$$

Since this model is to be computed on real hardware, a discretization of (4.24) is inevitable. However, the inventors of the NODE leave the choice of discretization, or, equivalently, the choice of the solver, to the user or engineer. The inner workings of an arbitrary solver are taken out of the equation by only working on the solution trajectory, independent how it was obtained. This is made possible by computing gradients with the adjoint sensitivity method [Pon87], which retrieves gradients by solving an additional augmented

ODE backwards in time (with another arbitrary ODE solver), instead of with classic error backpropagation. Following this, the loss is defined for a solution $\hat{\boldsymbol{x}}(t_1)$ at time t_1 that starts from t_0 as

$$\mathcal{L}\big(\boldsymbol{x}(t_1), \hat{\boldsymbol{x}}(t_1)\big) = \mathcal{L}\left(\boldsymbol{x}(t_1), \hat{\boldsymbol{x}}(t_0) + \int_{t_0}^{t_1} \boldsymbol{f}_{\boldsymbol{\theta}}\big(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)\big)\,\mathrm{d}t\right) \tag{4.25}$$

In order to compute the desired gradient $\partial\mathcal{L}/\partial\boldsymbol{\theta}$, first the so-called adjoint $\boldsymbol{a}(t) = \partial\mathcal{L}/\partial\hat{\boldsymbol{x}}(t)$ is to be calculated, whose dynamics are given by

$$\frac{\mathrm{d}\boldsymbol{a}}{\mathrm{d}t} = -\boldsymbol{a}(t)^{\mathsf{T}} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t))}{\partial \hat{\boldsymbol{x}}}. \tag{4.26}$$

This ODE can be solved backwards in time by an ODE solver that solves the terminal value problem (4.26) with $\boldsymbol{a}(t_1)$, and at the same time, the required trajectory $\hat{\boldsymbol{x}}(t)$ will be recomputed together with the terminal value problem starting from its terminal value $\hat{\boldsymbol{x}}(t_1)$. Eventually, a third integral yields the desired gradient with

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}} = \int_{t_1}^{t_0} \boldsymbol{a}(t)^{\mathsf{T}} \frac{\partial \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t))}{\partial \boldsymbol{\theta}} \mathrm{d}t. \tag{4.27}$$

While the vector-Jacobian products in (4.26) and (4.27) can be efficiently computed with AD, all integrals that solve for $\hat{\boldsymbol{x}}$, $\boldsymbol{a}$, and $\partial\mathcal{L}/\partial\boldsymbol{\theta}$ can be obtained by a single call to an ODE solver with a concatenated state vector [Che+19]. The adjoint sensitivity method is not just one atomic tool but can be carried out in different configurations (numerical vs. automatic differentiation, forward- vs. reverse-mode, continuous vs discrete, etc.) with implications for the performance and scaling efficiency depending on the problem at hand [Ma+21]. An extension to NODEs is represented by universal ordinary differential equations, which is claimed to encompass not only (4.24) with $\boldsymbol{f}_{\boldsymbol{\theta}}$ being a single ANN, but also all other functionals with more application-specific domain knowledge embedded [Rac+21].

5 Data set description and design of experiments

In contrast to a series vehicle drive train, a test bench is required to provide additional measurement equipment and sensors in order to track system states that are not measurable otherwise. Among other states, useful information, in the context of a thermal modeling task for electric motors, are

- currents in d/q coordinates,
- motor speed,
- ambient and coolant temperature,
- coolant flow rate,
- DC-link voltage,
- miscellaneous control configurations, such as pulse patterns,
- several temperature sensors across the stator,
- several temperature sensors across the rotor.

Usually the motor component temperatures denote the dependent variables that are to be estimated, whereas remaining information act as independent variables. For those independent features one has to be careful what to select as actual input feature to a thermal model, however. As a general rule, only sensor information that is readily available through the set of ECUs in the target vehicle should be used as thermal model input, in order to avoid the necessity of additional hardware in the series production just for the thermal model.

Among those states that are usually not available, the motor torque can be invoked as an interesting state to track for operation analysis, since, together with the motor speed, it reveals the mechanic power output. Yet, it is usually obtainable through sophisticated measurement equipment only, that is not available in series applications, and, thus, not an input feature candidate that can be expected to exist on the field.

Moreover, one often can find precise power analyzers and power meters next to electric drive test benches for accurate measurement of losses within motors and inverters that form the drive train. In the context of system identification, this equipment is commonly used for the design of classic lumped-parameter thermal network (LPTN)-based thermal models (although not mandatory), where power measures for different operation points are stored in LUTs for the series application. While this information is also interesting for operation analysis, in this work the power information is deliberately ignored, and rather to be inferred from the remaining sensory records by ML-based approaches.

Another noteworthy temperature state is measured by the so-called NTC thermistor at a certain stator position, that would be found in some automotive motors as an additional support temperature sensor next to potential ambient and coolant temperature sensors. This sensor is usually more precise than thermocouples which would be deployed in prototypical motors at test benches for measuring additional component temperatures. Due to this sensor's proximity to the target variables, it renders a very useful in-series-available state for thermal modeling. However, functional safety requirements become increasingly strict in the automotive domain, and due to, e.g., slow sensor degradation throughout a vehicle's lifetime, it can happen during the requirement specification phase that this sensor is actually to be estimated itself rather than used as input feature.

Apart from – but still in view of – the sensor arrangement, the appropriate sample time should be discussed. While high sampling frequencies around 20 kHz are common for measuring electrical quantities in electrical drives for control tasks, the temperature estimation task can and should be executed substantially slower. A

rule of thumb for the sample time T_s for a LTI system with its smallest system time constant $T_{x,\min}$ is given by [IM11] as

$$\frac{T_s}{T_{x,\min}} = \frac{1}{5} \dots \frac{3}{5}. \tag{5.1}$$

However, as was highlighted in [Wal17], an electric motor is not only nonlinear and time-variant, but also its components' thermal time constants vary significantly: the stator windings time constant would commonly be around 2 min, while the permanent magnets' vary dynamically with the motor speed between 4 . . . 25 min. This spread renders the system of ODEs that describe the heat transfer between components a stiff system. In order to trade off too long sample times in which a derating algorithm would miss overheating protection, and too short sample times, in which electrical operation points could be captured sufficiently but numerical and computational demand would soar into infeasibility, a sample time of roughly 100 ms . . . 2000 ms is recommended (e.g., $T_s = 1$ s as in [Wal17]). Larger sample times would ease strain on embedded hardware but might miss out characteristics required for high dynamic driving, for which shorter sample times are to be used together with a more premium control platform.

5.1 Relevant data sets

In general, measurement data from an electric drive train – be it mounted on a test bench or inside an actually operating vehicle – are subject to strict non-disclosure agreements (NDAs), if the investigation is funded by industrial partners of the research institution or R & D department. Drive test benches are expensive – typical setups with contemporary measurement equipment and sufficient power electronics (which can become obsolete within a decade due to the high innovation rate of further developments in the electric drive domain) cost up to millions of Euros. This leads to the fact that many test benches are located within industry sites managed by private companies, that have no interest in publishing their work in the first place. Then, publicly available data from investigations on off-the-shelf motors equipped with an array of thermocouples do not exist yet, as the embedding of these temperature sensors usually requires the motor to be assembled along with them. An a posteriori embedding is not feasible from a construction point of view.

A consequence of this fact is that there are almost no publicly accessible data sets from electric drives, neither from a test bench nor an on-road vehicle, especially with thermal information of the motor. Today, the sole exception to this is represented by a 185 h measurement data set[1] uploaded on `Kaggle.com` by the author of this work [KWB21b]. It was, therefore, used in several publications dealing with thermal modeling, especially with tools from the ML domain. The data set stems from the author's affiliation's lab, where a test bench with a prototypical 52 kW PMSM was deployed for thermal modeling purposes, see Sec. A.1. This data set (as of here labeled $\mathcal{A}$) denotes the main investigative subject of this thesis, due to its transparency and unbureaucratic accessibility, which is meant to ensure reproducibility and foster research around thermal modeling of electric motors. It also represents the largest data set among those that are available to the author (see an overview in Tab. 5.1), with the most comprehensive operating area coverage, as can be seen in Fig. 5.1. Note that the amount of sensors in Tab. 5.1 might correspond to an already reduced or aggregated set from an originally larger array, due to the signal cleaning process or sensor defects. Moreover, data set $\mathcal{A}$ yields the cleanest temperature signals, especially for the rotor component, with a high signal-to-noise ratio, which is problematic for some of the other data sets. Thus, any data set other than $\mathcal{A}$ will not be further discussed in this thesis, except for minor comparative illustrations.

A selection of three measurement profiles from data set $\mathcal{A}$ is illustrated in Fig. 5.2. It shows a profile with constant, piece-wise constant, and random intermittent excitation, which is representative for the whole data set. It also becomes evident that the ambient and coolant temperature slightly suffer from measurement artifacts

[1] Initially, the data set was also subject to an NDA, but it could be repealed in accordance with the OEM due to the motor's prototypical design, which was not carried over to a series production.

Data set	Size in hours	Sensors		Access	Application
		Stator	Rotor		
$\mathcal{A}$	184.8	3	1	Public	Automotive
$\mathcal{B}$	32.4	12	1	Undisclosed	Automotive
$\mathcal{C}$	52.7	28	7	Undisclosed	Automotive
$\mathcal{D}$	54.3	2	1	Undisclosed	Automotive
$\mathcal{E}$	133.4	4	1	Undisclosed	Industrial

Table 5.1: Properties of some relevant data sets. Note that only data set $\mathcal{A}$ will be used throughout this work.

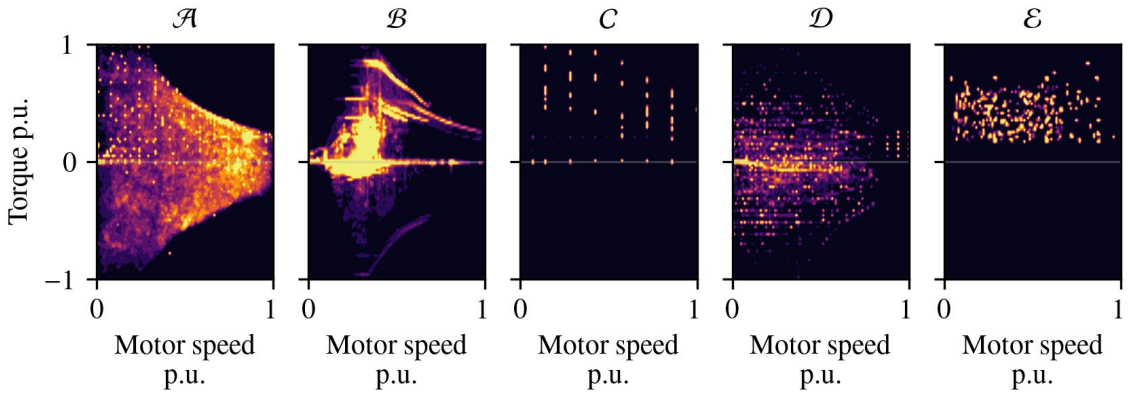


Figure 5.1: Heat map for operation point frequency in each data set. Brighter areas were visited more often. The operating area was charted in a 100×100 grid and the sample frequency clipped at 200 samples per bin as upper bound.

that present themselves in short signal spikes. These can be filtered and smoothed out in a preprocessing phase prior to modeling. However, such amendments should be discussed and it is to be surveyed whether these artifacts stem from poor sensor quality at the test bench or whether this is the expected series quality. In case of the latter, it might be worthwhile to keep these artifacts at least for the test set during cross validation in order to get a grasp on the real on-road performance.

Data set $\mathcal{A}$ was sampled at 2 Hz, which amounts to 1330815 multi-dimensional samples. For all following modeling strategies, the input and target sensor set is compiled in Tab. 5.2. Note that torque is deliberately omitted because of its rare availability in series production vehicles. The geometrical component arrangement of the PMSM is illustrated in Fig. 5.3. There might be more than one sensor actually embedded per component than becomes evident from Tab. 5.1 and Tab. 5.2, which were aggregated for cleaner signals (see Sec. A.1 for a comprehensive overview).

In view of the scarce availability of measurement data, the workaround of synthesizing data through FEA/CFD simulations suggests itself. Such data is expected to lack some parasitic effects, though, e.g., measurement noise and high-order physical effects, and the difference to the real hardware implementation would remain more unknown than with a test bench setup.

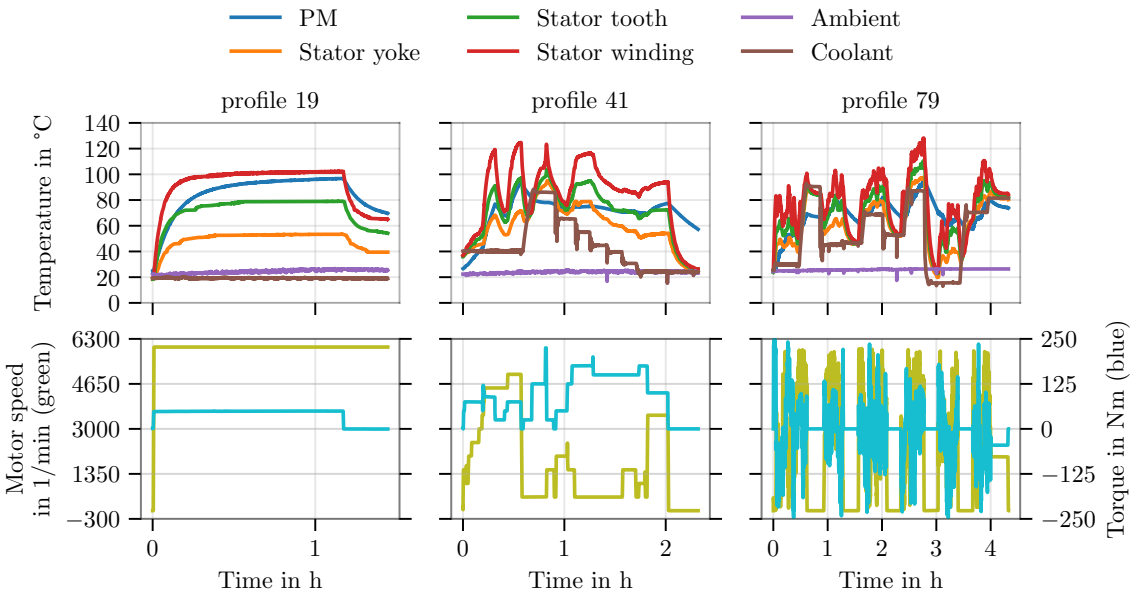


Figure 5.2: Three exemplary measurement profiles in data set $\mathcal{A}$. First row displays all temperatures while the second row shows the motor speed in green and torque in blue.

Measured input sensors		Measured target temperatures	
Parameter name	Symbol	Parameter name	Symbol
Ambient temperature	ϑ_{A}	Permanent magnet	ϑ_{PM}
Liquid coolant temperature	ϑ_{C}	Stator tooth	ϑ_{ST}
Actual voltage d/q-axes	$v_{\mathrm{d}}, v_{\mathrm{q}}$	Stator winding	ϑ_{SW}
Actual current d/q-axes	$i_{\mathrm{d}}, i_{\mathrm{q}}$	Stator yoke	ϑ_{SY}
Motor speed	n_{mech}		

Table 5.2: Measured input and target variables (data set $\mathcal{A}$)

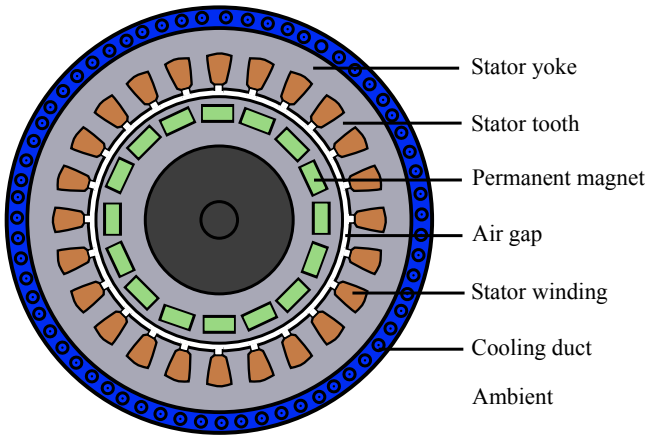


Figure 5.3: Motor cross-section (data set $\mathcal{A}$)

5.2 Excitation planning – Design of experiments

Regardless of whether the system states are measured or synthesized through simulation, the question of optimal system excitation arises. Usually, the time budget is limited, and one would strive for data sets that do not take long to record but inherit all important system operation points such that a data-driven thermal model will be sufficiently informed during the offline training phase.

It becomes obvious in Fig. 5.1 that excitation trajectories are not standardized in any means when it comes down to test bench measurements that are supposed to be used for thermal modeling with ML. In fact, the topic of excitation planning – or in even more general terms the topic of design of experiments (DoE) for nonlinear dynamic systems – has experienced very little attention in both industry and academia, especially when compared to the ever-increasing number of publications for new modeling techniques. This is odd the more so in face of the fact that ML models are known to be good at inter- but not extrapolating from data, as mentioned earlier, which makes the DoE an important aspect of the modeling process. The most appropriate model will not be able to accurately estimate high-dynamic load profiles if it was trained with mere constant excitation trajectories only.

Answering the question for optimal system excitation is difficult because not only the estimation performance and model robustness but also time and resources for data generation are involved. A multi-criteria optimization accrues with partially contrary objectives for which only little research exists yet. Essentially, requirements for an informative system excitation are only worked out for LTI systems by means of the Fisher information matrix [Meh74; PS12], whereas for nonlinear time-variant systems no general policies could be identified yet [SL19]. Noteworthy publications treating DoE for nonlinear systems are briefly outlined in the following.

In [HN16], common passive excitation signal forms for nonlinear system identification are compared with each other on their coverage of the system input signal domain. In particular, sinusoidal signals and those with jump discontinuities as well as piecewise-constant signals are examined [Bau+08; PS12]. Furthermore, a hybrid variant is elaborated, which takes the control input constraints of the considered system into account.

However, it follows not much later in [HN17; HN18] that passive excitation signals are inferior to actively adapted signals that incorporate process knowledge. Specifically, a method is presented that heuristically chooses operation points that are sequentially approached such that successive points have the highest distance according to the minimax distance to previous operation points – the local nearest neighbors. An iterative optimization considering the observed system response does not take place though. Moreover, the safety-critical system boundaries of the model that is to be identified are only partially taken into account, and the required experimental effort is not optimized systematically. Eventually, the methodology is only applicable to stable systems due to the neglected feedback of the system response.

It is another challenge to identify instable systems whose operation area cannot be explored safely without a stabilizing controller [GLS77; IM11]. Popular instances of instable systems are the inverse pendulum (Segway) [Gon+19] as well as some airplane and rocket applications [MGA14]. Albeit the thermal response of a PMSM does not fall into this category, it should be highlighted that optimal excitation is closely connected to the model identification process, the more so for some certain systems with no less relevance.

Automatic excitation profile generation in data set $\mathcal{A}$

The generation of data set $\mathcal{A}$, or more particular, the motor operation planning at the test bench employed another heuristic for its system excitation trajectories. A 2-D random walk in the speed-torque-plane denotes the basis for each load profile, in order to establish random excitation trajectories. These walks would need to adhere to the motor power limit, such that a free unconstrained walk had to be transformed accordingly. Moreover, similarity to real-world driving cycles was another objective for the random walk scheme, such that,

e.g., city drive cycles with many stops have to be simulated. In order to accommodate for all these factors, the following heuristic was elaborated:

1. Generate a unitless 2-D random walk of random length, that is, in each step sample randomly from $\mathcal{U} \in \{-1, 0, 1\}$ for both dimensions independently, and accumulate to the previous step's result.
2. Scale the resulting trajectory in the speed-torque-plane such that all four domain boundaries are touched upon.
3. Since significant portions of the trajectory might end up in an invalid operating range, mirror all those points above the power rate limit at one of two straight lines (the tangent line on the power limit boundary at maximum speed), whichever is closer.
4. Repeat last step if reflections are transformed into invalid regions again.
5. Optionally, add a random-length resting phase ($0\,\frac{1}{\mathrm{min}}$ and $0\,\mathrm{N\,m}$) and repeat at step 1. Concatenate all random walks and resting phases to a single profile.

The generation process can be seen for three exemplary trajectories (that are not part of data set $\mathcal{A}$) in two steps in Fig. 5.4. Here, profile 0 stems from a random walk of 10,000 steps, profile 1 from 100,000 steps, and profile 2 is stitched together from six random walks of 3000 steps each with resting phases of random length up to a few minutes in between. Adding resting phases contributes to real driving cycle similarity, and combining several random walks where each is scaled to just fit into the operation region increases the dynamic range of the profile. Accordingly, profile 0 shows a medium dynamic range, whereas profile 1 inherits the slowest dynamic and profile 2 the highest. Fig. 5.5 visualizes the same profiles in the time domain.

It becomes visible that by concatenating several random walks as is done in profile 2 and by mirroring the trajectories on a straight line, which is just a rough linear approximation of the hyperbolic operation boundary, several jumps in the speed-torque-plane are induced. Such sudden increases of the current rate are dampened at the test bench through control input rate constraints, thus, the references in Fig. 5.5 are to be understood as desired trajectories but are not ideally followed by the control strategy. What is more, the test bench requires a derating mechanism depending on the motor component temperatures, such that overheating is avoided through further reduction of the allowed power limit, which, again, affects the actually followed trajectories. A complete overview of the whole data set's operation trajectory in the speed-torque-plane can be inferred from Fig. A.5.

The presented excitation planning scheme is not actively adapted, which renders it a passive scheme. Although this is to be considered inferior, a look onto Fig. 5.1 reveals that this scheme shows a substantially more homogeneous coverage of the operating area (data set $\mathcal{A}$) than what was achieved for the other data sets. Consequently, the random walk strategy can be considered an improvement compared to standard excitation strategies from industry and academia. However, it remains a heuristic, and an optimal adaptive planning policy that also considers the coverage of the system response is to be preferred for future work.

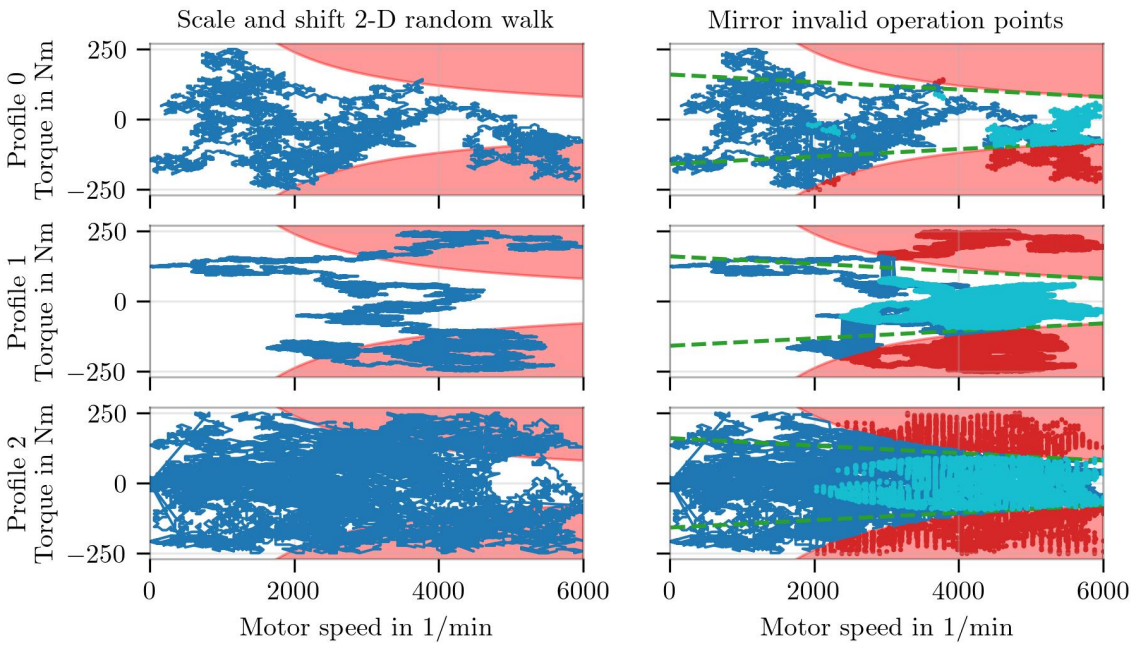


Figure 5.4: Three exemplary excitation profiles are illustrated in two steps of their generation process.

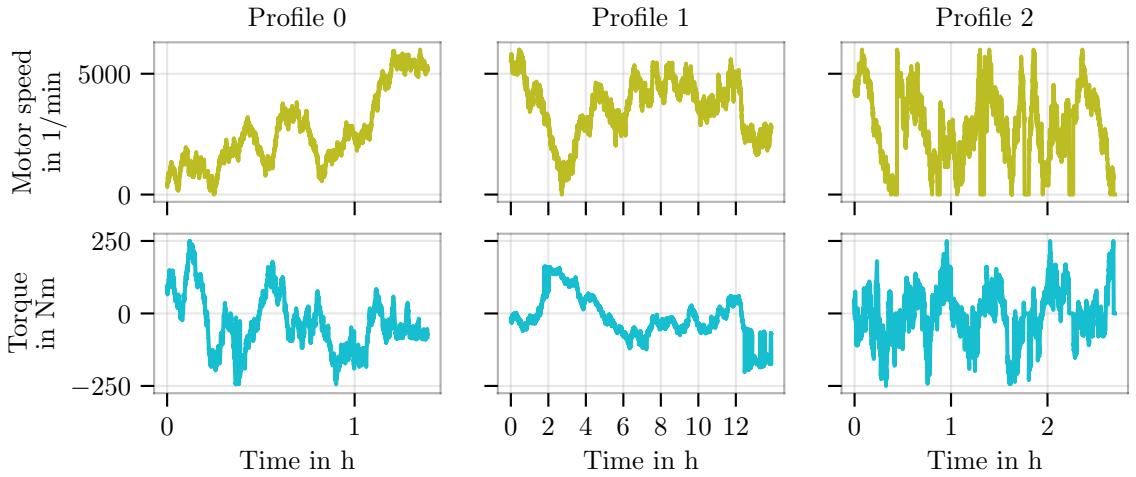


Figure 5.5: Three exemplary excitation profile trajectories

5.3 Means and metrics for measurement profile comparison

Having recorded a data set, the question arises in how far the measurement profiles resemble each other, and whether these profiles have covered the input (and output) domain sufficiently yet. Answers to such questions could give hints on how much more data should be recorded, how likely observations on-road are that lack in the recorded data set, and what a fair split of the data set into training and evaluation sets could be.

The similarity of signals can be simplified to a study of their probability distributions, or more particular, their histograms. Two popular methods for evaluating whether an empirical distribution function stems from a theoretical prior distribution function are the Kolmogorov-Smirnov test [Mas51] and the Kullback-Leibler divergence (KLD) [KL51]. The former subtracts the cumulative distribution function π_{cdf} of an empirical sample $\mathcal{E}$ from the assumed theorical prior histogram $\mathcal{F}$, and the supremum across all differences is the test measure $d_{\mathcal{E},\mathcal{F}}$:

$$d_{\mathcal{E},\mathcal{F}} = \sup_u |\pi_{\text{cdf},\mathcal{E}}(u) - \pi_{\text{cdf},\mathcal{F}}(u)|. \tag{5.2}$$

which has to be below a certain value in order to confirm the empirical distribution to stem from the prior under a given significance level. This procedure is generally easy to compute and not resource-hungry. The KLD between discrete distributions for an empirical random variable E and a prior F is defined as

$$d_{\text{KL}}(E,F) = \sum_{u\in\mathcal{U}} \pi_E(u) \cdot \log\left(\frac{\pi_E(u)}{\pi_F(u)}\right), \tag{5.3}$$

where $\mathcal{U}$ includes all realizations u of E and F, and $\pi(\cdot)$ is the probability density function. It is also called information divergence or relative entropy, and it indicates mutual information between two random variables. Note that the probability density functions of E and F would denote their empirical histograms for our purposes.

With the two mentioned tests, a selection of measurement profiles, that, e.g., denote a test set candidate, can then be tested against the superpositioned distribution of all remaining profiles in order to evaluate whether it would represent the training set appropriately. Conversely, during measurements, one can plan in advance whether a new measurement profile might be different enough from the so far collected recordings in order for it to be worth collecting.

Although this seems to be a procedure with many desirable properties, there are considerable drawbacks in practice. First, the tests require predefined thresholds to determine a distribution to be similar or not to a given prior distribution, which is data-set-specific and a degree of freedom. Second, in case of the search for train-test splits, the combinatorics of k possible subsets out of n available measurements profiles is $\binom{n}{k}$, which quickly becomes intractable for common amounts of measurement profiles despite the simple algorithms behind the Kolmogorov-Smirnov test and the KLD. Morever, reducing a signal to its histogram obviously eliminates the time-correlation information of the signal, which is equally relevant for similarity examinations. An alternative could be the use of unsupervised learning, which is often used for clustering.

In [Win19] – a research thesis supervised by this work's author – different measures to compare subsets of drive cycles are investigated. Especially different flavors of the dynamic time warping (DTW) method, specifically, the fast [SC07] and soft variant [CB17], and the minimum jump cost distance [SA12] as metrics that also consider the time series characteristic are examined. Methods that require equal-length time series, such as those based on the Minkowski-distance, or threshold-based measures, which are usually used for data set reduction, are systematically excluded [Lia05]. First, all profiles are pairwise compared according to a DTW flavor, and their distance is fed into an adjacency matrix, which is then used for spectral clustering [SM00] – an unsupervised learning algorithm that is particularly useful for non-convex clusters. Second,

four splits are generated, such that two splits would accommodate a balanced selection and two other an unbalanced. A balanced selection means to pick training and test set profiles in such a way that all clusters are represented in both the training and test set. Too small clusters of four or fewer profiles are ignored entirely. An unbalanced selection allows for training sets that include clusters that are not represented in the test set and vice versa. Third, each of the four splits are used for training and testing an array of (static) ML models. The idea behind this is that for balanced sets a ML model would not be forced to do an out-of-distribution estimate in the test set, and the test set performance would be better than with an unbalanced set. If this observation can be made for a majority of the ML algorithms for a certain time series distance measure, then it is appropriate to use for measurement profile categorization. Eventually, the result was shown to be that a 2-D KLD over $P_{\mathrm{el}} = v_{\mathrm{d}} i_{\mathrm{d}} + v_{\mathrm{q}} i_{\mathrm{q}}$ and $\dot{P}_{\mathrm{el}}$, and the DTW method were the most effective approaches with the most striking difference between balanced and unbalanced assignments from spectral clusters, leaving methods behind that are based on aggregated features. The remarkable finding here is that static model performance could be improved by strategic training and test splits according to metrics that do not consider the system response, i.e., the component temperatures. It remains open, however, whether the resulting test set is also representative for on-road observations, which can be also attributed to the question whether a data set as a whole is representative for on-road records to begin with. Connected to this open point, it is still not solidly found whether static ML temperature model performance on a held-out test set is the right objective when searching for data set splits since this might facilitate overfitting.

The DTW plus spectral clustering approach belongs to the category of time series similarity measures that act directly on the sequences in time. In contrast, a different class of methods work on aggregated features that are engineered from the original sequence of data. A possible way of choosing clusters on such aggregations is by the k-means method, which will find k clusters by positioning k centroid points in the multidimensional parameter space, such that no nearest neighbor to one of the centroids is falsely assigned to another. The algorithm starts by placing the centroids randomly in the data set space, then assigns the nearest neighbors to the centroid's cluster, moves the centroids to the center of the assigned group, and then repeats the process until convergence. A downside of this method is that it only works well for rather convex clusters in whatever space or coordinate system they are projected.

In Fig. 5.6, the entire data set is aggregated, transformed und projected onto a two-dimensional coordinate system. The transformation steps up to this data set visualization include the following:

1. Two additional features are concatenated to the original set for each profile: The minimum temperature subtracted from the maximum temperature across the target components

$$\vartheta_{\mathrm{dispersion}}[k] = \max_{q}(\vartheta[k]) - \min_{q}(\vartheta[k]), \tag{5.4}$$

   and the current amplitude

$$i_{\mathrm{s}}[k] = \sqrt{(i_{\mathrm{d}}[k])^2 + (i_{\mathrm{q}}[k])^2}. \tag{5.5}$$

2. Each feature is normalized through division by the physical sensor's maximum absolute reading in the data set, except for temperatures, which are divided by 100 °C.
3. Each of the 69 profiles is aggregated into a set of 59 scalar values, which represent a new descriptive vector. Specifically, the minimum, maximum, standard deviation, and mean for each feature, as well as the fraction of time no torque was applied relative to the profile duration, the sum of the exponentially weighted moving standard deviation (see Sec. 6.1.1) with a span of 700 samples of both the torque and the motor speed, and, eventually, the relative size of the profile compared to the whole data set size, denote the new representation of each profile. Each of these vectors consists of 59 elements accordingly.
4. All 69 descriptive vectors are composed into a new data set $\mathcal{D}_{\mathrm{agg}} \in \mathbb{R}^{69\times 59}$.
5. This new data set is normalized such that the population mean equals zero, and the population standard deviation equals one (see Sec. 6.1.2 for an overview of normalization schemes).

6. $\mathcal{D}_{\text{agg}}$ is transformed into a new coordinate system that is spanned by the two eigenvectors that correspond to the two biggest eigenvalues of $(\mathcal{D}_{\text{agg}})^{\mathsf{T}}\mathcal{D}_{\text{agg}}$. These are also called principal components, and the coordinate transform is also known as principal component analysis (PCA) [Pea01; Hot33].

Following this chain of transformations, one obtains a visually accessible representation of the data set, such that profiles can be distinguished from one another by similarity. What is more, the newly found coordinate system exhibits a desirable property, in which profiles, that were categorized into excitation classes visually by the author, are clustered according to their assigned category. The different excitation classes are: *constant* for mainly constant excitation trajectories, *random* for the random walk excitation, and *mixed* for excitation trajectories that are characterized by many different operating points that are hold on for shorter amount of times than for those in the constant category but were not stemming from a random walk. It is worth noting, however, that slight alterations to the transformation chain can lead to significantly different clusters and visual representations. The two visualized principal components explain over 50 % of the total variance with the first explaining twice as much as the second, see Tab. 5.3. For the sake of visual ease this ratio shall be sufficient. However, it should be also acknowledged that some significant room between neighboring profiles might not be rendered, and rating profile similarity solely from the first two principal components could be erroneous. Employing clustering methods can give further insight.

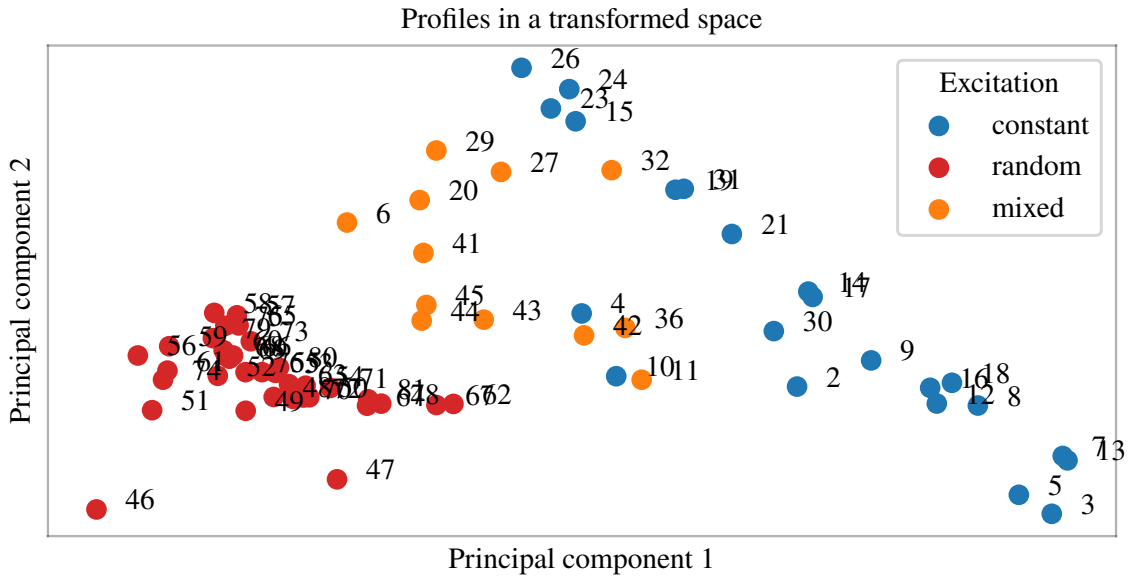


Figure 5.6: All profiles aggregated by several statistical moments and additional characteristics and projected onto the first two principal components of the resulting data matrix, which is normalized and bias-free

Principal component	1	2	3	4	5	6	7	8	9	10
Explained variance ratio (%)	35.6	16.9	12.1	7.0	6.0	3.4	3.1	2.9	1.9	1.7

Table 5.3: Explained variance ratios of the PCA

In Fig. 5.7, first row, the result of a k-means clustering is shown for varying k. In the second row, the same coordinate system is used but with profiles being color-encoded according to the spectral clustering scheme on an adjacency matrix filled with pairwise DTW distances considering all sensor signals $[\boldsymbol{U} \quad \boldsymbol{Y}]$. It becomes evident that the spectral clustering approach on DTW features outputs a significantly different label correspondence to differently many clusters than the k-means method on aggregated features. Note that although the k-means clusters seem to align the data visually better, it does not mean that the actual

profile similarity is better represented than with the spectral clustering approach, which does not build on the aggregated features that determine the view, after all.

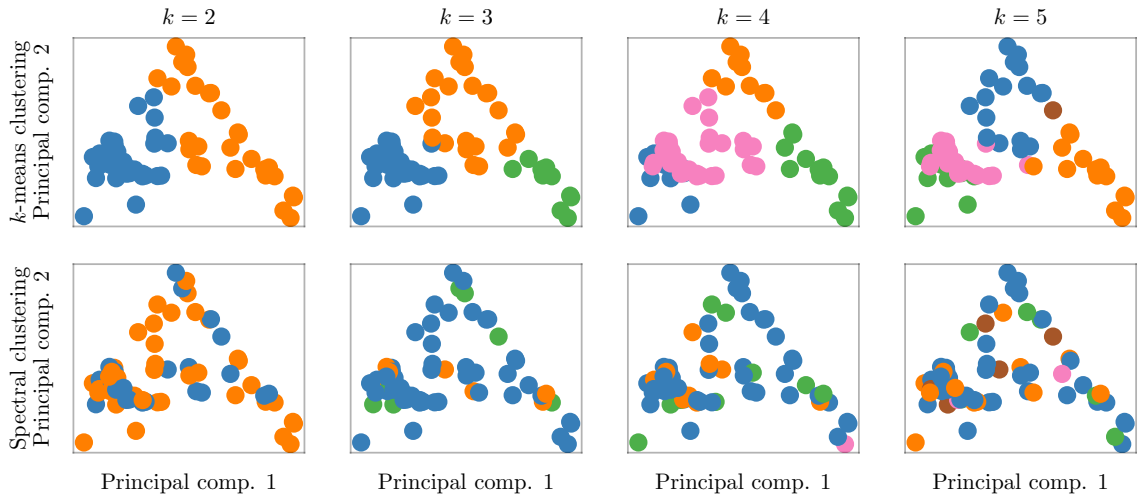


Figure 5.7: First row: k-means clustering for varying k for an aggregated data set featuring all profiles. Second row: Spectral clustering for varying k according to an affinity matrix consisting of the pairwise DTW distances. The view is projected onto the first two principal components of a PCA on $\mathcal{D}_{\text{agg}}$.

It is now the free choice of the engineer to use the information from these clustering methods to determine a) how to pick training and test sets for a fair split, and b) to plan the system excitation on the test bench if the measurement is ongoing. See Sec. 6.2 for a description of how the split is determined for this thesis. Future research will focus on finding systematic policies that use these clustering techniques also for excitation planning in nonlinear dynamic systems.

6 Black-box temperature estimation with machine learning

In this chapter, the temperature estimation task is solved with pure black-box machine learning (ML) models without incorporation of any physics-based knowledge in form of heat transfer principles. The following investigations are based on the author's publications on this task, more specifically, solutions by linear regression [KWB19b], by several static ML models [KWB21a], and by residual recurrent and convolutional artificial neural networks (ANNs) [WKB17; KWB19a; KWB20]. Some passages in this chapter may be copied from these works, yet all experiments are genuine and newly conducted for this dissertation for the sake of consistency. Following on or derived from the author's previous works are contributions that present similar attempts by means of NARX-structure extensions [LH20], AdaBoost as gradient boosting machine (GBM) variant and linear regression with regularization variants [CGK21], decision tree derivations [Anu20], and attention-based encoder-decoder long short-term memorys (LSTMs) [LA22], or also by means of the same topology paradigms [HSA22; Sar+22].

A black-box model is characterized by its abstract structure that serves the general-purpose task of fitting (non)linear maps from one set of random (independent) variables onto another (dependent) set. Formally, in this work, it means an estimator function $\boldsymbol{f_\theta} : \boldsymbol{u} \rightarrow \hat{\boldsymbol{y}}$ is searched for, where

$$\begin{aligned} \boldsymbol{u} &= [v_\mathrm{d} \quad v_\mathrm{q} \quad i_\mathrm{d} \quad i_\mathrm{q} \quad n_\mathrm{mech} \quad \vartheta_\mathrm{A} \quad \vartheta_\mathrm{C}], \\ \boldsymbol{y} &= [\vartheta_\mathrm{PM} \quad \vartheta_\mathrm{ST} \quad \vartheta_\mathrm{SW} \quad \vartheta_\mathrm{SY}]. \end{aligned} \tag{6.1}$$

The dependence on time step k is omitted for readability. Furthermore, most statistical models require some sort of normalization of the input and output data, in order to credit each feature's contribution to the overall map fairly. Moreover, it is often helpful to enrich the input feature vector $\boldsymbol{u}$ with further engineered features into $\boldsymbol{u}_\mathrm{ext}$, when in lack of additional sensors, in order to reveal patterns that are easier to capture by a model's structure. See Fig. 6.1 for a sketch of a block diagram that represents the signal flow for all black-box models that are treated in this chapter.

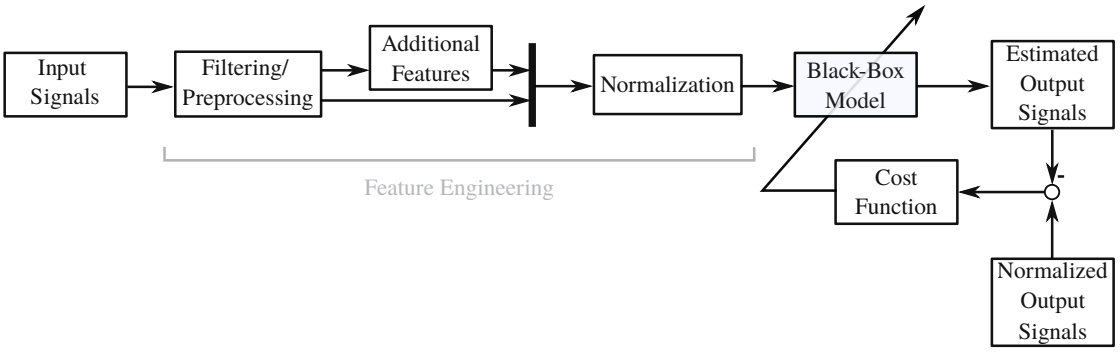


Figure 6.1: Black-box-model training process with optional feature engineering

Throughout this work, the mean squared error (MSE) cost function between estimated and actual temperatures

is used for model fitting and evaluation, where the average is taken across samples and targets:

$$\mathcal{L}_{\mathrm{MSE}} = \frac{1}{KQ} \sum_{q=1}^{Q} \sum_{k=1}^{K} \left(\hat{\vartheta}_q[k] - \vartheta_q[k] \right)^2. \tag{6.2}$$

Hence, each motor component and each sample in the data set is treated equally.

6.1 Feature engineering

The processing of feature vectors either by transformation or by (non)linear combination is referred to as feature engineering (FE) [Dom12]. It marks an essential step in the whole ML training and validation pipeline, and it is considered the most impactful degree of freedom along the process chain. Instances of transformations are the sine and cosine of angles, the logarithm of a sensor value, and normalizations based on various limits or thresholds. Common signal combinations are denoted by multiplications, divisions, additions, or subtractions between different feature vectors, as well as polynomial constellations.

Although the field of FE seems unconditional, unbound, and rife with possibilities, there are drawbacks affiliated with the increase of input features, especially because it can be perpetuated arbitrarily.

- The chance to fall for statistical flukes increases when new noisy features are generated endlessly until a feature emerges that, by chance, seems to have a strong correlation with the dependent variable.
- Adding more and more columns to a data set can lead to a situation where the fraction of samples over features decreases significantly, and the amount of model parameters, which tend to scale with the amount of input features, might approach the order of samples in the data set. This is when the pattern recognition task becomes a memorization task (overfitting).
- The curse of dimensionality [Bel66] looms larger the more features are introduced while the sample size is kept fix, which leads to the feature space becoming sparse. Patterns in this space become less and less identifiable, and extrapolation is more often resorted to than interpolation.
- More features means the model has to process more information, which taxes the computational capacity of the runtime platform.

It is, therefore, advisable to be careful with new input features even if they are genuine new information such as additional sensors in a system, and the ratio of features to samples must not get out of sight. It might be even beneficial to evaluate the information gain of unaltered sensor readings in order to illuminate chances for further model reductions by dropping these sensors from modeling altogether.

In this work, there is no feature-by-feature comparison of model performances, but rather groups of additional features are compared with each other, in order to ease the experimental effort and complexity. Particularly, we denote FE strategies *plain* $\boldsymbol{u}$, *basic* $\boldsymbol{u}_{\mathrm{basic}}$, and *extensive* $\boldsymbol{u}_{\mathrm{extensive}}$ with:

$$\boldsymbol{u}_{\mathrm{basic}} = \begin{bmatrix} \boldsymbol{u} & i_{\mathrm{s}} & v_{\mathrm{s}} \end{bmatrix} \tag{6.3}$$

$$\boldsymbol{u}_{\mathrm{extensive}} = \begin{bmatrix} \boldsymbol{u}_{\mathrm{basic}} & S_{\mathrm{el}} & P_{\mathrm{el}} & n_{\mathrm{mech}} \cdot \boldsymbol{i}_{\mathrm{s}} & n_{\mathrm{mech}} \cdot \boldsymbol{S}_{\mathrm{el}} \end{bmatrix}, \tag{6.4}$$

where

$$\begin{aligned} i_{\mathrm{s}} &= \sqrt{i_{\mathrm{d}}^2 + i_{\mathrm{q}}^2}, \\ v_{\mathrm{s}} &= \sqrt{v_{\mathrm{d}}^2 + v_{\mathrm{q}}^2}, \\ S_{\mathrm{el}} &= i_{\mathrm{s}} v_{\mathrm{s}}, \\ P_{\mathrm{el}} &= i_{\mathrm{d}} v_{\mathrm{d}} + i_{\mathrm{q}} v_{\mathrm{q}}. \end{aligned} \tag{6.5}$$

Most of these combinations are reasoned to be suggestive of the power loss in the motor, which in turn affects heat output (see Sec. 3.1.4). On top of the mentioned FE strategies, the use of exponentially weighted moving averages (EWMAs) and exponentially weighted moving standard deviations (EWMSs) are considered, whose motivation is detailed in the next section. Moreover, three normalization strategies are evaluated, as can be read in Sec. 6.1.2.

6.1.1 The mixed blessing of exponentially weighted moving averages

During the author's first studies on ML for electric motor component temperature estimation, the high relevance of EWMAs and EWMSs for sufficient estimation accuracy were noticed empirically [KWB19b; KWB20; KWB21a]. Initially proposed in the econometric field by [Rob59], EWMAs are a special form of geometric moving averages in which past observations are weighted exponentially decreasing the further in the past they are located. Consequently, recent observations have larger contributions to the current moving average calculation. In contrast to the windowed classic moving average, the EWMA is smoother since there are potentially infinite previous samples considered in the limit with no discontinuities that would be else involved.

The EWMA u_μ and the EWMS u_σ at time k are computed for a quantity $u[k]$ by

$$u_\mu[k] = \frac{\sum_{i=0}^{k} w_i u[k-i]}{\sum_{i=0}^{k} w_i} \quad \text{and} \quad u_\sigma^2[k] = \frac{\sum_{i=0}^{k} w_i (u[k-i] - u_\mu[k])^2}{\sum_{i=0}^{k} w_i}, \tag{6.6}$$

where $w_i = (1-\alpha)^i$ with $\alpha = 2/(s+1)$ and s being the *span* that is to be determined beforehand. The weights of the first s past observations describe approximately 86.5 % of the total of all applied weights. Consecutive calculations of the EWMA and EWMS can be derived from a more computationally efficient form [Fin09] with

$$u_\mu[k] = \begin{cases} u[k] & k = 0, \\ (1-\alpha)u_\mu[k-1] + \alpha u[k] & k > 0 \end{cases} \quad \text{and} \tag{6.7}$$

$$u_\sigma^2[k] = \begin{cases} 0 & k = 0 \\ (1-\alpha)\left(u_\sigma^2[k-1] + \alpha(u[k] - u_\mu[k-1])^2\right) & k > 0 \end{cases}, \tag{6.8}$$

which is highly relevant especially for automotive applications where embedded systems run on cost-optimized hardware. However, (6.7) and (6.8) are only equivalent to (6.6) for infinitely long sequences. In the beginning of a time series, significantly different results will arise between both calculation methods.

