
**Ostfalia**

Hochschule für angewandte Wissenschaften

Fakultät für Maschinenbau

Institut für Mechatronik

# Studienarbeit

## Parametereinflussanalyse eines künstlichen neuronalen Netzes am Beispiel einer Ladezustandsschätzung für Lithium-Ionen-Batterien

Verfasser: Alexander Kohn
Matrikelnummer: 70455099

Prüferin: Prof. Dr.-Ing. X. Liu-Henke
Betreuer: S. Jacobitz, M. Eng
O. A. Yarom, M. Eng

# Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte Hilfe angefertigt, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Königslutter, 21. September 2021

# Inhaltsverzeichnis

1 Einleitung und Aufgabenstellung ..... 1

2 Stand der Technik ..... 2

2.1 Batterietechnik ..... 2

2.2 Neuronale Netze ..... 14

2.2.1 Biologische Grundlagen ..... 14

2.2.2 Künstliche Neuronale Netze ..... 15

2.2.3 Netzstrukturen und -arten ..... 18

2.2.4 Lern- und Trainingsverfahren ..... 22

2.2.5 Anwendungsmöglichkeiten für KNN ..... 25

3 Allgemeine Methodik/Rahmbedingungen ..... 26

3.1 Vorgehensweise beim Erstellen eines Neuronalen Netzes ..... 26

3.2 Anforderungen ..... 27

4 Konzeption der Methode zur Parametereinflussanalyse/Versuchsaufbau ..... 29

4.1 Auswahl der Netzart ..... 29

4.2 Vorauswahl der Hyperparameter ..... 30

4.3 Versuchsplanung ..... 31

4.4 Automatisierte Versuchsdurchführung ..... 32

4.5 Auswahl von Trainings- und Testprofil ..... 34

5 Versuchsdurchführung und Auswertung ..... 36

5.1 Erstellung von Trainings- und Testprofil ..... 36

5.2 Versuchsdurchführung mit Trainingsdaten ..... 38

5.3 Erprobung mit unbekannten Testdaten ..... 43

6 Analyse der Ergebnisse ..... 46

7 Zusammenfassung und Ausblick ..... 49

Abkürzungsverzeichnis ..... 50

Tabellenverzeichnis ..... 51

Abbildungsverzeichnis ..... 52

Literaturverzeichnis ..... 54

# 1 Einleitung und Aufgabenstellung

Künstlich Intelligenz (KI) ist in der heutigen Zeit nicht mehr wegzudenken. Dabei wird versucht, die menschliche Intelligent nachzuahmen. Sie kommt in den unterschiedlichsten Bereichen zum Einsatz. Die bekanntesten Einsatzgebiete im Alltag sind zum Beispiel die Übersetzer für Sprachen oder die Gesichtserkennung bei Smartphones. Ein wichtiges Teilgebiet der KI ist das maschinelle Lernen. Durch historische Daten werden die umfangreichsten Aufgaben gelöst. Dazu zählen beispielsweise die Vorhersagen von Maschinenstörungen oder Vorgängen in der Logistik.

Beim maschinellen Lernen gibt es eine Vielzahl von Methoden um spezifische Aufgaben zu lösen. Eine weit verbreitet und sehr spannende Methode ist das künstlich neuronale Netz. Dabei werden die Vorgänge im Gehirn künstlich nachgebildet. Dieses Nachbilden wird auch als „Deep Learning“ bezeichnet.

Im Rahmen dieser Arbeit wird eine Parametereinflussanalyse eines künstlichen neuronalen Netzes durchgeführt. Die Motivation hierfür entstand im Verlauf einer weiteren Studienarbeit. Dort wurden verschieden KI-Ansätze für die Ladezustandsschätzung verglichen und die Wahl fiel auf das KNN. Das Auslegen eines KNN ist keine einfache Aufgabe, da viele verschiedene Parameter für die Architektur im Vorfeld eingestellt werden müssen, um somit ein Training auch erfolgreich durchführen zu können und lösungsorientierte Ergebnisse zu erzielen. Das KNN ist im weitesten Sinne ein Black-Box-Modell in dem die Vorgänge bekannt sind. In der Praxis jedoch ist es nahezu unmöglich über die Parameter einen Einblick in die erlernte Lösung zu erhalten. Aus diesem Grund werden KNN durch iterative Prozesse durch Vorgabe von bereits gelöstem Problem trainiert. Die Architektur, dass die beste Perfomance liefert, wird dann zum Lösen von gleichen Problemen verwendet.

Die Aufgabe in dieser Arbeit besteht darin, eine geeignete Architektur für die Ladezustandsschätzung zu finden und dabei die Einflüsse der verwendeten Parameter zu analysieren und im besten Fall eine gute Grundlage für die Ladezustandsschätzung in der weiterführenden Arbeit zu erhalten. Außerdem kann das Auslegen des KNN für zukünftige Anwendungen mit ähnlichen Problemen effektiver durchgeführt werden.

