Technische Universität München
TUM School of Engineering and Design

KI-Batteriespeichermodell auf Basis neuronaler Netze - Entwicklung einer neuen Datenvorverarbeitungspipeline und Trainingsmethodologie

Daniel Jerouschek

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Malte Jaensch, PhD

Prüfer*innen der Dissertation:

1. Prof. Dr.-Ing. Dr. h.c. Ralph Kennel
2. Prof. Dr.-Ing. Andreas Jossen

Die Dissertation wurde am 03.03.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 10.11.2022 angenommen.

Kurzfassung

Ein Bestandteil der Energiewende im Verkehrssektor ist die Elektrifizierung des Antriebsstrangs. Fahrzeuge mit einem 48V-Mild-Hybrid Bordnetz bieten eine höhere Energieeffizienz als vergleichbare Fahrzeuge mit reinem verbrennungsmotorischem Antrieb. Aufgrund der im Verhältnis zur gespeicherten Energie hohen Leistungen, besteht die Herausforderung dieser Form des Bordnetzes aus einer exakten Spannungsprädiktion. Für die Bordnetzstabilität ist die Spannungsprädiktion entscheidend und damit ein sicherheitsrelevanter Baustein in den Fahrzeugen. Die aktuell in der Industrie genutzten Modelle basieren auf elektrischen Ersatzschaltbildmodellen, die neben messbaren physikalischen Größen zusätzlich abhängig vom Ladezustand der Batterie sind. In Bezug auf diese beiden Punkte werden in der vorliegenden Arbeit alternative Modellierungsmethoden entwickelt. Zum einen basiert der Modellierungsansatz auf künstlichen neuronalen Netzen, die mit Backpropagation Algorithmen trainiert werden. Zum anderen nutzen die Netze ausschließlich physikalisch messbare Parameter als Input. Die Datenvorverarbeitung wird neben der Sequenzialisierung und Normalisierung um neuartige Algorithmen zum Under- und Oversampling ergänzt. Das Undersampling wird auf kontinuierliche Daten adaptiert, mit der Möglichkeit mehrere Features parallel dem Reduktionsprozess zu unterstellen. Die dabei erzeugte Flexibilität im Multi Feature Undersampling ist für unausgewogene Datensätze ein wichtiges Kriterium, um schnell und präzise Datenverteilungen zu erlangen, die für das Training neuronaler Netze besser geeignet sind. Das anschließende Oversampling eliminiert die unterrepräsentierten Featurebereiche in einem mehrstufigen Prozess, der auf kontinuierliche In- und Outputs angepasst ist. Die Zeitabhängigkeiten im Spannungsverhalten der Batterie werden durch rekurrente neuronale Netze aufgegriffen. Insbesondere zeigen Long Short-Term Memory Zellen die Fähigkeit selbst langsame elektrochemische Effekte zu modellieren. Ein Teil der Hyperparameter wurde anhand von Vergleichsmodellen experimentell bestimmt, für den anderen Teil wurden sinnvolle Parameterräume definiert. Dabei ist das Hyperparametertuning in eine kaskadierte Pipeline gebettet, die unabhängige Parametergruppen mit einem Bayesschen Optimierungsframework sequenziell optimiert. Die Datenvorverarbeitung und das Hyperparametertuning wurden im Rahmen dieser Arbeit anhand zweier Lithium-Ionen-Batteriepacks erfolgreich verifiziert und über einen sehr großen Temperatur- und Leistungsbereich validiert. Der gesamte Prozess zur Entwicklung eines Batteriemodells auf Basis von neuronalen Netzen ist komplex und interdisziplinär. Die in dieser Arbeit entwickelte Datenverarbeitungs- und Hyperparametertuningpipeline stellen ein Tool dar, welches die Entwicklung solcher Modelle vereinfacht.

Abstract

One basic part of the energy transition in the transport sector is the electrification of the powertrain. Vehicles with a 48V mild hybrid electrical power supply system offer higher energy efficiency than comparable vehicles with a purely internal combustion engine drive. Due to the high power in relation to the stored energy, the challenge of this form of vehicle electrical system consists of precise voltage prediction.Voltage prediction is crucial for ensuring the stability of the vehicle electrical system and is therefore a safety-relevant component in the vehicles. The models currently used in the industry are based on electrical equivalent circuit models which, in addition to measurable physical quantities, are also dependent on the state of charge of the battery. With respect to these two issues, alternative modeling methods are developed in the present work. On the one hand, the modeling approach is based on artificial neural networks trained with backpropagation algorithms. On the other hand, the networks use only physically measurable parameters as input. In addition to sequentialization and normalization data preprocessing is complemented by novel algorithms for undersampling and oversampling. Undersampling is adapted to continuous data, with the possibility of subjecting multiple features to the reduction process in parallel. The flexibility generated in the multi feature undersampling is an important criterion for unbalanced datasets to quickly and accurately obtain data distributions that are more suitable for neural network training. The subsequent oversampling eliminates the underrepresented feature regions in a multistep process adapted to continuous inputs and outputs. Time dependencies in battery voltage behavior are addressed by recurrent neural networks. In particular, Long Short-Term Memory cells show the ability to model even slow electrochemical effects. One Part of the hyperparameters were determined experimentally using comparative models, and reasonable parameter spaces were defined for the other part. Here, the hyperparameter tuning is embedded in a cascaded pipeline that sequentially optimizes independent parameter groups using a bayesian optimization framework. The data preprocessing and hyperparameter tuning were successfully verified and validated over a very wide temperature and power range using two lithium-ion battery packs in this work. The entire process of developing a neural network-based battery model is complex and interdisciplinary. The data processing and hyperparameter tuning pipeline developed in this work provide a tool that simplifies the development of such models.

Vorwort und Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als Doktorand bei der IAV GmbH in München in Kooperation mit dem Lehrstuhl für Elektrische Antriebssysteme und Leistungselektronik (EAL) an der Technischen Universität München (TUM).

Mein besonderer Dank gilt an erster Stelle Prof. Dr.-Ing. Ralph Kennel für die Möglichkeit zur Promotion und für die Betreuung dieser Arbeit. Vielen Dank für alle inneren und äußeren Freiheiten, Ihre Unterstützung und das mir entgegengebrachte Vertrauen.

Bei Prof. Dr.-Ing. Andreas Jossen möchte ich mich ausdrücklich für die schnelle Erstellung des umfangreichen Zweitgutachtens sowie für seine Unterstützung und wertvollen Hinweise bedanken.

Des Weiteren danke ich Prof. Malte Jaensch für die Übernahme des Prüfungsvorsitzes.

Ein großes Dankeschön geht an meinen Mentor Dr. rer. nat. Ahmet Taskiran für die fachlichen Diskussionen, wertvolle Anregungen und das mir entgegengebrachte Vertrauen.

Dieser Dank gilt ebenso den Kolleginnen und Kollegen der IAV GmbH, die Zusammenarbeit hat mir große Freude bereitet.

Mein herzlicher Dank geht an meine Familie und Freunde. Bei meinen Eltern bedanke ich mich vielmals für ihre fortwährende Unterstützung, ohne die mir meine Ausbildung so nicht möglich gewesen wäre.

Für den bedingungslosen Rückhalt und ihr Verständnis möchte ich mich ganz besonders bei meiner Frau Viola bedanken. Ohne ihren Zuspruch wäre mir die Fertigstellung dieser Arbeit nicht möglich gewesen. Ein ganz besonderer Dank gilt meiner Tochter Marie, die mich in den letzten Zügen der Ausarbeitung meiner Dissertation maßgeblich motiviert hat. Ich freue mich nun auf viele gemeinsame Abenteuer mit euch.

München, Daniel Jerouschek

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Akronyme

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

1. Einleitung

1.1. Motivation

Künstliche Intelligenz (KI) ist eine branchenübergreifende Schlüsseltechnologie mit vielfältigen Einsatzmöglichkeiten in der Industrie und Forschung. Nach dem Mooreschen Gesetz verdoppelt sich alle 12 bis 24 Monate die Komplexität integrierter Schaltkreise und damit die Rechenleistung von Computern. Mit der Entwicklung der Rechenleistung wurden in den letzten Jahren Möglichkeiten geschaffen, große Datenmengen effizient zu verarbeiten und für eine schwache KI zu nutzen. Seit den 1950er Jahren wird an KI-Systemen geforscht. Dabei ergab sich eine deutliche Dynamisierung in den letzten Jahren. Neben bestehenden Algorithmen werden somit laufend neue Ansätze vorgestellt und erprobt.

Im Allgemeinen werden Algorithmen, die spezielle Fragen auf Basis von vorher selbst erlernten Zusammenhängen beantworten können, als „schwache KI“ bezeichnet. Die sogenannte „starke KI“ kann Probleme genereller Art lösen, ist aber bis jetzt noch nicht realisiert worden. Abbildung 1 zeigt die Zusammenhänge der im Sprachgebrauch oft diffus verwendeten Begriffe KI, Data Mining, maschinelles Lernen und Deep Learning. Data Mining ist eine wichtige Grundlage für KI-Produkte. Deep Learning dagegen ist ein Teilgebiet des maschinellen Lernens. Je nachdem ob gelabelte Daten vorhanden sind oder nicht, spricht man bei den algorithmischen Ansätzen des maschinellen Lernens von überwachtem oder unüberwachtem Lernen.

Der große Vorteil von maschinellem Lernen ist, dass die zum Teil sehr großen Datenmengen verarbeitet werden und dann für das Lernen der Zusammenhänge im Datensatz genutzt werden. Das Lernen erfolgt nicht regelbasiert aber automatisiert, wodurch ein Einsatz von Fachleuten aus dem zu lernenden Gebiet nicht notwendig ist. Der Algorithmus bleibt auch während dem Einsatz lernfähig und kann richtige Entscheidungen in Sekundenschnelle treffen. Zudem können aufgrund der hochkomplexen Strukturen Zusammenhänge erkannt werden, die höchst nichtlinear und von Menschen schwer bzw. nicht zu erkennen sind.

In einigen Gebieten werden KI-Algorithmen bereits erfolgreich eingesetzt. Googles KI „Alpha Zero“ schlug 2017 das bis dahin beste Schachprogramm „Stockfish“ in 100 Partien Schach 28-mal, ohne dabei ein einziges Mal zu verlieren [1]. Im Feld der Computerlinguistik (engl. Natural Language Processing (NLP) nutzen Online-Übersetzungsdienste riesige Datenmengen um neuronale Netze (NN) zu trainieren. Die dabei erzielten Ergebnisse sind nicht perfekt, weisen aber eine derartige Güte auf, sodass sie im alltäglichen Gebrauch oft genutzt werden. Auch in lebenswichtigen Anwendungen, wie der Krebs-

diagnose, zeigen KI-Systeme die gleiche Güte in der Analyse wie Fachärzte. Jedoch benötigen die Maschinen deutlich weniger Zeit dafür [2]. Die Energiewende ist ein weiteres gegenwärtiges Problem, welches mithilfe von KI effizienter gelöst werden kann.

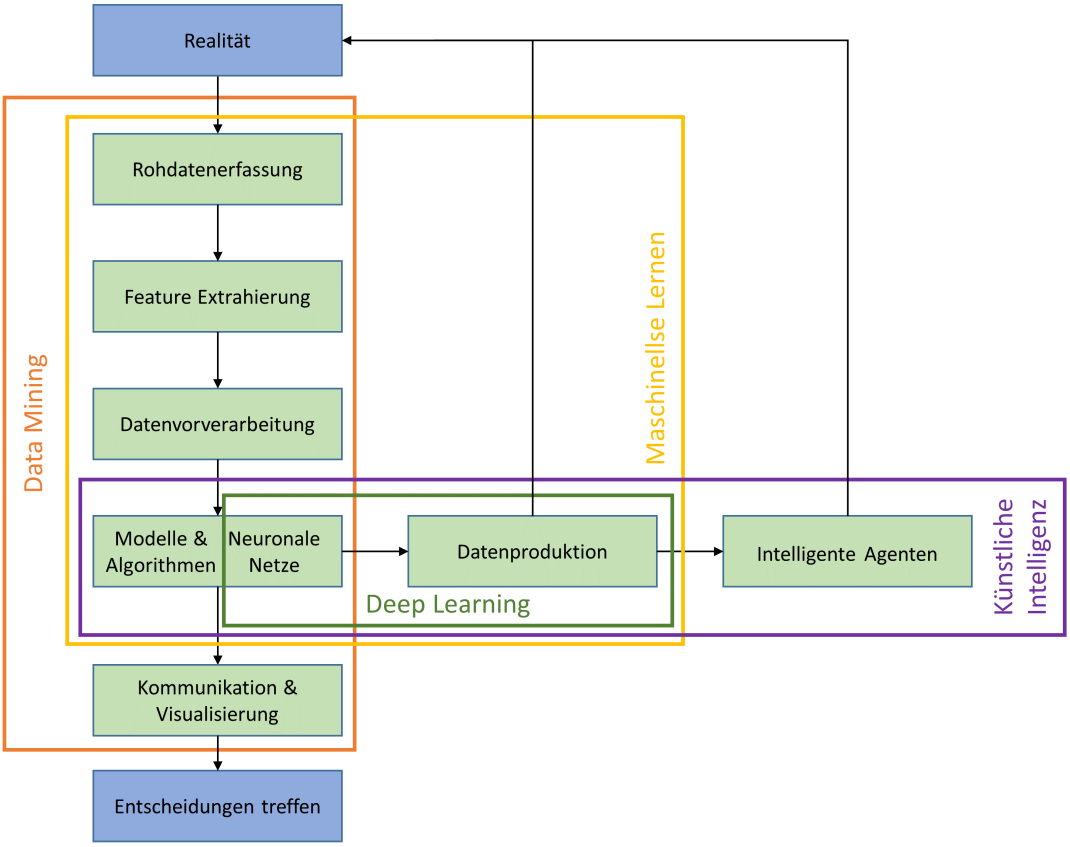


Abbildung 1 Zusammenhang der Begrifflichkeiten künstliche Intelligenz, Data Mining, maschinelles Lernen und Deep Learning

Einen wichtigen Beitrag zur Energiewende leistet die Elektrifizierung des Verkehrssektors. Um diesen Schritt zu erfüllen, ist die Entwicklung und Produktion von vielen verschiedenen und großen elektrischen Energiespeichern notwendig. Neben batteriebetriebenen Elektrofahrzeugen, mit ihren Nachteilen in Bezug auf die Infrastruktur, Kosten und Reichweite, besteht eine vielversprechende Lösung zur Reduzierung der CO_2-Emissionen in der Hybridisierung von Fahrzeugen mit Verbrennungsmotor. Ein Mild-Hybrid-Fahrzeug (MHEV) mit einem 48-V-Bordnetz ist eine kostengünstige und effektive Elektrifizierungsvariante [3]. Eine elektrische Maschine sorgt für die Energierückgewinnung beim Bremsen und unterstützt den Verbrennungsmotor beim Beschleunigen sowie bei der Lastpunktanpassung. Außerdem können riemengetriebene Verbraucher elektrifiziert und mit der gewonnenen Energie betrieben werden. Die rekuperierte Energie muss in einem elektrischen Energiespeicher, meist in Form einer Lithium-Ionen-Batterie (LIB), gespeichert werden.

Die Anforderung bezüglich des Reifegrades, der Energie- und Leistungsdichte, der Sicherheit sowie der Kosten erfüllen derzeit LIB, im Vergleich zu den anderen Batterietechnologien, am ausgewogensten. Trotz der Vorteile einer LIB ist das Entwicklungspotential in dieser Technologie noch immer groß. Alternative Anoden- und Kathodenmaterialien, optimiertes Packaging und Kühlung sowie ein effizientes Batterie-Management-System (BMS) sind vielversprechende Entwicklungsbereiche in der LIB-Forschung. Insbesondere durch ein intelligentes und leistungsfähiges BMS, mit den Zustandsschätzern zum Lade- und Leistungszustand, können die Potentiale der Zellchemie einer Batterie vollumfänglich genutzt werden.

In Bezug auf die Batteriekapazität führt bei MHEV-Bordnetzen die hohe angewandte Leistung dazu, dass die Vorhersage von Leistungsfähigkeit und Spannung kritischer ist als die Schätzung des Ladezustands (engl. State-of-Charge (SOC)). Durch eine bessere Spannungsvorhersage kann das System häufiger unbegrenzt betrieben werden. Im Energiemanagement eines Bordnetzes werden die kumulierten Anforderungen der Verbraucher hinsichtlich der Leistung und des Stroms in Bezug auf die Bordnetzstabilität geprüft. Dazu wird die geforderte Leistung an das BMS weitergeben, welches daraufhin eine zu erwartende Spannung schätzt. Liegt diese Spannung innerhalb der festgelegten Grenzen, können die Verbraucher uneingeschränkt arbeiten. Werden Spannungsgrenzen überschritten, müssen Maßnahmen zur Leistungsreduzierung bzw. Leistungserhöhung eingeleitet werden. Durch eine genauere Spannungsprädiktion kann eine größere Nutzbarkeit des Speicherinhalts sichergestellt werden.

Die Kombination der beiden Schlüsseltechnologien KI und Batterietechnik birgt großes Potential, mithilfe dessen die Energiewende schneller und effizienter gestaltet werden kann.

1.2. Ziel der Arbeit

Der im Kapitel 2.1.6 beschriebene Stand der Technik zur Batteriemodellierung zeigt unterschiedliche Modellierungstechniken für State-of-Power Predictions. Modelle auf Basis neuronaler Netze sind in der Literatur selten zu finden, insbesondere ist keine Anleitung zum Aufbau dieser Modelle und der damit eng verbundenen Datenvorverarbeitung vorhanden. Die Fortschritte in der Rechengeschwindigkeit der Chips sowie die stetige Weiterentwicklung neuer Modelle und Algorithmen im Bereich des maschinellen Lernens versprechen eine erfolgreiche Anwendung von künstlicher Intelligenz in der Batteriemodellierung.

Der Lernfortschritt in einem NN kommt ausschließlich durch die eingegebenen Rohdaten zustande. Derartige Black-Box Modelle erfordern in der reinsten Form keinerlei technisch-wissenschaftliches Wissen über das zu modellierende Sample. Das Knowhow in der Batteriemodellierung kann genutzt werden, um die richtigen Input-Features zu wählen, die Datenverarbeitung zu optimieren und zu plausibilisieren. Diese Vorteile durch Wissen sind allerdings für Batterien allgemeingültig und müssen nicht auf die einzelne zu modellierende Zelle mit ihren Eigenschaften, ihrer Leistung, der Zellchemie sowie der Größe etc. angepasst werden. Somit kann ein Modell mit dem in dieser Arbeit entwickelten Schema erstellt werden, ohne auf Expertenwissen zurückgreifen zu müssen. Durch die vermehrten Einsatzgebiete und Varianten der Lithium-Ionen-Batterien werden immer häufiger Batteriemodelle benötigt. Eine manuelle Parametrierung der State-of-the-Art Modelle ist sehr aufwändig und oft nicht vereinbar mit den immer kürzer werdenden Entwicklungszeiten. Daher ist eine standardisierte und teilweise automatisierte Modellierungstechnik ein Beitrag zur Entwicklungsgeschwindigkeit, ohne dabei Einschränkungen bei der Modellgüte hinnehmen zu müssen. Dabei rückt das Wissen über NN in den Vordergrund, unabhängig von der Anwendung und dem Batteriesample.

Auf Basis dessen können für diese Dissertation folgende vier Ziele formuliert werden:

1. Die Nutzbarkeit von KI für die Batteriemodellierung ermöglichen: Eine Kernaufgabe besteht darin, die kürzlich in der Forschung entstandenen Vorteile der KI auf den Anwendungsfall Batterie zu transferieren. Wichtige Bestandteile sind dabei die schnelle automatisierte Modellierung, die Güte der Modelle und die Möglichkeit der Adaption an aktuelle Bedingungen. Die theoretischen Vorteile sollen anhand der Applikation verifiziert und verdeutlicht werden.

2. Eine geregelte Vorgehensweise zur Datenverarbeitung und Modellierung von NN im Batteriekontext etablieren: Die Modelle lernen aus den Daten, welche in das Netz eingespeist werden. Dementsprechend ist die Datenvorverarbeitung ein zentraler Bestandteil der Modellierung. Das Einlesen der Rohdaten und das Entfernen der Datenanomalien soll automatisiert erfolgen. Ein- und Ausgangsfeatures sind

zu bestimmen und eine schnelle effiziente Datenstruktur muss aufgebaut werden. Die Datenaufbereitung mit der damit verbundenen Reduzierung, Normalisierung und Verteilungsoptimierung soll mit einem für Batteriemodelle optimiertem Algorithmus durchgeführt werden. Der Algorithmus soll sowohl die in der Literatur gängigen Methoden nutzen als auch bei Notwendigkeit neu entwickelte Ansätze mit einbeziehen. Unterschiedliche NN sollen verglichen werden und daraus konkrete Modellierungsansätze vorgeschlagen werden. Trainingsparameter sollen insofern möglichst pauschal erarbeitet werden, andernfalls ein Hyperparametertuning Algorithmus entwickelt werden. Dafür müssen alle Hyperparameter erläutert und die Parameteroptionen eingegrenzt werden. Der Tuning Algorithmus soll schnell und präzise arbeiten.

3. Die Verifikation der Datenverarbeitung und Modellierung in einer Applikation: Die Verifikation des vorherigen Ziels soll anhand der Modellierung von zwei speziellen Anwendungsfällen nachgewiesen werden. Ziel ist es, je ein Batteriespannungsmodell zu erstellen, welches in der State-of-Power Prädiktion im Mild-Hybrid-Fahrzeugbordnetz genutzt werden kann, ohne dabei auf eine vorhergehende SOC Schätzung zurückgreifen zu müssen. Dabei wird die Funktion der Pipeline sowie die Implementierung der dafür verwendeten Algorithmen im Rahmen dieser Machbarkeitsstudie nachgewiesen.

4. Die Validierung des für die Verifikation erstellten Modells: Die in der Verifikation erstellten Modelle müssen in einem Rahmen validiert sein, der die tatsächlichen möglichen Arbeitspunkte der Batterie abdeckt. Die vom Trainingsdatensatz separierten Validierungsmessungen werden mit Testsamples an Prüfstanden erzeugt. Die aus der Validierung resultierende Modellgüte wird mit anderen Modellierungstechniken verglichen.

1.3. Einordnung der Arbeit

Die Arbeit darf nicht als allgemeingültiges Batteriemodell für bestimmte Zellchemien, Batteriegrößen oder Anwendungsfälle verstanden werden, sondern als ein Tool zur Parametrisierung eines Batteriemodells auf Basis von NN inklusive der damit verbundenen Datenvorverarbeitung.

In Abgrenzung zu vielen Batteriemodellen wird in dieser Arbeit weder der SOC modelliert noch auf den SOC als Input zurückgegriffen. SOC-Modellierungen sind in der Literatur vielfach in einigen verschiedenen Ausführungsformen diskutiert worden, wie auch in Kapitel 2.1.6 beschrieben wird. Die übliche Modellierungsmethode mit SOC als Input geht mit einem Verlust an Flexibilität im Entwicklungsprozess und einer begrenzten maximalen Modellgüte einher, weshalb in dieser Arbeit untersucht wird, inwiefern dieses vermieden werden kann.

1.4. Methodik

Die Entwicklung eines neuen Modellierungsansatzes benötigt eine systematische Vorgehensweise. Die im Folgenden dargestellten Schritte werden sequentiell abgearbeitet, wobei die einzelnen Schritte im Entwicklungsprozess iterativ wiederholt werden können:

- **Kapitel 2:** Um das Ziel „Die Nutzbarkeit von KI für Batteriemodellierung ermöglichen“ zu erreichen, wird in Kapitel 2.1 und 2.2 der aktuelle Stand der Technik in Bezug auf die Anwendung im Batterie- bzw. Fahrzeugbordnetzbereich anhand von aktueller Literatur diskutiert. Die Erkenntnisse aus dem physikalischen Verhalten werden genutzt, um Technologien im Feld der KI so auszuwählen, dass sie sinnvoll anwendbar sind. Die ausgewählten Methoden werden in Kapitel 2.3 beschrieben.
- **Kapitel 3:** Grundlage für das Training von NN sind Daten, die zunächst erhoben werden müssen. Die Trainingsdaten werden ergänzt um Validierungsdaten, welche separat an einem HIL Prüfstand erfasst werden.
- **Kapitel 4:** In diesem Kapitel werden Algorithmen vorgestellt und entwickelt, die zum Erreichen des Ziels „Eine geregelte Vorgehensweise zur Datenverarbeitung und Modellierung von NN im Batteriekontext etablieren“ dienen. In der Literatur übliche Verfahren zum Splitten der Datensätze, der Sequenzialisierung und der Normalisierung wurden ergänzt durch neu entwickelte Algorithmen. Insbesondere das Under- und Oversampling wurde speziell für diesen Anwendungsbereich um Methoden erweitert, welche die besonderen Gegebenheiten der Batteriemodellierung, wie die Anzahl und den Einfluss der Features auf die Modellgüte, variabel austesten und somit das Optimum finden kann. Zusammen mit der Auswahl der Features entsteht eine Datenverarbeitungspipeline, die auf die Eingangsdaten geregelt angewendet werden kann.
- **Kapitel 5:** Hier wird beschrieben, wie und in welcher Reihenfolge die Hyperparameter des NN getuned werden. Das Tuning wird an zwei konkreten Batterie Samples durchgeführt. Zusammen mit der Anwendung der Datenverarbeitungspipeline wird das Ziel der erfolgreichen Verifikation angestrebt.
- **6:** Anschließend werden die Modelle mit den am Prüfstand erhobenen Daten verifiziert und validiert, sowie in Relation zur Modellgüte von herkömmlichen Batteriemodellen gestellt.

2. Stand der Technik

Dieses Kapitel beschreibt die relevanten Grundlagen zu Lithium-Ionen-Batterien und deren Anwendung im 48 V Fahrzeugbordnetz. Dabei wird vorranging auf die physikalischen Eigenschaften, welche das Spannungs- und Leistungsverhalten beeinflussen eingegangen. Die Theorie hinter Deep Learning wird in Hinblick auf die in Kapitel 5 erstellten neuronalen Netze erläutert.

2.1. Lithium-Ionen-Batterien

Die Komplexität der Lithium-Ionen-Batterie wird stark von ihren physikalischen und chemischen Eigenschaften beeinflusst und ist daher grundlegend für die Modellierung. Das BMS überwacht die Zustände der Batterie sowohl mit Hilfe von Sensoren als auch mit Zellmodellen.

2.1.1. Funktionsweise

Lithium-Ionen-Batterien sind galvanische Zellen, die aufgrund der zwei verschiedenen Elektroden und dem Elektrolyten aus elektrischer Sicht eine Spannungsquelle darstellen. Durch die Redoxreaktionen wird die gespeicherte chemische Energie in elektrische Energie umgewandelt. Der Aufbau von LIB Zellen besteht aus dem Aktivmaterial, welches direkt die Kapazität der Zelle bestimmt sowie aus weiteren Komponenten, die für einen sicheren und optimalen Betrieb der Zelle integriert werden. Abbildung 2 zeigt den grundsätzlichen Aufbau und die Funktionsweise einer Lithium-Ionen-Batterie Zelle. Bei der geladenen Batterie sind die Lithium-Ionen Li^+ in der Anode interkaliert, gehen während der Entladung im Elektrolyt in Lösung und diffundieren durch den Separator zur Kathode. Dort werden die Lithium-Ionen interkaliert. Der Fluss der Elektronen e^- findet über die Ableiter und den externen Kreislauf statt. In Laderichtung dreht sich die Flussrichtung der Lithium-Ionen und Elektronen entsprechend um. [4]

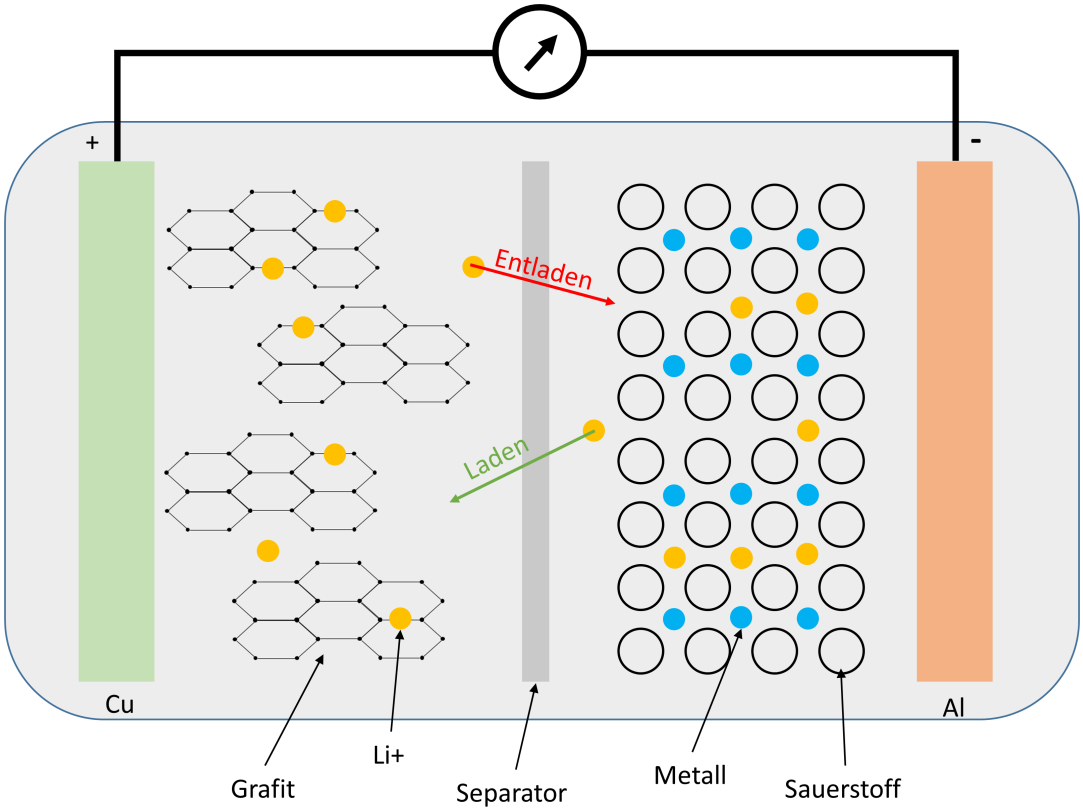


Abbildung 2 Funktionsweise einer Lithium-Ionen-Batterie mit Anode, Separator, Kathode und den Ableitern

Die zugrundeliegende Oxidation der Lithium-Eisenphosphat-Batterie (LFP) Kathode ist in Gleichung 2.1 für den Ladefall dargestellt.

$$\mathrm{LiFePO_4} \xrightarrow{\text{Laden}} \mathrm{FePO_4} + \mathrm{Li^+} + \mathrm{e^-} \tag{2.1}$$

Gleichung 2.2 zeigt exemplarisch die Reduktion an der Graphit Anode.

$$\mathrm{C_6} + \mathrm{Li^+} + \mathrm{e^-} \xrightarrow{\text{Laden}} \mathrm{LiC_6} \tag{2.2}$$

Die daraus resultierende Redoxreaktion ist in Gleichung 2.3 sowohl für den Lade- als auch für den Entladefall gegeben.

$$\mathrm{FePO_4} + \mathrm{LiC_6} \underset{\text{Laden}}{\overset{\text{Entladen}}{\rightleftharpoons}} \mathrm{LiFePO_4} + \mathrm{C_6} \tag{2.3}$$

Die Wahl der Materialien beeinflusst die elektrischen Eigenschaften der Zelle. Die Ionenleitfähigkeit des Elektrolyten bestimmt maßgeblich die mögliche Lade- und Entladerate. Flüssige Elektrolyte finden aufgrund der hohen technologischen Reife, der hohen Ionenleitfähigkeit und der großen thermischen Stabilität derzeit einen großen Anwendungsbereich. Die Elektrolytlösung besteht typischerweise aus einem Solvat (Lithiumhexaflourphosphaten $\mathrm{LiPF_6}$) und einem Lösungsmittel wie Diethylcarbonat (DEC) oder Dimethylcarbonat (DMC). [5]

Der Separator ist eine poröse Membran, die eine Ionen Diffusion zulässt, aber elektrisch isoliert. Damit wird eine Trennung der beiden Aktivmaterialien sowohl räumlich als auch elektrisch gewährleistet. Bezeichnenderweise werden Separatoren aus Polypropylen oder Polyethylen [6] mit einer Dicke von weniger als 25 µm hergestellt [7].

Die Stromableiter dienen zur Kontaktierung der Zelle bzw. des Aktivmaterials mit dem angeschlossenen externen Stromkreis. Durch die Auswahl der Materialien, Kupfer auf der Anodenseite und Aluminium auf der Kathodenseite, wird ein möglichst kleiner elektrischer Widerstand realisiert [8]. Dieser Widerstand wirkt sich direkt auf die Überspannungen unter Belastung aus.

2.1.2. Aktivmaterialien

Es existieren einige verschiedene Aktivmaterialen, die in LIB eingesetzt werden. Die folgenden beschriebenen Anoden- und Kathodenmaterialien beziehen sich auf die in dieser Arbeit modellierten Batteriesamples.

- **Graphit:** Graphit ist das kommerziell am häufigsten verwendete Anodenmaterial (Béguin et al. 2005). Die Vorteile von Graphit liegen in den niedrigen Herstellkosten und den guten elektrochemischen Eigenschaften. Graphit ist eine Modifikation von Kohlenstoff und besteht aus Graphen-Schichten. Während der Lithium De-/Interkalation lagern sich die Lithium-Ionen zwischen den Graphen-Schichten ein, wobei sich das Material nur gering mechanisch ausdehnt. Graphit kommt in der Natur makrokristallin oder als amorpher Kohlenstoff vor. Die synthetische Herstellung von Graphit erfolgt in der Flake Form [9]. In Kombination mit den gängigen Kathodenmaterialien begünstigt das niedrige Elektrodenpotential von 0,05 V gegenüber Li/Li^+ ein hohes Spannungsniveau [4]. Die theoretische Kapazität von Graphit liegt bei 372 mAh/g, welche bei anderen Lithium-Legierungen wie Silizium $\mathrm{Li_4 \cdot 4\,Si}$ mit 4200 mAh/g oder Zinn $\mathrm{Li_4 \cdot 4\,Sn}$ mit 994 mAh/g deutlich größer ist [10]. Die Solid-Electrolyte-Interphase (SEI) ist eine Passivierungsschicht an der Anode, die zum einen für einen stabilen Betrieb der Zelle notwendig ist, andererseits aber auch Gefahren in sich birgt. Die SEI bildet sich initial in den ersten Zyklen einer Zelle, vor allem aus der Zersetzung des Elektrolyten. Dabei werden ca. 8-15 % des zyklisierbaren Lithiums in Abhängigkeit von der Elektrolytzusammensetzung sowie der Menge an Aktivmaterial verbraucht [11]. In erster Linie dient die SEI als Schutz vor der elektrochemischen Reduktion des Elektrolyten sowie als Schutz vor einer chemischen Reaktion zwischen Anode und Elektrolyt. [12]. Während der Zyklisierung der Zelle wächst die SEI weiter. Dabei verliert die Zelle zyklisierbares Lithium und indessen sinkt die Leitfähigkeit der SEI [12]. Durch die mögliche exotherme Dekomposition der SEI-Schicht erhöht sich zusätzlich die Gefahr eines Thermal Runaways [13].

- **LTO:** Ein weniger weit verbreitetes Material für Anoden ist Lithium-Titanat (LTO) $\mathrm{Li_4Ti_4O_{12}}$. Nachteile von LTO Anoden sind die geringere Energiedichte und die hohen Kosten. Dem gegenüber stehen die höhere Sicherheit und die Leistungsfähigkeit. Durch das im Vergleich zu Graphit Anoden höhere Potential von 1,6 V gegenüber Li/Li^+ reduziert sich die gesamte Zellspannung und dadurch auch die Energiedichte der Zelle. Die theoretische spezifische Kapazität von 175 mAh/g ist verglichen mit Graphit deutlich geringer [10]. Die Interkalation der Lithium-Ionen erfolgt unter einer sehr geringen mechanischen Ausdehnung, was einen positiven Effekt auf die Alterung der Zelle hat. Aus diesem Grund können Zellen mit LTO Anoden sowohl eine sehr große zyklische Stabilität als auch sehr hohe Stromraten erreichen [14]. Aufgrund des hohen Potentials gegenüber Li/Li^+ entsteht nahezu keine SEI-Schicht [15]. Die fehlende SEI-Schicht und die Tatsache, dass LTO Anoden selbst bei Temperaturen um 100℃ keinerlei gasförmige Dekompositionspodukte erzeugen, führen zu

einer sehr hohen Sicherheit in Bezug auf Thermal Runaways [16].

- **NMC:** Das am weitesten verbreitete Kathodenmaterial ist Lithium-Nickel-Mangan-Cobalt-Oxid (NMC), häufig in der Form $\mathrm{LiNi_{1/3}Co_{1/3}Mn_{1/3}O_2}$ (1:1:1-NMC). Eine Erhöhung des Nickelanteils zu 6:2:2-NMC sowie 8:1:1 führt zu einer größeren spezifischen Kapazität mit tendenziell niedrigerer Stabilität [17]. Neben der Tendenz nickelreiche Materialien zu verwenden sinkt der Anteil an Kobalt. Die Reduzierung des Kobaltanteils hat keinen Einfluss auf die Stabilität und die Sicherheit der Zelle [18]. Außerdem können Probleme durch den Einsatz von Kinderarbeit und gefährlichen Arbeitsbedingungen beim Abbau von Kobalt vermieden werden [19]. Die theoretische spezifische Kapazität von 1:1:1-NMC liegt bei 280 mAh/g und ist damit höher als die von anderen Kathodenmaterialien wie LFP, Lithium-Mangan-Oxid (LMO) oder Lithium-Nickel-Cobalt-Oxid (NCA) [17]. NMC hat mit einem Potential von 3,8 V gegenüber $\mathrm{Li/Li^+}$ eine hohe Spannungslage, was wiederum eine hohe Energiedichte begünstigt [20]. NMC Kathoden weisen eine sehr gute Performance auf, Stromraten bis zu 10 C sind hier üblich [21]. Die thermische Stabilität von NMC ist besser als die von anderen Kathoden mit Schichtstrukturen, wie NCA oder LCO, aber schlechter als die oktaedrische Phosphat Struktur von LFP [13].
- **LFP:** LFP $\mathrm{LiFePO_4}$ wurde durch das Doping von Kohlenstoffpartikeln und der damit verbundenen besseren Hochstromfähigkeit zu einem konkurrenzfähigen Kathodenmaterial [17, 22]. Sowohl die Schnelladefähigkeit als auch die thermische Sicherheit bieten Vorteile gegenüber anderen Materialien [13]. LFP ist bis zu 250 °C thermisch stabil, es gibt keine Sauerstoffentwicklung und es hat die besten Überladeeigenschaften [23]. Die Volumenänderung während der Zyklisierung beträgt nur 6,77 % [24], was eine große Lebensdauer begünstigt [25]. Die niedrige Energiedichte von LFP Kathoden resultiert zum einen aus der im Vergleich zu NMC geringen theoretischen spezifischen Kapazität von 170 mAh/g [17] und zum anderen aus dem relativ niedrigen Potential gegenüber $\mathrm{Li/Li^+}$ von 3,4 V [20].

2.1.3. Leerlaufspannung und Überspannugen

Die Differenz der beiden Potentialkennlinien der Aktivmaterialien bestimmt die Leerlaufspannung (engl. open-circuit-votlage (OCV)) einer Zelle. Die Nenn-OCV addiert sich bei C-LFP Zellen auf ca. 3,2 V, sowie bei LTO-NMC Zellen auf ca. 2,4 V. Durch das hohe Potential der LTO Anode gegenüber $\mathrm{Li/Li^+}$, liegt die Spannung der LTO-NMC Zelle deutlich unterhalb der von einer C-LFP Zelle. Des weiteren ist der Verlauf der OCV Kennlinie von LFP-Zellen im mittleren SOC Bereich sehr flach.

Die tatsächlich an den Batteriepolen gemessene Spannung U_{pol} ist für die meisten Anwendungsfälle die entscheidende Größe, da anhand dieser die Betriebsstrategie der Zelle bzw. des Batteriepacks ausgelegt und angepasst wird. Diese setzt sich zum einen aus der OCV und zum anderen aus Überspannungen U_{over} zusammen, die sich je nach aktueller und vergangener Belastung der Zelle einstellen.

$$U_{pol} = OCV + U_{over} \tag{2.4}$$

Die Überspannungen der Zelle setzen sich aus drei Effekten zusammen:

- **Widerstände:** Es tritt ein instantaner Spannungsabfall unter Belastung durch ohmsche Widerstände in Ableitern, Aktivmaterialien und dem Elektrolyten auf. Der Widerstand ist stark temperaturabhängig.
- **Diffusionsüberspannungen:** Der Ausgleich von Konzentrationsunterschieden im Elektrolyten und den Aktivmaterialien geschieht aufgrund der Diffusion. Die auftretende Spannung ist strom-, zeit- und temperaturabhängig.
- **Durchtrittsüberspannung:** Die Überspannung entsteht durch den Phasenübertritt bei Redoxreaktionen im Aktivmaterial, welche durch die Butler-Volmer-Gleichung beschrieben werden kann. Die auftretende Spannung ist strom-, zeit- und temperaturabhängig.

Aufgrund der vielen nichtlinearen Abhängigkeiten, ist die aus dem Betrieb der Zelle resultierende Spannung eine komplexe Größe der Batterie. Sie ist sowohl von den aktuellen als auch von vergangenen Belastungen und den Stati der Zelle abhängig.

2.1.4. Batteriekennwerte

Der Ladezustand der Batterie ist ein sowohl für die Modellierung der Batterie als auch für die Berechnung der entnehmbaren Energiemenge verwendeter Status. Viele interne Batterieparameter und Effekte, wie der Innenwiderstand oder die Diffusion, sind abhängig von der Lithiumkonzentration in den Aktivmaterialien und damit indirekt abhängig vom SOC. Der SOC wird berechnet mit der noch entnehmbaren Kapazität Q_{ent} im Verhältnis zur Nennkapazität Q_{nenn}:

$$SOC = \frac{Q_{ent}}{Q_{nenn}} \tag{2.5}$$

Der SOC ist eine Hilfsgröße, die physikalisch nicht direkt gemessen werden kann. Für die exakte Bestimmung müsste die Batterie komplett entladen werden und dabei der gemessene Strom über die Zeit integriert werden.

$$Q_{ent} = \int_{SOC_{ini}}^{SOC_0} I_{Batt} dt \tag{2.6}$$

In der realen Anwendung im Fahrzeug ist eine Entladung der Batterie zur Bestimmung des SOCs zu aufwändig oder nicht möglich, da die Batterie weiter belastet und geladen wird. Ist der initiale SOC bekannt, kann das Amperestundenzählen den SOC in Echtzeit

berechnen. Nachteil dieses Verfahrens ist, dass Fehler in der Strommessung durch die Integration akkumuliert werden. Dies ist vor allem für lange Berechnungszeiträume nicht zu vernachlässigen. [26]

Eine weitere Möglichkeit der Bestimmung des SOCs besteht über die OCV. Ist die OCV Kennlinie der Batterie bekannt, kann anhand einer Ruhespannungsmessung bestimmt werden, welcher SOC vorliegt. Dafür müssen alle Prozesse, die Überspannungen verursachen, abgeklungen sein. Dies bedeutet, dass die Batterie für mehrere Stunden nicht belastet werden darf. Da die OCV Methode in den meisten Applikationen unpraktikabel ist und die Amperestundenbilanzierung die Problematik der Fehlerintegration hat, wird der SOC meist mithilfe anderer Verfahren geschätzt.

Adaptive modellbasierte Filtermethoden kombinieren das Amperestundenzählen mit modellbasierten Methoden, um die SOC Schätzung zu korrigieren. Dabei gibt es Ansätze, die unter anderem einen Kalman-Filter KF [27], einen extended Kalman-Filter [28] oder unscented Kalman-Filter [29] nutzen.