In order to illustrate this, different spans for the EWMA of the normalized motor speed in profile 4 are shown in Fig. 6.2. Here, the motor speed is divided by 6000 1/s and the permanent magnet (PM) temperature by 100 °C. In the beginning of the profile, the unrolled calculation leads to EWMAs with negligible differences with respect to each other, before they fan out and converge to the trajectories of the recursive calculation. Apart from this, it can be seen that for some intervals one certain EWMA of the motor speed seems to align well with the PM temperature. This visually indicates the usefulness of these features especially for static models.

In this work, the adherence of several differently spanned EWMAs and EWMSs of each input feature to the original feature set is investigated. The question arises, how many of such additional features should be generated and with which spans. Considering the curse of dimensionality, it is advisable to search for the most critical spans, e.g., by a hyperparameter optimization. With regard to the amount of moving averages and standard deviations, a view on classic lumped-parameter thermal networks (LPTNs) might be worthwhile (see Sec. 3.2.2). Examining LPTNs and their electrical equivalent circuit structure reveals that these are

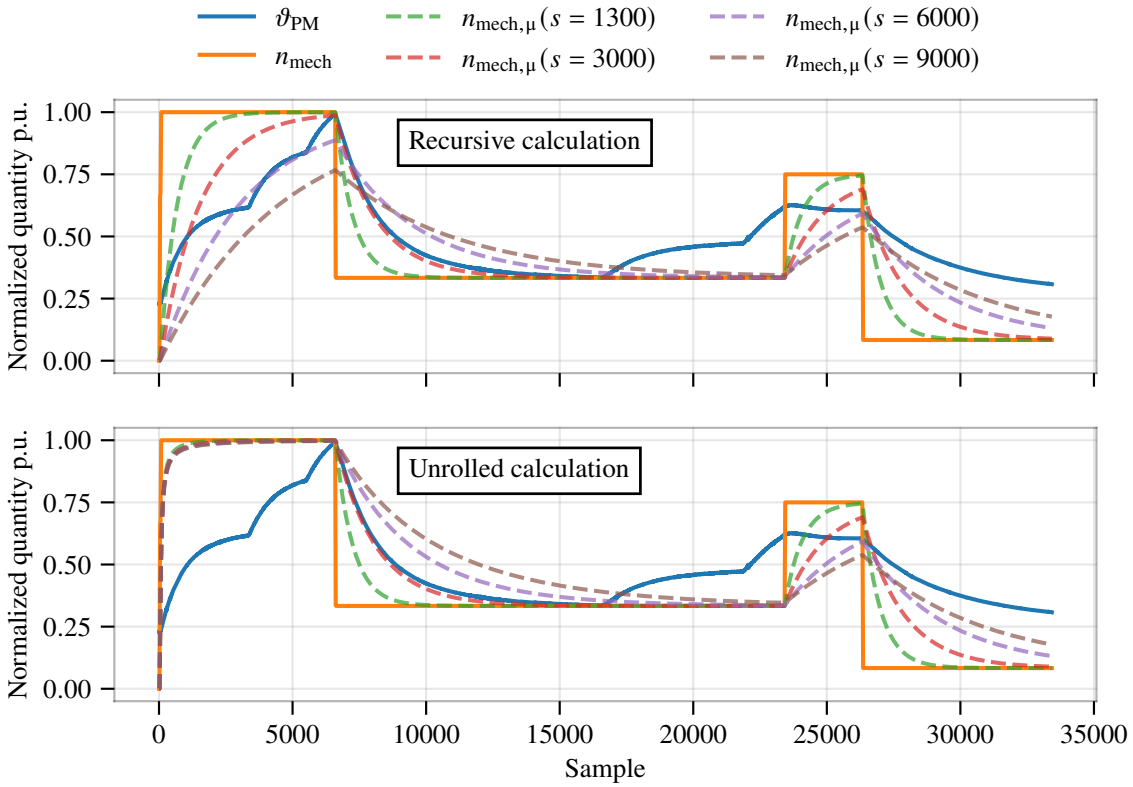


Figure 6.2: The normalized signal trajectories for the PM temperature, the motor speed, and four differently spanned EWMAs thereof are shown for profile 4. The upper figure displays the EWMA calculation by the recursive (6.7), whereas the lower by (6.6). Significant differences can be seen in the beginning of the profile.

characterized by low-pass filters or, equivalently, RC circuits smoothing the raw input data. From signal theory, it is known that RC circuits are infinite impulse response (IIR) filters of the form

$$u = y + RC\frac{\mathrm{d}y}{\mathrm{d}t}, \tag{6.9}$$

which can be discretized to

$$u[k] = y[k] + RC\frac{y[k] - y[k-1]}{\Delta t}, \tag{6.10}$$

with Δt being the step size. Rearranging (6.10) gives

$$y[k] = \frac{RC}{RC + \Delta t}y[k-1] + \frac{\Delta t}{RC + \Delta t}u[k], \tag{6.11}$$

which resembles (6.7) with $\alpha = (RC + \Delta t)^{-1}\Delta t$. Consequently, it is reasonable to directly apply low-pass filters on sensor recordings in order to obtain regressors exhibiting patterns similar to those in LPTNs, and if a LPTN with four nodes was found to be of sufficient accuracy for the contemplated motor [WB16a], then four spans might be an adequate choice. On the other hand, it must be noted that the expert-based LPTN features varying thermal properties depending on the operation point, which does not align with the constant filter strength of the EWMAs and EWMSs as introduced in this work. Consequently, it is allocated to the black-box model to account for this variability.

Considering the manifoldness of the FE process, one could presume further features to exist that exhibit

indicative patterns for the target temperatures. In a student work supervised by the author, an automatic search was conducted for a variety of time-series-based features in order to identify further, potentially ad-hoc, sensor reading combinations with strong correlation to the internal motor component temperatures [Ilt20]. It had been found that no other feature has the predictive abilities of EWMAs and EWMSs. In fact, those that came close were variants of EWMAs and EWMSs or similar transforms, e.g., windowed averages or the absolute energy of the time series, which is the sum of squared values. It is, hence, reasonable to stick with EWMAs and EWMSs, and rather follow feature selection schemes to further reduce the dimensionality of the feature space.

Enriching the data set with EWMAs and EWMSs marks a substantial increase in the amount of input features, especially when applied on all sensor readings with many different spans. Not only is the data set at risk of becoming sparse, but also the computational demand is not to be underestimated, which will soar due to the calculation of the transforms and the increased amount of model parameters. A hardware-based calculation with low-pass filters might be beneficial for real-world applications. Because of the accuracy gain from EWMAs and EWMSs (as will be confirmed later), they cannot be simply ignored, which renders them a mixed blessing in the tooling repertoire for tackling the thermal modeling task.

6.1.2 Normalization

The statistical assessment of physical quantities almost always involves prior normalization into a unified value range. Notable exceptions to this are represented by the application of decision-tree-based models such as extremely randomized trees (ETs), random forests (RFs), or gradient boosting machines (GBMs), whose splitting mechanism along the value range of input features according to an information gain metric is scale-invariant. Nonetheless, in this work, three different normalization schemes will be compared during the assessment of static modeling techniques, in order to then stick with the most adequate scheme when investigating dynamical models. For a certain input feature u_p in the data set $\mathcal{D}$, the standard normalization scheme is denoted for all time steps k by

$$\tilde{u}_p = \frac{u_p - \mu_p}{\sigma_p}, \tag{6.12}$$

with $\mu_p = \sum_{k=1}^{K} u_p[k]/K$ being the population/sample mean and $\sigma_p = \sqrt{\sum_{k=1}^{K}(u_p[k] - \mu_p)^2/K}$ the corresponding standard deviation. Treating u_p as random variable U_p, the resulting variable has no bias and a so-called unit standard deviation. The min-max normalization scheme is likewise denoted by

$$\tilde{u}_p = \frac{u_p - \min_k(\boldsymbol{u}_p)}{\max_k(\boldsymbol{u}_p) - \min_k(\boldsymbol{u}_p)}, \tag{6.13}$$

which ensures a guaranteed value range of $\tilde{\boldsymbol{u}}_p \in [0, 1]$. The third scheme is the simplest with

$$\tilde{u}_p = \frac{u_p}{\max_k(|\boldsymbol{u}_p|)} \tag{6.14}$$

and will be denoted the limit normalization. As an exception, for temperature readings, the denominator is set to be 200 °C. The resulting value range can exceed 1, but all temperatures will receive the same mapping, which can be beneficial when comparing relative temperatures between components. Thus, no particular guarantees for the average value, nor both bounds can be given except for that the bounds will be somewhere within $\tilde{\boldsymbol{u}}_p \in [-2, 2]$ depending on the chosen denominator. More importantly, though, it can be ascertained that when the system is at rest (i.e., no motor speed, no currents, and, thus, no torque) the corresponding sensor records will stay at 0 also after normalization, which could be supportive for power loss estimation (assuming the black-box model finds an abstraction for the concept of power losses). Eventually, scaling all target quantities with the same factor is reasonable when the cost function averages across them evenly (as in

(6.2)). For a standard or min-max scaling, in theory, a model will treat all targets equally, but the inherently different variance in each target will result in unbalanced accuracy after denormalization.

6.2 Cross-validation

In this thesis, not only data-driven models are trained, but also hyperparameter optimizations (HPOs) are conducted. Both these facts lead to the extension of the previously outlined split of the data set into a training and test set onto two additional sets – the validation and generalization set. Since the total data set size remains the same, the training set, and only that set, is shrinked moderately in order to accommodate for four different groups of profiles accordingly. In order to ease experimental effort and to simplify comparison of model performances, all experiments will be conducted on a one-fold base, with the training set denoting the largest chunk of the data set. Refer to Fig. 6.3 for an illustrative overview.

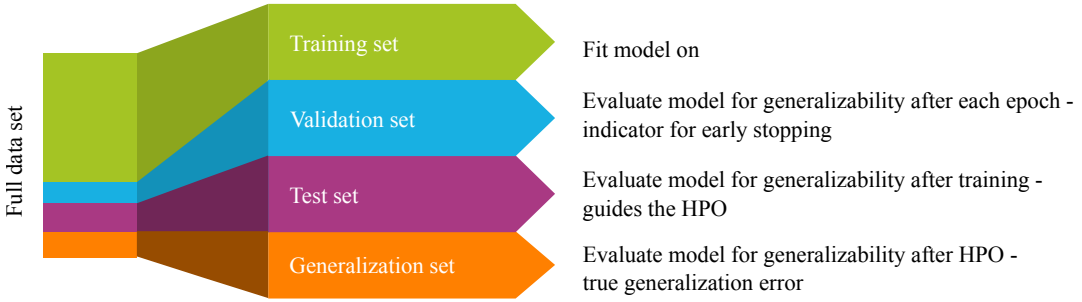


Figure 6.3: Extended cross-validation

The training set consists of those profiles that are seen by a statistical model during training, and which will be used to tune the model coefficients. The validation set exists only for ANN-based training (in this thesis), where it will be used for model evaluation after each epoch, in order to determine the potential deterioration of the generalization capability. If that is the case, an early stopping of the gradient descent is initiated to avoid overfitting. The test set is not used for early stopping but for evaluation after a full training, in order to determine the generalization capability of a model under certain hyperparameters. Using this set to erroneously report a model's generalization capability after a HPO is too optimistic since hyperparameters might be optimized on the test set. Eventually, the generalization set is the set which will yield true generalization capabilities as it will never be seen by the model and never be used to tune any parameters[1].

The potential drawback that accompanies a one-fold cross-validation (CV) scheme is that if profiles are assigned inadequately into the four groups, model evaluation could end up with an out-of-distribution estimate within the nontraining sets, and generalization capabilities might be under- or overestimated. In order to mitigate this risk, a careful assignment of profiles is necessary. Building on the clustered representation in Fig. 5.6, all profiles in nontraining sets were, therefore, hand-picked to be no outlying profile. In Fig. 6.4, the chosen assignment is visualized within the principal component coordinate system. It becomes evident that all profiles that are used for evaluation are not located too far outside of clusters. The resulting assignment for all sets is compiled in Tab. 6.1, where the main training set is shown to denote around 86 % of the whole data set,

[1] In some literature, it is more common to name the last held-out set the test set, instead of the generalization set, and the set for guiding a HPO the validation set instead of test set. However, this clashes with the likewise common notation of the validation set being the set used for early stopping in deep learning.

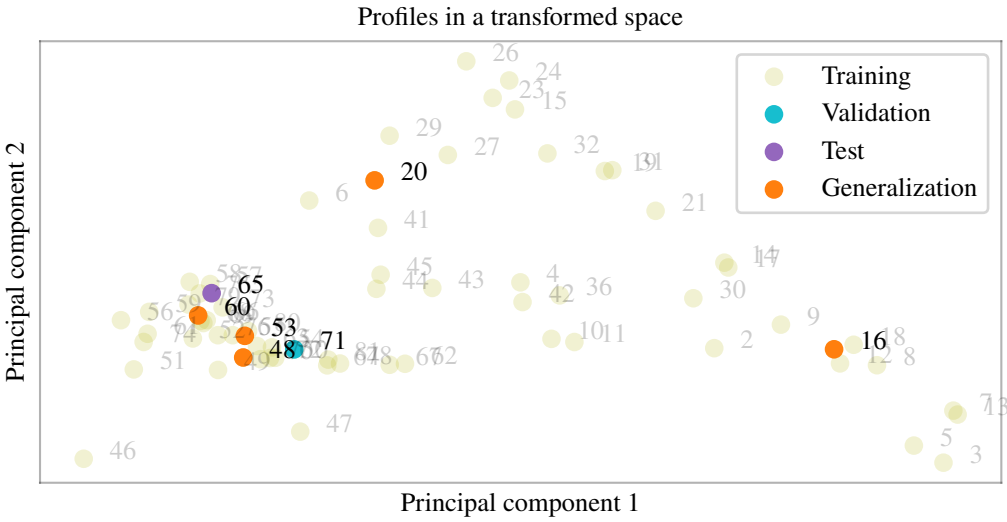


Figure 6.4: Evaluation set assignment

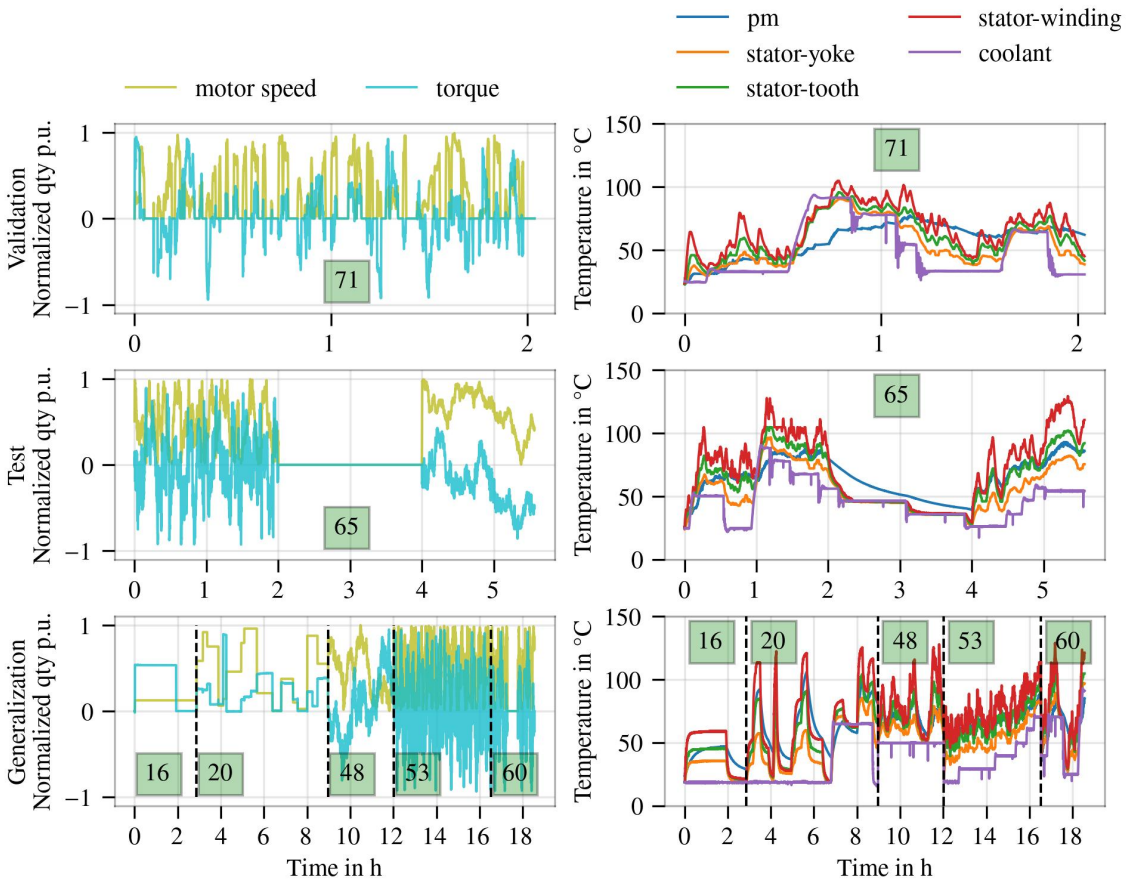


Figure 6.5: The time series plot of the evaluation sets is shown. For the generalization set, multiple profiles are concatenated and visually separated by vertical, dashed, black lines.

and the generalization set, which is going to be used for the final report of generalization capabilities, denotes 10 %. A time series illustration of the exciting features as well as temperature trends for all nontraining sets can be seen in Fig. 6.5.

Label	Symbol	Profiles	Total length	Relative size
Generalization	$\mathcal{G}$	16, 20, 48, 53, 60	18.55 h	10.0 %
Test	$\mathcal{T}_\mathrm{e}$	65	5.56 h	3.0 %
Validation	$\mathcal{V}$	71	2.03 h	1.1 %
Training	$\mathcal{T}_\mathrm{r}$	all remaining	158.68 h	85.9 %

Table 6.1: Profile assignment

6.3 Static modeling overview

In order to get an overview over the performance of the many static models, their hyperparameters, and the variations in the FE phase, a rough grid search is conducted first. For an explanation of the linear regression variants see Sec. 4.3.1, for NN-based approaches see Sec. 4.3.2, and for tree-based variants see Sec. 4.3.3.

The normalization is varied between limit, min-max, and standard normalization, while the FE set is varied between plain, basic, and extensive. Moreover, it is varied whether EWMAs or EWMSs are added to the feature set. For either of EWMAs or EWMSs, four different spans are used at the same time for all input features. The particular span values are $(1320, 3360, 6360, 9480)$ samples, and they are not further optimized for the sake of the overview. The certain choice of span values stems from a previous work [KWB19a], where they have been found to be the most striking for the data set.

The result of the grid search is compiled in Tab. 6.2. It contains the MSE in $(°C)^2$ according to (6.2) for the ordinary least squares (OLS), least absolute shrinkage and selection operator (LASSO), RIDGE, support vector regression (SVR), extremely randomized tree (ET), random forest (RF), gradient boosting machine (GBM), and multi-layer perceptron (MLP) method when trained on the training, validation[2], and test set and evaluated against the generalization set. No HPO is conducted, so the validation and test set can be merged into the training set. Since only rough FE schemes and categorical enablers for groups of features are varied during the grid search, the risk of overfitting through the one-fold CV is marginal. The corresponding map to the number of model parameters can be found in Tab. 6.3. More features usually result in more model parameters.

Furthermore, scores in Tab. 6.2 for all models except for OLS and RIDGE are the best across ten different seeds for the random number generator. Randomness in MLPs ensues from optimizing the nonconvex scalar cost function through gradient descent and starting from different initial positions, which converges in different local minima. A similar reason holds for both LASSO and SVR, which are fitted with coordinate descent and a truncated Newton method with inner conjugate gradients [Fan+08; GL21], respectively. For tree-based algorithms, randomness is the result of the splitting algorithm at each node, which may consider random subsets of samples and features as well as a random order of features.

The scatter for each model is shown in Fig. 6.6. For each model the particular best FE scheme is shown. That is, for all models both EWMAs and EWMSs were used, except for the RF method, which was best with only EWMAs. All models performed best with the extensive FE additions, except for the MLP, which was best with just the basic additions. Eventually, the normalization scheme was more mixed: The OLS had the same performance for any normalization, whereas most models performed best with the limit normalization, except for LASSO and GBM, which preferred the standard normalization. It is worth to mention that tree-based models are scale-invariant and the normalization is not contributing much to the end performance, as can be

[2]The validation set is not used for training in case of the MLP method

	Norm	limit				min-max				standard			
	EWMS	False		True		False		True		False		True	
	EWMA	False	True	False	True	False	True	False	True	False	True	False	True
Model	FE												
OLS	plain	118.33	26.38	97.40	23.45	118.33	26.38	97.40	23.45	118.33	26.38	97.40	23.45
	basic	107.20	17.66	86.81	12.56	107.20	17.66	86.81	12.56	107.20	17.66	86.81	12.56
	extensive	83.64	15.43	83.81	11.07	83.64	15.43	83.81	11.07	83.64	15.43	83.81	11.07
LASSO	plain	122.37	25.00	99.86	21.64	117.99	23.67	95.10	20.47	118.32	24.81	96.68	22.33
	basic	109.80	15.33	83.72	11.90	107.12	14.33	82.79	10.43	107.21	15.87	87.24	9.50
	extensive	102.99	14.90	78.83	11.00	95.85	13.85	75.71	9.02	86.40	12.90	76.65	7.51
RIDGE	plain	118.33	25.92	97.29	23.51	118.33	25.89	97.23	23.34	118.33	26.35	97.39	23.45
	basic	107.16	17.30	86.78	11.56	107.20	17.22	86.75	11.63	107.20	17.61	86.81	12.45
	extensive	83.73	14.31	77.66	10.19	83.78	14.19	77.63	10.22	83.65	15.14	82.61	10.98
SVR	plain	118.38	25.12	96.68	22.20	118.37	25.20	96.50	22.37	118.34	26.19	97.34	23.61
	basic	107.06	15.68	86.88	9.33	107.19	15.79	86.11	9.67	107.20	17.41	86.76	11.70
	extensive	83.85	12.79	75.06	7.28	83.74	13.12	75.14	7.86	83.65	14.53	79.28	10.34
ET	plain	86.44	18.05	68.29	18.67	85.76	18.57	68.50	17.78	86.74	19.04	70.66	20.44
	basic	84.89	14.91	65.71	15.03	86.94	15.57	66.47	15.56	87.42	16.22	66.61	16.75
	extensive	88.56	14.98	61.75	14.20	86.82	14.82	65.80	15.04	89.04	15.30	68.60	15.75
RF	plain	106.50	29.49	98.24	33.92	118.37	28.33	108.68	30.99	119.51	28.39	114.46	30.79
	basic	102.53	24.67	95.35	26.59	116.04	29.52	101.24	34.09	118.44	29.67	104.61	29.94
	extensive	110.28	24.48	92.15	25.59	110.73	29.18	102.49	30.70	112.85	27.02	98.80	27.15
GBM	plain	70.41	9.17	46.64	9.59	70.39	9.16	46.64	9.60	70.39	9.17	45.97	9.59
	basic	70.03	8.42	46.24	8.79	70.03	8.42	46.23	8.79	70.11	8.42	46.31	8.79
	extensive	68.26	7.88	44.61	7.69	68.18	7.88	44.61	7.69	68.26	7.88	44.61	7.68
MLP	plain	68.58	6.56	46.07	5.42	70.29	5.94	53.54	6.07	68.10	5.14	54.54	6.57
	basic	64.62	4.24	42.33	**3.58**	72.50	4.91	41.27	4.12	67.98	4.73	54.01	4.75
	extensive	55.96	4.66	40.40	4.24	56.39	3.73	41.04	4.19	56.55	4.78	59.60	4.46

Table 6.2: Grid search for FE configurations with static black-box models (minimum MSE in $(°C)^2$ across seeds)

Mov. Avgs.	Neither			EWMA or EWMS			EWMA and EWMS		
FE	plain	basic	extensive	plain	basic	extensive	plain	basic	extensive
Model									
OLS	32	40	56	144	184	264	256	328	472
LASSO	32	40	56	144	184	264	256	328	472
RIDGE	32	40	56	144	184	264	256	328	472
SVR	28	36	52	140	180	260	252	324	468
ET	120458	120902	120500	121634	121778	121754	122762	122654	122756
RF	122582	122498	122486	122420	122402	122378	122834	122840	122834
GBM	18194	18194	18197	18197	18194	18182	18200	18200	18197
MLP	196	228	292	644	804	1124	1092	1380	1956

Table 6.3: Grid search for FE configurations with static black-box models (number of model parameters)

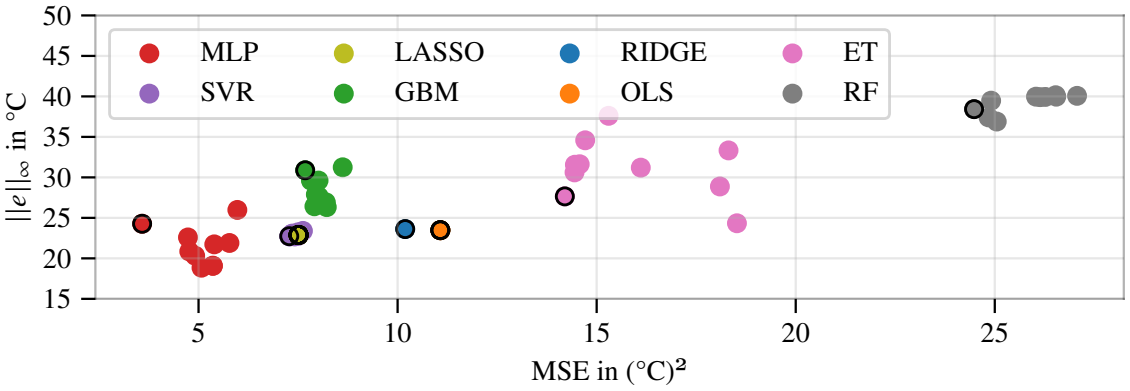


Figure 6.6: The performance of all static models over the MSE and worst-case error $||e||_\infty$ on $\mathcal{G}$ is shown. The scatter for MLP, LASSO, and SVR is due to different random initializations and for RF, ET, and GBM due to varied searches for splits. The lowest MSE per model is encircled in black.

inferred from Tab. 6.2 as well. From Fig. 6.6 it becomes evident that the ET and RF method lack performance in terms of the MSE by a large margin. Among the tree-based algorithms, the GBM is clearly the most precise and in the range of linear models. In addition, the superiority of the MLP over the other models becomes obvious. The best MSE score of the MLP is an outlier with respect to the other repetitions, which emphasizes the importance of repeated experiments. The linear models are very close to each other in accuracy, however, SVR and LASSO – which are nearly identical in performance – are slightly better than OLS and RIDGE. A time series illustration of the estimation performance in the generalization set for LASSO is shown in Fig. 6.7, for the best MLP in Fig. 6.8, and for the GBM in Fig. 6.9.

Here, all profiles of the generalization set are concatenated one after another, and the concatenation points in time are marked with upper black vertical lines. Apart from the fact that the MLP exhibits the highest accuracy with the whitest error noise, it becomes clear that high errors spike in the beginning of some profiles. These spikes usually denote the worst-case error along the trajectories and can be attributed to the skewed EWMA and EWMS value at the beginning of a measurement profile. The effect is especially severe for profiles that start with a pre-heated motor, which explains why these spikes do not occur for every measurement profile. These errors cannot be avoided because the explicit black-box structure does not allow for setting the initial prediction, especially for static and stateless models. Consequently, in a production implementation a calibration time of a few seconds to minutes would become necessary.

In order to interpret Tab. 6.2 and Tab. 6.3 appropriately, the models' hyperparameters need to be mentioned. The regularization coefficient for LASSO and RIDGE was $\alpha = 1 \cdot 10^{-4}$ and $\alpha = 1.0$, respectively. The linear SVR was solving the bias-free primal problem rather than the dual one, with $\epsilon = 0$, so no insensitive band existed in the ℓ_2 loss with a likewise ℓ_2 penalty on the linear coefficients. The regularization parameter was $\alpha = 1.0$, and the maximum number of iterations for the truncated Newton method was 1000. In case of RF and ET, the number of estimators was 20, and the maximum tree depth denoted 10 levels. The GBM, however, was granted 50 estimators at a maximum depth of 5. Eventually, the MLP comprised one hidden layer with 16 neurons followed by the rectified linear unit (ReLU) activation function [NH10], and an output layer with four neurons and the identity activation function. The batch size denoted 2^{11} and the number of epochs was 400.

From Tab. 6.2 it can be derived that the best scores apply for the MLP with the basic feature set. Especially EWMAs improve the score significantly, which can be further improved by adding EWMSs. Having only EWMSs is not as striking as EWMAs. This is further emphasized in Fig. 6.10, which displays statistical quantiles for each FE scheme. The boxplots are organized such that each box filters for either EWMA/S, the normalization scheme, or the feature addition strategy, whereas the other two categories are included entirely. It becomes apparent that enabling EWMAs has the most striking impact on the MSE among all FE

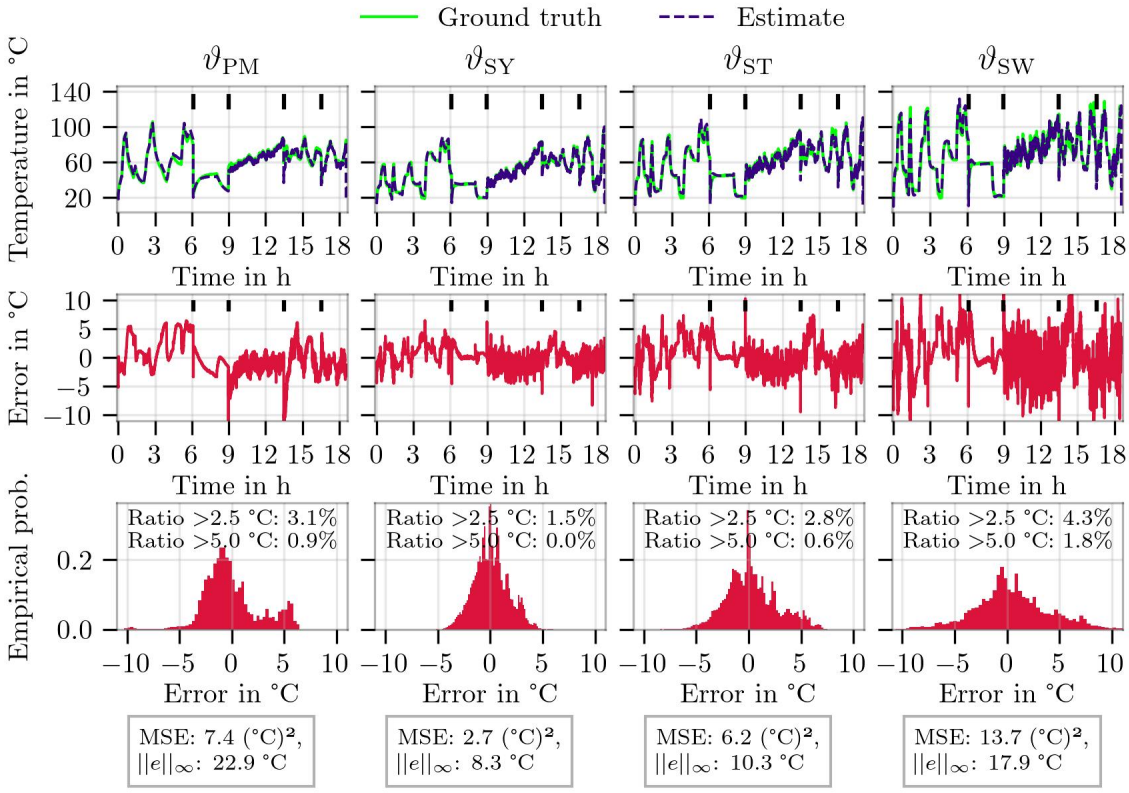


Figure 6.7: LASSO estimation performance for the generalization set

schemes. Adding more features generally leads to a higher median accuracy, although the extensive set gives only a slight edge with respect to the basic set. The standard normalization scheme results in a worse median performance than the other two, which are mostly equal in performance. Since the limit normalization denotes the simplest scaling, it will be used from now on. Moreover, it can be concluded that it is advisable to use a basic or extensive FE addition and at least EWMAs for high accuracies during the next investigations with dynamic models.

A look at Tab. 6.3 reveals that linear models feature the least amount of model parameters, followed by the MLP with roughly four times the amount, and tree-based models require by far the most parameters with roughly up to 100 times the amount of what MLPs need (10 times in case of the GBM). From a memory requirement point of view, tree-based models seem to fall short of achieving real-time temperature estimation with high accuracies. It should be noted, however, that in contrast to linear models and ANNs, a tree-based model does not need all its parameters for an estimate. The parameters of the differently ensembled trees consist of thresholds at different nodes and estimates at leaf nodes. For any estimate only a certain path down the tree is followed. Thus, the parameter set of a tree-based model can be understood as a LUT, and computational demand does not scale as steep with more parameters as it does for linear models and ANNs. On the other hand, storing over 120 thousand parameters for a thermal model is significant and seems to be not efficient in view of the small number of parameters a MLP requires for a better MSE. The accuracy of the considered tree-based models can be further increased by allowing for more estimators in the ensemble, and for a larger depth of the trees. However, this would result in even more parameters, whose feasibility in embedded systems becomes more and more unlikely. This disadvantage is so substantial that it cannot be alleviated by the fact that tree-based models are invariant to the scaling of features, and the amount of features. The invariance attributes may be declared as assets of tree-based models, speaking in their favor when dealing with data sets with many features. Yet, here the plain size of the parameter space approximates the magnitude of the data set size, which is the diametrical opposite of avoiding overfitting, see Sec. 6.1.

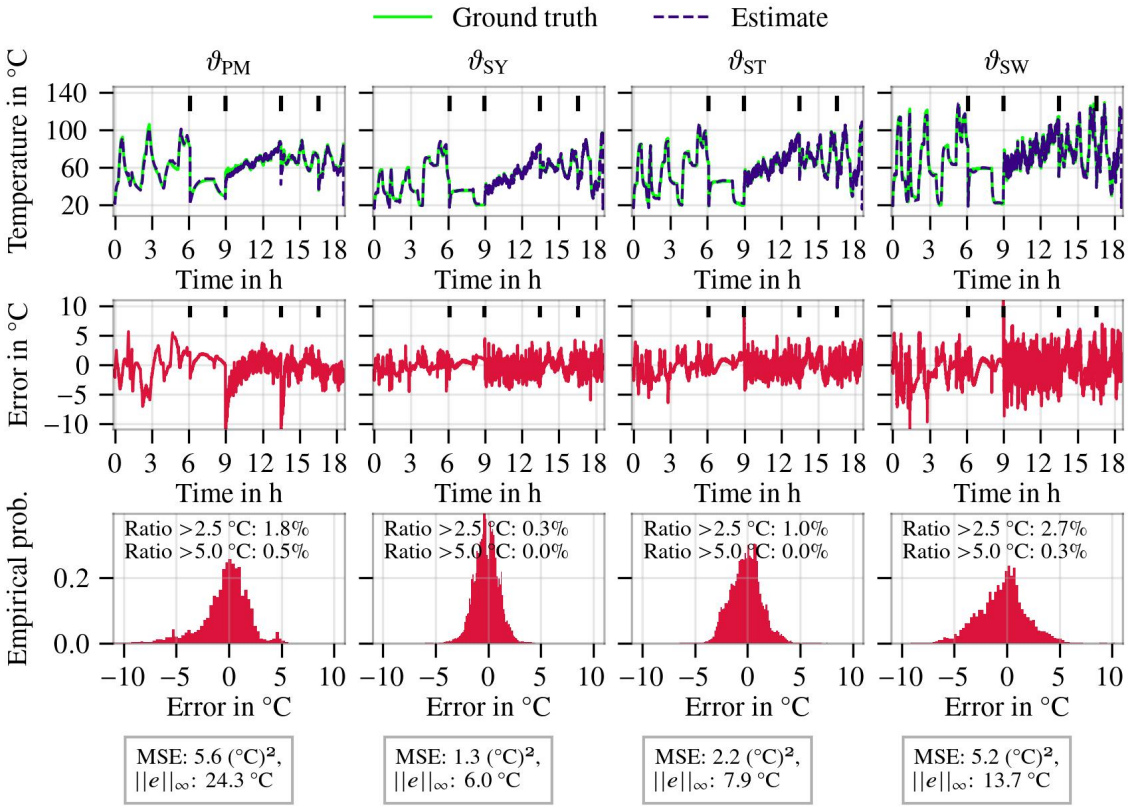


Figure 6.8: MLP estimation performance for the generalization set

Aside from the accuracy and model size, the training duration might be interesting, since a fundamental benefit of ML-based approaches to thermal modeling is their automatic design in a presumably short amount of time. The training time for each model with its corresponding best FE scheme for a given central processing unit (CPU) is compiled in Tab. 6.4. Note that iterative training methods can be made faster by reducing the number of iterations. Especially in case of the MLP method the training and validation score trend in Fig. 6.11 suggests that convergence is reached already after 100 epochs. More generally, it becomes evident that tree-based models, despite their required number of parameters, train rather quickly on the data set with at most 100 seconds on a 20-core machine, where they can utilize a high degree of parallelization. The longest median training times can be found for the iteratively optimized LASSO and MLP with 782 and 1094 seconds, respectively. Next to the potential reduction of training iterations to speed up training duration, however, it is also relevant to mention that the MLP method used ten threads not to fit one model but ten models with different seeds in parallel in contrast to all other models. In order to get the same sample of models for LASSO, SVR, ET, RF, and GBM, the median training time needs to be multiplied by ten, consequently. Only the OLS and RIDGE method utilize the analytically closed solution for linear models, which involves a matrix pseudo inverse by means of the singular value decomposition with a compute complexity of $O(KP^2)$. It is not an iterative optimization, and, thus, no repeated experiments are necessary. They denote the fastest training algorithms, followed by the MLP when taking the amount of parallel seeds and only 100 epochs into account.

All implementations except for MLP use scikit-learn v1.0 [Ped+11] and higher in Python 3.9. The SVR algorithm, in particular, wraps LIBLINEAR [Fan+08], whereas OLS and RIDGE wrapped LAPACK [And+99] using singular value decomposition and Cholesky decomposition, respectively. The MLP method employs PyTorch v1.10.0 [Pas+19] in Python 3.9. All training runtimes are highly implementation-dependent and scale differently with better hardware. Hence, the numbers in Tab. 6.4 are to be understood as a qualitative orientation.

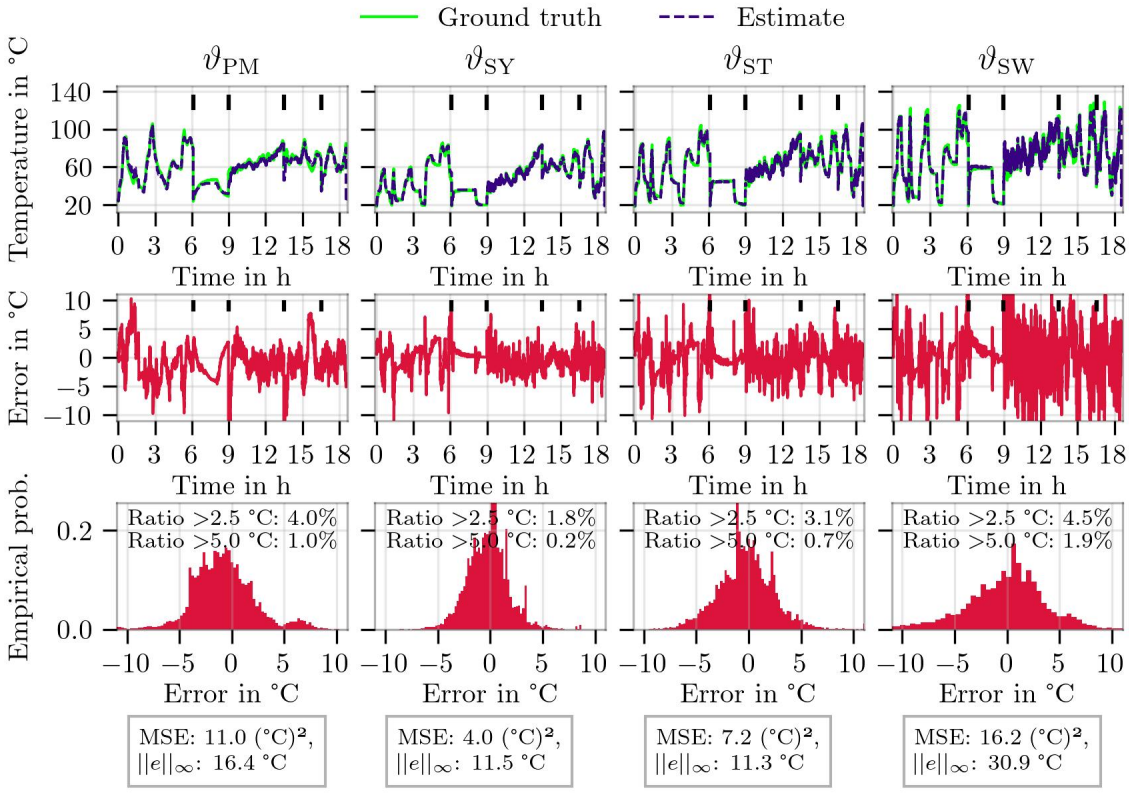


Figure 6.9: GBM estimation performance for the generalization set

In summary, it can be concluded that tree-based models, despite their short training duration, are rather inadequate for real-time temperature estimation tasks. This is majorly due to the high amount of model parameters they require at modest to poor estimation accuracies. The reason behind the high amount is fundamentally inherent in the model concept, such that a HPO will not mitigate this drawback substantially. A similar conclusion was drawn in a master's thesis supervised by the author that was focused on GBM implementations [Not20]. Hence, tree-based models will not be investigated further in this work.

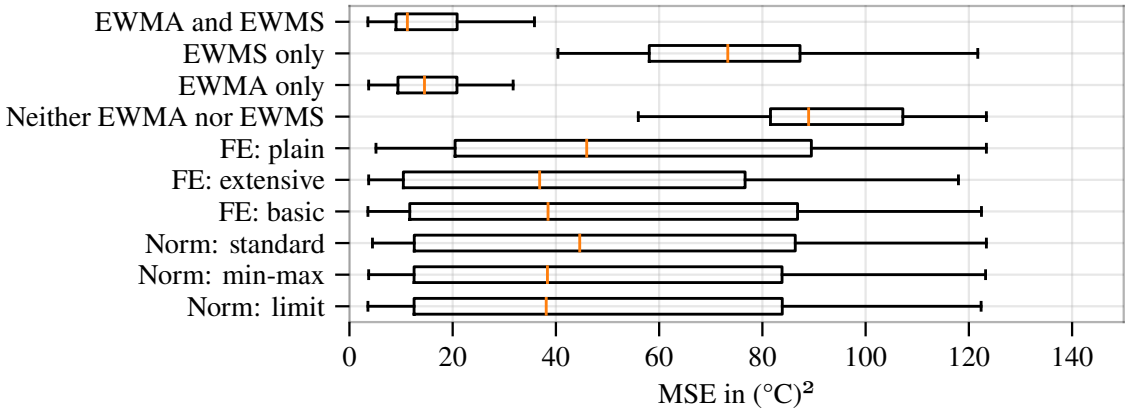


Figure 6.10: Box plots across FE schemes over the MSE in $(°C)^2$ including all models

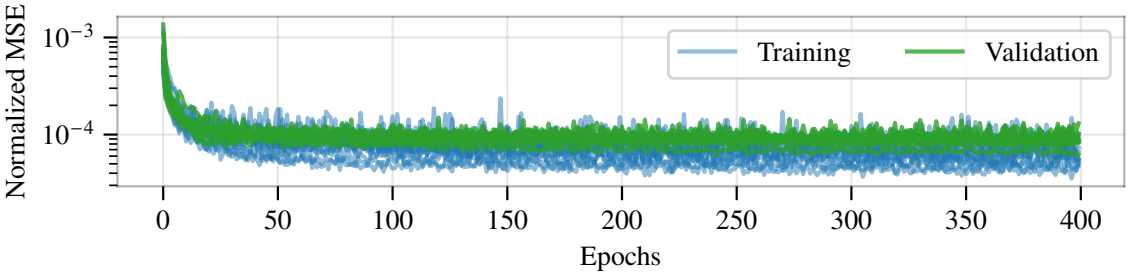


Figure 6.11: Training and validation trend of the MLP with its best FE scheme

Model	Median training time in seconds	CPU
OLS	12	Intel i7-7700K – 1 thread across 4 cores @ 4.5 GHz
LASSO	782	Intel i7-7700K – 8 threads across 4 cores @ 4.5 GHz
RIDGE	1	Intel i7-7700K – 1 thread across 4 cores @ 4.5 GHz
SVR	217	Intel i7-7700K – 1 thread across 4 cores @ 4.5 GHz
ET	35	2x Intel Xeon Silver 4114 – 20 threads across 20 cores @ 3.0 GHz
RF	100	2x Intel Xeon Silver 4114 – 20 threads across 20 cores @ 3.0 GHz
GBM	44	Intel i7-9800X – 16 threads across 8 cores @ 4.4 GHz
MLP	1094	Intel i7-9800X – 10 threads across 8 cores @ 4.4 GHz

Table 6.4: Training times of static black-box models under best FE configurations

6.4 Linear thermal modeling analysis

Due to the relatively good performance of linear models at few model parameters and a fast training time, they shall be investigated in more detail in the following. In particular, span values and the regularization strength are optimized, the effect of reduced training sets are investigated, and feature selection strategies are compared with each other. For all experiments in this chapter, the extensive feature set extension with EWMAs only and the limit normalization strategy is followed.

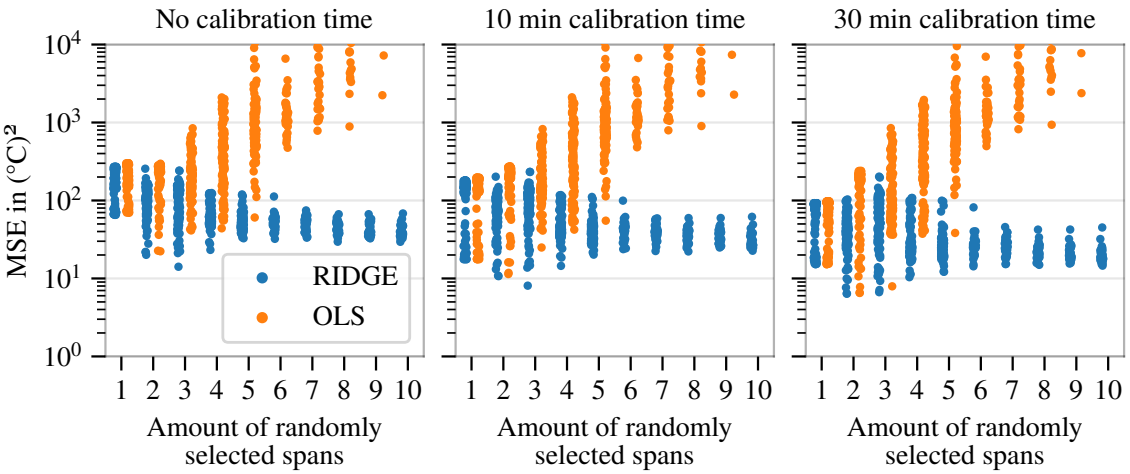


Figure 6.12: Random search over an increasing amount of spans and different initial dead times (calibration). The MSE is evaluated on $\mathcal{T}_\mathrm{e}$.

In Fig. 6.12, it can be seen that the best MSE can be achieved with three EWMAs in case of the RIDGE. The OLS method performs best even with only two spans. The MSE as evaluated on $\mathcal{T}_\mathrm{e}$ and trained with $\mathcal{T}_\mathrm{r} \subseteq \mathcal{D} \setminus (\mathcal{G} \cup \mathcal{T}_\mathrm{e})$ is reported here. The RIDGE model always achieves better best-case scores than the OLS model. With an increasing amount of spans the RIDGE model converges to a certain best-case MSE, while the OLS method will suffer from numerical instability with a worsening regressor matrix condition. In this experiment 99 random seeds for one to five differently sampled spans are used and 33 for six and above spans. Random spans are sampled between 1 minute and 2 hours with a 30 s resolution. More specifically, in order to mitigate multicollinearity between EWMAs with similar spans, the spans are sequentially selected in dependence on previously selected spans. Already sampled span values for one experiment with a certain seed will preclude this span value and a ±5 min window around it for the next samples in the same experiment and seed. All scores are divided into three different virtual calibration times of 0, 10, and 30 minutes. In case of a nonzero calibration time, the MSE is calculated just after that time has passed in the test profile. This corresponds to an ECU that is responsible for temperature estimation to block its estimates for the mentioned amount of (calibration) time in order to let the moving average features get tuned to the trajectory first. It becomes evident that the average and best-case MSE improve significantly for any amount of spans when waiting for a longer calibration time. This highlights the aforementioned high errors in the beginning of profiles where EWMAs with high span values might be detuned.

The optimal span values denote 699, 10290, and 13570 samples, which translates to 5.8 min, 85.8 min, and 113.1 min, respectively, due to the sampling frequency of 2 Hz. Keeping this EWMA setting, the regularization strength parameter α is varied next for the RIDGE and LASSO method. In this grid search, the optimal regularization strength is to be determined.

The resulting MSE on $\mathcal{T}_\mathrm{e}$ is displayed in Fig. 6.13. It becomes clear that stronger regularization improves performance for RIDGE but worsens it for LASSO and for lighter regularization vice versa. The best MSE

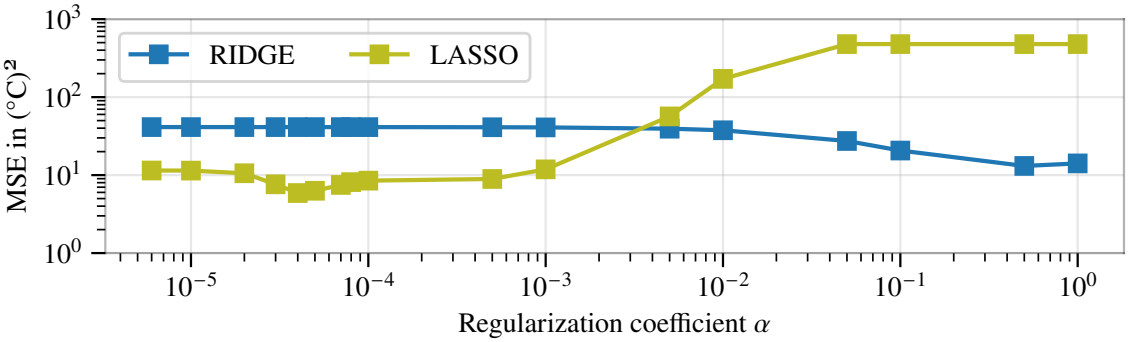


Figure 6.13: Grid search over different values for α. The MSE is evaluated on $\mathcal{T}_\mathrm{e}$.

is achieved with $\alpha = 4 \cdot 10^{-5}$ for LASSO and $\alpha = 0.5$ for RIDGE. The corresponding performance on $\mathcal{G}$ is 8.38 °C^2 and 13.1 °C^2 for LASSO and RIDGE, respectively. This is an improvement over 14.90 °C^2 and 14.31 °C^2 from Tab. 6.2 for LASSO and RIDGE, respectively, with the same FE configuration but different regularization strength and EWMA spans. Yet it is not better than the best configuration that was used in the overview experiment, i.e., with standard scaling (LASSO) or limit scaling (RIDGE) and additional EWMS features (both), that led to 7.51 °C^2 and 10.19 °C^2 for LASSO and RIDGE, respectively. The optimized spans and regularization strengths are, thus, not leading to peak performance when other FE schemes are available, but they do maximize the performance (generalization capability) for the given amount of model parameters. Moreover, their ranking among static black-box models according to the achievable cross-validated MSE does not change just by optimizing spans and the regularization parameter α.

Another simple approach to further reducing the required number of model parameters for any black-box model is to reduce the number of input features. It is likely that some input features are more important than others, such that it will be crucial to eliminate only the less relevant. Moreover, a reduced feature set mitigates the curse of dimensionality, and, hence, better performance with fewer features might be even possible. In the following, two different recursive feature eliminiation (RFE) strategies are evaluated for the three linear models OLS, LASSO, and RIDGE. During the elimination, spans and regularization strengths are kept as found above, and one input feature is dropped at a time according to the particular strategy.

The first elimination strategy drops the input feature with the highest variance inflation factor (VIF) [HTF09]. The VIF for the i-th regressor is defined as

$$\mathcal{L}_{\mathrm{VIF},i} = \frac{1}{1 - r_{d,i}^2}, \tag{6.15}$$

where $r_{d,i}^2$ is the coefficient of determination for an OLS fit with the regressor matrix containing all input features except for the i-th regressor, which acts as dependent variable. A high VIF value means that the corresponding regressor is easily predictable by the other regressors, which is an indication of collinearity. Since highly correlated input features do not contribute to a better linear fit of the data, their relevance can be declared to be minor. All VIFs are calculated by fitting and testing on $\mathcal{T}_\mathrm{r}$.

The second elimination strategy drops the input feature with the lowest absolute value for its corresponding coefficient $|\beta_i|$. In contrast to the first strategy, the coefficients are obtained by utilizing the LASSO method instead of by OLS in order to mitigate the high variance problem with multicollinear input features. In fact, the ℓ_1-norm in the cost (4.8) tends to shrink the coefficients of fewer relevant input features down to zero, which yields a natural feature selection mechanism in itself [HTF09]. The performance curves for the different subsets of input features for all three linear models when trained on $\mathcal{T}_\mathrm{r}$ and cross-validated on $\mathcal{T}_\mathrm{e}$ are plotted in Fig. 6.14.

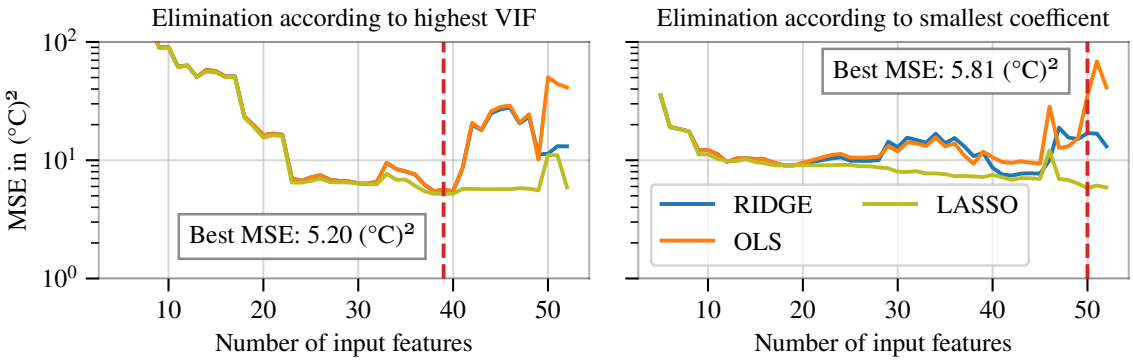


Figure 6.14: Recursive feature elimination according to two different strategies, VIF values and absolute coefficient values. The dashed red vertical line indicates the lowest cost on $\mathcal{T}_\mathrm{e}$ for any of the three linear models.

The best MSE (6.2) is almost always achieved with LASSO for any considered subset of input features. The fewer regressors are used the more all three linear models tend to converge to the same performance as multicollinearity is eliminated. It becomes evident that the best performance can be found during the VIF elimination strategy with a MSE of 5.2 $^\circ\mathrm{C}^2$ and 39 input features (down from 52). The starting point of 52 input features stems from the extensive feature addition and three EWMA spans per regressor. Among the eliminated features are all low-pass filtered versions of the motor speed n_mech. Besides this, no other feature or its filtered variants are removed consistently.

The resulting performance of each linear model on $\mathcal{G}$ with the 39 input features subset is 15.1 $^\circ\mathrm{C}^2$ for OLS and RIDGE, and 19.6 $^\circ\mathrm{C}^2$ for LASSO. Hence, in contrast to the previous optimization measures, the reduction of input features leads to an improvement only for the set according to which the subset is collated, but the generalization capability worsens. This is not necessarily only due to the RFE but might also stem from $\mathcal{T}_\mathrm{e}$ being too different compared to $\mathcal{G}$. Because of the generalization deterioration the reduced subset is not kept for future experiments.

Interestingly, when following the RFE for absolute coefficient values, the MSE for all linear models can be kept at approximately 10 $^\circ\mathrm{C}^2$ for $\mathcal{T}_\mathrm{e}$ until as few as 12 regressors, which amounts to just 13 model parameters per modeled temperature. Thus, if the resulting runtime is of utmost concern and the set that is used for optimization is representative, this procedure will give the most promising feature subset.

Eventually, it should be reiterated that the curves in Fig. 6.14 are the result for a certain set of spans, normalization, and regularization parameters. Altering these factors might lead to varying curves. Moreover, it is worth mentioning that the found subset of features that minimize the cost is applicable to linear models, but is not guaranteed to be also the best subset for nonlinear models. Therefore, the found subset does not denote the starting point for the following experiments with dynamic models.

A concluding analysis of linear models is the variation of the training set size given $\mathcal{T}_\mathrm{e}$ as cross-validation set. This investigation shall give an overview of how much measurement data might be necessary to achieve a desired accuracy. The result is shown in Fig. 6.15: The training set size is reduced in 10 %-steps from 100 % to 10 %, and for each size 50 different subsets of the original training set are sampled. This is repeated for each of OLS, LASSO, and RIDGE.

It becomes evident that with subsets of the training data a comparable or even better performance can be achieved for each of the three linear models. Moreover, for LASSO and RIDGE a similar performance as with the full training set can be achieved with a certain subset denoting just 10 % of the full training data. However, the performance scatter tends to increase with smaller subsets, which is expected as the chance increases for the subsets to have substantially different characteristics compared to $\mathcal{T}_\mathrm{e}$. The scatter is the largest for OLS,

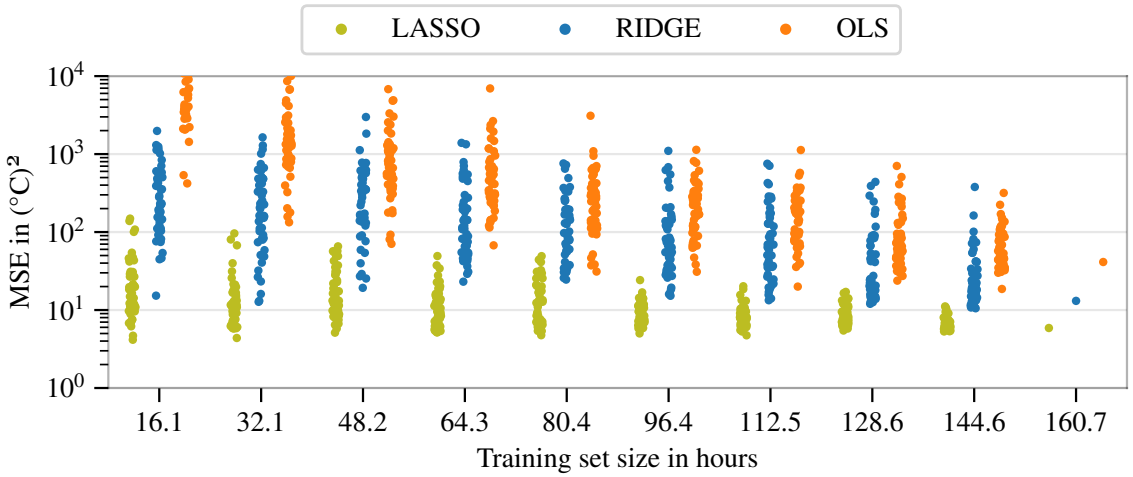


Figure 6.15: The training set size is varied in 10 %-steps where all linear models are evaluated on $\mathcal{T}_e$. 50 samples are drawn per size. Profiles are shuffled before a coherent subset is sampled.

then RIDGE, and eventually LASSO. The same ranking can be spotted for the maximum and mean MSE per portion across random seeds. The evaluation in Fig. 6.15 explains that 160 h of training data is not necessary for achieving good generalization capabilities with a linear model as long as the test set is representative for all future scenarios and the smaller training subset features similar characteristics as the test set. Note, however, that the subset size is not to be treated as another hyperparameter that could be optimized, but rather as an indication for future measurements. Doing so, nonetheless, will lead to overfitting, since the finiteness of the test or generalization set would be ultimately exploited by handpicking profiles with very similar statistics, and the insufficient representation of the test or generalization set for all future scenarios would be ignored.

In summary, among the linear models OLS, RIDGE, and LASSO, the latter outperforms the other two in every investigation. This finding aligns with a master thesis on a different motor [Sto20]. Yet it was not confirmed in a previous work on linear thermal modeling from this dissertation's author [KWB19b]. There, some hyperparameters, feature engineering schemes, cross-validation strategies, and the data itself were different that all might have played a role in not recognizing the superiority of the LASSO method. Especially the regularization strength was not varied down to a scale where the LASSO begins to outperform RIDGE. Notwithstanding this discrepancy, the analyses in this work are more thorough, and together with the results in [Sto20] a recommendation for LASSO can be given in case a linear thermal model is sought after for electric motor components.

6.5 ANN-based dynamic thermal modeling

Heat transfer mechanics can be described by a dynamic system or – more specifically – a system of differential equations, see (3.32). This implies that consecutive temperature sensor readings cannot fluctuate arbitrarily. When put into a stochastic context where temperature sensor values are treated as repeatedly sampled realizations of random variables, the dynamic property is to be translated into the statistical property of these random variables not being independent and identically distributed. This property, however, is a central assumption over the observational data for all static models that were investigated so far. The violation of this assumption is one of the reasons why only with EWMAs as additional regressors the estimation performance of static models can reach an acceptable level.

The question arises whether inherently dynamic models might better capture the statistical patterns in a

temperature sensor readings data set, or at least are capable of achieving the same estimation accuracy with fewer model parameters. In this section, dynamic ANN-based models are investigated for their fit on the thermal modeling task. In particular, three different topologies are considered, which were introduced in Sec. 4.3.2: The long short-term memory (LSTM) (4.12) and gated recurrent unit (GRU) (4.13), as recurrent topologies, as well as the sliding-kernel-based temporal convolutional neural network (TCN) (4.14). Beyond their standard configuration, some additional topological modifications and data processing variants are considered and outlined in the following, which are further tailored to the task at hand: data-driven thermal modeling with measurement sessions of different lengths. In this work, all experiments around dynamic NN models are facilitated by PyTorch v1.13 and Python 3.9.

6.5.1 Further topological enhancements for sequence modeling

The LSTM, GRU, and TCN are topologies that were designed for the general purpose of modeling time series data with important events also far in the past. Memory blocks in the LSTM and GRU were invented to mitigate the exploding and vanishing gradient problem for long sequences, and the dilatated convolution in a TCN enables it to increase its receptive field exponentially with more layers, that is, along its depth. On top of this, further modifications are worked out on these models that are tailored to the thermal modeling task without touching heat transfer principles yet (which, in turn, is treated in Sec. 7).

Among these modifications, residual connections represent important architectural extensions. First proposed in [He+15a] for image processing and then formally described in [BKK18] for sequence modeling, a residual or shortcut connection is denoted by the addition of a NN layer's input to its output according to

$$\boldsymbol{h}^{(l)} = \boldsymbol{\sigma}_{\text{act}}\left(\boldsymbol{h}^{(l-1)} + \boldsymbol{f}(\boldsymbol{h}^{(l-1)})\right), \tag{6.16}$$

where $\boldsymbol{h}^{(l)}$ is the output of layer l, $\boldsymbol{\sigma}_{\text{act}}$ is an activation function, and $\boldsymbol{f}(\cdot)$ is the layer's processing function for its input tensor. This simple topological variant has been proven to increase learning efficiency especially for deep neural networks. As a side note, this modification can be related to approaching the function approximation task with an explicit Euler solver, where $\boldsymbol{f}(\cdot)$ denotes the right-hand side of a differential equation, which was the foundation for the idea of NODEs (see Sec. 4.4.3).

The originally proposed residual connection is usually accompanied by two weight layers in a TCN, which would form a so-called residual block, see Fig. 6.16. In this work, however, in order to reduce the model size for an improved real-time computation capability, a residual connection will be also drawn over single weight layers, that is, for recursive architectures after a memory block, and for the TCN, optionally, after one convolutional layer. Note that a residual connection was not proposed for recurrent architectures in [BKK18]. The shortcut connections will be denoted by the identity function when start and end of the shortcut consist of the same amount of dimensions (i.e., the same shape). Since this is seldomly the case, the shortcut connection is more often rendered by an MLP for recurrent architectures or a CNN with a kernel size of one for TCNs. Weight normalization [SK16] and dropout [Sri+14] are applied for the TCN only, in accordance with [BKK18] and in contrast to [KWB20] where batch normalization was used [IS15]. What is more, for TCNs the dilation rate will be doubled after each residual block, which could consist of only one convolutional layer accordingly.

Another possible topological extension could be to expand the output layer into multiple sublayers, also called multiple heads. For example, in [KWB20], the rotor temperature was granted an own output layer and another was assigned for the three stator component temperatures in order to accommodate the substantially higher time constant of the PM temperature with respect to the stator component temperatures. This had been empirically shown to be beneficial for the estimation accuracy, but also to increase the model size and complexity. Moreover, it was shown in Tab. 6.2 and especially in Fig. 6.8 that a small MLP with no multiple heads can likewise estimate all component temperatures with no significant imbalance between rotor and stator components. The possibility to incorporate multiple heads is, thus, excluded from this work.

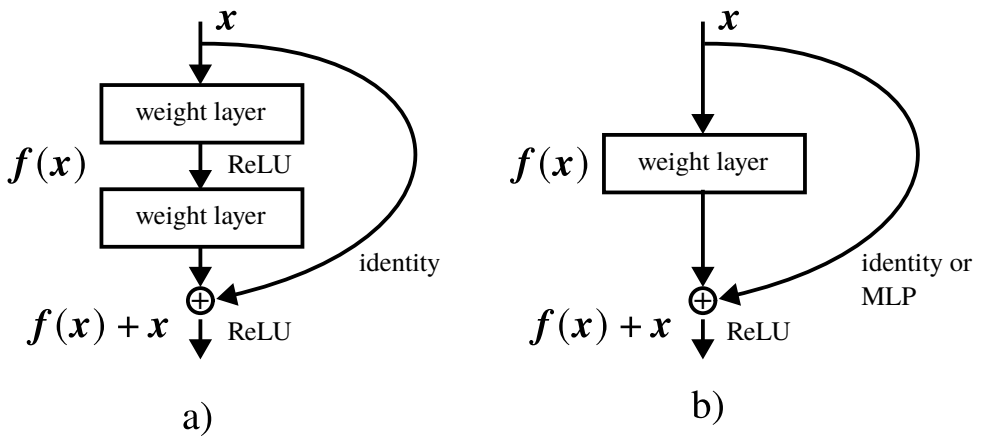


Figure 6.16: Residual block as originally proposed (a) (source: [BKK18]) and as simplified for this work (b)

It is important to note a few more differences in the topology for the upcoming experiments with respect to those in the author's previous work [KWB20]. Here, a NN architecture will inherit a single layer type consistently along its depth, whereas in previous works MLPs were attached to each model as last layer. While for recurrent topologies this just translates to another recurrent connection in the last layer now, for TCNs this means there are no time windows anymore other than the measurement profiles. Hence, so-called MaxPooling layers become redundant as no time dimension has to be reduced to a feature vector to be parsable by a MLP. Instead, a pure TCN is naturally continuously estimating arbitrarily sized profiles with sliding kernels from start to end in time. However, during training, it is still preferrable to resort to shorter sequences to increase the number of updates per epoch, which is further outlined in the following subsection Sec. 6.5.2. Moreover, since recurrent topologies will now also have a recurrent connection in the last layer, initialization of the cell state, that is, especially for GRU the first previous estimate, will denote zero only for non-output layers, and the first ground truth temperature otherwise. This seeding with the first ground truth temperature is similar to how the ODE of LPTNs is initialized. It will be later shown that this is an important detail for high estimation performance.

6.5.2 Mini-batch training scheme over subsequences

Facilitating dynamic models necessitates the adherence to the order of consecutive sensor readings. Other than for static models, observational sensor readings cannot be mixed up arbitrarily anymore before they are fed into a dynamic model. Some degrees of freedom in the data processing remain though, and particular choices can have significant performance implications. The most straightforward strategy would be to feed each measurement profile into a model one after the other and update the ANN weights after some constant amount of observations. However, this is no efficient approach since most automatic differentiation frameworks leverage the linearity of matrix multiplications in ANNs and the growing set of single-instruction-multiple-data (SIMD) operations that contemporary CPUs offer, in order to process observational data in (mini-)batches. Not to mention the high amount of computing cores in special hardware accelerators (e.g., GPUs and TPUs) that can only be capitalized on with large tensors in a batch where the memory copy from CPU RAM to accelerator RAM as overhead loses relevance. Such a batch is not represented by many observations along the time dimension, as these cannot be processed at once, but rather by many different scenarios. In this work, the different scenarios equal different measurement sessions at the test bench.

Thermal modeling usually encompasses only few regressors and component temperatures – up to 11 raw

sensor signals in this work, and up to around 200 features when moving averages are also included. Other research domains where ML is successfully used operate on much larger input dimensions (e.g., each pixel in a picture with tens of thousands of pixels). In order to reach these magnitudes in a batch in thermal modeling, the number of measurement sessions would need to increase substantially. Unfortunately, this is not possible in most cases but the simplest where simulations are used instead because test bench measurements require much resources, see Sec. 5. It is reasonable, therefore, to simply gather all M available measurement sessions into a batch, which is a three-dimensional tensor $\boldsymbol{B} \in \mathbb{R}^{L \times M \times P+Q}$ with L denoting the duration of the longest session in samples.

The next fundamental question, then, is how to deal with the different lengths of the various measurement profiles. The duration of a measurement session can range from a couple of minutes to several hours. In order to gather all measurement profiles into a single batch, post- or prepadding will occur in abundance in the resulting three-dimensional tensor. This will lead to having the dynamic model process plenty of padded samples and eventually biasing the weight updates if estimates on padded samples are included in the cost function. A simple remedy for this bias is to mask estimates on padded samples during the cost function evaluation by, e.g., multiplying these instances in time with zero. Then, the backpropagation algorithm will effectively ignore padded sample estimates. Moreover, post-padding is to be deployed instead of pre-padding in order to avoid biasing the dynamic models before they reach real observed samples. This technique was demonstrated in [KWB19a; KWB20; KWB23], a sketch can be seen in Fig. 6.17.

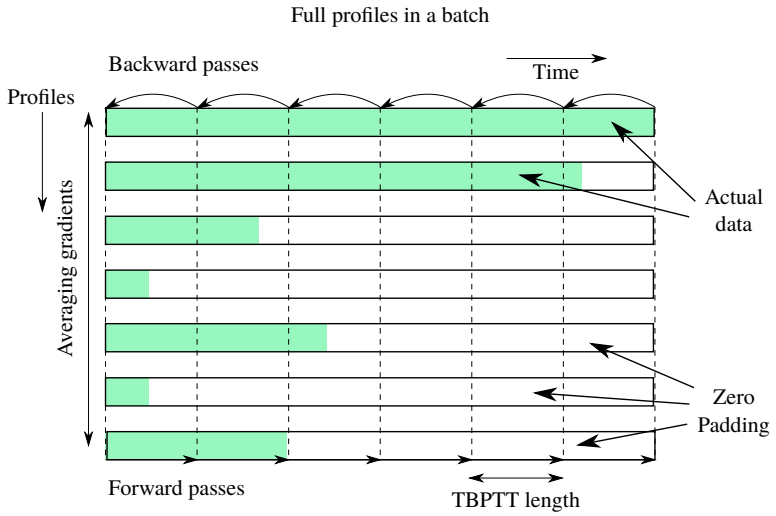


Figure 6.17: Measurement session processing during training with plenty of post-padding (source: [KWB22])

Although batch processing is enabled through this model feeding scheme, a downside that affects weight updates remains. Particularly, having a few especially long profiles will lead to weight updates that are solely based on these profiles in the later part of an epoch, which is prone to distribution shifts. A single profile is most of the time not representative for the full data set, but it would dominate training in case it is of exceptional duration.

In order to overcome also this last disadvantage, splitting all measurement sessions up into subsequences and treating them as independent sessions can be a viable option. If the subsequence length is chosen small enough, most profiles will feature the same duration except for very few that need post-padding to fill up the subsequence length. Such a scheme will avoid updates that depend only on long profiles [KWB22]. What is more, having more smaller measurement profiles yields the possibility to introduce mini-batch training where weight updates are based only on a subset of all sessions. This effectively increases the amount of weight updates per epoch at the cost of a noisier gradient descent direction. At the very least, having a multitude of

measurement sessions to choose from helps to tune the batch size to the most efficient number for the deployed CPU or other hardware accelerator. A sketch of this model feeding scheme is depicted in Fig. 6.18.

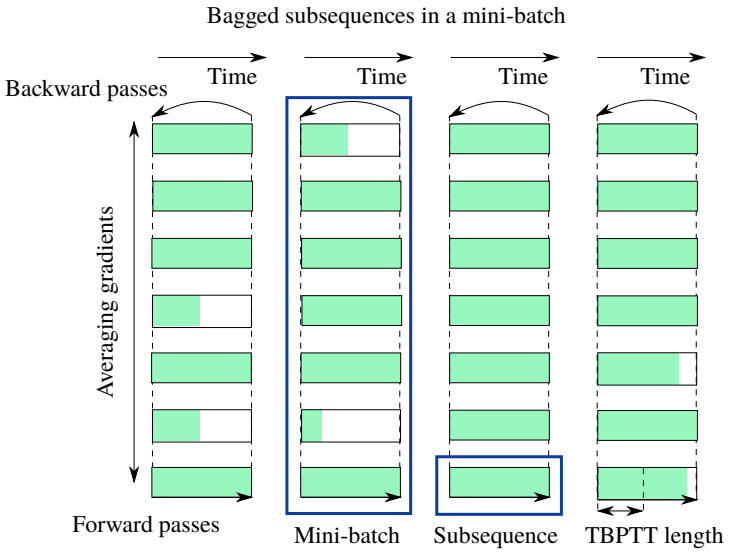


Figure 6.18: Measurement session processing during training with many subsequences (source: [KWB22])

Post-padding of some profiles is not completely avoided but the effect of weight updates depending on only few sessions is minimized especially when incorporating mini-batched training. A possible alternative, which eliminates padded samples completely, would be to copy previously seen samples of a too small subsequence and prepend them until the subsequence is filled up. However, this would lead to an imbalance in the entire data set towards samples that are close to the last subsequence in an original measurement session, whereas the post-padding variant would not yield such data distribution shift.

A potential disadvantage of splitting measurement profiles is that errors will not grow as long any more, which might be helpful for training otherwise. Note that test or generalization sets will not be split up but remain in their original shape in order to avoid unfair advantages for stateful models where the initial condition can be set, and to actually observe whether errors grow beyond acceptable borders during typical operation. Nonetheless, the impact of the trade-off in profile splitting will be apparent during hyperparameter optimizations later in this work.

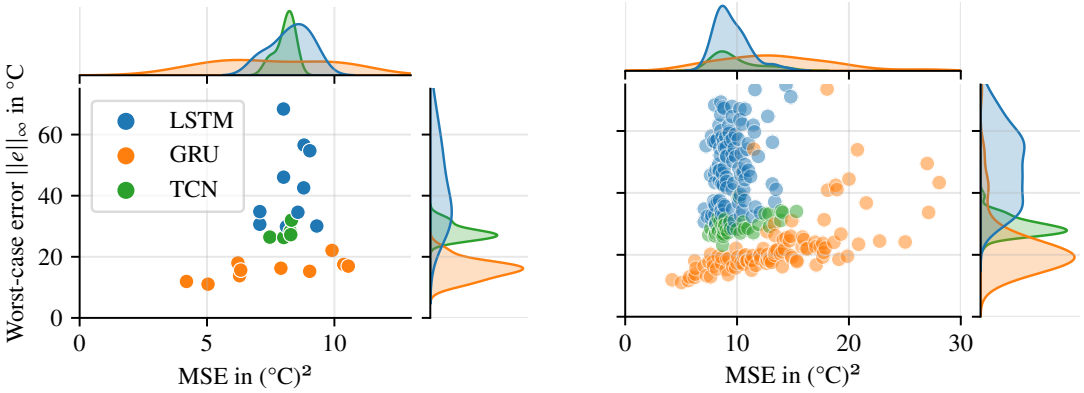
6.5.3 Performance overview

Similar to the overview experiment for static models in Sec. 6.3, a coarse grid search for FE extension schemes is conducted first. Each experiment will be repeated 10 times with different random number generator seeds. In previous works, it was found that EWMAs and EWMSs were still necessary to achieve good results [KWB20], such that they are treated here as well despite employing dynamic models. The normalization scheme is set to the limit normalization due to its previously found advantages. For the overview, all topological hyperparameters are set as follows: one hidden layer with 16 neurons followed by the last layer of four neurons. For TCNs, a layer corresponds to a residual block, which consists of two convolutional layers here.

For all models, a subsequence length (chunk size) of one hour at a TBPTT length of 10 min is utilized, where one epoch consists of four randomly composed mini-batches. While for TCNs, strictly speaking, the standard error backpropagation instead of TBPTT is used, weights will be updated across the same intervals nonetheless, to keep the number of updates comparable between TCNs and RNNs. When moving averages are involved, the same four spans are used as in Sec. 6.3. The TCN will employ dropout with a probability of 20 % for this

	EWMS	False				True	
	EWMA	False		True		True	
	Residual	False	True	False	True	False	True
Model	FE						
LSTM	plain	73.31	35.12	10.25	8.14	9.76	8.59
	basic	65.03	51.12	7.72	7.71	7.88	8.00
	extensive	59.11	73.12	7.23	**7.07**	8.06	7.33
GRU	plain	9.28	91.09	8.19	18.93	13.10	18.82
	basic	5.64	118.25	7.88	20.76	10.07	14.89
	extensive	**4.19**	83.00	7.08	11.50	11.30	18.05
TCN	plain	314.27	96.93	251.13	10.98	271.31	10.28
	basic	307.62	96.16	257.85	8.09	262.06	7.66
	extensive	293.73	94.35	295.37	8.37	252.37	**7.47**

Table 6.5: Grid search over FE configurations with dynamic black-box models (minimum MSE in $(°C)^2$ across seeds)



(a) Best FE configuration

(b) All FE configurations

Figure 6.19: Performance overview as estimated on $\mathcal{G}$

overview. Furthermore, the learning rate denotes $\eta = 1 \cdot 10^{-3}$ and training duration was capped at 160 epochs, during which the learning rate was halved after 80 and 120 epochs each. This learning rate schedule will not change for the remainder of this work.

In Tab. 6.5, the performance overview across the three architectures LSTM, GRU, and TCN is compiled along with a plain, basic, and extensive feature engineered extension, residual connections, and variations on included moving averages. The model sizes are listed in Tab. 6.6 accordingly. The corresponding best configuration across all 10 seeds is shown for all three architectures as scatterplot between the MSE and worst-case error in Fig. 6.19 next to all configurations.

It becomes evident that among the FE extension schemes, the extensive variant outperforms the other two for most configurations. In particular, the best performing configuration for the three architectures is consistently featuring the extensive variant. Interestingly, the addition of EWMSs on top of EWMAs will lead to a slightly worse best performance for the recurrent architectures. In order to save on a substantial amount of model parameters, the exclusion of EWMS features for follow-up experiments is, thus, an obvious decision.

A conclusion can be less clearly drawn for residual connections and the incorporation of EWMAs. While for the LSTM and TCN the best configuration is to be found for both residual connections and EWMAs, for

	Mov. Avgs.	Neither		EWMA only		EWMA and EWMS	
	Residual	False	True	False	True	False	True
Model	FE						
LSTM	plain	1952	2148	3744	4388	5536	6628
	basic	2080	2308	4384	5188	6688	8068
	extensive	2336	2628	5664	6788	8992	10948
GRU	plain	1464	1660	2808	3452	4152	5244
	basic	1560	1788	3288	4092	5016	6396
	extensive	1752	2044	4248	5372	6744	8700
TCN	plain	1424	1620	2768	3412	4112	5204
	basic	1520	1748	3248	4052	4976	6356
	extensive	1712	2004	4208	5332	6704	8660

Table 6.6: Grid search over FE configurations with dynamic black-box models (number of model parameters)

GRU it is the opposite. Even more strikingly, having no residual connections is downright fatal for the mean estimation performance of TCNs. Since both shortcuts and EWMAs lead to more input features, it is very fortunate to find at least the GRU architecture to not rely on them. In fact, the GRU shows the best MSE performance among the three architectures.

It stands out that the GRU is the strongest in terms of the MSE whereas the LSTM is on a par with the weakest although both architectures are recurrent and feature memory blocks with only slight differences between each other. Moreover, the strong GRU performance at almost no additional FE was not found in the author's previous work [KWB20]. Therefore, it can be derived that the crucial improvement here is the omission of a MLP as last layer and initializing the last recurrent layer with the ground truth temperature. Note that the previous time step's GRU memory block estimate traverses fewer transformations before it is added to the current time step's estimate compared to a LSTM block (compare (4.13) with (4.12)).

In terms of number of model parameters, a TCN is the smallest for the given "16 to 4" topology, although it features two convolutional layers per residual block. Slightly more parameters will be found in the GRU, and significantly more in the LSTM, eventually. Note that a MLP, in comparison, still features a better performance at fewer model parameters, namely, ranging from 196 to 1956 parameters, see Tab. 6.3.

However, many degrees of freedom exist in training dynamic NNs, and a hyperparameter optimization (HPO) will give a definitive answer on what approach is superior. The following HPO will exclusively follow the extensive FE scheme, but allows for selecting EWMAs and residual connections optionally. What is more, the HPO will be conducted only for GRU and TCN, ignoring LSTMs from now on, due to the inferior performance, and undesirable complexity and model size, especially in contrast to the similar GRU.

6.5.4 Hyperparameter optimization

Next to the actual parameters of a model, like NN weights, there exists another higher-order layer of parameters that can have a significant impact on model performance. These are called hyperparameters and denote, e.g., the number of neurons per layer, the number of hidden layers, or the learning rate. Hyperparameters are usually not optimized by gradient descent, and most of the time they are configured by the user through trial-and-error. Systematic optimization schemes exist though in the form of, e.g., heuristics such as particle swarm optimization [KE95; PKB07] and genetic algorithms [GS86; AL21] or even random search [BB12]. In this work, an efficient approach through Bayesian optimization with a tree-structured Parzen estimator (TPE) is pursued [Ber+11; Sha+16]. Note that beyond the HPO layer of optimization, there is even another, higher-level

layer that is concerned with, e.g., the choice of the SW frameworks or the HPO interval bounds. This aspect is out of this work's scope.

The TPE algorithm features many advantages over other approaches on HPO: Being based on Bayesian optimization, previously sampled and evaluated hyperparameters can be utilized efficiently to make out promising areas in the search space to sample from next. The goodness of potential candidates is evaluated by a so-called acquisition function, in particular the probability of improvement, which trades off exploration and exploitation. Moreover, in contrast to generation-based heuristics, the TPE algorithm has no concept of a swarm or tribe for in which each member would have to finish its evaluation before the next generation (sampling) can start. The TPE algorithm can rather sample new candidates continuously and with a minimum of delay, which is important for HPOs that run on distributed systems and on an evaluation metric that takes a varying amount of time depending on the sampled set of hyperparameters, in order to avoid idle time. Furthermore, categorical and conditional hyperparameters are modeled appropriately by the TPE's tree structure, which makes the question of, e.g., "how to treat the number of neurons in the second hidden layer if the current sample denotes only a single hidden layer" trivial. Eventually, compared to Bayesian optimization with Gaussian processes as surrogate model, the TPE scales well with large data sets of sampled hyperparameters and their evaluations. The TPE algorithm is further illuminated in the appendix Sec. A.2.

The model family of NNs is characterized by plenty of hyperparameters that are found to be of high importance for a successful convergence of the error backpropagation algorithm for the lower-level NN weights [SLA12; Wat+23]. In this work in particular, the number of layers, the neurons or units per layer, the initial learning rate, the chunk size, the amount of mini-batches per epoch, the TBPTT length, having residual connections or EWMAs, and the optimizer are varied for both GRU and TCN architectures. The optimizer is chosen to be either SGD, RMSprop [TH+12], adaptive moment (ADAM), ADAMAX [KB17], or ADAM with Nesterov Momentum (NADAM) [Doz16], which are all first-order gradient descent optimizers with optional momentum terms and do not incorporate second-order approximations. All hyperparameters, their intervals, and found optima are compiled in Tab. 6.7. Some hyperparameters are specific to GRU or TCN. In contrast, activation functions were not varied, as these are determined by the architecture. Moreover, the spans for EWMAs were not optimized, as these were already found optimal at $[1320, 3360, 6360, 9480]$ samples in a previous work on the same data set [KWB20].

The HPO is guided by training on $\mathcal{T}_\mathrm{r}$ and evaluating on $\mathcal{T}_\mathrm{e}$. Particularly, the score of a certain hyperparameter set is the median MSE on $\mathcal{T}_\mathrm{e}$ across 11 different random number generator seeds. The score on $\mathcal{G}$ will be reported after the HPOs conclude.

From Tab. 6.7 it becomes evident that the best performance was achieved with rather big models, choosing the maximum of layers and mostly many neurons per layer over smaller models. The amount of mini-batches was chosen at the maximum of four, which means that more updates per epoch benefits the gradient descent even at the cost of a more stochastic direction. The chunk-size was not the maximum of 3 hours (considering the sample frequency of 2 Hz) but still rather large, in contrast to the TBPTT length for the GRU, which was found to be optimal at around 18 min. For the TCN, the kernel sizes were always chosen to be the maximum at 7, and the dilatation rate had to be as large as possible with $\delta = 2^2$ right from the beginning of the model. It stands out that the TCN needs both residual connections and EWMAs, whereas the GRU does best without any of the two modifications. Avoiding residual connections in the GRU could be explained by the weakly transformed initial cell state, which is initialized with the first ground truth sample of a trajectory, and which would be only skewed by a residual connection.

The HPO trends, that is, the score improvement along consecutive samples, are shown in Fig. 6.20 and Fig. 6.21 for GRU and TCN, respectively. For the TCN, there were three different HPOs conducted with different topological variations: Two HPOs with a variable TBPTT, similar to GRU, for more weight updates per epoch at the cost of repeated initialization of the TCN kernels in the beginning of a subsequence, while one would have a single TCN layer per residual block (blue evaluations with an orange cumulative best score),

Name	Description	Bounds	Optimum	
			GRU	TCN
n-layers	No. of hidden layers in addition to output layer	$[0, 2]$	2	2
n-units-layer-0	No. of neurons in first hidden layer if available	$[2, 64]$	60	57
n-units-layer-1	No. of neurons in second hidden layer if available	$[2, 64]$	15	60
n-batches	No. of mini-batches per epoch	$[1, 4]$	4	4
initial-lr	Initial learning rate (log)	$[1 \cdot 10^{-5}, 1]$	$3.8 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$
optimizer	Optimization algorithm	[SGD, ADAM, ADAMAX, NADAM, RMSPROP]	NADAM	ADAM
chunk-size	Subsequence length in samples	$[450, 28800]$	23983	19051
TBPTT length	After how many samples in time are weights updated?	$[4,$ chunk-size$]$	2205	-
is-residual	Are residual connections used?	no/yes	no	yes
has-EWMA	Are EWMAs used?	no/yes	no	yes
starting-dilation-rate	Dilation rate exponent in the first layer	$[0, 2]$	-	2
kernel-size-layer-0	Kernel size in first hidden layer if available	$[2, 7]$	-	7
kernel-size-layer-1	Kernel size in second hidden layer if available	$[2, 7]$	-	7
kernel-size-layer-out	Kernel size in output layer	$[2, 7]$	-	7

Table 6.7: Hyperparameter optimization configurations and found optima

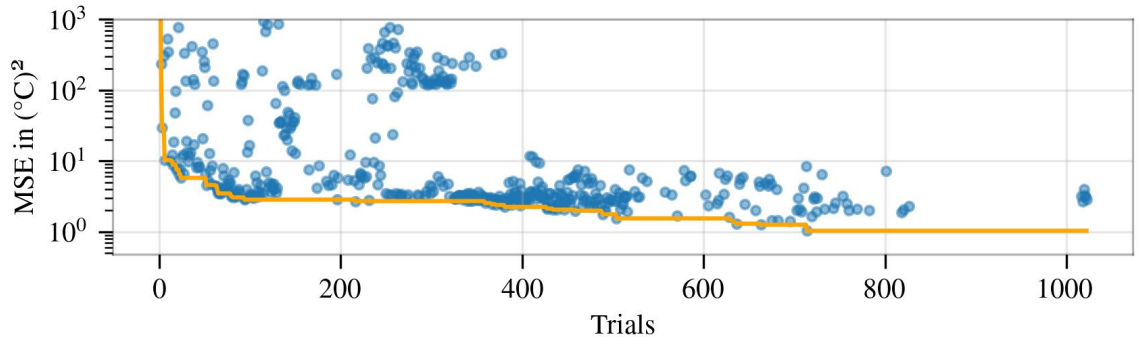


Figure 6.20: HPO trend for GRU (Median MSE across 11 seeds on $\mathcal{T}_e$)

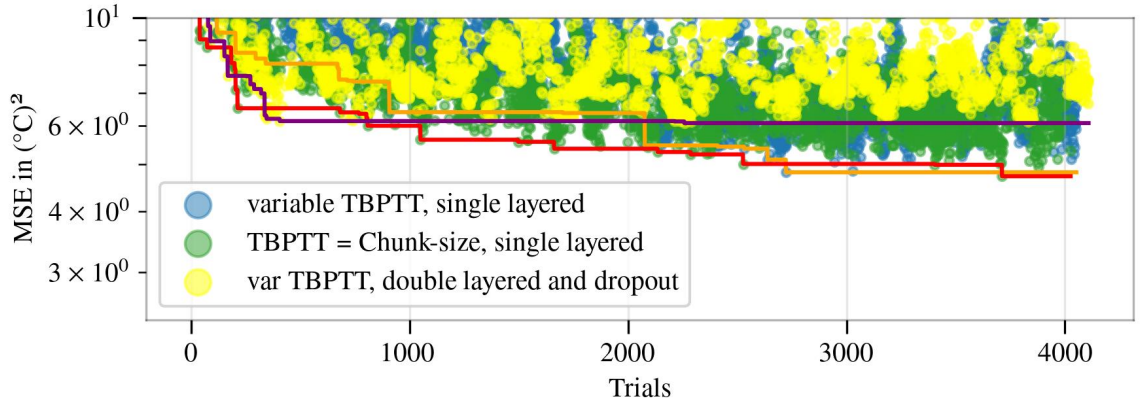


Figure 6.21: HPO trend for TCN (Median MSE across 11 seeds on $\mathcal{T}_e$)

and the other would feature two layers per residual block and dropout in contrast to the other two HPOs (yellow evaluations with purple cumulative best score). The third HPO ignores the TBPTT concept completely and is updated at the end of a chunk (subsequence) with a single layer per residual block and no dropout (green evaluations with a red cumulative best score). Among the three TCN HPOs, the TBPTT-ignoring TCN achieves the best score in the end, and the optimum thereof is shown in Tab. 6.7. This TCN variant is focused on in the following while the other two are ignored. It becomes clear that the TCN architecture cannot reach the performance of the GRU in terms of the MSE, even with three different searches and four times the number of HPO iterations compared to GRU each.

The superiority of the GRU becomes further evident in Tab. 6.8. Here, the top five GRU and TCN models each are selected from the fleet of models trained during the HPO according to the test set MSE also across all seeds. The hyperparameters of these five models do not necessarily correspond to the optimum set as shown in Tab. 6.7, since the optimum set is picked according to the best median MSE across the 11 seeds that each sampled hyperparameter set was evaluated with. This was useful for guiding the HPO in a statistically robust and meaningful manner, but in terms of maximum achievable estimation performance outlying seeds are more insightful. From Tab. 6.8 it becomes evident that the MSE and worst-case error for both models are larger on the generalization set than on the test set, which is reasonable. The hyperparameters were optimized according to $\mathcal{T}_e$ and not $\mathcal{G}$, after all. Particularly, the difference between both sets is higher for the TCN, and also the absolute performance is worse for the TCN compared to the GRU, as had been already indicated previously several times. Through the HPO the performance could be improved for both, and even more significantly for the GRU, but at the cost of more model parameters, compare Tab. 6.8 with Tab. 6.5 and Tab. 6.6. It is interesting that the model size is substantially lower for one of the five GRU models at 4788 parameters, whereas the others start at over 17000 parameters, and for the TCN even just as of 48000 parameters. The estimation performance of that small model is only slightly worse at 2.45 $°C^2$ MSE than for

Metric	MSE in °C^2				$\|\|e\|\|_\infty$ in °C				Model size	
Set	$\mathcal{T}_e$		$\mathcal{G}$		$\mathcal{T}_e$		$\mathcal{G}$			
Model	GRU	TCN	GRU	TCN	GRU	TCN	GRU	TCN	GRU	TCN
Rank										
1	1.03	3.55	**1.11**	6.19	5.29	13.06	6.64	25.66	17217	49716
2	1.15	3.60	1.61	10.01	5.14	11.59	10.01	25.46	18477	54162
3	1.19	3.67	1.15	7.14	6.18	10.52	**5.46**	24.27	24744	48950
4	1.20	3.69	2.45	10.17	5.15	12.00	7.51	29.36	**4788**	59141
5	1.26	3.71	2.06	8.29	6.23	16.85	6.82	46.91	21126	62772

Table 6.8: Top five models according to the test set MSE

the best ranked version at 1.11 °C^2 MSE, but still notably better than of the 1752 parameter GRU from the overview experiment at 4.19 °C^2 MSE on $\mathcal{G}$.

In order to get an insight into the importance of the different optimized hyperparameters, the distribution of all sampled hyperparameters are illustrated in Fig. 6.22 for the GRU, and in Fig. 6.23 for the TCN, that is, the variant with no TBPTT. For GRU, many hyperparameters show a relatively steep trend during their variations with a certain value being obviously at the minimum MSE score. In particular, the chunk-size is clearly best at around 24000 samples, the TBPTT length at around 2500 samples, the initial learning rate (LR) should be between $2 \cdot 10^{-3}$ and $4 \cdot 10^{-3}$, the amount of neurons should be high at over 60 for the first hidden layer while as low as 15 for the second, both hidden layers are clearly better than one or none, the amount of mini-batches is to be maximized, and Nesterov momentum adaptive moment estimation (NADAM) is the best optimizer, closely followed by ADAM and ADAMAX, whereas the plain SGD is clearly the worst, and, eventually, no EWMAs and especially no residual connections performed best. The only hyperparameter that is not showing a very clear trend is the amount of neurons in the second hidden layer, which could be also due to it being sampled less often as it is only relevant for one of three hidden layer configurations. In contrast, for the TCN, the hyperparameter distribution is often plateaued along the hyperparameter value range with respect to the MSE on the test set. Significant trends are evident in favor of employing EWMAs and the LR to be in $[2 \cdot 10^{-4}, 4 \cdot 10^{-3}]$, but not so much for residual connections, which renders the latter a less important hyperparameter or detail in training TCNs. Slight trends can be observed in more layers and more mini-batches being better, and even slighter ones for the amount of neurons per hidden layer, where more is also slightly better, as well as having the chunk size at around 18000 samples, but for the rest of the hyperparameters there is not so much significance evident. Especially the kernel size seems arbitrary for any hidden layer. Any ADAM variant as optimizer is desirable, with the SGD being the worst again.

Among the five models in Tab. 6.8, the GRU and TCN estimation with the lowest worst-case error on $\mathcal{G}$ (that is, rank three) are depicted in Fig. 6.24 and Fig. 6.25, respectively. It becomes obvious that the GRU achieves a desirable low worst-case error among all four target component temperatures with no concerning anomalies. The stateless TCN, however, is troubled to keep the worst-case error even below 20 °C and especially at the beginning of a profile (note the upper black markers) the error tends to spike up. This can be attributed to the use of EWMAs, that are not precise in the beginning of a profile, and also the kernel structure of TCNs, which has not the regular amount of information in the very first estimated samples.