Der State-of-Power (SOP) einer Batteriezelle gibt an, wieviel elektrische Leistung unter Berücksichtigung der geltenden Rahmenbedingungen einer Zelle entnommen oder zugeführt werden kann. Auf Zellebene sind die Rahmenbedingungen bestimmt durch die vom Hersteller definierten Strom- und Spannungsgrenzen, die einen sicheren Betrieb der Zelle gewährleisten. Darüber hinaus können die Rahmenbedingungen für ganze Batteriepacks je nach Anwendungsfall verschärft werden. Gelten für den Strom die maximalen Grenzen I_{max_dis} und I_{max_charge} sowie für die Spannung die Grenzen U_{min} und U_{max}, so kann die maximale Entladeleistung P_{max_dis} wie folgt dargestellt werden:

$$P_{max\_dis} = \begin{cases} I_{max\_dis} * U & \text{wenn } U > U_{min}, \\ I * U_{min} & \text{sonst.} \end{cases} \tag{2.7}$$

Analog dazu wird die maximale Ladeleistung P_{max_charge} definiert zu:

$$P_{max\_charge} = \begin{cases} I_{max\_charge} * U & \text{wenn } U < U_{max}, \\ I * U_{max} & \text{sonst.} \end{cases} \tag{2.8}$$

Die Leistungsfähigkeit einer Batterie ist zudem von der Dauer der Belastung abhängig. Durch längere Belastungen kumulieren sich die Polarisationsüberspannungen und der Ladezustand der Batterie ändert sich ggf. signifikant. Der tatsächliche Strom an der Batterie ergibt sich aus der Superposition aller Strombedarfe der Verbraucher sowie der Stromabgabe der Erzeuger. Die Spannung, die sich im Bordnetz einstellt, ist hingegen rein von der Batterie als einzigen Spannungserzeuger abhängig. Damit ist die SOP-Modellierung im Bordnetz in erster Linie eine Frage der Spannungsreaktion der Batterie

auf die superpositionierten Strombedarfe und –abgaben der Verbraucher bzw. Erzeuger. Die Leistungsfähigkeitsprognose der Batterie wird somit zu einer Abfrage im Bordnetz, ob die Spannungsgrenzen U_{max} und U_{min} bei einem zeitlich abhängigen Bordnetzstrom $I_{BN(t)}$ eingehalten werden:

$$U_{min} < U(I_{BN(t)}) < U_{max} \tag{2.9}$$

Um die Vergleichbarkeit verschiedener Zellen bezüglich des Stromes herzustellen, wird die C-Rate eingeführt. Die C-Rate wird definiert über das Verhältnis zwischen Strom I und Nennkapazität Q_{nenn}.

$$C_{rate} = \frac{|I|}{Q_{nenn}} \tag{2.10}$$

Die maximale C-Rate unterscheidet sich je nach Zellchemie in Entlade- und Laderichtung [30].

2.1.5. Batteriemanagementsystem

Das Batteriemanagement-System (BMS) vereinigt verschiedene Funktionalitäten in einem übergeordneten System, um die Zellen sicher und effizient betreiben zu können. Pro Energiespeicher, der aus mehreren hundert Zellen bestehen kann, wird ein (Master-) BMS eingesetzt, welches alle einzelnen Zellen und Module überwacht. [31]

Die wesentlichen Funktionen eines BMS kann in sieben Kategorien unterteilt werden [32]:

- **Zustandsmessung:** Die physikalisch messbaren Größen Spannung, Strom und Temperatur werden über Sensoren erfasst und an die weiteren Funktionen weitergegeben. Diese Sensoren sind meist an jeder einzelnen Zelle bzw. an jedem Modul integriert. Strom wird meist mit hochgenauen Shunts gemessen, die Temperatur mit Thermistoren [32].
- **Status Schätzung:** Die gemessenen Werte aus der Zustandsüberwachung werden an einen Micro Controller gesendet, der die zusätzlichen Status schätzt und überwacht. Einer der wichtigsten Status ist der SOC, der angibt wie viel Kapazität anteilig an der Gesamtkapazität entnommen werden kann [33, 34]. Neben der direkten Aussagekraft des SOCs über die entnehmbare Kapazität, können anhand des SOCs auch andere Batterieparameter wie Impedanz und Innenwiderstand genauer geschätzt werden. Die unterschiedlichen Herangehensweisen für eine SOC-Modellierung werden in Kapitel 2.1.6 diskutiert. Der Gesundheitszustand (engl. State-of-Health (SOH)) der Batterie definiert sich über die aktuelle Restkapazität der Batterie im Verhältnis zur Nominalkapazität. Sowohl durch Zyklisierung als auch während der Lage-

rung verlieren Batterien aufgrund von internen Alterungsmechanismen zyklisierbare Kapazität [25]. Durch die Alterung steigt auch der Innenwiderstand der Zelle. Der Innenwiderstand [35] sowie die Impedanz [36] werden geschätzt und für weitere Modelle wie die Leistungsschätzung [37], den Funktionsstatus (engl. State-of-Function (SOF)) und den Sicherheitszustand (engl. State-of-Safety (SOS)) [38] genutzt.

- **Sicherheit:** Durch die Wahl der Komponenten und deren chemische Eigenschaften werden die elektrischen Betriebsbereiche einer Zelle festgelegt. Die gemessenen Werte werden permanent mit den Spannungs- und Stromgrenzen verglichen. Bei einer Verletzung dieser Extrema werden Maßnahmen eingeleitet, um den sicheren Betrieb der Zelle zu gewährleisten. In erster Linie werden Maßnahmen zur Reduzierung der Leistung ergriffen, die einen weiteren Betrieb der Zelle ermöglichen. In zweiter Instanz kann die Zelle oder das gesamte Batteriepack durch ein Trennelement vom externen Stromkreis getrennt werden und somit vor Über- oder Unterspannungen sowie vor zu großen Strömen geschützt werden [39]. Thermisch wird zum einen der absolute Wert getrackt und dementsprechend an das thermische Management Anweisungen zum Heizen oder Kühlen gegeben. Zum anderen wird aber auch der Temperaturgradient beobachtet, da dieser ein Indikator für einen Thermal Runaway darstellt [40].

- **Ladekontrolleinheit:** In der Ladekontrolleinheit wird die Strategie und Methode für das (Ent-) Laden festgelegt und überwacht. Neben den maximalen Stromraten werden auch Optimierungen hinsichtlich der Ladezeit und dem Einfluss auf die Alterung der Zelle durch das Laden vorgenommen [39].

- **Thermisches Management:** Das thermische Management nutzt zum einen die Messwerte der Temperatur als auch thermische Modelle, um das Heizen und Kühlen zu steuern. Die einzelnen Zellen in den Batteriepacks werden aktiv durch einen Heiz-/Kühlkreislauf mit einem Liquid direkt umströmt oder durch Kühlplatten indirekt temperiert [41]. Ziel ist es, einen möglichst optimalen Temperaturbereich der Zellen zu erreichen. Typischerweise liegt der in Bezug auf Alterungseffekte optimale Temperaturbereich von Batterien während dem Betrieb im Bereich von 20 °C bis 45 °C [42] und während der Lagerung bei 25 °C [43]. Außerdem kann ein thermisches Event erkannt und durch Kühlung verzögert werden.

- **Balancing:** Die Temperaturinhomogenitäten zwischen den Zellen führen zu Innenwiderstands- und Alterungsdifferenzen [44]. Zusätzlich gibt es in der Herstellung der Batterien Differenzen in Bezug auf ihre Kapazität und den Innenwiderstand der Zellen, die zwar noch im Toleranzbereich sind, aber einen Einfluss auf das Zellverhalten im Verbund haben [45]. Diese Abweichungen der Batterieparameter führen im Betrieb zu Abweichungen im SOC und SOH, wodurch die nutzbare Ladungsmenge des Batteriepacks und damit auch die Leistung sinken [44]. Um den Ladungsdifferenzen entgegenzuwirken, werden Balancing-Schaltungen integriert. Die Methoden werden danach klassifiziert, ob das Verfahren dissipativ ist. Dissipative Systeme sind

mit geringerem Aufwand zu integrieren und ebenso weniger fehleranfällig, jedoch ist der thermische Einfluss auf das Gesamtsystem hier nicht zu vernachlässigen [44]. Nicht dissipative Systeme leiten die Energie direkt an andere Zellen weiter [46, 47]. Das Balancing der Zellen wird vom BMS, je nach festgestelltem Bedarf, getriggert.

- **Topologie und Schnittstellen:** Die Topologie von BMS in Energiespeichern kann zentralisiert, modularisiert oder dezentralisiert aufgebaut sein. Zentrale BMS nutzen Sensoren, um die Zustände der einzelnen Zellen zu erfassen. Wohingegen modularisierte BMS auf eine Master–Slave Verwaltung zurückgreifen. Ein dezentrale BMS besteht aus mehreren gleichen BMS Einheiten, die ausschließlich miteinander kommunizieren [48, 49]. Die Kommunikation nach außen, zu anderen Steuergeräten oder übergeordneten Steuergeräten, findet meist über CAN-Bus statt [39].

2.1.6. Modellierung

Die nicht messbaren Zustände einer Batterie, wie der SOC, SOH und SOP, sind wichtige Kenngrößen für ein BMS. Im Gegensatz zu physikalischen Größen (Spannung, Strom und Temperatur) können diese Zustände nicht direkt gemessen werden und müssen mit Modellen geschätzt werden.

In der Literatur werden die unterschiedlichen Modellierungsansätze für den SOC breit diskutiert. Einen etablierten und seit Langem weit verbreiteten Ansatz stellt die Modellierung mit einem Ersatzschaltbildmodell (ESB) dar. ESBs modellieren die makroskopischen Effekte der elektrochemischen Prozesse, welche in einer Batteriezelle während der Ladung und Entladung auftreten. Die Spannungspolarisierung, die durch nichtlineare Effekte aufgrund von Diffusion, Ladungstransfer und die elektrochemische Doppelschicht entstehen, werden mit einem oder mehreren RC-Gliedern modelliert. Instantane Spannungsabfälle werden durch einen einfachen Widerstand dargestellt. Die Komponenten des ESB werden in Abhängigkeit von Umgebungsfaktoren parametriert. In der Implementierung werden die einzelnen Parameter der Komponenten per Interpolation aus Look-Up-Tables generiert. Für das Parameter-Fitting müssen hochgenaue Messungen durchgeführt werden, die einem bestimmten Schema folgen. Madani et al. [50] haben ein ESB mit zwei RC Gliedern auf eine LTO-Batterie angewendet. Bei einer Untersuchung von Farmann [51] verdeutlichte sich, dass eine hohe Anzahl an RC-Gliedern theoretisch einen Genauigkeitsvorteil bringen sollte, jedoch mehr als zwei RC-Glieder tatsächlich keinen merkbar positiven Effekt liefern. Physikochemische Modelle basieren auf den fundamentalen Gleichungen aller physikochemischen Effekte in der Batteriezelle. Dadurch sind die Modelle inhärent genauer und für weitere Abschätzungen wie die Alterung interpretierbar, jedoch sind sie sehr rechenintensiv [32].

Eine weitere Methode zur SOC-Modellierung ist der Einsatz von adaptiven modellbasierten Filtermethoden. Der Kalman-Filter schätzt Systemzustände anhand von fehlerbehafteten Beobachtungen und minimiert den Fehler der Zielgröße. Ding et al. [28] haben den SOC einer LFP-Batterie mit einer Weiterentwicklung des KF, den extended-KF, model-

liert. Meng et al. [34] nutzen sowohl KF als auch extended-KF und unscented-KF für eine kaskadierte SOC-Modellierung.

Die aktuelle Literatur weist zusätzlich zu den klassischen ESB und KF Modellen ebenso erste datengestützte Modellierungsansätze auf. Neben den Anwendungen von fuzzy logic [52] und feed forward neural networks (FFNN) [53] zeigen aktuelle Forschungen gute Ergebnisse im Bereich von RNN. Der Vorteil von RNN ist die Möglichkeit, Zeitabhängigkeiten in die Modellierung zu integrieren. Da zu lange Betrachtungszeiträume bei einfachen RNNs die Ergebnisse verschleiern lassen (vgl. vanishing gradients), nutzen aktuelle Modellierungsansätze [54, 55] Long Short-Term Memory (LSTM) erfolgreich zur SOC-Modellierung.

Vidal et al. [56] zeigten in ihrer Untersuchung, dass RNNs bessere Modellierungsergebnisse liefern als FFNNs, jedoch ähnliche Ergebnisse wie Modelle mit KF hervorbringen.

Die SOH-Modellierung bedient sich an den gleichen Ansätzen wie die SOC-Modellierung. In der Literatur finden sich sowohl Beispiele für die Modellierung des SOHs mit einem ESB [57], als auch mit modellbasierten Filtermethoden wie KF und deren Weiterentwicklungen [58, 59], sowie datengestützte Modelle auf Basis von NN [60, 61, 62].

Die Modellierung des SOP kann je nach Anwendungsfall, wie in Kapitel 2.1.4 bereits ausgeführt wurde, entweder über die tatsächliche Leistungsfähigkeit (vgl. Gleichungen 2.7 und 2.8) oder über die Einhaltung der Spannungsgrenzen (vgl. Gleichung 2.9) erfolgen. Im Nachfolgenden werden sowohl Spannungsmodelle als auch reine Leistungsschätzungen als SOP-Modelle aufgefasst.

Farmann et al. [63, 51] haben die SOP-Modellierung in zwei grundlegende Methoden unterteilt. Zum einen sind das Prädiktionen basierend auf adaptiven charakteristischen Look-up-Tables (LUT), welche in Abhängigkeit von Batterieparametern SOC, Temperatur, Dauer des Leistungspulses, angeforderte Leistung und Spannung eine mögliche Leistungsfähigkeit ausgeben. Die Parametrierung dieser LUTs ist mit einer Vielzahl von spezifizierten Messungen, wie Hybrid Pulse Power Characterization (HPPC) und Elektrochemische Impedanzspektroskopie (EIS) relativ aufwändig. Dagegen ist die Rechengeschwindigkeit des Modells in der Anwendung auf dem BMS sehr hoch. In der Vergangenheit wurden einige dieser LUTs aufgestellt [64, 65, 66]. Zum anderen wird von den Autoren die modellbasierte Prädiktion genannt. Ähnlich wie bei der SOC Schätzung wird dafür ein ESB Modell mit einem Widerstand und ggf. mehreren RC-Gliedern parametriert. Beispielsweise wird in [67] eine Leistungsprognose auf einem einzelnen Widerstand, dem Direct Current Resitance (DCR), aufgebaut. Ein einzelner Widerstand kann die Anforderung, die elektrochemischen Prozesse in der Batterie so genau wie möglich nachzustellen, nicht erfüllen. Aufgrund der Komplexität der Prozesse innerhalb einer

Batterie ist die Modellierung nur mit sehr großem Aufwand und mehreren zusätzlichen RC-Gliedern möglich. Daher werden die ESB um KF ergänzt, um den Schätzungsfehler zu minimieren. Neben den Ansätzen mit einem klassischen KF [68] sind in der Literatur ebenso Erweiterungen zum KF, wie dem adaptive-extended-KF (AEKF) [69] oder dem dual-extended-KF (DEKF) [70], erfolgreich validiert worden.

Die datengestützte Modellierung ist in der SOP-Prädiktion verglichen mit der SOC- oder SOH-Modellierung noch nicht in gleichem Umfang in der Literatur zu finden. Haiying et al. [71] haben ein SOP-Modell mit einem FFNN aufgestellt, welches jedoch keine zeitliche Abhängigkeit in den Eingangsvariablen besitzt. Fleischer et al. [72, 73, 74] haben eine Spannungsprognose mit einem Adaptiven Neuro-Fuzzy-Inferenzsystem (ANFIS) auf eine KF SOC-Schätzung aufgebaut. Die Idee hinter ANFIS ist, die Vorteile der Fuzzylogik und der künstlich neuronalen Netzwerke zu vereinen, um somit eine effektive Modellierung für nichtlineare Verhaltensmuster zu schaffen. Zeitliche Abhängigkeiten können in diesem Ansatz nicht berücksichtigt werden. Zhao et al. [75] haben eine Spannungsprognose mit einem RNN implementiert. Die Modellierung erfolgt mit zwei Gated Recurrent Unit (GRU) Schichten mit je 30 hidden Units. Als Eingangsfeature dienen der SOC, die Leistung, die Temperatur und die Spannung im vorherigen Zeitschritt. Damit kann die zeitliche Abhängigkeit der Spannungsentwicklung während der Belastung der Zelle abgebildet werden. Modelliert werden zwei Batterien, eine Panasonic NCA und eine Sony NMC Zelle, anhand von dynamischen Batteriemessungen im Temperaturbereich von -10 °C bis 25 °C. In der Dissertation von Carlos Vidal [76] wird ein RNN Spannungsmodell sowohl mit LSTM Zellen als auch mit GRU Zellen trainiert und anschließend mit der herkömmlichen ESB Modellierung verglichen. Die RNN Modelle bestehen aus zwei Layern mit jeweils zehn bis zwölf hidden Units. Als Inputfeatures dienen der SOC, der Strom und die Temperatur. Das ESB ist ein Modell dritter Ordnung mit einem nichtlinearen Widerstand, parametrisiert mit einem HPPC Test Verfahren. Der Vergleich der Modellierungsansätze zeigt, dass das LSTM Modell das Spannungsverhalten der Batterie deutlich besser schätzt als das ESB. Dieses Verhalten ist vor allem bei kälteren Temperaturen zu beobachten. Im Vergleich der RNN Technologien zeigt das LSTM Netzwerk eine leicht bessere Genauigkeit gegenüber dem GRU Netzwerk.

2.2. 48 V Bordnetz

Neben den klassischen 12 V Bordnetzen in verbrennungsmotorisch angetriebenen Fahrzeugen (engl. Internal Combustion Enginge Vehicle (ICV)), den Hochvolt-Bordnetzen von Plug-In Hybriden Fahrzeugen (engl. Plug-In Hybrid Vehicle (PHEV)) und batterieelektrische Fahrzeuge (engl. Battery Electric Vehicle (BEV)) werden 48 V Bordnetze als leistungsstarke Ergänzung eingesetzt.

2.2.1. Einsatzzweck

48 V Bordnetze dienen in erster Linie dazu, den Antriebsstrang zu elektrifizieren und dadurch Emissionen einzusparen. Die Kraftstoffeinsparungen werden erzielt, indem die rekuperierte und zwischengespeicherte Bremsenergie anschließend für das Boosten, Segeln oder den Start-Stopp Betrieb verwendet wird. Hybride Antriebssysteme, die diese Funktionalitäten aufweisen, jedoch nicht in der Lage sind weite Strecken rein elektrisch zu bewerkstelligen, werden oft als MHEV bezeichnet [77].

Die Auslegung der E-Maschine ist abhängig von der Topologie. MHEVs in einer seriell-Hybriden Topologie müssen aufgrund der Leistungsübertragung aus dem Verbrenner sehr groß dimensioniert werden, für seriell-parallel-Hybride Topologien werden zwei E-Maschinen benötigt. Aus diesen Gründen werden MHEVs in der Regel als parallel-Hybride Topologie ausgeführt [3].

Die E-Maschine kann an verschiedenen Stellen im Antriebsstrang verbaut werden. Folgend werden die möglichen Topologien aufgelistet und beschrieben:

- **P0-Hybrid:** Der 12 V Generator wird an gleicher Stelle ersetzt durch eine 48 V E-Maschine, die sowohl generatorisch als auch motorisch über den Riemen mit dem Verbrennungsmotor betrieben werden kann [78]. Der Vorteil bei dieser Bauweise liegt in der einfachen Integration in den Antriebsstrang. Jedoch kann nur eine begrenzte Leistung über den Riemen übertragen werden [79].
- **P1-Hybrid:** In dieser Topologie ist die E-Maschine direkt auf der Kurbelwelle des Motors verbaut. Die starre mechanische Verbindung mit der Antriebswelle ermöglicht zwar höhere Leistungen, jedoch kann die E-Maschine nicht betrieben werden, ohne dass sich der Verbrennungsmotor mitdreht [79].
- **P2-Hybrid:** Die E-Maschine ist zwischen der Kupplung und dem Getriebe angeordnet. Daraus ergibt sich die Möglichkeit das Fahrzeug rein elektrisch anzutreiben. Aus diesem Grund ist der Einsatz von Motor-Aus-Segeln und elektrischem Kriechen möglich [79]. Eine Spezialform stellt ein E-Doppelkupplungsgetriebe dar, wobei die E-Maschine im Getriebe am Eingang einer der beiden Wellen angebracht ist [80].
- **P3-Hybrid:** Die E-Maschine ist am Getriebeausgang angebracht. Dabei können die Übersetzungen des Getriebes nicht genutzt werden. Ansonsten besitzt dieser Hybrid ähnliche Eigenschaften wie ein P2-Hybrid [77].
- **P4-Hybrid:** In dieser Bauweise wird ein Allradsystem realisiert, indem eine Achse vom Verbrennungsmotor und eine andere von der E-Maschine angetrieben wird. Hier ist eine sehr leichte Integration möglich, da kein Eingriff auf den herkömmlichen Antriebsstrang stattfindet [77].

Das Kraftstoffeinsparpotential hängt von der Bauart ab. Bei P0 und P1 Topologien wer-

den zwischen 4 % und 8 % Kraftstoff eingespart. Bei P2, P3 und P4 Topologien sind es bis zu 15 % [78]. Die tatsächliche Einsparung ist von der Auslegung der Komponenten und der Betriebsstrategie abhängig.

Der größte Teil der Einsparung wird durch das Rekuperieren der Bremsenergie erreicht [81]. Beim Rekuperieren wird die E-Maschine generatorisch betrieben und somit elektrische Leistung erzeugt. Die Energie, welche sonst durch mechanische Bremsen in Wärme umgewandelt worden wäre, wird nun in der Batterie zwischengespeichert [82]. Der Wirkungsgrad der Rekuperation ist abhängig von der gewählten Topologie. Je näher die E-Maschine an der angetriebenen Achse liegt und somit weniger Komponenten zwischengelagert die Effizienz mindern, desto höher ist der Wirkungsgrad. P3 und P4 Topologien erreichen 91 %, P2 85 %, P1 79 %, und P0 72 %. Für das elektrische Fahren und Boosten gelten die gleichen Wirkungsgrade, da die Kraft über die gleichen Komponenten hin zur Achse übertragen wird [82]. Reines elektrisches Fahren kann nur von Topologien unterstützt werden, in denen die E-Maschine mit einer Kupplung vom Motor getrennt werden kann [79]. Die Unterstützung des Verbrennungsmotors durch die E-Maschine beim Beschleunigen kann von allen Topologien gewährleistet werden. Dadurch wird die Verbrennungskraftmaschine in einem effizienteren Betriebspunkt betrieben und die aus der Rekuperation gewonnene Energie kann abgegeben werden [83]. Eine weitere Funktionalität, die bereits in den Micro-Hybriden eingesetzt wird, ist das Start-Stopp-System. Hier wird der Verbrennungsmotor in Standphasen abgestellt und das Bordnetz rein aus den Batterien versorgt. Dabei ist immer sicherzustellen, dass das Fahrzeug startfähig bleibt [82]. Weiteres Einsparpotential bringt das Motor-Aus-Segeln mit sich. In diesem Zustand wird der Verbrennungsmotor während der Fahrt auch bei großen Geschwindigkeiten ausgeschaltet und das Fahrzeug rollt mit verminderter Reibung aus (passives Segeln) oder die Geschwindigkeit wird durch den Antrieb der E-Maschine gehalten (aktives Segeln). Die CO_2 Emissionseinsparung bei realen Geschwindigkeitsprofilen, wie im WLTP-Zyklus, liegt beim Einsatz von aktivem und passivem Segeln je nach Fahrzeugklasse zwischen 12 % und 14 %. [84] Die Start-Stop Funktionalität bringt eine erhebliche CO_2 Einsparung im urbanen Umfeld mit sich. Das Segeln hingegen ist vor allem auf extra urbanen Straßen bzw. auf Autobahnen effizient [81].

Die beschriebenen Betriebsmodi des MHEV verfolgen das Ziel der Kraftstoffeinsparung. Im Vergleich zu einem Fahrzeug mit einem Dieselmotor werden mit einem MHEV System ca. 9 % Kraftstoff und Harnstoff eingespart. Ein HEV spart im Vergleich ca. 22 % ein [85]. Die Vorteile eines MHEV liegen in der einfachen Integration und den damit verbundenen geringen Kosten. Dadurch, dass die VDE-Kleinspannungsgrenze von 60 V nicht überschritten wird, können die 48 V Komponenten im Vergleich zu 12 V Komponenten ohne erhöhte Schutz- und Sicherheitseinrichtungen verbaut werden. Durch die einfache Integrierbarkeit sowie die schnell zu erreichenden großen Stückzahlen dient der MHEV als Einstieg in die Elektrifizierung des Antriebes. Große Stückzahlen sind durch kleine Markteintrittsbarrieren möglich, da beispielsweise im Gegensatz zu BEVs oder

PHEVs keine Ladeinfrastruktur aufgebaut werden muss. Das Gewicht der zusätzlichen Komponenten wird teilweise durch die kleineren Leitungsquerschnitte im 48 V Bordnetz kompensiert. Neben den Vorteilen der CO_2 Emissionseinsparung gibt es auch kundenrelevante Vorteile. Ein elektrischer Zusatzverdichter erzeugt mehr Drehmoment im unteren Drehzahlbereich, wodurch ein komfortableres Fahrgefühl entsteht [86]. Die Leistungsfähigkeit des 48 V Bordnetzes ermöglicht eine weitere Verbesserung des Komforts durch Komponenten wie die elektrische Servolenkung oder die Wankstabilisierung [87].

2.2.2. Komponenten

Ein MHEV besteht im Kern aus den drei Komponenten E-Maschine, DC/DC Wandler sowie Batterie und kann um elektrische Verbraucher erweitert werden.

Der DC/DC Wandler koppelt das 12 V Bordnetz mit dem 48 V Bordnetz. Die meisten elektrischen Verbraucher sind auf der 12 V Seite, aber die Energie wird auf der 48 V Seite in das System eingebracht. Somit wird der DC/DC Wandler meist im Buck-Mode betrieben, um die Spannung im 12 V Bordnetz zu halten. In Ausnahmefällen, wie dem Verbrennungsmotorstart mit der 48 V E-Maschine, kann der Wandler auch im Boost-Mode betrieben werden, um die großen Leistungen im 48 V System zu dämpfen. Üblich sind Leistungen von 1 kW bis 3 kW. [87]

Die 48 V E-Maschine dient als direkter Ersatz für den 12 V Generator und je nach Auslegung des 48 V Bordnetzes auch als Ersatz für den Anlasser. Das Energiemanagementsystem entscheidet über den Betriebsmodus und die geforderte Leistung [87]. Die E-Maschinen werden typischerweise auf 10 kW – 15 kW Leistung ausgelegt. Eine höhere Leistung würde nicht zu signifikant mehr rekuperierbarer Energie während dem WLTP-Zyklus führen [88, 89].

Die Batterie dient im 48 V System als Puffer, um die beim Rekuperieren gewonnene Energie zwischenzuspeichern und für die Betriebsmodi Segeln, Motorstart und Boost bzw. für die elektrischen Verbraucher vorzuhalten. Somit hat die Kapazität der Batterie einen direkten Einfluss auf die CO_2 Emissionen eines MHEV. Die Abwägung zwischen mehr Gewicht und größerem Speicher resultiert in einem optimalen Energieinhalt von 0,3 kWh [81].

Ein zweiter Auslegungsparameter der Batterie ist die maximale Leistungsfähigkeit. Die Anforderung ist, dass die Rekuperationsleistung der E-Maschine von der Batterie aufgenommen werden kann. Bei einer Leistung von 15 kW und einem Batterieenergieinhalt von 0,5 kWh wird die Batterie mit 30 C geladen.

$$\frac{15\,\text{kW}}{0,5\,\text{kWh}} = 30\,\text{C} \tag{2.11}$$

Im Vergleich zu den Anforderungen bei einem BEV, ist diese Ladeleistung deutlich mehr herausfordernd. Der „Taycan“ von Porsche kann bei einem Batterieenergieinhalt von 95 kWh mit maximal 350 kW geladen werden, was zu einer maximalen C-Rate von 3,68 C führt [90].

$$\frac{350\,\mathrm{kW}}{95\,\mathrm{kWh}} = 3,68\,\mathrm{C} \tag{2.12}$$

Die hohen Stromraten stellen sowohl eine Herausforderung in der Funktionsweise der Batterie als auch in der Modellierung dar. Die Hochstromfähigkeit der Batterien wird durch die Diffusion im Aktivmaterial und die Leitfähigkeit begrenzt [91]. Des weiteren altern Batterien bei hohen C-Raten schneller [30].

2.3. Maschinelles Lernen

Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz, welches sich mit dem Lernen von Computern anhand von Daten beschäftigt. Dabei wird in drei algorithmische Ansätze unterteilt:

- **Unüberwachtes Lernen:** Anhand der ungelabelten Inputdaten extrahiert das Modell Merkmale, ohne dabei überwacht zu werden.
- **Reinforcement Lernen:** Ein Agent bewegt sich in einer bestimmten Umgebung und interagiert mit dieser. Vom Diskriminator bekommt er abhängig vom Status eine Belohnung bzw. Bestrafung. Ziel ist es, dass der Agent die Nutzenfunktion maximiert, um so implizit das bestrebte Verhalten zu erlernen.
- **Überwachtes Lernen:** Das Modell wird anhand von gelabelten Daten trainiert. Für jeden Input ist ein Label (Ground Truth) vorhanden, mit dem Ziel, dass das Modell jeden dieser Punkte möglichst genau prädiziert.

Bei diskreten Ergebnisräumen, spricht man von Klassifikation, bei kontinuierlichen Ergebnisräumen von Regression. Da in der vorliegenden Arbeit gelabelte, kontinuierliche Daten vohranden sind, wird vor allem das überwachte Lernen mit Regressionsmodellen betrachtet.

2.3.1. Trainingsprozess

Die Abbildung 3 zeigt den Trainingsprozess von überwachtem Lernen mit gelabelten Daten. In der Datenvorverarbeitung werden die Rohdaten vorverarbeitet, um ein möglichst genaues und effizientes Training zu erreichen. Nach der Datenvorverarbeitung werden die benötigten Features bestimmt und extrahiert. Die resultierenden Daten werden dann

dreigeteilt und für verschiedene Phasen des Trainings bzw. der Validierung eingesetzt. Somit wird sichergestellt, dass jeder Datenpunkt jeweils exklusiv in einem der drei Datensätze vorkommt.

- **Training Set:** Diese Daten werden verwendet, um das Netz zu trainieren. Das Netz lernt aus den vorhandenen Daten Muster und Zusammenhänge, indem die Gewichte und Biases mit dem Backpropagation Through Time Algorithmus angepasst werden. Die Gleichverteilung der Daten ist für das Training wichtig, da sonst eine algorithmische Voreingenommenheit in das Modell übertragen wird. Unterrepräsentierte Datenbereiche werden als unwahrscheinlich, bzw. als unwichtig eingestuft, überrepräsentierte Bereiche als besonders wahrscheinlich gesehen. Algorithmen zum Under- und Oversampling optimieren das Trainingsdatenset dahingehend. Mithilfe der Verlustfunktion wird der Loss (vgl. Formel 2.13) aus dem Trainingsdatensatz einer Epoche bestimmt.
- **Validation Set:** Das Validation Set wird für das Hyperparamertuning benutzt. Nach jeder Epoche die trainiert wird, werden Fehlermetriken wie der Validation Loss auf Basis des Validation Sets berechnet, die dann Einfluss auf die Wahl der Hyperparameter haben. Die Validierungsdaten überschneiden sich nicht mit den Trainingsdaten und können daher als ungesehene Daten zur Modellvalidierung genutzt werden.
- **Test Set:** Mit dem Test Set wird das final trainierte Modell bewertet. Die Daten sollten eine ähnliche Verteilung wie Validation und Training Set haben. Die vom Modell prädizierten Werte können direkt mit der Ground Truth aus dem Test Set verglichen werden.

Das Kreuzvalidierungsverfahren, bei welchem während des Trainingsprozesses dynamisch Datenblöcke zwischen Training Set und Validation Set getauscht werden, weist eine ähnliche Performance auf wie die last block Validierung. In der last Block Validierung wird ein Datensatz fest als Training Set verwendet und ein anderer fest als Validation Set [92].

Der Loss wird anhand der Verlustfunktion f_{loss} aus der Ground Truth GT und dem geschätzten Wert des Modells Y_{Pred} berechnet:

$$Loss = f_{loss}(GT, Y_{Pred}) \tag{2.13}$$

Die Metrik wird sowohl für das Trainingsdatenset als Training-Loss als auch für das Validationsset als Validation-Loss berechnet. Anhand dieser beiden Metriken kann abgeleitet werden, wie der Trainingsfortschritt ist und ob Overfitting vorliegt.

Um trainierte Modelle mit anderen zu vergleichen, werden für das Training Set separate Metriken berechnet. In der Literatur werden oft die mittlere quadratische Abweichung (engl. Mean Square Error (MSE)) und der mittlere absolute Fehler (engl. Mean Absolute

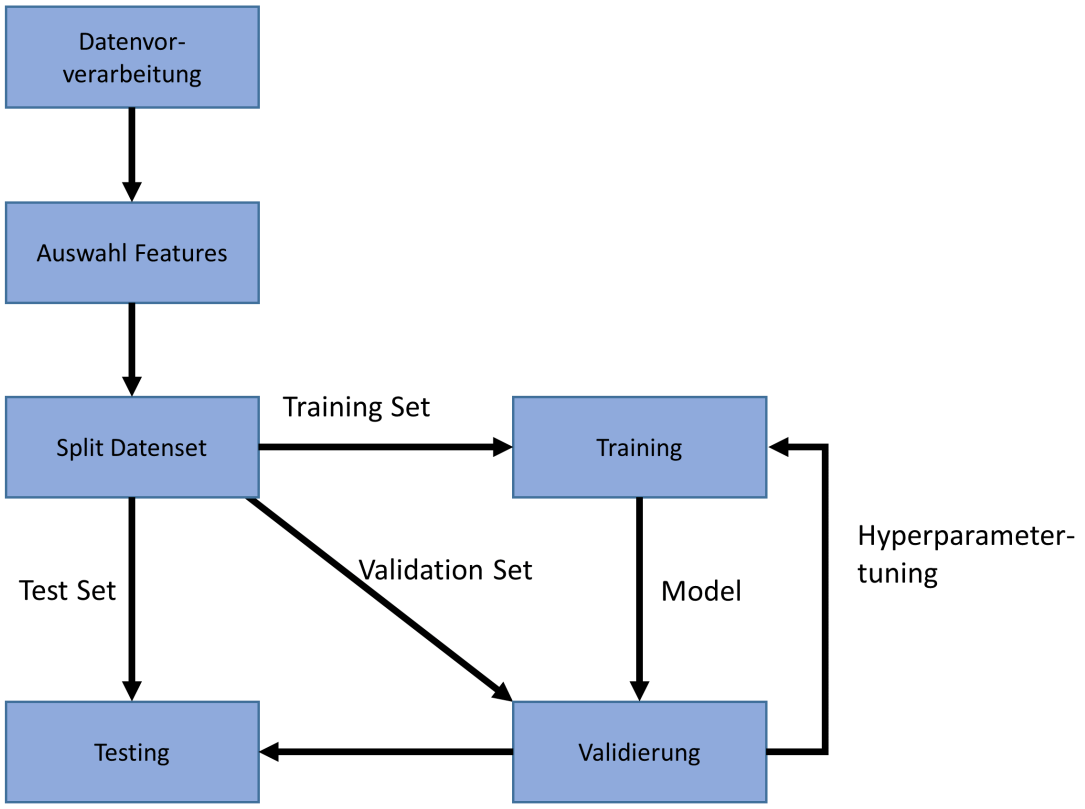


Abbildung 3 Schema der Verarbeitung des Training, Validation und Test Set in der gesamten Datenverarbeitungs- und Trainingspipeline

Error (MAE)) als beschreibende Metrik angegeben:

$$MSE = \frac{1}{N} * \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{2.14}$$

$$MAE = \frac{1}{N} * \sum_{i=1}^{n} |Y_i - \hat{Y}_i| \tag{2.15}$$

Um die Spannungsstabilität im Bordnetz sicherzustellen, ist es wichtig die Extremwerte vorherzusagen, damit dementsprechend Maßnahmen eingeleitet werden können. Da ein Modell, welches einen guten MSE oder MAE aufweist, nicht zwingend auch die Extremwerte gut prädiziert, wird der maximale absolute Fehler (engl. Max Absolute Error (MAXE)) als zusätzliche maßgebliche Metrik im Testdatensatz eingeführt.

$$MAXE = \max_{i \in [1,n]} (Y_i - \hat{Y}_i) \tag{2.16}$$

Für eine besser Vergleichbarkeit von Modellierungsansätzen wird der MAE oft auf die Nennspannung U_{nenn} referenziert. Der mittlere absolute prozentuale Fehler (engl. Mean

Absolute Percentage Error (MAPE)) berechnet sich zu:

$$MAPE = \frac{MAE}{U_{nenn}} \tag{2.17}$$

2.3.2. Deep Learning

Werden Modelle mit mehreren Schichten trainiert, spricht man vom Deep Learning. Der ersten Schicht wird der Input übergeben. In den tieferen Schichten werden abstrakte Muster trainiert, die dann in der letzten Schicht, dem Output, resultieren.

Es existieren verschiedene Netzwerktypen, die je nach den zu modellierenden Gegebenheiten angewandt werden.

- **Multi Layer Perceptron:** Ein Multi Layer Perceptron (MLP) ist ein einfaches Feed Forward Neural Network, welches aus drei oder mehr Schichten besteht. Alle Neuronen sind mit jedem Neuron der vorhergegangen bzw. nachfolgenden Schicht vernetzt. Daher ist es ein vollvernetztes neuronales Netzwerk. Die Neuronen, außer die der Input Schicht, werden über eine nichtlineare Funktion aktiviert. Dabei wird meist auf $tanh$ oder $sigmoid$ Funktionen zurückgegriffen. Durch diese Aktivierungsfunktionen können MLP sämtliche Funktionen approximieren [93].
- **Convolutional Neural Network:** Convolutional Neural Network (CNN) sind eine der bekanntesten Netzwerktypen, basierend auf der mathematischen Operation, der Faltung (Convolution) von Matrizen. Neben den Faltungsschichten werden in einem CNN auch Pooling-Schichten und Vollvernetzte Schichten genutzt. Das Pooling soll die Komplexität für die folgenden Schichten reduzieren und dabei helfen die dominanten Merkmale zu extrahieren. Die Performance ist vor allem in Bild- und Sprachverarbeitungsanwendungen sehr gut [94].
- **Generative Adversarial Networks:** Bei Generative Adversarial Network (GAN) werden zwei Modelle simultan trainiert, das generative Modell (GM) und das diskriminative Modell (DM). Das GM lernt die Verteilung der Input Daten und erzeugt synthetische Daten im Raum der Input Daten. Das DM klassifiziert die ausgegebenen Punkte vom GM in die Kategorien „real“ oder „synthetisch“. Während dem Training konkurrieren die beiden Modelle miteinander, wobei beide versuchen den Fehler zu minimieren. Dadurch wird das GM darauf trainiert Punkte zu erzeugen, die von den Input Daten nicht zu unterscheiden sind [95, 96]. Das DM soll die synthetischen Punkte erkennen. Dieser Netzwerktyp wurde bereits erfolgreich in Applikationen wie Style Transfer, Bildeinfärbung und bei der Erzeugung neuer Daten eingesetzt [93].
- **Recurrent Neural Networks:** RNNs sind künstliche neuronale Netze, in denen nicht nur die Schichten untereinander verbunden sind, sondern auch Verbindungen zwischen Neuronen derselben oder vorhergegangenen Schicht existieren. Dies gibt dem

Netz die Möglichkeit persistent Informationen zu speichern. Somit können sequenzielle Daten eingelesen und im Netz verarbeitet werden [97].

Im Hinblick auf die in dieser Arbeit betrachtete Zeitreihenprognose, wird im nächsten Kapitel genauer auf den Aufbau von RNN sowie deren Sonderformen eingegangen.

2.3.3. Recurrent Neural Networks

In diesem Kapitel wird zunächst auf die Funktionsweise der RNN eingegangen sowie anschließend die häufig verwendeten GRU, LSTM, Dropout Layern und Dense Layern erläutert. Abbildung 4 zeigt den Aufbau von einem Neuron in einem RNN. Jedes Neuron besteht aus mehreren RNN Zellen, welche die Inputs x_{0-t} zu einem Zellzustand C und dem Ausgang h weiterverarbeiten. Der Input wird in Form von sequenzierten Daten gespeist. Der Output der letzten Schicht ist der letzte Wert des Ausgangs h_t. Der Ausgang eines Neurons wird, wie bei einem FFNN, mit w gewichtet und an das nächste Neuron als Output o_t übergeben. Die sequenzierten Inputs entsprechen in der Regel Ausschnitten aus der zeitlichen Reihenfolge der Daten. Die Sequenzlänge t gibt vor, wie viele RNN Zellen pro Neuron vorhanden sind. Das RNN lernt anhand der Sequenzen das zeitliche Verhalten der Daten und schätzt bei der Eingabe einer neuen Sequenz den letzten Wert.

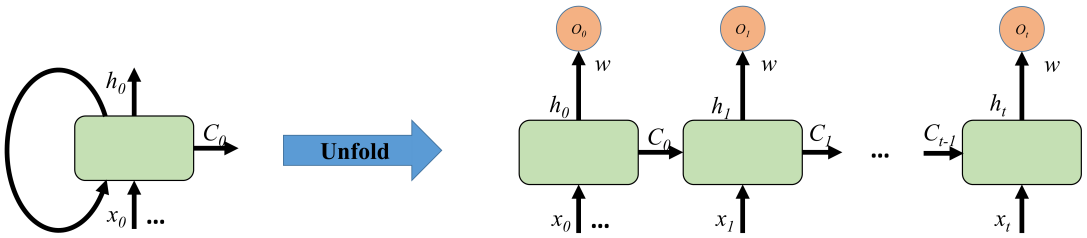


Abbildung 4 Aufbau eines aufgeschlüsselten RNN Neurons

Backpropagation (BP) ist das Verfahren, welches angewandt wird, um das Netz zu trainieren. Die gelabelten Daten werden zuerst durch das Netz propagiert und anschließend das Ergebnis mit dem Label verglichen. Der entstandene Fehler hat Einfluss auf die Anpassung der Gewichte bei der Backpropagation. Je nach Größe des Fehlers und Einfluss des einzelnen Neurons auf das Ergebnis wird das Gewicht der Neuronenverbindung aktualisiert. Bei RNN wird ein Backpropagation Through Time Algorithmus (BPTT) angewendet. Die Zeitreihen werden während der BP aufgefalten und die Fehlerberechnung findet in jedem Zeitschritt statt. Die berechneten Fehler werden addiert und für das Gradientenabstiegsverfahren verwendet [98]. Der Aufwand einen einzelnen Parameter bei BPTT zu aktualisieren ist hoch, daher wird oft der Algorithmus der Truncated Backpropagation Through Time (TBPTT) verwendet. Bei TBPTT werden die Sequenzen in *Truncs* unterteilt, die dann einzeln berechnet werden. Die Länge der *Truncs* gibt die maximale Zeitabhängigkeit vor [99].

Funktionsweise Recurrent Neural Network Zellen

Der Aufbau einer einzelnen RNN Zelle ist in Abbildung 5 dargestellt. Der Input x_t wird zusammen mit h_{t-1} über die Aktivierungsfunktion $tanh$ zum Output h_t verarbeitet.

$$y_t = \sigma(W^{hy} h_y + c) \tag{2.18}$$

$$h_t = \sigma(W^{xh} x_t + W^{hh} h_{t-1} + b) \tag{2.19}$$

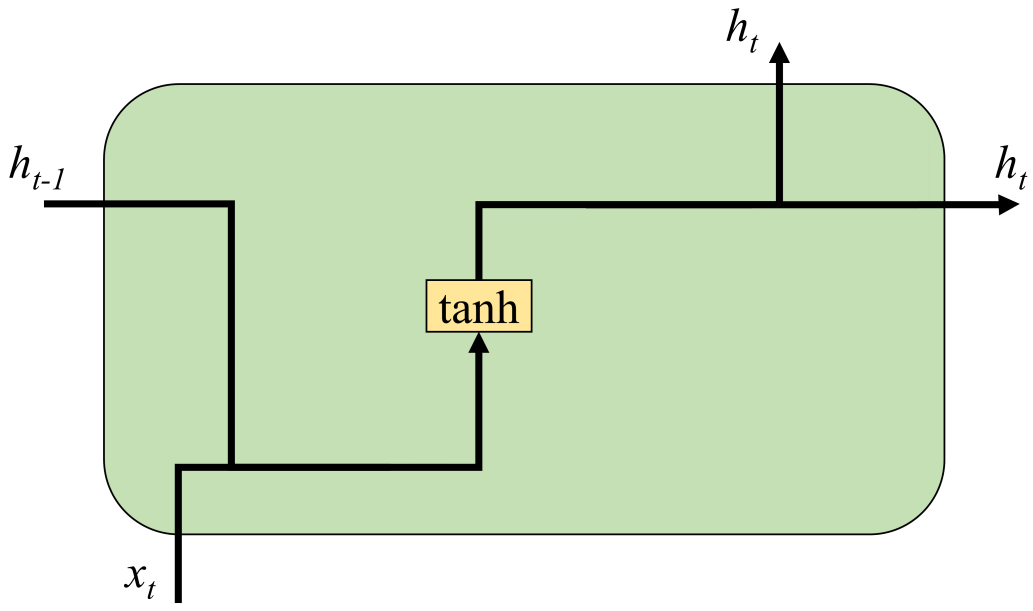


Abbildung 5 Schematische Struktur einer RNN Zelle

RNN Zellen können nur unzureichend Langzeitabhängigkeiten nachbilden. Der Butterfly Effekt der Chaos Theorie [100] beschreibt die sensitive Abhängigkeit der initialen Bedingungen. So können kleine Änderungen am Eingang eine große Veränderung am Ausgang des Netzes verursachen [101]. Zusätzlich führt das vanishing gradients Problem zu einem verschwimmen der Gradienten innerhalb der Zeitabhängigkeit. Dies haben zuerst Hochreiter [102] und Bengio [103] beschrieben. Während der BPTT werden die Gradienten wegen der Kettenregel mit rekurrenten Matrix Multiplikationen berechnet. Sind die Gradienten kleiner als eins, tendieren diese exponentiell gegen 0 zu konvergieren [104]. Die Lösung dieser Problematik wird in den nächsten Sektionen beschrieben. Sind die Gradienten in der BPTT größer als eins, explodieren die Werte. Das damit einhergehende Problem ist das exploding gradients Problem, dem mit Hilfe von Gradient clipping, einem Verfahren, bei welchem die Gradienten begrenzt werden, entgegengewirkt werden kann [104].

Long Short-Term Memory Zellen

LSTM Zellen wurden initial von Hochreiter et al. [105] in ihrer Thesis eingeführt. In diesem Ansatz wird das vanishing gradient Problem damit gelöst, dass ein zusätzlicher Status zu jeder Zelle hinzugefügt wird. Abbildung 6 zeigt die Struktur einer LSTM Zelle. Der versteckte Zustandsvektor h_t ist stark vom Zellzustand C_t abhängig. Der Zellzustand ermöglicht es dem Netzwerk Informationen über einen längeren Zeitraum zu speichern, ohne auf das vanishing gradient Problem zu stoßen. Das Forgetgate f_t, das Inputgate i_t und der vorherige Zellzustand C_{t-1} haben direkte Auswirkungen auf den Zellzustand C_t. Das Forgetgate wird mit dem vorherigen versteckten Zustand h_{t-1}, dem Inputvektor und einer Aktivierungsfunktion berechnet. Der Kandidat für das Aktualisierungsgate C_t hat die gleichen Inputs, wird aber mit einer $tanh$ Funktion aktiviert. Das Inputgate i_t und das Outputgate o_t werden ähnlich wie das Forgetgate berechnet. W und U beschreiben die entsprechenden Matrizen, b ist der Vorspannungsvektor und σ ist die Aktivierungsfunktion.

$$f_t = \sigma_{sig}(W_f x_t + U_f h_{t-1} + b_f) \tag{2.20}$$

$$i_t = \sigma_{sig}(W_i x_t + U_i h_{t-1} + b_i) \tag{2.21}$$

$$o_t = \sigma_{sig}(W_o x_t + U_o h_{t-1} + b_o) \tag{2.22}$$

$$\widetilde{C_t} = \sigma_{tanh}(W_C x_t + U_C h_{t-1} + b_c) \tag{2.23}$$

$$c_t = f_t c_{t-1} + i_t \widetilde{C_t} \tag{2.24}$$

$$h_t = o_t \sigma_{tanh}(c_t) \tag{2.25}$$

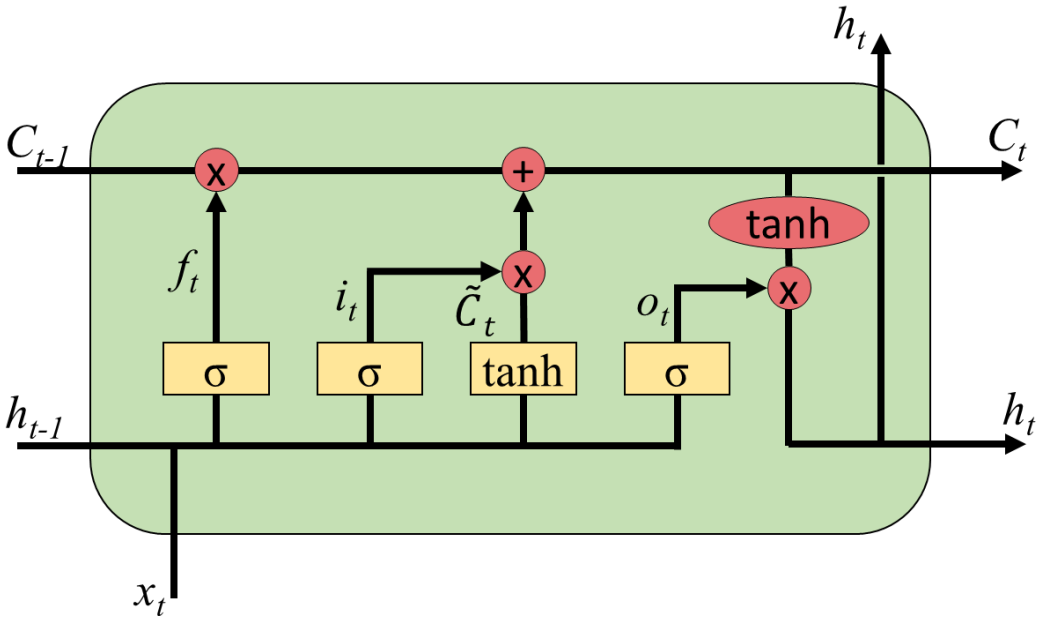


Abbildung 6 Schematische Struktur einer LSTM Zelle

Gated Recurrent Units

GRUs wurden zuerst von Cho et al. [106] vorgeschlagen. Die Struktur einer GRU Zelle ist ähnlich der einer LSTM Zelle, jedoch beinhaltet sie ein Outputgate weniger. Dadurch können die Matrix Berechnungen während dem Fitting und dem BPTT schneller durchgeführt werden als beim LSTM. Typischerweise sind die Trainingszeiten pro Epoche kürzer, jedoch ist die Genauigkeit geringer. Der versteckte Zustandsvektor h_t ist stark abhängig vom Aktualisierungsgate z_t, dem vorherigen versteckten Zustandsvektor h_{t-1} und dem Kandidaten für den versteckten Zustandsvektor $\widetilde{h_t}$. Der Kandidat für den versteckten Zustandsvektor $\widetilde{h_t}$ wird mit einer $tanh$ Funktion aktiviert und zusätzlich mit dem Resetgatevektor r_t multipliziert. Der Resetgatevektor r_t und der Aktualisierungsgatevektor z_t werden mit dem vorherigen versteckten Zustand h_{t-1} und dem Inputvektor x_t berechnet. W und U beschreiben die entsprechenden Matrizen, b ist der Biasvektor und s ist die Aktivierungsfunktion.

$$z_t = \sigma_{sig}(W_z x_t + U_z h_{t-1} + b_z) \tag{2.26}$$

$$r_t = \sigma_{sig}(W_r x_t + U_r h_{t-1} + b_r) \tag{2.27}$$

$$\widetilde{h_t} = \sigma_{tanh}(W_h x_t + U_h(r_t h_{t-1}) + b_h) \tag{2.28}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\widetilde{h_t} \tag{2.29}$$

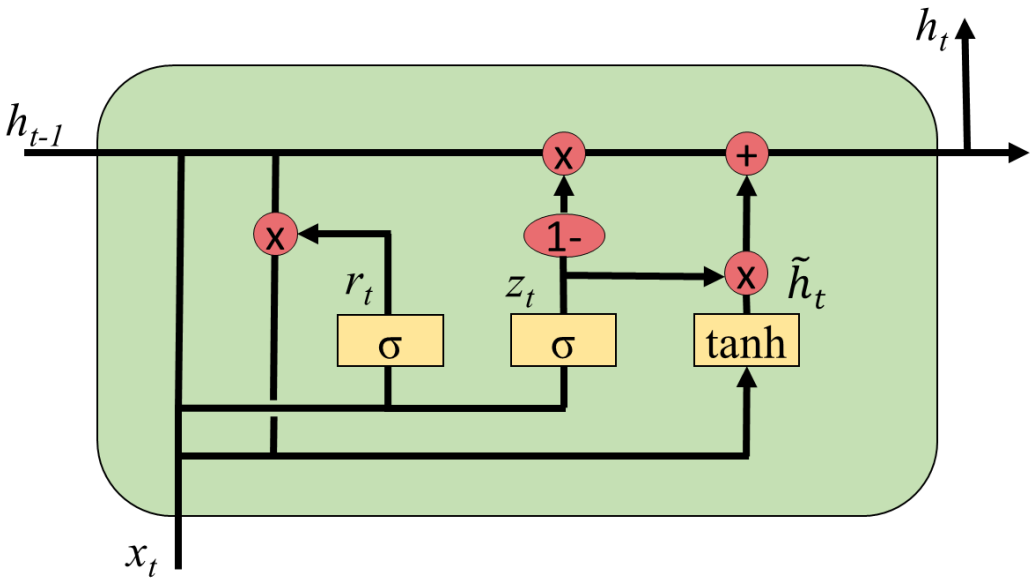


Abbildung 7 Schematische Struktur einer GRU Zelle

Dense Layer

Dense Layer sind vollvernetzte FFNN Layers, die oft genutzt werden um lineare Probleme zu fitten [107]. Wird eine nichtlineare Aktivierungsfunktion, wie beispielsweise $tanh$ [108], genutzt, kann auch nichtlineares Verhalten gefitted werden. Nutzt man diese Schichten im Anschluss an GRUs oder LSTMs, werden die abstrakten Outputs der RNNs weiterverarbeitet. Es wurde gezeigt, dass ein bis drei Dense Layer nach den RNN Schichten die Performance der Netze verbessern [109, 108].

Dropout

Das Trainieren von neuronalen Netzen hat das Ziel, die Input-Daten möglichst gut zu generalisieren. Wie in den ersten 20 Epochen der Abbildung 8 (b) zu sehen ist, kommt es am Anfang eines Trainings in der Regel zu einem Underfitting, bei dem die Datenpunkte nur unzureichend genau geschätzt werden. Im weiteren Verlauf des Trainings sollten sowohl der Loss als auch der Validation Loss weiter abnehmen, bis eine Art Sättigung erreicht ist. In Teil (a) ist ein typischer Verlauf von Overfitting mit dem charakteristischen Anstieg des Validation Loss im Verlauf des Trainings zu sehen. Das Trainingsdatenset wird dabei so genau nachgeahmt, dass eine Generalisierung, wie sie in der Validierung geprüft wird, nicht auftritt. Diese Art der trivialen Abbildung der Eingangsdaten auf die Ausgangsdaten ist begünstigt durch geringe Datenmengen und großen NN mit vielen hidden Layern und vielen hidden Units. Eine pauschale Aussage darüber, ob Overfitting auftreten wird oder nicht, kann nicht gemacht werden. Daher wird dem Effekt mit Dropouts präventiv entgegengewirkt.

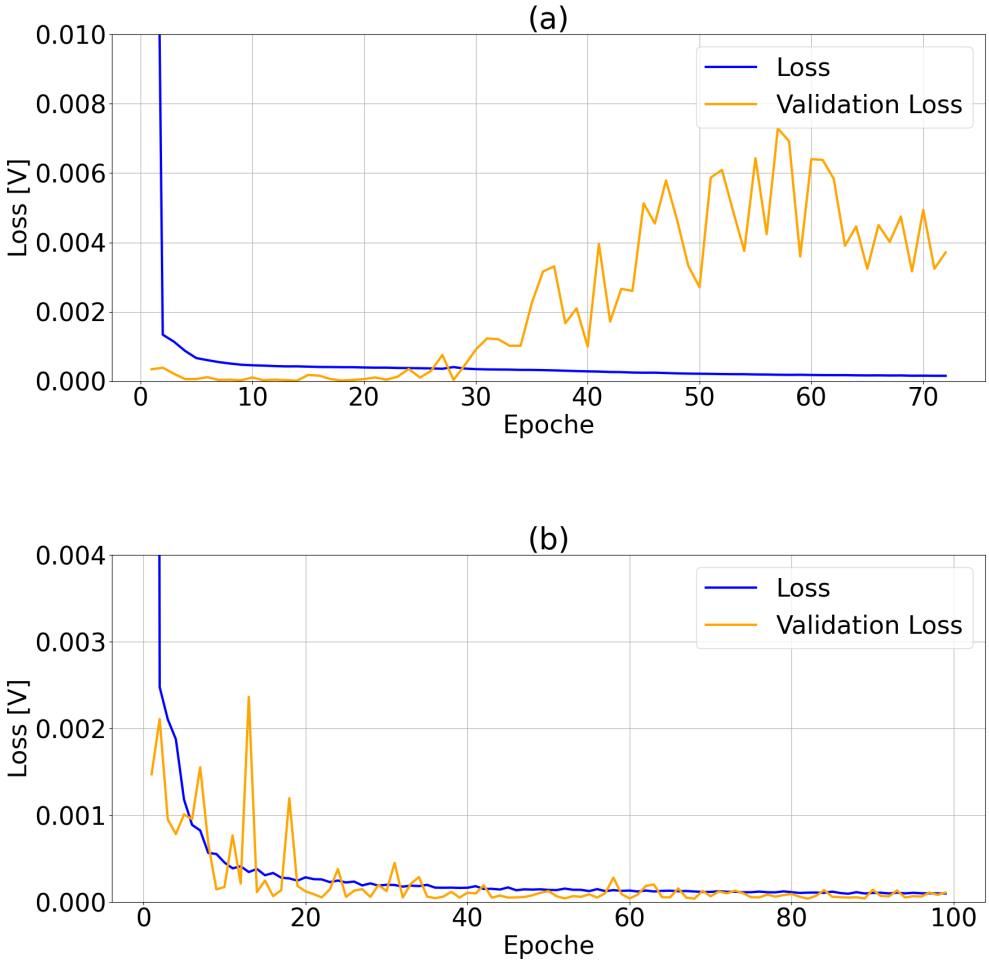
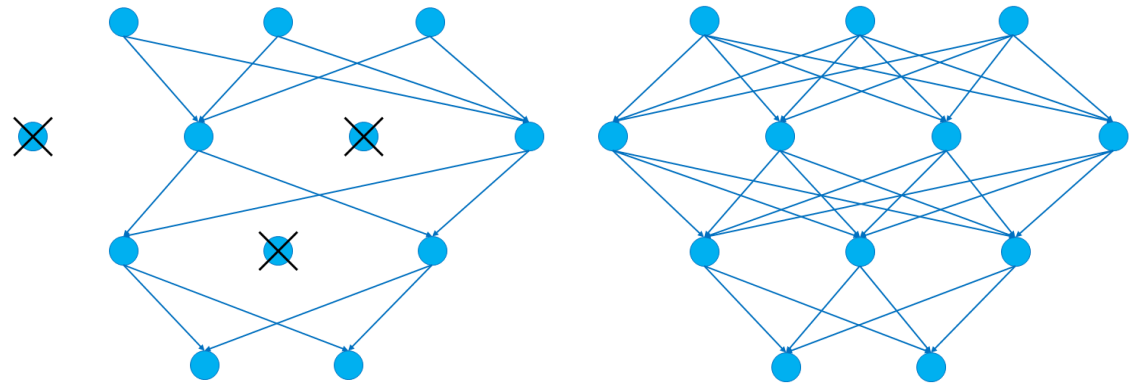


Abbildung 8 Verhalten von Loss und Validation Loss bei Overfitting (a) und keinem Overfitting (b)

Srivastava et al. haben 2014 [110] Dropouts zur Vermeidung von Overfitting eingeführt. Dropout Layer können nach jedem hidden Layer hinzugefügt werden, um die Robustheit der jeweiligen und vorhergegangenen Schichten zu verbessern. Abbildung 9 zeigt das Prinzip hinter Dropout. Es werden zufällig Neuronen ausgesucht, welche während des Trainings nicht beachtet werden. Die Dropout Rate gibt an, wie viele Neuronen in jeder Iteration anteilig ausgelassen werden. Eine Konsequenz ist hier, dass das Netz weniger sensitiv auf kleine Änderungen am Eingang reagiert. Wird die Gefahr der Entstehung eines Overfittings reduziert, ist es möglich mit höheren Lernraten zu trainieren und damit den Lernprozess zu beschleunigen.



(a) Mit Dropout (b) Ohne Dropout

Abbildung 9 Beispielhaftes neuronales Netz mit (a) und ohne (b) Dropout

2.3.4. Transformer

Die von Vaswani et. al. [111] vorgestellten Transformer zeigen im Bereich der Computerlinguistik (NLP) deutlich bessere Ergebnisse als LSTM oder andere RNN. Zum Beispiel wurden mit dem vortrainierten BERT [112] in Anwendungen des Sprachverständnisses (engl. Language Understanding, LU) LSTMs deutlich übertroffen. Der Transformer von Radfar et al. [113] aus dem Bereich der spoken language understanding kann Audio Sequenzen interpretieren. Außerhalb der Sprach- und Bildverarbeitung wird in der Literatur keine weitere Anwendung diskutiert. Die Fähigkeit der Transformer, Sequenzen zu verarbeiten und Regressionsprobleme zu lösen, lässt auf eine mögliche Anwendung in der Spannungsprädiktion bei Batterien schließen.

Transformer haben, wie die meisten Modelle der neuronalen Sequenztransduktion, eine Encoder-Decoder Struktur. Im Gegensatz zu RNNs greifen Transformer nicht auf das Prinzip der rekurrenten Strukturen zurück, sondern bauen auf dem Attention Mechanismus auf.

Wie in Abbildung 10 zu sehen ist, sind Transformer in einer Encoder-Decoder Struktur aufgebaut. Da Transformer keine Rekurrenz enthalten, wird mithilfe von positional Encoding die Information der Reihenfolge des Inputs dem Encoder-Decoder übergeben. Eine Encoder Schicht verarbeitet die Inputs anhand eines Multi-Head Attention Moduls und einem vollvernetztem FFN zu einer Repräsentation des Eingangs. Der Aufbau der Decoder Schicht ist ähnlich dem des Encoders, sieht aber zusätzlich ein Multi-Head Attention Modul für die Verarbeitung des Inputs aus dem Encoder vor. Der Ausgang des Decoders wird mit einer Softmax Funktion zu Ausgabe Wahrscheinlichkeiten aktiviert. Die Encoder und Decoder werden aus N-fach gestapelten Encoder-Decoder Schichten gebildet.

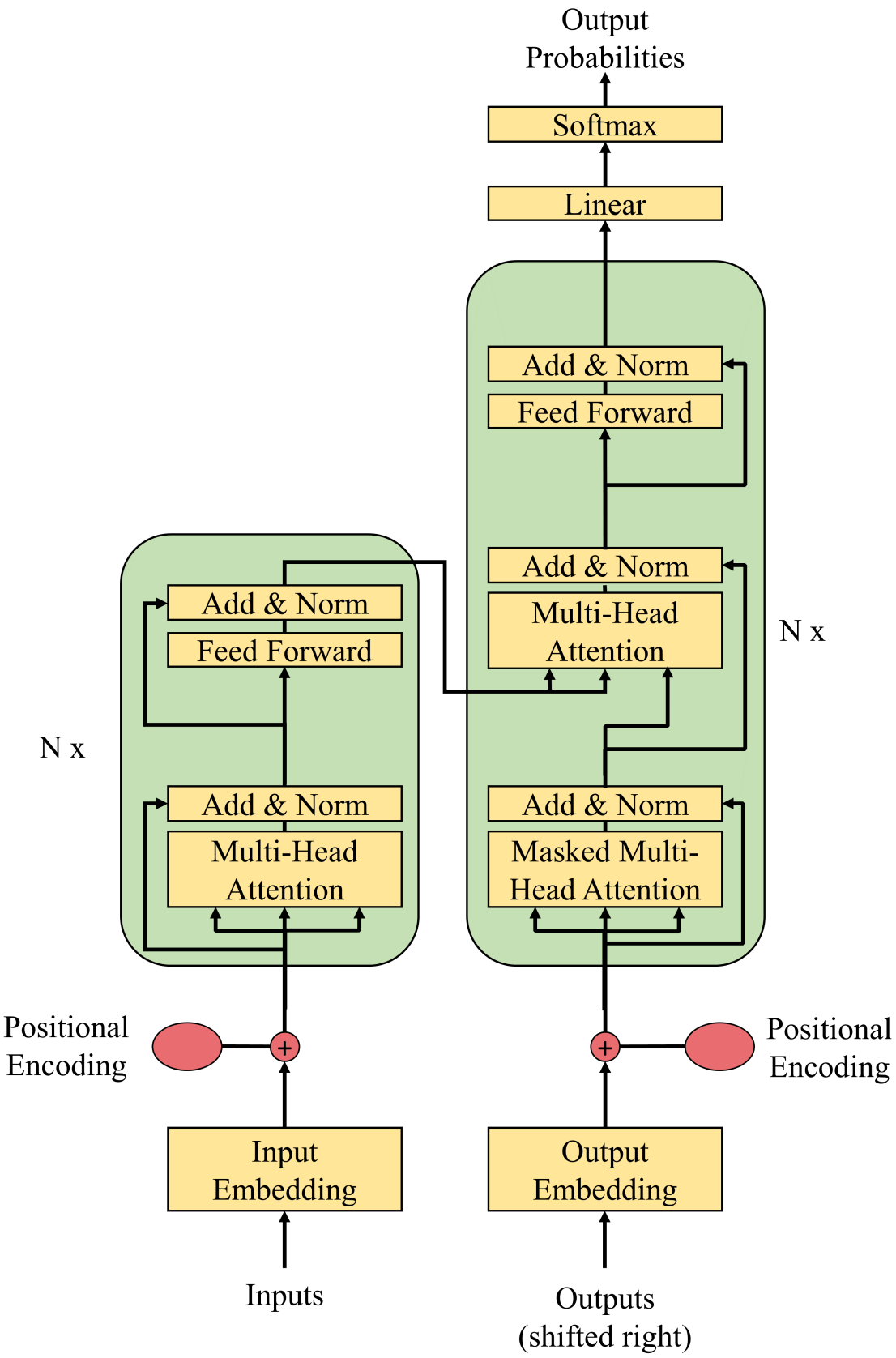


Abbildung 10 Modellarchitektur eines Transformers

Die in der Encoder-Decoder Struktur verwendeten Multi-Head Attention Module bestehen aus mehreren Scaled Dot-Product Attention Mechanismen, die jeweils aus den Queries q, Keys k und Values v ihre eigenen Projektionen berechnen. Wie in Abbildung 11 gezeigt, werden die parallel ausgeführten Projektionen zu einem Ausgang verkettet.

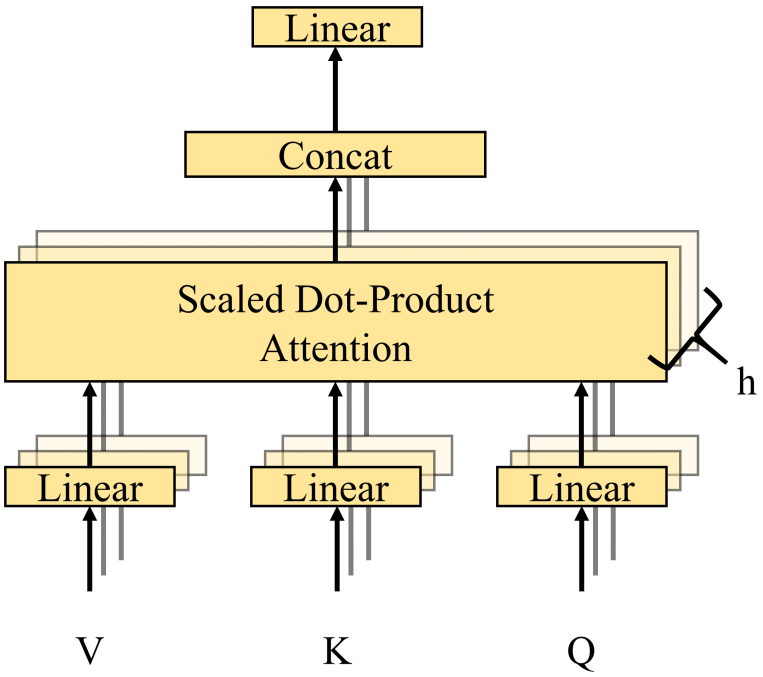


Abbildung 11 Multi-Head Attention Modul

Der Scaled Dot-Product Attention Mechanismus aus Abbildung 12, bildet ein Skalarprodukt aus den normierten Querys q und Keys k, das dann mit einer Softmax Funktion die Gewichte der Values v aktiviert.

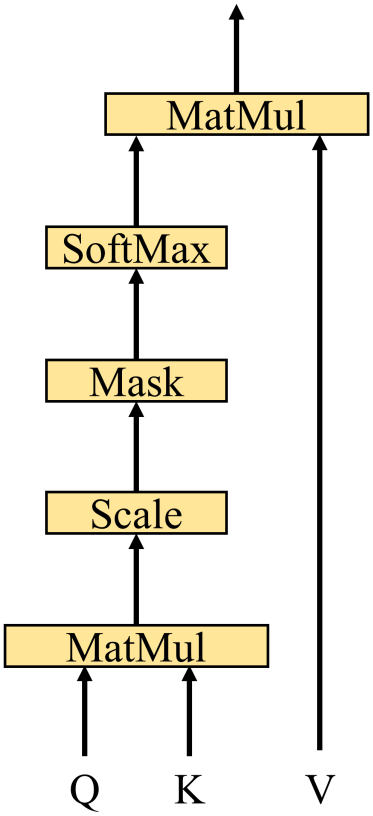


Abbildung 12 Scaled Dot-Product Attention Mechanismus

Durch die Attention Mechanismen entstehen kurze Pfade zwischen langen Zeitabhängigkeiten innerhalb des Netzwerks. Je kürzer diese Pfade sind, desto einfacher kann ein Modell einen Zusammenhang zwischen zwei Punkten erkennen.

Wie in Kapitel 2.3.3 beschrieben, werden bei Transformern Dropout Layer mit der Dropout Rate dr eingefügt, um die Gefahr von Overfitting zu reduzieren. Zusätzlich zum Dropout wird nach jedem Sub-Layer eine Normalisierungsschicht integriert [111].

2.3.5. Hyperparameter

Die Güte von NN ist nicht nur abhängig von den Daten, die in das Netz gespeist werden, sondern auch von den Hyperparametern, die zu Beginn des Trainings festgelegt werden. Hyperparameter sind von den Modellparametern zu unterscheiden. Modellparameter, wie die Gewichte und der Bias, werden während dem Training automatisch vom Algorithmus angepasst. Das Hyperparametertuning findet manuell, meist mit Hilfe von Such- und Optimierungsalgorithmen statt.

Trainingsbasierte Hyperparameter

Die folgenden Hyperparameter beziehen sich auf das Training des Modells und müssen im Voraus festgelegt werden:

- **Sequenzlänge:** Die Sequenzlänge hat Einfluss auf die Anzahl der Sequenzen, die zeitliche Abhängigkeit der Modellierung und die Performance des Trainings. Durch die obligatorische zeitliche Aneinanderreihung der Messpunkte ergeben sich bei größeren Sequenzlängen weniger Sequenzen. Dieser Einfluss ist zu beachten, wenn nur wenige Rohdatenpunkte vorliegen, die viele zeitliche Unterbrechungen enthalten. Für die zeitliche Abhängigkeit der Modellierung sind zum einen die zugrundeliegenden physikochemischen Effekte zu berücksichtigen und zum anderen die typischen Belastungsszenarien in der jeweiligen Applikation. Bei schnellen, volatilen Belastungsänderungen sind kürzere Sequenzlängen ausreichend. Um längere Effekte in der Batterie abzubilden, wird die Sequenzlänge größer gefasst. Die Anzahl der RNN Zellen je Neuron ist gleich der Sequenzlänge, somit steigt der Rechenaufwand mit steigender Sequenzlänge.
- **Verlustfunktion:** Mit der Backpropagation wird versucht die Verlustfunktion zu minimieren. Die Verlustfunktion berechnet sich aus der Schätzung des Netzes $\hat{Y}_i$ und der GT Y_i. Oft wird bei Regressionsproblemen der MSE verwendet [109].
- **Lernrate:** Die Lernrate η beeinflusst maßgeblich die Geschwindigkeit sowie die Genauigkeit des Optimierungsverfahrens, da sie die Änderung der Gewichte und der Bias während des Lernprozesses definiert. Lernraten werden häufig im Bereich zwischen 0,01 und 0,9 verwendet. Grundsätzlich hängt die Wahl der richtigen Lernrate immer vom zu lösenden Problem, dem verwendeten Netz und den Trainingsdaten

ab. Wird die Lernrate zu groß gewählt, werden die Gewichte schneller und in größeren Schritten geändert. Dies kann zur Folge haben, dass ein Minimum übersprungen wird. Eine kleine Lernrate führt dazu, dass die Verlustfunktion nur langsam gegen das Minimum konvergiert, was lange Trainingslaufzeiten zur Folge hat. Der Anwender muss einen Kompromiss aus Trainingslaufzeit und Genauigkeit finden. Um diesem Problem entgegenzuwirken, können variable Lernraten implementiert werden. Variable Lernraten nehmen stufenweise ab. Anfangs werden Sie größer gewählt, um sich schneller an das globale Minimum heranzutasten und fallen während der Trainingslaufzeit ab, um eine Konvergenz in kleinen Schritten gegen das Minimum sicherzustellen [114].

- **Optimierer:** Die Optimierung der Gewichte wird durch das Gradientenabstiegsverfahren (GD) realisiert. Häufig werden aber modifizierte GD genutzt, um die Effizienz des Optimierungsverfahrens zu erhöhen. Sie unterscheiden sich in der Anzahl der Daten, die verwendet werden, um den Gradienten zu berechnen. Auch hier muss ein Kompromiss zwischen Genauigkeit und Lernzeit getroffen werden [115]. In der Literatur wird der Optimierer Adaptive Moment Estimation (Adam) im Vergleich zu AdaGrad, RMSprop und anderen GD Algorithmen als Optimierer für Regressionsprobleme empfohlen [116]. Adam ist ein Algorithmus zur gradientenbasierten Optimierung stochastischer Zielfunktionen erster Ordnung, basierend auf adaptiven Schätzungen von Momenten niedrigerer Ordnung [117].

- **Batchgröße:** Mit Stochastic Gradient Descent (SGD) wird die Backpropagation für jeden einzelnen Datenpunkt aus dem Training Set durchgeführt. Im Kontrast dazu steht der Batch Gradient Descent (BGD), bei dem nur einmal pro Epoche das Updaten der Modellparameter stattfindet. SGD ist sehr rechen- und daher zeitaufwändig, wohingegen mit BGD die Gefahr besteht, dass weniger erfolgreich generalisiert werden kann und somit die Modelle ungenauer werden [118]. Um die beiden Nachteile der Verfahren zu minimieren, wurde das sogenannte Mini-Batch stochastic gradient descent Verfahren eingeführt. Hier wird in einem Schritt eine Teilmenge (Batch) des Datensatzes verarbeitet und pro Batch einmal die BP durchgeführt. Bei Regressionsmodellen wurde festgestellt, dass mit einem Mini-Batch SGD keine Generalisierungslücke entsteht [119]. Mini-Batch SGD werden in der Praxis häufig angewendet [120]. Die Batchgröße gibt an, wie viele Datenpunkte pro Batch gebündelt werden.

- **Batchnormalisierung:** Die von Ioffe et al. [121] eingeführte Batchnormalisierung dient der Limitierung der kovariablen Verschiebung dadurch, dass jede Schicht des NN normalisiert wird. Eine Normalisierung der Eingangsdaten macht eine Batchnormalisierung nicht obsolet.

- **Epochen:** Eine Trainingsepoche ist abgeschlossen, wenn jeder Eingangsdatenpunkt für das Training einmal verwendet wurde. Je mehr Epochen trainiert werden, desto besser kann das Netz die Input Daten abbilden. Ein zu langes Training kann dazu führen, dass Overfitting entsteht, oder das Netz in eine Art Sättigung läuft und nicht mehr dazulernt. An diesem Punkt kann das Training unterbrochen werden.

Modellbezogene Hyperparameter RNN

Bei einem RNN gibt es hauptsächlich die folgenden drei zu tunenden modellbezogenen Hyperparameter:

- **Anzahl Schichten:** Die Anzahl der Schichten in einem Deep Neural Network hängt nicht nur von der Anzahl der Inputs und Outputs ab, sondern auch von der Größe des Training Set und dessen Komplexität. Durch die richtige Wahl der Anzahl der Schichten können die Ergebnisse verbessert und eine geringere Trainingszeit erreicht werden. Bis zu drei hidden Layers können die Genauigkeit eines Netzes erhöhen, jedoch wird die Trainingszeit vielfach erhöht [122].
- **Anzahl Neuronen:** Die Anzahl der Neuronen pro Schicht ist in Abhängigkeit von der Anzahl der Schichten zu dimensionieren. Je mehr Neuronen pro Schicht vorhanden sind, desto komplexer ist das Netz. Das Training dauert länger und die Tendenz zum Overfitting steigt [122, 123].
- **Dropout Rate:** Wie in Kapitel 2.3.3 beschrieben gibt die Dropout Rate an, welcher Anteil der Neuronen während dem Training auf zufälliger Basis ignoriert wird. Mit einer erhöhten Dropout Rate kann man bei großen Netzen das Risiko eines Overfittings reduzieren.

Modellbezogene Hyperparameter Transformer

Für Transformer existieren spezielle modellbezogene Hyperparameter, die sich von denen der RNNs unterscheiden:

- **Anzahl der Stacks:** Die Anzahl der Stacks N gibt an, wie viele Stacks aus Multi-Head Attention Modulen und FFNN pro Encoder-Decoder verwendet werden.
- **Länge des Input Vektors:** Der Input Vektor der Länge d_{model} wird mit den durch das Training erlernten Query-, Key- und Value-Matrizen zu den entsprechenden Vektoren multipliziert.
- **Größe von Query, Key und Value:** Die Größe der Vektoren von Query q und Key k sind ausschlaggebend dafür, welche Range das Skalarprodukt betrachtet, um die Value Vektoren v zu gewichten. Die Größe dieser Vektoren ist abhängig von d_{model} und der Anzahl der Attention Heads h.

- **Anzahl der Attention Heads:** Die Anzahl der Heads h sagt aus, wie viele Attention Heads pro Attention Layer parallel berechnet werden.
- **Dropout Rate:** Die Dropout Rate dr gibt an, wie in Kapitel 2.3.3 beschrieben, welcher Anteil der Neuronen während dem Training auf zufälliger Basis ignoriert wird.
- **Positional Encoding Length:** Die positional Encoding length pe_{period} gibt an, welcher Zeitraum beim positional Encoding betrachtet wird. Der Betrachtungszeitraum ist analog zur Sequenzlänge der RNNs zu sehen.

Tuning

Das Hyperparametertuning der hier erklärten Hyperparameter wird durch einen oder eine Kombination der folgenden Algorithmen durchgeführt. Dabei ist darauf zu achten, dass sich einige Hyperparameter gegenseitig beeinflussen, jedoch nicht alle Hyperparameter gleichzeitig in einem Durchgang optimiert werden können. Der Parameterraum wächst je Parameter um eine weitere Dimension. Eine zeitgleiche Optimierung aller Parameter würde zu viel Rechenzeit beanspruchen. Daher werden Hyperparameter, welche wenig Einfluss auf andere haben, primär festgelegt.

- **Rastersuche:** Die Rastersuche ist eine erschöpfende Suche, die in einem definierten Parameterraum alle möglichen Kombinationen testet. Für kontinuierliche Parameter werden Schrittweiten und Grenzen definiert, um sie in eine diskrete endliche Form zu transferieren. Die Optimierung vieler Parameter stellt ein Problem für die Rastersuche dar, da die Anzahl der zu untersuchenden Punkte um die Potenz der Dimensionalität steigt. Ein Lösungsansatz ist, die Suche für unabhängige Parameter parallel laufen zu lassen. Aufgrund der Diskretisierung kann nicht sichergestellt werden, dass das Optimum erreicht wird.
- **Zufallssuche:** In der Zufallssuche werden die zu testenden Kombinationen zufällig ausgewählt. Das Verfahren kann sowohl für diskrete als auch kontinuierliche Parameter angewandt werden. Die Zufallssuche kann die Performance der Rastersuche übertreffen, vor allem bei einer kleinen Anzahl an Hyperparametern [124].
- **Bayes'sche Optimierung:** Die Bayes´sche Optimierung ist eine globale Optimierungsstrategie, die mit einer zufälligen Initialisierung beginnt. Der Algorithmus setzt eine A-priori-Wahrscheinlichkeitsverteilung auf, welche das Verhalten der Funktion erfasst. Mit den Evaluierungen wird die A-priori Funktion verwendet, um die A-posteriori-Wahrscheinlichkeitsverteilung zu wählen. Diese Wahrscheinlichkeitsverteilung erzeugt die Erfassungsfunktion, die dann den nächsten zu testenden Punkt schätzt. Damit nutzt die Bayes´sche Optimierung die Ergebnisse der getesteten Kombinationen und ist somit der Zufallssuche sowie der Rastersuche überlegen [125, 126]. Optuna ist ein open-source Hyperparameterframework, welches die Bayes´sche Optimierung nutzt. Dem Optuna Algorithmus wird eine Zielfunktion vorgegeben, die unter Berücksichtigung der definierten Parameterräume minimiert wird. Zusätzlich wird eine intelligente

Pruning-Methode bereitgestellt, die wenig vielversprechende Parameterzweige abbricht [127].

Die Hyperparametersuche für Transformer wurde von Vaswani et al. [111] als eine kaskadierte Rastersuche ausgeführt. Dabei werden die einzelnen Parameter in Kaskaden getuned. Die Ergebnisse der vorhergehenden Kaskade werden für die nachfolgenden Kaskaden als Information genutzt. Ein Nachteil ist, dass Abhängigkeiten von Parametern nicht erkannt werden, wodurch potentielle Optimierungsminima nicht gefunden werden.

3. Daten

In diesem Abschnitt wird zuerst auf die modellierten Batteriepacks, deren Aufbau und Zelleigenschaften eingegangen. Anschließend wird erläutert, wie die Daten erhoben wurden und in welchem Umfang diese vorliegen.

3.1. Batterie Samples

In dieser Arbeit wurde die Modellierungsmethodik auf zwei unterschiedliche Batteriepacks angewandt. Beide Batterien werden in 48 V MHEV Bordnetzen mit dem Zweck der Energiepufferung eingesetzt (vgl. Kapitel 2.2). Das jeweils integrierte zentralisierte BMS empfängt von den Sensoren die Temperatur, den Strom sowie die Spannungsdaten der Zellen und nutzt diese für die in Kapitel 2.1.5 beschriebenen Funktionen. Zur Vereinfachung und leichteren Lesbarkeit werden die beiden Batteriesysteme fortan nur noch verkürzt als LTO- und LFP-Batterie bezeichnet.

In Tabelle 1 sind die Eigenschaften der zwei Batteriepacks gegenübergestellt. Die LFP-Batterie weist zum einen eine mit 20 Ah deutlich höhere Kapazität im Vergleich zu der 11 Ah Kapazität der LTO-Batterie auf. Zum anderen ist das Spannungsniveau der LFP um ca. 2 V höher. Des Weiteren zeigt sich das Spannungsverhalten der beiden Batterien sehr unterschiedlich und die Batteriepacks werden in zwei verschiedenen MHEV Topologien betrieben. Dabei ist die Leistungsfähigkeit der Batterien auf die Anforderung bezüglich des Bordnetzes abgestimmt.

Tabelle 1 Kennwerte der Batteriesamples

	LTO	**LFP**
Zellen seriell	20	14
Zellen parallel	1	1
Anode	LTO	Graphit
Kathode	NMC	LFP
Nominalspannung	44,0 V	46,2 V
Kapazität	11 Ah	20 Ah
Energieinhalt	484 Wh	924 Wh
Obere Spannungsgrenze	52 V	52 V
Untere Spannungsgrenze	38 V	38 V
Maximaler Strom	350 A	600 A

3.2. Datengenerierung

3.2.1. CAN-Daten

Die Daten, welche zum Trainieren und Testen genutzt werden, sind aus Testfahrzeugmessungen bereitgestellt worden. Diese Fahrzeuge wurden unter kundenorientierten Bedingungen, bezogen auf die Fahrzeuggeschwindigkeit, die Umgebungstemperatur, die Fahrercharakteristiken und das Nutzungsverhalten bewegt. Die Daten beinhalten die vom BMS intern gemessenen Größen Strom, Terminalspannung und Temperatur, welche über einen CAN-Bus mit einer Abtastrate von 10 Hz verschickt und aufgezeichnet werden.

Insgesamt werden Messdaten mit einer Länge von ca. 2.600 h für die LFP und ca. 5.500 h für die LTO-Batterie verarbeitet. Der aufgezeichnete Temperaturbereich liegt dabei zwischen -23 °C und 60 °C für beide Batterien. Da diese Messdaten reine Aufzeichnungen realer Fahrprofile sind, erweisen sich die Daten für das Training eines NN in ihrer ursprünglichen Form als nicht geeignet. In den Messdaten ist ein natürlicher Datenbias vorhanden, welcher während der Datenvorverarbeitung eliminiert werden soll.

Die Rundung der Daten erfolgt bei der Temperatur auf ganze Zahlen in der Einheit °C, bei der Spannung auf 20 mV und beim Strom auf 25 mA. Ungenauigkeiten in der Messung und Unterschiede der Batterien in Bezug auf die Alterung, Produktion und Verbindungswiderstände etc. würden bei höheren Genauigkeiten zu inkonsistenten Daten führen. Insgesamt ist die Genauigkeit der Daten für diese Art der Modellierung ausreichend.

3.2.2. HiL-Prüfeinrichtung

Für das Test Set werden Messungen an einem Hardware-in-the-loop Prüfstand durchgeführt. Ein Vorteil durch diesen Typ der Messung ist, dass Umweltbedingungen, wie die Temperatur und der Ladezustand, vorkonditioniert werden können. Im ersten Schritt werden die Batterieprüflinge mit einem Paar aus parallelgeschalteten DC-Spannungsquellen und –senken elektrisch konditioniert. Zunächst wird die Batterie im CC/CV Verfahren mit einem Strom von 1 C und einem Abschaltstrom von 0,02 A geladen. Die anschließende Entladung erfolgt bei einem Strom von 1 C, kontrolliert durch ein Ah-Counting. Ist die elektrische Konditionierung abgeschlossen, werden die Batterieprüflinge in einer Klimakammer thermisch konditioniert.