In view of the outlyingly small GRU in Tab. 6.8, the question arises how many model parameters are necessary to achieve as high of an estimation accuracy as possible. In Fig. 6.26, the amount of model parameters for GRU and TCN are rolled out across both the MSE and worst-case error $||e||_\infty$ on $\mathcal{T}_e$ as found during the HPO. It can be seen that the best MSE is achievable by a GRU at around 2000 parameters, whereas a TCN needs at least 3000. More interestingly, though, is the fact that the smallest possible topology is substantially smaller and has comparatively small cuts in performance (on the test set). Among the smallest GRUs with below 5 °C^2

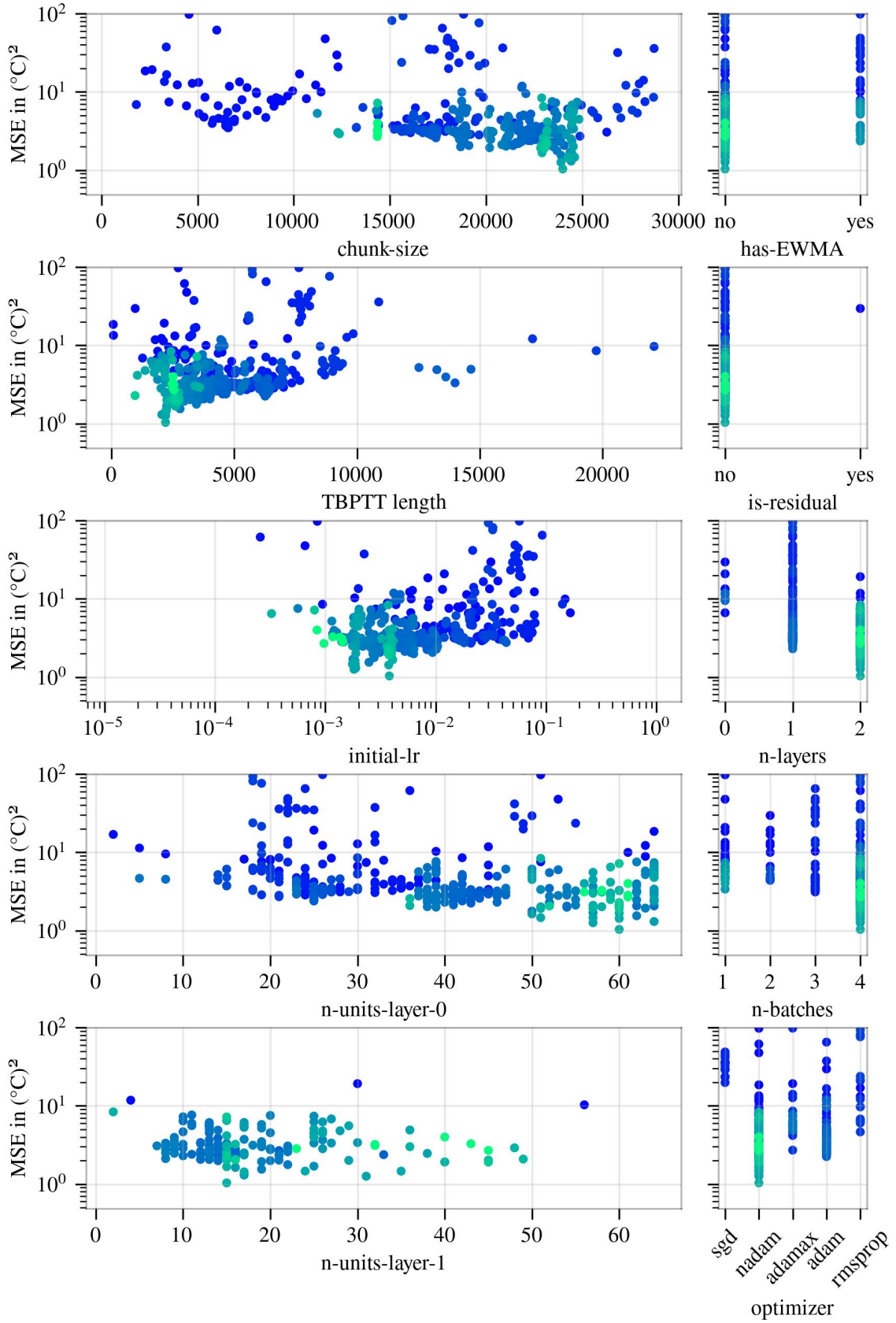


Figure 6.22: Hyperparameter distribution for GRU during HPO (greener samples were more recent, evaluated on the test set)

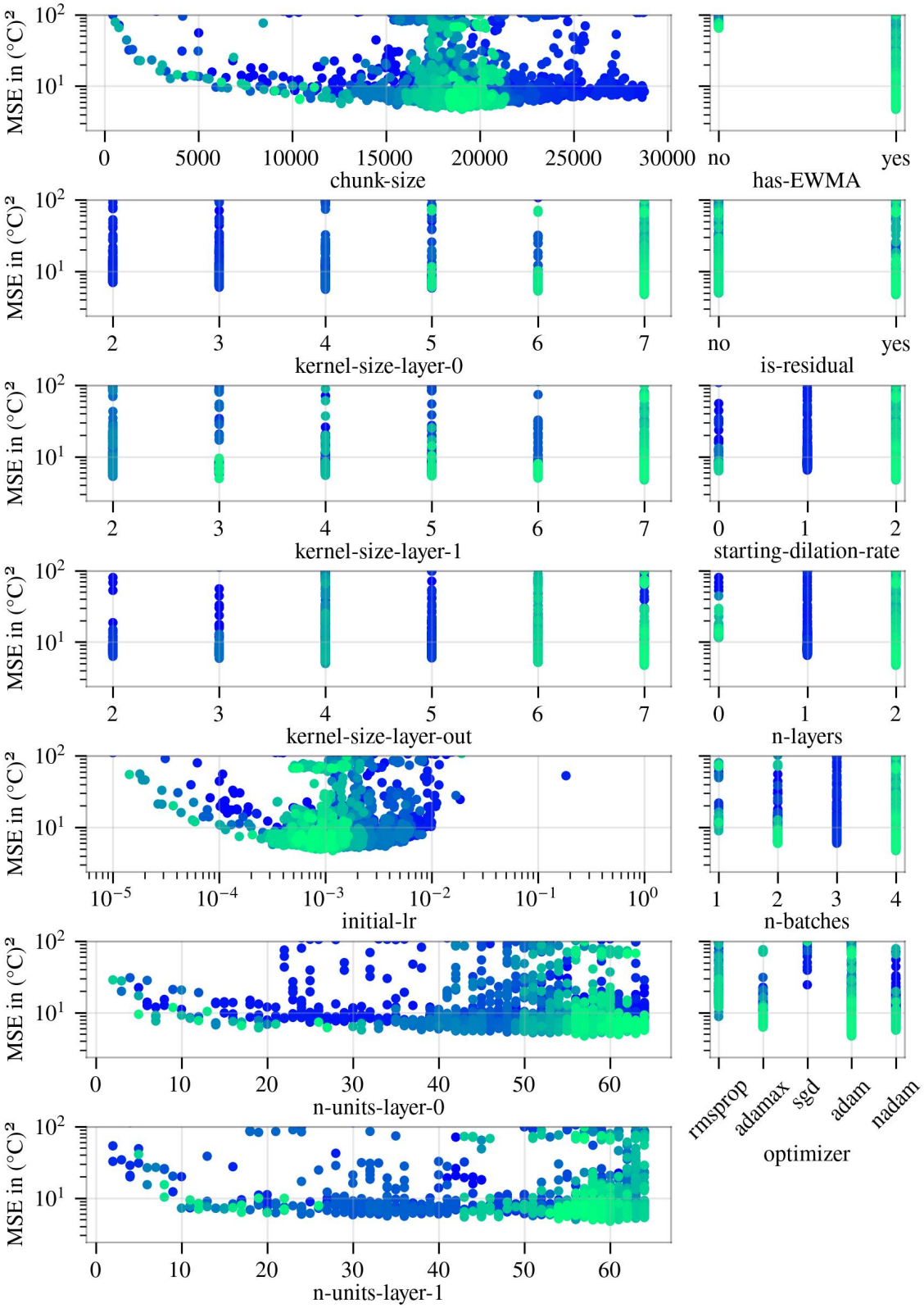


Figure 6.23: Hyperparameter distribution for TCN during HPO (greener samples were more recent, evaluated on the test set)

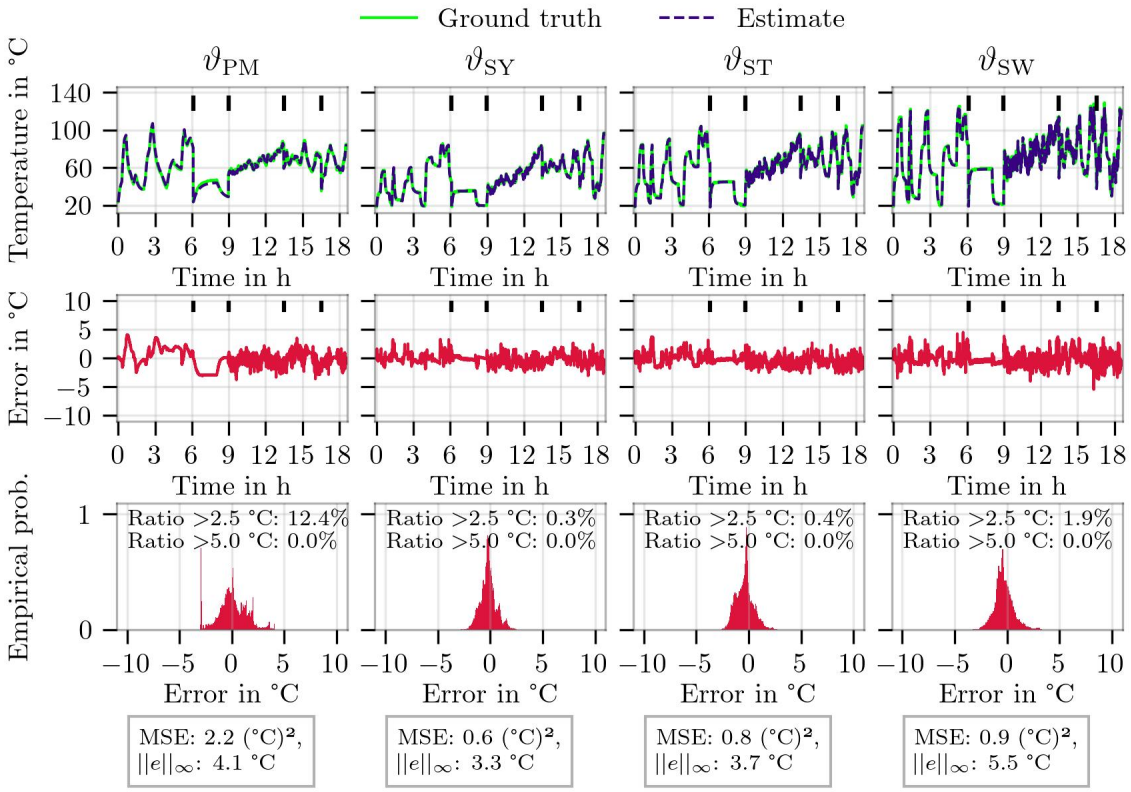


Figure 6.24: GRU hyperparameter optimum estimation performance on the generalization set $\mathcal{G}$

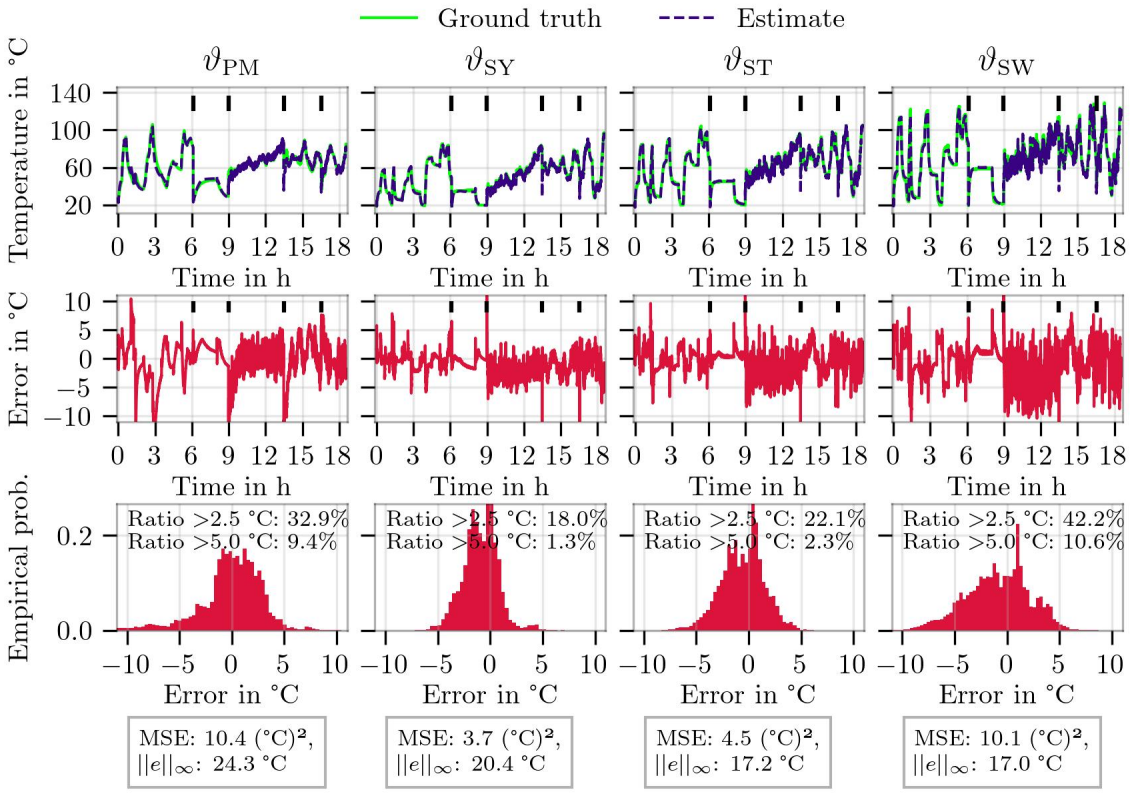


Figure 6.25: TCN hyperparameter optimum estimation performance on the generalization set $\mathcal{G}$

MSE there is a model with 1002 parameters at 3.0 °C^2 MSE and 7.06 °C worst-case error on $\mathcal{T}_e$. Among the smallest TCN with below 10 °C^2 MSE, in turn, there is a model with 1048 parameters at 8.2 °C^2 MSE and 25.34 °C worst-case error on $\mathcal{T}_e$. On the generalization set, of course, these performances vary again, as will be shown in Sec. 6.6.

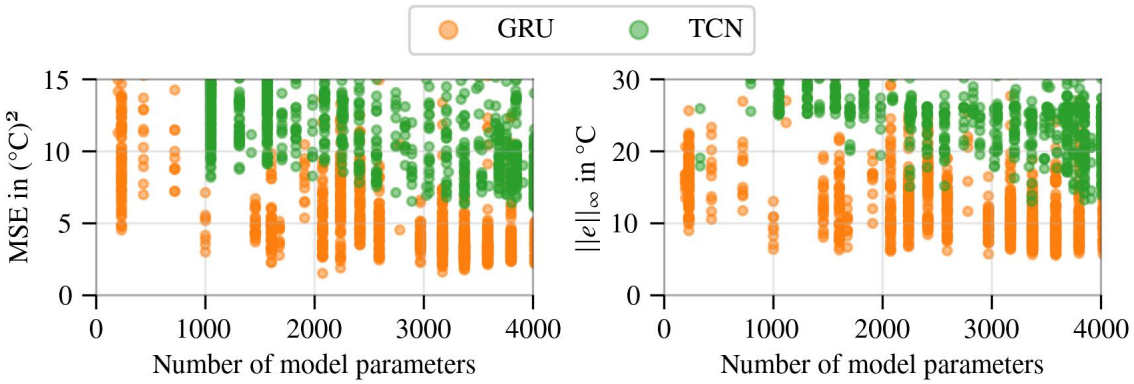


Figure 6.26: Estimation performance of the smallest models found during the HPO of GRU and TCN on the test set $\mathcal{T}_e$

A small GRU topology would take roughly 2.5 h for the training of 11 models in parallel on 11 cores on an AMD Milan 7764 compute node with 2.45 GHz base clock speed, compared to 3.5 h for the optimal GRU with over 24000 parameters on the same machine type. A small TCN, on the other hand, would take around 10 min whereas the optimal structure takes 1 h on, again, the same machine setting.

6.6 Discussion

The performance of the hyperparameter-optimized GRU and TCN is compared with the previously found – yet not hyperparameter-optimized – MLP, LSTM, and LASSO in Fig. 6.27. It becomes apparent that the LSTM and TCN, despite being models tailored for time series estimation, are not able to surpass the static and EWMA-fed MLP. The GRU, however, being a dynamic and recurrent model, outrivals all other models in terms of both, worst-case error and MSE. In terms of model sizes, the LASSO features the fewest model parameters with 472, which could be further eliminated by dropping EWMSs at the cost of some accuracy. The second smallest model is a GRU-variant with 1002 parameters and a similar performance as the best GRU, which is the best black-box model at the same time with over 24000 parameters. The TCN, in contrast, apparently requires up to over 48000 at no noteworthy estimation performance. A static MLP is in the medium region in terms model size and denotes the second best model type in terms of accuracy. Note, however, that the MLP was not hyperparameter-optimized, and smaller variants of the MLP could be likely found in a similar fashion. On the other hand, the non-optimized LASSO is reported here, since optimizations in Sec. 6.4 have led to a better test set score but deteriorated the generalization capability. These findings underline that dynamic black-box models are not to be understood superior compared to static black-box models in the context of real-time thermal modeling without further distinction.

The HPO has further consolidated the findings of the performance overview in Sec. 6.5.3 for that the recurrent GRU topology achieves the highest performance and the TCN among the lowest. The latter is in so far unexpecting that previous work on TCNs found this topology to feature the highest estimation performance when neglecting model size [KWB20]. It is likely that the addition of an MLP layer on top of the TCN with additional MaxPooling layers, as was done in previous work, will make the difference, but this would further increase the amount of model parameters, and is, thus, not going to be investigated further in this work.

The unprecedented performance of the GRU is likely to be attributed to the very precise initialization of the initial cell state to the ground truth temperature, from which on estimates are propagated further on in each measurement profile. This is very similar to how classical, state-of-the-art, expert-knowledge-based LPTN models would be deployed, and it is a feature that is not present in black-box ML models other than the GRU (and in a more skewed version in the LSTM).

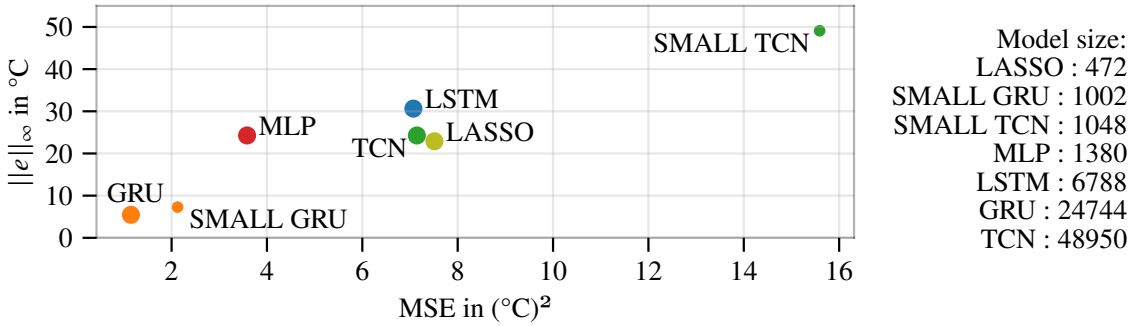


Figure 6.27: Estimation performance on the generalization set $\mathcal{G}$ for the best pure black-box models

Model interpretability

The chain of nonlinear transformations in ANNs makes them notoriously intransparent such that they are generally considered noninterpretable. This is where the name *black-box* model comes from after all. Usually, it cannot be easily discerned which input feature is of high importance and which are not. In some fields, model interpretability plays a vital role, and black-box models are, thus, difficult to employ there despite high predictive accuracies [Lip18].

Considerable work on black-box model interpretability through feature importance evaluation was done with a game-theoretic approach called Shapley additive explanations (SHAP) [LL17]. Particularly, SHAP unifies several approaches to feature importance evaluation, such that its tools range from model-agnostic variants with approximations to model-specific developments with exact and faster computations, e.g., for tree ensembles. However, most of the approximations assume that input feature contributions on the model's output can be linearized locally, and the fundamental building blocks expect static models to evaluate. The theory is not far elaborated for multivariate time series data, not to mention for dynamic systems. For instance, SHAP value calculations require the choice of uninformed reference values or distributions for each input feature. However, this is operation-point-specific at best for dynamic systems, and can make the evaluation of input feature importances only locally valid, in addition to other approximations. Although a reliable metric for input feature importances could help to decide whether some sensor readings can be dropped from modeling, the saving on model parameters would be marginal, as the number of input features do not affect the number of model parameters after the first hidden layer.

A different perspective on reducing model parameters can be obtained when shifting one's view to the classical thermal modeling domain where LPTNs (Sec. 3.2.2) with very few parameters are deployed. These are expert-driven and obey the governing laws of heat transfer. Infusing LPTNs into ANNs will give rise to previously unavailable potential for model parameter reduction and model interpretability, as will be outlined in Sec. 7.3.

Application-tailored cost function

In academic literature, comparability and reproducibility are important pillars of scientific best practices. A common cost function, like the MSE (6.2), is part of what is promoting these virtues. In industry applications, however, the MSE might miss on some key goals of a desirable strong thermal model. Ultimately, a deployed thermal model that is polled in real time shall be as accurate as possible *at high temperatures* in order to have a derating algorithm fully utilize the overload capabilities of the manufactured materials. A high component temperature constitutes a premise for an overload condition after all. Moreover, due to irreversible defects that could be caused by overheat, not only a precise estimate at peak temperatures is of paramount importance, but also an overestimation should be preferred over an underestimation. Both factors – higher accuracy at high temperatures and overestimation over underestimation – are not taken into account by the MSE.

Albeit it is of no importance for the experiments in this work, an alternative weighted cost function shall be presented here for the interested reader. The MSE cost can be weighted on a per-sample basis in order to incorporate the regard for overheating, which will be coined thermally weighted mean squared error (TWMSE), and is described by

$$\mathcal{L}_{\mathrm{TWMSE}} = \frac{1}{KQ} \sum_{q=1}^{Q} \sum_{k=1}^{K} \lambda_{\vartheta,\mathrm{high},q}[k] \lambda_{\vartheta,\mathrm{under},q}[k] \left(\hat{\vartheta}_q[k] - \vartheta_q[k]\right)^2, \tag{6.17}$$

with weighting coefficients

$$\lambda_{\vartheta,\mathrm{high},q}[k] = \left(\frac{\vartheta_q[k]}{\max_k \vartheta_q[k]}\right)^{\alpha_{\vartheta,\mathrm{high}}},$$

$$\lambda_{\vartheta,\mathrm{under},q}[k] = \left\{ \begin{array}{ll} 1 + \alpha_{\vartheta,\mathrm{under}}, & \text{if} \quad \hat{\vartheta}_q[k] < \vartheta_q[k] \\ 1, & \text{otherwise} \end{array} \right\}$$

where $\alpha_{\vartheta,\mathrm{high}}$ and $\alpha_{\vartheta,\mathrm{under}}$ denote hyperparameters that can be tuned according to the application requirements. In Fig. 6.28, two hypothetical profiles are illustrated, one with a relative long duration at low temperatures and the other at high temperatures. Furthermore, four baseline temperature estimators are shown: two naive approaches where the maximum temperature and the average temperature is estimated regardless the true trajectory, as well as two better estimators with one tending to estimate close to the average (conservative) and the other tending to over- and underestimate according to the actual ground truth (extreme). For $\alpha_{\vartheta,\mathrm{high}} = 2$ and $\alpha_{\vartheta,\mathrm{under}} = 1$ the corresponding TWMSE costs are compiled in Tab. 6.9. With this configuration, samples are gradually weighted less the further away the ground truth temperature is from the (arbitrary) maximum temperature, and squared errors on underestimated samples are double-weighted as penalty.

	Avg. temp		Max. temp		Conservative		Extreme	
	prof. 1	prof. 2	prof. 1	prof. 2	prof. 1	prof. 2	prof. 1	prof. 2
MSE in $(°C)^2$	2017.2	2017.2	2097.8	9136.5	80.7	80.7	80.7	80.7
Max. Temp. weighted MSE in $(°C)^2$	2245.4	1536.9	412.3	2331.2	89.8	61.5	89.8	61.5
Max. Temp. weighted MSE with underestimation penalty in $(°C)^2$	2263.8	1549.8	412.3	2331.2	90.6	62.0	88.4	60.7

Table 6.9: Scores on two hypothetical profiles for different baselines

From Tab. 6.9 it can be seen that the conservative and extreme estimator are equivalent in performance on both

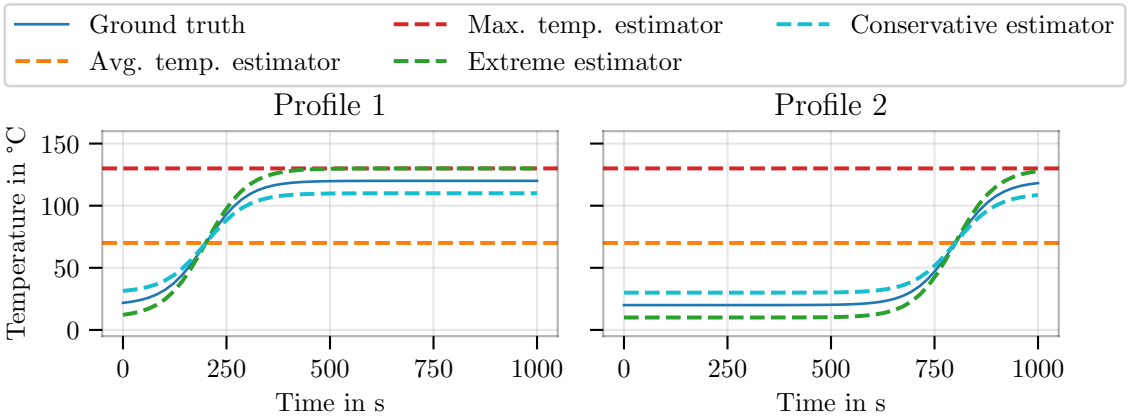


Figure 6.28: Different baseline temperature estimators during two hypothetical profiles

profiles according to the MSE. Yet, in view of the application requirements, the extreme estimator shall be scored better because of its overestimation during high temperatures in contrast to the conservative estimator. In any case, the conservative and extreme estimator should be better than the other two naive baselines. As can be further inferred from Tab. 6.9, these requirements are met for the TWMSE, especially with both weightings applied. A model trained according to the TWMSE instead of the MSE will be less accurate at low temperatures in favor of a high accuracy at high temperatures, and will tend to overestimate instead of underestimating.

7 Gray-box temperature estimation with state-space machine learning

When collected data is utilized to support the parameter identification within differential equations, the realm of gray-box modeling is entered (compare Fig. 3.6). Unlike in black-box modeling, the function search space is usually confined by the choice of governing equations but the amount of approaches to this paradigm are still plenty and with great potential, as will be shown in this chapter.

The classical approach to real-time-capable thermal modeling with lumped-parameter thermal networks (LPTNs) requires expert knowledge for the function design and determination of parameters. High accuracies would require a sophisticated model of the system that considers multiple spatial dimensions and material information in order to align the parameter variation sufficiently (white-box modeling). This fidelity comes at the cost of real-time capability. During the last two decades, research focussed on regaining computational speed by abstracting LPTNs to a lower order, which requires tuning not exclusively by experts but also by measurement data in order to overcome estimation accuracy penalties from oversimplifying the physical system (gray-box modeling). In the recent years – and also in this work – this trend was pushed to the very end of the spectrum with black-box modeling, where in theory no expert is mandatory anymore, since all functional mappings are to be learned from scratch and from data. While the black-box estimation accuracy is comparable to that of classical LPTN approaches, the amount of required model parameters is still relatively large with at least 1000 parameters, as has been shown in the previous chapter.

Arguably, the gist of the success of black-box modeling – or machine learning (ML) in general – is in parts due to improved hardware, but also due to the rise of automatic differentiation (AD) and its software implementations. Contrary to common belief, AD is not bound to the structure of artificial neural networks (ANNs), but rather universally applicable to computer routines or programs, including conditions and loops [Bay+18]. This means that if a LPTN can be written into a program then it can surely be made end-to-end differentiable in order to be optimized by gradient descent and AD. In this chapter, gray-box modeling through the combination of AD and heat transfer principles is investigated.

7.1 Related work

The first attempt to combine ANNs with the structure of a mathematical model goes back to the 90's [Sei96; Dol00], which coined the term structured ANN. By virtue of the early development of ANN training at the time, the practical use of the principles of AD were just gaining momentum as special case in the form of error backpropagation for multilayered NNs. Training of structured NNs seemed difficult and slow even for shallow and small topologies, but it was already better than for standard NNs. This was worked around with, e.g., the delta rule – a special case of error backpropagation where the gradient can be directly attributed to each weight with no dependence on other weights at the cost of utilizing no hidden layers – or with heuristics such as genetic algorithms or other nonlinear optimization algorithms. However, the advantages of incorporating mathematical formulations known for the system at hand were recognized already back then: fewer training data and model parameters necessary, and faster convergence to sufficient estimation accuracy.

A more recent work where a structured ANN is combined with an electrically equivalent circuit model can be found in [And+13] in the context of state of health estimation for lithium-ion automotive traction batteries. Despite its recency, instead of resorting to AD for the training, the model is kept shallow with one input and output layer, and no additional hidden layers, in order to apply gradient descent with the delta rule. While differential equations are in fact combined with ML here, the proposed concept would not be extensible to full multi-layer perceptrons (MLPs) without also incorporating AD. A work explicitly based on structured NNs but not on an electrically equivalent circuit is given in [Gar+07], where it is used to track the saliency in a three-phase electric machine for an improved control aided by carrier-signal injection. They started to harness AD through the Levenberg-Marquardt algorithm [Mor06] in the neural-network-toolbox of MATLAB. A very recent work, also based on a true AD framework, is presented in [Hoc+21], where the so-called central-difference lagrange network is introduced, which learns contact dynamics of solid bodies while preserving laws of conservation of momentum and energy.

It is common among all these works that a recurrent topology enables learning of a difference equation, which may be discretized from a continuous-time differential equation. The attempt to combine standard recurrent neural networks (RNNs) with state-space representations was pioneered in state-space NNs, as described in Sec. 4.4.

Applications of physics-informed neural networks (PINNs) to heat transfer can be found in [ZH21; Li+23], see an introduction in Sec. 4.4.2. There, the heterogeneous heat distribution in a solid with convective boundary conditions is investigated with a contemporary AD framework. It should be reiterated that in spite of their general nomenclature, PINNs are a brand name for approaches where differential equations are additively combined with estimation accuracy loss functions exclusively, see Sec. 4.4.2. That is, no other constraints are applied on the learned ANN, which remains a standard topology, and which is only hoped to be both precise and consistent with a differential equation at the same time, but can equally be neither. This is in contrast to the approaches that will be outlined in this section, which are topologically designed to be consistent with the first principles of a system in the first place.

The conjunction of thermodynamic differential equations with a proper AD framework can be found in [Mas+21], which is introduced as thermodynamics-based NNs. These were developed in parallel to the similar-sounding thermal neural networks (TNNs), which will be presented later in this work. Unlike TNNs, thermodynamics-based NNs are applicable to constitutive modeling, which restricts itself to thermodynamically closed systems (no heat transfer).

The TNN was published by the author of this work, and its preprint was the first public contribution where ANNs were merged with the LPTN difference equation for heat transfer applications [KWB21c]. Another early work on combining ANNs with a low-order LPTN achieved very promising results in terms of estimation accuracy and model size [GWB21]. There, some expert knowledge was employed when determining thermal parameter value boundaries as well as when sensor readings were manually selected as input features for each thermal parameter according to a-priori contemplations of the system. Moreover, optimization through a nonlinear programming solver was deployed instead of by means of AD and gradient descent, which constrained the topology to below 100 parameters due to growingly restrictive training runtimes. The restriction to few parameters is not problematic in itself because that implies the overall objective, however, also for few parameters the retrieval of gradients by AD instead of finite differences can yield shorter training times and, hence, a more fast-paced experimental iteration.

Not much later, another work where the LPTN difference equation is combined with ANNs is denoted by [Di +22] in the context of heat transfer within buildings. They utilize PyTorch and an encoder-LSTM-decoder architecture, but they acknowledge that other topologies were also possible. Similar properties as for a TNN were found.

A very similar paper to the TNN works is denoted by [GHN23]. There, a TNN-like structure with only one node is deployed on the temperature estimation task of an electric motor's rotor. With a tweak on the power loss sub-ANN, they derive the property of input-to-state stability.

7.2 Learning an explicit-Euler-discretized nonlinear function

A first, and rather soft, incorporation of a priori knowledge into the modeling approach is to entail the explicit-Euler-discretized difference equation on (4.24) with

$$\hat{\boldsymbol{\vartheta}}[k+1] = \hat{\boldsymbol{\vartheta}}[k] + T_\mathrm{s} \boldsymbol{f}_{\boldsymbol{\theta}}(\hat{\boldsymbol{\vartheta}}[k], \boldsymbol{u}_{\text{extensive}}[k]), \quad \hat{\boldsymbol{\vartheta}}[0] = \boldsymbol{\vartheta}[0], \tag{7.1}$$

where $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ is an MLP with parameters $\boldsymbol{\theta}$ according to (4.10) and $T_\mathrm{s} = 0.5\,\mathrm{s}$. Note that (7.1) is neither a standard MLP due to the necessity of the last estimate for the current estimate, nor a standard RNN, since the last estimate comes as a nonweighted term additionally. This algorithm will be trained by TBPTT instead of the adjoint sensitivity method (see Sec. 4.4.3). Equipped with 32 neurons in a single hidden layer, the resulting nonlinear, recurrent, and slightly informed model is trained similarly to the previous experiments with 160 epochs and a piece-wise reduced learning rate on $\mathcal{T}_\mathrm{r}$. The chunksize was 14400 samples (2 h) and the TBPTT length was 10 min at 4 mini-batches with an initial learning rate of $1 \cdot 10^{-3}$. The generalization performance is displayed in Fig. 7.1.

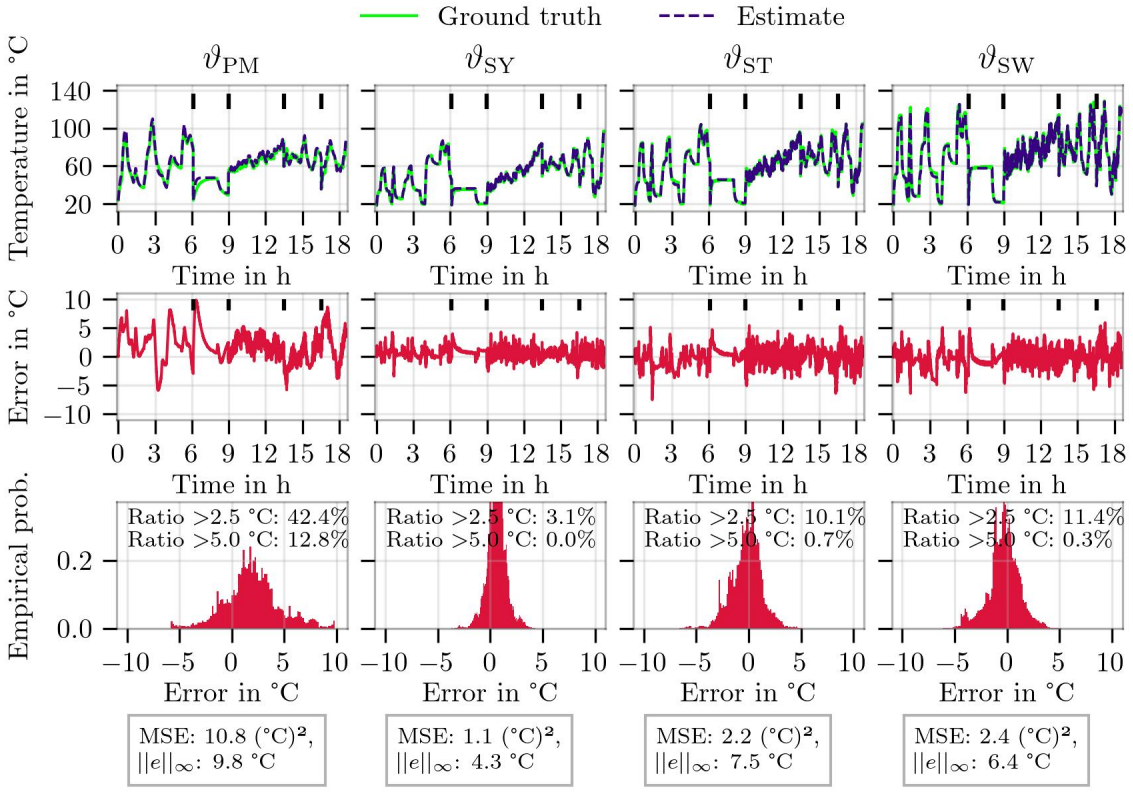


Figure 7.1: A lightly informed nonlinear recurrent model

It becomes evident that the performance offers room for improvement but is not too far off for a very dark gray-box model with 708 parameters. Nonetheless, compared to a black-box gated recurrent unit (GRU) (as shown in Fig. 6.24), this gray-box model's estimation accuracy is inferior. Furthermore, more domain

knowledge in the form of topological structure needs to be incorporated in order to interpret the model's states or parameters physically.

7.3 Thermal neural networks

Up to this point, ANNs (Sec. 4.3.2) and LPTNs (Sec. 3.2.2) were viewed as disjunct model classes. While the black-box nature of ANNs offers the convenience of designing data-driven thermal models without geometry nor material knowledge, they require substantially more model parameters than the LPTN counterpart, and are noninterpretable. In an endeavor to leverage the benefits of both modeling paradigms, the author proposed to embed ANNs into the system of differential equations of an LPTN, and coined it (lumped-parameter) thermal neural networks (TNNs) [KWB22; Kir+22; KWB23][1]. The motivation behind such a merged construct is the fact that the thermal parameters in a LPTN (3.32) are seldomly constant and rather describe nonlinear, operation-point-dependent functions. Especially convective and radiative heat transfer inherit highly nonlinear thermal characteristics making the transient definition of thermal parameters a challenging endeavor [Bog+06; HCH12]. At the same time, ANNs are known to be universal function approximators. Hence, it seems natural to employ (multiple) ANNs as estimators not for temperatures but rather thermal parameters. The general applicability of AD to arbitrary programs is the reason why such a constellation can be end-to-end differentiable although true values are only known for temperature values rather than for the thermal parameters.

The following architecture leverages the strong nonlinear modeling capacity of ANNs and circumvents the intricate and error-prone design effort that accompanies the identification of a system's thermal properties' functional dependence on the system operation point. The causal, statistical relationship between temperatures and other readily available sensor information are learned from empirical data without geometry or material information, similar to ANNs, but under the constraint of the LPTN system of ODEs. A TNN reformulates (3.32) after a first-order Euler discretization to

$$\hat{\vartheta}_q[k+1] = \hat{\vartheta}_q[k] + T_\mathrm{s}\kappa_q[k]\left(\pi_q[k] + \sum_{j\in\mathcal{M}\setminus q}(\hat{\vartheta}_j[k] - \hat{\vartheta}_q[k])\gamma_{q,j}[k] + \sum_{j=1}^{n}(\tilde{\vartheta}_j[k] - \hat{\vartheta}_q[k])\gamma_{q,j}[k]\right), \tag{7.2}$$

with $\hat{\vartheta}_q[k]$ denoting the q-th node's normalized temperature estimate at discrete time k, T_s being the sample time, $\mathcal{M} = \{1, 2, \ldots, Q\}$, and κ_q, π_q as well as $\gamma_{q,j}$ denoting arbitrary feed-forward ANN outputs dependent on $\boldsymbol{\zeta}[k] = [\tilde{\boldsymbol{\vartheta}}^\mathsf{T} \quad \hat{\boldsymbol{\vartheta}}^\mathsf{T} \quad \boldsymbol{\xi}^\mathsf{T}]^\mathsf{T}$, which in turn consists of the n ancillary temperatures $\tilde{\boldsymbol{\vartheta}}[k]$, the temperature estimates $\hat{\boldsymbol{\vartheta}}[k]$, and additional observables $\boldsymbol{\xi}[k] \in \mathbb{R}^{P-n}$ [KWB23]. Note that the ancillary temperatures and the additional observables denote the input vector $\boldsymbol{u}[k] = [\tilde{\boldsymbol{\vartheta}}[k]^\mathsf{T} \quad \boldsymbol{\xi}[k]^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^P$. Particularly, $\boldsymbol{\gamma}(\boldsymbol{\zeta})$ approximates all thermal conductances between temperature nodes, $\boldsymbol{\pi}(\boldsymbol{\zeta})$ all power losses at these nodes, and $\boldsymbol{\kappa}(\boldsymbol{\zeta})$ determines all inverse thermal capacitances.

For a matrix notation of (7.2), let $\mathbb{1}_{[z]} = [1, 1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^z$ and $\hat{\boldsymbol{\vartheta}}_\mathrm{ext} = [\hat{\boldsymbol{\vartheta}}^\mathsf{T} \quad \tilde{\boldsymbol{\vartheta}}^\mathsf{T}]^\mathsf{T}$, then

$$\hat{\boldsymbol{\vartheta}}[k+1] = \hat{\boldsymbol{\vartheta}}[k] + T_\mathrm{s}\boldsymbol{\kappa}[k] \odot \left(\boldsymbol{\pi}[k] + \left(\left(\mathbb{1}_{[Q]} \cdot \hat{\boldsymbol{\vartheta}}_\mathrm{ext}[k]^\mathsf{T} - \hat{\boldsymbol{\vartheta}}[k] \cdot \mathbb{1}_{[Q+n]}^\mathsf{T}\right) \odot \boldsymbol{G}[k]_{\{1\ldots Q\}}\right) \cdot \mathbb{1}_{[Q+n]}\right), \tag{7.3}$$

[1] Some passages in Sec. 7.3 and Sec. 7.4 may be copied from these works, yet all experiments are genuine and newly conducted for this dissertation for the sake of consistency except otherwise noted.

with $\odot$ denoting element-wise multiplication, and $\boldsymbol{G}$ denoting the symmetric thermal conductance adjacency matrix, where $\boldsymbol{G}[k]_{\{1\dots Q\}}$ denotes the first Q rows of $\boldsymbol{G}$ at time k, with

$$\boldsymbol{G} = \begin{bmatrix} 0 & \gamma_{1,2} & \gamma_{1,3} & \cdots & \gamma_{1,Q+n} \\ \gamma_{1,2} & 0 & \gamma_{2,3} & \cdots & \gamma_{2,Q+n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_{1,Q+n-1} & \gamma_{2,Q+n-1} & \gamma_{3,Q+n-1} & \cdots & \gamma_{Q+n-1,Q+n} \\ \gamma_{1,Q+n} & \gamma_{2,Q+n} & \gamma_{3,Q+n} & \cdots & 0 \end{bmatrix}.$$

In (7.3), two dyadic products form a difference, followed by a horizontal tensor contraction. All variables can be expanded by another batch dimension, as is typical for AD frameworks, in order to average gradients across multiple scenarios or measurement profiles.

Without loss of generality, $\boldsymbol{\kappa}(\boldsymbol{\zeta})$ can usually be reduced to just end-to-end trainable constants $\boldsymbol{\theta}_\mathrm{c}$ in the form

$$\boldsymbol{\kappa} = e^{\boldsymbol{\theta}_\mathrm{c}}, \quad \boldsymbol{\theta}_\mathrm{c} \in \mathbb{R}^Q,$$

whereas

$$\begin{aligned} \boldsymbol{\pi} &: \mathbb{R}^{P+Q} \to \mathbb{R}^Q, \\ \boldsymbol{\gamma} &: \mathbb{R}^{P+Q} \to \mathbb{R}^{(Q+n)(Q+n-1)/2} \end{aligned}$$

are distinct ANNs. This reduction corresponds to a simplification through the lumped capacitance method, which is feasible in many engineering applications [Ber+07]. The TNN concept is sketched in Fig. 7.2. A completely uninformed TNN assumes a thermal conductance between each temperature node pair, which can be obviously optimized when geometry information is available.

Hence, except for $\boldsymbol{\kappa}$, the general algorithmic shape is described by the MLP algorithm (4.10) fed by $\hat{\boldsymbol{\vartheta}}$ and $\boldsymbol{u}$:

$$\begin{aligned} \boldsymbol{h}^{(0)}[k] &= \boldsymbol{g}^{(0)}(\boldsymbol{W}_\mathrm{r}\hat{\boldsymbol{\vartheta}}[k-1] + \boldsymbol{W}_\mathrm{h}^{(0)}\boldsymbol{u}[k] + \boldsymbol{w}_0^{(0)}), \\ \boldsymbol{h}^{(l)}[k] &= \boldsymbol{g}^{(l)}(\boldsymbol{W}_\mathrm{h}^{(l)}\boldsymbol{h}^{(l-1)}[k] + \boldsymbol{w}_0^{(l)}), \quad \forall\, l > 0, \end{aligned} \tag{7.4}$$

with $\boldsymbol{g}^{(l)}(\cdot)$ denoting the nonlinear, non-negative activation function at layer l, and

$$\boldsymbol{\theta} = \{\boldsymbol{W}_\mathrm{r}, \boldsymbol{W}_\mathrm{h}^{(l)}, \boldsymbol{w}_0^{(l)} : l \in [0, L-1]\} \tag{7.5}$$

describing the trainable parameters that exist independently for both, $\boldsymbol{\pi}$ and $\boldsymbol{\gamma}$. Note that if one finds the assumptions under the lumped capacitance method to be too simplistic for the task, lifting $\boldsymbol{\kappa}$ to the MLP form would be an appropriate first measure. It is noteworthy that the non-negative output of $(\boldsymbol{\kappa}, \boldsymbol{\pi}, \boldsymbol{\gamma})$ is important in order to ensure input-to-state stability (see Sec. 7.3.3). This can be assured in case of $\boldsymbol{\kappa}$ by the exponential function, and in case of $\boldsymbol{\pi}$ and $\boldsymbol{\gamma}$ by the choice of the last layer's activation function $\boldsymbol{g}^{(L-1)}(\cdot)$. The exponential function for $\boldsymbol{\kappa}$ plays another role next to the non-negative output: gradients with respect to $\boldsymbol{\theta}_c$ will enable the gradient descent algorithm to traverse more orders of magnitude in fewer iterations. This is important since all elements of $\boldsymbol{\theta}_c$ are initialized equally but true values in the sense of a high modeling accuracy will differ by several orders of magnitude.

The TNN inherits recurrent connections like a RNN, but its cell consists of three (arbitrarily complex) ANNs. The TNN's state is represented by a vector containing the estimated temperatures. In contrast to standard RNN topologies, a TNN's topology can be varied through the three sub-NNs, but it is not meant to be stacked as a whole onto several layers, as this would foil the resemblance to the ODE structure. In the given application particularly, $\hat{\boldsymbol{\vartheta}} = [\hat{\vartheta}_\mathrm{PM} \quad \hat{\vartheta}_\mathrm{SW} \quad \hat{\vartheta}_\mathrm{ST} \quad \hat{\vartheta}_\mathrm{SY}]^\mathsf{T}$ and $\tilde{\boldsymbol{\vartheta}} = [\vartheta_\mathrm{A} \quad \vartheta_\mathrm{C}]^\mathsf{T}$, thus, $Q = 4$ and $n = 2$.

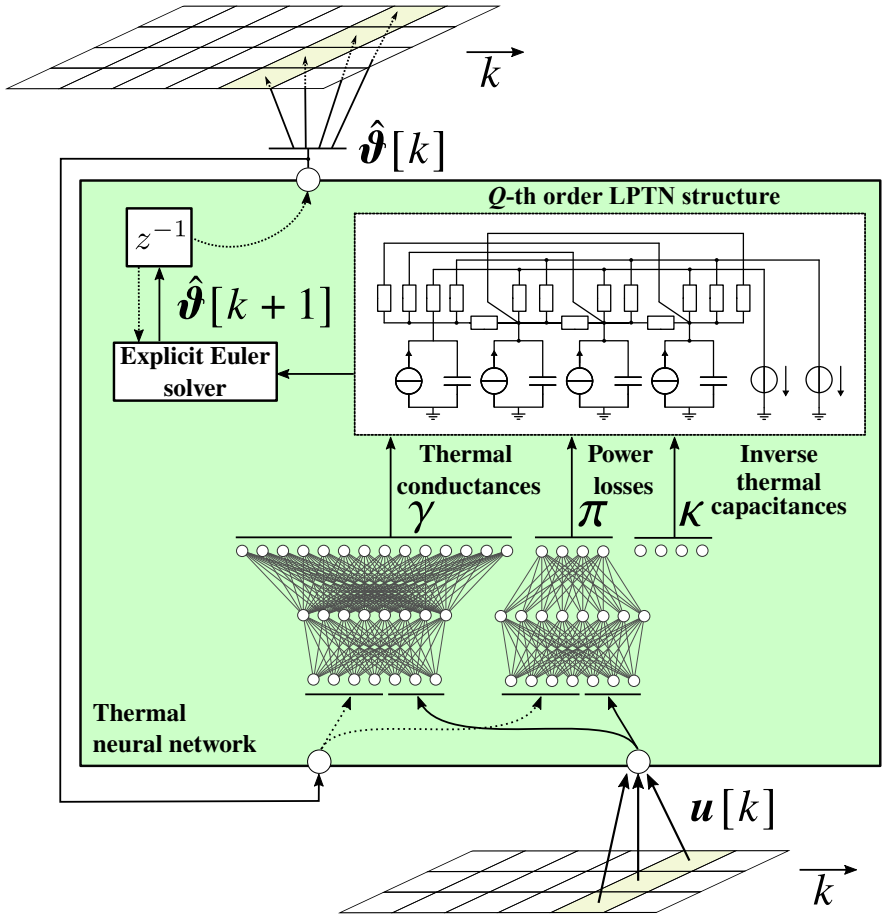


Figure 7.2: A TNN considers all input features of the current time step $\boldsymbol{u}[k]$ and its last estimate $\hat{\boldsymbol{\vartheta}}[k]$ to estimate the next time step's temperatures $\boldsymbol{\vartheta}[k+1]$. This estimate acts as cell information within the TNN to further inform the three function approximators $\boldsymbol{\gamma}$, $\boldsymbol{\kappa}$, and $\boldsymbol{\pi}$. An LPTN structure of fourth order with two ancillary thermal sources is exemplarily depicted, and $\boldsymbol{\kappa}$ denoting mere trainable constants (derived from: [KWB23]).