Das Einmessen der Validierungsmessungen erfolgt an einem Prüfstand mit einer 9 kW starken DC-Spannungsquelle und einer 17 kW starken elektronischen Last, ansteuerbar per USB-Schnittstelle mit der Systementwicklungssoftware LabVIEW. Die Aufzeichnung der Ströme, Spannungen und Temperaturen erfolgt, wie in den Fahrzeugmessungen, anhand der vom BMS gemessenen und verschickten Werte mit einer Abtastrate von 10 Hz.

4. Datenverarbeitungspipeline

Die Rohdaten müssen zum einen in ein für das Netz kompatible Format gebracht werden. Zum anderen werden die Daten so aufbereitet, dass das Netz möglichst effizient lernen kann. Dazu werden zuerst die Input- und Output Variablen bestimmt und definiert, um anschließend die Datenvorverarbeitung durchführen zu können. Die generierten Datensets werden anschließend sowohl für das Training als auch für die Validierung genutzt. Beim Hyperparametertuning wird, je nach dem zu tunenden Parameter, eine rekursive Schleife zwischen Validierung und Undersampling bzw. Training angesetzt.

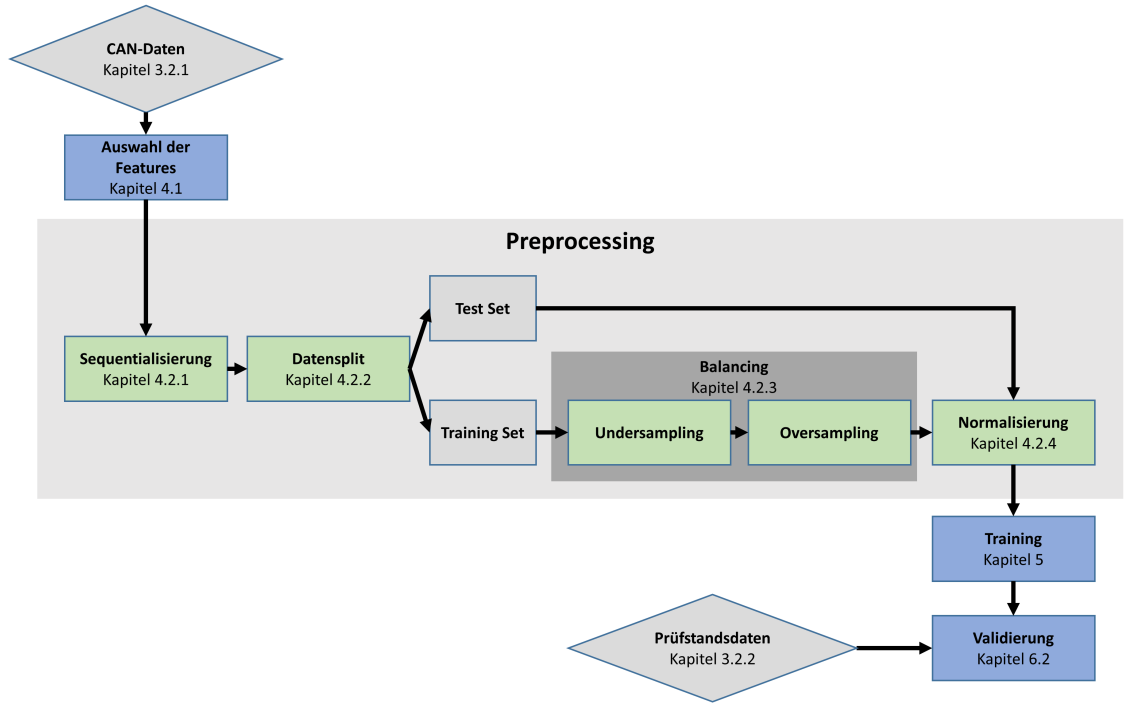


Abbildung 13 Einordnung der einzelnen Maßnahmen in die Datenverarbeitungspipeline

4.1. Auswahl der Features

In der Literatur werden Batterien auf der Basis von den Variablen Strom I, Spannung U, Temperatur T und SOC modelliert. Die drei zuerst genannten Variablen sind rein physikalisch messbare Größen, daher einfach und genau zu erzeugen. Der SOC ist eine Hilfsvariable, die selbst auf einem Modell basiert und somit durch den Modellierungsfehler eine Ungenauigkeit an das Modell weitergibt. Somit ist diese Art der Modellierung von der Genauigkeit und Verfügbarkeit eines SOC-Modells abhängig. Für Spannungsmodelle wird die direkte Information des SOC, die verfügbare Ladungsmenge, nicht benötigt. Anstatt dessen wird der SOC genutzt, um andere Variablen zu parametrieren sowie um die Ruhespannung über die OCV Kennlinie zu bestimmen. Neuronale Netze sind dazu fähig, diese Abhängigkeiten zu erkennen, ohne den SOC explizit als Input zu nutzen. Bei volatilen Stromprofilen, wie sie in der MHEV-Anwendung vorkommen, ist eine Unterscheidung zwischen Überspannung und Ruhespannung anspruchsvoll.

Um dem Netz die Information der Spannungslage zu übergeben, wird eine zusätzliche Variable eingeführt. Der Spannungstrend U_{trend} wird als Mittelwert der Spannungen in der aktuellen Sequenz berechnet. Während des Trainings wird die *teacher forcing* Methode [128] angewandt, wobei die Ground Truth, die Spannung U, von einem vorangegangen Schritt als Eingang für das Modell genutzt wird. In dieser Arbeit wird die Ground Truth indirekt über den berechneten Spannungstrend U_{trend} in das Modell eingeführt. Teacher forcing Algorithmen konvergieren im Allgemeinen schneller, bergen aber das Problem des Exposure Bias [129]. Das Problem resultiert aus dem Unterschied zwischen dem Input während des Trainings sowie dem Input für die Inferenz. Ein Fehler in der Prädiktion wirkt sich demnach direkt auf den Modell Input und somit auf die nächste Prädiktion aus, was zu Instabilitäten in der Inferenz führt. Um diesem Effekt entgegenzuwirken, wird der Spannungstrend nicht für jede Sequenz neu berechnet, sondern nur nach einer konstanten Zeit aktualisiert. Diese Zeitspanne ist länger als die Sequenzlänge zu wählen. Mit diesem Updateprozess ist das Verfahren *Scheduled Sampling* [130], bei welchem über eine Wahrscheinlichkeitsverteilung entschieden wird sowie ob die GT oder die Prädiktion verwendet wird, nicht notwendig.

Gleichung 4.1 beschreibt die Bildung des U_{mean} während der Validierung mit dem Test Set. Für die ersten 70 s wird der U_{mean} als Durchschnittswert der Spannungen in der ersten Sequenz berechnet. Nach den ersten 70 s eines jeden Test Sets ist der U_{mean} gleich seinem vorherigen Wert, mit einer Anpassung nach je 60 s an den Durchschnittswert der aktuellen Sequenz.

$$U_{mean(t)} = \begin{cases} mean(U(t=0)) & \text{wenn } t < 70\,\text{s}, \\ U_{mean}(t = t-1) & \text{wenn } t \geq 70\,\text{s}, \\ mean(U(t)_{t-128}^{t-1}) & \text{wenn } t \geq 70\,\text{s} \wedge t \mod 60 = 0. \end{cases} \tag{4.1}$$

Stehen für ein Modell viele Features zur Auswahl kann eine L1 und L2 Regulierung helfen, um die Features mit dem größten Einfluss zu bestimmen [131]. Da im Fall der Batteriemodellierung nur wenige Features zur Auswahl stehen, ist eine Reduktion der Anzahl dieser nicht notwendig.

Neben dem Spannungstrend wird auf weitere generische Features in den Trainingsdaten verzichtet. Es wird davon ausgegangen, dass das neuronale Netz wichtige Zusammenhänge zwischen den Features, wie beispielsweise der Leistung, den Kennzahlen sowie dem maximalen Strom innerhalb der Sequenz, selbst erkennt und nutzt.

4.2. Datenvorverarbeitung

Das Datenvorverarbeitung stellt sicher, dass die Rohdaten so gefiltert und transformiert werden, damit ein effizientes Training möglich ist. Die Reihenfolge der durchzuführenden Methoden ist einzuhalten.

Die Rohdaten liegen nach der Feature-Auswahl in einem Array $X_{roh} \in \mathbb{R}^{s \times n}$ mit s Datenpunkten und n Features vor.

4.2.1. Sequenzialisierung

RNN wie LSTM und GRUs basieren auf zeitlich sequenzierten Daten, die als Input in das Modell eingegeben werden. Für das Modell müssen dafür die Rohdaten in Sequenzen abgebildet werden, resultierend in einer Eingangsmatrixdimension von $X \in \mathbb{R}^{m \times n}$ mit n Features und der Sequenzlänge m.

Die Sequenzen werden aus m zeitlich aufeinanderfolgenden Punkten der Rohdaten gebildet. Wird zwischen zwei benachbarten Punkten eine zeitliche Diskrepanz festgestellt, wird die Sequenz verworfen. Der Sequenzialisierungsprozess verläuft iterativ mit einer Verschiebelänge von 1. Durch diese Verschiebelänge, treten zwar Teilsequenzen mehrfach auf, jedoch werden dadurch alle möglichen Sequenzen extrahiert. Sequenzen, die ähnliche Eigenschaften besitzen werden gegebenenfalls während dem Undersampling verworfen. Die einzelnen Sequenzen werden zu einer Matrix mit der Dimension $X \in \mathbb{R}^{m \times n \times s1}$ verkettet, wobei $s1$ die Anzahl der Sequenzen abbildet.

4.2.2. Split

Nach der Sequenzierung werden die Daten in ein Training, Validation und Test Set aufgeteilt. Die Messungen vom Prüfstand dienen als Test Set und werden zuerst separiert. Die Messdaten der Fahrzeugmessungen werden in Trainings- und Validierungsdaten aufgeteilt.

Um die Generalisierungsgüte des Modells feststellen zu können, ist es wichtig, dass der Validierungsdatensatz eine ähnliche Verteilung wie die ursprünglichen Rohdaten aufweist. Um dies zu gewährleisten, erfolgt die Auswahl der Sequenzen bei großen Datenmengen und dem damit verbundenen Gesetz der großen Zahlen für das Validierungsset zufällig.

Häufig wird ein Splitverhältnis zwischen Training und Validation Set von 80/20 verwendet, ist aber bei großen Datensätzen weniger wichtig [132]. Da während dem Under- und Oversampling Sequenzen aus dem Trainingsdatensatz verworfen bzw. hinzugefügt werden, ist das endgültige Verhältnis der beiden Datensätze unterschiedlich zu dem Splitverhältnis.

4.2.3. Balancing

Das Klassenungleichgewicht des Trainingsdatensatzes wird mit dem Balancing Verfahren bearbeitet. Eine ungleiche Verteilung in den Trainingsdaten würde dazu führen, dass das Modell aufgrund des eingebrachten Bias nicht ausreichend generalisiert. Neben dem Ansatz die Datenverteilung zu verändern, existieren zudem Ansätze, welche die Probleme detektieren und diese während des Trainings angehen. Dazu zählen Ansätze wie beispielsweise das „Balancen des Losses“. Dabei wird anhand der Häufigkeitsverteilung der jeweiligen Klasse während der Kreuzentropie der Loss gebalanced [93]. Diese Algorithmen sind ausschließlich anwendbar auf Klassifikationsprobleme.

Die Ansätze bezüglich der Änderung der Datenverteilung werden häufig benutzt, da sie leicht implementierbar und anwendbar, sowie unabhängig von der Wahl des Modells sind [133]. In der Literatur wird häufig auf die Ungleichverteilung bei Klassifikationsproblemen eingegangen, meist jedoch nur unter der Berücksichtigung von entweder Over- oder von Undersampling Methoden [134].

Die Abbildung 14 zeigt schematisch den Effekt des Under- und Oversamplings auf die Klassenverteilung. Beim Undersampling werden überrepräsentierte Daten eliminiert, um einen Ausgleich der Daten zu erzeugen. Durch das Oversampling werden Daten in unterrepräsentierten Bereichen künstlich erzeugt und dem Set hinzugefügt oder überproportional oft verwendet, um das Netz zu trainieren.

Elhassan et al. [135] zeigen, dass das Undersampling in der Performance Vorteile ge-

genüber dem Oversampling hat, eine Kombination beider Methoden jedoch am effizientesten ist.

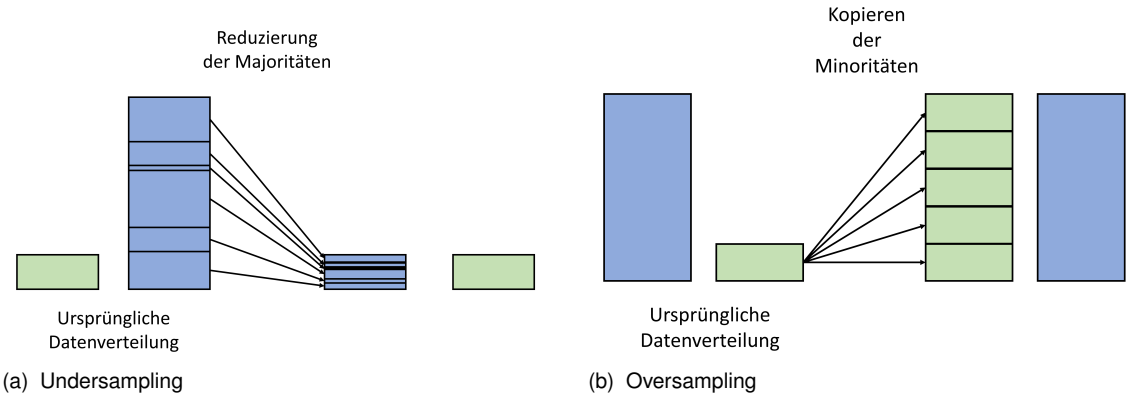


Abbildung 14 Prinzipielle Funktionsweise von Undersampling (a) und Oversampling (b).

Undersampling

In der Literatur wird weitgehend zwischen Random-Undersampling (RUS) und Focused-Undersampling (FUS) Methoden unterschieden. Beim FUS wird die Grenze zwischen zwei Klassen bereinigt, um eine exaktere Abgrenzung zu erlangen [135, 136]. Die RUS Methode entfernt zufällig Datenpunkte von überrepräsentierten Klassen. Der Nearest Neighbour Undersampling Algorithmus ist ein oft verwendeter RUS Algorithmus [137, 138]. Dieser entfernt Daten, die in der näheren Umgebung bereits in ähnlicher Weise vorkommen. Zusätzlich werden Datenpunkte, die sich von ihrer näheren Umgebung nur durch das Label unterscheiden, ebenfalls entfernt. Die Entfernung der Anomalien erhöht zwar die Robustheit des Modells, ist aber nur auf Klassifikationsprobleme anwendbar. Eine Erweiterung dazu ist der Tomek–Link [139], der die Entfernung zweier Punkte als Entscheidungsfaktor für das Eliminieren der Datenpunkte sieht. Diese genannten Methoden sind bisher nur bei Klassifikationsproblemen angewandt worden. Torgo et al. [140] haben eine auf Regressionsmodelle anwendbare Undersampling Methode vorgeschlagen, die eine Funktion bestimmt, welche die Wichtigkeit des jeweiligen Datenpunktes auf das Ergebnis schätzt und auf diese Weise die Daten aussortiert. Der Fokus dieser Methode liegt auf dem Vorhersagen von seltenen extremen Werten, wie Wirtschaftskrisen oder meteorologischen Ereignissen.

Für das in dieser Arbeit erstellte Regressionsmodell, welches auf der gesamten Feature Breite eine gute Generalisierung erreichen soll, wurde ein Algorithmus zum mehrdimensionalen Feature Undersampling Multi-Feature-Undersampling (MFU) entwickelt. Dieser Algorithmus beschränkt sich nicht nur auf Batterieanwendungen, er kann auch auf andere Regressionsdatensätze angewendet werden, die eine ungleiche Datenverteilung innerhalb der Feature Bereiche aufweisen.

Die Besonderheit des Algorithmus besteht darin, dass nicht nur die Verteilung der Zielva-

riable verbessert wird, sondern auch Eingangsvariablen sowie eigens für das Balancing definierte Variablen optimiert werden. Ein lediglich auf die Zielvariable bezogener RUS-Algorithmus würde dazu führen, dass Zustände aus dem Training Set, die zur besseren Generalisierbarkeit notwendig sind, möglicherweise entfernt werden. Im Beispiel der Spannungsprädiktion wäre dies der Fall, wenn häufig auftretende Spannungsbereiche so minimiert werden, dass beispielsweise die wenigen Daten, die bei tiefen Temperaturen vorhanden sind, nicht berücksichtigt werden. Das Modell würde hier lediglich auf häufig vorkommende Temperaturen trainiert werden und somit weiterhin mit einem Datenbias versehen sein.

Der MFU basiert auf einer Abwandlung des eindimensionalen RUS. Hierbei wird die Minoritätsklasse nicht zwischen zwei Klassen bzw. Features bestimmt, sondern innerhalb einzelner Feature Bereiche identifiziert, die überrepräsentiert sind. Der Wertebereich des ausgewählten Features wird in m gleich große Bins unterteilt, die jeweils mit einer maximalen Anzahl an Datenpunkten, dem Bin Limit bin_{limit} befüllt werden. Die Daten werden sequenziell den Bins zugeordnet. Wird ein Datenpunkt einem Bin zugeordnet, das die maximale Anzahl an Datenpunkten erreicht hat, wird dieser Datenpunkt verworfen. Würde das RUS nur auf ein einzelnes Feature angewendet werden, verbessert sich die Datenverteilung des gewählten Features. Allerdings kann somit kein Rückschluss auf die Verteilung anderer Features gezogen werden. Dieser Nachteil kann durch das MFU ausgeglichen werden.

Ein sequenzielles, eindimensionales Undersampling von mehreren Features führt zu zwei grundlegenden Problemen. Zum einen wird die resultierende Datenmenge sehr klein, da das Bin Limit bei jeder Iteration verringert werden muss. Zum anderen führt das unabhängige Undersampling der Features dazu, dass sich die Datenverteilung der zuerst genutzten Features durch nachgehende Undersamplings zum Negativen verändern kann. Die Lösung dieser beiden Probleme ist das MFU. Beim MFU wird das Undersampling nicht nur auf ein Feature angewandt, sondern auf alle n ausgewählten Balancing Features. Somit werden m^n Bins erzeugt, welche gleichzeitig befüllt werden. Das Auffüllen der Bins erfolgt, wie beim eindimensionalen Undersampling, durch die sequenzielle Verarbeitung der Eingangsdaten. Die Auswahl der Balancing Features bestimmt für welche Features eine bessere Verteilung erzielt werden soll.

Der Algorithmus kann in folgende sechs Schritte unterteilt werden:

1. Definition der n Balancing Features, anhand derer eine gleichmäßigere Datenverteilung hergestellt werden soll.

2. Unterteilung jedes Features in binrange m gleichgroße Wertebereiche anhand der jeweiligen minimal und maximal vorkommenden Werte im Trainingsdatenset.

3. Erzeugung eines Arrays A mit m^n Elementen (Bins). Die Bingrenzen werden durch

die Unterteilung der Featurebereiche aus 2. festgesetzt. Initial hat jedes Bin den Wert 0.

4. Definition des Bin Limits bin_{limit}.

5. Auffüllen der Bins und Erzeugung des Filtervektors F. Jede Sequenz der Eingangsdaten $Data_{in}$ wird einem entsprechenden Bin, je nach Wert der Features, zugeordnet.

   a. Ist der Wert des Bins größer als das Bin Limit bin_{limit}, dann wird der Filtervektor F um ein $False$ erweitert und der Wert des Bins bleibt bestehen.

   b. Ist der Wert des Bins kleiner oder gleich dem Bin Limit bin_{limit}, dann wird der Filtervektor F um ein $True$ erweitert und der Wert des Bins inkrementiert.

6. Der Filtervektor F wird auf die Eingangsdaten $Data_{in}$ angewendet und somit werden nur die Sequenzen berücksichtigt, welche in einem noch nicht überfüllten Bin zugeordnet wurden.

Der Pseudocode in Algorithmus 1 zeigt die algorithmische Funktionsweise des MFU.

Algorithmus 1 Multi-Feature-Undersampling Algorithmus

Input: n ▷ Anzahl der Undersampling Features
Input: m ▷ Anzahl der Bins pro Feature
Input: bin_{limit} ▷ Anzahl der Datenpunkte pro Bin
Input: k_{in} ▷ Anzahl der Datenpunkte vor dem Undersampling
Input: k_{out} ▷ Anzahl der Datenpunkte nach dem Undersampling
Ensure: $Data_{in} \in \mathbb{R}^{k_{in}}$ ▷ Input Datensatz
Ensure: $Data_{out} \in \mathbb{R}^{k_{out}}$ ▷ Output Datensatz
Ensure: $bin_{array} \in \mathbb{R}^{m^n}$ ▷ Bin-Array zur temporären Speicherung der Datenverteilung
Ensure: $bin_{value} \leftarrow 0$ ▷ aktuelle Anzahl Datenpunkte im Bin
Ensure: $F \in \mathbb{R}^{k_{in}} \leftarrow 0$ ▷ Filtervektor

for $feature = 1 : n$ **do**
    $feature_{max} \leftarrow$ Maximum value of $Data_{In}$ in row $feature$
    $feature_{min} \leftarrow$ Minimum value of $Data_{In}$ in row $feature$
    $feature_{range} \leftarrow feature_{max} - feature_{min}$
    $feature_{step} \leftarrow \frac{feature_{range}}{m}$
    for $o = 1 : m$ **do**
        $bin_{borders} \leftarrow feature_{min} + o \times feature_{step}$ ▷ Berechne die Grenzen der Bins
    end for
end for
for $date$ in $Data_{In}$ **do**
    for each feature in $date$ **do**
        find bin where $bin_{border_a} < date < bin_{border_b}$ ▷ Weise $date$ einem Bin zu
    end for
    if $bin_{value} < bin_{limit}$ **then**
        Append $True$ to F ▷ Erzeuge den Filter
        $bin_{value} = +1$
    else
        Append $False$ to F
        $bin_{value} = +0$
    end if
end for
$Data_{out} \leftarrow Data_{in}(F)$ ▷ Wende den Filter F auf den Datensatz $Data$ an
return $Data_{out}$

Die Auswahl der Parameter für diesen Algorithmus stellt einen Zielkonflikt dar. Zum einen sollen die Daten möglichst über ein breites Spektrum und alle Features gleichverteilt sein. Zum anderen braucht das NN möglichst viele Daten, um das Verhalten zu erlernen. Die folgenden drei Parameter müssen unter der Berücksichtigung vorangegangener Punkte gewählt werden:

- **Features:** Die Balancing Features stammen nicht nur aus den Input- und Output-Variablen der Daten. Zusätzlich zu diesen können für jede Sequenz weitere Kennzahlen bestimmt werden, welche eine Sequenz charakterisieren. Neben den Durchschnittswerten sind auch Minima und Maxima, deren Differenz und möglicherweise die Varianz einer Größe, von Interesse. Die Balancing Features müssen selektiert werden. Zu viele ausgewählte Features führen zu dem Effekt, dass zwei Datenpunkte zum Datenset hinzugefügt werden, obwohl der zusätzliche Informationsgehalt möglicherweise nur gering ist. Dies geschieht, wenn sich zwei Punkte grundsätzlich sehr ähnlich sind, sich jedoch die Werte eines Features deutlich unterscheiden. Werden zu wenige oder unwichtige Balancing Features gewählt, ist ein Ungleichgewicht in dem fehlenden Feature durch das Undersampling nicht ausgleichbar. Aus diesem Grund sollten die Balancing Features nach Relevanz mit dem Grundsatz „so wenige wie möglich, so viele wie nötig“ ausgewählt werden. Die Wahl und Anzahl der Features muss immer in Bezug zu den anderen Parametern getroffen werden.
- **Bin Range:** Die Bin Range m gibt an, in wie viele Bins der Feature Wertebereich unterteilt wird. Die Messgenauigkeit der Feature sollte höher sein als die resultierenden Teilwertebereiche. Dementsprechend ist das Maximum der Bin Range vorgegeben. Die minimale Bin Range ist dadurch begrenzt, dass eine kleine Bin Range Ungleichheiten weniger fein auflöst. Die Wahl der Bin Range zwischen den beschriebenen Minima und Maxima ist abhängig von der Wahl der Balancing Features und dem Bin Limit. Die Bin Range kann sich für jedes Feature unterscheiden, wird aber bei kontinuierlichen Features oft gleich gewählt.
- **Bin Limit:** Das Bin Limit l begrenzt die Anzahl der Sequenzen, die pro Bin abgespeichert werden. Wird das Limit zu groß gewählt, bleibt das Ungleichgewicht bestehen. Wohingegen ein zu kleines Limit wichtige Daten möglicherweise nicht berücksichtigt. Die Wahl des Bin Limits ist stark abhängig von der Bin Range.

Abbildung 15 stellt mit den grauen Punkten eine exemplarische Datenverteilung vor dem Undersampling dar, sowie mit den schwarzen Punkten, welche nach dem Undersampling weiterhin berücksichtigt werden.

Zur besseren Vergleichbarkeit werden im Folgenden die Undersampling Parameter in der Form $feautres\_features\_m/l$ zu jedem Datensatz angegeben. Ein Undersampling das mit den Features U_{mean} und T_{mean} sowie einer Bin Range von $m = 50$ und einem Bin Limit von $l = 20$ durchgeführt wird, wird als $U_{mean}\_T_{mean}\_50/20$ angegeben.

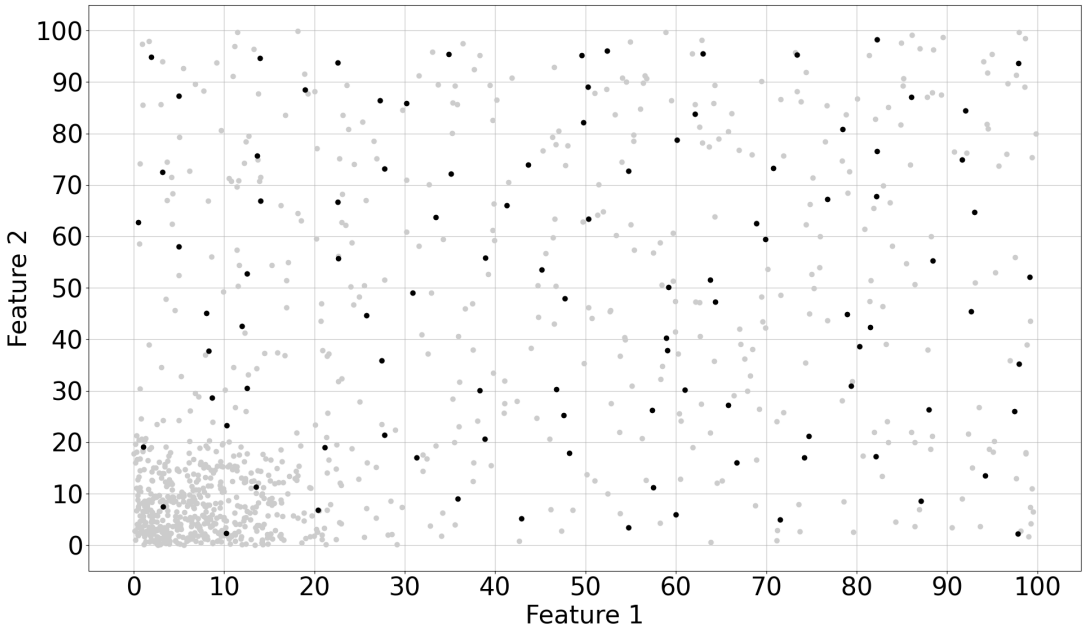


Abbildung 15 Graphische Darstellung eines exemplarischen Datensatzes vor (grau) und nach (schwarz) dem Undersampling

Oversampling

Als Ergänzung zum Undersampling wird zusätzlich eine Oversampling Methodik auf den Trainingsdatensatz angewandt. Ziel des nachgelagerten Oversamplings ist es, die unterrepräsentierten Klassen bzw. Feature Bereiche mit synthetischen Datenpunkten aufzufüllen. Dadurch kann eine homogenere Datenverteilung sichergestellt werden, ohne dabei das Datenset unverhältnismäßig stark zu undersamplen.

In der Literatur wurden viele Oversampling Methoden diskutiert. Methoden wie *ADASYN* [141] und *instance-based learning* [142] sind nur auf Klassifikationsprobleme anwendbar. Vimalraj et. al. [136] sehen die Synthetic Minority Oversampling Technique (SMOTE), initial eingeführt von Chawla et. al. [143], als den performantesten Algorithmus unter den aktuell in der Literatur diskutierten Algorithmen.

Bei der einfachsten Form des Oversamplings, dem Random Oversampling, werden aus der Minoritätsklasse zufällig Datenpunkte ausgewählt, die dann in vielfacher Ausführung dem Training Set hinzugefügt werden. Durch diese Methode tendiert ein neuronales Netz zum Overfitting, da bestimmte Datenpunkte häufiger vorkommen, somit wichtiger erscheinen und dadurch direkt auf den Ausgang abgebildet werden. Eine Weiterentwicklung des Random Oversamplings, die SMOTE, vermeidet diese Gefahr. Der SMOTE Algorithmus bestimmt in den vom Anwender definierten Minoritätsklassen für jeden darin enthaltenen Datenpunkt mit Hilfe des k-Nearest-Neighbor-Algorithmus neue Datenpunkte. Ein zufällig ausgewählter Punkt der als nächste Nachbarn klassifizierten Objekte wird dabei mit einer Zufallszahl zwischen 0 und 1 multipliziert. Torgo et. al. [140] haben den

SMOTE Algorithmus auf Regressionsprobleme adaptiert. Die Nachteile dieser Methode liegen zum einen in den Eigenschaften des k-Nearest-Neighbor-Algorithmus und zum anderen in der Sensitivität mancher Features auf die Zielvariable. Die Anzahl k der in Betracht gezogenen Objekte ist nur schwer zu parametrieren und hängt dabei auch vom Einzelfall ab. Ein zu hoch gewähltes k erweitert die Entfernung der Nachbarn derart, dass nicht von einer Ähnlichkeit mit dem ursprünglichen Datenpunkt ausgegangen werden kann. Wird andernfalls k zu klein gewählt, kann bei eng zusammenliegenden Punkten die Robustheit des Datensatzes negativ beeinträchtigt werden. Das Problem der Sensitivität mancher Features resultiert aus dem Berechnungsverfahren des k-Nearest-Neighbor-Algorithmus. Ein Punkt kann als einer der nächsten Nachbarn gezählt werden, obwohl die Entfernung der beiden Punkte in einem Feature weit ist. Wenn dieses Feature stark sensitiv ist, führt das zu einer möglichen Änderung der Zielvariable.

Da SMOTE in der Literatur als vielversprechend gilt, jedoch die oben genannten Nachteile mit sich bringt, wurde der Algorithmus auf den Anwendungsfall dieser Arbeit angepasst. Die Minoritätsklassen werden anhand eines Histogramms des Datensets nach dem Undersampling ausgewählt. Dabei wird vor allem das Feature Temperatur fokussiert, da der Wert innerhalb einer Sequenz wenig Volatilität aufweist und für die Robustheit des Modells Daten in allen Temperaturbereichen wichtig sind.

Der Algorithmus bietet zudem die Möglichkeit, mehrere Minoritätsklassen zu definieren, die individuelle Faktoren bzw. Schrittweiten in der Vervielfachung aufweisen können. Der Algorithmus 2 zeigt das algorithmische Vorgehen beim mehrschichtigen Oversampling. Jeder Datenpunkt in der Minoritätsklasse wird zur Verbesserung der Robustheit mit einem Temperatur- und Spannungsadditum beaufschlagt und anschließend dem Datenset hinzugefügt. Das Additum ist ein künstlich erzeugtes Rauschen, das maximal so groß gewählt ist, dass die ursprüngliche Feature Korrelation nicht beeinträchtigt wird. Der Bereich, in dem sich das Additum befindet, ist abhängig von der Sensitivität des Features. Inhomogene thermische Verteilungen im Batteriepack und der kleine temperaturabhängige Gradient ermöglichen eine sinnvolle Temperaturrange von $range_{temp}$ = ±2 °C. Die Zielvariable Spannung ist deutlich sensitiver auf kleine Veränderungen, weshalb die Spannungsrange $range_{volt}$ auf ±0,05 V festgelegt wird. Die Quantität des Oversamplings wird nach Bedarf an die jeweilige Minoritätsklasse angepasst, indem die step size adaptiert wird. Die Schrittweite $stepsize$ gibt an, in welcher Schrittweite die Rauschvektoren innerhalb der Spannungsrange $range_{volt}$ und Temperaturrange $range_{temp}$ gebildet werden. Je kleiner die Schrittweite, desto mehr Duplikate werden dem Datenset hinzugefügt.

Algorithmus 2 Mehrschichtiger Oversampling Algorithmus

Input: $MinorityClasses$ ▷ Minoritätsklassen, z.B. [T<5℃, T<-10℃]
Input: $stepsize$ ▷ Schrittweite zur Erzeugung der Rauschvektoren
Input: $range_{volt}$ ▷ Spannungsrange des Rauschvektors
Input: $range_{temp}$ ▷ Temperaturrange des Rauschvektors
Input: k_{in} ▷ Anzahl der Datenpunkte vor dem Oversampling
Input: k_{out} ▷ Anzahl der Datenpunkte nach dem Oversampling
Ensure: $Data_{in} \in \mathbb{R}^{k_{in}}$ ▷ Input Datensatz
Ensure: $Data_{out} \in \mathbb{R}^{k_{out}}$ ▷ Output Datensatz
Ensure: $Data_{os}$ ▷ Durch Oversampling erzeugte Daten

$temp_{noises} = -range_{temp} : stepsize : +range_{temp}$
$volt_{noises} = -range_{temp} : stepsize : range_{temp}$
for $MinorityClass$ in $MinorityClasses$ **do**
    for $data$ in $Data_{in}$ **do**
        if $data \in MinorityClass$ **then**
            for $temp_{noise}$ in $temp_{noises}$ **do**
                for $volt_{noise}$ in $volt_{noises}$ **do**
                    $Data_{new} \leftarrow data + temp_{noise} + volt_{noise}$ ▷ Addiere Rauschvektoren
                    Append $Data_{new}$ to $Data_{os}$ ▷ Erzeuge $Data_{os}$
                end for
            end for
        end if
    end for
end for
$Data_{out} \leftarrow Data_{in} + Data_{os}$ ▷ Füge $Data_{os}$ zu $Data_{in}$ hinzu
return $Data_{out}$

Die größte Gefahr beim Oversampling besteht darin, dass dem Datenset keine neuen Informationen hinzugefügt werden und dadurch eine Tendenz zum Overfitting entsteht [136]. Neben adäquaten Einstellung der Rauschterme, ist es wichtig darauf zu achten, dass das Oversampling immer in Zusammenhang mit dem Undersampling betrachtet wird und nur die Gesamtperformance bewertet wird.

Im Rahmen dieser Arbeit wurde in der Regel sowohl die Spannungsrange $range_{volt}$ als auch die Temperaturrange $range_{temp}$ auf den gleichen Wert gesetzt. Die Minoritätsklassen $MinorityClasses$ werden ausschließlich über die Temperaturgrenzen definiert. Die Temperaturgrenze ist so zu verstehen, dass unterhalb dieser ein Oversampling stattfindet. Um Datensätze mit verschiedenen Oversampling Parametern vergleichen zu können wird folgende Form eingeführt und in Zusammenhang mit den jeweiligen Datensätzen angegeben:
$range(range_{volt})/step(stepsize)/MC(MinorityClasses)$
Ein Oversampling mit einer Temperatur- und Spannungsrange von 2, einer Schrittweite von 1 und den Minoritätsklassen $\{-5, 5\}$ wird als $range(2)/step(1)/MC(-5/5)$ angegeben.

4.2.4. Normalisierung

Sowohl die absoluten Werte als auch die Range der Features unterscheiden sich deutlich. Der dadurch entstehende Bias würde zu einem langsamen Trainingsfortschritt, einer numerischen Instabilität und einer schlechteren Generalisierung beim Training von neuronalen Netzen führen.

Die Literatur ist sich uneinig über die beste Normalisierungsmethode in Bezug auf Klassifikationsprobleme [144, 145]. Regressionsprobleme werden oft mit dem Min-Max Scaler normalisiert [146, 147].

Da in keinem der Features des verwendeten Datensatzes extreme Ausreißer vorhanden sind, stellt der Z-Scaler keinen Vorteil gegenüber dem Min-Max Scaler dar.

Mit dem Min-Max Scaler wird der Wertebereich jedes Features auf den Bereich [0, 1] skaliert:

$$\hat{X}[:, i] = \frac{X[:, i] - min(X[:, i])}{max(X[:, i]) - min(X[:, i])} \tag{4.2}$$

Die minimalen und maximalen Werte werden aus dem Trainingsdatenset bestimmt und in der Normalisierung des Test- und Validierungsdatensets genutzt. Dadurch wird sichergestellt, dass alle Daten mit derselben Berechnungsvorschrift verändert werden und somit gleiche Feature Werte auf gleiche normalisierte Werte abgebildet werden.

5. Training und Hyperparametertuning

Für das Training wird zuerst das Modell mit dem TensorFlow framework initialisiert und das Trainingsdatenset geladen. Die Gewichte und Bias werden in mehreren Batches pro Epochen an die eingespeisten Trainingsdaten angepasst. Nach jeder Epoche wird anhand der Validierungsdaten der Validation Loss berechnet und mit dem Test Set der MAXE und der MAE ermittelt. Mithilfe dieser Metriken findet das Hyperparametertuning statt. Dieses optimiert sowohl die trainingsbasierten als auch die modellbezogenen Parameter dahingehend, dass das Modell möglichst in der gesamten Feature Breite das reale Verhalten der Batterie darstellt. Die Ergebnisse des Tunings werden mit Metriken und Fehlerdiagrammen bewertet.

Im Hyperparametertuning werden die Datensets, resultierend aus der Datenvorverarbeitung, auf konkrete Modelle angewandt. Ziel ist es, alle möglichen Hyperparameter so einzustellen, dass ein Modell entsteht, welches die größte Modellierungsgenauigkeit aufweist. Dabei ist auch darauf zu achten, dass manche Hyperparameter deutlichen Einfluss auf die Performance des Trainings haben. Um die Dauer eines Trainings im angemessenen Rahmen zu halten, wird allen Hyperparametern ein Parameterraum zugewiesen. Im nächsten Schritt werden Hyperparameter identifiziert, die festgelegt werden können, ohne auf ein extensives Training zurückgreifen zu müssen (Kapitel 5.1). In einem weiteren Schritt werden je Batterie, neun Datensätze mit unterschiedlichen Balancingparametern miteinander verglichen und der Datensatz mit dem besten Verhältnis zwischen Modellgüte und Trainingsgeschwindigkeit ausgewählt (Kapitel 5.3 und 5.4). Anschließend werden die trainingsbasierten Hyperparameter vorgetuned, um anschließend ein erstes Tuning der modellbezogenen Hyperparameter durchzuführen. Dieser Schritt wird mit den gewonnenen Erkenntnissen rekursiv wiederholt. Vidal et al. [76] zeigen, dass das Training von 50 gleichen Netzen nicht zu 50 gleichen Validierungsergebnissen führt. Aus diesem Grund wird das Netz mit den im Hyperparametertuning bestimmten Hyperparametern fünfmal trainiert, um das beste Netz zu finden.

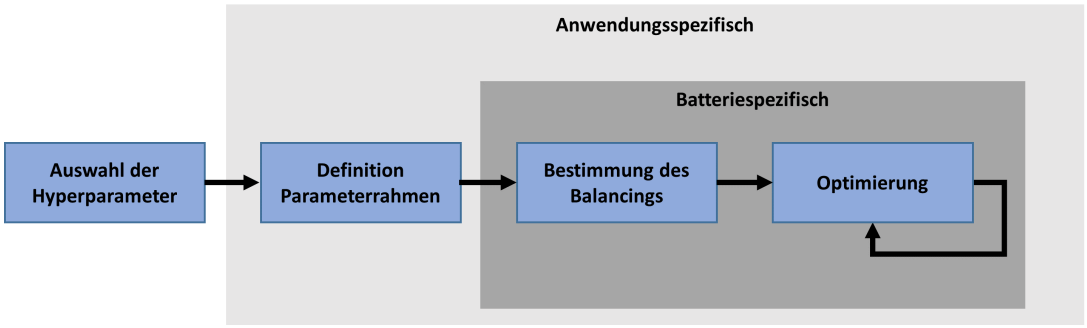


Abbildung 16 Prozess des Hyperparametertunings untergliedert in anwendungs- und batteriespezifische Schritte

In Abbildung 16 wird dieser Prozess visualisiert. Der Schritt „Definition Parameterrah-

men“ ist unabhängig von der zu modellierenden Batterie auf die Anwendung anzupassen. Wohingegen die Prozessschritte „Bestimmung des Balancings“ und „Optimierung“ für jedes Batteriesample einzeln auszuführen sind. Die resultierenden Hyperparameter können in jedem Fall als Anhaltspunkt für weitere Modellierungen genutzt werden.

5.1. Trainingsbasierte Hyperparameter

In erster Instanz werden die trainingsbasierten Hyperparameter getuned. Diese sind untergliedert in Parameter, die unabhängig von anderen Parametern pauschal definiert werden und in Parameter, die im Tuningalgorithmus berücksichtigt werden müssen. Diese Parameter sind unabhängig von der Verwendung des Modelltypen.

5.1.1. Unabhängige Parameter

Die nachfolgenden Parameter werden aufgrund von Erfahrungswerten und aktuellen Erkenntnissen der Wissenschaft pauschal definiert und im Trainingsprozess nicht mehr verändert.

- **Verlustfunktion:** Bei Regressionsproblemen wird in der Regel der mean-squared-error (MSE) als Verlustfunktion verwendet [109, 76]. Da sich der Einsatz des MSE bewährt hat, wird er in dieser Arbeit verwendet.
- **Batchgröße:** Die Batchgröße hat einen Einfluss auf die Trainingsgeschwindigkeit und auf den benötigten Arbeitsspeicher. Je kleiner ein Batch ist, desto häufiger müssen die Gewichte angepasst werden. Die Erfahrungen im Umgang mit der Batchgröße sowie Erkenntnisse aus der Literatur zeigen, dass auch bei großen Batches keine Generalisierungslücke auftritt [119]. Aus diesem Grund wird die Batchgröße so groß gewählt, wie es der Trainingsrechner ermöglicht. In dem vorliegenden Fall einer NVIDIA RTX 2080 Ti, ist die maximal mögliche Batchgröße auf 1024 begrenzt.
- **Sequenzlänge:** Die Sequenzlänge wirkt sich auf den für die Berechnung erforderlichen Speicherplatz sowie die Rechengeschwindigkeit direkt aus. In Anbetracht der Tatsache, dass Batterieeffekte stark zeitabhängig sind, zeigt sich eine große Sequenzlänge als vorteilhaft. Die internen Effekte der Diffusion, des Ladungstransfers, der elektrochemischen Doppelschicht und der Leitfähigkeit haben unterschiedliche Zeitabhängigkeiten. Für die Modellierung dieser Effekte reichen die Zeitkonstanten von Millisekunden bis Stunden und sind abhängig vom SOC und der Temperatur. Um einen Kompromiss zwischen Rechenkosten und Modellgenauigkeit zu finden, wurde die Sequenzlänge auf 128 Datenpunkte festgelegt. Die Daten wurden mit einer Abtastrate von 10 Hz aufgezeichnet, was bedeutet, dass jede Sequenz die letzten 12,8 s der Aufzeichnung darstellt [148, 149].
- **Updatezeit:** Die Updatezeit für U_{trend} ist auf 60 s festgelegt. Sie unterliegt dem Zielkonflikt, dass sich einerseits eine zu kurze Updatezeit zu einem Prädiktionsdrift entwickeln kann, da sich Schätzfehler kumulieren. Andererseits weisen zu lange Updatezeiten zu große Abweichungen von der aktuellen Spannung auf. 60 s hat sich dabei als eine funktionierende Zykluslänge gezeigt.

- **Initialisierungszeit:** Mit einer Initialisierungszeit von 70 s wird das Netz auf die aktuelle Spannung initialisiert. Dadurch setzt der Updateprozess im ersten Zyklus aus.
- **Pruning:** Der Hyperparametertuningframework Optuna unterstützt intelligente Pruning Methoden, um wenig vielversprechende Parameterzweige abzubrechen. Akiba et al. [127] sehen den Asynchronous Successive Halving Algorithm (ASHA), vorgestellt von Li et al. [150], als Stand der Technik.
- **Anzahl der Trials:** Die Anzahl der Trials pro Optuna-Lauf wurde an die jeweilige Situation, in Abhängigkeit von der Anzahl und Fülle der Variablen, angepasst.

5.1.2. Tuning Parameter

Für die folgenden Parameter werden Parameterräume definiert, die dann im Hyperprametertuning untersucht werden. Im Unterschied zu den unabhängigen Hyperparametern, werden diese Parameter in Abhängigkeit von den anderen Hyperparametern getuned.

- **Lernrate:** Wie in Kapitel 2.3.5 beschrieben, beeinflusst die Lernrate maßgeblich die Genauigkeit und Geschwindigkeit des Trainings. Laut Kriesel et al. [114] liegen Lernraten in der Regel zwischen 0,01 und 0,9, was sich in dieser Arbeit als deutlich zu groß herausgestellt hat. Die Erfahrung zeigt, dass in unserer Anwendung eine Lernrate im Bereich von

  $$0,0001 < \eta < 0,001 \tag{5.1}$$

  zielführend ist.
- **Optimierer:** Laut Rana et al. [116] wird der Adam Optimierer für Regressionsprobleme empfohlen. In Abbildung 20 und Abbildung 28 ist eine Vergleichsserie von Trainings mit den Optimierern Adam, AdaGrad, RMSprop und SGD zu sehen. Adam und RMSprop zeigen sowohl für die LTO-Batterie als auch für die LFP-Batterie Daten ähnlich gute Ergebnisse, wobei der Adam Optimierer den RMSprop leicht übertrifft. Aus diesem Grund wird das Hyperparametertuning mit Adam durchgeführt.
- **Batchnormalisierung:** Li et al. [151] sehen eine Disharmonie zwischen der Batchnormalisierung und der Anwendung von Dropouts. Um diese Aussage zu bewerten, wurde ein Vergleich von Modellen mit und ohne Batchnormalisierung angestellt. Dieser ist in Abbildung 21 und in Abbildung 29 dargestellt. Da eine Batchnormalisierung in diesem Fall das Ergebnis um bis zu 38 % verbessert, wird im Folgenden sowohl die Batchnormalisierung als auch Dropout Layer genutzt.
- **Anzahl der Epochen:** Die Anzahl der Epochen variiert je nach Datenmenge, Netzgröße und Rechenleistung. Komatsuzaki et al. [152] sprechen sich bezogen auf Transformermodelle für lediglich eine Epoche aus. Fleischer et. al [73] trainieren das ANFIS Modell in 30 Epochen, Zhao et al. [75] trainieren 800 Epochen und Vidal [76] bis

zu 20.000 Epochen. Im Rahmen der in dieser Arbeit verwendeten neuronalen Netze und Datenmengen, hat sich ein Training über 100 Epochen als ausreichend herausgestellt. Bei mehr als 100 Epochen stellt sich entweder ein Overfitting ein oder der Validation Loss stagniert. Letzteres Verhalten zeigt der Verlauf eines beispielhaften Trainings über 1000 Epochen in Abbildung 24 und Abbildung 32. Aus diesem Grund wird die Anzahl der Epochen auf 100 begrenzt.

5.2. Modellbezogene Hyperparameter

Jeder der modellbezogenen Hyperparameter ist abhängig von den jeweils anderen Hyperparametern. Dementsprechend müssen diese in einem zusammenhängenden Optimierungsverfahren getuned werden. Die Parameterräume sollen so groß wie notwendig und so klein wie möglich gewählt werden, damit die Optimierung mit einer rechenzeitverträglichen Anzahl an Iterationen durchgeführt werden kann.

5.2.1. Recurrent Neural Networks

Die elektrochemischen Prozesse in der Batterie weisen eine starke Zeitabhängigkeit auf. Um eine Zeitreihenprognose zu ermöglichen, sind rekurrente NN notwendig. Wie in Kapitel 2.3.3 bereits erläutert wurde, stoßen simpleRNNs bei größeren Zeitreihen auf das vanishing gradients Problem. Aus diesem Grund werden ausschließlich die Weiterentwicklungen der simpleRNNs, LSTM und GRU in die Optimierung miteinbezogen. Sowohl mit LSTM Zellen [153, 55] als auch mit GRU Zellen [154, 54] sind erfolgreiche Batteriemodelle erstellt worden.

Karsoliya et al. [122] haben untersucht, welche Anzahl an hidden Units und hidden Layer für NN benötigt werden, um zum einen gute Ergebnisse zu liefern und zum anderen den Rechenaufwand gering zu halten. Die Grundaussage ist, dass es keine allgemeingültige Daumenregel für diese beiden Parameter gibt. Jedoch gehen sie davon aus, dass der Einsatz von null bis drei hidden Layer sinnvoll ist. Eine größere Menge würde zu einem zu beträchtlichen Rechenaufwand führen. Die Autoren sind sich darüber einig, dass die Anzahl der hidden Units sowohl von dem speziellen Anwendungsfall, von der Datenmenge und -beschaffenheit als auch der Anzahl der Layer abhängt.

Anhand dieser Erkenntnisse sowie den im Rahmen der vorliegenden Arbeit gewonnenen Erfahrungen wurde die Anzahl der hidden Layer auf einen bis drei begrenzt. Die Anzahl der hidden Units wird während des Tunings aufgrund der Rechenperformance sowie von Speichergründen auf 128 begrenzt und in den hidden Layer konstant gehalten. Srivastava et al. [110] sehen eine Dropout Rate im Bereich von

$$0,2 < dr < 0,5 \tag{5.2}$$

als üblich an, empfehlen jedoch diese stets in Abhängigkeit von der Anzahl der hidden

Units festzulegen. Mit bis zu 128 hidden Units sind möglicherweise kleine Dropout Raten ebenso zielführend, was in dieser Arbeit zu einem Parameterraum von

$$0 < dr < 0.4 \tag{5.3}$$

führt. Das Dropout Layer wird zwischen dem letzten RNN Layer und dem ersten Dense Layer bzw. dem Output eingefügt.

Ein bis drei Dense Layer sind üblich und bringen einen Performancevorteil [109, 108]. Um die Wirksamkeit zu beweisen und gleichzeitig den Rechenaufwand zu beschränken, wird der Parameterraum für die Anzahl der Dense Layer auf null bis zwei gesetzt. Zur Vereinfachung wird in den Dense Layern dieselbe Anzahl an hidden Units verwendet wie in den RNNs.

5.2.2. Transformer

Die jüngsten Forschungen zeigen, dass die Transformer in gewissen Anwendungsbereichen deutlich bessere Ergebnisse liefern als LSTMs [112, 113]. Im Bereich der Regressionsprobleme sind in der Literatur keine Transformer-Modelle zu finden. Vaswani et al. [111] sehen in ihrer Einführung der Transformer den Vorteil, dass sich durch die Attention Mechanismen einfach und genau lange Zeitabhängigkeiten darstellen lassen. Die Pfade in der Modellierung sind auch für lange Zeitabstände kurz, somit wäre es denkbar, dass beispielsweise der Einfluss von großen Strompeaks aus langer Vergangenheit auf den aktuellen Zustand erkannt werden. Zudem kann durch die hohe Parallelisierung Rechenzeit eingespart werden. Des Weiteren ist die Komplexität des Transformer Algorithmus im Vergleich zu RNNs geringer, wenn die Sequenzlänge kleiner als die Dimension des Inputvektors ist. Da sich die Transformer zur Zeitreihenprognose eignen und jüngst gute Ergebnisse gezeigt wurden, werden im Rahmen dieser Arbeit die Transformer zusätzlich als Modelltyp in Betracht gezogen.

Die Parameterräume für die Transformer-Hyperparameter orientieren sich an dem Vorschlag von Vaswani et al. [111]. In Tabelle 2 sind die für das Hyperparametertuning verwendeten Parameterräume eingetragen.

Tabelle 2 Parameterräume für das Hyperparametertuning von Transformern

Parameter	**Parameterraum**
N	{2, 4, 6, 8}
D_{model}	{256, 512, 1028}
q	{16, 32, 64, 128, 512}
k	{16, 32, 64, 128, 512}
v	{16, 32, 64, 128, 512}
h	{1, 4, 8, 16, 32}
dr	{0,0, 0,1, 0,2}
pe_{period}	{128}

Um eine bessere Vergleichbarkeit zu gewährleisten, wird das Hyperparametertuning des Transformers, wie bei den RNNs, mit dem Hyperparameterframework Optuna durchgeführt.

5.3. Hyperparametertuning des LTO-Batterie Modells

Im nachfolgenden wird der Prozess und die Ergebnisse des Hyperparametertunings vorgestellt.

5.3.1. Auswahl des Balancings

Zunächst erfolgt die Auswahl der Under- und Oversamplingparameter, anhand derer die Daten für das Training des Netzes bestimmt werden. Eine vollständige Raster- oder Zufallsoptimierung ist in diesem Zusammenhang nicht praktikabel, da dies einen zu großen Rechen- und Speicheraufwand bedeuten würde. Erfahrungswerte und augenscheinliche graphische Betrachtungen führen in diesem Fall in kürzerer Zeit zu guten Ergebnissen.

Der erste Schritt besteht darin die möglichen Undersampling Features zu bewerten und dementsprechend eine nicht zu große Anzahl an Features auszuwählen.

In Tabelle 3 sind die generischen Undersampling Features beschrieben sowie deren Bewertung in Bezug auf den Kontext der Spannungs- und Leistungsprädiktion einer Batterie formuliert.

Tabelle 3 Subjektive Bewertung der Wichtigkeit von generischen Undersampling Features

Name	Berechnung	Bewertung	Begründung
T_{mean}	$Mean(T)$	Sehr wichtig	Großer Einfluss der Temperatur auf das Verhalten der Batterie
dU	$Max(U) - Min(U)$	Wichtig	Hohe Spannungsvolatilitäten beinhalten wichtige Extremsituationen
U_{mean}	$Mean(U)$	Sehr wichtig	Abbildung des Ladezustands mit Überproportionalität in den Grenzbereichen
$U_{meandev}$	$Std(U)$	Wichtig	Hohe Spannungsvolatilitäten beinhalten wichtige Extremsituationen
dI	$Max(I) - Min(I)$	Wichtig	Hohe Stromvolatilitäten beinhalten wichtige Extremsituationen
I_{mean}	$Mean(I)$	Sehr wichtig	Beachtung positiver und negativer Konstantbelastungen
$I_{meandev}$	$Std(I)$	Wichtig	Hohe Stromvolatilitäten beinhalten wichtige Extremsituationen
dI_{last}	$I(t = seq_{len})$	Eher unwichtig	Reduktion der Infromationsdimension
dU_{last}	$U(t = seq_{len})$	Eher unwichtig	Reduktion der Infromationsdimension

Im ersten Schritt erfolgt die Auswahl der Features:

- $\mathbf{T_{mean}}$**:** Viele Messpunkte sind bei Temperaturen > 0 °C und insbesondere >15 °C aufgezeichnet worden, daher ist ein auf die Temperatur bezogenes Undersampling notwendig. Der Mittelwert der Temperatur innerhalb einer Sequenz spiegelt den Zustand der Batterie wieder.
- $\mathbf{U_{mean}}$**:** Die Batterie wird im MHEV Bordnetz oft im mittleren Ladezustandsbereich betrieben. Dadurch treten Spannungen aufgrund der Korrelation von OCV und SOC Spannung häufig im mittleren Wertebereich auf. Ein Undersampling auf das arithmetische Mittel der Spannung hilft, um extreme Spannungswerte sowie extreme Ladezustände trotzdem gleichgewichtet abzubilden.
- $\mathbf{I_{mean}}$**:** Hohe absolute Ströme kommen nur in wenigen Fahrsituationen vor. Um diese Extremwerte zu repräsentieren ist ein Undersampling auf das arithmetische Mittel des Stromes sinnvoll. Zudem wird dabei sowohl der Lade- als auch der Entladefall miteinbezogen.

Die restlichen Features beziehen sich vor allem auf extreme Volatilitäten in der Sequenz ($I_{meandev}$, dI, $U_{meandev}$, dU) oder reduzieren den Informationsgehalt (dI_{last}, dU_{last}) und werden daher anfänglich nicht berücksichtigt.

In Abbildung 17 sind die Datenverteilungen der einzelnen Eingangsfeatures vor und nach dem Undersampling dargestellt. Die sehr häufigen Datenpunkte im Strombereich von -5 A bis 20 A und im Bereich um eine Spannung von 44 V sind nach dem MFU deutlich seltener vertreten. Dies hat den Nebeneffekt, dass die seltenen Fälle, wie hohe und niedrige Ströme, im Verhältnis häufig vorkommen. Das Ziel, eine ausgewogenere Datenverteilung entlang der Features zu erreichen, ist damit durch den MFU erfüllt worden.

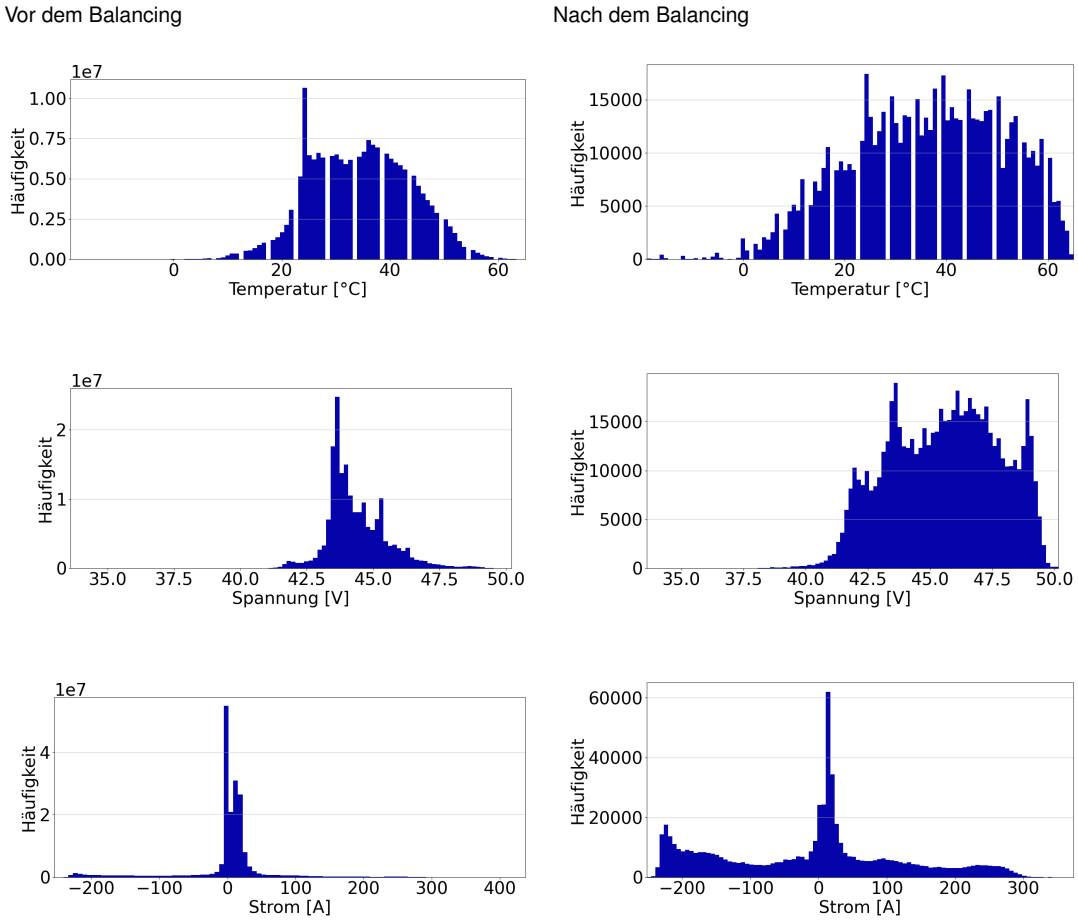


Abbildung 17 LTO-Batterie - Datenverteilung der Features Temperatur, Spannung und Strom vor und nach einem Undersampling der Daten mit den Undersampling Parametern $I_{mean}\_T_{mean}\_U_{mean}\_50/20$

Das Undersampling wird für die ausgewählten Features anhand der neun Kombinationen der beiden Parameter Bin Limit und Bin Range mit den Wertebereichen

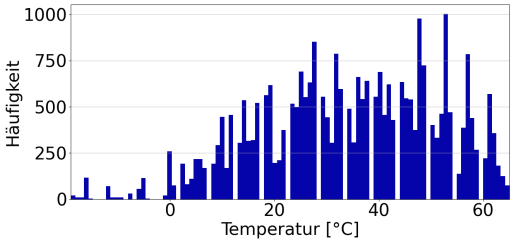
$$l \in \{20, 50, 100\} \tag{5.4}$$

und

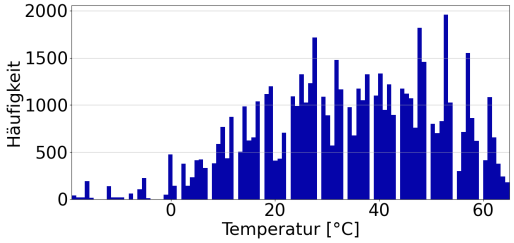
$$m \in \{10, 50, 100\} \tag{5.5}$$

durchgeführt.

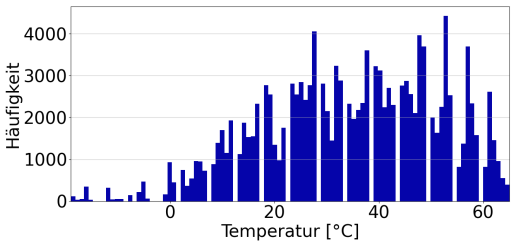
In Abbildung 18 sind die Histogramme der Temperatur für die neun Kombinationen dargestellt. Die Bewertung der Datenverteilungen erfolgt nach den Kriterien „Datenmenge“ und „Gleichverteilung“. Eine zu große Datenmenge verlangsamt das Training. Zu wenige Daten führen dagegen zu einer schlechteren Generalisierbarkeit. Durch den nachfolgenden Schritt des Oversamplings kann nicht abschließend beurteilt werden, welches MFU am besten geeignet ist.



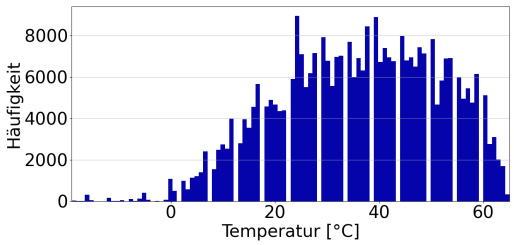
(a) $I_{mean}\_T_{mean}\_U_{mean}\_20/10$



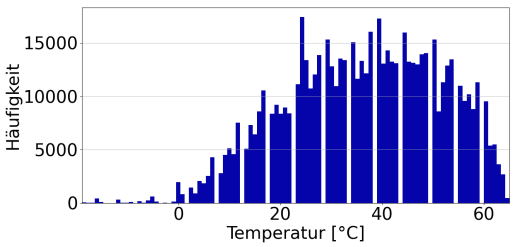
(b) $I_{mean}\_T_{mean}\_U_{mean}\_20/20$



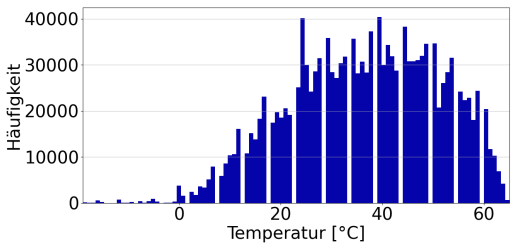
(c) $I_{mean}\_T_{mean}\_U_{mean}\_20/50$



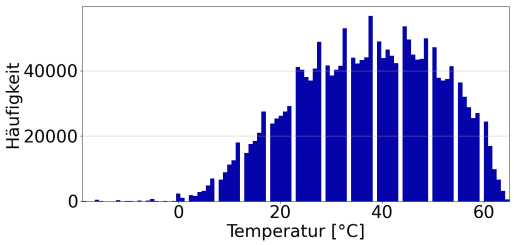
(d) $I_{mean}\_T_{mean}\_U_{mean}\_50/10$



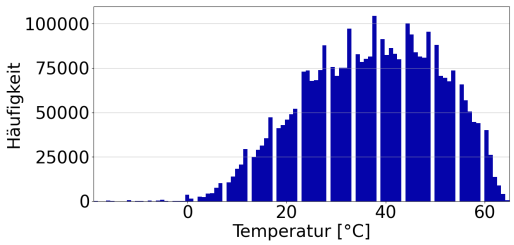
(e) $I_{mean}\_T_{mean}\_U_{mean}\_50/20$



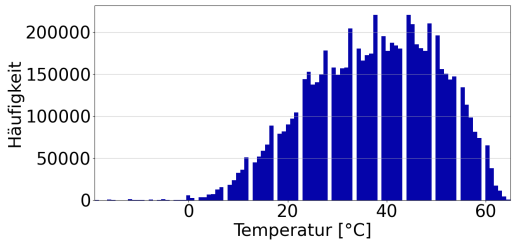
(f) $I_{mean}\_T_{mean}\_U_{mean}\_50/50$



(g) $I_{mean}\_T_{mean}\_U_{mean}\_100/10$



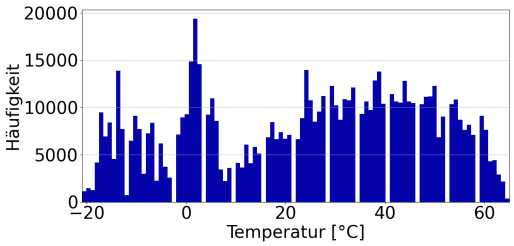
(h) $I_{mean}\_T_{mean}\_U_{mean}\_100/20$



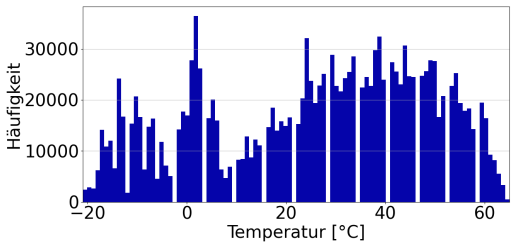
(i) $I_{mean}\_T_{mean}\_U_{mean}\_100/50$

Abbildung 18 LTO-Batterie - Vergleich der Datenverteilung anhand des Features Temperatur bei unterschiedlichen Undersampling Parametern

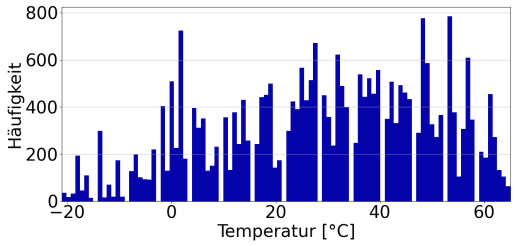
Abbildung 19 zeigt die Auswahl der MFU, welche für die weitere Verarbeitung am vielversprechendsten sind. Nach dem MFU besteht die Möglichkeit des Oversamplings, was im Hinblick auf die stark unterrepräsentierten tiefen Temperaturen zielführend scheint. Im Undersampling Datensatz $I_{mean}\_T_{mean}\_U_{mean}\_20/10$ ist eine Minorität bei Temperaturen < 5 °C zu erkennen. Durch ein einstufiges Oversampling in diesem Bereich ergibt sich ein Datensatz, der über eine ausreichende Menge an Daten verfügt sowie eine niedrige Varianz der Temperaturverteilung aufzeigt. Die Datensätze $I_{mean}\_T_{mean}\_U_{mean}\_50/20$ und $I_{mean}\_T_{mean}\_U_{mean}\_50/50$ weisen eine größere Diskrepanz zwischen den niedrigen und milden Temperaturen auf. Hierbei wurde ein zweistufiges Oversampling für die Minoritätsbereich $T < 5C$ und zusätzlich für den Bereich $T < -5C$ angewendet. Die resultierenden Temperaturhistogramme sind ausgewogen und die Datenmenge ist ausreichend.



(a) $I_{mean}\_T_{mean}\_U_{mean}\_50/20$
$range(2)/step(1)/MC(-5/5)$



(b) $I_{mean}\_T_{mean}\_U_{mean}\_50/50$
$range(2)/step(1)/MC(-5/5)$



(c) $I_{mean}\_T_{mean}\_U_{mean}\_20/10$
$range(2)/step(2)/MC(5)$

Abbildung 19 LTO-Batterie - Datenverteilungen von verschiedenen Oversampling Parametern für ausgewählte MFU Datensätze

Um die erzeugten Datensätze letztendlich vergleichen zu können, wurden damit die ersten Netze trainiert. In Tabelle 4 sind die Ergebnisse der Trainings dargestellt. Trainiert wurden in erster Linie ein RNN mit LSTM Zellen. Die modellbezogenen Hyperparameter wurden in einer kurzen Optuna Variante getuned. Am schlechtesten schneidet das Balancing Nr. 3 ab, was durch die geringe Datenmenge begründet werden kann. Das Balancing Nr. 1 hat mit einem MAXE von 3,26 V eine bessere Generalisierbarkeit vorzuweisen als das Balancing Nr. 2. Zusätzlich zu den LSTMs wurden im Anschluss Transformer trainiert. Das Balancing Nr. 3 wurde dabei aufgrund der schlechten Performance bei den LSTMs nicht berücksichtigt. Die Transformer weisen ebenso bei der Variante mit Balancing Nr. 1 die besseren Ergebnisse auf.

Tabelle 4 LTO-Batterie - MAXE von RNN und Transformer Modellen für verschiedene Balancing Kombinationen

Nr.	Undersampling	Oversampling	RNN	Transformer
1	$I_{mean}\_T_{mean}\_U_{mean}\_50/20$	$range(2)/step(1)/MC(-5/5)$	3,26 V	3,74 V
2	$I_{mean}\_T_{mean}\_U_{mean}\_50/50$	$range(2)/step(1)/MC(-5/5)$	3,46 V	5,9 V
3	$I_{mean}\_T_{mean}\_U_{mean}\_20/10$	$range(2)/step(2)/MC(5)$	4,10 V	nan

Im nächsten Schritt werden von dem Balancing Nr. 2 zwei Varianten gebildet. Zum einen wird das Oversampling reduziert auf ein einstufiges Verfahren (vgl. Tabelle 5 Balancing Nr. 4) und zum anderen wird die Balancing Feature Kombination (vgl. Tabelle 5 Balancing Nr. 5) geändert. Die Reduktion des Ovesamplings bringt, wie in Tabelle 5 zu sehen, im Vergleich zur Basisvariante keine Verbesserung mit sich. Jedoch ist der MAXE bei der Variante mit dem Undersampling $I_{mean}\_T_{mean}\_dU\_50/20$ und dem zweistufigen Oversampling $range(2)/step(1)/MC(-5/5)$ bei 3,04 V. Die Auswahl des Balancings ist damit abgeschlossen. Da die einstufige Oversampling Variante schlechter abschneidet als die zweistufige, wird auf ein Transformer Training mit diesem Datensatz verzichtet. Der Transformer, der mit dem finalen Datensatz trainiert wurde, erzielt den geringsten MAXE im Vergleich der aller vorher trainierten Transformer, weist aber eine deutlich höhere Fehlerrate als das LSTM-Pendant auf. Da die Performance der Transformer mit allen Balancing Varianten schlechter als die der RNNs ist, wird im Folgenden der Fokus auf RNNs gesetzt.

Tabelle 5 LTO-Batterie - MAXE von RNN und Transformer Modellen für weitere Balancing Kombinationen

Nr.	Undersampling	Oversampling	RNN	Transformer
4	$I_{mean}\_T_{mean}\_U_{mean}\_50/20$	$range(2)/step(1)/MC(5)$	3,87 V	nan
5	$I_{mean}\_T_{mean}\_dU\_50/20$	$range(2)/step(1)/MC(-5/5)$	3,04 V	4,54 V

5.3.2. Begründung für Wahl der Hyperparameter

Nach der Wahl des Balancings müssen die Hyperparameter, wie in Kapitel 5.1.2 beschrieben, getuned werden. Tabelle 6 zeigt die zu tunenden Parameter mit deren Parameterräumen sowie das Ergebnis des Tunings.

Als vorläufig bestes RNN hat ein LSTM mit 100 hidden Layer, ein Dense Layer, zwei hidden Units und einer Dropout Rate von 0.19 abgeschnitten. Dieses NN wird im Folgenden als Basis für die Auswahl der Modellparameter genutzt.

Tabelle 6 LTO-Batterie - Parameterräume für das Hyperparametertuning mit den jeweiligen Ergebnissen des Tunings

	Parameterraum	**Ergebnis des Tunings**
Optimizer	{AdaGrad, Adam, RMSprop, SGD}	Adam
Learning Rate	{0,0001 : 0,001}	0,0002
Batchnormalisierung	{On, Off}	On

Die Wahl des Optimierers ist basierend auf vier Vergleichstrainings getroffen worden. Vier Trainings und Validierungen wurden für vier gleiche NN mit dem gleichen Datensatz mit je einem anderen Optimierer durchgeführt. Wie in Abbildung 20 zu sehen ist, schneiden im Vergleich zu Adam und RMSprop die Optimierer AdaGrad und SGD sowohl in Bezug auf den Loss als auch auf den MAXE schlechter ab. Im Loss sind zwischen RMSprop und Adam kaum Unterschiede zu erkennen. Bezüglich des MAXEs ist der Adam Optimierer zwar volatiler, jedoch in einigen Epochen mit Abstand am kleinsten.

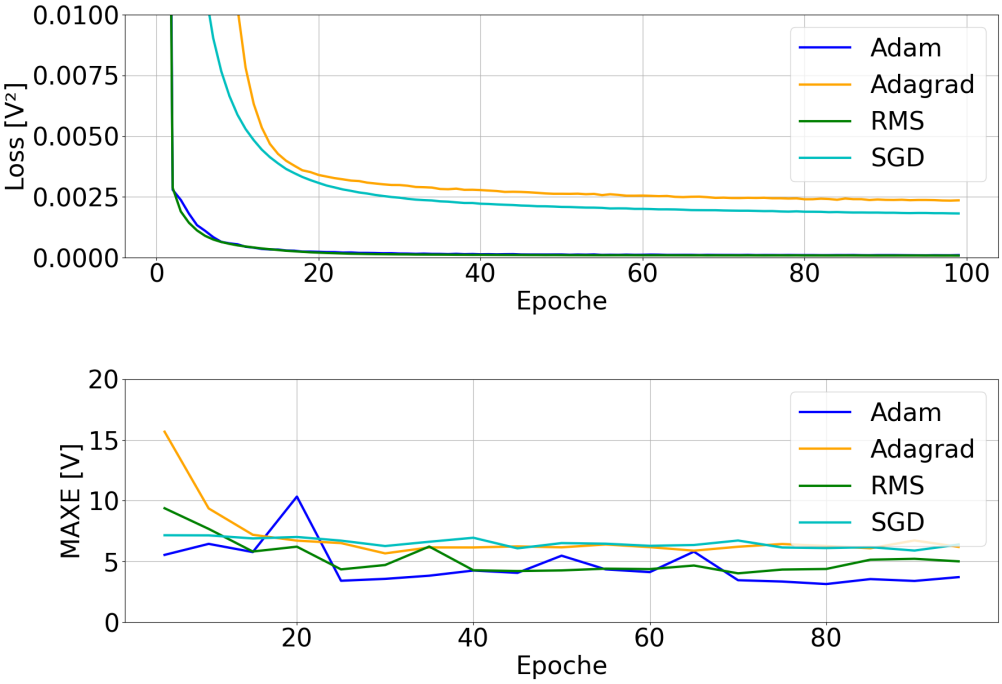


Abbildung 20 LTO-Batterie - Vergleich der Trainingsperformance verschiedener Optimierer

Die Learningrate (LR) ist in zehn Trials in einem Bereich zwischen 0.0001 und 0.001 getestet worden. Bei keinem dieser LRs ist innerhalb der ersten 100 Epochen ein Overfitting festzustellen, wobei weder der Loss noch der Validation Loss in der zweiten Hälfte des Trainings signifikant abnimmt. Insgesamt ist kein direkter Einfluss der LR auf das Ergebnis erkennbar. Das Modell mit dem kleinsten MAXE wurde mit einer LR von 0.002 trainiert, somit wird im weiteren Verlauf des Hyperparametertunings die LR auf diesen Wert festgesetzt.

Durch Vergleichstrainings konnte die Aussage von Li et al. [151] über die Disharmonie von Batchnormalisierung und Dropout widerlegt werden. In Abbildung 21 sind zwei Trainings von zwei gleichen LSTM Netzen mit einer Dropout Rate von 0,1 dargestellt. Der Loss wird während des Trainings ohne Batchnormalisierung (BN) schneller kleiner als mit BN. Jedoch wird in der Validierung ein deutlich kleinerer MAXE erreicht. Somit ist die BN in diesem Anwendungsfall ein probates Mittel, um den Fehler zu minimieren.

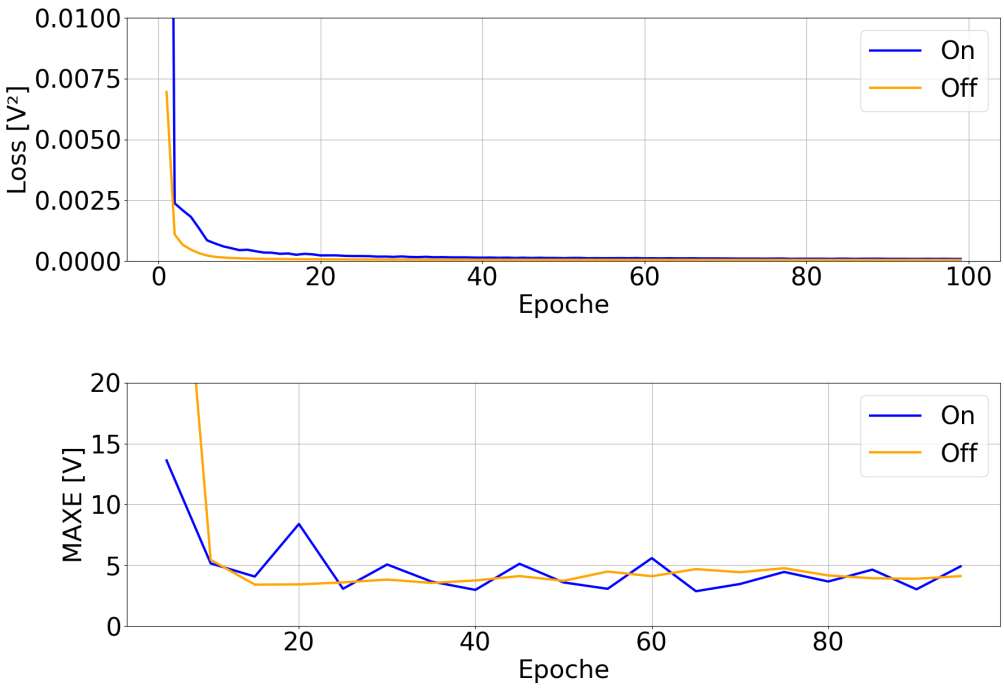


Abbildung 21 LTO-Batterie - Vergleich der Trainingsperformance zweier Modelle mit aktiver und inaktiver Batchnormalisierung

5.3.3. Modellparametertuning

Das finale Tuning der Modellparameter wurde unter der Berücksichtigung der vorher bestimmten Hyperparameter sowie der Auswahl des Datensatzes mit Optuna durchgeführt. Die Parameterräume aus Kapitel 5.2 wurden Optuna als zu tunende Parameter übergeben und in 34 Trials getuned. Abbildung 22 zeigt das Ergebnis des Tunings. Ein LSTM Netzwerk mit 100 hidden Units, in ein RNN Layer und zwei Dense Layern mit einer Dropout Rate von 0.01 erreicht den kleinsten MAXE. Aufgrund der Größe des Datensatzes konnte in keinem der Trials Overfitting festgestellt werden.

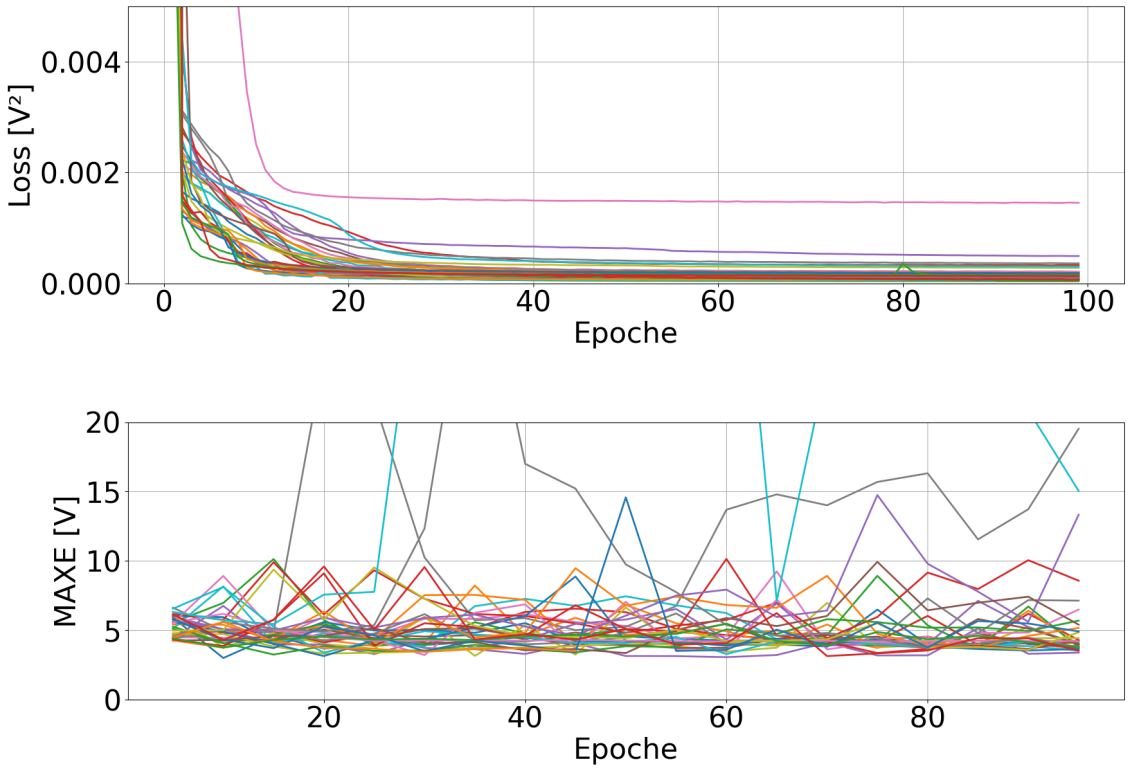


Abbildung 22 LTO-Batterie - Modellparametertuning in 34 Trials des Optuna-Algorithmus

Das bis dahin beste Modell wurde mit denselben modellbezogenen Parametern nach dem Ansatz von Vidal et al. [56] fünf Mal trainiert. Durch die unterschiedlichen Initialisierungen der Gewichte und Biases zwischen den Neuronen ergeben sich unterschiedliche Trainingsverläufe. In diesem Fall kann der MAXE weiter minimiert werden. Abbildung 23 zeigt den Trainingsprogress mit einem Optimum in Epoche 20.

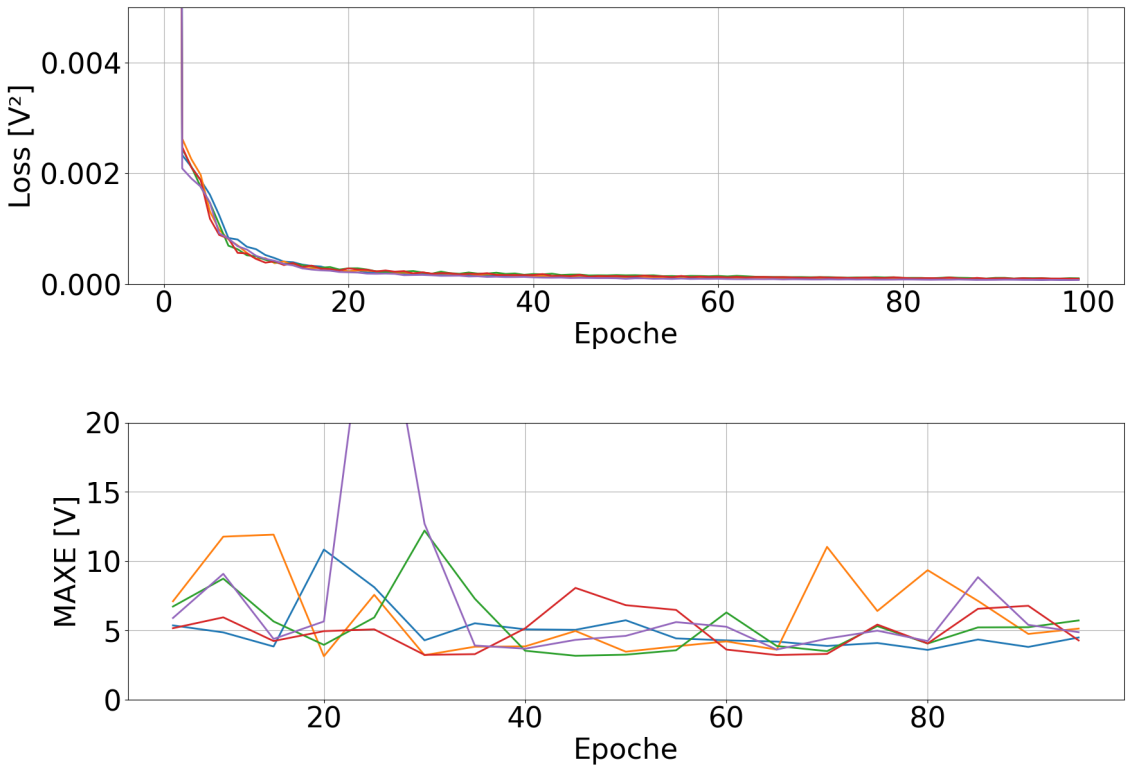


Abbildung 23 LTO-Batterie - Fünfmalige Wiederholung des Trainings

Abbildung 24 zeigt den Loss, Validation Loss und MAXE im Verlauf eines Trainings mit 1000 Epochen. Der über die gesamte Laufzeit sinkende Loss und Validation Loss deutet daraufhin, dass kein Overfitting auftritt. Dagegen erreicht der MAXE sein Minimum innerhalb der ersten 100 Epochen. Aus diesem Grund ist ein Training über mehr als 100 Epochen in der in dieser Arbeit vorgestellten Anwendung nicht zielführend.