Compared to the classic LPTN state-space system of equations (3.33), the TNN estimates power losses, too, and does not require their a priori measurement through a power analyzer. Note, however, that also not all LPTNs require the total power loss information for their design.

It is well known that training RNNs with the error backpropagation method suffers from the vanishing and exploding gradient problem [PMB13]. It describes the phenomenon of gradients becoming very large or close to zero the longer the sequence on which gradients are accumulated for a weight update. This is especially severe for arbitrary long sequences on which thermal models conduct estimates. Early on in machine learning literature, this problem was curbed with the introduction of memory blocks (LSTM and GRU, see Sec. 4.3.2). However, the general-purpose topology of an LSTM/GRU is in conflict with the physically motivated TNN, such that a TNN has to overcome this hurdle with other methods, e.g., gradient normalization, clipping, and truncated backpropagation through time (TBPTT) [WP90].

Since no information about the geometry or material of the system is assumed to be known a priori, a fully-connected LPTN denotes the starting point of the TNN design, i.e., $\boldsymbol{G}$ is a dense, symmetric matrix with zeros on the main diagonal. For a thermal model that is modeling $Q+n$ different temperatures, this translates to $(Q+n)(Q+n-1)/2$ thermal conductances and, thus, a parameter size complexity of $O((Q+n)^2)$. In practice, however, several components within a system are often sufficiently detached such that the heat transfer

between them is negligible. Featuring physically interpretable states, a fitted TNN can evidence such prunable connections, that can be removed in a subsequent TNN design iteration. This parameter reduction method is presented in Sec. 7.3.2.

If domain knowledge is available and shall be utilized, this can be incorporated in several ways: for instance, by means of constraining the choice of topology defining parameters like the activation functions of the output layers of $\boldsymbol{\gamma}$, $\boldsymbol{\kappa}$, and $\boldsymbol{\pi}$; by means of feature engineering; or by eliminating thermal parameters a priori. Without loss of generality, it can be recommended to apply the ℓ_1-norm on all the ANNs' outputs, as negative thermal conductances, capacitances, and power losses are physically implausible. More importantly, this promotes the recovery to the ground truth as will be discussed in Sec. 7.3.3.

From (7.2) it becomes obvious, that ancillary temperatures play a distinguished role among the input features in $\boldsymbol{u}$. They are treated differently in the system of ODEs in so far that they have a well-known linear impact on heat transfer between all targeted components. Hence, the conclusion can be already drawn that such sensory information might play a pivotal role. However, the lack thereof is not performance-critical upfront. It has become evident in miscellaneous experiments that the incorporation of at maximum two dummy ancillary temperatures, e.g., a relative cold and hot constant temperature, tend to expand the TNN's degrees of modeling freedom substantially, especially when no other ancillary temperatures are available. Although these dummy temperatures do not carry useful information, a TNN can leverage additional varying thermal conductances connected to these constant sources/sinks in order to model otherwise latent heat transfer.

As is usual for state-space models, the TNN structure asks for an initial temperature estimate. In simulations and training, these initial values can be set with the ground truth values, while in a field application the ambient temperature is a good approximation. Note that setting an initial value is mostly not possible for black-box models that have completely abstract parameter structures, excluding recurrent last layers like a GRU layer, where the last estimate is transformed only slightly. The ability to set the initial condition cleanly is to be seen as a major advantage in favor of models representing a state space. The severeness of a biased initial guess is demonstrated in Sec. 7.3.3. In the following, the performance of a TNN and its minimal realizations are investigated.

7.3.1 Hyperparameter optimization

Similar to the hyperparameter optimization (HPO) for black-box models in Sec. 6.5.4, a Bayesian optimization with sampling according to a tree-structured Parzen estimator (TPE) is conducted for the TNN architecture. The following remained the same: the cross-validation strategy as described in Sec. 6.2; the amount of epochs at 160 with learning rate halving after 80 and 120 epochs; the range for the initial learning rate; the selection of optimizers; and the topological variations of the number of neurons and layers. However, since the TNN itself is to be considered one-layered, the sub-NNs ($\boldsymbol{\gamma}, \boldsymbol{\pi}$) are varied instead. Additionally, the mean of the Gaussian distribution from which the inverse thermal capacitance exponents $\boldsymbol{\theta}_c$ are initially sampled is included in the HPO. Moreover, while the FE extension scheme is varied, the EWMAs or EWMSs are not considered anymore. Residual connections are also not included anymore because the explicit Euler formulation of the system of ODEs assumes this role. Similar to the GRU HPO (Tab. 6.7), the chunk size, the amount of mini-batches per epoch, the TBPTT length, and the optimizer algorithm are optimized. Eventually, the activation function for each sub-NN and each layer is optimized by selecting from a range of candidate functions as shown in Fig. 7.3. The function palette contains the sigmoid, the tanh, the rectified linear unit (ReLU) [NH10], a biased exponential linear unit (BELU), the sine [Sit+20], and the growing cosine unit (GCU) [Noe+23]. A TNN is not defined to have certain activation functions except for the non-negative output constraint, in contrast to a GRU or LSTM. Among the activation function candidates, the biased ELU is atypical and a non-negative

variant of the exponential linear unit (ELU) [CUH15] with

$$\mathbf{BELU}(\boldsymbol{u}) = \mathbf{ELU}(\boldsymbol{u}) + 1. \tag{7.6}$$

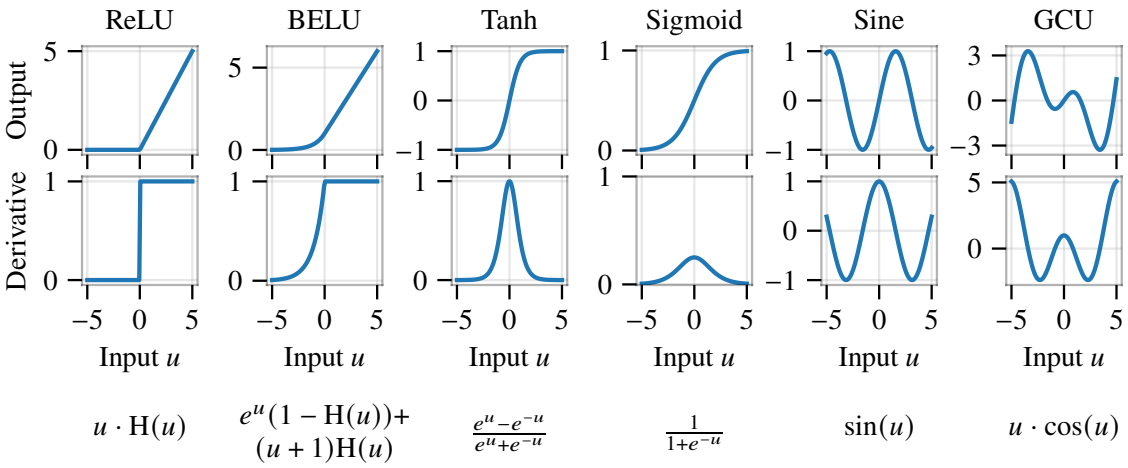


Figure 7.3: Different activation functions – note that $\mathrm{H}(\cdot)$ is the Heaviside function.

The HPO is, again, guided by training on $\mathcal{T}_\mathrm{r}$ and evaluating on $\mathcal{T}_\mathrm{e}$. As previously, the score of a certain hyperparameter set is the median MSE on $\mathcal{T}_\mathrm{e}$ across 11 different random number generator seeds. All hyperparameters, their intervals, and found optima are compiled in Tab. 7.1. The optimization trend and the hyperparameter distribution are illustrated in Fig. 7.4 and Fig. 7.5, respectively.

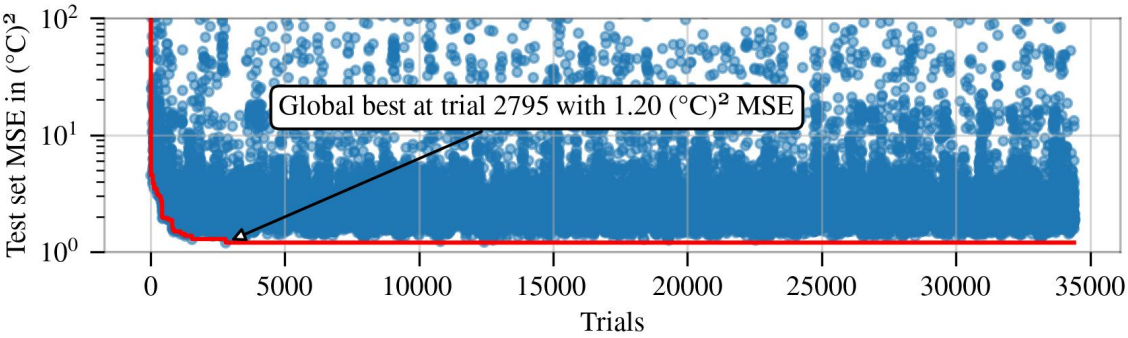


Figure 7.4: HPO trend for TNN (Median MSE across 11 seeds on $\mathcal{T}_\mathrm{e}$)

From the HPO cost trend in Fig. 7.4 a quick convergence is recognizable, which hits an optimum at 1.2 °C^2 median MSE across 11 seeds on $\mathcal{T}_\mathrm{e}$ after 2795 trials already. No further improvements were achieved even after near 35000 trials. The optimal hyperparameters, as compiled in Tab. 7.1, show that peak performance was achieved with just one hidden layer per sub-NN and 62 and 18 neurons for $\boldsymbol{\gamma}$ and $\boldsymbol{\pi}$, respectively. The optimal activation functions vary across output and hidden layers as well as sub-NNs. Again, four mini-batches performed best with a learning rate at $6.72 \cdot 10^{-3}$ which is in the same order of magnitude as for the previous two HPOs. In contrast to the HPO GRU, however, the TNN optimum has a TBPTT at 443 with a chunk-size of 49083 samples, and the RMSPROP optimizer could outrival the others. Moreover, the basic feature engineering extension was better than the other two schemes. Among the Gaussian means for the initial inverse thermal capacitance exponents different orders of magnitude become evident as expected, but not too close to the chosen boundaries.

In order to obtain some context for this optimum, a look at the hyperparameter distribution across all trials

Name	Description	Intervals	Optimum
n-{g,p}-layers	No. hidden layers in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ in addition to output layer	$\{0, 1, 2, 3\}$	{1,1}
n-units-{g,p}-layer-0	No. neurons in first hidden layer in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ if available	$\{2, 3, \ldots, 64\}$	{62, 18}
n-units-{g,p}-layer-1	No. neurons in second hidden layer in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ if available	$\{2, 3, \ldots, 64\}$	{none, none}
n-units-{g,p}-layer-2	No. neurons in third hidden layer in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ if available	$\{2, 3, \ldots, 64\}$	{none, none}
n-batches	No. mini-batches per epoch	$\{1, 2, 3, 4\}$	4
initial-lr	Initial learning rate (log)	$[1 \cdot 10^{-5}, 1]$	$6.72 \cdot 10^{-3}$
optimizer	Optimization algorithm	5 choices*	RMSPROP
chunk-size	Subsequence length in samples	$\{450, \ldots, 28800\}$	4983
TBPTT length	Time steps between weight updates	[4, chunk-size]	443
act-{g,p}-layer-0	Activation function in first hidden layer in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ if available	see Fig. 7.3	{ReLU, tanh}
act-{g,p}-layer-1	Activation function in second hidden layer in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ if available	see Fig. 7.3	{none, none}
act-{g,p}-layer-2	Activation function in third hidden layer in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$ if available	see Fig. 7.3	{none, none}
{g,p}-out-act	Output layer activation function in $\{\boldsymbol{\gamma}, \boldsymbol{\pi}\}$	see Fig. 7.3	{BELU, SIGMOID}
fe-extension	FE extension scheme	$\{\boldsymbol{u}, \boldsymbol{u}_{\text{basic}}, \boldsymbol{u}_{\text{extensive}}\}$	$\boldsymbol{u}_{\text{basic}}$
inv-caps-init-pm	Gaussian mean for initial inverse thermal capacitance exponent for PM	$[-20, -0.1]$	-0.992
inv-caps-init-st	Gaussian mean for initial inverse thermal capacitance exponent for ST	$[-20, -0.1]$	-10.604
inv-caps-init-sw	Gaussian mean for initial inverse thermal capacitance exponent for SW	$[-20, -0.1]$	-9.353
inv-caps-init-sy	Gaussian mean for initial inverse thermal capacitance exponent for SY	$[-20, -0.1]$	-15.741

* see text for details

Table 7.1: Hyperparameter optimization configurations and found optima (TNN)

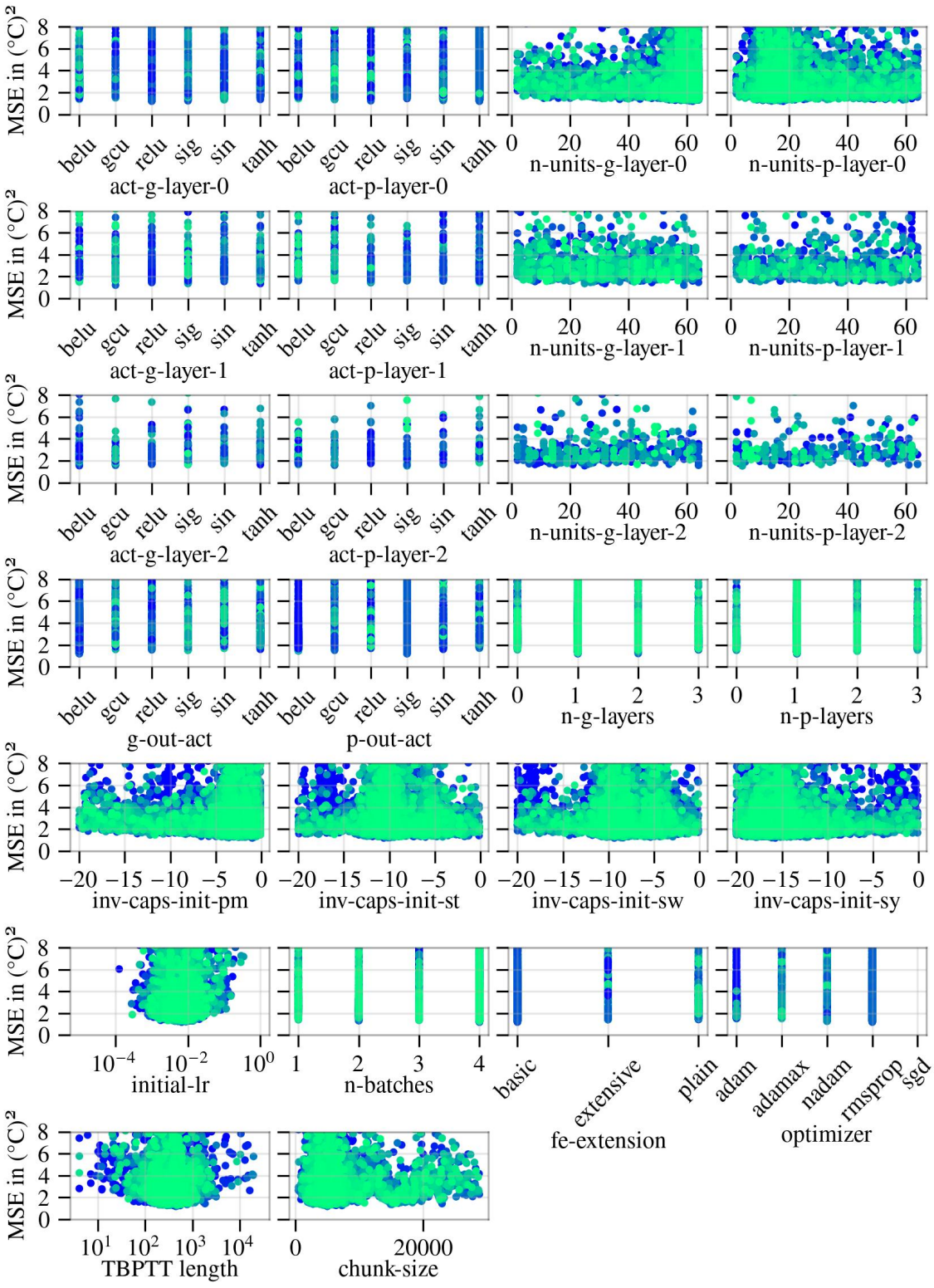


Figure 7.5: Hyperparameter distribution for TNN during HPO (greener samples were more recent). Evaluated on the test set.

against the achieved MSE in Fig. 7.5 is recommended. It becomes obvious that only few hyperparameters exhibit a score gradient along their intervals, i.e., most hyperparameters could have been arbitrarily chosen. Noteworthy hyperparameters whose value seems to make a difference are the initial learning rate at around $5 \cdot 10^{-3}$ and the optimizer, which should be chosen to be not the SGD optimizer. For all other hyperparameters, one or two variants or value ranges could be made out to be optimal, but not in a significant fashion, as each hyperparameter value is able to achieve a decent MSE of below 2 °C^2.

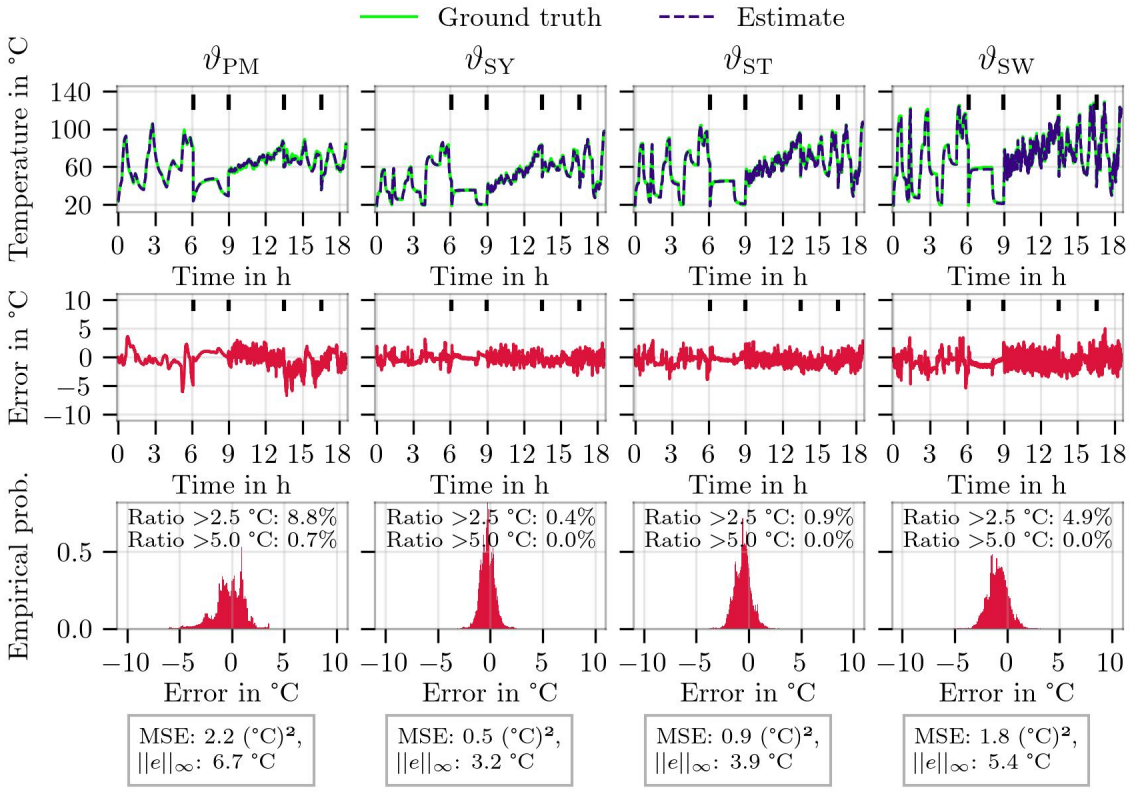


Figure 7.6: TNN hyperparameter optimum estimation performance on the generalization set $\mathcal{G}$

The performance of the top five TNNs according to the test set MSE does not differ largely in terms of generalization set accuracy, neither on average nor during worst-case scenarios, and also not in terms of model size. The performance of the best model among these is depicted in Fig. 7.6, and the model contains 1812 parameters. With double the parameter size of a small GRU but a tenth of that of the optimal GRU, the performance of the TNN lies in between both GRU variants, too. The best performing TNN staying at a humble amount of parameters and showing no strong dependence on hyperparameters, is already an indication of the lesser reliance on data in favor of the TNN training due to the incorporated differential equations. Moreover, much potential for further parameter elimination exists and will be investigated in the following.

In view of the high hyperparameter independence, it is likely to find smaller TNNs performing similar to the optimum. In Fig. 7.7, the model size of all HPO trials is plotted over their MSE and worst-case error $||e||_\infty$. Compared to the distribution of the GRU in Fig. 6.26, a substantially denser TNN distribution can be seen at model parameter numbers below 400 and, at the same time, at better accuracies on $\mathcal{T}_\mathrm{e}$. Along the Pareto front, among the smallest possible TNNs, the model that scores the best worst-case error on $\mathcal{T}_\mathrm{e}$ exhibits 117 model parameters, and achieves a score of 2.07 °C^2 MSE and 6.61 °C $||e||_\infty$ on $\mathcal{G}$. This is even smaller than the sparsified linear LASSO model at 160 model parameters, and at a considerably better accuracy. The worst-case error is even slightly better for the tiny TNN than for the hyperparameter optimum in Fig. 7.6. This tiny TNN features sub-ANNs that both contain one hidden layer with two neurons.

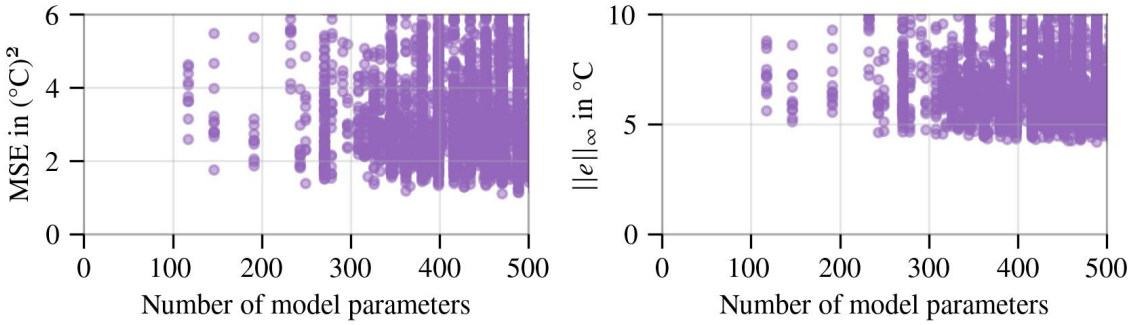


Figure 7.7: Estimation performance of the smallest models found during the HPO of the TNN on the test set

7.3.2 Model sparsification

A small and high-performing TNN could be found from analysis of the trials alone, but there are even more ways to further reduce the model size, which is unique to gray-box models. The TNN topology so far assumed a thermal conductance between each thermal node pair. When the sub-NNs' output is tracked, more conclusions about the necessity of some thermal conductances can be drawn, in order to suggest eliminations for a subsequent TNN design iteration. Thermal parameter analysis and subsequent sparsification of TNN models as outlined in this chapter were partially shown in [KWB22; KWB23].

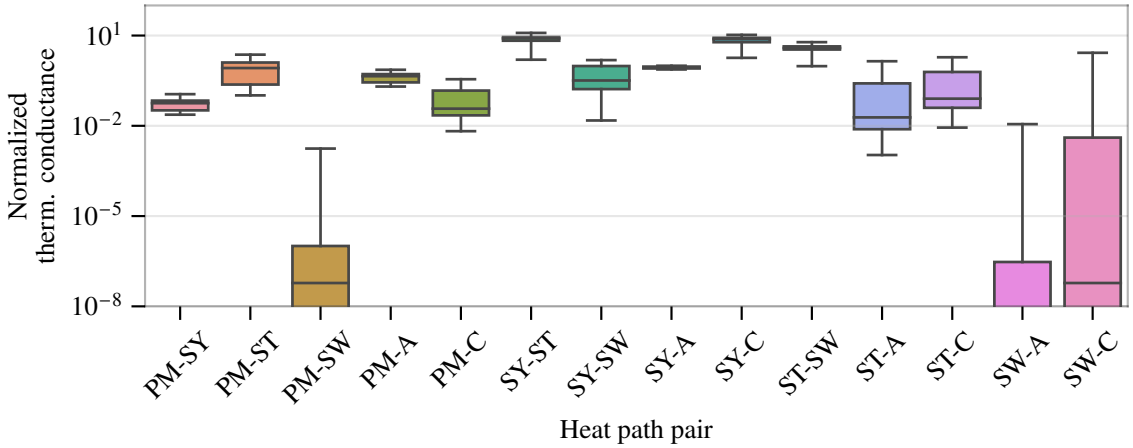


Figure 7.8: Normalized thermal conductances as estimated by the best tiny TNN (117 model parameters)

The thermal conductance as estimated by the tiny TNN of 117 model parameters for all heat paths when fed with the full data set is shown in Fig. 7.8. Likewise, the estimated power losses for this certain model are shown in Fig. 7.10 on the left, while on the right the distribution of the constant thermal capacitances across all eleven seeds can be seen. From the power losses, it becomes obvious that the stator winding component seems to experience the highest losses, followed by the stator yoke, stator teeth, and eventually the permanent magnets. This aligns conveniently with the known power loss sources as outlined in Sec. 3.1.4. From the thermal capacitances plot, the rank between permanent magnets and the stator targets seems to be captured by the TNN as well, however, for the stator yoke not all models agreed on whether the thermal capacitance should be larger or smaller compared to the PM. The median is in plausible order, nonetheless. Note that all thermal parameters are output as normalized quantities, that is, the output of the sub-ANNs is shown. These values cannot be easily transformed back in actual physical quantities if not at least one class of thermal parameters is known, e.g., the power losses. This is due to the several unknowns per equation (7.2), which is ambiguous especially since the thermal parameters appear in pair-wise products.

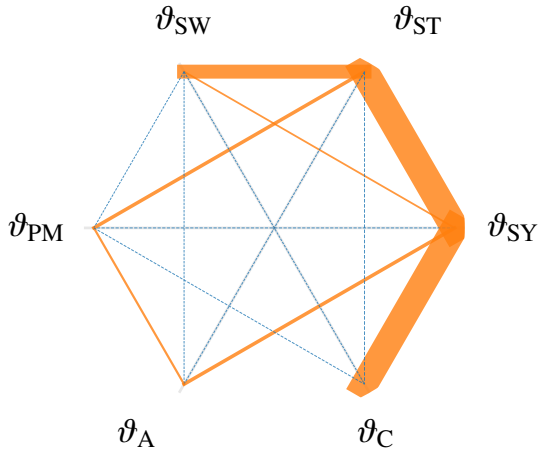


Figure 7.9: Median thermal conductances denote line thickness in this star chart. Dashed blue lines represent dropped conductances.

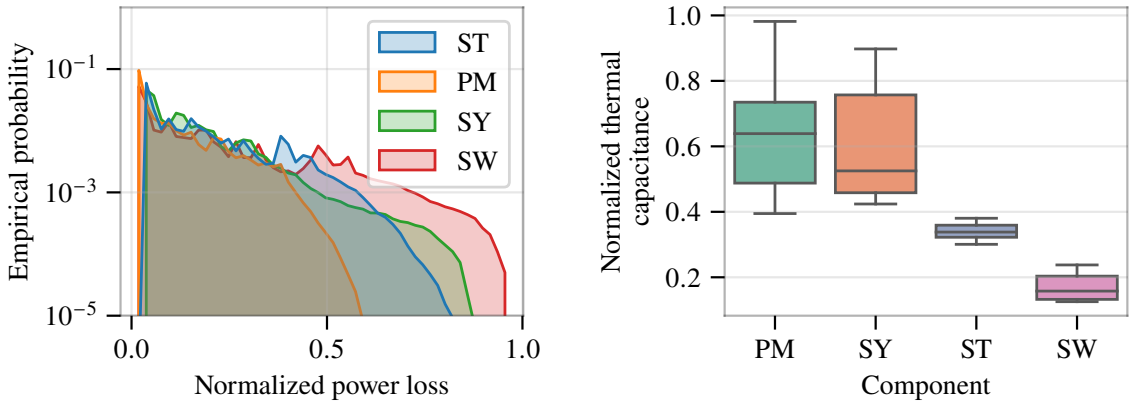


Figure 7.10: Normalized power loss distribution of the best tiny TNN and thermal capacitances across all 11 seeds

From the thermal conductance distribution over the permanent magnet (PM), stator yoke (SY), stator tooth (ST), stator winding (SW), coolant (C), and ambient temperature (A) in Fig. 7.8, it becomes evident that three pairs show dominant heat paths consistently (SY-ST, SY-C, ST-SW), and three other pairs are often relatively recessive (PM-SW, SW-A, SW-C). The dominant component pairs are, in fact, closely adjacent construction-wise, which aligns conveniently again (refer to the motor sketch in Fig. 5.3).

In order to further reduce the model size, the following seven component pairs are neglected: PM to SW; SW to A; SW to C; PM to SY; PM to C; ST to A; ST to C. This will be referred to as sparse TNN since $\boldsymbol{G}$ from (7.3) will have more entries equal to zero. The negligible connections are further illustrated in Fig. 7.9, where they denote dashed blue lines. Note that this diagram does not visualize the variance of any thermal conductance. Furthermore, the two sub-ANNs $\boldsymbol{\gamma}$ and $\boldsymbol{\pi}$ will be merged into one MLP with an output layer that has an extended amount of neurons to accommodate both thermal conductances and power losses. This unified, sparse TNN will be evaluated with two neurons in the hidden layer, as was the case with the tiny TNN from the previous chapter, and with one single neuron in the hidden layer, for minimal model parameters. The activation functions will be ReLU in the hidden and biased ELU in the output layer while on the input side the basic feature extension scheme is utilized. The performance distribution of these minimized architectures across 30 seeds is illustrated in Fig. 7.11.

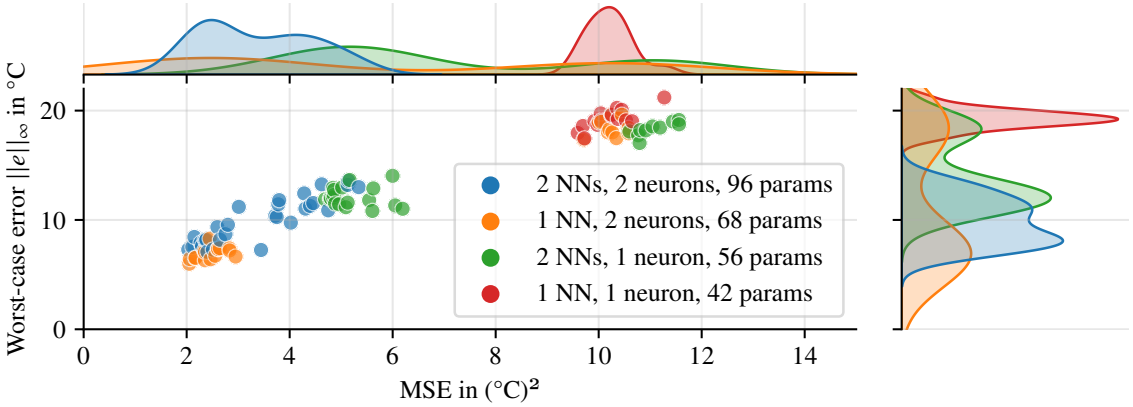


Figure 7.11: Performance of minimal architectures (sparse TNN) on $\mathcal{G}$

It becomes evident that facilitating at least two neurons in the hidden layer is significantly more important than using two sub-ANNs instead of one unified sub-ANN. Moreover, for any topology smaller than two sub-ANNs with two neurons there will be two modes of clusters into which the gradient descent will converge depending on the random initialization of the ANN weights. One is substantially more accurate than the other. The best performance under the least amount of parameters with an ANN structure is a sparse TNN that has a unified sub-ANN for both power losses and thermal conductances, that features one hidden layer with two neurons. The performance on the generalization set denotes 2.05 °C^2 MSE and 5.99 °C $||e||_\infty$ as worst-case error at 68 model parameters. Insights from previous experiments on black-box models in this work corroborate that no black-box model can achieve such performance with these few model parameters.

The array of thermal conductances could be further reduced up to the point where each target component would have only one thermal conductance to another thermal node (but not fewer in order to sustain stability, see next chapter). Moreover, a feature selection could also be conducted to save some weights on the input side. However, the performance would likely degrade further such that these optimizations are out of this work's scope.

7.3.3 Input-to-state stability and the severity of skewed initialization

Another advantage of TNNs over black-box models is the possibility to set the initial condition, which is the initial estimate at the same time. In simulations, the initial condition is just the first ground truth temperature of the target components, whereas in real-world applications it is conventional to set it to the ambient temperature as approximation. However, if this initial guess is inaccurate, will the TNN estimation recover to the ground truth temperature?

Another question orthogonal to the previous question is for input-to-state stability (ISS). An ISS system has bounded state values if the input is bounded, and they tend to an equilibrium when the inputs tend to zero. Note that a fulfilled ISS property does not necessarily guarantee that wrong initial state values can be recovered after a certain time, since the ISS property claims boundedness of states dependent on the boundedness of inputs, which is independent of the initial state values. What is more, equilibrium states that are exhibited by the model could not exist in the real system in case the model is simply wrong. In light of these limitations, both the ISS property and recovery to the ground truth of a certain TNN model are investigated in the following.

Characteristics of the ISS property are given in [SW95] and are as follows: assume a general nonlinear system $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}(t))$ with $\boldsymbol{f} : \mathbb{R}^Q \times \mathbb{R}^P \to \mathbb{R}^Q$ being continuously differentiable and

$$\boldsymbol{f}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0}. \tag{7.7}$$

The inputs $\boldsymbol{u}$ have to be measurable locally bounded functions $\boldsymbol{u} : \mathbb{R}_{\geq 0} \to \mathbb{R}^P$ with the supremum norm $\|\boldsymbol{u}\| = \sup\{|\boldsymbol{u}(t)|, t \geq 0\} \leq \infty$. For any such $\boldsymbol{u} \in \mathcal{U}$ and initial condition $\boldsymbol{x}_0$ the solution of the initial value problem shall be given by $\boldsymbol{x}(\boldsymbol{x}_0, \boldsymbol{u}, t)$. From system theory it is established that a $\mathcal{K}$-function $\rho : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is defined to be continuous, strictly increasing and outputs zero for a zero input $\rho(0) = 0$. Further, a $\mathcal{K}_\infty$-function is a $\mathcal{K}$-function for which additionally $\rho(s) \to \infty$ as $s \to \infty$, and it is positive-definite when $\rho(s) > 0$ for all $s > 0$, and $\rho(0) = 0$. Moreover, a $\mathcal{KL}$-function $\xi : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is defined to be a $\mathcal{K}$-function $\xi(\cdot, t)$ for each fixed $t \geq 0$, and for each fixed $s \geq 0$ it tends to zero as $t \to \infty$. Then [SW95], a system is globally ISS if there exists a $\mathcal{KL}$-function ξ and a $\mathcal{K}$-function ρ such that

$$|\boldsymbol{x}(\boldsymbol{x}_0, \boldsymbol{u}, t)| \leq \xi(|\boldsymbol{x}_0|, t) + \rho(\|\boldsymbol{u}\|) \tag{7.8}$$

for each $\boldsymbol{u} \in \mathcal{U}$, $\boldsymbol{x}_0 \in \mathbb{R}^Q$, and $t \geq 0$. In (7.8), ρ is also called the gain. It holds further that a general nonlinear system "is ISS if and only if it admits an ISS-Lyapunov function" [SW95]. An ISS-Lyapunov function for a general nonlinear system, in turn, is defined to be a smooth function $V : \mathbb{R}^Q \to \mathbb{R}_{\geq 0}$ for which there exist $\mathcal{K}_\infty$-functions α_1, α_2 and $\mathcal{K}$-functions α_3 and χ such that

$$\alpha_1(|\boldsymbol{x}_0|) \leq V(\boldsymbol{x}_0) \leq \alpha_2(|\boldsymbol{x}_0|) \quad \forall \boldsymbol{x}_0 \in \mathbb{R}^Q \tag{7.9}$$

and

$$\nabla V(\boldsymbol{x}_0) \cdot \boldsymbol{f}(\boldsymbol{x}_0, \mu) \leq -\alpha_3(|\boldsymbol{x}_0|) \tag{7.10}$$

for any $\boldsymbol{x}_0 \in \mathbb{R}^Q$ and any $\mu \in \mathbb{R}^P$ such that $|\boldsymbol{x}_0| \geq \chi(|\mu|)$. χ is also called the Lyapunov gain, and $\nabla V(\boldsymbol{x}_0)$ is the gradient of V with respect to $\boldsymbol{x}$ at $\boldsymbol{x}_0$. What is more, for interconnected systems, such as our application, essentially the same holds true with

$$V_q(x_q) \geq \max\{\max_{j=1}^{Q} \chi_{qj}(V_j(x_j)), \chi_q(|u|)\} \Rightarrow \nabla V_q(x_q) \cdot f_q(x_1, \ldots, x_Q, u) \leq -\alpha_q(V_q(x_q)). \tag{7.11}$$

Having these definitions in place, proving the ISS of a TNN is transferred to finding a suitable ISS-Lyapunov function. A typical candidate ISS-Lyapunov function for each of the Q target states is

$$V_q(\vartheta_q) = \frac{1}{2}\vartheta_q^2, \tag{7.12}$$

for which we further plug the continuous-time version of (7.3) into (7.10) to obtain

$$\dot{V}_q(\vartheta_q) \cdot \frac{\mathrm{d}\vartheta_q}{\mathrm{d}t} = \vartheta_q \cdot \kappa_q\Big(\pi_q(u) + \sum_{i \in \mathcal{M}^+ \setminus q} (\vartheta_i - \vartheta_q)\gamma_{i,q}(u)\Big). \tag{7.13}$$

Here, $\boldsymbol{\vartheta}$ consists of all target temperatures extended with the ancillary temperatures, such that $\mathcal{M}^+$ denotes the extended array of indices. With a typical Lyapunov gain

$$\chi(r) = \frac{1}{1-\epsilon} r, \tag{7.14}$$

where $\epsilon \in]0, 1[$, it can be obtained that for $\vartheta_q, u : \left|\vartheta_q\right| \geq \chi(|u|)$ and treating each temperature that is not ϑ_q

as another input it holds

$$\dot{V}_q(\vartheta_q) \cdot \frac{\mathrm{d}\vartheta_q}{\mathrm{d}t} \leq \underbrace{|\vartheta_q| \kappa_q \pi_q(u)}_{\geq 0} - \epsilon \kappa_q \sum_{i \in \mathcal{M}^+ \setminus q} \gamma_{i,q}(u) \left\|\vartheta_q\right\|^2, \tag{7.15}$$

$$\dot{V}_q(\vartheta_q) \cdot \frac{\mathrm{d}\vartheta_q}{\mathrm{d}t} \leq -\epsilon(Q + n - 1)ab\left\|\vartheta_q\right\|^2 = -\alpha_3(|\vartheta_q|) \tag{7.16}$$

While π_q can get equal to 0, we enforce topologically that $\kappa_q \geq a$ and for at least one γ per target $\gamma_{i,q} \geq b$ with $a, b \in \{x \in \mathbb{R} | x > 0\}$. This is ensured by using, e.g., a sigmoid or biased ELU output activation function in contrast to the ReLU. Eventually, (7.10) is fulfilled since $Q + n \geq 2$ for at least one thermal conductance per target, which makes $\alpha_3(\cdot)$ a positive-definite function.

Thus, in order for a TNN to be input-to-state stable, all target components have to have at least one thermal conductance greater zero to another component and no negative thermal conductances. Furthermore, non-negative power losses and all thermal capacitances being greater zero are further important to this property. More subtle, in order for the mathematical derivation to hold, the assumption (7.7) shall not be violated. However, this is not guaranteed in this work since the power loss sub-ANN is not topologically constructed to output zero if all input readings are zero. A TNN could come to this conclusion by training but since it is not ensured it would be far-fetched to claim a TNN to be ISS. Future research into ensuring such a property in the power loss branch of the TNN could achieve the ISS property though. In particular, a possible topological variation on the power loss branch is demonstrated in [GHN23] for a one-node TNN, which fulfills (7.7), and, thus, is shown to be ISS.

The recovery to ground truth temperatures for TNN models was shown empirically in [KWB22; KWB23]. For the minimized, sparse, 68-parameter TNN from the previous chapter, the estimation during the first half to one hour of each profile in $\mathcal{G}$ is plotted in Fig. 7.12, where 30 different detuned initializations are assumed. Hence, also for the TNNs trained in this work, the recovery to the ground truth is empirically demonstrated. It becomes further evident that different time constants for each target component lead to different recovery times, which range from a couple of minutes up to an hour worst-case for the PM and large initial deviations of up to 100 °C.

Based on the empirical results of this section, the claimed advantages of a TNN from [KWB23] can be confirmed :

- Flexible model structures in sub-ANNs and the internal LPTN;
- physically interpretable model states are featured as a result of ensuring consistency with the heat transfer ODE, which can be further constrained to promote input-to-state stability;
- based on the LPTN system of ODEs, initial values can be set;
- end-to-end differentiable, which enables fast, parallelized, and distributed training scenarios with automatic differentiation frameworks;
- neither expert knowledge, nor material or geometry information are required for the design;
- being completely data-driven, parasitic real-world effects are considered through observational data;
- a quasi-LPV system is identified where unknown relationships between scheduling variables and system matrices are automatically discovered;
- even relatively small model sizes achieve a high estimation accuracy, which enables real-time capability;
- less training data is required, since the search space for the function approximation is already confined by the system of ODEs.

One particular drawback that TNNs inherit is the fact that heat transfer under the assumption of lumped parameters has to be sufficiently accurate for the application requirements. However, classic (data-driven) LPTNs exhibit the same disadvantage but are utilized in industry for decades as the state of the art and have fewer advantages than a TNN.

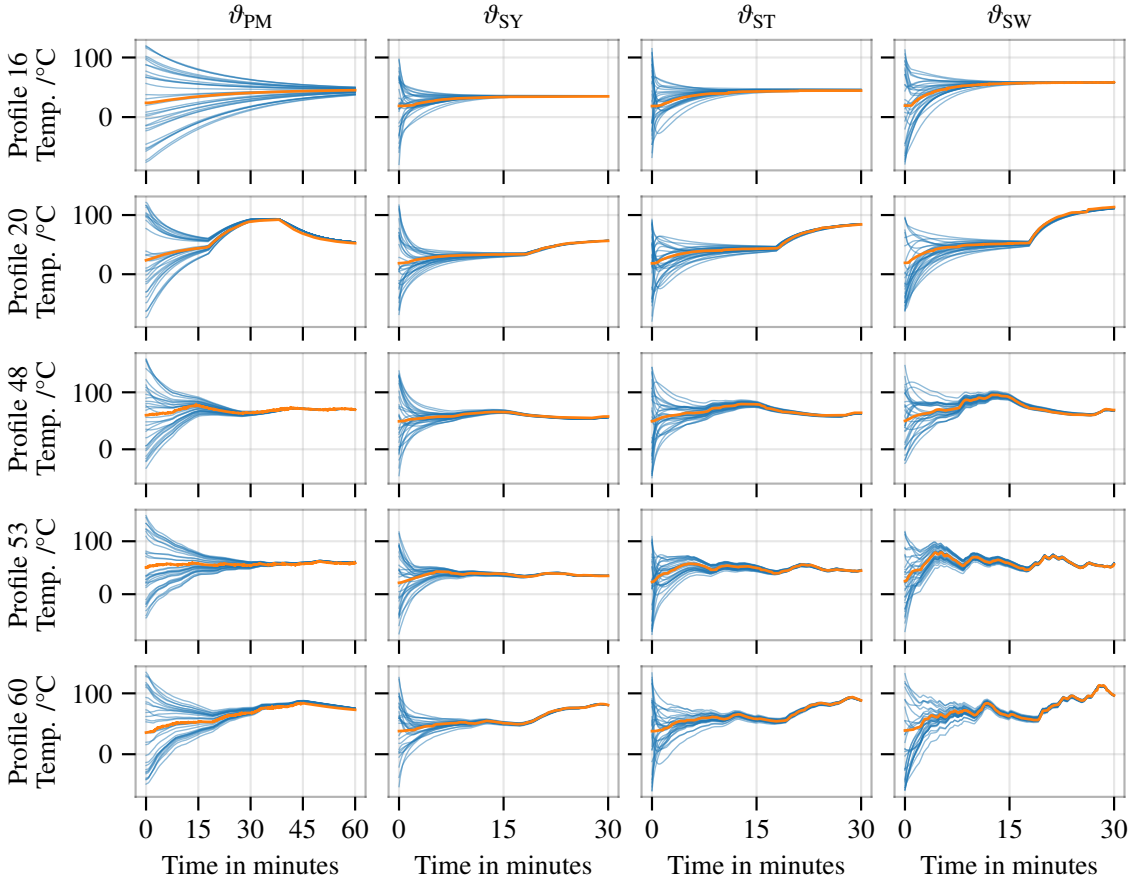


Figure 7.12: Estimation (blue) recovery to the ground truth (orange) for the small, sparse, 68-parameter TNN during all profiles of $\mathcal{G}$

7.4 Generalization to neural ordinary differential equations

NODEs (see Sec. 4.4.3) were motivated by the resemblance of residual neural networks (6.16) with explicit-Euler discretized ODEs. Since TNNs are trained with TBPTT, the question emerges whether NODEs with the adjoint sensitivity method as training algorithm might do better. Formulating a TNN as a NODE is straight-forward and just resorts to the continuous-time version of (7.3) with

$$\frac{\mathrm{d}\boldsymbol{\vartheta}}{\mathrm{d}t} = \boldsymbol{\kappa}(t) \odot \left(\boldsymbol{\pi}(t) + \left(\left(\mathbf{1}_{[Q]} \cdot \hat{\boldsymbol{\vartheta}}_{\mathrm{ext}}(t)^{\mathsf{T}} - \hat{\boldsymbol{\vartheta}}(t) \cdot \mathbf{1}_{[Q+n]}^{\mathsf{T}} \right) \odot \boldsymbol{G}(t)|_{Q} \right) \cdot \mathbf{1}_{[Q+n]} \right). \tag{7.17}$$

This is an extension of NODEs to further establish heat transfer principles, which could be also called a universal differential equation [Rac+21] accordingly. Using NODEs over classic TNNs could be beneficial because

- training suffers less from the exploding and vanishing gradient problem [PMB13], and
- the free choice of the ODE solver enables a NODE to be applied also to irregularly sampled data [Che+19].

The empirical performance of TNNs as NODEs was demonstrated in [KWB22]. There, it was shown that NODEs would achieve slightly better performances for a substantially increased training time, see an excerpt from [KWB22] in Fig. 7.13. Training times differ depending on the degree of parallelization through

mini-batches (MB) per epoch, but are larger than for a TNN in any configuration. Note that the training and validation set in Fig. 7.13 denote a different constellation than for the rest of this work, such that this graph is to be understood as qualitative indication.