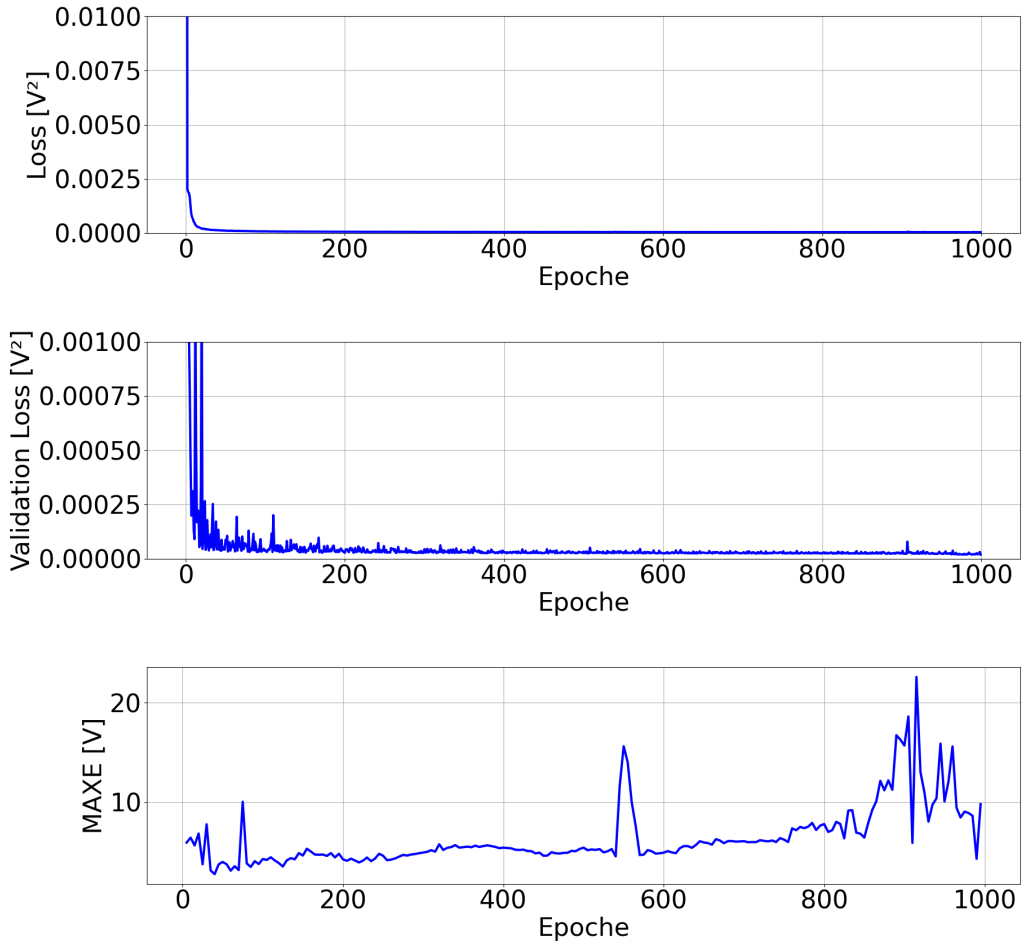


Abbildung 24 LTO-Batterie - Loss, Validation Loss und MAXE im Verlauf eines Trainings mit 1000 Epochen

Die Tabelle 7 zeigt die Fehlermetriken des LTO-Modells mit dem geringsten MAXE.

Tabelle 7 LTO-Batterie - Fehlermetriken des finalen LTO-Batterie Modells

Balancing Konfiguration	
Nr.	5
Undersampling	$I_{mean}\_T_{mean}\_dU\_50/20$
Oversampling	$range(2)/step(1)/MC(-5/5)$
Netzwerk Konfiguration	
Netzwerktyp	LSTM
Anzahl hidden Units	100
Anzahl RNN Layer	1
Anzahl Dense Layer	2
Dropout Rate	0,01
MAE	0,47 V
MAXE	2,77 V

5.4. Hyperparametertuning des LFP-Batterie Modells

Der Prozess des Hyperparametertunings wurde im vorherigen Kapitel 5.3 am Beispiel der LTO-Batterie beschrieben und soll in diesem Abschnitt auf die LFP Batterie angewandt werden. Ziel ist die Verifikation dieses Prozesses.

5.4.1. Auswahl des Balancings

Der erste Schritt, die Auswahl der Balancing Parameter, erfolgt wie bei der LTO-Batterie unter der Voraussetzung gleicher Annahmen bezüglich der Wichtigkeit der Parameter. Maßgeblich ist wieder Tabelle 3, mit der Schlussfolgerung, dass die Features I_{mean}, T_{mean}, U_{mean} für die ersten MFU genutzt werden.

Abbildung 25 zeigt die Datenverteilungen der einzelnen Eingangsfeatures vor und nach dem MFU. Die Temperatur weist eine deutlich bessere Datenverteilung auf. Zu sehen ist die Reduktion der überproportional häufigen Datenpunkte im Spannungsbereich von rund 46 V. Zudem ist der Strompeak um 0 A deutlich kleiner. Das Ziel, eine ausgewogenere Datenverteilung entlang der Features zu erreichen, ist auch für diesen Datensatz durch den MFU erfüllt worden.

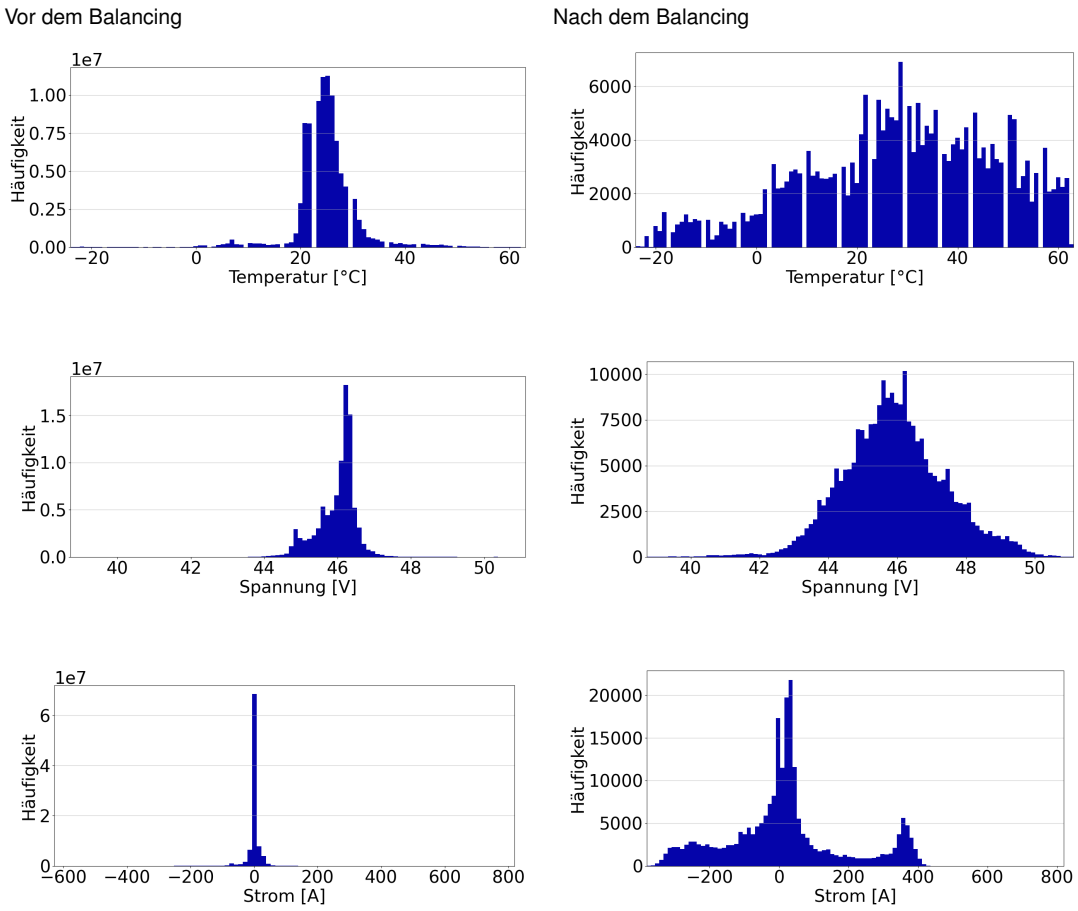
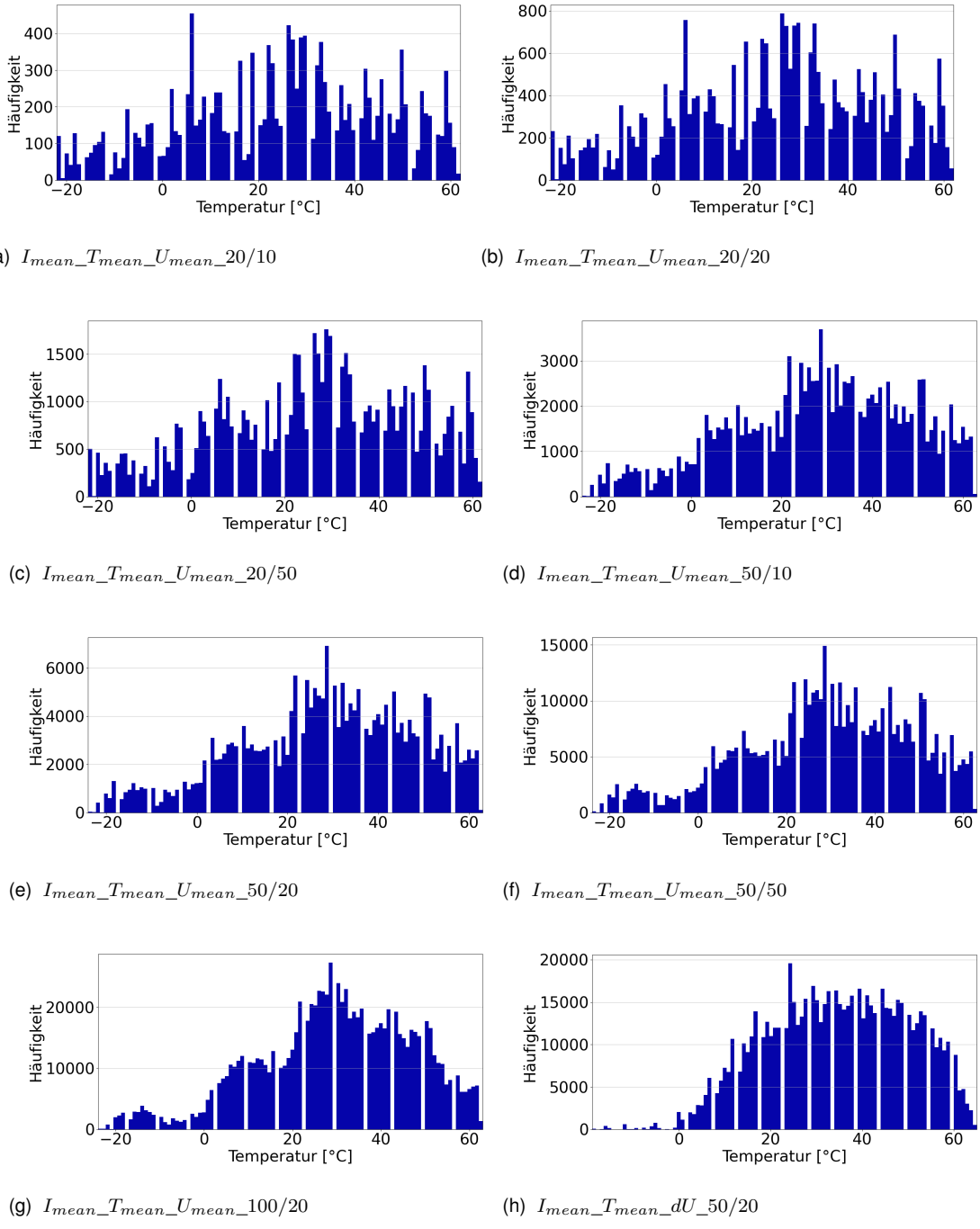


Abbildung 25 LFP-Batterie - Datenverteilung der Features Temperatur, Spannung und Strom vor und nach einem Undersampling der Daten mit den Undersampling Parametern $I_{mean}\_T_{mean}\_U_{mean}\_50/20$

Das MFU wird, wie bei der LTO-Batterie, für die neun Varianten aus Bin Limits und Bin Range durchgeführt. Aufgrund der guten Performance bei der LTO-Batterie wird zusätzlich ein MFU auf die Balancing Feature Kombination I_{mean}, T_{mean}, dU angewandt. Die sich daraus ergebenden acht Histogramme der Datenverteilung der Temperatur sind in Abbildung 26 zu sehen.



(a) $I_{mean}\_T_{mean}\_U_{mean}\_20/10$

(b) $I_{mean}\_T_{mean}\_U_{mean}\_20/20$

(c) $I_{mean}\_T_{mean}\_U_{mean}\_20/50$

(d) $I_{mean}\_T_{mean}\_U_{mean}\_50/10$

(e) $I_{mean}\_T_{mean}\_U_{mean}\_50/20$

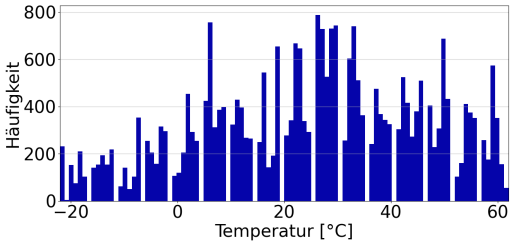
(f) $I_{mean}\_T_{mean}\_U_{mean}\_50/50$

(g) $I_{mean}\_T_{mean}\_U_{mean}\_100/20$

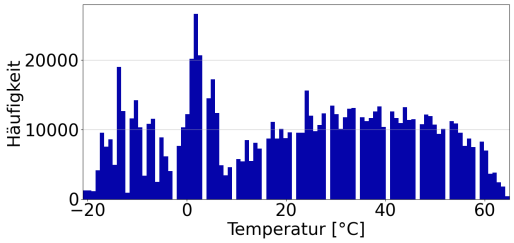
(h) $I_{mean}\_T_{mean}\_dU\_50/20$

Abbildung 26 LFP-Batterie - Vergleich der Datenverteilung anhand des Features Temperatur bei unterschiedlichen Undersampling Parametern

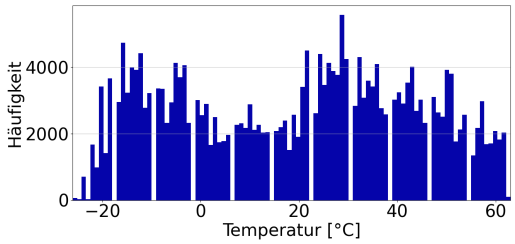
Die unterrepräsentierten Bereiche werden durch das gezielte Oversampling stärker gewichtet. Die Abbildung 27 zeigt die Datensätze, für welche erste Modelle trainiert und anhand der Fehlermetriken bewertet werden. Das Ergebnis des MFU $I_{mean}\_T_{mean}\_U_{mean}\_20/20$ ist eine relativ gleichverteilte Temperaturhäufigkeit. Da bei diesem Datensatz kein klarer Minoritätsbereich erkennbar ist, wird hier auf das Oversampling verzichtet. Das MFU $I_{mean}\_T_{mean}\_U_{mean}\_50/20$ hat deutlich mehr Datenpunkte, jedoch ist eine Minorität bei Temperaturen < 0℃ zu erkennen. Diese Minorität wird mit einem einstufigen Oversampling aufgewertet. Der Datensatz $I_{mean}\_T_{mean}\_dU\_50/20$ wird mit einem zweistufigen Oversampling, jeweils für die Temperaturen zwischen -5℃ und 5℃, versehen.



(a) $I_{mean}\_T_{mean}\_U_{mean}\_20/20$



(b) $I_{mean}\_T_{mean}\_dU\_50/20$
$range(2)/step(1)/MC(-5/5)$



(c) $I_{mean}\_T_{mean}\_U_{mean}\_50/20$
$range(2)/step(2)/MC(0)$

Abbildung 27 LFP-Batterie - Datenverteilungen von verschiedenen Oversampling Parametern für ausgewählte MFU Datensätze

Der Datensatz, mit der Balancing Konfiguration $I_{mean}\_T_{mean}\_dU\_50/20\ range(2)/step(1)/MC(-5/5)$, weist im Gegensatz zur Modellierung der LTO-Batterie bei der LFP-Batterie einen deutlich größeren MAXE auf. Vergleichbare Datensätze mit einem $I_{mean}\_T_{mean}\_U_{mean}\_50/20$ Undersampling zeigen eine bessere Performance, wobei der Datensatz mit einem Undersampling von $I_{mean}\_T_{mean}\_U_{mean}\_20/20$ insgesamt zu wenige Datenpunkte aufweist. Dies ermöglicht eine Generalisierung nur minderwertig. Die Variante $I_{mean}\_T_{mean}\_dU\_50/20\ range(2)/step(2)/MC(0)$ hat mit einem maximalen Fehler von 2,76 V das beste Modell trainiert. Wie in Tabelle 8 zu sehen ist, werden infolgedessen zwei weitere ähnliche Datensätze hinsichtlich eines Verbesserungspotentials untersucht. Auf ein Training von Transformer wurde in diesem Schritt wegen der schlechten Performance bei den LTO-Batterie Datensätzen verzichtet.

Tabelle 8 LFP-Batterie - MAXE von RNN und Transformer Modellen für verschiedene Balancing Kombinationen

Nr.	Undersampling	Oversampling	RNN	Transformer
1	$I_{mean}\_T_{mean}\_U_{mean}\_20/20$	-	4,10 V	nan
2	$I_{mean}\_T_{mean}\_dU\_50/20$	$range(2)/step(1)/MC(-5/5)$	4,41 V	nan
3	$I_{mean}\_T_{mean}\_U_{mean}\_50/20$	$range(2)/step(2)/MC(0)$	2,76 V	nan

Da im ersten Schritt die Modellgüte analog zur Datensatzgröße gestiegen ist, wird im zweiten Schritt die Datensatzgröße weiter erhöht. Zum einen wird das Bin Limit bei Datensatz Nr. 4 aus Tabelle 9 auf 50 erhöht. Zum anderen wird im Vergleich zu Datensatz Nr. 2 in Datensatz Nr. 5 die Bin Range auf 100 erhöht und ein einstufige Oversampling durchgeführt. Beide Datensätze erzeugen bessere Modelle, wobei der Datensatz Nr. 5 mit der Balancing Konfiguration $I_{mean}\_T_{mean}\_U_{mean}\_100/20\ range(2)/step(1)/MC(0)$ das beste Modell mit einem MAXE von 2,31 V erreicht. Der vielversprechendste Datensatz bildet zudem die Grundlage für das Training von Transformern. Wie in Abbildung 9 zu sehen ist, ist das beste Transformer Modell, getuned nach dem in Kapitel 2.3.5 vorgestellten Hyperparamtertuning-Algorithmus, auf einen MAXE von 2,66 V validiert und somit zwar besser als das bisher beste Transformer-Modell, jedoch deutlich schlechter als das beste RNN.

Tabelle 9 LFP-Batterie - MAXE von RNN und Transformer Modellen für weitere Balancing Kombinationen

Nr.	Undersampling	Oversampling	RNN	Transformer
4	$I_{mean}\_T_{mean}\_U_{mean}\_50/50$	$range(2)/step(2)/MC(0)$	2,52 V	2,49 V
5	$I_{mean}\_T_{mean}\_U_{mean}\_100/20$	$range(2)/step(1)/MC(0)$	2,31 V	2,66 V

5.4.2. Begründung für Wahl der Hyperparameter

Anschließend an die Wahl des Balancings müssen die Hyperparameter, wie in Kapitel 5.1.2 beschrieben, getuned werden. Die Tabelle 10 zeigt die zu tunenden Parameter mit deren Parameterräumen sowie das Ergebnis des Tunings.

Für das weitere Hyperparametertuning wird das vorläufig beste Modell genutzt. In diesem Fall stellt das Modell ein LSTM mit fünf hidden Units jeweils in den zwei hidden Layer und den zwei Dense Layern in Kombination mit einer Dropout Rate von 0.011 dar.

Tabelle 10 LFP-Batterie - Parameterräume für das Hyperparametertuning mit den jeweiligen Ergebnissen des Tunings

	Parameterraum	**Ergebnis des Tunings**
Optimizer	{AdaGrad, Adam, RMSprop, SGD}	Adam
Learning Rate	{0,0001 : 0,001}	0,0002
Batchnorm	{On, Off}	On

Um den am besten geeigneten Optimierer auszuwählen, werden vier gleiche Modelle trainiert und validiert. Das Ergebnis der Validierung ist in Abbildung 28 zu sehen. Der Optimierer AdaGrad konvergiert sehr langsam. Der Loss ist um eine Größenordnung größer und der MAXE ist minimal bei 13,2 V. Das SGD schneidet ebenfalls deutlich schlechter als die anderen beiden Optimierer ab, sowohl in Bezug auf den Loss als auch auf den MAXE. Adam und RMSprop zeigen eine ähnliche Performance. Der RMSprop Optimierer konvergiert schneller und der Adam Optimierer hat vergleichsweise den kleinsten MAXE von 2,46 V. Die Ergebnisse decken sich mit dem Vergleich der Optimierer in Kapitel 2.3.5.

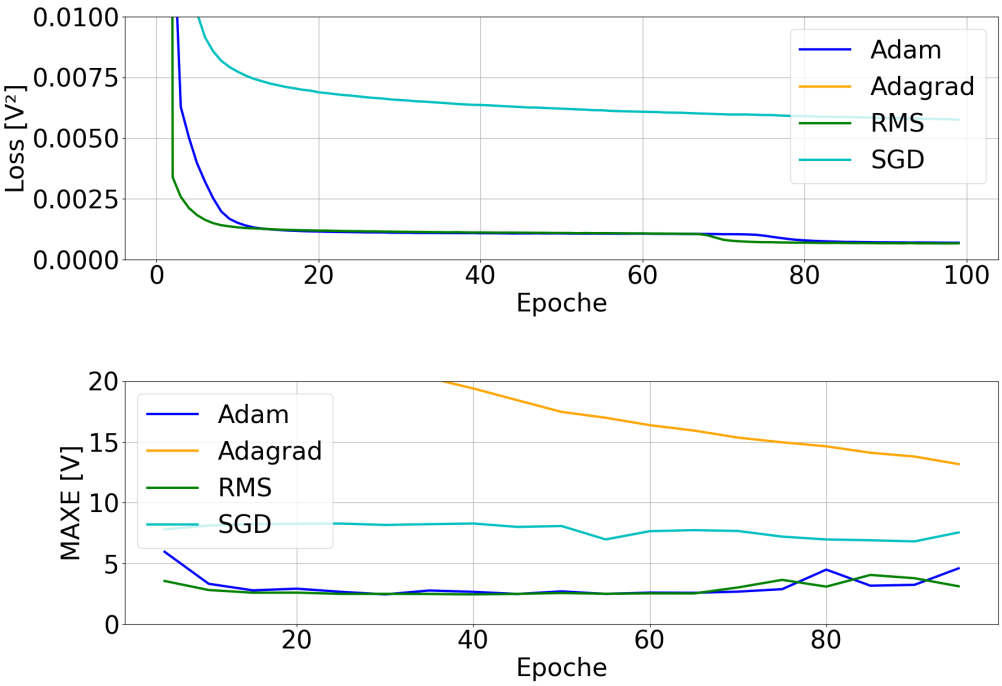


Abbildung 28 LFP-Batterie - Vergleich der Trainingsperformance verschiedener Optimierer

Ähnlich wie bei dem LTO-Batterie Datensatz ergab die Gegenüberstellung der Lernraten das Ergebnis, dass die Lernrate keinen großen Einfluss auf die Genauigkeit und die Konvergenz der Modelle hat. Im Bereich zwischen 0.0001 und 0.001 konnte in zehn Trials kein Overfitting festgestellt werden, jedoch ist eine positive Tendenz hin zu den kleineren Lernraten erkennbar. Dabei liegt das Optimum zwischen 0.0002 und 0.0003. Im Folgenden wird, wie bei der LTO, dieselbe Lernrate von 0.0002 verwendet.

Der Einfluss der Batchnormalisierung auf die Güte des Modells ist positiv, wie es bei der LTO-Batterie ebenso festzustellen war. Abbildung 29 zeigt den Vergleich zwischen einer aktiven und inaktiven Batchnormalisierung. Das Ergebnis zeigt, dass der Loss bei beiden Varianten ähnlich ist, jedoch der minimale MAXE bei der aktivierten Batchnormalisierung deutlich kleiner ist.

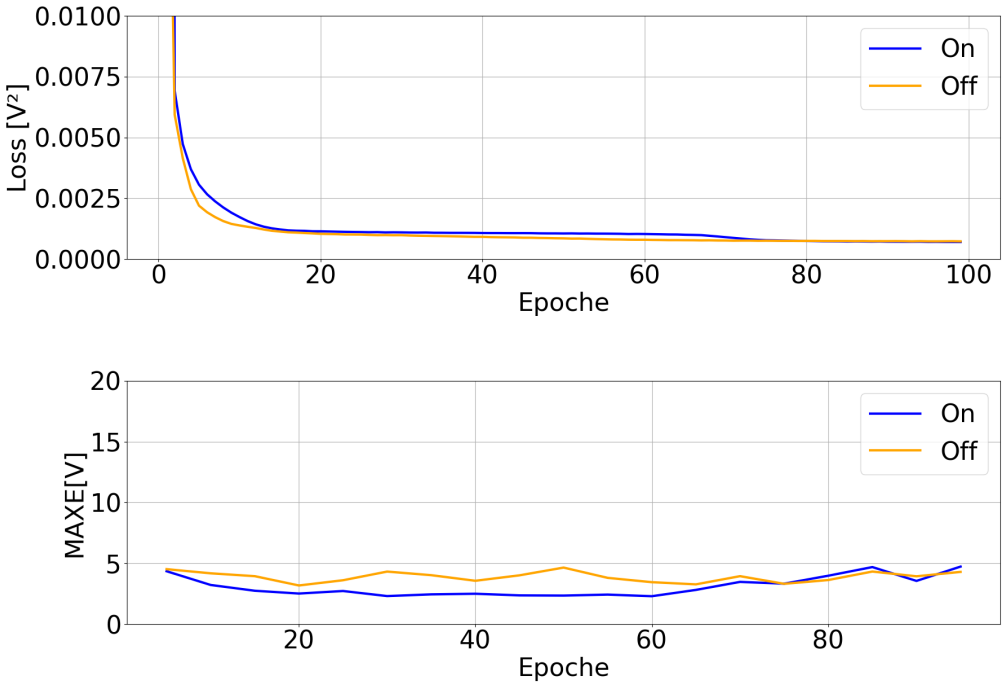


Abbildung 29 LFP-Batterie - Vergleich der Trainingsperformance zweier Modelle mit aktiver und inaktiver Batchnorm

5.4.3. Modellparametertuning

Das Hyperparametertuning der Modellparameter mit Optuna wird basierend auf den ausgewählten Datensets sowie den im vorherigen Kapitel bestimmten Parametern durchgeführt. Es wurden 15 Trials trainiert. Davon wurden vier Trials vorzeitig abgebrochen, da der Pruner diese Zweige als nicht aussichtsreich erkannt hat. Ein Overfitting konnte in keinem der Trials festgestellt werden. Allerdings zeigten sich, wie in Abbildung 30 zu sehen, deutliche Unterschiede im MAXE. Das Modell mit dem geringsten MAXE, ist ein LSTM Netz mit fünf hidden Units, zwei RNN Layern sowie zwei Dense Layern mit einer Dropout Rate von 0,11. Da dieses Modell nur fünf hidden Units hat gibt es wenige modellinterne Gewichte und Bias die eingestellt werden müssen. Hierdurch stellt sich das Optimum bereits in Epoche zehn ein.

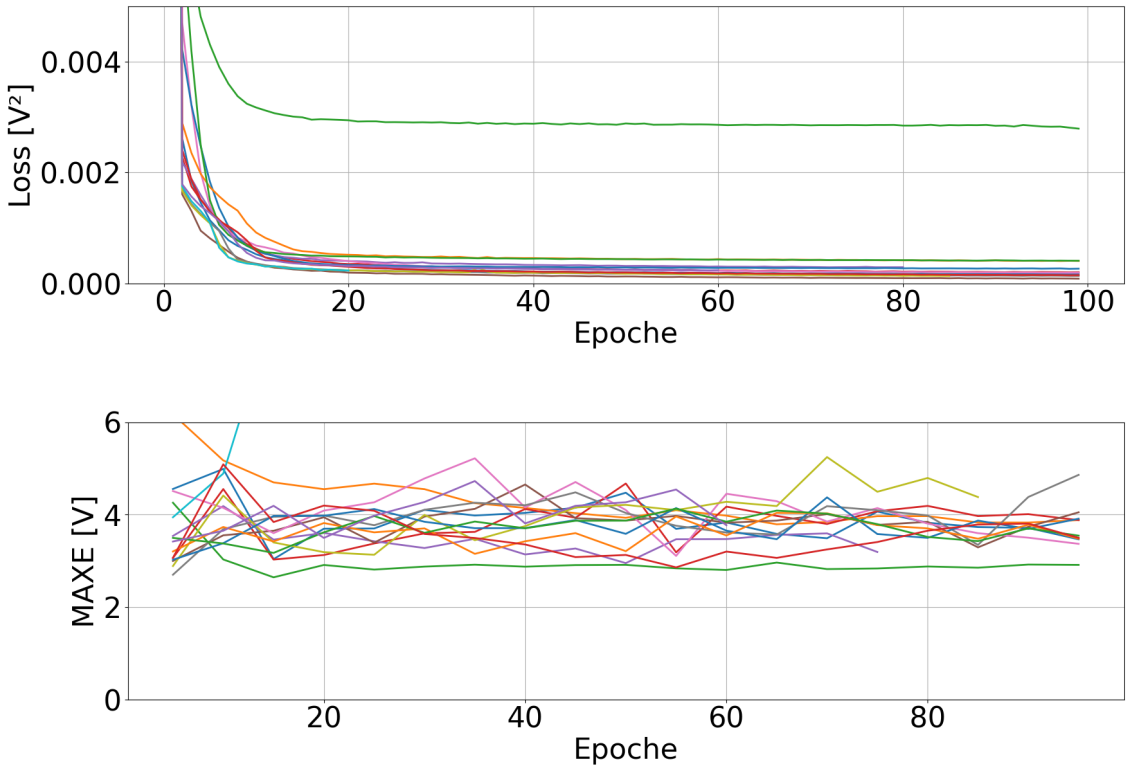


Abbildung 30 LFP-Batterie - Modellparametertuning in 15 Trials des Optuna-Algorithmus

Die randomisierte Initialisierung der Gewichte und Bias im NN führt zu unterschiedlichen Verläufen im Trainingsprogress, auch wenn gleiche Netze trainiert werden. Dementsprechend wurde das beste NN mit fünf verschiedenen Initialisierungen trainiert. Das Ergebnis der fünf Vergleichstrainings ist in Abbildung 31 dargestellt. Eine weitere Optimierung konnte in diesem Fall jedoch nicht erreicht werden.

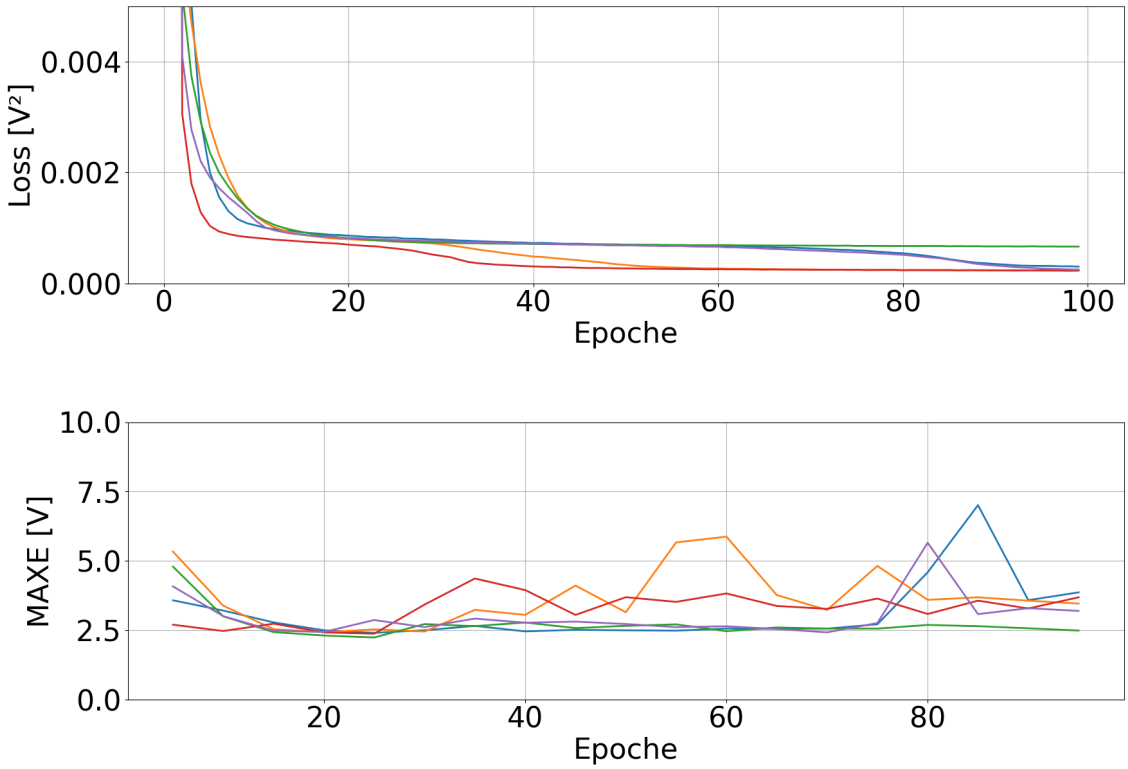


Abbildung 31 LFP-Batterie - Fünfmalige Wiederholung des Trainings

Abbildung 32 zeigt ein ähnliches Verhalten wie das Training von 1000 Epochen bei der LTO-Batterie. Es tritt weder Oversampling auf, noch kann der MAXE nach über 100 Epochen verbessert werden, womit auch in diesem Fall ein Training für 100 Epochen ausreichend ist.

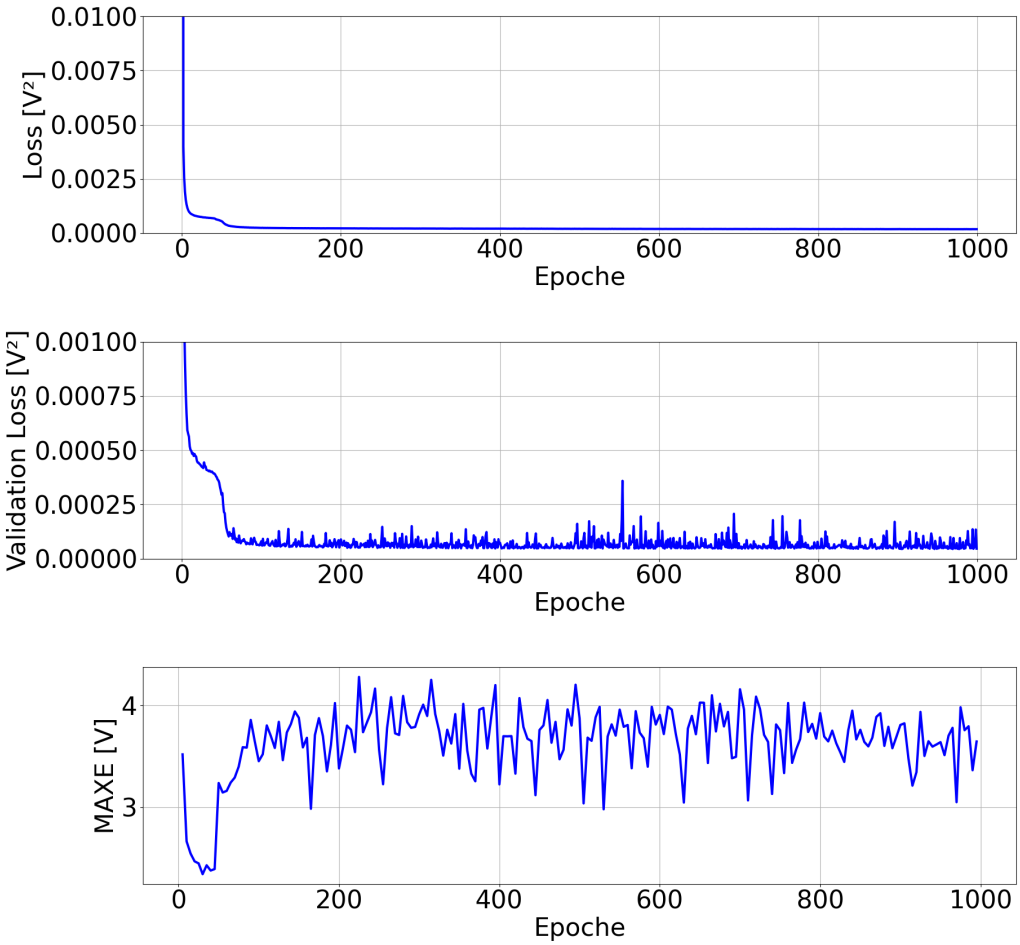


Abbildung 32 LFP-Batterie - Loss, Validation Loss und MAXE im Verlauf eines Trainings mit 1000 Epochen

Die Tabelle 11 zeigt die Fehlermetriken des LFP-Modells mit dem geringsten MAXE.

Tabelle 11 LFP-Batterie - Fehlermetriken des finalen LFP-Batterie Modells

Balancing Konfiguration	
Nr.	5
Undersampling	$I_{mean}\_T_{mean}\_U_{mean}\_100/20$
Oversampling	$range(2)/step(1)/MC(0)$
Netzwerk Konfiguration	
Netzwerktyp	LSTM
Anzahl hidden Units	5
Anzahl RNN Layer	2
Anzahl Dense Layer	2
Dropout Rate	0,11
MAE	0,34 V
MAXE	2,24 V

5.5. Zwischenfazit

Das Hyperparametertuning hat verdeutlicht, welche Parameter in welchem Umfang die Güte des Modells beeinflussen. Zusätzlich konnte gezeigt werden, dass einige Parameter nicht für jeden Datensatz einzeln getuned werden müssen, sondern im Rahmen dieser Anwendung auf einen bestimmten Wert festgelegt werden können.

Tabelle 12 Subjektive Einschätzung bezüglich des Einflusses der Parameter auf die Güte des Modells

Parameter	**Direkter Einfluss auf Güte des Modells**
Undersampling (Bin Range, Bin Limit)	+++
Oversampling (Minoritätsklassen, Range, Schrittweite)	+++
Lernrate	+
Optimierer	++
Batchnorm	+
Netztyp	++
Anzahl hidden Units	+
Anzahl RNN Layer	+
Anzahl Dense Layer	+
Dropout Rate	+

In Tabelle 12 sind die Parameter und deren Einfluss auf die Güte des Modells aufgetragen. Die Datenvorverarbeitung mit den Algorithmen zum Under- und Oversampling hat den größten Einfluss, sowohl auf die Güte als auch auf die Konvergenz der zu trainierenden Modelle. Eine generelle Empfehlung zu den Parametern kann dabei nicht gegeben werden. Es besteht eine große Abhängigkeit von der Datenverteilung und -größenordnung der Rohdaten. Des Weiteren hat die Wahl des Optimierers einen gewichtigen Einfluss auf das Ergebnis. Wobei die Untersuchungen zeigen, dass der Adam Algorithmus am geeignetsten ist. Die Hyperparameter Lernrate und Batchnormalisierung haben einen geringeren Einfluss, zeigen aber die besten Ergebnisse mit aktivierter Batchnormalisierung und einer Lernrate von 0.0002. Der Einfluss der modellbezogenen Parameter kann nicht für jeden Hyperparameter einzeln bewertet werden, da diese immer im Zusammenhang getuned werden. Pauschale Aussagen sind aus diesem Grund nicht zu treffen. Der Modellierungsansatz mit Transformern ist in dieser Anwendung nicht zu empfehlen, da sich die Performance in jedem Vergleich mit RNNs als schlechter zeigt.

6. Verifikation und Validierung

Das spezifizierte Konzept, eine Pipeline zur Datenvorverarbeitung und Modellierung zu erstellen und dessen Umsetzung, muss verifiziert werden. Der untere Bereich des V-Modells in Abbildung 33 zeigt die theoretische Detaillierung der Spezifikation bis hin zur Integration sowie den Test zur Verifikation. Am Ende dieses Prozesses wird die Anforderung mit Anwendungsszenarien im finalen Abnahmetest validiert.

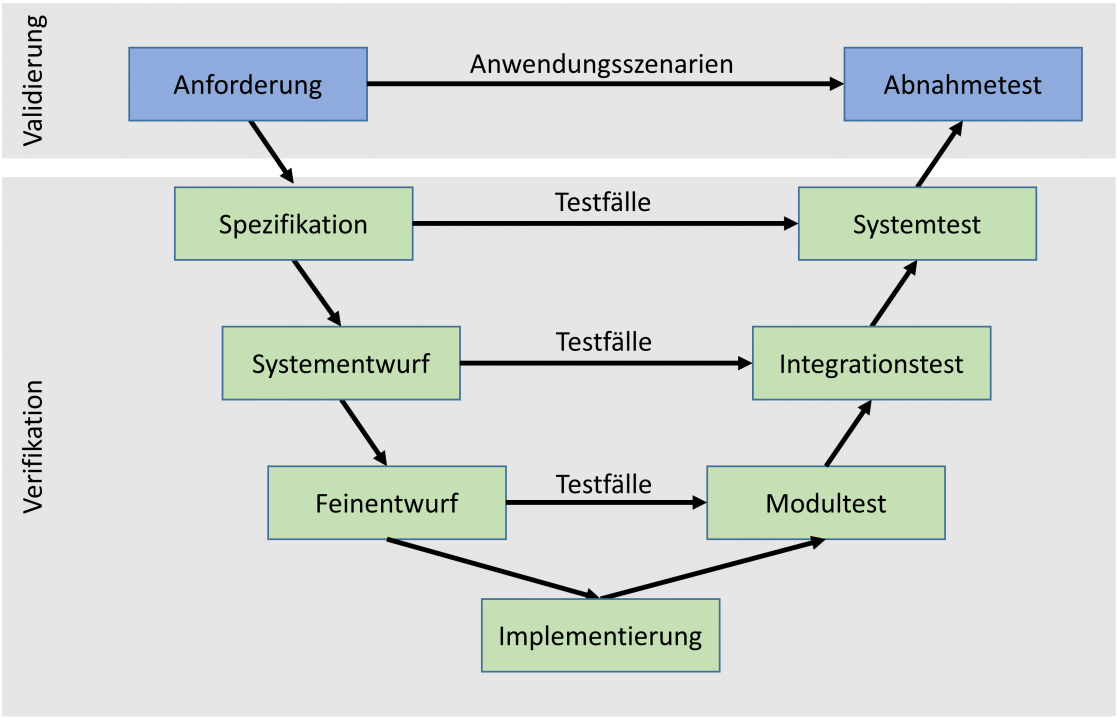


Abbildung 33 Validierung und Verifizierung im Rahmen des V-Modells mit den jeweiligen Entwicklungsstufen

6.1. Verifikation

Die Verifikation beginnt mit der Spezifikation des Systems. Das Modell soll das Spannungsverhalten der Batterie wiedergeben, auf der Basis von Deep-Learning Algorithmen aufgebaut werden, nur auf physikalisch messbare Eingangsgrößen zurückgreifen und aus Rohdaten erstellt werden können.

Der gesamte Systemtest wurde in Kapitel 5 beschrieben, wobei die einzelnen Bestandteile der Spezifikation anhand von zwei unterschiedlichen Batterie-Typen verifiziert werden konnten.

Der Systementwurf beinhaltet die Auswahl der Eingangsfeatures in Kapitel 4.1, die Datenvorverarbeitungspipeline in Kapitel 4.2, das Hyperparametertuning in Kapitel 2.3.5 sowie das Training der Transformer Modelle und der RNN Modelle in den Kapiteln 5.3 und 5.4. Dabei wird auf eine manuelle, nicht formale Verifikation durch dynamische Tests zurückgegriffen.

Teile aus dem Systementwurf wurden im Feinentwurf detaillierter formuliert. Dazu zählt das Balancing der Datenvorverarbeitungspipeline, insbesondere das in Kapitel 4.2.3 beschriebene Oversampling und entworfene MFU. Die implementierten Algorithmen wurden während dem Entwicklungsprozess ständig Modultests unterzogen.

6.2. Validierung

Während des Trainings und der Inferenz findet die Vorhersage nur innerhalb einer Sequenz statt, wobei die meisten Anwendungen eine Vorhersage über einen längeren Zeitraum erfordern. Da in dem vorgeschlagenen Ansatz auf den Teacher-Forcing-Algorithmus verzichtet wird, untersucht dieser Abschnitt die Genauigkeit der vorhergesagten Spannung auf die nächsten Zeitschritte. Kleine Fehler zu Beginn könnten aufgrund der fehlenden Korrektur zur Ground Truth, zu einer Instabilität der Vorhersage führen.