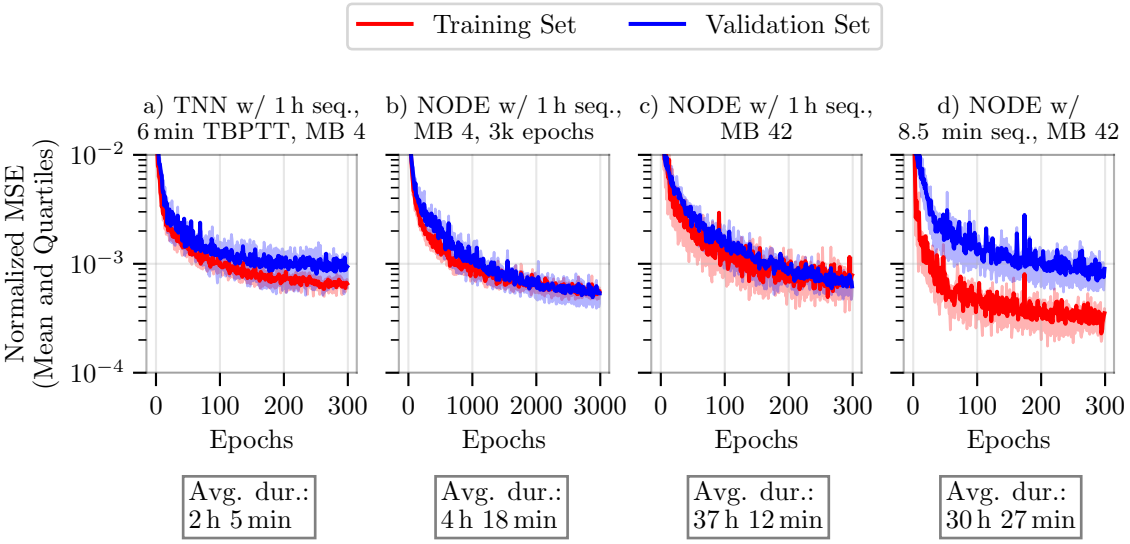


Figure 7.13: Training and validation set trends over epochs for different models and mini-batches per epoch. All variants experienced the same amount of weight updates. The mean (opaque) and 25 % / 75 % quartile bands (shaded) across all seeds are shown. a) A TNN performs decently with no overfitting. b)-d) NODE learning with certain hyperparameters scales better with the number of epochs than TNN learning but at increased training times (source: [KWB22]).

What is more, instead of the explicit Euler solver, the Runge-Kutta 4th-order solver was applied for the hyperparameters of Fig. 7.13 d), which, however, yielded the exact same training and validation trend. After all, the data set was regularly sampled at a relatively high frequency of 2 Hz with respect to the system state time constants, and the benefit of a more advanced solver over the explicit Euler cannot be leveraged in such a scenario. It can be summarized for NODEs that they represent an interesting tool for gray-box modeling but their advantages come at the cost of additional computational demand during training, which has to be justified by the data set conditions.

7.5 Optimization of a data-driven, expert-designed LPTN

Moving down the black- to white-box modeling scale (Fig. 3.6) further to light-gray-box modeling, heavily domain-expertise-driven LPTNs can be found where empirical data is merely utilized to fine-tune parameterized functions within a narrow range. One such LPTN design for the motor that generated data set $\mathcal{A}$ can be found in [Wal17]. In this section, the very same LPTN design is taken and optimized in a data-driven fashion according to two different paradigms. First, as it was done in [Wal17], the 34 free design parameters are optimized by a particle swarm optimization (PSO) [KE95], and then by a nonlinear, quasi-Newton optimization method under bound constraints. Particularly, the truncated Newton conjugate (TNC) method [NW06] as implemented in Scipy 1.7 [Vir+20] is utilized for this. Second, as an alternative approach, the free parameters are initially optimized by gradient descent (Nesterov ADAM) within the AD framework PyTorch [Pas+19], and then fine-tuned by the limited-memory Broyden-Fletcher-Goldfarb-Shannon (LBFGS) algorithm as quasi-Newton optimizer in the same framework.

The LPTN structure corresponds to the one shown in Fig. 3.9. The ODE of the LPTN (3.32) is restated as

$$c_i\big(\boldsymbol{\zeta}(t)\big)\frac{\mathrm{d}\vartheta_i}{\mathrm{d}t} = p_i\big(\boldsymbol{\zeta}(t)\big) + \sum_{j\in\mathcal{M}\setminus i}\frac{\vartheta_j-\vartheta_i}{r_{i,j}\big(\boldsymbol{\zeta}(t)\big)} + \sum_{j\in\{\mathrm{A,C}\}}\frac{\tilde{\vartheta}_j-\vartheta_i}{r_{i,j}\big(\boldsymbol{\zeta}(t)\big)}, \tag{7.18}$$

where c is the thermal capacitance, p the power loss, and $r_{i,j}$ the thermal resistance between temperature nodes i and j, where $i \in \mathcal{M}$ with $\mathcal{M} = \{\mathrm{PM, SY, ST, SW}\}$, and $r_{i,j} = r_{j,i}$. The scheduling vector $\boldsymbol{\zeta}$ merely indicates the operation point dependency of the thermal parameters. Instead of declaring such a vector, each thermal parameter is rather defined individually depending on material and geometry information. Particularly, it is determined in [Wal17] that all four thermal capacitances as well as $r_{i,j}$ for $i, j \in \{\mathrm{SY, ST, SW}\}$ are constant and non-negative. Further, $r_{\mathrm{ST,PM}}$, $r_{\mathrm{SW,PM}}$, and $r_{\mathrm{PM,\,A}}$ denote a mix of conductive and convective heat transfer depending on the motor speed n_{mech} with

$$r_{i,j}(n_{\mathrm{mech}}) = r_{i,j,0}\cdot e^{-\frac{n_{\mathrm{mech}}}{n_{\mathrm{mech,max}}b_{i,j}}} + a_{i,j}, \tag{7.19}$$

where the three free parameters lie between non-negative bounds. For the thermal path between the stator yoke and the coolant jacket a simple linear relationship with the coolant temperature is assumed

$$r_{\mathrm{SY,\,C}} = r_{\mathrm{SY,C,0}}\cdot[1+\alpha_{\mathrm{SY,C}}(\vartheta_{\mathrm{C}}-\vartheta_{\mathrm{C,0}})], \tag{7.20}$$

where $r_{\mathrm{SY,C,0}}$ is the thermal resistance at the coolant reference temperature $\vartheta_{\mathrm{C,0}} = 20\,^{\circ}\mathrm{C}$, and $\alpha_{\mathrm{SY,C}}$ is a nonpositive constant. Moreover, the thermal resistance between the permanent magnets and the cooling jacket is to be modeled as first-order polynomial in dependence of the coolant temperature and the motor speed:

$$r_{\mathrm{PM,C}}(n_{\mathrm{mech}},\vartheta_{\mathrm{C}}) = r_{\mathrm{PM,\,C,0}}\left(1 + a_{\mathrm{PM,\,C}}\frac{n_{\mathrm{mech}}}{n_{\mathrm{mech,max}}} + b_{\mathrm{PM,\,C}}\frac{\vartheta_{\mathrm{C}}}{\vartheta_{\mathrm{C,\,max}}} + c_{\mathrm{PM,\,C}}\frac{n_{\mathrm{mech}}}{n_{\mathrm{mech,max}}}\frac{\vartheta_{\mathrm{C}}}{\vartheta_{\mathrm{C,\,max}}}\right), \tag{7.21}$$

where $r_{\mathrm{PM,\,C,0}}$ is non-negative, while $a_{\mathrm{PM,\,C}}$ and $b_{\mathrm{PM,\,C}}$ are nonpositive. Furthermore, the maximum values are determined to be $\vartheta_{\mathrm{C,\,max}} = 100\,^{\circ}\mathrm{C}$ and $n_{\mathrm{mech,max}} = 6000\,\frac{1}{\mathrm{min}}$.

For the power losses, a well-thought combination of empirical loss measurements at a test bench and copper as well as mechanical loss models is employed. This implies that, for the LPTN design to work as intended, measuring the total losses with a power analyzer for the full torque and the positive speed range beforehand is required. What is more, torque has to be measured or estimated during operation in order to index these LUTs. Notwithstanding these facts, the copper loss model pertains to the power loss in the stator winding node with

$$p_{\mathrm{SW}} = \frac{3}{2}r_{\mathrm{DC}}\left\|\boldsymbol{i}_{\mathrm{dq}}\right\|_2^2\left(\alpha_{\mathrm{SW,\,1}}(\vartheta_{\mathrm{SW}}) + \alpha_{\mathrm{SW,\,2}}(n_{\mathrm{mech}})\frac{\alpha_{\mathrm{SW,\,2}}(n_{\mathrm{mech}})-1}{\big(\alpha_{\mathrm{SW,\,1}}(\vartheta_{\mathrm{SW}})\big)^{\beta_{\mathrm{CU}}}}\right), \tag{7.22}$$

where

$$\alpha_{\mathrm{SW,\,1}}(\vartheta_{\mathrm{SW}}) = 1 + \alpha_{\mathrm{CU}}(\vartheta_{\mathrm{SW}} - \vartheta_{\mathrm{SW,\,0}}), \tag{7.23}$$

$$\alpha_{\mathrm{SW,\,2}}(n_{\mathrm{mech}}) = 1 + \alpha_{\mathrm{AC,\,1}}\frac{n_{\mathrm{mech}}}{n_{\mathrm{mech,\,max}}} + \alpha_{\mathrm{AC,\,2}}\left(\frac{n_{\mathrm{mech}}}{n_{\mathrm{mech,\,max}}}\right)^2 \tag{7.24}$$

with the free design parameters $\theta_{\mathrm{LPTN,\,SW}} = [\alpha_{\mathrm{CU}}, \beta_{\mathrm{CU}}, \alpha_{\mathrm{AC,\,1}}, \alpha_{\mathrm{AC,\,2}}, r_{\mathrm{DC}}]$. Morever, the empirical iron loss model uses information from two LUTs, one for the measured total losses and one for the mechanical loss portion thereof approximated by drag characteristics. Then, the iron losses are described by

$$p_{\mathrm{FE}} = p_{\mathrm{L}} - \alpha_{\mathrm{drag}}d - \frac{3}{2}r_{\mathrm{DC}}\left\|\boldsymbol{i}_{\mathrm{dq}}\right\|_2^2\alpha_{\mathrm{SW,2}} \tag{7.25}$$

with α_{drag} denoting a free parameter and d denoting the LUT-informed drag. Furthermore, two ratios $k_1, k_2 : \{k \in \mathbb{R} | 0 \leq k \leq 1\}$ are determined where k_1 is operation point dependent:

$$k_1(n_{\text{mech}}, \boldsymbol{i}_{\text{dq}}) = k_{1,0} + k_{1,1}\frac{n_{\text{mech}}}{n_{\text{mech, max}}} + k_{1,2}\frac{\left\|\boldsymbol{i}_{\text{dq}}\right\|_2^2}{i_{\text{max}}} + k_{1,3}\frac{n_{\text{mech}}\left\|\boldsymbol{i}_{\text{dq}}\right\|_2^2}{n_{\text{mech, max}} i_{\text{max}}} \tag{7.26}$$

with $i_{\text{max}} = 256\,\text{A}$ and all coefficients being free parameters. With these ratios the total losses are assigned onto the remaining three temperature nodes such that at reference temperature no loss surplus occurs. On top of that, a negative linear relationship with the corresponding component temperature is considered through a free parameter α_{FE}. Hence, the resulting losses are

$$p_{\text{PM}} = (1 - k_1) p_{\text{FE}} \left(1 + \alpha_{\text{FE}} \left(\vartheta_{\text{PM}} - \vartheta_{\text{PM,0}}\right)\right), \tag{7.27}$$

$$p_{\text{SY}} = k_2 k_1 p_{\text{FE}} \left(1 + \alpha_{\text{FE}} \left(\vartheta_{\text{SY}} - \vartheta_{\text{SY,0}}\right)\right), \tag{7.28}$$

$$p_{\text{ST}} = (1 - k_2) k_1 p_{\text{FE}} \left(1 + \alpha_{\text{FE}} \left(\vartheta_{\text{ST}} - \vartheta_{\text{ST,0}}\right)\right) \tag{7.29}$$

with the reference temperatures $\vartheta_{\text{PM,0}} = 63\,^\circ\text{C}$, $\vartheta_{\text{SY,0}} = 55\,^\circ\text{C}$, and $\vartheta_{\text{ST,0}} = 65\,^\circ\text{C}$. In order to ensure consistency with the first principles of heat transfer physics through non-negative power loss terms, thermal resistances and thermal capacitances, the so far outlined functional models are not sufficient. During optimization of the free design parameters, the bounds are also to be expert-designed and chosen such that these properties hold at any time.

The boundaries for each of the 34 design parameters as well as the optimum as stated in [Wal17] are compiled in Tab. 7.2. A very similar optimization approach is conducted in this work with a PSO followed by a TNC optimization. The found optimum is presented in Tab. 7.2, too. All optimizations are conducted on the training set $\mathcal{T}_{\text{r}}$ whereas the generalization set $\mathcal{G}$ is only used for the final evaluation after training.

The PSO in this work started off with the optimum in [Wal17] as global optimum informing all particles – see Fig. 7.14 for the optimization trend. Hence, the search had an advantage over the PSO in Wallscheid's work but could not surpass its optimum's performance at least on the training set regardless. Further differences were the following: the deployed PSO consisted of 120 particles that ran for 10000 iterations with no early stopping, and each particle could inform one another. The best five particles were handed over to the TNC optimization. In [Wal17], over 300 particles ran for as long as early stopping with 50 iterations without an improvement of greater than 0.01 % did not cancel the search or as long as 506 iterations, while each particle could inform only 10 % of other particles in the swarm. There, the best 8 particles were handed over to a sequential quadratic programming (SQP) solver. More strikingly, this work's basis denotes a substantially larger data set with 185 h whereas Wallscheid's work was based on a mere subset of around 30 h.

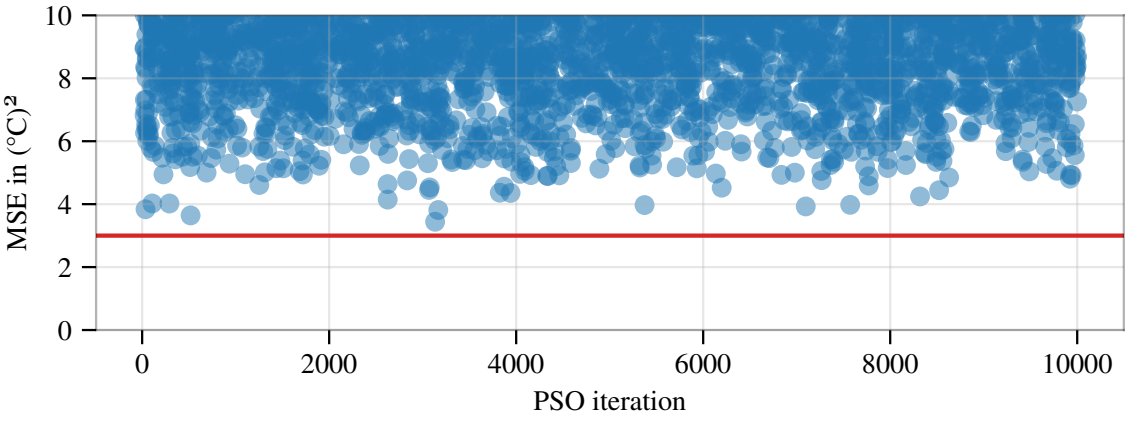


Figure 7.14: PSO cost trend on $\mathcal{T}_{\text{r}}$. Wallscheid LPTN performance is shown as red horizontal line.

Parameter	Bounds	Wallscheid's optimum [Wal17]	This work's optimum
c_{PM}	$[10^2, 10^5]\,\mathrm{J/K}$	10 666.0 J/K	9984.1 J/K
c_{SY}	$[10^2, 10^5]\,\mathrm{J/K}$	6509.3 J/K	6787.3 J/K
c_{ST}	$[10^2, 10^5]\,\mathrm{J/K}$	437.1 J/K	196.3 J/K
c_{SW}	$[10^2, 10^5]\,\mathrm{J/K}$	3510.5 J/K	3766.2 J/K
$r_{\mathrm{SY,SW}}$	$[0.01, 1]\,\mathrm{K/W}$	37.5 mK/W	37.2 mK/W
$r_{\mathrm{SY,ST}}$	$[0.01, 1]\,\mathrm{K/W}$	70.7 mK/W	67.6 mK/W
$r_{\mathrm{SW,ST}}$	$[0.01, 1]\,\mathrm{K/W}$	89.9 mK/W	93.2 mK/W
$\alpha_{\mathrm{SY,C}}$	$[-10^{-2}, -10^{-4}]\,1/\mathrm{K}$	$-54 \cdot 10^{-4}$ 1/K	$-50.8 \cdot 10^{-4}$ 1/K
$r_{\mathrm{SY,C,0}}$	$[10^{-4}, 10^{-1}]\,\mathrm{K/W}$	18 mK/W	18.1 mK/W
$r_{\mathrm{ST,PM,0}}$	$[0.1, 4]\,\mathrm{K/W}$	1727.5 mK/W	1868.8 mK/W
$r_{\mathrm{SW,PM,0}}$	$[0.1, 4]\,\mathrm{K/W}$	848.6 mK/W	616.2 mK/W
$r_{\mathrm{PM,A,0}}$	$[0.1, 4]\,\mathrm{K/W}$	634.9 mK/W	662.7 mK/W
$b_{\mathrm{ST,PM}}$	$[10^{-2}, 50 \cdot 10^{-2}]$	$157.3 \cdot 10^{-3}$	$147.9 \cdot 10^{-3}$
$b_{\mathrm{SW,PM}}$	$[10^{-2}, 50 \cdot 10^{-2}]$	$142.8 \cdot 10^{-3}$	$123.3 \cdot 10^{-3}$
$b_{\mathrm{PM,A}}$	$[10^{-2}, 50 \cdot 10^{-2}]$	$118.4 \cdot 10^{-3}$	$125.2 \cdot 10^{-3}$
$a_{\mathrm{ST,PM}}$	$[10^{-2}, 1]\,\mathrm{K/W}$	303.9 mK/W	243.0 mK/W
$a_{\mathrm{SW,PM}}$	$[10^{-2}, 1]\,\mathrm{K/W}$	231.9 mK/W	247.7 mK/W
$a_{\mathrm{PM,A}}$	$[10^{-2}, 1]\,\mathrm{K/W}$	120.5 mK/W	109.7 mK/W
$r_{\mathrm{PM,C,0}}$	$[10^{-4}, 1]\,\mathrm{K/W}$	352.8 mK/W	397.6 mK/W
$a_{\mathrm{PM,C}}$	$[-1, 0]$	$-248.4 \cdot 10^{-3}$	$-278.9 \cdot 10^{-3}$
$b_{\mathrm{PM,C}}$	$[0, 1]$	$27.6 \cdot 10^{-3}$	$89.8 \cdot 10^{-3}$
$c_{\mathrm{PM,C}}$	$[0, 1]$	$333.1 \cdot 10^{-3}$	$340.4 \cdot 10^{-3}$
r_{DC}	$[10^{-4}, 1]\,\mathrm{K/W}$	14.6 mK/W	15.2 mK/W
α_{CU}	$[10^{-5}, 10^{-2}]\,1/\mathrm{K}$	$2.0 \cdot 10^{-3}\,1/\mathrm{K}$	$1.7 \cdot 10^{-3}\,1/\mathrm{K}$
$\alpha_{\mathrm{AC,\,1}}$	$[0.1, 1]$	$56.2 \cdot 10^{-2}$	$49.18 \cdot 10^{-2}$
$\alpha_{\mathrm{AC,\,2}}$	$[0.1, 1]$	$24.1 \cdot 10^{-2}$	$28.0 \cdot 10^{-2}$
β_{CU}	$[0.1, 5]$	$2566.7 \cdot 10^{-3}$	$2810.0 \cdot 10^{-3}$
$k_{1,0}$	$[0, 1]$	$544.1 \cdot 10^{-3}$	$555.7 \cdot 10^{-3}$
$k_{1,1}$	$[0, 1]$	$7.8 \cdot 10^{-3}$	$24.9 \cdot 10^{-3}$
$k_{1,2}$	$[0, 1]$	$35.2 \cdot 10^{-3}$	$9.6 \cdot 10^{-3}$
$k_{1,3}$	$[-1, 1]$	$-743.8 \cdot 10^{-3}$	$-795 \cdot 10^{-3}$
k_2	$[0, 1]$	$865.5 \cdot 10^{-3}$	$830.8 \cdot 10^{-3}$
α_{FE}	$[-10^{-2}, 0]\,1/\mathrm{K}$	$-28 \cdot 10^{-4}\,1/\mathrm{K}$	$-20.2 \cdot 10^{-4}\,1/\mathrm{K}$
α_{drag}	$[0.1, 10]$	$1476.2 \cdot 10^{-3}$	$1885.0 \cdot 10^{-3}$

Table 7.2: Expert-designed LPTN parameters, their boundaries, and found optima

As another contending optimization algorithm, a gradient-descent- and AD-based optimization with the NADAM algorithm [Doz16] within the PyTorch framework is conducted on a slightly adapted LPTN design. The training and validation trend is displayed in Fig. 7.15. Since the gradient descent algorithm does not consider parameter bounds trivially, the topology of the LPTN is changed such that the parameters are transformed into the permitted domain by appropriate activation functions, among other adaptations: first, the reference temperatures are normalized by dividing through 200 °C. Second, instead of the thermal resistances $r_{\mathrm{ST,\,PM}}$, $r_{\mathrm{SW,PM}}$, and $r_{\mathrm{PM,\,A}}$ the reciprocal counterpart is modeled with

$$g_{i,j} = g_{i,j,0} \cdot \sigma\left(b_{i,j} \cdot \frac{n_{\mathrm{mech}}}{n_{\mathrm{mech,max}}}\right) + a_{i,j}, \tag{7.30}$$

where σ is the sigmoid activation function. All free parameters are initialized with zero, except for the thermal capacitances, whose inverse is modeled instead and drawn from a Gaussian $\mathcal{N}(\mu = -7\,\mathrm{K/J}, \sigma = 1\,\mathrm{K/J})$. Moreover, all parameters except for $\{\alpha_{\mathrm{AC},1}, \alpha_{\mathrm{AC},2}, k_{1,0}, k_{1,1}, k_{1,2}, k_{1,3}\}$ are transformed by an activation function: while $\{\alpha_{\mathrm{FE}}, a_{\mathrm{PM,C}}, b_{\mathrm{PM,C}}, c_{\mathrm{PM,C}}, k_2, k_1(n_{\mathrm{mech}}, \boldsymbol{i}_{\mathrm{dq}})\}$ are transformed by the sigmoid function (and the former three also negated afterwards to keep the nonpositivity), the remaining parameters are transformed by the softplus activation function, which is very similar to the biased ELU and denoted by $f_{\mathrm{sp}}(x) = \ln(1 + e^x)$. With this design choice several magnitudes of parameter values can be realized while sustaining the non-negativity property.

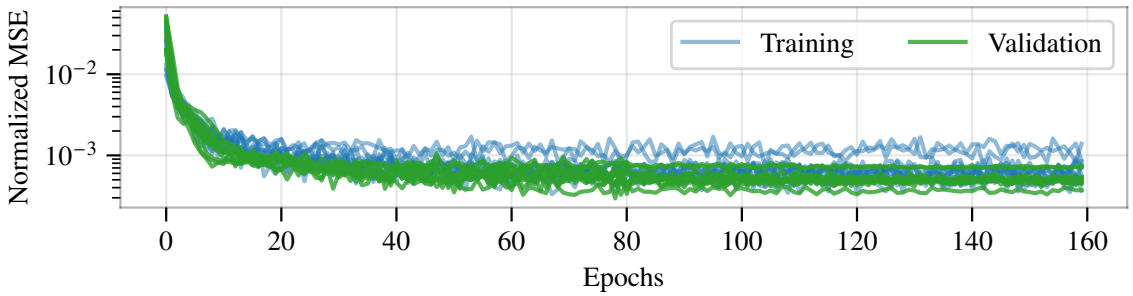


Figure 7.15: Expert-designed LPTN optimization trend (NADAM)

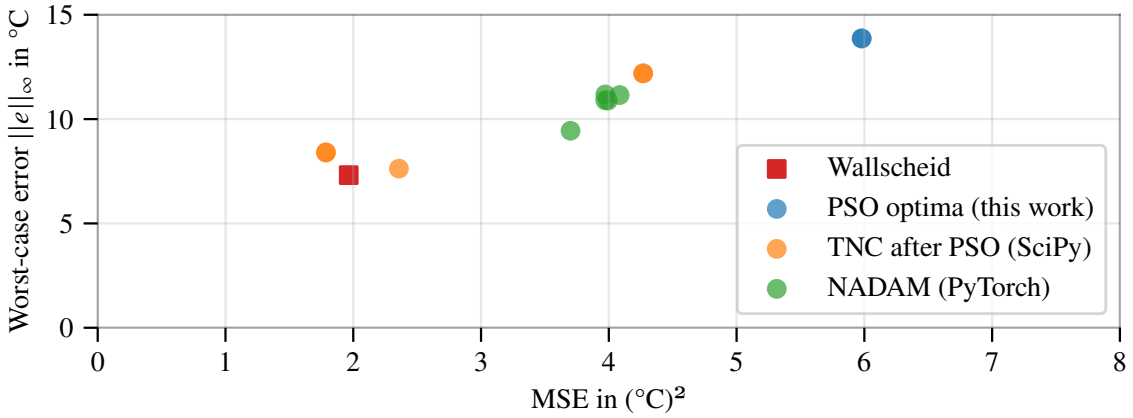


Figure 7.16: Expert-designed LPTN optimizations (Scores on $\mathcal{G}$)

The final performance of the differently optimized LPTNs is illustrated in Fig. 7.16. It becomes evident that only one PSO parameter set out of the five best could be optimized by TNC so far that the Wallscheid optimum is slightly surpassed in terms of the MSE. The unoptimized PSO parameter sets, on the other hand, show very weak performance on $\mathcal{G}$. The gradient-descent-optimized LPTNs take a middle position between the parameter sets found by heuristics and by second-order optimization methods.

8 Conclusion

This thesis addresses the data-driven thermal modeling of permanent magnet synchronous motors (PMSMs) with machine learning algorithms in an automotive context. The foundation of all investigations was a comprehensive measurement data set that was collected at a laboratory test bench. The characteristics of this data set, the excitation strategy, and the separation into well diversified clusters was treated in Sec. 5. It was shown that, in contrast to various other motor data sets, the data set at hand features a wide coverage of the motor-speed-torque-plane, which was achieved through random walks.

In Sec. 6 the whole process of designing a black-box machine learning (ML) model was illuminated. The importance of feature engineering was empirically shown by a coarse grid search through different normalization schemes and feature additions for various classical ML models ranging from linear regression over tree-based approaches up to multi-layer perceptrons. It could be recorded that the ANN-based model shows the best performance followed by the linear least absolute shrinkage and selection operator (LASSO), whereas more model parameters are necessary than for state-of-the-art lumped-parameter thermal networks (LPTNs). Moreover, the worst-case error across all four target component temperatures has been consistently above 20 °C for all these models, which falsely assume a static relationship between regressors and target temperatures. Since these large errors often occur during the beginning of measurement profile estimations, they could be ascribed to one of the most important feature – the exponentially weighted moving average (EWMA) – which would not be well-conditioned at the beginning of a profile, but have been indispensable to overcome the inappropriate independent and identically distributed (i.i.d.) data assumption. More detailed analysis of the linear model class led to a regularized set of coefficients that could reduce the worst-case error to slightly above 10 °C on an optimistic test set, which could not be taken over to generalizing estimates. Then, recurrent and convolutional ANNs, which were invented to be applied on time series data, were investigated including a hyperparameter optimization. It turned out that the gated recurrent unit (GRU) exhibits by far the best performance in terms of both the mean squared error and the worst-case error at medium to large model sizes. In contrast to earlier work, it could be shown that this performance was achievable without 1) the use of EWMAs, 2) appending a feed-forward output layer, or 3) adding residual connections. Avoiding these augmentations all led to substantial parameter reductions.

Since even a GRU is not physically interpretable, the domain of gray-box modeling was entered in Sec. 7 in an endeavor to further reduce the memory footprint of the data-driven thermal models in an eventual microcontroller. Here, a gradual movement from the dark-gray-box modeling paradigm over to bright-gray-box modeling schemes that incorporate plenty of domain expertise was empirically evaluated. Starting with a nonlinear function approximator that is enveloped into a difference equation according to the explicit Euler discretization scheme, the many advantages of TNNs were established, which build upon the LPTN difference equation and provide one function approximator per thermal parameter category. Not only was the estimation performance among the best but also the amount of model parameters could be reduced drastically, while at the same time potential for further sparsification depending on observations of the ANN output after training as well as preceding geometry examinations ensues. Moreover, the recovery to the ground truth from skewed initial conditions is achievable under certain light conditions on the activation functions within the TNN. Eventually, an expert-designed LPTN was retrained on the data set at hand with heuristics, second-order optimizers as well as gradient descent within an automatic differentiation framework in order to give context for the effectiveness of ANN-based models. The employed optimization algorithms are efficient for large data sets and parallelizable, such that more extensive experiments, e.g., the hyperparameter optimizations (HPOs),

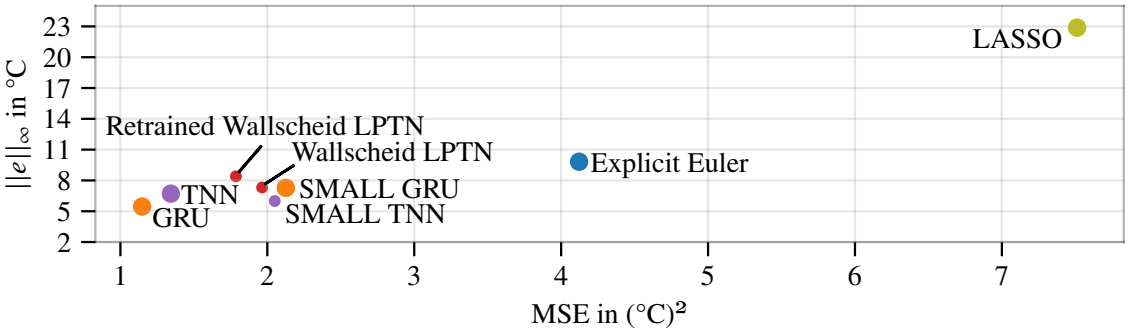


Figure 8.1: Final performance comparison of best data-driven thermal models on $\mathcal{G}$

	Model size	LUT entries	FLOPs
(Retrained) Wallscheid LPTN	34	12810	125
Small TNN	68	0	151
LASSO	472	0	942
Explicit Euler	708	0	368
Small GRU	1002	0	1814
TNN	1812	0	3516
GRU	24744	0	48478

Table 8.1: Model size and computational demand of best data-driven thermal models

were conducted on a modern high-performance computing cluster (Paderborn Center for Parallel Computing - PC²).

A performance overview for the best models from all categories are plotted in Fig. 8.1, while Tab. 8.1 compiles the corresponding required amount of model parameters (model size), the number of LUT entries, and the amount of floating-point operations (FLOPs). Models with less than 100 parameters are shown with a smaller dot. The claimed FLOPs and LUT entries values ignore activation functions that would also require additional compute and memory in an embedded system depending on how much samples are used per function to interpolate from. The FLOPs value denotes the amount necessary to estimate four target component temperatures at any time instant, and ignores scaling operations for feature normalization. Here, multiplications and additions denote distinct FLOPs each.

It becomes evident that the linear LASSO model has the weakest performance at a modest parameter number and computational demand, but is also the simplest and fastest to obtain. Better performance is achievable with the explicit Euler model from Sec. 7.2, which requires more parameters but significantly fewer FLOPs. At some distance, the remaining modeling paradigms cluster at around 1.7 °C^2 mean squared error (MSE) and 7 °C worst-case error. Obviously, the best performance is achievable with a full GRU or a TNN, while the smaller variants forfeit some accuracy for their (much) smaller size. However, the TNN is substantially smaller in the number of parameters and computational demand, among several other previously mentioned advantages compared to the GRU. The expert-designed LPTNs according to [Wal17], eventually, feature the smallest amount of model parameters with a performance between the small and full variants of the TNN and GRU. In fact, the performances of the GRUs, TNNs, and LPTNs are so close together that a certain announced ranking would be susceptible to change from subtle experimental fluctuations, such as different training-test splits, other motor data sets, or even other random initializations in case of the ANN-based models. In contrast, what can be definitely recapped is the fact that ANN-based models do not require look-up tables or foregoing power analyzer measurements to function, and scrutinizing the material or geometry does not

need to precede their design. The barrier for designing an ANN-based thermal model is, thus, much lower than for a LPTN, and special domain expertise is not mandatory. Moreover, although a small TNN requires exactly double the amount of model parameters as the LPTN, the number of FLOPs does not scale as much and is rather slightly higher, while plenty of memory reserved for LUTs can be released. In summary, the TNN approach provides the most advantages across all paradigms - high accuracy, small memory footprint, low computational demand, interpretable states, data-driven and, hence, automatic identification, as well as lower data volume requirements due to the constrained search space. The combination of these properties facilitates a fast time-to-market, which is especially desirable in the automotive sector.

Outlook

This thesis investigated many different data-driven thermal models for a PMSM and, thus, offers a comprehensive comparison for the interested engineer. Due to the many assets and the novelty of the TNN approach, it is likely most auspicious and rewarding to continue research into the gray-box paradigm that TNNs have opened up. For instance, the power loss modeling through a MLP is very uninformed of the heat transfer principles behind. Hence, this sub-model probably exhibits still room for improvement in terms of accuracy and required parameters. In general, AD is a powerful tool for optimizing arbitrary functions and programs, and denotes an exciting path for further research in any engineering field. Apart from that, the experiments in this work do not have the final say in terms of optimization. It remains open whether the HPOs were equipped with the best set of hyperparameters, the most appropriate sampling strategies, or the best random initialization distributions. The gradient descent algorithm is susceptible to local minima after all. How to set up model structures that mitigate the risk of local minima, effectively reducing scatter across different random number generator seeds, is one of the most pressing questions. Furthermore, splitting records up in training, validation, test and generalization sets is poorly surveyed in literature yet, although they denote a significant impact on performance results. Similarly, the excitation planning of nonlinear dynamic systems with the objective of generating a very diverse, informative data set with as least samples as possible has little to no room in the academic landscape despite the large amount of engineers that are regularly confronted with this task.

The developed thermal modeling algorithms in this work are easily applicable to neighboring academic and industrial applications, such as computer chips, power electronic semiconductor modules, batteries, or different motor types. The empirical investigation of these applications is obviously one of many next steps. Further it remains an open question in how far a certain manufacturing tolerance affects the thermal modeling accuracy when trained on a single prototypical motor. Eventually, transfer learning to other motors with different geometry, material or even function principles denotes an interesting future research prospect.

Bibliography

[ADK12] Y. Avenas, L. Dupont, and Z. Khatir. "Temperature Measurement of Power Semiconductor Devices by Thermo-Sensitive Electrical Parameters—A Review". In: *IEEE Transactions on Power Electronics* 27.6 (2012), pp. 3081–3092. DOI: 10.1109/TPEL.2011.2178433.

[Agh+21] I. Aghabali, J. Bauman, P. J. Kollmeyer, Y. Wang, B. Bilgin, and A. Emadi. "800-V Electric Vehicle Powertrains: Review and Analysis of Benefits, Challenges, and Future Trends". In: *IEEE Transactions on Transportation Electrification* 7.3 (2021), pp. 927–948. DOI: 10.1109/TTE.2020.3044938.

[AGH18] P. M. Addo, D. Guegan, and B. Hassani. "Credit risk analysis using machine and deep learning models". In: *Risks* 6.2 (2018), p. 38. DOI: 10.3390/risks6020038.

[AH19] A. A. Adly and A. Huzayyin. "The impact of demagnetization on the feasibility of permanent magnet synchronous motors in industry applications". In: *Journal of advanced research* 17 (2019), pp. 103–108. DOI: 10.1016/j.jare.2019.02.002.

[AL21] H. Alibrahim and S. A. Ludwig. "Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization". In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. 2021, pp. 1551–1559. DOI: 10.1109/CEC45853.2021.9504761.

[AM20] E. Adamopoulou and L. Moussiades. "An overview of chatbot technology". In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer. 2020, pp. 373–383. DOI: 10.1007/978-3-030-49186-4_31.

[Ana+09] T. Anastasakos, D. Hillard, S. Kshetramade, and H. Raghavan. "A Collaborative Filtering Approach to Ad Recommendation using the Query-Ad Click Graph". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009, pp. 1927–1930. DOI: 10.1145/1645953.1646267.

[And+13] D. Andre, A. Nuhic, T. Soczka-Guth, and D. U. Sauer. "Comparative study of a structured neural network and an extended Kalman filter for state of health determination of lithium-ion batteries in hybrid electric vehicles". In: *Engineering Applications of Artificial Intelligence* 26.3 (2013), pp. 951–961. DOI: 10.1016/j.engappai.2012.09.013.

[And+99] E. Anderson, Z. Bai, C. Bischof, et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[Anu20] K. Anuforo. "Temperature Estimation in Permanent Magnet Synchronous Motor (PMSM) Components using Machine Learning". MA thesis. Dublin, National College of Ireland, 2020.

[Árp+21] I. Árpád, J. T. Kiss, G. Bellér, and D. Kocsis. "Investigation of the impact of EU and governmental measures on the spread and the energy supply of electromobility in Hungary". In: *Environmental Progress & Sustainable Energy* 40.4 (2021), e13575. DOI: 10.1002/ep.13575.

[Ata+99] K. Atallah, D. Howe, P. H. Mellor, and D. A. Stone. "Rotor loss in permanent magnet brushless AC machines". In: *IEEE International Electric Machines and Drives Conference. IEMDC'99. Proceedings (Cat. No.99EX272)*. 1999, pp. 60–62. DOI: 10.1109/IEMDC.1999.769027.

[AW08] H. Abbas and H. Werner. “Polytopic Quasi-LPV Models Based on Neural State-Space Models and Application to Air Charge Control of a SI Engine”. In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 6466–6471. DOI: 10.3182/20080706-5-KR-1001.01090.

[Bah+16] A. S. Bahman, K. Ma, P. Ghimire, F. Iannuzzo, and F. Blaabjerg. “A 3-D-Lumped Thermal Network Model for Long-Term Load Profiles Analysis in High-Power IGBT Modules”. In: *IEEE Journal of Emerging and Selected Topics in Power Electronics* 4.3 (2016), pp. 1050–1063. DOI: 10.1109/JESTPE.2016.2531631.

[Bau+08] W. Baumann, S. Schaum, K. Roepke, and M. Knaak. “Excitation Signals for Nonlinear Dynamic Modeling of Combustion Engines”. In: *Proceedings of the 17th World Congress, The International Federation of Automatic Control, Seoul, Korea*. 2008, pp. 1066–1067.

[Bay+18] A. G. Baydin, B. A. Pearlmutter, A. Radul, and J. M. Siskind. “Automatic differentiation in machine learning: A survey”. In: *Journal of Machine Learning Research* 18 (2018), pp. 1–43. arXiv: 1502.05767 [cs.SC].

[BB12] J. Bergstra and Y. Bengio. “Random search for hyper-parameter optimization”. In: *Journal of Machine Learning Research* 13.2 (2012).

[BBO17] A. Borovykh, S. Bohte, and C. W. Oosterlee. *Conditional Time Series Forecasting with Convolutional Neural Networks*. 2017. arXiv: 1703.04691 [stat.ML].

[Bel66] R. Bellman. “Dynamic programming”. In: *Science* 153.3731 (1966), pp. 34–37. DOI: 10.1126/science.153.3731.34.

[Ber+07] T. L. Bergman, F. P. Incropera, D. P. DeWitt, and A. S. Lavine. *Fundamentals of Heat and Mass Transfer*. 6th. John Wiley & Sons, 2007.

[Ber+11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. “Algorithms for Hyper-Parameter Optimization”. In: *25th annual conference on neural information processing systems (NIPS 2011)*. Vol. 24. 2011.

[BFJ96] J. Boyan, D. Freitag, and T. Joachims. “A Machine Learning Architecture for Optimizing Web Search Engines”. In: *AAAI Workshop on Internet Based Information Systems*. 1996, pp. 1–8.

[Bil+19] B. Bilgin, J. Liang, M. V. Terzic, et al. “Modeling and Analysis of Electric Motors: State-of-the-Art Review”. In: *IEEE Transactions on Transportation Electrification* 5.3 (2019), pp. 602–617. DOI: 10.1109/TTE.2019.2931123.

[Bin17] A. Binder. *Elektrische Maschinen und Antriebe*. 2nd ed. Berlin Heidelberg: Springer Vieweg, 2017. DOI: 10.1007/978-3-662-53241-6.

[Bis06] C. M. Bishop. *Pattern recognition and machine learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[BK22] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. 2nd. Cambridge University Press, 2022. DOI: 10.1017/9781009089517.

[BKK18] S. Bai, J. Z. Kolter, and V. Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. 2018. arXiv: 1803.01271 [cs.LG].

[BL18] E. Barbarini and V. Le Troadec. *Tesla Model 3 Inverter with SiC Power Module from STMicroeletronics*. Tech. rep. SystemPlus Consulting, 2018, p. 100.

[Bla73] F. Blaschke. “Das Verfahren der Feldorientierung zur Regelung der Drehfeldmaschine”. PhD thesis. TU Braunschweig, 1973.

[BMH22] M. A. Buettner, N. Monzen, and C. M. Hackl. “Artificial Neural Network Based Optimal Feedforward Torque Control of Interior Permanent Magnet Synchronous Machines: A Feasibility Study and Comparison with the State-of-the-Art”. In: *Energies* 15.5 (2022). DOI: 10.3390/en15051838.

[Bog+06] A. Boglietti, A. Cavagnino, M. Parvis, and A. Vallan. “Evaluation of Radiation Thermal Resistances in Industrial Motors”. In: *IEEE Transactions on Industry Applications* Vol. 42, N (2006), pp. 688–693. DOI: 10.1109/TIA.2006.873655.

[Bog+09] A. Boglietti, A. Cavagnino, D. Staton, M. Shanel, M. Mueller, and C. Mejuto. “Evolution and Modern Approaches for Thermal Analysis of Electrical Machines”. In: *IEEE Transactions on Industrial Electronics* 56.3 (2009), pp. 871–882. DOI: 10.1109/TIE.2008.2011622.

[Bog+16] A. Boglietti, E. Carpaneto, M. Cossale, and S. Vaschetto. “Stator-Winding Thermal Models for Short-Time Thermal Transients: Definition and Validation”. In: *IEEE Transactions on Industrial Electronics* 63.5 (2016), pp. 2713–2721. DOI: 10.1109/TIE.2015.2511170.

[Bog+19] A. Boglietti, M. Cossale, M. Popescu, and D. A. Staton. “Electrical Machines Thermal Model: Advanced Calibration Techniques”. In: *IEEE Transactions on Industry Applications* 55.3 (2019), pp. 2620–2628. DOI: 10.1109/TIA.2019.2897264.

[Bol+08] S. Bolognani, S. Bolognani, L. Peretti, and M. Zigliotto. “Design and implementation of model predictive control for electrical motor drives”. In: *IEEE Transactions on Industrial Electronics* 56.6 (2008), pp. 1925–1936. DOI: 10.1109/TIE.2008.2007547.

[Bra+12] N. Bracikowski, M. Hecquet, P. Brochet, and S. V. Shirinskii. “Multiphysics Modeling of a Permanent Magnet Synchronous Machine by Using Lumped Models”. In: *IEEE Transactions on Industrial Electronics* 59.6 (2012), pp. 2426–2437. DOI: 10.1109/TIE.2011.2169640.

[Bre01] L. Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[BRG20] F. Birnkammer, C. Roth, and D. Gerling. “High Resolution Lumped Parameter Thermal Network for Load Cycle Analysis of Electrical Machines”. In: *2020 23rd International Conference on Electrical Machines and Systems (ICEMS)*. IEEE. 2020, pp. 1527–1532. DOI: 10.23919/ICEMS50442.2020.9291138.

[Bro+20a] A. Brosch, S. Hanke, O. Wallscheid, and J. Böcker. “Data-driven recursive least squares estimation for model predictive current control of permanent magnet synchronous motors”. In: *IEEE Transactions on Power Electronics* 36.2 (2020), pp. 2179–2190. DOI: 10.1109/TPEL.2020.3006779.

[Bro+20b] T. B. Brown, B. Mann, N. Ryder, et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. “Learning Long Term Dependencies with Gradient Descent is Difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181. arXiv: arXiv:1211.5063v2.

[CB17] M. Cuturi and M. Blondel. “Soft-DTW: A Differentiable Loss Function for Time-Series”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, 2017, pp. 894–903.

[CG16] T. Chen and C. Guestrin. “XGBoost: A Scalable Tree Boosting System”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785.

[CGK21] D. Czerwinski, J. Gęca, and K. Kolano. “Machine Learning for Sensorless Temperature Estimation of a BLDC Motor”. In: *Sensors* 21.14 (2021), p. 4655. DOI: 10.3390/s21144655.

[Cha+17] K.-T. Chau, J. Chaoqiang, W. Han, and C. H. T. Lee. “State-of-the-Art Electromagnetics Research in Electric and Hybrid Vehicles”. In: *Progress In Electromagnetics Research* 159 (2017), pp. 139–157. DOI: 10.2528/PIER17090407.

[Cha+19] S. Chakraborty, H.-N. Vu, M. M. Hasan, D.-D. Tran, M. E. Baghdadi, and O. Hegazy. “DC-DC converter topologies for electric vehicles, plug-in hybrid electric vehicles and fast charging stations: State of the art and future trends”. In: *Energies* 12.8 (2019), p. 1569. DOI: 10.3390/en12081569.

[Cha+20] L. Chang, W. Lee, T. M. Jahns, and J. Kim. “Comparative Analysis of PWM Power Losses in IPM Machines with Different Modulation Schemes Using Wide-Bandgap-Based Inverters”. In: *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*. 2020, pp. 3629–3636. DOI: 10.1109/ECCE44975.2020.9235591.

[Cha02] C. C. Chan. “The state of the art of electric and hybrid vehicles”. In: *Proceedings of the IEEE* 90.2 (2002), pp. 247–275. DOI: 10.1109/5.989873.

[Cha07] C. C. Chan. “The State of the Art of Electric, Hybrid, and Fuel Cell Vehicles”. In: *Proceedings of the IEEE* 95.4 (2007), pp. 704–718. DOI: 10.1109/JPROC.2007.892489.

[Cha88] Y. Chauvin. “A back-propagation algorithm with optimal use of hidden units”. In: *Advances in neural information processing systems* 1 (1988).

[Che+18] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, and A. Emadi. “Long Short-Term Memory Networks for Accurate State-of-Charge Estimation of Li-ion Batteries”. In: *IEEE Transactions on Industrial Electronics* 65.8 (Aug. 2018), pp. 6730–6739. DOI: 10.1109/TIE.2017.2787586.

[Che+19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. “Neural Ordinary Differential Equations”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2019, pp. 6571–6583. arXiv: 1806.07366 [cs.LG].

[Chi+18] J.-W. Chin, S.-W. Hwang, H.-J. Park, and J.-P. Hong. “Thermal Analysis and Verification of PMSM Using LPTN Considering Mechanical Components and Losses”. In: *2018 XIII International Conference on Electrical Machines (ICEM)*. 2018, pp. 1323–1329. DOI: 10.1109/ICELMACH.2018.8507164.

[Cho+14] K. Cho, B. van Merriënboer, C. Gulcehre, et al. “Learning Phrase Representations using RNN Encoder –Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. arXiv: 1406.1078 [cs.CL].

[Cho+15] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. *Attention-Based Models for Speech Recognition*. 2015. arXiv: 1506.07503 [cs.CL].

[Chu+14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[CK15] A. Chiba and K. Kiyota. “Review of research and development of switched reluctance motor for hybrid electrical vehicle”. In: *2015 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD)*. 2015, pp. 127–131. DOI: 10.1109/WEMDCD.2015.7194520.

[ĆOP12] M. Ćalasan, M. Ostojić, and D. Petrović. “The retardation method for bearings loss determination”. In: *International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion*. 2012, pp. 25–29. DOI: 10.1109/SPEEDAM.2012.6264383.

[Cor+08] P. Cortés, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodríguez. “Predictive Control in Power Electronics and Drives”. In: *IEEE Transactions on Industrial Electronics* 55.12 (2008), pp. 4312–4324. DOI: 10.1109/TIE.2008.2007480.

[CUH15] D. Clevert, T. Unterthiner, and S. Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2015. arXiv: 1511.07289 [cs.LG].

[Cuo+22] S. Cuomo, V. S. di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. *Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next*. 2022. arXiv: 2201.05624 [cs.LG].

[CV95] C. Cortes and V. Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018.

[DD16] R. Dash and P. K. Dash. "A hybrid stock trading framework integrating technical analysis with machine learning techniques". In: *The Journal of Finance and Data Science* 2.1 (2016), pp. 42–57. DOI: 10.1016/j.jfds.2016.03.002.

[Deg+22] J. Degrave, F. Felici, J. Buchli, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning". In: *Nature* 602.7897 (2022), pp. 414–419. DOI: 10.1038/s41586-021-04301-9.

[Dep87] M. Depenbrock. "Direct self-control (DSC) of inverter fed induktion machine". In: *1987 IEEE Power Electronics Specialists Conference*. 1987, pp. 632–641. DOI: 10.1109/PESC.1987.7077236.

[Dev+18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. arXiv: 1810.04805 [cs.CL].

[DG08] J. Dean and S. Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113. DOI: 10.1145/1327452.1327492.

[Di +22] L. Di Natale, B. Svetozarevic, P. Heer, and C. Jones. "Physically Consistent Neural Networks for building thermal modeling: Theory and analysis". In: *Applied Energy* 325 (2022), p. 119806. DOI: https://doi.org/10.1016/j.apenergy.2022.119806.

[Doe+18] J. Doerr, T. Attensperger, L. Wittmann, and T. Enzinger. "The new electric axle drives from Audi". In: *MTZ worldwide* 79.6 (2018), pp. 18–25. DOI: 10.1007/s38313-018-0042-4.

[Doe+20] J. Doerr, G. Fröhlich, A. Stroh, and M. Baur. "Das elektrische Antriebssystem mit Drei-Motor-Layout im Audi E-tron S". In: *MTZ Motortechnische Zeitschrift* 81.7 (2020), pp. 18–27. DOI: 10.1007/s38313-018-0042-4.

[Dol00] M. Dolen. "Modeling and estimation by structured neural networks for CNC machine tools". PhD thesis. The University of Wisconsin-Madison, 2000.

[Dom12] P. Domingos. "A few useful things to know about machine learning". In: *Communications of the ACM* 55.10 (2012), p. 78. DOI: 10.1145/2347736.2347755. arXiv: 9605103 [cs].

[Dor+11] D. G. Dorrell, M.-F. Hsieh, M. Popescu, L. Evans, D. A. Staton, and V. Grout. "A Review of the Design Issues and Techniques for Radial-Flux Brushless Surface and Internal Rare-Earth Permanent-Magnet Motors". In: *IEEE Transactions on Industrial Electronics* 58.9 (2011), pp. 3741–3757. DOI: 10.1109/TIE.2010.2089940.

[Doz16] T. Dozat. "Incorporating Nesterov Momentum into Adam". In: *ICLR Workshop* (2016).

[DPV20] R. W. De Doncker, D. W. J. Pulle, and A. Veltman. *Advanced Electrical Drives: Analysis, Modeling, Control*. Springer Nature, 2020. DOI: 10.1007/978-3-030-48977-9.

[DR09] M. P. Deisenroth and C. E. Rasmussen. "Efficient Reinforcement Learning for Motor Control". In: *Proceedings of the 10th International Workshop on Systems and Control*. Hluboká nad Vltavou, Czech Republic, 2009.

[DSC51] W. C. Duesterhoeft, M. W. Schulz, and E. Clarke. “Determination of instantaneous currents and voltages by means of alpha, beta, and zero components”. In: *Transactions of the American Institute of Electrical Engineers* 70.2 (1951), pp. 1248–1255. DOI: 10.1109/T-AIEE.1951.5060554.

[DZ19] W. Deng and S. Zuo. “Electromagnetic Vibration and Noise of the Permanent-Magnet Synchronous Motors for Electric Vehicles: An Overview”. In: *IEEE Transactions on Transportation Electrification* 5.1 (2019), pp. 59–70. DOI: 10.1109/TTE.2018.2875481.

[Ele16] Elektromaschinenbau. *Magnetische und technologische Eigenschaften vollveredelter Elektrobleche*. Ingenieurbüro für Elektro-Maschinenbau GmbH. 2016. URL: https://tinyurl.com/yc2h4rku (visited on 07/07/2023).

[Elm90] J. L. Elman. “Finding structure in time”. In: *Cognitive science* 14.2 (1990), pp. 179–211. DOI: 10.1016/0364-0213(90)90002-E.

[Eng+19] T. Engelhardt, J. Lange, S. Oechslen, and A. Heitmann. “Maximierung der Leistungsdichte elektrischer Maschinen durch elektromagnetische und thermische Maßnahmen”. In: *Elektrische Antriebstechnologie für Hybrid und Elektrofahrzeuge; Schäfer, H., Ed.; Expert Verlag GmbH: Würzburg, Germany* (2019), pp. 13–24.

[Fan+08] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. “LIBLINEAR: A library for large linear classification”. In: *Journal of Machine Learning Research* 9 (2008), pp. 1871–1874.

[Far+23] S. A. Faroughi, N. Pawar, C. Fernandes, et al. *Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks in Scientific Computing*. 2023. arXiv: 2211.07377 [cs.LG].

[Fer+19] D. Fernandez, M. Martinez, D. Reigosa, A. B. Diez, J. M. Guerrero, and F. Briz. “Comparative Analysis of Magnet Thermal and Magnetization State Monitoring in PMSMs Based on High Frequency Signal Injection”. In: *IEEE Transactions on Industry Applications* (2019). DOI: 10.1109/TIA.2019.2952027.

[FES04] B. Fahimi, A. Emadi, and R. B. Sepe. “A switched reluctance machine-based starter/alternator for more electric cars”. In: *IEEE Transactions on Energy Conversion* 19.1 (2004), pp. 116–124. DOI: 10.1109/TEC.2003.822322.

[FFG88] J. A. Feldman, M. A. Fanty, and N. H. Goodard. “Computing with Structured Neural Networks”. In: *Computer* 21.3 (1988), pp. 91–103. DOI: 10.1109/2.34.

[Fin09] T. Finch. “Incremental Calculation of Weighted Mean and Variance”. In: *University of Cambridge* 4.11-5 (2009), pp. 41–42.

[Fra20] A. L. Fradkov. “Early History of Machine Learning”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1385–1390. DOI: 10.1016/j.ifacol.2020.12.1888.

[Gan+11] M. Ganchev, C. Kral, H. Oberguggenberger, and T. Wolbank. “Sensorless Rotor Temperature Estimation of Permanent Magnet Synchronous Motor”. In: *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*. 2011, pp. 2018–2023. DOI: 10.1109/IECON.2011.6119449.

[Gar+07] P. Garcia, F. Briz, D. Raca, and R. D. Lorenz. “Saliency-Tracking-Based Sensorless Control of AC Machines Using Structured Neural Networks”. In: *IEEE Transactions on Industry Applications* 43.1 (2007), pp. 77–86. DOI: 10.1109/TIA.2006.887309.

[GBC16] I. A. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[GC09] T. S. Guzella and W. M. Caminhas. “A Review of Machine Learning Approaches to Spam Filtering”. In: *Expert Systems with Applications* 36.7 (2009), pp. 10206–10222. DOI: 10.1016/j.eswa.2009.02.037.

[GC99] F. A. Gers and F. Cummins. “Learning to Forget: Continual Prediction with LSTM”. In: *Ninth International Conference on Artificial Neural Networks*. Vol. 2. 1999, pp. 1–19. DOI: 10.1162/089976600300015015.

[GEW06] P. Geurts, D. Ernst, and L. Wehenkel. “Extremely randomized trees”. In: *Machine Learning* 63.1 (Apr. 2006), pp. 3–42. DOI: 10.1007/s10994-006-6226-1.

[Gha+17] I. Ghalehkhondabi, E. Ardjmand, G. R. Weckman, and W. A. Young. “An overview of energy demand forecasting methods published in 2005–2015”. In: *Energy Systems* 8.2 (2017), pp. 411–447. DOI: 10.1007/s12667-016-0203-y.

[GHN23] L. Glass, W. Hilali, and O. Nelles. “An Input-to-State Stable Virtual Sensor for Electric Motor Rotor Temperature”. In: *IFAC-PapersOnLine* 56.1 (2023). 12th IFAC Symposium on Nonlinear Control Systems NOLCOS 2022, pp. 240–245. DOI: https://doi.org/10.1016/j.ifacol.2023.02.041.

[Gie08] J. F. Gieras. *Advancements in electric machines*. Springer Science & Business Media, 2008. DOI: 10.1007/978-1-4020-9007-3.

[GK21] P.-O. Gronwald and T. A. Kern. “Traction Motor Cooling Systems: A Literature Review and Comparative Study”. In: *IEEE Transactions on Transportation Electrification* 7.4 (2021), pp. 2892–2913. DOI: 10.1109/TTE.2021.3075844.

[GKK10] M. Ganchev, B. Kubicek, and H. Kappeler. “Rotor Temperature Monitoring System”. In: *The XIX International Conference on Electrical Machines* (2010), pp. 1–5. DOI: 10.1109/ICELMACH.2010.5608051.

[GL21] L. Galli and C.-J. Lin. “A Study on Truncated Newton Methods for Linear Classification”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021). DOI: 10.1109/TNNLS.2020.3045836.

[GLS77] I. Gustavsson, L. Ljung, and T. Söderström. “Identification of Processes in Closed Loop — Identifiability and Accuracy Aspects”. In: *Automatica* 13.1 (1977), pp. 59–75. DOI: 10.1016/0005-1098(77)90009-7.

[Gon+19] Y. Gong, R. Hartley, X. Da, et al. “Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 4559–4566. DOI: 10.23919/ACC.2019.8814833.

[Goo+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680. arXiv: 1406.2661 [stat.ML].

[Gör17] D. Görges. “Relations between Model Predictive Control and Reinforcement Learning”. In: *IFAC-PapersOnLine* (2017). DOI: 10.1016/j.ifacol.2017.08.747.

[Gre+16] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. “LSTM: A Search Space Odyssey”. In: *IEEE Transactions on Neural Networks and Learning Systems.* 28.10 (2016), pp. 2222–2232. DOI: 10.1109/TNNLS.2016.2582924. arXiv: 1503.04069.

[GS86] D. E. Goldberg and M. P. Samtani. “Engineering optimization via genetic algorithm.” In: *Electronic Computation (1986)*. ASCE. 1986, pp. 471–482.

[Gu+18] J. Gu, Z. Wang, J. Kuen, et al. “Recent advances in convolutional neural networks”. In: *Pattern Recognition* 77 (2018), pp. 354–377. DOI: 10.1016/j.patcog.2017.10.013.

[GWB17] D. Gaona, O. Wallscheid, and J. Böcker. “Glocal Identification Methods for Low-Order Lumped Parameter Thermal Networks used in Permanent Magnet Synchronous Motors”. In: *2017 IEEE 12th International Conference on Power Electronics and Drive Systems (PEDS)*. IEEE. 2017, pp. 1–126. DOI: 10.1109/PEDS.2017.8289163.

[GWB20] E. Gedlu, O. Wallscheid, and J. Böcker. “Permanent Magnet Synchronous Machine Temperature Estimation using Low-Order Lumped-Parameter Thermal Network with Extended Iron Loss Model”. In: *The 10th International Conference on Power Electronics, Machines and Drives (PEMD 2020)*. 2020, pp. 937–942. DOI: 10.1049/icp.2021.1017.

[GWB21] E. Gedlu, O. Wallscheid, and J. Böcker. “Temperature Estimation of Electric Machines using a Hybrid Model of Feed-Forward Neural and Low-Order Lumped-Parameter Thermal Networks”. In: *2021 IEEE International Electric Machines Drives Conference (IEMDC)*. 2021, pp. 1–8. DOI: 10.1109/IEMDC47953.2021.9449548.

[Hae+21] R. Haeb-Umbach, J. Heymann, L. Drude, S. Watanabe, M. Delcroix, and T. Nakatani. “Far-Field Automatic Speech Recognition”. In: *Proceedings of the IEEE* 109.2 (Feb. 2021), pp. 124–148. DOI: 10.1109/JPROC.2020.3018668.

[Hah22] I. Hahn. *Wolfspeed Announces the Lucid Air is Using its SiC Power Modules*. eepower. 2022. URL: https://tinyurl.com/yv657dac (visited on 08/23/2023).

[Han+19] S. Hanke, S. Peitz, O. Wallscheid, S. Klus, J. Böcker, and M. Dellnitz. *Koopman Operator-Based Finite-Control-Set Model Predictive Control for Electrical Drives*. 2019. arXiv: 1804.00854 [math.OC].

[Has69] K. Hasse. “Zur Dynamik Drehzahlgeregelter Antriebe mit Stromrichtergespeisten Asynchron-Kurzschlussläufer-Maschinen”. PhD thesis. TH. Darmstadt, 1969.

[HBM] HBM. *T10F torque flange data sheet*. Hottinger Baldwin Messtechnik GmbH. URL: https://tinyurl.com/4by9et4p (visited on 07/07/2023).

[HCH12] D. A. Howey, P. R. N. Childs, and A. S. Holmes. “Air-Gap Convection in Rotating Electrical Machines”. In: *IEEE Transactions on Industrial Electronics* Vol. 59, N (2012), pp. 1367–1375. DOI: 10.1109/TIE.2010.2100337.

[He+15a] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[He+15b] K. He, X. Zhang, S. Ren, and J. Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on Imagenet Classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[Hei16] J. Heintze. *Lehrbuch zur Experimentalphysik Band 3: Elektrizität und Magnetismus*. Ed. by P. Bock. 1st ed. Springer Spektrum, Berlin, Heidelberg, 2016. DOI: 10.1007/978-3-662-48451-7.

[Hel+20] C. Helbing, K. Bennewitz, P. Lück, J. Tousen, and J. Peter. “The powertrain of the ID.CROZZ - Volkswagen expands the portfolio of the MEB”. In: *41st International Vienna Motor Symposium*. Wien, 2020.

[HG14] D. Huger and D. Gerling. “An advanced lifetime prediction method for permanent magnet synchronous machines”. In: *2014 International Conference on Electrical Machines (ICEM)*. IEEE. 2014, pp. 686–691. DOI: 10.1109/ICELMACH.2014.6960255.

[HK70] A. E. Hoerl and R. W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems”. In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: 10.1080/00401706.1970.10488634.

[HN16] T. O. Heinz and O. Nelles. “Vergleich von Anregungssignalen für Nichtlineare Identifikationsaufgaben”. In: *Proceedings 26. Workshop Computational Intelligence*. 2016, pp. 139–158.

[HN17] T. O. Heinz and O. Nelles. “Iterative Excitation Signal Design for Nonlinear Dynamic Black-box Models”. In: *Procedia computer science* 112 (2017), pp. 1054–1061. DOI: 10.1016/j.procs.2017.08.112.

[HN18] T. O. Heinz and O. Nelles. “Excitation Signal Design for Nonlinear Dynamic Systems with Multiple Inputs - A Data Distribution Approach”. In: *Automatisierungstechnik* 66.9 (2018), pp. 714–724. DOI: 10.1515/auto-2018-0027.

[Hoc+21] A. Hochlehnert, A. Terenin, S. Saemundsson, and M. Deisenroth. “Learning Contact Dynamics using Physically Structured Neural Networks”. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, 13–15 Apr 2021, pp. 2152–2160. arXiv: 2102.11206 [cs.LG].

[Hof+19] E. Hoffer, R. Banner, I. Golan, and D. Soudry. *Norm matters: efficient and accurate normalization schemes in deep networks*. 2019. arXiv: 1803.01814 [stat.ML].

[Hon+97] Y. Honda, T. Nakamura, T. Higaki, and Y. Takeda. “Motor design considerations and test results of an interior permanent magnet synchronous motor for electric vehicles”. In: *IAS’97. Conference Record of the 1997 IEEE Industry Applications Conference Thirty-Second IAS Annual Meeting*. Vol. 1. IEEE. 1997, pp. 75–82. DOI: 10.1109/IAS.1997.643011.

[HOT06] G. E. Hinton, S. Osindero, and Y.-W. Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527.

[Hot33] H. Hotelling. “Analysis of a Complex of Statistical Variables into Principal Components.” In: *Journal of educational psychology* 24.6 (1933), p. 417. DOI: 10.1037/h0071325.

[HS97] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[HSA22] S. Hosseini, A. Shahbandegan, and T. Akilan. “Deep Neural Network Modeling for Accurate Electric Motor Temperature Prediction”. In: *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2022, pp. 170–175. DOI: 10.1109/CCECE49351.2022.9918222.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. “Multilayer Feedforward Networks are Universal Approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. 2nd. Springer Verlag, 2009. DOI: 10.1007/b94608.

[Hua+13] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. “Learning Deep Structured Semantic Models for Web Search using Clickthrough Data”. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 2333–2338. DOI: 10.1145/2505515.2505665.

[HW15] C. Hoffmann and H. Werner. “A Survey of Linear Parameter-Varying Control Applications Validated by Experiments or High-Fidelity Simulations”. In: *IEEE Transactions on Control Systems Technology* 23.2 (2015), pp. 416–433. DOI: 10.1109/TCST.2014.2327584.

[HZS06] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. “Extreme Learning Machine: Theory and Applications”. In: *Neurocomputing* 70.1-3 (2006), pp. 489–501. DOI: 10.1016/j.neucom.2005.12.126.

[Ian+20] G. Ianucs, D. Cojocaru, M. C. Opricsan, V. Paleu, and D. N. Olaru. “Power loss in grease lubricated ball bearings”. In: *{IOP} Conference Series: Materials Science and Engineering* 724.1 (Jan. 2020), p. 12009. DOI: 10.1088/1757-899x/724/1/012009.

[IC15] M. Z. Islam and S. Choi. “Design of five-phase ferrite magnet assisted synchronous reluctance motor using lumped parameter model based optimizer and FEA”. In: *2015 IEEE International Electric Machines Drives Conference (IEMDC)*. 2015, pp. 689–695. DOI: 10.1109/IEMDC.2015.7409134.

[Ilt20] R. Iltner. *Temperaturschätzung in Elektromotoren mittels AutoML*. Tech. rep. Paderborn University, 2020.

[IM11] R. Isermann and M. Münchhof. *Identification of dynamic systems: an introduction with applications*. 1st. Vol. 85. Springer, 2011. DOI: 10.1007/978-3-540-78879-9.

[Iru+20] I. Irurzun-Arana, C. Rackauckas, T. O. McDonald, and I. F. Trocóniz. "Beyond deterministic models in drug discovery and development". In: *Trends in pharmacological sciences* 41.11 (2020), pp. 882–895. DOI: 10.1016/j.tips.2020.09.005.

[IS15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning – Volume 37*. ICML 2015. Lille, France: JMLR, 2015, pp. 448–456.

[ISO18] ISO. *ISO 26262:2018 - Road Vehicles - Functional Safety*. International Organization for Standardization. 2018. URL: https://www.iso.org/standard/68383.html (visited on 07/07/2023).

[Iwa+08] S. Iwasaki, R. Deodhar, Y. Liu, A. Pride, Z. Q. Zhu, and J. Bremner. "Influence of PWM on the proximity loss in permanent magnet brushless AC machines". In: *2008 IEEE Industry Applications Society Annual Meeting*. IEEE. 2008, pp. 1–8. DOI: 10.1109/08IAS.2008.53.

[Jae01] H. Jaeger. *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks - with an Erratum note*. Tech. rep. 34. 2001, p. 13.

[Jah87] T. M. Jahns. "Flux-Weakening Regime Operation of an Interior Permanent-Magnet Synchronous Motor Drive". In: *IEEE Transactions on Industry Applications* IA-23.4 (1987), pp. 681–689. DOI: 10.1109/TIA.1987.4504966.

[Jou+18] N. Jouppi, C. Young, N. Patil, and D. Patterson. "Motivation for and Evaluation of the First Tensor Processing Unit". In: *IEEE Micro* 38.3 (2018), pp. 10–19. DOI: 10.1109/MM.2018.032271057.

[JSW98] D. R. Jones, M. Schonlau, and W. J. Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13 (1998), pp. 455–492. DOI: 10.1023/A:1008306431147.

[Jun+11] C. Jungreuthmayer, T. Bauml, O. Winter, et al. "A detailed heat and fluid flow analysis of an internal permanent magnet synchronous machine by means of computational fluid dynamics". In: *IEEE Transactions on Industrial Electronics* 59.12 (2011), pp. 4568–4578. DOI: 10.1109/TIE.2011.2176696.

[Jun+21] H.-S. Jung, H. Kim, S.-K. Sul, and D. J. Berry. "Temperature Estimation of IPMSM by Using Fundamental Reactive Energy Considering Variation of Inductances". In: *IEEE Transactions on Power Electronics* 36.5 (2021), pp. 5771–5783. DOI: 10.1109/TPEL.2020.3028084.

[JZS15] R. Jozefowicz, W. Zaremba, and I. Sutskever. "An Empirical Exploration of Recurrent Network Architectures". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 2342–2350.

[Kam06] M. Kamiya. "Development of Traction Drive Motors for the Toyota Hybrid System". In: *IEEJ Transactions on Industry Applications* 126.4 (2006), pp. 473–479. DOI: 10.1541/ieejias.126.473.

[Kat+20] T. Kato, K. Sasaki, D. F. Laborda, D. F. Alonso, and D. D. Reigosa. "Magnet Temperature Estimation Methodology by using Magnet Flux Linkage Observer for Variable Leakage Flux IPMSM". In: *IEEJ Journal of Industry Applications* 9.6 (2020), pp. 723–730. DOI: 10.1541/ieejjia.19008074.

[KB17] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[KBP13] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274. DOI: 10.1177/0278364913495721.

[Ke+17] G. Ke, Q. Meng, T. Finley, et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[KE95] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[KL51] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694.

[Koh95] R. Kohavi. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Vol. 2. Morgan Kaufmann, 1995, pp. 1137–1143.

[KOL19] J. Kim, J. Oh, and H. Lee. "Review on battery thermal management system for electric vehicles". In: *Applied Thermal Engineering* 149 (2019), pp. 192–212. DOI: 10.1016/j.applthermaleng.2018.12.020.

[Kou+15] S. Kouro, M. A. Perez, J. Rodriguez, A. M. Llor, and H. A. Young. "Model predictive control: MPC's role in the evolution of power electronics". In: *IEEE Industrial Electronics Magazine* 9.4 (2015), pp. 8–21. DOI: 10.1109/MIE.2015.2478920.

[Kri14] A. Krings. "Iron Losses in Electrical Machines-Influence of Material Properties, Manufacturing Processes, and Inverter Operation". PhD thesis. KTH Royal Institute of Technology, 2014.

[KSH12] A. Krizhevsky, I. Sulskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information and Processing Systems (NIPS)*. Vol. 60. 6. 2012, pp. 84–90. DOI: 10.1145/3065386.

[KW19] D. P. Kingma and M. Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. DOI: 10.1561/2200000056.

[LA22] J. Li and T. Akilan. "Global Attention-based Encoder-Decoder LSTM Model for Temperature Prediction of Permanent Magnet Synchronous Motors". In: (2022). arXiv: 2208.00293 [cs.LG].

[Lac+06] A. Lacerda, M. Cristo, M. A. Gonçalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. "Learning to advertise". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 2006, pp. 549–556. DOI: 10.1145/1148170.1148265.

[Lam18] F. Lambert. *Tesla Model 3 dual motor performance version features both an AC induction and a permanent magnet motor*. 2018. URL: https://tinyurl.com/4hx5tcww (visited on 07/07/2023).

[Laz+18] N. Lazic, C. Boutilier, T. Lu, et al. "Data center cooling using model-predictive control". In: *Advances in Neural Information Processing Systems* 31 (2018).

[Leb+13] K. Lebeau, J. Van Mierlo, P. Lebeau, O. Mairesse, and C. Macharis. "Consumer attitudes towards battery electric vehicles: a large-scale survey". In: *International Journal of Electric and Hybrid Vehicles* 5.1 (2013), pp. 28–41. DOI: 10.1504/IJEHV.2013.053466.

[LeC+89] Y. LeCun, B. Boser, J. S. Denker, et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[LeC+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[Lee+08] G.-H. Lee, S.-I. Kim, J.-P. Hong, and J.-H. Bahn. “Torque Ripple Reduction of Interior Permanent Magnet Synchronous Motor Using Harmonic Injected Current”. In: *IEEE Transactions on Magnetics* 44.6 (2008), pp. 1582–1585. DOI: 10.1109/TMAG.2008.915776.

[Lee+14] C. H. T. Lee, K. T. Chau, C. Liu, T. W. Ching, and F. Li. “A High-Torque Magnetless Axial-Flux Doubly Salient Machine for In-Wheel Direct Drive Applications”. In: *IEEE Transactions on Magnetics* 50.11 (2014), pp. 1–5. DOI: 10.1109/TMAG.2014.2326682.

[Leo01] W. Leonhard. *Control of Electrical Drives*. Springer Science & Business Media, 2001. DOI: 10.1007/978-3-642-56649-3.

[LH20] J. Lee and J. Ha. “Temperature Estimation of PMSM Using a Difference-Estimating Feedforward Neural Network”. In: *IEEE Access* 8 (2020), pp. 130855–130865. DOI: 10.1109/ACCESS.2020.3009503.

[Li+23] S. Li, G. Wang, Y. Di, L. Wang, H. Wang, and Q. Zhou. “A physics-informed neural network framework to predict 3D temperature field without labeled data in process of laser metal deposition”. In: *Engineering Applications of Artificial Intelligence* 120 (2023), p. 105908. DOI: 10.1016/j.engappai.2023.105908.

[Lia+21] D. Liang, Z. Q. Zhu, Y. Zhang, et al. “A Hybrid Lumped-Parameter and Two-Dimensional Analytical Thermal Model for Electrical Machines”. In: *IEEE Transactions on Industry Applications* 57.1 (2021), pp. 246–258. DOI: 10.1109/TIA.2020.3029997.

[Lia05] T. W. Liao. “Clustering of Time Series Data — A Survey”. In: *Pattern recognition* 38.11 (2005), pp. 1857–1874. DOI: 10.1016/j.patcog.2005.01.025.

[Lie+21] E. Liegmann, T. Schindler, P. Karamanakos, A. Dietz, and R. Kennel. “UltraZohm—An Open-Source Rapid Control Prototyping Platform for Power Electronic Systems”. In: *2021 International Aegean Conference on Electrical Machines and Power Electronics (ACEMP) 2021 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*. 2021, pp. 445–450. DOI: 10.1109/OPTIM-ACEMP50812.2021.9590016.

[Lin+12] X. Lin, H. E. Perez, J. B. Siegel, et al. “Online parameterization of lumped thermal dynamics in cylindrical lithium ion batteries for core temperature estimation and health monitoring”. In: *IEEE Transactions on Control Systems Technology* 21.5 (2012), pp. 1745–1755. DOI: 10.1109/TCST.2012.2217143.

[Lin+96] T. Lin, B. G. Horne, P. Tino, and C. L. Giles. “Learning Long Term Dependencies in NARX Recurrent Neural Networks”. In: *IEEE Transactions on Neural Networks* 7.6 (1996), pp. 1–23. DOI: 10.1109/72.548162.

[Lip+19] C. F. Lipinski, V. G. Maltarollo, P. R. Oliveira, A. B. F. Da Silva, and K. M. Honorio. “Advances and perspectives in applying deep learning for drug design and discovery”. In: *Frontiers in Robotics and AI* 6 (2019), p. 108. DOI: 10.3389/frobt.2019.00108.

[Lip18] Z. C. Lipton. “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” In: *Queue* 16.3 (2018), pp. 31–57. DOI: 10.1145/3236386.3241340.

[Liu+16] X. Liu, H. Chen, J. Zhao, and A. Belahcen. “Research on the Performances and Parameters of Interior PMSM Used for Electric Vehicles”. In: *IEEE Transactions on Industrial Electronics* 63.6 (2016), pp. 3533–3545. DOI: 10.1109/TIE.2016.2524415.

[LKH16] J. Lei Ba, J. R. Kiros, and G. E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[LL17] S. M. Lundberg and S.-I. Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4768–4777.

[LVD14] J. Lemmens, P. Vanassche, and J. Driesen. “Optimal Control of Traction Motor Drives Under Electrothermal Constraints”. In: *IEEE Journal of Emerging and Selected Topics in Power Electronics* 2.2 (2014), pp. 249–263. DOI: 10.1109/JESTPE.2014.2299765.

[LWH90] K. Lang, A. Waibel, and G. E. Hinton. “A Time-Delay Neural Network Architecture for Isolated Word Recognition”. In: *Neural Networks* 3.1 (1990), pp. 23–43. DOI: 10.1016/0893-6080(90)90044-L.

[Ma+21] Y. Ma, V. Dixit, M. J. Innes, X. Guo, and C. Rackauckas. “A Comparison of Automatic Differentiation and Continuous Sensitivity Analysis for Derivatives of Differential Equation Solutions”. In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE. 2021, pp. 1–9. DOI: 10.1109/HPEC49654.2021.9622796.

[Maa97] W. Maass. “Networks of Spiking Neurons: The Third Generation of Neural Network Models”. In: *Neural Networks* 10.9 (1997), pp. 1659–1671. DOI: 10.1016/S0893-6080(97)00011-7.

[Mad+19] V. Madonna, P. Giangrande, L. Lusuardi, A. Cavallini, C. Gerada, and M. Galea. “Thermal overload and insulation aging of short duty cycle, aerospace motors”. In: *IEEE Transactions on Industrial Electronics* 67.4 (2019), pp. 2618–2629. DOI: 10.1109/TIE.2019.2914630.

[Mal+14] S. von Malottki, M. Gregor, A. Wanke, and K. Hameyer. “Magnet design based on transient behavior of an IPMSM in event of malfunction”. In: *2014 International Conference on Electrical Machines (ICEM)*. IEEE. 2014, pp. 1262–1266. DOI: 10.1109/ICELMACH.2014.6960344.

[Mal16] I. Malmgren. “Quantifying the Societal Benefits of Electric Vehicles”. In: *World Electric Vehicle Journal* 8.4 (2016), pp. 996–1007. DOI: 10.3390/wevj8040996.

[Mar21] Marklines.com. *Tesla Model Y Teardown: Electric Powertrain Technology*. Marklines. 2021. URL: https://www.marklines.com/en/report/Munro007_202105 (visited on 08/23/2023).

[Mas+21] F. Masi, I. Stefanou, P. Vannucci, and V. Maffi-Berthier. “Thermodynamics-based Artificial Neural Networks for constitutive modeling”. In: *Journal of the Mechanics and Physics of Solids* 147 (2021), p. 104277. DOI: 10.1016/j.jmps.2020.104277.

[Mas51] F. Massey Jr. “The Kolmogorov-Smirnov Test for Goodness of Fit”. In: *Journal of the American Statistical Association* 46.253 (1951), pp. 68–78. DOI: 10.1080/01621459.1951.10500769.

[Meh74] R. Mehra. “Optimal input signals for parameter estimation in dynamic systems–Survey and new results”. In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 753–768. DOI: 10.1109/TAC.1974.1100701.

[Mej+08] C. Mejuto, M. Mueller, M. Shanel, A. Mebarki, M. Reekie, and D. Staton. “Improved Synchronous Machine Thermal Modelling”. In: *8th International Conference on Electrical Machines*. 2008, pp. 1–6. DOI: 10.1109/ICELMACH.2008.4799886.

[Mey10] M. Meyer. “Wirkungsgradoptimierte Regelung hoch ausgenutzter Permanentmagnet-Synchronmaschinen im Antriebsstrang von Automobilen”. Dissertation. Paderborn University, 2010.

[MGA14] D. T. McRuer, D. Graham, and I. Ashkenas. *Aircraft Dynamics and Automatic Control*. Vol. 740. Princeton University Press, 2014.

[Min+21] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. “Image segmentation using deep learning: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* (2021). DOI: 10.1109/TPAMI.2021.3059968.

[MNM02] W. Maass, T. Natschläger, and H. Markram. “Real-Time Computing Without Stable States: A new Framework for Neural Computation Based on Perturbations”. In: *Neural computation* 14.11 (2002), pp. 2531–2560. DOI: 10.1162/089976602760407955.

[Mor06] J. J. Moré. “The Levenberg-Marquardt algorithm: implementation and theory”. In: *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*. Springer. 2006, pp. 105–116. DOI: 10.1007/BFb0067700.

[MP69] M. Minsky and S. Papert. “An Introduction to Computational Geometry”. In: *Cambridge tiass., HIT* 479 (1969), p. 480. DOI: 10.7551/mitpress/11301.001.0001.

[MS-] MS-Schramberg. *Data sheet: NdFeB 210/250 h Seltenerdmagnete (anisotrop)*. MS-Schramberg GmbH and Co. KG. URL: https://tinyurl.com/2yepw8b7 (visited on 07/07/2023).

[MSB03] C. Mi, G. R. Slemon, and R. Bonert. “Modeling of Iron Losses of Permanent-Magnet Synchronous Motors”. In: *IEEE Transactions on Industry Applications* 39.3 (2003), pp. 734–742. DOI: 10.1109/TIA.2003.810635.

[Mün+20] T. Münker, G. Kampmann, M. Schüssler, and O. Nelles. “System Identification and Control of a Polymer Reactor”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 437–442. DOI: 10.1016/j.ifacol.2020.12.212.

[Myl+04] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown. “An Introduction to Decision Tree Modeling”. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 18.6 (2004), pp. 275–285. DOI: 10.1002/cem.873.

[MZ22] T. Meng and P. Zhang. “A Review of Thermal Monitoring Techniques for Radial Permanent Magnet Machines”. In: *Machines* 10.1 (2022). DOI: 10.3390/machines10010018.

[Nat+13] S. Nategh, Z. Huang, A. Krings, O. Wallmark, and M. Leksell. “Thermal Modeling of Directly Cooled Electric Machines Using Lumped Parameter and Limited CFD Analysis”. In: *IEEE Transactions on Energy Conversion* 28.4 (2013), pp. 979–990. DOI: 10.1109/TEC.2013.2283089.

[Nat+19] S. Nategh, H. Zhang, O. Wallmark, A. Boglietti, T. Nassen, and M. Bazant. “Transient Thermal Modeling and Analysis of Railway Traction Motors”. In: *IEEE Transactions on Industrial Electronics* 66 (2019), pp. 79–89. DOI: 10.1109/TIE.2018.2821619.

[Nel01] O. Nelles. *Nonlinear System Identification*. 1st ed. Springer Berlin, Heidelberg, 2001. DOI: 10.1007/978-3-662-04323-3.

[NH10] V. Nair and G. E. Hinton. “Rectified linear units improve Restricted Boltzmann machines”. In: *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*. 2010.

[Nie16] D. Nielsen. “Tree Boosting With XGBoost - Why Does XGBoost Win "Every" Machine Learning Competition?” MA thesis. NTNU: Norwegian University of Science and Technology, 2016.

[Noe+23] M. M. Noel, A. L, A. Trivedi, and P. Dutta. “Growing Cosine Unit: A Novel Oscillatory Activation Function That Can Speedup Training and Reduce Parameters in Convolutional Neural Networks”. In: (2023). arXiv: 2108.12943 [cs.LG].

[Not20] C. Nottelmann. “Untersuchung von Gradient Boosting Machines zur Online-Temperaturschätzung von wichtigen Komponenten in einem PMSM”. MA thesis. Paderborn University, 2020.

[NW06] J. Nocedal and S. J. Wright. *Numerical optimization*. 2nd ed. Springer, 2006. DOI: 10.1007/978-0-387-40065-5.

[Ope23] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[Owe+08] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. “GPU Computing”. In: *Proceedings of the IEEE* 96.5 (2008), pp. 879–899. DOI: 10.1109/JPROC.2008.917757.

[PAC18] I. Portugal, P. Alencar, and D. Cowan. “The use of Machine Learning Algorithms in Recommender Systems: A Systematic Review”. In: *Expert Systems with Applications* 97 (2018), pp. 205–227. DOI: 10.1016/j.eswa.2017.12.020.

[Par+21] S.-H. Park, E.-C. Lee, J.-C. Park, S.-W. Hwang, and M.-S. Lim. “Prediction of Mechanical Loss for High-Power-Density PMSM Considering Eddy Current Loss of PMs and Conductors”. In: *IEEE Transactions on Magnetics* 57.2 (2021), pp. 1–5. DOI: 10.1109/TMAG.2020.3007439.

[Par29] R. H. Park. “Two-reaction theory of synchronous machines generalized method of analysis-part I”. In: *Transactions of the American Institute of Electrical Engineers* 48.3 (1929), pp. 716–727. DOI: 10.1109/T-AIEE.1929.5055275.

[Pas+19] A. Paszke, S. Gross, F. Massa, et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[PB15] M. Paradkar and J. Böcker. “3D Analytical Model for Estimation of Eddy Current Losses in the Magnets of IPM Machine Considering the Reaction Field of the Induced Eddy Currents”. In: *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*. 2015, pp. 2862–2869. DOI: 10.1109/ECCE.2015.7310061.

[Pea01] K. Pearson. “On Lines and Planes of Closest Fit to Systems of Points in Space”. In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Pet+17] I. Petrov, D. Egorov, J. Link, R. Stern, S. Ruoho, and J. Pyrhönen. “Hysteresis Losses in Different Types of Permanent Magnets Used in PMSMs”. In: *IEEE Transactions on Industrial Electronics* 64.3 (2017), pp. 2502–2510. DOI: 10.1109/TIE.2016.2548440.

[Pet14] W. Peters. “Wirkungsgradoptimale Regelung von permanenterregten Synchronmotoren in automobilen Traktionsanwendungen unter Berücksichtigung der magnetischen Sättigung”. Dissertation. Paderborn University, 2014.

[Pin+18] S. Pint, N. Ardey, G. Mendl, et al. *Das neue vollelektrische Antriebssystem von Audi*. Tech. rep. VDI, 2018. DOI: 10.1007/s41491-018-0001-z.

[Pin89] F. J. Pineda. “Recurrent Backpropagation and the Dynamical Approach to Adaptive Neural Computation”. In: *Neural Computation* 1.2 (1989), pp. 161–172. DOI: 10.1162/neco.1989.1.2.161.

[PKB07] R. Poli, J. Kennedy, and T. Blackwell. *Particle swarm optimization*. Vol. 1. 1. London [u.a.] : ISTE, Nov. 2007, pp. 33–57. DOI: 10.1007/s11721-007-0002-0.

[PMB13] R. Pascanu, T. Mikolov, and Y. Bengio. “On the difficulty of training recurrent neural networks”. In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1310–1318.

[Pon87] L. S. Pontryagin. *Mathematical Theory of Optimal Processes*. CRC press, 1987.

[PPP12] P. Ponomarev, M. Polikarpova, and J. Pyrhönen. “Thermal modeling of directly-oil-cooled permanent magnet synchronous machine”. In: *2012 XXth International Conference on Electrical Machines*. 2012, pp. 1882–1887. DOI: 10.1109/ICElMach.2012.6350138.

[Pro63] A. I. Propoi. “Application of linear programming methods for the synthesis of automatic sampled-data systems”. In: *Avtomatika i Telemekhanika* 24 (1963), pp. 912–920.

[PS12] R. Pintelon and J. Schoukens. *System Identification: a Frequency Domain Approach*. John Wiley & Sons, 2012.

[PSA20] A. Pal, S. Saha, and R. Anita. “Musenet : Music Generation using Abstractive and Generative Methods”. In: *International Journal of Innovative Technology and Exploring Engineering* 9.6 (Apr. 2020), pp. 784–788. DOI: 10.35940/ijitee.f3580.049620.

[PWB12] W. Peters, O. Wallscheid, and J. Böcker. “A precise open-loop torque control for an interior permanent magnet synchronous motor (IPMSM) considering iron losses”. In: *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*. IEEE. 2012, pp. 2877–2882. DOI: 10.1109/IECON.2012.6389438.

[Qi+15] F. Qi, A. Stippich, S. Koschik, and R. W. De Doncker. “Model Predictive Overload Control of Induction Motors”. In: *2015 IEEE International Electric Machines & Drives Conference (IEMDC)*. IEEE. 2015, pp. 999–1005. DOI: 10.1109/IEMDC.2015.7409183.

[Qi+16] F. Qi, D. A. Ly, C. van der Broeck, D. Yan, and R. W. De Doncker. “Model order reduction suitable for online linear parameter-varying thermal models of electric motors”. In: *2016 IEEE 2nd Annual Southern Power Electronics Conference (SPEC)*. 2016, pp. 1–6. DOI: 10.1109/SPEC.2016.7846147.

[QSD14] F. Qi, M. Schenk, and R. W. De Doncker. “Discussing Details of Lumped Parameter Thermal Modeling in Electrical Machines”. In: *7th IET International Conference on Power Electronics, Machines and Drives (PEMD 2014)*. 2014, pp. 1–6. DOI: 10.1049/cp.2014.0479.

[Rac+21] C. Rackauckas, Y. Ma, J. Martensen, et al. *Universal Differential Equations for Scientific Machine Learning*. 2021. arXiv: 2001.04385 [cs.LG].

[Ram+13] R. Ramakrishnan, A. Gebregergis, M. Islam, and T. Sebastian. “Effect of position sensor error on the performance of PMSM drives for low torque ripple applications”. In: *2013 International Electric Machines & Drives Conference*. IEEE. 2013, pp. 1166–1173. DOI: 10.1109/IEMDC.2013.6556307.

[Ram+22] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. “Hierarchical Text-Conditional Image Generation with CLIP Latents”. ArXiv e-prints: 2204.06125[cs.CV], 2022. DOI: 10.48550/ARXIV.2204.06125.

[Reb16] V. Reber. “E-power: New possibilities with 800-volt charging”. In: *Porsche Eng. Mag.* (2016), pp. 10–15.

[Rei+10] D. D. Reigosa, F. Briz, P. Garcia, J. M. Guerrero, and M. W. Degner. “Magnet Temperature Estimation in Surface PM Machines Using High-Frequency Signal Injection”. In: *IEEE Transactions on Industry Applications* 46.4 (2010), pp. 1468–1475. DOI: 10.1109/TIA.2010.2049816.

[Rei+15] D. D. Reigosa, D. Fernandez, H. Yoshida, T. Kato, and F. Briz. “Permanent-Magnet Temperature Estimation in PMSMs Using Pulsating High-Frequency Current Injection”. In: *IEEE Transactions on Industry Applications* 51.4 (2015), pp. 3159–3168. DOI: 10.1109/TIA.2015.2404922.

[Rei+16] D. D. Reigosa, D. Fernandez, T. Tanimoto, T. Kato, and F. Briz. “Permanent-Magnet Temperature Distribution Estimation in Permanent-Magnet Synchronous Machines using Back Electromotive Force Harmonics”. In: *IEEE Transactions on Industry Applications* 52.4 (2016), pp. 3093–3103. DOI: 10.1109/TIA.2016.2536579.

[Rei+19a] D. Reigosa, D. Fernández, M. Martínez, J. M. Guerrero, A. B. Diez, and F. Briz. “Magnet Temperature Estimation in Permanent Magnet Synchronous Machines Using the High Frequency Inductance”. In: *IEEE Transactions on Industry Applications* 55.3 (2019), pp. 2750–2757. DOI: 10.1109/TIA.2019.2895557.

[Rei+19b] J. Reimers, L. Dorn-Gomba, C. Mak, and A. Emadi. “Automotive Traction Inverters: Current Status and Future Trends”. In: *IEEE Transactions on Vehicular Technology* 68.4 (2019), pp. 3337–3350. DOI: 10.1109/TVT.2019.2897899.

[RH11] R. Rothe and K. Hameyer. “Life expectancy calculation for electric vehicle traction motors regarding dynamic temperature and driving cycles”. In: *2011 IEEE International Electric Machines & Drives Conference (IEMDC)*. IEEE. 2011, pp. 1306–1309. DOI: 10.1109/IEMDC.2011.5994793.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning Representations by Back-Propagating Errors”. In: *nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0.

[Rib+16] J.-R. Riba, C. López-Torres, L. Romeral, and A. Garcia. “Rare-earth-free propulsion motors for electric vehicles: A technology review”. In: *Renewable and Sustainable Energy Reviews* 57 (2016), pp. 367–379. DOI: 10.1016/j.rser.2015.12.121.

[RL19] P. Ramesh and N. C. Lenin. “High Power Density Electrical Machines for Electric Vehicles—Comprehensive Review Based on Material Technology”. In: *IEEE Transactions on Magnetics* 55.11 (2019), pp. 1–21. DOI: 10.1109/TMAG.2019.2929145.

[Rob59] S. W. Roberts. “Control Chart Tests Based on Geometric Moving Averages”. In: *Technometrics* 1.3 (1959), pp. 239–250. DOI: 10.1080/00401706.1959.10489860.

[Rom+22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[Ros58] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386. DOI: 10.1037/h0042519.

[RP96] I. Rivals and L. Personnaz. “Black-Box Modeling With State-Space Neural Networks”. In: *Neural Adaptive Control Technology*. World Scientific, 1996, pp. 237–264. DOI: 10.1142/9789812830388_0008.

[RPK19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

[RS02] K. M. Rahman and S. E. Schulz. “Design of high-efficiency and high-torque-density switched reluctance motor for vehicle propulsion”. In: *IEEE Transactions on Industry Applications* 38.6 (2002), pp. 1500–1507. DOI: 10.1109/TIA.2002.805571.

[Ruo18] C. Ruoff. *Tesla’s top motor engineer talks about designing a permanent magnet machine for Model 3*. 2018. URL: https://tinyurl.com/zmm992rx (visited on 07/07/2023).

[RW05] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. DOI: 10.7551/mitpress/3206.001.0001.

[SA12] J. Serra and J. L. Arcos. "A Competitive Measure to Assess the Similarity Between two Time Series". In: *International Conference on Case-Based Reasoning*. Springer. 2012, pp. 414–427. DOI: 10.1007/978-3-642-32986-9_31.

[SA90] F. M. Silva and L. B. Almeida. "Acceleration techniques for the backpropagation algorithm". In: *European Association for Signal Processing Workshop*. Springer. 1990, pp. 110–119. DOI: 10.1007/3-540-52255-7_32.

[San+12] J. de Santiago, H. Bernhoff, B. Ekergård, et al. "Electrical Motor Drivelines in Commercial All-Electric Vehicles: A Review". In: *IEEE Transactions on Vehicular Technology* 61.2 (2012), pp. 475–484. DOI: 10.1109/TVT.2011.2177873.

[Sar+22] S. F. Sarcheshmeh, S.-E. Asmussen, N. Koenig, and M. Nienhaus. "Real-time AI-based Temperature Estimation of Small Electric Drives". In: *IKMT 2022; 13. GMM/ETG-Symposium*. 2022, pp. 1–6.

[SB18] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SB20] D. Schröder and J. Böcker. *Elektrische Antriebe - Regelung von Antriebssystemen*. 5th ed. Berlin: Springer Vieweg, 2020. DOI: 10.1007/978-3-662-62700-6.

[SC07] S. Salvador and P. Chan. "Toward accurate dynamic time warping in linear time and space". In: *Intelligent Data Analysis* 11.5 (2007), pp. 561–580.

[SD20] T. Schindler and A. Dietz. "Real-Time Inference of Neural Networks on FPGAs for Motor Control Applications". In: *2020 10th International Electric Drives Production Conference (EDPC)*. IEEE. 2020, pp. 1–6. DOI: 10.1109/EDPC51184.2020.9388185.

[Sei96] D. R. Seidl. "Motion and motor control using structured neural networks". PhD thesis. The University of Wisconsin-Madison, 1996.

[Sem] Semikron. *SKiiP 1242GB120-4D data sheet*. Semikron. URL: https://tinyurl.com/bdhet362 (visited on 07/07/2023).

[SFK13] M. Schweizer, T. Friedli, and J. W. Kolar. "Comparative Evaluation of Advanced Three-Phase Three-Level Inverter/Converter Topologies Against Two-Level Systems". In: *IEEE Transactions on Industrial Electronics* 60.12 (2013), pp. 5515–5527. DOI: 10.1109/TIE.2012.2233698.

[SH13] J. Schützhold and W. Hofmann. "Analysis of the Temperature Dependence of Losses in Electrical Machines". In: *2013 IEEE Energy Conversion Congress and Exposition*. IEEE. 2013, pp. 3159–3165. DOI: 10.1109/ECCE.2013.6647114.

[Sha+16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. "Taking the Human out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE*. Vol. 104. 1. 2016, pp. 148–175. DOI: 10.1109/JPROC.2015.2494218.

[Shv+10] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. "The hadoop distributed file system". In: *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. IEEE. 2010, pp. 1–10. DOI: 10.1109/MSST.2010.5496972.

[Sil+16] D. Silver, A. Huang, C. J. Maddison, et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961.

[Sit+20] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. "Implicit neural representations with periodic activation functions". In: *Advances in neural information processing systems* 33 (2020), pp. 7462–7473.

[SK08] F. Salewski and S. Kowalewski. “Hardware/Software Design Considerations for Automotive Embedded Systems”. In: *IEEE Transactions on Industrial Informatics* 4.3 (2008), pp. 156–163. DOI: 10.1109/TII.2008.2002919.

[SK15] A. G. Sarigiannidis and A. G. Kladas. “Switching Frequency Impact on Permanent Magnet Motors Drive System for Electric Actuation Applications”. In: *IEEE Transactions on Magnetics* 51.3 (2015), pp. 1–4. DOI: 10.1109/TMAG.2014.2358378.