Die trainierten Modelle müssen hinsichtlich der Fehlermetriken validiert werden, um eine Aussage über die Verwendbarkeit dieser Modelle treffen zu können. Mit dieser ergebnisbezogenen Validierung wird die Übereinstimmung der Ergebnisse des Modells mit dem realen Verhalten der Batterie geprüft.

6.2.1. Messungen der HiL-Prüfeinrichtung

Für die Validierung werden Messungen mit den Batteriesamples an einem Hardware-in-the-Loop Prüfstand aufgenommen. Ein Vorteil gegenüber den zufällig entstehenden CAN-Traces aus Versuchsfahrzeugen ist, dass die Umweltbedingungen und Batteriekon-

ditionen manuell eingestellt werden können. Somit kann sichergestellt werden, dass die Validierung in einem breiten Spektrum stattfindet. Die Temperaturniveaus von

$$T \in \{-25°C, 0°C, 25°C, 50°C\} \tag{6.1}$$

wurden mit niedrigen, mittleren und hohen Ladezuständen kombiniert, resultierend in zwölf Validierungsmessungen je Batterie. Auf eine kontinuierliche Temperatur- und Ladezustandsverteilung wurde verzichtet, um die Datenmenge und damit die Rechenzeit klein zu halten. Der Erkenntnisgewinn aus Messungen zwischen den genannten Stützstellen ist zu vernachlässigen, da keine weiteren schwer modellierbaren Extremsituationen auftreten. Eine strikte Einhaltung der Temperaturstufen ist ebenso zu vernachlässigen, da sie sich negativ auf die Diversifizierung der Datenverteilung des Validation Sets auswirken würde.

Abbildung 34 zeigt die Streudiagramme der zwölf Validierungsmessungen in Bezug auf die Temperatur, die Spannung und den Strom für die LTO-Batterie. Die Temperaturniveaus sind in den ersten beiden Diagrammen durch die jeweils vier horizontalen Streifen deutlich zu erkennen. Charakteristisch für die Polarisationsüberspannungen bei tieferen Temperaturen ist, dass die Ströme absolut kleiner sind, sich die Spannung jedoch über einen großen Bereich erstreckt. Am anderen Ende der Temperaturskala sind dementsprechend die Ströme tendenziell größer und die Spannungen weniger stark volatil. Die zu erkennende Diagonalität im Streudiagramm Spannung/Strom ist durch die Laderichtung des Stromes und deren Auswirkung auf die Spannung zu begründen. Die Spannung erstreckt sich im Validation Set in einem Bereich zwischen 39,8 V und 51,1 V, wodurch die Batterie bis nahe an die Betriebsgrenzen validiert werden kann. Die Ströme sind maximal 230 A in Entladerichtung und 160 A in Laderichtung, dies entspricht einer C-Rate von 21 C respektiv 15 C.

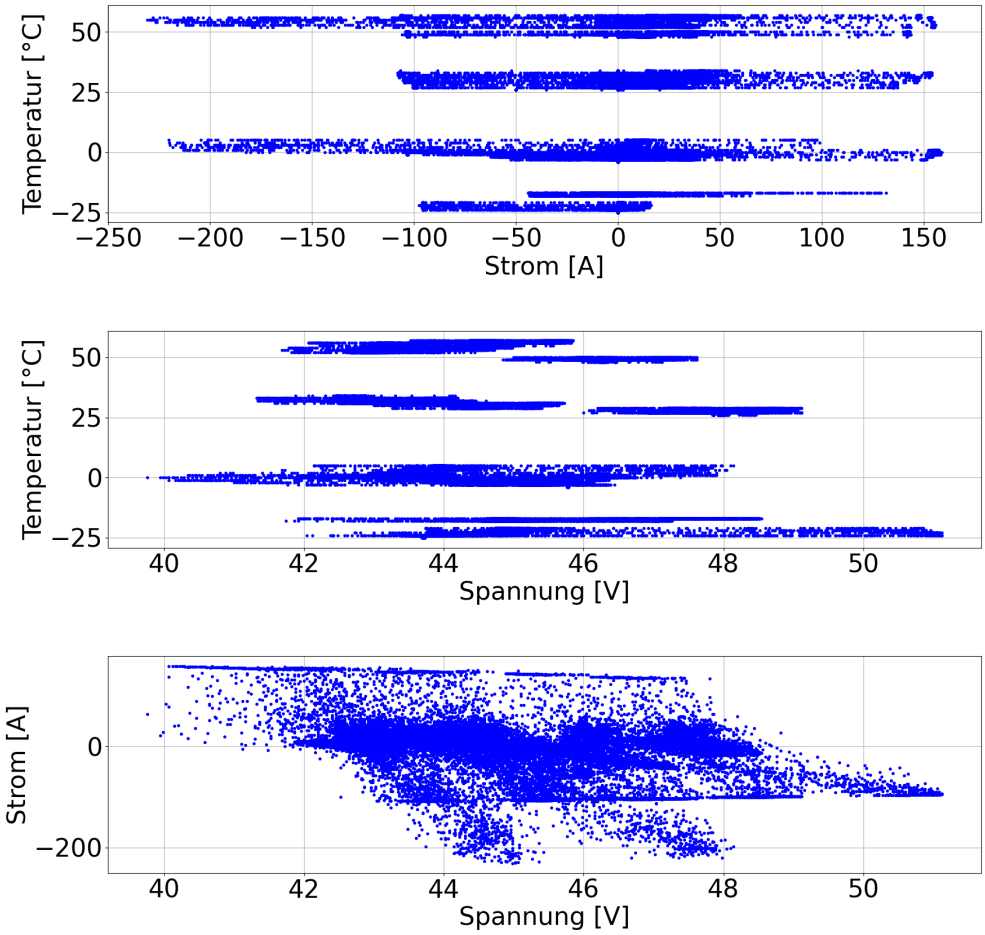


Abbildung 34 LTO-Batterie - Streudiagramm des Validation Set

Die Datenverteilung von Temperatur, Strom und Spannung im Validation Set der LFP-Batterie ist in Abbildung 35 dargestellt. In Analogie zu der LTO-Batterie ist zu erkennen, dass die Stromrange aufgrund der temperaturabhängigen Polarisationswiderstände mit steigender Temperatur zunimmt und die Spannungsrange zugleich abnimmt. Die Temperaturschritte sind auf dieselben Basiswerte aufgeteilt, wobei eine strikte Einhaltung der Temperatur vernachlässigt wurde. Die Ströme sind begrenzt auf 525 A im Ladefall und 223 A in der Entladung, was einer maximalen C-Rate von 26 C entspricht. Im Validation Set sind Spannungen aus dem gesamten Betriebsbereich der Batterie enthalten.

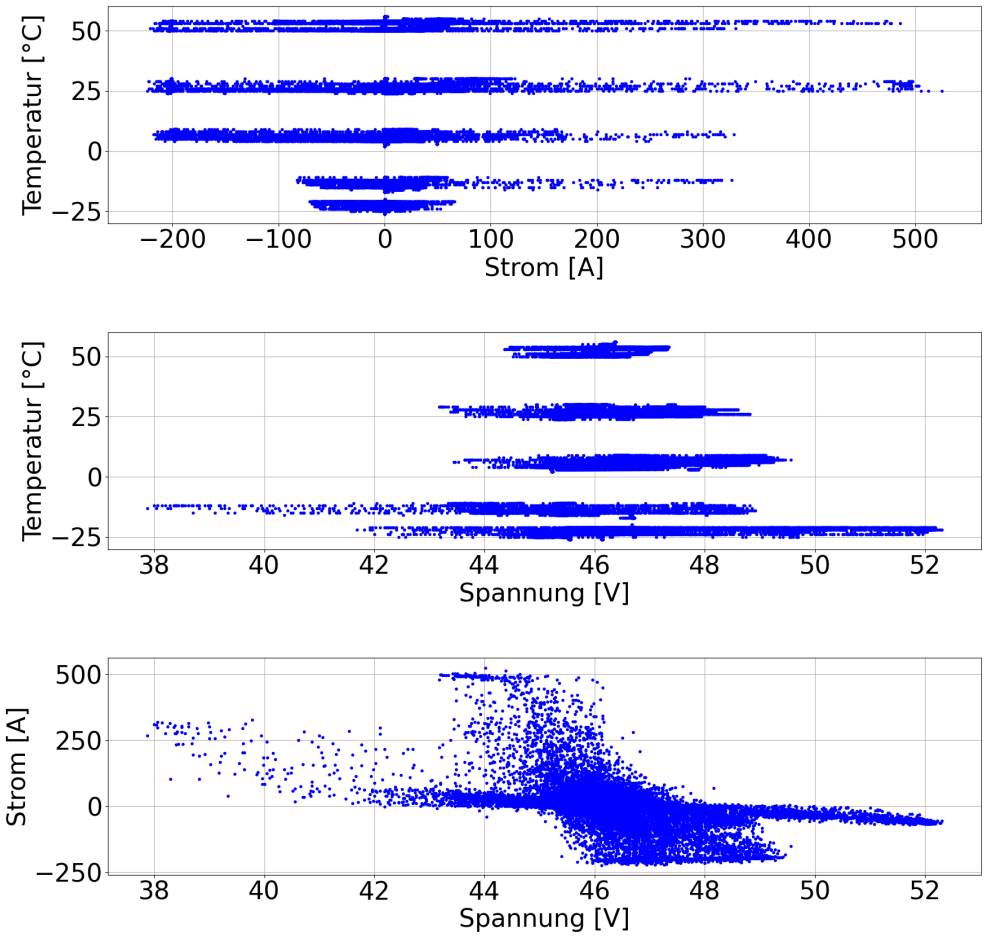


Abbildung 35 LFP-Batterie - Streudiagramm des Validation Set

Das für das Einmessen des Validation Set verwendete Stromprofil ist in Abbildung 36 beispielhaft für die LTO-Batterie bei einem Temperaturniveau von 0 °C und einem mittleren Ladezustand dargestellt. Das Basis-Stromprofil stammt aus einer realen, kundenorientierten Testfahrzeugmessung unter realen Umweltbedingungen. Das Ur-Profil wird dann je nach Temperatur und Ladezustand so skaliert, dass die Spannungsgrenzen der Batterie zwar möglichst erreicht, jedoch nicht überschritten werden. Somit kann in jeder Kombination der größtmögliche Spannungsbereich abgedeckt werden. Das reale Profil enthält neben großen, kurzzeitigen Volatilitäten sowohl Konstantstromphasen als auch Ruhephasen. Das Profil beinhaltet neben Ladephasen (negative Ströme) auch Entladephasen (positive Ströme). Die Länge des Profils ist auf 480 s begrenzt, um einerseits die Datenmenge gering zu halten und andererseits das Modell über einen ausreichend langen Zeitraum validieren zu können.

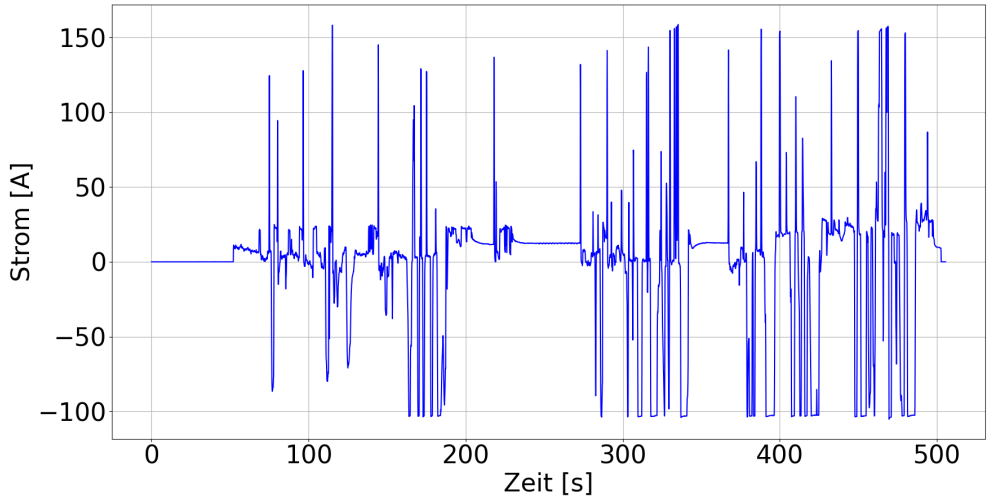


Abbildung 36 Stromprofil bei 0 °C und mittlerem Ladezustand

6.2.2. Validierungsergebnisse

Die trainierten NN werden mit den Validierungsmessungen beaufschlagt, um somit eine Aussage über die Güte des Modelles zu erlangen. Daraus folgt zum einen ein Erkenntnisgewinn bezüglich der Stärken und Schwächen des Modells, zum anderen ist die Vergleichbarkeit mit anderen Modelltypen gegeben.

Validierung LTO-Batterie

Die Spannungsprädiktionen werden für alle zwölf Validierungsmessungen mit den gemessenen Spannungen, der GT, verglichen. Dabei wird in erster Linie auf den Error, der Differenz aus Prädiktion und GT geachtet und zusätzlich der qualitative Verlauf der beiden Spannungen bewertet.

Abbildung 37 zeigt die Prädiktion auf die Messung bei -18℃ und mittlerem SOC. Der MAXE liegt bei knapp über 1 V und tritt zu einer Phase mit sehr hohen Belastungen auf. Das exposure Bias Problem, bei welchem die fehlende direkte Rückkopplung der GT zu einem Drift im Fehler führt, kann in diesem Fall selbst über acht Minuten Prädiktionshorizont nicht festgestellt werden. Die Feedback Prozedur der Spannung zum Eingangsparameter U_{trend}, als Methode zur Kompensation des nicht einsetzbaren Teacher Forcing Algorithmus, gibt der Prädiktion Stabilität. Der leicht erkennbare Bias nach ca. 300 s ist nicht auf die Berechnung des U_{trend} zurückzuführen, sondern auf die in diesem Profil auftretenden Volatilitäten.

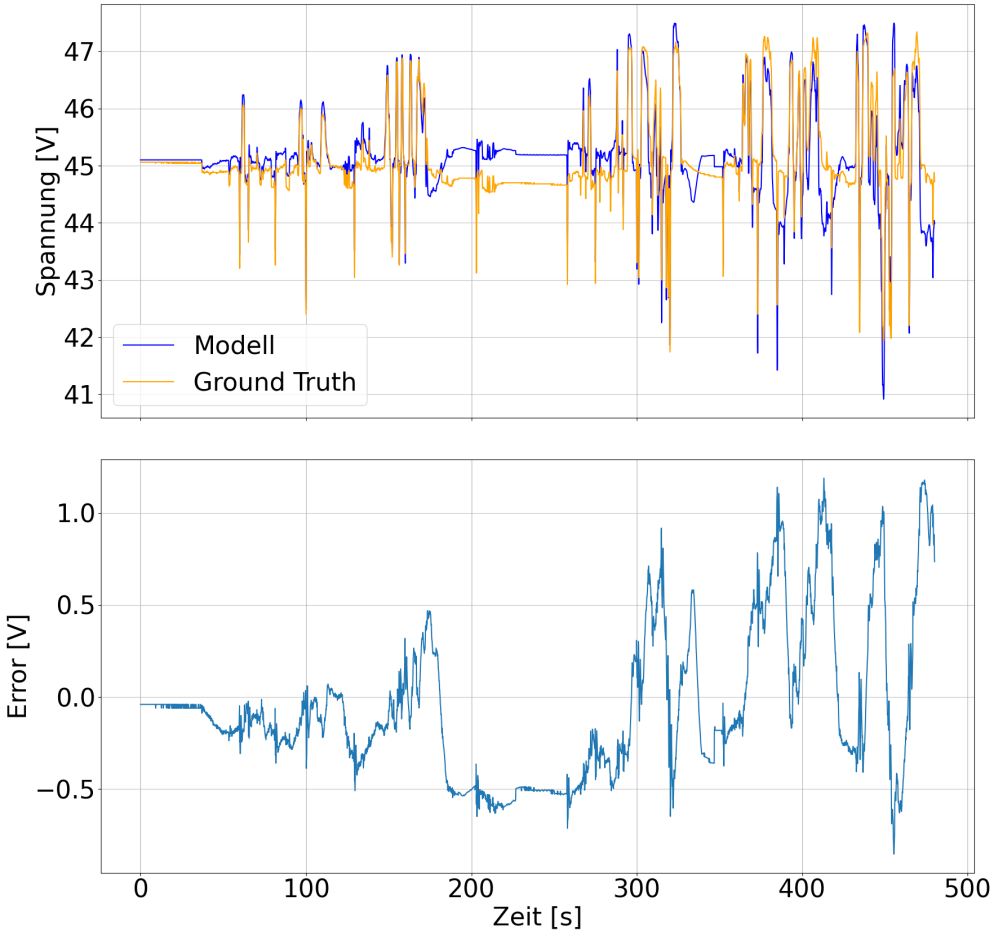


Abbildung 37 LTO-Batterie - Validierung des Modells mit dem Profil „Val_LTO_5“ bei mittlerem Ladezustand und einer Temperatur von durchschnittlich -18 °C

Wie in Abbildung 38 zu beobachten, tritt im Gegensatz zu der Prädiktion von Profil Val_-LTO_5 tritt im Profil Val_LTO_2 ein Error-Drift auf, der nicht auf das exposure Bias Problem zurückzuführen ist. Die Ursache in diesem Fall ist die Änderung des Ladezustands über die untersuchte Zeitspanne von acht Minuten hinweg. Bei niedrigen Ladezuständen wurden die Stromprofile so angepasst, dass die Entladeströme herunterskaliert werden, jedoch die Ladeströme gleichbleiben. Das führt zu einem Nettoenergieeintrag der Batterie und einer damit verbundenen Erhöhung der OCV. Eine Änderung des Ladezustands erfolgt meist über einen längeren Zeitraum, besonders in hochvolatilen Stromverläufen mit wenigen hohen Konstantstromphasen im Automotive Energiebordnetz. Somit kann das NN den Zusammenhang zwischen Ladezustand und OCV bzw. Spannung nicht erlernen, weil zu wenige Daten mit signifikanten Ladezustandsänderungen innerhalb der Sequenzlänge vorliegen. Der U_{trend} Updateprozess mit der indirekten Rückwirkung von der OCV auf den Input des Netzes wirkt dem Fehlerdrift entgegen und wird mit einer Updatezeit von 70 s zu selten durchgeführt, um den Trend umzukehren. Der maximale absolute Fehler im gesamten Validierungsset ist die Folge dieses Fehlerdrifts in Kombination mit den hohen Belastungen im Stromprofil.

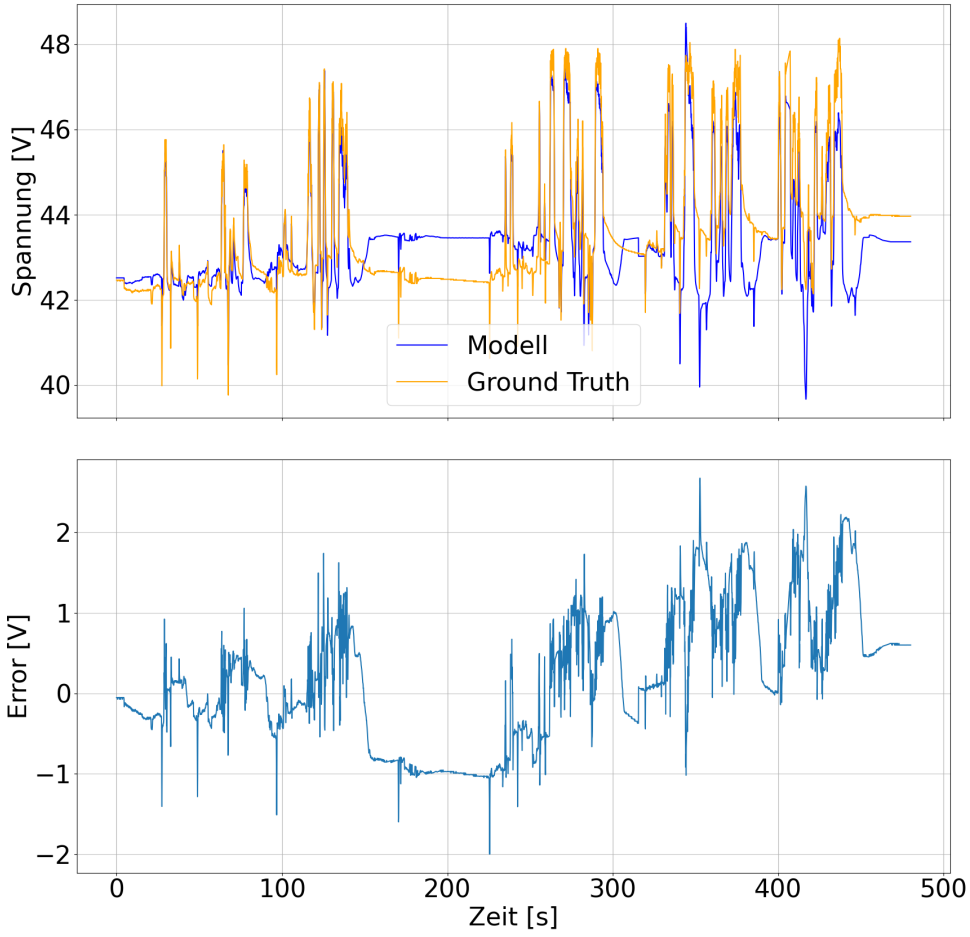


Abbildung 38 LTO-Batterie - Validierung des Modells mit dem Profil „Val_LTO_2“ bei niedrigem Ladezustand und einer Temperatur von durchschnittlich 1 ℃

Der maximale absolute Fehler von 2,77 V tritt in der Messung bei niedrigem SOC und -23 °C, wie in Abbildung 39 dargestellt, auf. Eine starke Temperaturabhängigkeit im Fehler kann nicht festgestellt werden, da das Validierungsprofil mit dem größten MAXE bei der gleichen Temperatur wie das Profil mit dem kleinsten MAXE gemessen wurde. Das Verhalten tritt auf, obwohl tiefe Temperaturen für die Spannungsprädiktion deutlich herausfordernder sind, da hier die Polarisationsüberspannungen deutlich höher liegen.

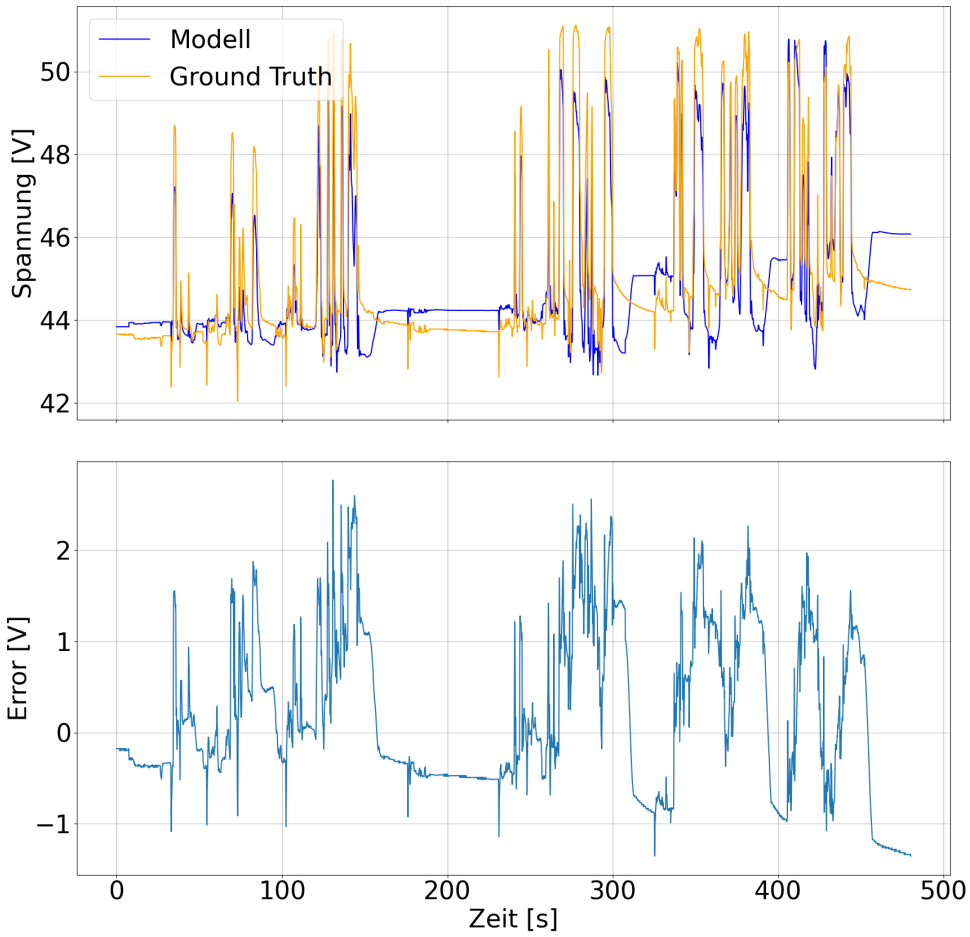


Abbildung 39 LTO-Batterie - Validierung des Modells mit dem Profil „Val_LTO_1“ bei niedrigem Ladezustand und einer Temperatur von durchschnittlich -23 °C

Die Tabelle 13 zeigt die Fehlermetriken der Validierung des LTO-Modells. Über das gesamte Validation Set erreicht das LTO-Batterie Modell einen MSE von 0,41 V², einen MAE von 0,47 V und einen MAXE von 2,77 V. Bezogen auf die Nominalspannung ergibt sich ein maximaler relativer Fehler von 5,77 % und ein durchschnittlicher relativer Fehler von 1,00 %.

Tabelle 13 LTO-Batterie - Fehlermetriken Validierung

Profil	**Ladezustand**	**Mittelwert Spannung**	**Mittelwert Temperatur**	**MSE**	**MAE**	**MAXE**
Val_LTO_1	Niedrig	44,8 V	-23 °C	0,67 V²	0,63 V	2,77 V
Val_LTO_2	Niedrig	43,3 V	1 °C	0,59 V²	0,58 V	2,68 V
Val_LTO_3	Niedrig	42,9 V	32 °C	0,28 V²	0,39 V	1,59 V
Val_LTO_4	Niedrig	43,1 V	53 °C	0,59 V²	0,54 V	2,22 V
Val_LTO_5	Mittel	45,1 V	-18 °C	0,15 V²	0,30 V	1,19 V
Val_LTO_6	Mittel	44,0 V	0 °C	0,36 V²	0,44 V	2,57 V
Val_LTO_7	Mittel	44,4 V	30 °C	0,55 V²	0,62 V	1,84 V
Val_LTO_8	Mittel	44,7 V	56 °C	0,26 V²	0,37 V	1,39 V
Val_LTO_9	Hoch	47,6 V	-17 °C	0,23 V²	0,35 V	2,17 V
Val_LTO_10	Hoch	44,9 V	-3 °C	0,54 V²	0,59 V	2,64 V
Val_LTO_11	Hoch	47,6 V	27 °C	0,23 V²	0,35 V	1,56 V
Val_LTO_12	Hoch	46,3 V	49 °C	0,46 V²	0,51 V	1,93 V
Mittelwert				0,41 V²	0,47 V	2,05 V
Maximum				0,67 V²	0,63 V	2,77 V

Validierung LFP-Batterie

In Abbildung 40 ist die Validierungsmessung „Val_LFP_8“ zusammen mit der Prädiktion sowie dessen Fehler bei einer Temperatur von 53 °C und mittlerem Ladezustand dargestellt. Der maximale Fehler steigt bei einem MAE von 0,11 V nicht über 0,6 V. Im mittleren Ladezustandsbereich ist die OCV-Kurve der LFP-Batterie im Allgemeinen flach und somit annähernd konstant. Eine Änderung des SOCs während der Validierungsmessung hat somit nahezu keinen Einfluss auf die Ruhespannung, was durch den fehlenden Drift im Error verdeutlicht wird. Die langsame Spannungsänderung kann durch den Updateprozess von U_{trend} aufgefangen werden. Die prädizierte Spannung ist im gleitenden Mittelwert konstant kleiner als die GT, was auf eine initial falsche OCV Schätzung oder Polaristationsspannungsschätzung zurückzuführen ist. Die größten Volatilitäten im Error treten im Bereich mit hoch volatilen Stromkurven auf.

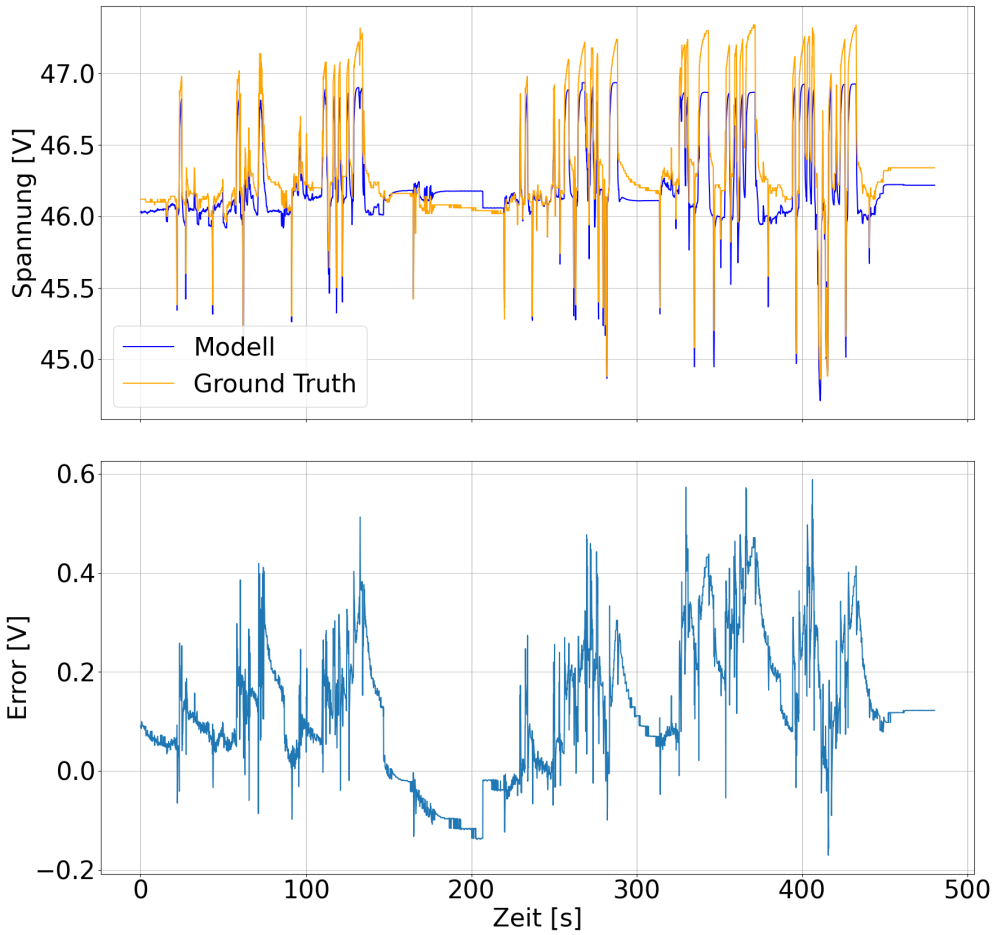


Abbildung 40 LFP-Batterie - Validierung des Modells mit dem Profil „Val_LFP_8“ bei mittlerem Ladezustand und einer Temperatur von durchschnittlich 53 °C

Der im Validation Set maximale absolute Fehler von 2,24 V ist in einer Messung mit extremen Bedingungen aufgetreten. Die in Abbildung 41 dargestellte Messung ist mit einer auf -23 °C temperierten und niedrigem Ladezustand konditionierten Batterie durchgeführt worden. Der maximale Fehler tritt bei Spannungsspitzen in einem Bereich von über 52 V auf, welcher nur selten im Trainingsdatensatz vorhanden ist. Zudem ist die Spannungsänderung zwischen OCV und der Spannungsspitze sehr groß. Hohe Ströme in Kombination mit tiefen Temperaturen stellen schwer vorhersagbare Bedingungen in der Batteriemodellierung dar. Der hohe MAE von 0,59 V ist auf diese Voraussetzungen zurückzuführen, kann jedoch aufgrund der großen Spannungsrange zwischen ca. 42 V und 52 V relativiert werden. Begünstigt durch die flache OCV-Kurve von LFP-Batterien ist in dieser Prädiktion der Einfluss von Ladezustandsänderungen mit einem Errordrift nicht festzustellen.

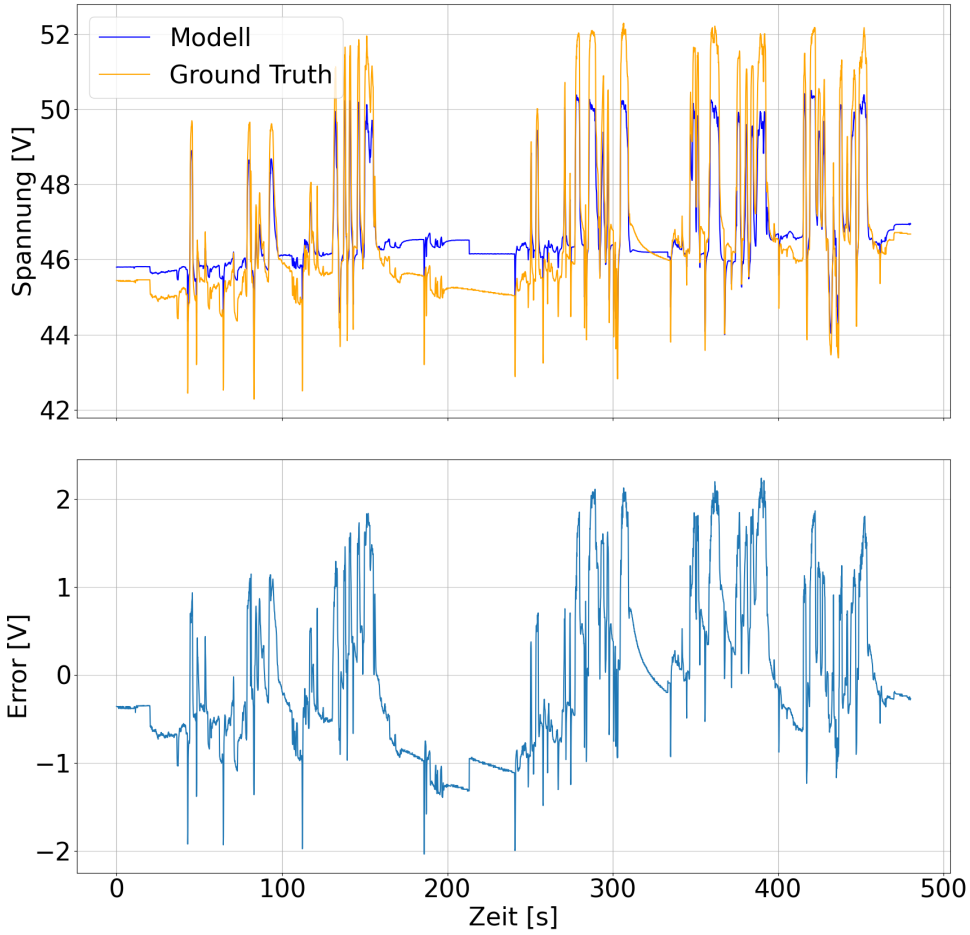


Abbildung 41 LFP-Batterie - Validierung des Modells mit dem Profil „Val_LFP_1“ bei niedrigem Ladezustand und einer Temperatur von durchschnittlich –23 °C

Wie in Tabelle 14 zu sehen, erreicht das LFP-Modell über das gesamte Validation Set einen MSE von 0,21 V², einen MAE von 0,34 V und einen MAXE von 2,24 V. Bezogen auf die Nominalspannung ergibt sich ein maximaler relativer Fehler von 4,67 % und ein durchschnittlicher relativer Fehler von 0,71 %.

Tabelle 14 LFP-Batterie - Fehlermetriken Validierung

Name	**Ladezustand**	**Mittelwert Spannung**	**Mittelwert Temperatur**	**MSE**	**MAE**	**MAXE**
Val_LFP_1	Niedrig	46,3 V	-23 °C	0,58 V²	0,59 V	2,24 V
Val_LFP_2	Niedrig	45,6 V	4 °C	0,39 V²	0,56 V	1,25 V
Val_LFP_3	Niedrig	45,9 V	25 °C	0,05 V²	0,19 V	0,74 V
Val_LFP_4	Niedrig	45,6 V	51 °C	0,25 V²	0,46 V	0,89 V
Val_LFP_5	Mittel	46,2 V	-23 °C	0,21 V²	0,32 V	1,54 V
Val_LFP_6	Mittel	46,6 V	6 °C	0,17 V²	0,31 V	1,34 V
Val_LFP_7	Mittel	46,1 V	26 °C	0,06 V²	0,20 V	0,90 V
Val_LFP_8	Mittel	46,3 V	53 °C	0,02 V²	0,11 V	0,59 V
Val_LFP_9	Hoch	45,8 V	-15 °C	0,26 V²	0,37 V	2,19 V
Val_LFP_10	Hoch	46,1 V	5 °C	0,19 V²	0,30 V	1,66 V
Val_LFP_11	Hoch	46,5 V	26 °C	0,37 V²	0,53 V	1,91 V
Val_LFP_12	Hoch	46,3 V	54 °C	0,03 V²	0,17 V	0,41 V
Mittelwert				0,21 V²	0,34 V	1,30 V
Maximum				0,58 V²	0,59 V	2,24 V

Erkenntnisse der Validierung

Im Folgenden wird die Interpretation der Validierung in Bezug auf den vorgeschlagenen Prozess der Modellierung mit NN im Allgemeinen sowie auf die beiden erstellten Modelle diskutiert. Diese Erkenntnisse sind anhand der Validierung für die Modellierung mit NN im vorgeschlagenen Prozess abzuleiten:

- Der Hyperparametertuningprozess ist mithilfe von performanten Rechnern und richtig dimensionierten Datensätzen in einer annehmbaren Zeit umsetzbar.
- Die Modellierung der Spannung ist mit dem vorgeschlagenen U_{trend} Updateprozess über längere Zeithorizonte mit akzeptablen Fehlern möglich. Das exposure Bias Problem tritt bei keiner Validierung auf.
- Die Batterien können in großen Temperatur-, Strom- und Ladezustandsbereichen modelliert werden. Die Kombination von extremen Zuständen wie beispielsweise hohe Ladeströme bei sehr niedrigen Temperaturen und hohem Ladezustand sind in die Validierung inkludiert.
- Die elektrochemischen Effekte in der Batteriezelle führen zu einer exponentiellen Spannungskurve, wohingegen die vorhergesagte Spannung eckiger bzw. sprunghafter ist. Der qualitative Verlauf der Spannung kann trotzdem nachgestellt werden.
- Der Optimierungsalgorithmus ist darauf ausgerichtet, den maximalen Fehler im Validation Set zu reduzieren. Dies führt zu Differenzen in den Verläufen zwischen der Spannungsvorhersage und der GT. Eine Optimierung auf der Basis des MAE würde dazu führen, dass die Prädiktion häufiger genauer, dafür der maximale Fehler größer ist. Die Wahl der Optimierungsmetrik ist an die jeweilige Anwendung entsprechend anzupassen.
- Die Polarisationsüberspannungen sind aufgrund der elektrochemischen Ursprünge stark nichtlinear, was zu den größten Fehlern im Bereich großer Belastungen führt. An diesem Problem leiden alle Modellierungstypen, wobei in der hier vorliegenden Arbeit der maximale Fehler durch die MAXE Optimierung minimiert wurde.
- Aufgrund von Ungleichmäßigkeiten in der Produktion und in der Alterung zeigen unterschiedliche Batterien vom selben Typ ein ungleiches Verhalten. Die Trainingsdaten entstammen vielen verschiedenen Batterien desselben Typs. Die damit verbundenen Unterschiede machen einerseits das Modell robuster, andererseits können einzelne Batterien stärkere Abweichungen zur Gesamtheit und damit zum Modell aufweisen. Der Offset im Fehler am Anfang der Messungen bezieht sich mitunter auch auf diesen Zusammenhang.
- Trotz der Unterschiede in der Größe der NN, beeinflusst durch die Anzahl der hidden Units und hidden Layers, kann von einer pauschalisierten Aussage bezüglich der Korrelation zwischen Netzgröße und Zellchemie abgesehen werden.

In Tabelle 15 sind die Fehlermetriken der beiden Modelle dargestellt, wobei das LFP-Batterie Modell in allen Metriken besser abschneidet als das LTO-Batterie Modell. Die Ursache für den Unterschied kann sowohl durch die Anzahl und Verteilung der Daten als auch durch das elektrochemische Verhalten der Zellchemien begründet werden.

Tabelle 15 Fehlermetriken der finalen Batterie Modelle über das gesamte Validation Set

Modell	**Mittelwert Spannung [V]**	**Mittelwert Temperatur [°C]**	**MSE [V²]**	**MAE [V]**	**MAXE [V]**
LTO-Batterie	44,9	16	0,41	0,47	2,77
LFP-Batterie	46,1	16	0,21	0,34	2,24

Das Modell der LTO-Batterie unterscheidet sich von dem LFP-Batterie Modell in zwei Punkten, welche durch das spezielle elektrochemische Verhalten der Zelle zu erklären sind.

Zum einen führt eine schnelle Ladezustandsänderung innerhalb der Validierungsmessung zu einem Fehlerdrift beim LTO-Batterie Modell. Dieses Verhalten ist darauf zurückzuführen, dass der U_{trend} Updateprozess einer Spannungsänderung aufgrund einer Ladezustandsänderung langsamer ist und somit die prädizierte Spannung im Vergleich zu der gemessenen leicht versetzt ist. Dieses Verhalten kann bei dem LFP-Batterie Modell nicht beobachtet werden, was durch die flache OCV-Kurve von LFP Zellen begründbar ist. Eine Ladezustandsänderung hat im mittleren Bereich kaum einen Einfluss auf die Spannung, wodurch die Geschwindigkeit des Updateprozesses eindeutig ausreichend ist. Zum anderen ist die Temperaturabhängigkeit des Fehlers beim LFP-Batterie Modell ausgeprägter als beim LTO-Batterie Modell. Dies ist auf die Temperaturabhängigkeit der elektrochemischen Prozesse im Zellinneren der beiden Batterien zurückzuführen.

6.3. Vergleich Modellierungsansätze

Für eine abschließende Bewertung der in dieser Arbeit vorgeschlagenen Modellierungsmethodik ist ein Vergleich mit ähnlichen Spannungs- und Leistungsmodellen in der Literatur anzustellen. Tabelle 16 zeigt eine Übersicht der Modelle, die in der Literatur vorgeschlagen wurden. Die gezeigten Fehlerwerte sind immer in Relation zur Nennspannung referenziert.

Das DCR Modell von Zheng et al. [67] ist nur auf sehr kleine Ströme mit bis zu C/3 und einer Temperatur von mindestens 10 °C validiert. In Relation zu den anderen Modellierungsansätzen wird hier deutlich, dass diese Art der Modellierung nur in spezifischen, einfachen Anwendungen möglich ist. Die Modelle mit einem KF zeigen gute Ergebnisse. Der DEKF von Pei et al. [70] ist bis zu einer Temperatur von -10 °C mit einem MAXE von 4,75 % validiert. Sun et al. [155] haben das extended-KF bei größeren Strömen bis 8C

mit einem MAXE von 2 % bei 25 °C validiert. Das NN von Haiying et al. [71] hat einen relativ kleinen MAXE von 1,01 % bei leicht modellierbaren Temperaturen von 25 °C. Fleischer et al. [73] modellieren mit ANFIS in ähnlicher Genauigkeit wie das Modell mit dem AEKF, allerdings bei niedrigeren Stromraten. Die GRU Modelle von Zhao et al. [75] und die LSTM Modelle von Vidal [56] sind bei Temperaturen von mindestens -10 °C auf einen RMSE von unter 1,6 % validiert. Die maximalen Stromraten sind in beiden Fällen ähnlich, der resultierende RMSE ist jedoch bei den LSTMs etwas niedriger.

Die in dieser Arbeit vorgestellten Modelle werden in der Tabelle 16 mit den Modellen aus der Literatur verglichen. Dazu sind folgende drei Aspekte zu nennen.