[SK16] T. Salimans and D. P. Kingma. “Weight normalization: A simple reparameterization to accelerate training of deep neural networks”. In: *Advances in neural information processing systems* 29 (2016).

[SK21] O. Schwedes and M. Keichel. *The Electric Car: Mobility in Upheaval*. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2021. DOI: 10.1007/978-3-658-29760-2.

[SL19] J. Schoukens and L. Ljung. “Nonlinear System Identification: A User-Oriented Road Map”. In: *IEEE Control Systems Magazine* 39.6 (2019), pp. 28–99. DOI: 10.1109/MCS.2019.2938121.

[SLA12] J. Snoek, H. Larochelle, and R. P. Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Advances in neural information processing systems* 25 (2012).

[SM00] J. Shi and J. Malik. “Normalized Cuts and Image Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905. DOI: 10.1109/34.868688.

[SP97] M. Schuster and K. Paliwal. “Bidirectional recurrent neural networks”. In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[Spa+12] B. Spanfelner, D. Richter, S. Ebel, U. Wilhelm, W. Branz, and C. Patz. “Challenges in applying the ISO 26262 for driver assistance systems”. In: *Tagung Fahrerassistenz, München* 15.16 (2012), p. 2012.

[Spe14] A. Specht. “Ermittlung der Rotortemperatur einer Synchronmaschine mit eingebetteten Permanentmagneten für einen automobilen Traktionsantrieb mittels Beobachter basierend auf elektrischen Größen”. Dissertation. Paderborn University, 2014.

[SQC17] R. Salay, R. Queiroz, and K. Czarnecki. *An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software*. 2017. arXiv: 1709.02435 [cs.AI].

[SQW18] J. Shen, X. Qin, and Y. Wang. “High-Speed Permanent Magnet Electrical Machines — Applications, Key Issues and Challenges”. In: *CES Transactions on Electrical Machines and Systems* 2.1 (2018), pp. 23–33. DOI: 10.23919/TEMS.2018.8326449.

[SR95] S. B. Serpico and F. Roli. “Classification of Multisensor Remote-Sensing Images by Structured Neural Networks”. In: *IEEE Transactions on Geoscience and Remote Sensing* 33.3 (1995), pp. 562–578. DOI: 10.1109/36.387573.

[Sri+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[Sti+12] S. Stipetic, M. Kovacic, Z. Hanic, and M. Vrazic. “Measurement of Excitation Winding Temperature on Synchronous Generator in Rotation Using Infrared Thermography”. In: *IEEE Transactions on Industrial Electronics* 59.5 (2012), pp. 2288–2298. DOI: 10.1109/TIE.2011.2158047.

[Sto20] C. Stoll. “Optimization of Thermal Models of Electric Machines for Up-coming Electric Drive Systems”. MA thesis. University of Stuttgart, 2020.

[Sut+13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. “On the importance of initialization and momentum in deep learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147.

[SVV07] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. “An Overview of Reservoir Computing: Theory, Applications and Implementations”. In: *Proceedings of the 15th european symposium on artificial neural networks*. 2007, pp. 471–482.

[SW21] M. Schenke and O. Wallscheid. “A Deep Q-Learning Direct Torque Controller for Permanent Magnet Synchronous Motors”. In: *IEEE Open Journal of the Industrial Electronics Society* 2 (2021), pp. 388–400. DOI: 10.1109/OJIES.2021.3075521.

[SW95] E. D. Sontag and Y. Wang. “On characterizations of the input-to-state stability property”. In: *Systems and Control Letters* 24.5 (1995), pp. 351–359. DOI: 10.1016/0167-6911(94)00050-6.

[SWB14] A. Specht, O. Wallscheid, and J. Böcker. “Determination of Rotor Temperature for an Interior Permanent Magnet Synchronous Machine Using a Precise Flux Observer”. In: *International Power Electronics Conference*. 2014, pp. 1501–1507. DOI: 10.1109/IPEC.2014.6869784.

[SWS17] D. Shen, G. Wu, and H.-I. Suk. “Deep learning in medical image analysis”. In: *Annual review of biomedical engineering* 19 (2017), p. 221. DOI: 10.1007/978-3-030-33128-3_1.

[Sze+15] C. Szegedy, W. Liu, Y. Jia, et al. “Going deeper with convolutions”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[Tak+17] T. Takahashi, M. Takemoto, S. Ogasawara, W. Hino, and K. Takezaki. “Size and weight reduction of an in-wheel axial-gap motor using ferrite permanent magnets for electric commuter cars”. In: *IEEE transactions on industry applications* 53.4 (2017), pp. 3927–3935. DOI: 10.1109/TIA.2017.2684739.

[TBL18] M. Tschannen, O. Bachem, and M. Lucic. “Recent Advances in Autoencoder-Based Representation Learning”. In: (2018). arXiv: 1812.05069 [cs.LG].

[TH+12] T. Tieleman, G. Hinton, et al. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[Tib96] R. Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: 10.2307/2346178.

[TN86] I. Takahashi and T. Noguchi. “A New Quick-Response and High-Efficiency Control Strategy of an Induction Motor”. In: *IEEE Transactions on Industry Applications* IA-22.5 (1986), pp. 820–827. DOI: 10.1109/TIA.1986.4504799.

[Tra+20a] D.-D. Tran, M. Vafaeipour, M. El Baghdadi, R. Barrero, J. Van Mierlo, and O. Hegazy. “Thorough state-of-the-art analysis of electric and hybrid vehicle powertrains: Topologies and integrated energy management strategies”. In: *Renewable and Sustainable Energy Reviews* 119 (2020), p. 109596. DOI: 10.1016/j.rser.2019.109596.

[Tuc+19] K. Tucki, O. Orynycz, A. Świć, and M. Mitoraj-Wojtanek. “The Development of Electromobility in Poland and EU States as a Tool for Management of CO2 Emissions”. In: *Energies* 12.15 (2019), p. 2942. DOI: 10.3390/en12152942.

[TY23] J. Togelius and G. N. Yannakakis. *Choose Your Weapon: Survival Strategies for Depressed AI Academics*. 2023. arXiv: 2304.06035 [cs.OH].

[Van+21] J. Van Mierlo, M. Berecibar, M. El Baghdadi, et al. “Beyond the State of the Art of Electric Vehicles: A Fact-Based Paper of the Current and Prospective Electric Vehicle Technologies”. In: *World Electric Vehicle Journal* 12.1 (2021). DOI: 10.3390/wevj12010020.

[Vap63] V. Vapnik. “Pattern recognition using generalized portrait method”. In: *Automation and remote control* 24 (1963), pp. 774–780.

[Vas+17] A. Vaswani, N. Shazeer, N. Parmar, et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[Vaz+14] S. Vazquez, J. I. Leon, L. G. Franquelo, et al. “Model predictive control: A review of its applications in power electronics”. In: *IEEE industrial electronics magazine* 8.1 (2014), pp. 16–31. DOI: 10.1109/MIE.2013.2290138.

[Vin+19] O. Vinyals, I. Babuschkin, W. M. Czarnecki, et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: *Nature* 575.7782 (2019), pp. 350–354. DOI: 10.1038/s41586-019-1724-z.

[Vir+20] P. Virtanen, R. Gommers, T. E. Oliphant, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[VPF14] M. Van Der Geest, H. Polinder, and J. A. Ferreira. “Influence of PWM switching frequency on the losses in PM machines”. In: *2014 International Conference on Electrical Machines (ICEM)*. IEEE. 2014, pp. 1243–1247. DOI: 10.1109/ICELMACH.2014.6960341.

[Wal+] O. Wallscheid, W. Kirchgässner, M. Schenke, et al. *UPB LEA Reinforcement Learning Course Materials*. DOI: 10.5281/zenodo.8159061.

[Wal+16] O. Wallscheid, T. Huber, W. Peters, and J. Böcker. “A Critical Review of Techniques to Determine the Magnet Temperature of Permanent Magnet Synchronous Motors Under Real-Time Conditions”. In: *EPE Journal* 26 (2016), pp. 1–10. DOI: 10.1080/09398368.2016.1209877.

[Wal17] O. Wallscheid. “Ein Beitrag zur thermischen Ausnutzung permanenterregter Synchronmotoren in automobilen Traktionsanwendungen”. Dissertation. Paderborn University, 2017.

[Wal21] O. Wallscheid. “Thermal Monitoring of Electric Motors: State-of-the-Art Review and Future Challenges”. In: *IEEE Open Journal of Industry Applications* 2 (2021), pp. 204–223. DOI: 10.1109/OJIA.2021.3091870.

[Wan+10] J. Wang, K. Atallah, R. Chin, W. M. Arshad, and H. Lendenmann. “Rotor Eddy-Current Loss in Permanent-Magnet Brushless AC Machines”. In: *IEEE Transactions on Magnetics* 46.7 (2010), pp. 2701–2707. DOI: 10.1109/TMAG.2010.2042963.

[Wat+23] S. Watanabe, N. Awad, M. Onishi, and F. Hutter. *Speeding Up Multi-Objective Hyperparameter Optimization by Task Similarity-Based Meta-Learning for the Tree-Structured Parzen Estimator*. 2023. arXiv: 2212.06751 [cs.LG].

[Wat23] S. Watanabe. *Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance*. 2023. arXiv: 2304.11127 [cs.LG].

[WB15] O. Wallscheid and J. Böcker. “Design and Identification of a Lumped-Parameter Thermal Network for Permanent Magnet Synchronous Motors Based on Heat Transfer Theory and Particle Swarm Optimisation”. In: *17th European Conference on Power Electronics and Applications*. 2015, pp. 1–10. DOI: 10.1109/EPE.2015.7311718.

[WB16a] O. Wallscheid and J. Böcker. “Global Identification of a Low-Order Lumped-Parameter Thermal Network for Permanent Magnet Synchronous Motors”. In: *IEEE Transactions on Energy Conversion* 31.1 (2016), pp. 354–365. DOI: 10.1109/TEC.2015.2473673.

[WB16b] J. West and M. Bhattacharya. “Intelligent financial fraud detection: A comprehensive review”. In: *Computers & Security* 57 (2016), pp. 47–66. DOI: 10.1016/j.cose.2015.09.005.

[WB17] O. Wallscheid and J. Böcker. “Derating of automotive drive systems using model predictive control”. In: *IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*. 2017, pp. 31–36. DOI: 10.1109/PRECEDE.2017.8071264.

[WD21] M. Wang and W. Deng. “Deep face recognition: A survey”. In: *Neurocomputing* 429 (2021), pp. 215–244. DOI: 10.1016/j.neucom.2020.10.081.

[Wer90] P. J. Werbos. “Backpropagation Through Time: What It Does and How to Do It”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[Wie21] M. Wienkötter. *Der Antrieb: Performance pur*. 2021. URL: https://tinyurl.com/9zymz3pc (visited on 09/10/2021).

[Win19] F. Winkel. “Evaluierung Optimaler Anregungsprofile für einen permanentmagneterregten Synchronmotor zur Erhöhung der Datendiversifikation im Kontext des Trainings neuronaler Netze”. MA thesis. Paderborn University, 2019.

[WMB15] O. Wallscheid, M. Meyer, and J. Böcker. “An Open-Loop Operation Strategy for Induction Motors Considering Iron Losses and Saturation Effects in Automotive Applications”. In: *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*. IEEE. 2015, pp. 981–985. DOI: 10.1109/PEDS.2015.7203404.

[Wöc+20] D. Wöckinger, G. Bramerdorfer, S. Drexler, et al. “Measurement-Based Optimization of Thermal Networks for Temperature Monitoring of Outer Rotor PM Machines”. In: *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*. 2020, pp. 4261–4268. DOI: 10.1109/ECCE44975.2020.9236388.

[Wol+20] T. Wolf, L. Debut, V. Sanh, et al. “Transformers: State-of-the-art natural language processing”. In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

[WP90] R. J. Williams and J. Peng. “An Efficient Gradient-Based Algorithm for On-line Training of Recurrent Network Trajectories”. In: *Neural computation* 2.4 (1990), pp. 490–501. DOI: 10.1162/neco.1990.2.4.490.

[Wro+13] R. Wrobel, D. E. Salt, A. Griffo, N. Simpson, and P. H. Mellor. “Derivation and scaling of AC copper loss in thermal modeling of electrical machines”. In: *IEEE Transactions on Industrial Electronics* 61.8 (2013), pp. 4412–4420. DOI: 10.1109/TIE.2013.2266088.

[Wro+16] R. Wrobel, P. H. Mellor, M. Popescu, and D. A. Staton. “Power Loss Analysis in Thermal Design of Permanent-Magnet Machines—A Review”. In: *IEEE Transactions on Industry Applications* 52.2 (2016), pp. 1359–1368. DOI: 10.1109/TIA.2015.2489599.

[WSB17] O. Wallscheid, A. Specht, and J. Böcker. “Observing the Permanent-Magnet Temperature of Synchronous Motors Based on Electrical Fundamental Wave Model Quantities”. In: *IEEE Transactions on Industrial Electronics* 64.5 (2017), pp. 3921–3929. DOI: 10.1109/TIE.2017.2652363.

[Wu+19] P.-S. Wu, M.-F. Hsieh, W. L. Cai, et al. “Heat Transfer and Thermal Management of Interior Permanent Magnet Synchronous Electric Motor”. In: *Inventions* 4.4 (2019). DOI: 10.3390/inventions4040069.

[Wu+20] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita. “Adaptive stock trading strategies with deep reinforcement learning methods”. In: *Information Sciences* 538 (2020), pp. 142–158. DOI: 10.1016/j.ins.2020.05.066.

[WW01] L.-X. Wang and F. Wan. “Structured Neural Networks for Constrained Model Predictive Control”. In: *Automatica* 37.8 (2001). Neural Network Feedback Control, pp. 1235–1243. DOI: 10.1016/S0005-1098(01)00091-7.

[Xu+15a] K. Xu, J. Ba, R. Kiros, et al. *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*. ArXiv e-prints: 1502.03044, 2015.

[Xu+15b] K. Xu, J. L. Ba, R. Kiros, et al. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Vol. 37. ICML’15. Lille, France: JMLR.org, 2015, pp. 2048–2057.

[YI10] K. Yamazaki and H. Ishigami. “Rotor-Shape Optimization of Interior-Permanent-Magnet Motors to Reduce Harmonic Iron Losses”. In: *IEEE Transactions on Industrial Electronics* 57.1 (2010), pp. 61–69. DOI: 10.1109/TIE.2009.2025285.

[Yok] Yokogawa. *Precision Power Analyzer WT3000 Datasheet*. Yokogawa Electric Corporation. URL: https://tinyurl.com/2mstxf87 (visited on 07/07/2023).

[Yu+19] Y. Yu, X. Si, C. Hu, and J. Zhang. “A review of recurrent neural networks: LSTM cells and network architectures”. In: *Neural computation* 31.7 (2019), pp. 1235–1270. DOI: 10.1162/neco_a_01199.

[Zap+21] M. O. Zapico, D. D. Reigosa, D. F. Laborda, M. M. Gómez, J. M. G. Muñoz, and F. B. del Blanco. “Use HF Signal Injection for Simultaneous Rotor Angle, Torque and Temperature Estimation in PMSMs”. In: *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*. 2021, pp. 5084–5091. DOI: 10.1109/ECCE47101.2021.9595710.

[Zei21] Zeit-Online. *EU-Kommission will Benzin- und Dieselautos bis 2035 verbieten*. 2021. URL: https://tinyurl.com/wrbxvu5t (visited on 10/09/2021).

[ZF18] A. Zeaiter and M. Fénot. “Thermal Sensitivity Analysis of a High Power Density Electric Motor for Aeronautical Application”. In: *2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles International Transportation Electrification Conference (ESARS-ITEC)*. 2018, pp. 1–6. DOI: 10.1109/ESARS-ITEC.2018.8607393.

[ZH07] Z. Q. Zhu and D. Howe. “Electrical Machines and Drives for Electric, Hybrid, and Fuel Cell Vehicles”. In: *Proceedings of the IEEE* 95.4 (2007), pp. 746–765. DOI: 10.1109/JPROC.2006.892482.

[ZH21] N. Zobeiry and K. D. Humfeld. “A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications”. In: *Engineering Applications of Artificial Intelligence* 101 (May 2021). DOI: 10.1016/j.engappai.2021.104232.

[Zha+18] K. Zhang, A. Guliani, S. Ogrenci-Memik, et al. “Machine Learning-Based Temperature Prediction for Runtime Thermal Management Across System Components”. In: *IEEE Transactions on Parallel and Distributed Systems* 29.2 (2018), pp. 405–419. DOI: 10.1109/TPDS.2017.2732951.

[ZP21] G. Zheng and Z. Peng. “Life Cycle Assessment (LCA) of BEV’s environmental benefits for meeting the challenge of ICExit (Internal Combustion Engine Exit)”. In: *Energy Reports* 7 (2021), pp. 1203–1216. DOI: 10.1016/j.egyr.2021.02.039.

[ZV98] J. M. Zamarreño and P. Vega. “State Space Neural Network. Properties and Application”. In: *Neural Networks* 11.6 (1998), pp. 1099–1112. DOI: 10.1016/S0893-6080(98)00074-4.

[ZWP23] S. Zhang, O. Wallscheid, and M. Porrmann. “Machine Learning for the Control and Monitoring of Electric Machine Drives: Advances and Trends”. In: *IEEE Open Journal of Industry Applications* 4 (2023), pp. 188–214. DOI: 10.1109/OJIA.2023.3284717.

Own publications

[Bal+21] P. Balakrishna, G. Book, W. Kirchgässner, M. Schenke, A. Traue, and O. Wallscheid. "gym-electric-motor (GEM): A Python toolbox for the simulation of electric drive systems". In: *Journal of Open Source Software* 6.58 (2021), p. 2498. DOI: 10.21105/joss.02498.

[Boo+21] G. Book, A. Traue, P. Balakrishna, et al. "Transferring Online Reinforcement Learning for Electric Motor Control From Simulation to Real-World Experiments". In: *IEEE Open Journal of Power Electronics* 2 (2021), pp. 187–201. DOI: 10.1109/OJPEL.2021.3065877.

[Kir+22] W. Kirchgässner, D. Wöckinger, O. Wallscheid, G. Bramerdorfer, and J. Böcker. "Application of Thermal Neural Networks on a Small-Scale Electric Motor". In: *IKMT 2022; 13. GMM/ETG-Symposium*. 2022, pp. 1–6.

[KWB19a] W. Kirchgässner, O. Wallscheid, and J. Böcker. "Deep Residual Convolutional and Recurrent Neural Networks for Temperature Estimation in Permanent Magnet Synchronous Motors". In: *IEEE International Electric Machines Drives Conference*. 2019, pp. 1439–1446. DOI: 10.1109/iemdc.2019.8785109.

[KWB19b] W. Kirchgässner, O. Wallscheid, and J. Böcker. "Empirical Evaluation of Exponentially Weighted Moving Averages for Simple Linear Thermal Modeling of Permanent Magnet Synchronous Machines". In: *Proceedings of the 28th International Symposium on Industrial Electronics*. 2019, pp. 318–323. DOI: 10.1109/ISIE.2019.8781195.

[KWB20] W. Kirchgässner, O. Wallscheid, and J. Böcker. "Estimating Electric Motor Temperatures with Deep Residual Machine Learning". In: *IEEE Transactions on Power Electronics* 36.7 (2020), pp. 7480–7488. DOI: 10.1109/TPEL.2020.3045596.

[KWB21a] W. Kirchgässner, O. Wallscheid, and J. Böcker. "Data-Driven Permanent Magnet Temperature Estimation in Synchronous Motors with Supervised Machine Learning: A Benchmark". In: *IEEE Transactions on Energy Conversion* 36.3 (2021), pp. 2059–2067. DOI: 10.1109/TEC.2021.3052546.

[KWB21b] W. Kirchgässner, O. Wallscheid, and J. Böcker. *Electric Motor Temperature Dataset*. Paderborn, 2021. DOI: 10.34740/KAGGLE/DSV/2161054.

[KWB21c] W. Kirchgässner, O. Wallscheid, and J. Böcker. *Thermal Neural Networks: Lumped-Parameter Thermal Modeling With State-Space Machine Learning*. 2021. arXiv: 2103.16323 [cs.LG].

[KWB22] W. Kirchgässner, O. Wallscheid, and J. Böcker. "Learning Thermal Properties and Temperature Models of Electric Motors with Neural Ordinary Differential Equations". In: *2022 International Power Electronics Conference (IPEC-Himeji 2022- ECCE Asia)*. 2022, pp. 2746–2753. DOI: 10.23919/IPEC-Himeji2022-ECCE53331.2022.9807209.

[KWB23] W. Kirchgässner, O. Wallscheid, and J. Böcker. "Thermal neural networks: Lumped-parameter thermal modeling with state-space machine learning". In: *Engineering Applications of Artificial Intelligence* 117 (2023), p. 105537. DOI: 10.1016/j.engappai.2022.105537.

[SKW20] M. Schenke, W. Kirchgässner, and O. Wallscheid. "Controller Design for Electrical Drives by Deep Reinforcement Learning: A Proof of Concept". In: *IEEE Transactions on Industrial Informatics* 16.7 (2020), pp. 4650–4658. DOI: 10.1109/TII.2019.2948387.

[Tra+20b] A. Traue, G. Book, W. Kirchgässner, and O. Wallscheid. “Toward a Reinforcement Learning Environment Toolbox for Intelligent Electric Motor Control”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–10. DOI: 10.1109/TNNLS.2020.3029573.

[WKB17] O. Wallscheid, W. Kirchgässner, and J. Böcker. “Investigation of Long Short-Term Memory Networks to Temperature Prediction for Permanent Magnet Synchronous Motors”. In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 2017-May. 2017, pp. 1940–1947. DOI: 10.1109/IJCNN.2017.7966088.

List of Figures

List of Tables

A Appendix

A.1 Test bench setup

This section represents a translated excerpt from the appendix in [Wal17], since the same test bench was used in that work. The test bench, that was facilitated for data set $\mathcal{A}$, is located in the laboratories of the department of power electronics and electrical drives (LEA), Paderborn University, in Paderborn, Germany. It is encapsulated in a cabin for thermal and safety reasons, as shown in Fig. A.1.

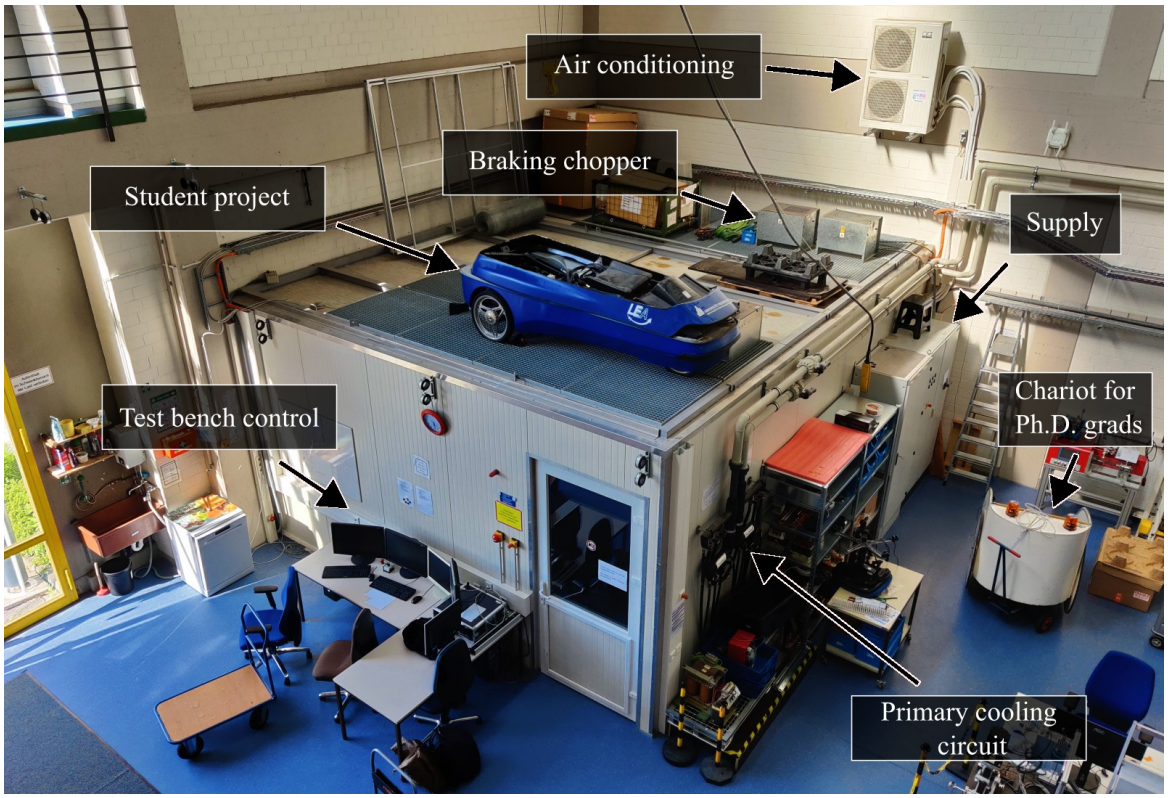


Figure A.1: Outside view of the test bench cabin

A schematic of the test bench is displayed in Fig. A.2. A noteworthy characteristic is the coupled DC voltage link between load motor and device under test (DUT) such that only power losses within this conversion chain have to be compensated from the grid. Motor control and measurement is handled with rapid prototyping systems from the company dSPACE through model-driven SW. Not shown in Fig. A.2, there is a steerable cooling circuit for the DUT with a separate coolant temperature control unit by the company Single (model STW150), as well as a coolant circuit for the converters.

Moreover, a high-precision power analyzer from the Yokogawa company denotes available equipment that can be used for power loss measurement of the upstream power electronics and the DUT [Yok]. Note that in this work a power loss LUT is necessary for the expert-based LPTN from Sec. 7.5 only. Furthermore, the used torque measurement flange is the T10F from the Hottinger Baldwin Messtechnik company [HBM]. Both

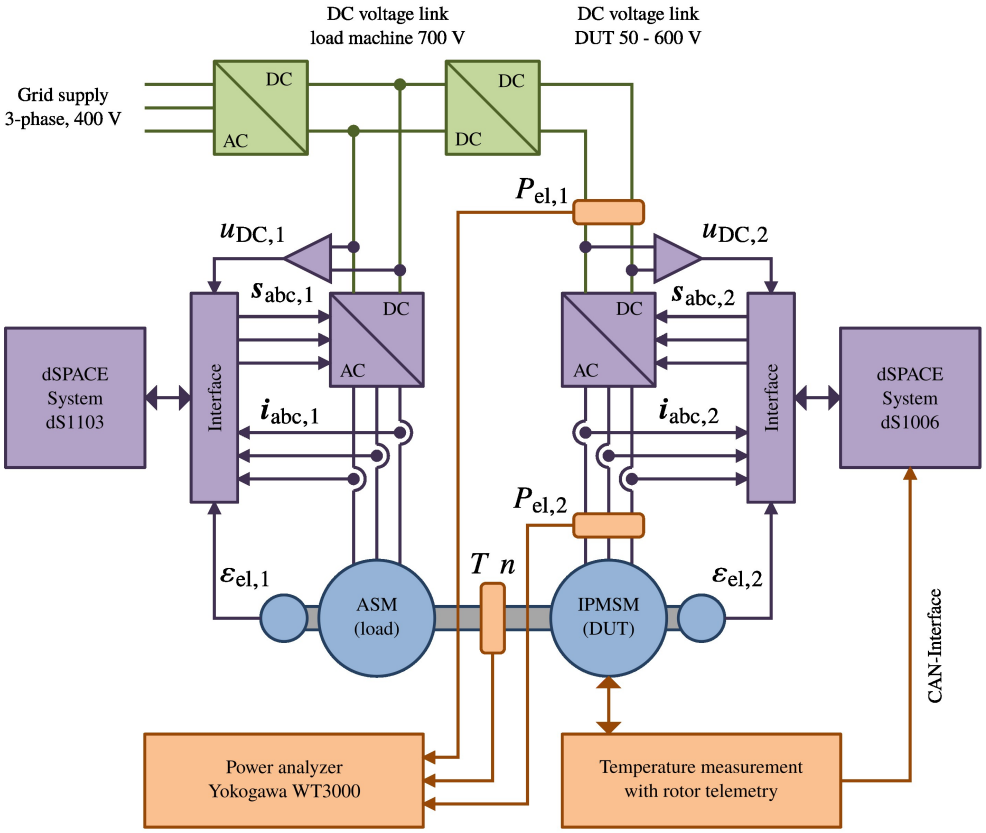


Figure A.2: Test bench setup (source: [Wal17; Pet14])

motors' characteristics are compiled in Tab. A.1, where it becomes evident that the load motor is of higher power in order to ensure control of a motor speed profile also under high load. The DUT, on the other hand, is usually torque-controlled. The inverter characteristics can be seen in Tab. A.2. It consists of half bridge modules (SKiiP 1242GB120-4DW) and DC link capacitors (SKCB 4m7-40-4-12) from the Semikron company. The employed inverter comes with internal sensors as well, which are compiled in Tab. A.3.

Cross-sections of the utilized IPMSM are shown in Fig. A.3 and Fig. A.4. These sketches are more detailed than the qualitative diagram in Fig. 5.3. The motor is constructed with a short shaft and large radial diameter because it was intended for a HEV drive train, which also has to accommodate the ICE, gearing, and torque converter. The permanent magnets consist of sintered, anisotropic NdFeB 210/250 h material from the MS-Schramberg company [MS-], whereas the iron in stator and rotor consists of magnetic steel sheets of type M330-35AP with 5 µm baking varnish one-sided [Ele16]. The stator winding sensor reading as well as the permanent magnet sensor reading in data set $\mathcal{A}$ are the average across several sensors, which can be seen in Fig. A.3. Four sensors are used for the PM with direct magnet contact and three for the winding with varying depth into the groove. The PM sensor readings are transmitted wirelessly along a telemetry unit. All sensors denote thermocouples of type K (class 1, max. measurement error ± 1.5 °C). Furthermore, all sensors are located in the axial center of the motor.

Following construction information is not traceable, unfortunately:

- coolant jacket schematic and dimensions,
- winding scheme (filling factor, isolation, wire dimensions, etc.),
- end winding dimensions,
- ball bearing construction and power loss characteristics,

	DUT	Load motor
Type	IPMSM	ASM
Nominal voltage	177 V	380 V
Nominal | max. current	110 A | 283 A	293 A | 450 A
Nominal | max. torque	110 N m | 250 N m	380 N m | 510 N m
Nominal | max. power	20 kW | 52 kW	160 kW | 210 kW
Nominal | max. speed	1700 1/min | 6000 1/min	4000 1/min | 8000 1/min
Pole pair number	8	1
Cooling type	Water-glycol	Active air

Table A.1: Comparison of load motor and DUT

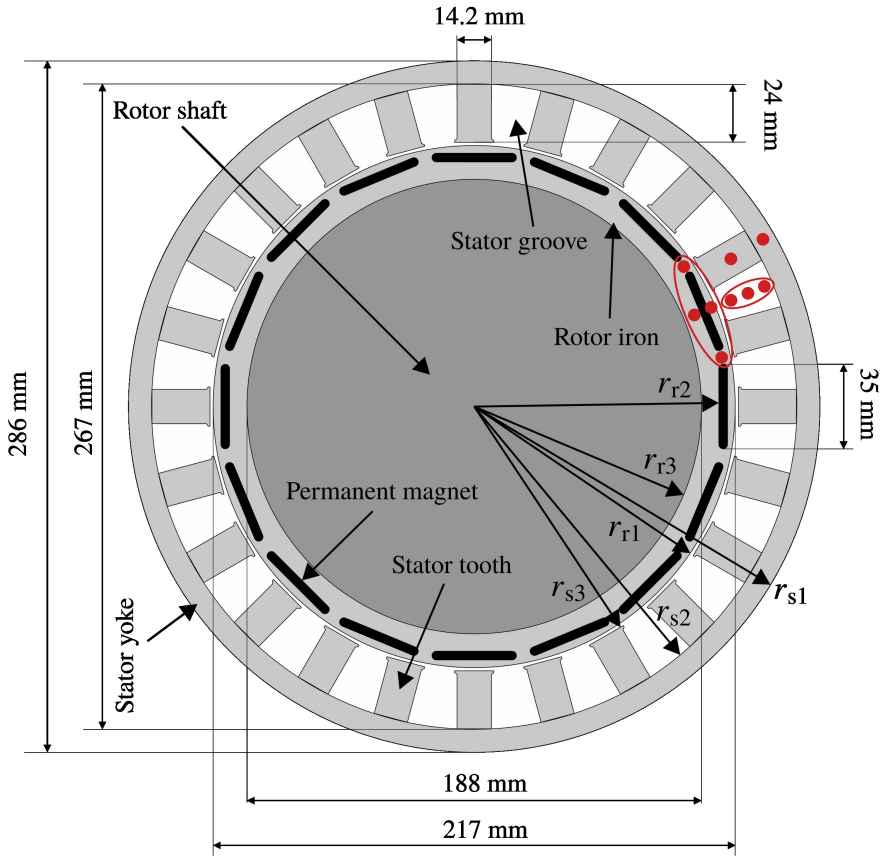


Figure A.3: Cross-section of the PMSM prototype (cooling jacket and stator winding not shown), red dots mark sensor locations (source: [Wal17])

Description	Value
Maximum DC link voltage	900 V
Maximum phase current (25 °C | 70 °C)	1200 A | 900 A
Maximum switching frequency	14 kHz
Junction temperature	−40 °C. . . 150 °C
Interlock time	3.3 µs
Pulse suppression	750 ns
Response time IGBT	1.4 µs
Response time diode	1.5 µs
Flux voltage IGBT (25 °C | 125 °C)	1.2 V | 1.3 V
Path resistance IGBT (25 °C | 125 °C)	1.3 mΩ | 1.8 mΩ
Flux voltage diode (25 °C | 125 °C)	1.3 V | 1.0 V
Path resistance diode (25 °C | 125 °C)	0.8 mΩ | 1.0 mΩ
DC link capacitance	29 mF

Table A.2: Inverter characteristics [Sem]

- mechanic spacer shaft (material, dimensions of the cavity, etc.),
- test bench casing dimensioning and material,
- power loss density, since a FEA cannot be parameterized reasonably.

The lack of these information impedes a rigorous design of a multi-physics, white-box model.

The DUT that provided data set $\mathcal{A}$ was disassembled off the test bench in 2019. It represented a prototype that is not manufactured for any series and is, thus, not acquirable. Hence, more measurement data cannot be collected anymore.

Data set operation points

All operation point trajectories can be seen in Fig. A.5. The profiles are not enumerated stringently since there were measurement sessions in between that were dropped due to measurement errors. The first half of profiles up to profile number 45 were recorded during the first half of the 2010s, whereas the random walk profiles as of profile number 46 were all collected within three weeks in 2018. Fault conditions were not recorded.

Sensor type	Parameter	Value
Phase current		
	Type	Current-compensating converter
	Cut-off frequency small signal	100 kHz
	Cut-off frequency large signal	1 kHz
	Latency	<1 µs
	Offset error	0.35 % of i_{max}
	Gain error	1.5 % of i_{meas}
	Temperature coefficient	0.001 %/K
Voltage		
	Type	High-impedance differential amplifier
	Measurement error	±2 %
	Filter time constant	500 µs
Temperature		
	Type	PTC
	Measurement error	±3 °C

Table A.3: Characteristics of the inverter's internal sensors [Sem]

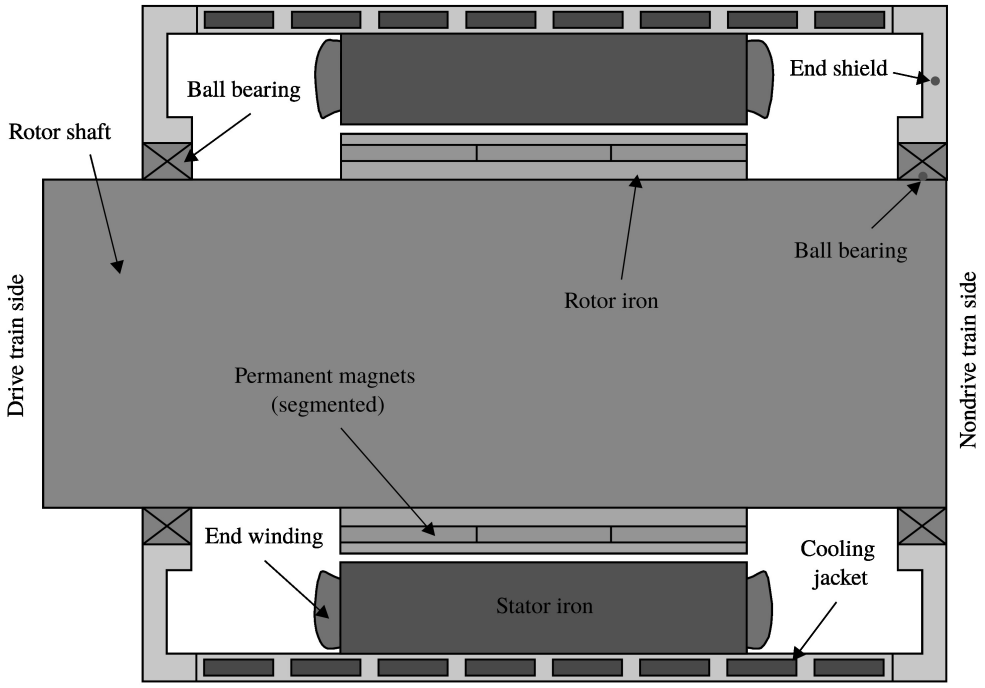


Figure A.4: Another cross-section of the PMSM prototype with qualitative dimensioning (source: [Wal17])

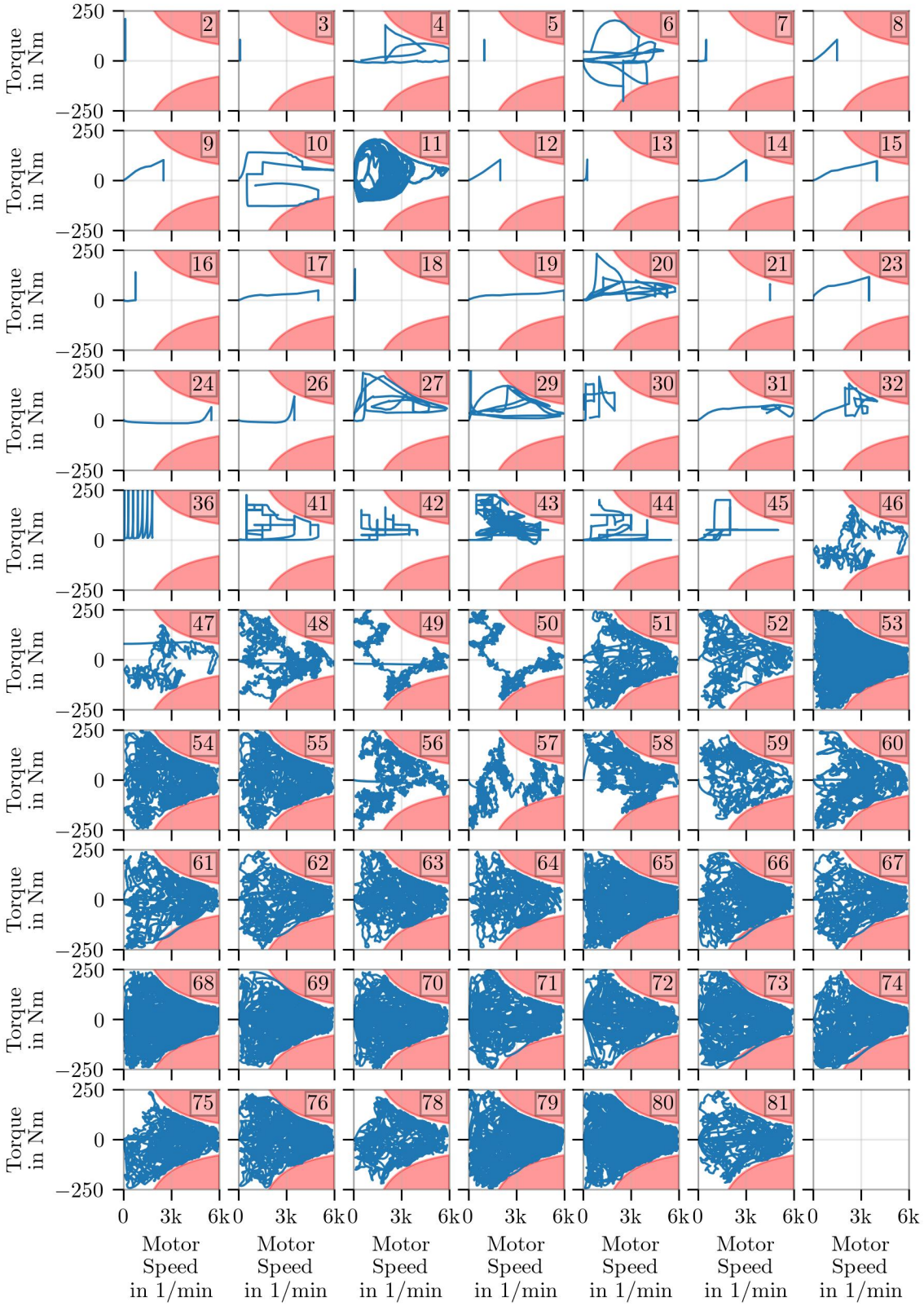


Figure A.5: Excitation trajectory per profile in the motor-speed-torque-plane for data set $\mathcal{A}$

A.2 Sampling methods for hyperparameter optimization

The HPO is often a critical step into achieving high function approximation performance within machine learning and especially deep learning. Like for classic optimization techniques, a cost function $\mathcal{L} : \mathcal{U} \to \mathcal{Y}$ is to be minimized. This function often has the same codomain $\mathcal{Y}$ as the cost function used for model training since the goal of HPO is to improve a model. However, the domain $\mathcal{U}$ is different as not the model parameters $\boldsymbol{\theta}$ are subject to variation but hyperparameters $\boldsymbol{\nu}$ such as the number of layers in an ANN or the coefficient regularization strength in a linear model. The difference between HPO strategies lies in the sampling method: how are new samples/hyperparameters chosen given the previously obtained mappings? HPO sampling methods are characterized by utilizing a small amount of function executions for the decision where in $\mathcal{U}$ to sample next since one function execution is usually expensive (e.g., it could represent training of 10 different ANNs for 200 epochs on a large data set with different seeds and a following evaluation on a test set). That is why gradient descent or higher-order optimization methods are rarely used for HPO.

Note that manual search, grid search, and random search denote very efficient and simple sampling methods [BB12]. However, they have in common that previous evaluations are not taken into account for further sampling. The following methods seek to improve on these baseline approaches by utilizing information from previous samples.

Bayesian sampling

In Bayesian optimization (BO), a surrogate predictive model $f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D})$ is trained on previous evaluations $\mathcal{D}$, which is computationally cheap to execute and approximates the mapping from hyperparameters $\boldsymbol{\nu}$ to costs $\mathcal{L}$. Assuming that the surrogate model is sufficiently accurate, promising new candidate hyperparameters are chosen by optimizating a so-called acquisition function, for which the expected improvement is commonly facilitated [JSW98]:

$$\mathrm{EI}_{\mathcal{L}^*}[\boldsymbol{\nu}|\mathcal{D}] = \int_{-\infty}^{\mathcal{L}^*} (\mathcal{L}^* - \mathcal{L}) f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D}) \mathrm{d}\mathcal{L}. \tag{A.1}$$

Candidate hyperparameter sets would be usually sampled from a grid and evaluated through the acquisition function. The acquisition function is a tool for trading off uncertainty and exploitation of good, previously found hyperparameters. Typical surrogate models are Gaussian processes or random forests [RW05].

Tree-structured parzen estimator sampling

The tree-structured Parzen estimator (TPE) is a BO method with the expected improvement as acquisition function [Ber+11]. Here, two surrogate models are distinguished with

$$f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D}) \coloneqq \begin{cases} f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D}^{(\mathrm{l})}) & (\mathcal{L} \leq \mathcal{L}_\alpha), \\ f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D}^{(\mathrm{g})}) & (\mathcal{L} > \mathcal{L}_\alpha), \end{cases} \tag{A.2}$$

where $\mathcal{D}^{(\mathrm{l})}, \mathcal{D}^{(\mathrm{g})}$ are the observation sets corresponding to $\mathcal{L}(\boldsymbol{\nu}_i) \leq \mathcal{L}_\alpha$ and $\mathcal{L}(\boldsymbol{\nu}_i) > \mathcal{L}_\alpha$, respectively. The threshold α is manually chosen to let $\mathcal{D}^{(\mathrm{l})}$ contain a nonnegligible but smaller portion of all observations. Moreover, the surrogate models are built with kernel density estimation, e.g., a mixture of Gaussians. Then, the expected improvement can be rearranged to

$$\mathrm{EI}_{\mathcal{L}^\alpha}[\boldsymbol{\nu}|\mathcal{D}] \overset{\text{rank}}{\simeq} \frac{f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D}^{(\mathrm{l})})}{f_{\text{surrogate}}(\mathcal{L}|\boldsymbol{\nu}, \mathcal{D}^{(\mathrm{g})})}. \tag{A.3}$$

In each sampling iteration, random samples are evaluated by TPE according to the maximization of the ratio (A.3). This procedure prefers samples that tend to stem from the better distribution. Due to the tree-structure, hierarchical hyperparameters can be modeled elegantly as well. For more information on TPE, refer to [Wat23].

Since kernels are fitted on an increasing set of samples, TPE can become a computational bottleneck the cheaper a function evaluation and the longer the search. Faster sampling methods are often denoted by heuristics.

Particle swarm optimization

Unlike Bayesian sampling techniques that utilize Bayes' rule, the particle swarm optimization (PSO) counts as heuristic that relies on a natural analogy [KE95; PKB07]. The sampling rule behind a PSO lends from swarm behavior, which is thought of enhancing the search for food by mutual information exchange between individuals and the swarm. Given an interval for each of the m hyperparameters $\nu_i \in [\nu_{i,\min}, \nu_{i,\max}]$ and an integer number many individuals or particles n, each particle will be distributed randomly over the search space initially. Each particle corresponds to a hyperparameter set that has to be evaluated. The current swarm position is described by $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ where each column corresponds to a hyperparameter and each row to a particle $\boldsymbol{\nu}_j$ with $j = \{1 \dots n\}$. The update rule for all particles' position in $\mathcal{U}$ has many variants in the literature, and the algorithm used in this work is presented in the following. The same way the position of each particle is bounded in each dimension beforehand, the maximum displacement into the next iteration is bounded per hyperparameter dimension. Here, the maximum displacement is 10 % of the corresponding hyperparameter domain, that is, $|d_{i,\max}| = 0.1 \cdot (\nu_{i,\max} - \nu_{i,\min})$. Having the boundaries set up, in each iteration a displacement matrix $\boldsymbol{D} \in \mathbb{R}^{n\times m}$ is calculated, which will be added to the swarm position

$$\boldsymbol{X} \leftarrow \boldsymbol{X} + \boldsymbol{D}, \tag{A.4}$$

with an eventual clipping to the predetermined bounds. The update rule for $\boldsymbol{D}$ reads

$$\boldsymbol{D} \leftarrow J\boldsymbol{D} + \lambda_1 r_1 (\boldsymbol{X}_{\text{ind}} - \boldsymbol{X}) + \lambda_2 r_2 (\mathbb{1}_n \boldsymbol{\nu}_{\text{global}}^{\mathsf{T}} - \boldsymbol{X}), \tag{A.5}$$

with $J = 0.9$ for the moment of inertia, $\lambda_1 = 2$ and $\lambda_2 = 2$ for the weighting of the position matrix that corresponds to each individual's best seen position $\boldsymbol{X}_{\text{ind}}$ and of the global best position $\boldsymbol{\nu}_{\text{global}}$ seen by any individual. In order to foster exploration, random terms r_1, r_2 are sampled uniformly from $[0, 1]$ and further mitigate displacement to the global best and the individual best position.

Other update rules for the displacement matrix incorporate reflections on the bounds or constrained information propagation in contrast to the broadcast as applied here. A disadvantage of the PSO is the difficult application of hierarchical domains as well as integer domains. Moreover, in case of long-running function evaluations with a considerable variance in run time, particles that finish early wait a correspondingly long time for the last particle in the iteration to finish before the next iteration can start. This can be mitigated with constrained information propagation though.

A.3 Reproducibility

Code, figures, and documents to this dissertation are published on GitHub[1] in a standalone repository, scan Fig. A.6. There, all figures in this work should be reproducible through Python files and Jupyter notebooks. Code examples for a TNN, independent of this thesis, can be found in another repository[2] also on GitHub. The data set is hosted on kaggle[3] [KWB21b].

Figure A.6: QR code to the GitHub repository containing all supplementary information to this dissertation

[1] https://github.com/wkirgsn/diss-wk
[2] https://github.com/wkirgsn/thermal-nn
[3] https://www.kaggle.com/dsv/2161054