- Die Modelle in der Literatur nutzen ausnahmslos den SOC der Batterie als Input für das Modell. Sie setzen damit voraus, dass ein valides Ladezustandsmodell existiert. Die Modelle aus dieser Arbeit benötigen den SOC nicht als Input, sollten aber nicht weniger gut generalisieren.
- Hohe Stromraten führen zu überproportionalen Polarisationsüberspannungen und verstärken somit die Nichtlinearität in diesen Bereichen. Für die meisten Anwendungsfälle reichen Stromraten bis zu 10 C aus, jedoch werden in Hochleistungsanwendungen wie Mild-Hybrid-Bordnetzen bis zu 30 C (vgl. Formel 2.11) angefordert. Die in dieser Arbeit betrachteten Stromraten betragen das drei- bis fünffache der anderen RNN Modelle.
- In Bezug auf die untersuchten Temperaturen werden die meisten Modelle minimal bis -10 °C validiert, das LSTM Samsung Modell von Vidal [56] sogar bis auf -20 °C. Diese Temperaturen sind für die meisten Anwendungen ausreichend. Im Automotive Sektor werden Betriebstemperaturen bis zu -25 °C betrachtet. Je tiefer die Temperatur ist desto höher ist die Nichtlinearität des Spannungsverhaltens der Batterie, wodurch die Prädiktion erschwert wird.

Tabelle 16 Fehermetriken in vergleichbaren Modellen

Ref.	**Modelltyp**	**Minmale Temperatur [℃]**	**Maximale C-Rate**	**MAPE**	**RMSE**	**MAXE**
[67]	DCR	10℃	C/3	1,80 %	Na	Na
[69]	AEKF	25℃	8 C	<1 %	Na	2 %
[70]	DEKF	-10℃	3 C	na	Na	4,75 %
[71]	BP NN	25℃	na	Na	na	1,01 %
[73]	ANFIS	10℃	1,5 C	Na	<1,6 %	2 %
[75]	GRU Panasonic	-10℃	6 C	1,30 %	<1,6 %	11,60 %
[75]	GRU Sony	-10℃	5 C	1,00 %	<1,4 %	6,10 %
[76]	LSTM Panasonic	-10℃	6 C	Na	<1,5 %	Na
[76]	LSTM Samsung	-20℃	6 C	Na	<1,3 %	Na
LTO-Batterie	LSTM	-25℃	20 C	0,98 %	1,30 %	5,60 %
LFP-Batterie	LSTM	-25℃	25 C	0,70 %	1,00 %	4,80 %

Zusätzlich zum Verzicht auf den SOC als Input verdeutlichen die drei genannten Aspekte, dass die in dieser Arbeit vorgestellten Modelle unter erschwerten Bedingungen validiert wurden. Trotz dieser Umstände sind die Fehler mit einem RMSE von 1,30 % und einem MAXE von 5,60 % beim LSTM Modell der LTO-Batterie kleiner als die der Modelle von Zhao et al. [75] und ähnlich den Modellen von Vidal [76]. Das vorgeschlagene LSTM Modell der LFP-Batterie zeigt indes mit einem RMSE von 1,00 % und einem MAXE von 4,80 % bessere Fehlermetriken als die übrigen Modelle.

7. Zusammenfassung und Ausblick

7.1. Zusammenfassung

Die fortschreitende Elektrifizierung des Antriebsstrangs führt zu einer großen Vielfalt an Batteriepacks, die in Fahrzeugen eingesetzt werden. Eine schnelle und präzise Modellierung des Verhaltens ist für den Betrieb der Fahrzeuge sowohl aus der Sicht der Sicherheit als auch des Energieverbrauchs sinnvoll.

Bisherige Modelle leiden an langen Entwicklungszeiten, bedingt durch den Prozess der Modellierungs-Methodik. Im Speziellen sind Modelle, die von anderen Modellen abhängen, nicht autark entwickelbar und in der Tendenz fehleranfälliger. Dem aktuellen Stand der Technik zufolge sind vor allem SOP-Modelle abhängig vom SOC und somit von der Güte und Entwicklungsgeschwindigkeit des SOC-Modells. In dieser Arbeit wurden die Input Parameter bewusst auf physikalisch messbare Größen und direkte Berechnungen aus diesen Größen begrenzt, um die Unabhängigkeit des Modells zu gewährleisten.

Für ein effizientes Training von NN müssen die Rohdaten aufbereitet werden. Undersampling und Oversampling Algorithmen werden genutzt, um die Datenverteilung der einzelnen Features zu optimieren, ohne dabei einen Informationsverlust zu erleiden. Im Anwendungsfall der Batteriemodellierung wird eine Regressionsanalyse angestellt. Die Datenverarbeitung wird dadurch, im Gegensatz zu den in der Literatur häufig diskutierten Klassifikationsproblemen, mit neuen Algorithmen durchgeführt. Das für diesen Anwendungsfall entwickelte MFU adressiert die Herausforderung beim Undersampling von stetigen Daten mit vielen unausgeglichenen Features. Durch die parallele Verarbeitung von mehreren Features können selbst große, willkürlich aufgezeichnete Datensätze für ein effizientes Training optimiert werden. Das entwickelte mehrstufige, auf die Regressionsanalyse angepasste Oversampling ergänzt das MFU, sodass die trainierten NN auch in Bereichen mit wenigen Datenpunkten eine gute Generalisierung aufweisen.

Nach der Datenvorverarbeitung ist das Hyperparametertuning entscheidend für die endgültige Güte eines Modells. Die Anzahl der Hyperparameter ist groß und in Verbindung mit den oftmals stetigen Parameterräumen ist eine finale Bestimmung des globalen Optimums der Hyperparameter nicht möglich. In dieser Arbeit werden für das Hyperparametertuning Vorschläge für Werte und eingrenzende Parameterräume erarbeitet, die sich im Laufe des Tunings als zielführend herausgestellt haben. Trotz der Eingrenzungen müssen viele Hyperparameter für den speziellen Anwendungsfall getuned werden. Die vorgeschlagene Hyperparametertuningpipeline kann den Prozess unterstützen und beschleunigen.

Der gesamte Prozess zur Entwicklung eines Batteriemodells auf Basis von NN ist komplex und interdisziplinär. Die Datenverarbeitungs- und Hyperparametertuningpipeline stellen ein Tool dar, welches die Entwicklung solcher Modelle vereinfacht. Um das bestmögliche Ergebnis zu erzielen, sollte die Reihenfolge der einzusetzenden Algorithmen befolgt werden. Anschließend wird diskutiert inwiefern die in Kapitel 1.2 aufgestellten Ziele im Rahmen dieser Arbeit erreicht wurden.

- **Ziel 1:** Die in vorliegender Arbeit durchgeführte Machbarkeitsstudie der Nutzbarkeit von KI in Bezug auf die SOP-Modellierung von Batterien kann das Ziel 1 nicht vollumfänglich erreichen. Jedoch ist hiermit ein wichtiger Beitrag geleistet worden diesem Ziel näher zu kommen. Durch die intensive Diskussion der Eingangs-, Trainings- und Modellparameter konnten Einflussfaktoren auf die KI-Modellierung beleuchtet und andere Entwickler dazu ermutigt werden in diesem Themengebiet weitere Forschungen anzustellen.
- **Ziel 2:** Mit der in dieser Arbeit entwickelten Datenvorverarbeitungs- und Trainingspipeline konnte das Ziel 2 erfüllt werden. Die Rohdaten werden automatisiert eingelesen, die Ein- und Ausgabeparameter extrahiert und in der Datenvorverarbeitung zu trainierbaren Datensets zusammengesetzt. Mit dem MFU wurde ein neuer Algorithmus kreiert, der es ermöglicht, die Datenverteilung eines Datensatzes anhand von mehreren erzeugten Features zu optimieren. Das nachgeschaltete SMOTE wurde auf den Anwendungsfall dahingehend angepasst, dass ein mehrstufiges Oversampling auf Regressionsdaten angewandt werden kann. Die Sequenzialisierung und Normalisierung sind mit Algorithmen, die dem Stand der Technik entsprechen, umgesetzt worden. Neben den RNNs LSTM und GRU wurden auch Transformer Modelle trainiert und miteinander verglichen. Die Parametrisierung ergab für einen Teil der Hyperparameter eindeutige pauschale Empfehlungen. Für den anderen Teil der Parameter wurden jeweils Parameterräume definiert, die den Optimierungsbereich deutlich reduzieren. Das Tuning wurde mit einem performanten Hyperparametertuning Framework implementiert.
- **Ziel 3:** Das Ziel der Verifikation wurde mit der Anwendung der Pipeline und der anschließenden Modellierung anhand von zwei Datensätzen zweier unterschiedlicher Batterien erfüllt. Die Algorithmen für die Datenvorverarbeitung sind funktionsfähig und effektiv. Die Prädiktion der Spannung erfolgt auch über mehrere Minuten problemlos, ohne auf den SOC zurückgreifen zu müssen.
- **Ziel 4:** Die beiden Modelle wurden erfolgreich in einem weiten Temperatur-, Strom- und Spannungsbereich validiert. Alle möglichen Arbeitspunkte der Batterien sind damit abgedeckt. Die Güte der erarbeiteten Modelle wurde mit Modellen aus der Literatur verglichen, mit dem Ergebnis, dass trotz größerer Validierungsbereiche eine ähnliche Genauigkeit vorliegt.

7.2. Ausblick

Aufgrund der vielzähligen Forschungstätigkeiten im Bereich der beiden Schlüsseltechnologien „künstliche Intelligenz“ und „Batterie“ werden sich neue Möglichkeiten und Herausforderungen ergeben. Die vorliegende Arbeit verbindet diese beiden Technologien und soll als Motivation dienen, die Kombination von KI und Batterie zu forcieren. Das maschinelle Lernen fungiert als Enabler für viele neue Möglichkeiten, die durch die steigenden Rechenleistungen und neuen Algorithmen entstehen. Der Beitrag der Elektrifizierung des Antriebsstrangs auf dem Weg zur Energiewende kann durch diese neuen Chancen effizienter und schneller erfolgen, was durch vorliegende Arbeit verdeutlicht wird.

Ein weiterer Schritt wäre die Überführung der in dieser Arbeit herausgearbeiteten Beiträge in industrielle Anwendungen. Denkbar ist sowohl die Nutzung der neu entwickelten Algorithmen zur Datenvorverarbeitung als auch die Übernahme der Vorschläge im Bereich des Hyperparametertunings. Für neue Anwendungsfälle ist eine Weiterentwicklung der Algorithmen möglich. Beispielsweise birgt eine Adaption des Modells eines bestimmten Batterietyps auf das einzelne in der Anwendung verbaute Batteriesample großes Verbesserungspotential. Im Zuge dieser Anpassung ist ein fortlaufendes Training der Modelle im Betrieb eine Möglichkeit zur Optimierung.

Neben der SOP-Modellierung sind SOC- und SOH-Modelle mit ähnlichen Herangehensweisen darstellbar. Die größte Fragestellung bei dieser Art der Modelle ist die Generierung der GT. SOC und SOH sind nicht physikalisch messbar und daher fiktive modellierte Größen, die für das Training erhoben werden müssen.

Literaturverzeichnis

[1] M. Hauck, “Mensch, ärgere dich nicht: 25 jahre ist es her, dass schachweltmeister garri kasparow erstmals gegen die computersoftware "deep blue" verlor. mittlerweile haben menschen keine chance mehr gegen die maschinen. hat das spiel seinen reiz verloren?” 2021. [Online]. Available: https://www.sueddeutsche.de/digital/schach-deep-blue-kasparow-ibm-1.5200655

[2] C. Witte, “Ki erkennt krebs besser als fachärzte: Diagnose dank artificial intelligence: Bei einem internationalen wettbewerb haben ai-systeme bei der erkennung von metastasen besser abgeschnitten als erfahrene pathologen.” 2018. [Online]. Available: https://www.hannovermesse.de/de/news/news-fachartikel/ki-erkennt-krebs-besser-als-fachaerzte

[3] D. S. Cardoso, P. O. Fael, and A. Espírito-Santo, “A review of micro and mild hybrid systems,” *Energy Reports*, vol. 6, pp. 385–390, 2020.

[4] L. A. Pfaffmann, “Charakterisierung von graphitelektroden für lithium-ionen-akkumulatoren mittels oso_4 exposition, xps und rem,” Ph.D. dissertation, Dissertation, Karlsruhe, Karlsruher Institut für Technologie (KIT), 2017 and Deutsche Nationalbibliothek. [Online]. Available: urn:nbn:de:swb:90-801805

[5] E. Quartarone and P. Mustarelli, “Review—emerging trends in the design of electrolytes for lithium and post-lithium batteries,” *Journal of the Electrochemical Society*, vol. 167, no. 5, p. 050508, 2020.

[6] C. F. J. Francis, I. L. Kyratzis, and A. S. Best, “Lithium-ion battery separators for ionic-liquid electrolytes: A review,” *Advanced materials (Deerfield Beach, Fla.)*, vol. 32, no. 18, p. e1904205, 2020.

[7] J. Nunes-Pereira, C. M. Costa, and S. Lanceros-Méndez, “Polymer composites and blends for battery separators: State of the art, challenges and future trends,” *Journal of Power Sources*, vol. 281, pp. 378–398, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775315002335

[8] M. Sterner and I. Stadler, “Energiespeicher - bedarf, technologien, integration,” 2017.

[9] K. N. V. Koch, “Elektrochemische und spektroskopische untersuchung von aktivmaterialien für die anwendung in lithium- und natrium-ionen-batterien,” Ph.D. dissertation, 2019.

[10] W.-J. Zhang, “A review of the electrochemical performance of alloy anodes for lithium-ion batteries,” *Journal of Power Sources*, vol. 196, no. 1, pp. 13–24, 2011.

[11] John Christensen and John Newman, “Cyclable lithium and capacity loss in li-ion cells,” *Journal of the Electrochemical Society*, vol. 152, no. 4, p. A818, 2005.

[12] F. Kindermann, “Implications of current density distribution in lithium-ion battery graphite anodes on sei formation,” Ph.D. dissertation, München, 2017.

[13] J. A. G. Rheinfeld, “Performance and safety of lithium-ion electrodes and cells: Modeling, simulation, and validation at elevated temperatures and currents,” Dissertation, Technische Universität München, München, 2019.

[14] C. P. Sandhya, B. John, and C. Gouri, “Lithium titanate as anode material for lithium-ion cells: a review,” *Ionics*, vol. 20, no. 5, pp. 601–620, 2014.

[15] K. Xu, A. von Cresce, and U. Lee, “Differentiating contributions to "ion transfer" barrier from interphasial resistance and li+ desolvation at electrolyte/graphite interface,” *Langmuir : the ACS journal of surfaces and colloids*, vol. 26, no. 13, pp. 11 538–11 543, 2010.

[16] I. Belharouak, G. M. Koenig, and K. Amine, “Electrochemistry and safety of li4ti5o12 and graphite anodes paired with limn2o4 for hybrid electric vehicle li-ion battery applications,” *Journal of Power Sources*, vol. 196, no. 23, pp. 10 344–10 350, 2011.

[17] N. Nitta, F. Wu, J. T. Lee, and G. Yushin, “Li-ion battery materials: present and future,” *Materials Today*, vol. 18, no. 5, pp. 252–264, 2015.

[18] H. Li, M. Cormier, N. Zhang, J. Inglis, J. Li, and J. R. Dahn, “Is cobalt needed in ni-rich positive electrode materials for lithium ion batteries?” *Journal of the Electrochemical Society*, vol. 166, no. 4, pp. A429–A439, 2019.

[19] S. Al Barazi, U. Näher, S. Vetter, P. Schütte, M. Liedtke, M. Baier, and G. Franken, “Cobalt from the dr congo – potential, risks and significance for the global cobalt market,” *Commodity TopNews*, no. 53, 2017.

[20] F. Wu, J. Maier, and Y. Yu, “Guidelines and trends for next-generation rechargeable lithium and lithium-ion batteries,” *Chemical Society reviews*, vol. 49, no. 5, pp. 1569–1614, 2020.

[21] S.-L. Wu, W. Zhang, X. Song, A. K. Shukla, G. Liu, V. Battaglia, and V. Srinivasan,

"High rate capability of li(ni 1/3 mn 1/3 co 1/3)o 2 electrode for li-ion batteries," *Journal of the Electrochemical Society*, vol. 159, no. 4, pp. A438–A444, 2012.

[22] S.-Y. Chung, J. T. Bloking, and Y.-M. Chiang, "Electronically conductive phospho-olivines as lithium storage electrodes," *Nature materials*, vol. 1, no. 2, pp. 123–128, 2002.

[23] D. Doughty and A. Pesaran, "Vehicle battery safety roadmap guidance," Ph.D. dissertation, Golden, 2012.

[24] W.-J. Zhang, "Structure and performance of lifepo4 cathode materials: A review," *Journal of Power Sources*, vol. 196, no. 6, pp. 2962–2970, 2011.

[25] X. Han, L. Lu, Y. Zheng, X. Feng, Z. Li, J. Li, and M. Ouyang, "A review on the key issues of the lithium ion battery degradation among the whole life cycle," *eTransportation*, vol. 1, p. 100005, 2019.

[26] J. Rivera-Barrera, N. Muñoz-Galeano, and H. Sarmiento-Maldonado, "Soc estimation for lithium-ion batteries: Review and future challenges," *Electronics*, vol. 6, no. 4, p. 102, 2017.

[27] P. A. Topan, M. N. Ramadan, G. Fathoni, A. I. Cahyadi, and O. Wahyunggoro, "State of charge (soc) and state of health (soh) estimation on lithium polymer battery via kalman filter," in *2016 2nd International Conference on Science and Technology-Computer (ICST)*. IEEE, 27.10.2016 - 28.10.2016, pp. 93–96.

[28] N. Ding, K. Prasad, T. T. Lie, and J. Cui, "State of charge estimation of a composite lithium-based battery model based on an improved extended kalman filter algorithm," *Inventions*, vol. 4, no. 4, p. 66, 2019.

[29] S. Peng, C. Chen, H. Shi, and Z. Yao, "State of charge estimation of battery energy storage systems based on adaptive unscented kalman filter with a noise statistics estimator," *IEEE Access*, vol. 5, pp. 13 202–13 212, 2017.

[30] F. Malmir, B. Xu, and Z. Filipi, "A heuristic supervisory controller for a 48v hybrid electric vehicle considering fuel economy and battery aging," in *A Heuristic Supervisory Controller for a 48V Hybrid Electric Vehicle Considering Fuel Economy and Battery Aging*, ser. SAE Technical Paper Series. SAE International400 Commonwealth Drive, Warrendale, PA, United States, 2019.

[31] P. Schmitz, "Elektrische verbindung von zylindrischen lithium-ionen-zellen zur her-

stellung von energiespeichersystemen," Dissertation, Technische Universität München, München, 2019.

[32] J. J. Sturm, "State-estimation of lithium-ion batteries using physicochemical models and experimental characterization techniques," Dissertation, Technische Universität München, München, 2021.

[33] W.-Y. Chang, "The state of charge estimating methods for battery: A review," *ISRN Applied Mathematics*, vol. 2013, no. 5, pp. 1–7, 2013.

[34] J. Meng, M. Boukhnifer, D. Diallo, and T. Wang, "A new cascaded framework for lithium-ion battery state and parameter estimation," *Applied Sciences*, vol. 10, no. 3, p. 1009, 2020.

[35] Y.-H. Chiang, W.-Y. Sean, and J.-C. Ke, "Online estimation of internal resistance and open-circuit voltage of lithium-ion batteries in electric vehicles," *Journal of Power Sources*, vol. 196, no. 8, pp. 3921–3932, 2011.

[36] W. Waag, C. Fleischer, and D. U. Sauer, "On-line estimation of lithium-ion battery impedance parameters using a novel varied-parameters approach," Ph.D. dissertation.

[37] T. Feng, L. Yang, X. Zhao, H. Zhang, and J. Qiang, "Online identification of lithium-ion battery parameters based on an improved equivalent-circuit model and its implementation on battery state-of-power prediction," *Journal of Power Sources*, vol. 281, pp. 192–203, 2015.

[38] E. Cabrera-Castillo, F. Niedermeier, and A. Jossen, "Calculation of the state of safety (sos) for lithium ion batteries," *Journal of Power Sources*, vol. 324, pp. 509–520, 2016.

[39] H. Rahimi-Eichi, U. Ojha, F. Baronti, and M.-Y. Chow, "Battery management system: An overview of its application in the smart grid and electric vehicles," *IEEE Industrial Electronics Magazine*, vol. 7, no. 2, pp. 4–16, 2013.

[40] Z. Y. Jiang, Z. G. Qu, J. F. Zhang, and Z. H. Rao, "Rapid prediction method for thermal runaway propagation in battery pack based on lumped thermal resistance network and electric circuit analogy," *Applied Energy*, vol. 268, p. 115007, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261920305195

[41] Y. Deng, C. Feng, J. E, H. Zhu, J. Chen, M. Wen, and H. Yin, "Effects of different

coolants and cooling strategies on the cooling performance of the power lithium ion battery system: A review," *Applied Thermal Engineering*, vol. 142, pp. 10–29, 2018.

[42] YUAN Hao, WANG Li-Fang, WANG Li-Ye, "Battery thermal management system with liquid cooling and heating in electric vehicles," *Journal of Automotive Safety and Energy*, vol. 3, no. 4, p. 371, 2012.

[43] P. Keil, S. F. Schuster, J. Wilhelm, J. Travi, A. Hauser, R. C. Karl, and A. Jossen, "Calendar aging of lithium-ion batteries," *Journal of the Electrochemical Society*, vol. 163, no. 9, pp. A1872–A1880, 2016.

[44] J. Cao, N. Schofield, and A. Emadi, "Battery balancing methods: A comprehensive review," in *IEEE Vehicle Power and Propulsion Conference, 2008*. Piscataway, NJ: IEEE, 2008, pp. 1–6.

[45] J. Gallardo-Lozano, E. Romero-Cadaval, M. I. Milanes-Montero, and M. A. Guerrero-Martinez, "Battery equalization active methods," *Journal of Power Sources*, vol. 246, pp. 934–949, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775313013669

[46] M.-Y. Kim, J.-H. Kim, and G.-W. Moon, "Center-cell concentration structure of a cell-to-cell balancing circuit with a reduced number of switches," *IEEE Transactions on Power Electronics*, vol. 29, no. 10, pp. 5285–5297, 2014.

[47] A. M. Imtiaz and F. H. Khan, ""time shared flyback converter" based regenerative cell balancing technique for series connected li-ion battery strings," *IEEE Transactions on Power Electronics*, vol. 28, no. 12, pp. 5960–5975, 2013.

[48] A. Reindl, H. Meier, and M. Niemetz, "Scalable, decentralized battery management system based on self-organizing nodes," in *Architecture of computing systems – ARCS 2020*, ser. LNCS sublibrary: SL 1, Theoretical computer science and general issues, A. Brinkmann, W. Karl, S. Lankes, S. Tomforde, T. Pionteck, and C. Trinitis, Eds. Cham: Springer, 2020, vol. 12155, pp. 171–184.

[49] A. Hauser and R. Kuhn, "High-voltage battery management systems (bms) for electric vehicles," 2015. [Online]. Available: http://dx.doi.org/10.1016/B978-1-78242-377-5.00011-X

[50] S. Madani, E. Schaltz, and S. Knudsen Kær, "An electrical equivalent circuit model of a lithium titanate oxide battery," *Batteries*, vol. 5, no. 1, p. 31, 2019.

[51] A. Farmann, “A comparative study of reduced-order equivalent circuit models for state-of-available-power prediction of lithium-ion batteries in electric vehicles,” Dissertations, RWTH Aachen University, 2019.

[52] X. Hu, S. E. Li, and Y. Yang, “Advanced machine learning approach for lithium-ion battery state estimation in electric vehicles,” *IEEE Transactions on Transportation Electrification*, vol. 2, no. 2, pp. 140–149, 2016.

[53] E. Chemali, P. J. Kollmeyer, M. Preindl, and A. Emadi, “State-of-charge estimation of li-ion batteries using deep neural networks: A machine learning approach,” *Journal of Power Sources*, vol. 400, pp. 242–255, 2018.

[54] F. Yang, W. Li, C. Li, and Q. Miao, “State-of-charge estimation of lithium-ion batteries based on gated recurrent neural network,” *Energy*, vol. 175, pp. 66–75, 2019.

[55] A. Khalid, A. Sundararajan, I. Acharya, and A. I. Sarwat, “Prediction of li-ion battery state of charge using multilayer perceptron and long short-term memory models,” in *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, 19.06.2019 - 21.06.2019, pp. 1–6.

[56] C. Vidal, P. Malysz, P. Kollmeyer, and A. Emadi, “Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art,” *IEEE Access*, vol. 8, pp. 52 796–52 814, 2020.

[57] J. Sihvo, T. Roinila, and D.-I. Stroe, “Soh analysis of li-ion battery based on ecm parameters and broadband impedance measurements,” in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 10/18/2020 - 10/21/2020, pp. 1923–1928.

[58] M. Gholizadeh and A. Yazdizadeh, “Systematic mixed adaptive observer and ekf approach to estimate soc and soh of lithium–ion battery,” *IET Electrical Systems in Transportation*, vol. 10, no. 2, pp. 135–143, 2020.

[59] M. Zeng, P. Zhang, Y. Yang, C. Xie, and Y. Shi, “Soc and soh joint estimation of the power batteries based on fuzzy unscented kalman filtering algorithm,” *Energies*, vol. 12, no. 16, p. 3122, 2019.

[60] G.-w. You, S. Park, and D. Oh, “Real-time state-of-health estimation for electric vehicle batteries: A data-driven approach,” *Applied Energy*, vol. 176, pp. 92–103, 2016.

[61] H. Chaoui and C. C. Ibe-Ekeocha, “State of charge and state of health estima-

tion for lithium batteries using recurrent neural networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8773–8783, 2017.

[62] C. Zhang, Y. Zhu, G. Dong, and J. Wei, "Data–driven lithium–ion battery states estimation using neural networks and particle filtering," *International Journal of Energy Research*, vol. 43, no. 8, p. 3681, 2019.

[63] A. Farmann and D. U. Sauer, "A comprehensive review of on-board state-of-available-power prediction techniques for lithium-ion batteries in electric vehicles," *Journal of Power Sources*, vol. 329, pp. 123–137, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775316310308

[64] O. Bohlen, "Impedance based battery monitoring," Ph.D. dissertation, RWTH Aachen University, 2008.

[65] A. BOEHM and J. WEBER, "Adaptives verfahren zur bestimmung der maximal abgebaren oder aufnehmbaren leistung einer batterie," Patent WO/2011/095 368, 03.01.2011. [Online]. Available: https://patentscope.wipo.int/search/de/detail.jsf?docId=WO2011095368

[66] W. Waag, C. Fleischer, and D. U. Sauer, "Critical review of the methods for monitoring of lithium-ion batteries in electric and hybrid vehicles," *Journal of Power Sources*, vol. 258, pp. 321–339, 2014.

[67] L. Zheng, J. Zhu, G. Wang, D. D.-C. Lu, P. McLean, and T. He, "Experimental analysis and modeling of temperature dependence of lithium-ion battery direct current resistance for power capability prediction," in *2017 20th International Conference on Electrical Machines and Systems (ICEMS)*. IEEE, 2017. [Online]. Available: http://dx.doi.org/10.1109/icems.2017.8056426

[68] L. W. Juang, P. J. Kollmeyer, T. M. Jahns, and R. D. Lorenz, "Improved nonlinear model for electrode voltage–current relationship for more consistent online battery system identification," *IEEE Transactions on Industry Applications*, vol. 49, no. 3, pp. 1480–1488, 2013.

[69] F. Sun, R. Xiong, and H. He, "Estimation of state-of-charge and state-of-power capability of lithium-ion battery considering varying health conditions," *Journal of Power Sources*, vol. 259, pp. 166–176, 2014.

[70] L. Pei, C. Zhu, T. Wang, R. Lu, and C. C. Chan, "Online peak power prediction based on a parameter and state estimator for lithium-ion batteries in electric vehicles," *Energy*, vol. 66, pp. 766–778, 2014.

[71] W. Haiying, H. Zhonghua, H. Yu, and L. Gechen, “Power state prediction of battery based on bp neural network,” in *2012 7th International Forum on Strategic Technology (IFOST)*. IEEE, 2012. [Online]. Available: http://dx.doi.org/10.1109/ifost.2012.6357617

[72] C. Fleischer, W. Waag, Z. Bai, and D. U. Sauer, “Self-learning state-of-available-power prediction for lithium-ion batteries in electrical vehicles,” in *IEEE Vehicle Power and Propulsion Conference (VPPC), 2012*. Piscataway, NJ: IEEE, 2012, pp. 370–375.

[73] ——, “Adaptive on-line state-of-available-power prediction of lithium-ion batteries,” Ph.D. dissertation.

[74] ——, “On-line self-learning time forward voltage prognosis for lithium-ion batteries using adaptive neuro-fuzzy inference system,” vol. 243, pp. 728–749.

[75] R. Zhao, P. J. Kollmeyer, R. D. Lorenz, and T. M. Jahns, “A compact methodology via a recurrent neural network for accurate equivalent circuit type modeling of lithium-ion batteries,” *IEEE Transactions on Industry Applications*, vol. 55, no. 2, pp. 1922–1931, 2019.

[76] C. Vidal, “Deep neural networks for improved terminal voltage and state-of-charge estimation of lithium-ion batteries for traction applications,” Dissertation, McMaster University, 2020. [Online]. Available: http://hdl.handle.net/11375/25851

[77] K. Reif, K.-E. Noreikat, and K. Borgeest, *Kraftfahrzeug-Hybridantriebe: Grundlagen, Komponenten, Systeme, Anwendungen*, ser. ATZ / MTZ-Fachbuch. Wiesbaden: Vieweg+Teubner Verlag, 2012. [Online]. Available: http://gbv.eblib.com/patron/FullRecord.aspx?p=1083174

[78] S. Hayslett, K. van Maanen, W. Wenzel, and T. Husain, “The 48-v mild hybrid: Benefits, motivation, and the future outlook,” *IEEE Electrification Magazine*, vol. 8, no. 2, pp. 11–17, 2020.

[79] R. Ellinger, C. Kaup, and T. Pels, “Potenziale und grenzen von 48-v-systemen,” *ATZextra*, vol. 22, no. S1, pp. 16–21, 2017.

[80] U. C. Blessing, J. Meissner, M. Schweiher, and T. Hoffmeister, “Scalable hybrid dual-clutch transmission,” *ATZ worldwide*, vol. 116, no. 12, pp. 4–9, 2014.

[81] M. Schudeleit, C. Sieg, and F. Küçükay, “The potential of 48v hev in real driving,” 2015.

[82] R. Bao, V. Avila, and J. Baxter, “Effect of 48 v mild hybrid system layout on powertrain system efficiency and its potential of fuel economy improvement,” in *SAE Technical Paper Series*, ser. SAE Technical Paper Series. SAE International400 Commonwealth Drive, Warrendale, PA, United States, 2017.

[83] A. Bongards, S. Mohon, D. Semenov, and W. Wenzel, “Comparing 48v mild hybrid concepts using a hybrid-simulation-toolkit,” in *19. Internationales Stuttgarter Symposium*, ser. ATZ live, A. Wagner, M. Bargende, H.-C. Reuss, and J. Wiedemann, Eds. Wiesbaden and [Heidelberg]: Springer Vieweg, 2019, pp. 1085–1100.

[84] Nahlbach, Körner, Kahnt, “Active engine-off coasting using 48v: Economic reduction of co2 emissions: Baden-baden, 14. and 15. october 2015 = 17th international congress eliv 2015,” *17. Internationaler Kongress ELIV 2015*, vol. 2249, 2015.

[85] J. Benajes, A. García, J. Monsalve-Serrano, and S. Martínez-Boggio, “Optimization of the parallel and mild hybrid vehicle platforms operating under conventional and advanced combustion modes,” *Energy Conversion and Management*, vol. 190, pp. 73–90, 2019.

[86] S. Barth, M. Fischer, and J. Böttcher, “Analyse kundenrelevanter vorteile der 48-v-hybridisierung,” *MTZ Motortech Z*, vol. 79, no. 12, pp. 52–57, 2018.

[87] Hans-Martin Fischer and Dr. Reiner Korthauer, “48-volt-bordnetz: Schlüsseltechnologie auf dem weg zur elektromobilität,” 2015.

[88] A. D. Wearing, J. Haybittle, R. Bao, J. W. Baxter, C. Rouaud, and O. Taskin, “Development of high power 48v powertrain components for mild hybrid light duty vehicle applications,” in *2018 IEEE Energy Conversion Congress and Exposition (ECCE)*. Piscataway, NJ: IEEE, 2018, pp. 3893–3900.

[89] A. Fatemi, T. Nehl, X. Yang, L. Hao, S. Gopalakrishnan, A. Omekanda, and C. Namuduri, “Design of an electric machine for a 48-v mild hybrid vehicle,” in *2018 IEEE Energy Conversion Congress and Exposition (ECCE)*. Piscataway, NJ: IEEE, 2018, pp. 2278–2285.

[90] C. Suarez and W. Martinez, “Fast and ultra-fast charging for battery electric vehicles – a review,” in *ECCE 2019*. Piscataway, NJ: IEEE, 2019, pp. 569–575.

[91] A. Eftekhari, “Lithium-ion batteries with high rate capabilities,” *ACS Sustainable Chemistry & Engineering*, vol. 5, no. 4, pp. 2799–2816, 2017.

[92] C. Bergmeir and J. M. Benítez, “On the use of cross-validation for time series predictor evaluation,” *Information Sciences*, vol. 191, pp. 192–213, 2012.

[93] A. Lahiani, “Deep learning solutions for cancer drug development in digital pathology,” Ph.D. dissertation, München, 2020.

[94] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *Proceedings of 2017 International Conference on Engineering & Technology (ICET'2017)*. Piscataway, NJ: IEEE, 2017, pp. 1–6.

[95] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014. [Online]. Available: https://arxiv.org/pdf/1406.2661

[96] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks.” [Online]. Available: https://arxiv.org/pdf/1511.06434

[97] S. Gessulat, “A deep learning model for the proteome-wide prediction of peptide tandem mass spectra,” Ph.D. dissertation, München, 2020.

[98] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[99] Ilya Sutskever, “Training recurrent neural networks,” Dissertation, University of Toronto, Toronto, 2013. [Online]. Available: https://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf

[100] E. N. Lorenz, “Deterministic nonperiodic flow,” *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[101] M. Madondo and T. Gibbons, *Learning and modeling chaos using lstm recurrent neural networks*, 2018. [Online]. Available: http://micsymposium.org/mics2018/proceedings/mics_2018_paper_26.pdf

[102] S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen*, 1991. [Online]. Available: https://scholar.google.de/citations?user=tvuh3wmaaaaj&hl=de&oi=sra

[103] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[104] O. Scheel, “Using deep neural networks for scene understanding and behaviour prediction in autonomous driving,” Ph.D. dissertation, Technische Universität München, München, 2020.

[105] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[106] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” 2014. [Online]. Available: https://arxiv.org/pdf/1406.1078

[107] A. M. Javid, S. Das, M. Skoglund, and S. Chatterjee, “A relu dense layer to improve the performance of neural networks,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6/6/2021 - 6/11/2021, pp. 2810–2814.

[108] F. Altche and A. de La Fortelle, “An lstm network for highway trajectory prediction,” in *IEEE ITSC 2017*. Piscataway, NJ: IEEE, 2017, pp. 353–359.

[109] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning,” Ph.D. dissertation, Cambridge, Massachusetts and London, England, 2016. [Online]. Available: http://www.deeplearningbook.org/

[110] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[111] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762

[112] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018. [Online]. Available: https://arxiv.org/pdf/1810.04805

[113] M. Radfar, A. Mouchtaris, and S. Kunzmann, “End-to-end neural transformer based spoken language understanding,” 2020. [Online]. Available: https://arxiv.org/pdf/2008.10984

[114] David Kriesel, “Ein kleiner überblick über neuronale netze,” Ph.D. dissertation, 2007. [Online]. Available: erhÃd'ltlichaufhttp://www.dkriesel.com

[115] S. Ruder, “An overview of gradient descent optimization algorithms,” Ph.D. dissertation, Dublin, 2016. [Online]. Available: https://arxiv.org/pdf/1609.04747

[116] M. Rana, M. M. Uddin, and M. M. Hoque, “Effects of activation functions and optimizers on stock price prediction using lstm recurrent networks,” in *Proceedings of 2019 3rd International Conference on Computer Science and Artificial Intelligence ;*, ser. ICPS. New York, New York: The Association for Computing Machinery, 2019, pp. 354–358.

[117] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” Ph.D. dissertation, University of Amsterdam, Amsterdam, 2014. [Online]. Available: https://arxiv.org/pdf/1412.6980

[118] X. Qian and D. Klabjan, “The impact of the mini-batch size on the variance of gradients in stochastic gradient descent,” 2020. [Online]. Available: https://arxiv.org/pdf/2004.13146

[119] H. Wang, K. Ren, and J. Song, “A closer look at batch size in mini-batch training of deep auto-encoders,” in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. Piscataway, NJ: IEEE, 2017, pp. 2756–2761.

[120] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” Ph.D. dissertation, Lehigh University Bethlehem, Bethlehem, 2016. [Online]. Available: https://arxiv.org/pdf/1606.04838

[121] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015. [Online]. Available: http://arxiv.org/pdf/1502.03167v3

[122] S. Karsoliya, “Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture,” *International Journal of Engineering Trends and Technology*, vol. 3, no. 6, pp. 714–717, 2012.

[123] J. M. Alvarez and M. Salzmann, “Learning the number of neurons in deep networks,” Ph.D. dissertation, 2018/10/11.

[124] P. Liashchynskyi and P. Liashchynskyi, “Grid search, random search, genetic algorithm: A big comparison for nas,” 2019. [Online]. Available: http://arxiv.org/pdf/1912.06059v1

[125] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, “Fast bayesian optimization of machine learning hyperparameters on large datasets,” vol. 54, pp. 528–536, 2017. [Online]. Available: http://proceedings.mlr.press/v54/klein17a.html

[126] T. Agrawal, “Hyperparameter optimization in machine learning: Make your machine learning and deep learning models more efficient,” Ph.D. dissertation, Berkeley CA, 2021.

[127] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna,” in *KDD’19*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds. New York, NY: Association for Computing Machinery, 2019, pp. 2623–2631.

[128] R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[129] T. He, J. Zhang, Z. Zhou, and J. Glass, “Quantifying exposure bias for open-ended language generation,” 2019. [Online]. Available: https://arxiv.org/pdf/1905.10617

[130] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” 2015. [Online]. Available: https://arxiv.org/pdf/1506.03099

[131] O. Demir-Kavuk, M. Kamada, T. Akutsu, and E.-W. Knapp, “Prediction using stepwise l1, l2 regularization and feature selection for small data sets with large number of features,” *BMC Bioinformatics*, vol. 12, no. 1, 2011.

[132] P. S. Crowther and R. J. Cox, “A method for optimal division of data sets for use in neural networks,” in *Knowledge-Based Intelligent Information and Engineering Systems*, R. Khosla, R. J. Howlett, and L. C. Jain, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–7.

[133] P. Skryjomski and B. Krawczyk, “Influence of minority class instance types on smote imbalanced data oversampling,” in *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, ser. Proceedings of Machine Learning Research, L. Torgo, B. Krawczyk, P. Branco, and N. Moniz, Eds., vol. 74. ECML-PKDD, Skopje, Macedonia: PMLR, 2017, pp. 7–21. [Online]. Available: http://proceedings.mlr.press/v74/skryjomski17a.html

[134] N. Nnamoko and I. Korkontzelos, “Efficient treatment of outliers and class imbalance for diabetes prediction,” *Artificial intelligence in medicine*, vol. 104, p. 101815, 2020.

[135] Elhassan AT, Aljourf M, Al-Mohanna F, and Shoukri M, *Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method*, 2016, vol. 01. [Online]. Available: https://www.researchgate.net/profile/mohamed_shoukri/publication/326590590_classification_of_imbalance_data_using_tomek_link_t-link_combined_with_random_under-sampling_rus_as_a_data_reduction_method

[136] S. Vimalraj and P. Dr.R, "A review on handling imbalanced data," 2018.

[137] Yuriko Okazaki, Shinichiro Okazaki, Shingo Asamoto, and Pang-jo Chun, "Undersampling strategy for machine-learned deterioration regression model in concrete bridges," *Journal of Advanced Concrete Technology*, vol. 18, no. 12, pp. 753–766, 2020. [Online]. Available: https://www.jstage.jst.go.jp/article/jact/18/12/18_753/_article/-char/ja/

[138] M. Beckmann, N. F. F. Ebecken, and B. S. L. Pires de Lima, "A knn undersampling approach for data balancing," *Journal of Intelligent Learning Systems and Applications*, vol. 07, no. 04, pp. 104–116, 2015.

[139] I. Tomek *et al.*, "An experiment with the edited nearest-nieghbor rule," 1976.

[140] L. Torgo, P. Branco, R. P. Ribeiro, and B. Pfahringer, "Resampling strategies for regression," *Expert Systems*, vol. 32, no. 3, pp. 465–476, 2015.

[141] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks, 2008*. Piscataway, NJ: IEEE, 2008, pp. 1322–1328.

[142] H. Dubey and V. Pudi, "Class based weighted k-nearest neighbor over imbalance dataset," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence, D. Hutchison, T. Kanade, and J. Kittler, Eds. Berlin/Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7819, pp. 305–316.

[143] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[144] A. Ali and N. Senan, "The effect of normalization in violence video classification performance," *IOP Conference Series: Materials Science and Engineering*, vol. 226, p. 012082, 2017.

[145] Gökhan AKSU, Cem Oktay GÜZELLER, and Mehmet Taha ESER, "The effect of the normalization method used in different sample sizes on the success of artificial neural network model," *International Journal of Assessment Tools in Education*, vol. 6, no. 2, pp. 170–192, 2019. [Online]. Available: https://dergipark.org.tr/en/pub/ijate/issue/44255/479404

[146] Z. M. Zain and N. M. Alturki, "Covid-19 pandemic forecasting using cnn-lstm: A hybrid approach," *Journal of Control Science and Engineering*, vol. 2021, pp. 1–23, 2021.

[147] B. Mu, J. Li, S. Yuan, and X. Luo, "Prediction of north atlantic oscillation index associated with the sea level pressure using dwt-lstm and dwt-convlstm networks," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–14, 2020.

[148] D. Jerouschek, O. Tan, R. Kennel, and A. Taskiran, "Modeling lithium-ion batteries using machine learning algorithms for mild-hybrid vehicle applications," in *2021 International Conference on Smart Energy Systems and Technologies (SEST)*. IEEE, 9/6/2021 - 9/8/2021, pp. 1–6.

[149] D. Jerouschek, Ö. Tan, R. Kennel, and A. Taskiran, "Data preparation and training methodology for modeling lithium-ion batteries using a long short-term memory neural network for mild-hybrid vehicle applications," *Applied Sciences*, vol. 10, no. 21, p. 7880, 2020.

[150] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, "A system for massively parallel hyperparameter tuning," *Conference on Machine Learning and Systems*, vol. 2020, 2020. [Online]. Available: https://arxiv.org/pdf/1810.05934

[151] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," 2018. [Online]. Available: https://arxiv.org/pdf/1801.05134

[152] A. Komatsuzaki, "One epoch is all you need," 2019. [Online]. Available: http://arxiv.org/pdf/1906.06669v1

[153] F. Yang, X. Song, F. Xu, and K.-L. Tsui, "State-of-charge estimation of lithium-ion batteries via long short-term memory network," *IEEE Access*, vol. 7, pp. 53 792–53 799, 2019.

[154] C. Li, F. Xiao, and Y. Fan, "An approach to state of charge estimation of lithium-ion

batteries based on recurrent neural networks with gated recurrent unit," *Energies*, vol. 12, no. 9, p. 1592, 2019.

[155] A. Haubrock, "Degradationsuntersuchungen von lithium-ionen batterien bei deren einsatz in elektro- und hybridfahrzeugen," Ph.D. dissertation, Göttingen, 2011.