# 2 Stand der Technik

In diesem Kapitel soll ein Überblick über die Grundlagen der Batterietechnik und der Neuronalen Netze gegeben werden.

## 2.1 Batterietechnik

In der Batterietechnik unterscheidet man zwischen Primär- und Sekundärbatterien. Die Primärbatterie wandelt chemische Energie irreversibel in elektrische Energie um. Sie wird im allgemeinen Sprachgebrauch als Batterie bezeichnet. Sekundärbatterien können dagegen die elektrische Energie über einen gewissen Zeitraum in Form von chemischer Energie speichern. Diese werden auch als Akkumulatoren bezeichnet. Es kommen die unterschiedlichsten Batterien für die unterschiedlichsten Einsatzgebiete zum Einsatz. In der nächsten Abbildung 2.1 sind die umsatzstärksten Batteriesysteme weltweit von 2015 abgebildet:

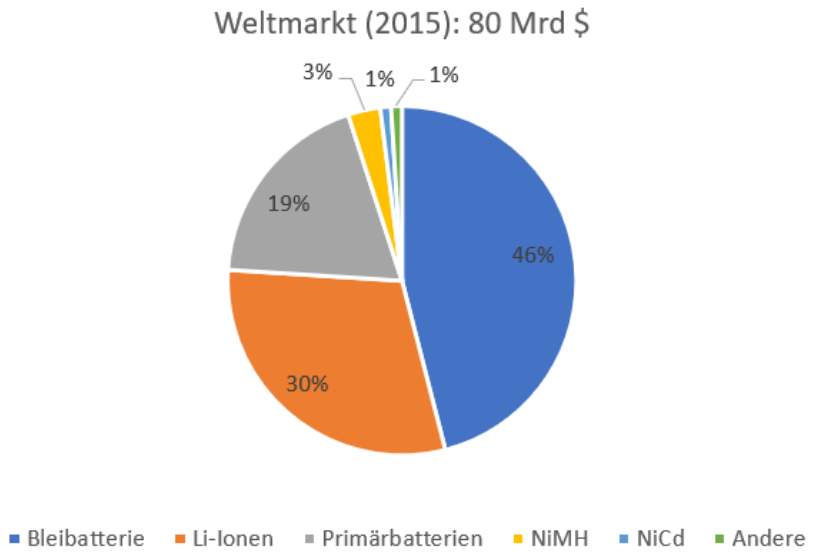


Abbildung 2.1: Weltweite Marktaufteilung nach Batteriesystemen auf Kostenbasis in Anlehnung an [JW19]

Wie aus der vorliegenden Abbildung zu erkennen ist, hat die Bleibatterie den weltweit höchsten Umsatz. Sie kommt zum Beispiel bei Fahrzeugen als Starterbatterien und für die Notstromversorgung zum Einsatz. Die Lithium-Ionen-Batterie (Li-Ion-Batterie) besitzt den zweitgrößten Anteil und ist der Spitzenreiter in der Elektromobilität, welche in den letzten Jahren ein starkes Wachstum erlebt hat.

Das Prinzip einer Batterie beruht auf der Redoxreaktion. Die Redoxreaktion setzt sich aus den Teilvorgängen von Reduktion und Oxidation zusammen und soll verdeutlichen, dass diese

immer kombiniert ablaufen. Bei der Redoxreaktion werden Elektronen von einem Reaktionspartner auf den anderen, im gleichen Maß, übertragen. Diese wandernden Elektroden stellen den Strom für den angeschlossenen Verbraucher bereit. Der Reaktionspartner, der die Elektronen aufnimmt, wird als Oxidationsmittel und der, der sie abgibt als Reduktionsmittel bezeichnet. Bei der Oxidation werden Elektronen vom Reduktionsmittel abgegeben. Sie werden als Donator bezeichnet. Das Oxidationsmittel, welches die Elektronen aufnimmt, also reduziert, wird als Akzeptor bezeichnet[Ku20].

Der Donator besteht bei einer Batterie aus einem unedlen Metallatom, da diese Atome eine höhere Tendenz haben, Elektronen zu verlieren. Der Akzeptor besteht dagegen aus einem edlen Metallatom, welches die Elektronen aufnimmt. Durch Oxidation und Reduktion werden Metallatome zu Ionen. Die positiv geladenen Ionen werden als Kationen und negativ die geladenen als Anionen bezeichnet. Der dabei entstehende elektrochemische Potenzialunterscheid zwischen unedlen und edlen Metallen bestimmt die Gleichgewichtsspannung einer Batterie. In Abbildung 2.2 ist die elektrochemische Spannungsreihe von unterschiedlichen Materialien dargestellt. Der daraus entstehende Potenzialunterscheid ergibt die gesamte Spannung der Batterie. Zum Beispiel wird bei Platin (ca. +1,5V) und Zink (ca. -0,75V) ein Potenzialunterscheid von 2.25V erreicht[Sch14].

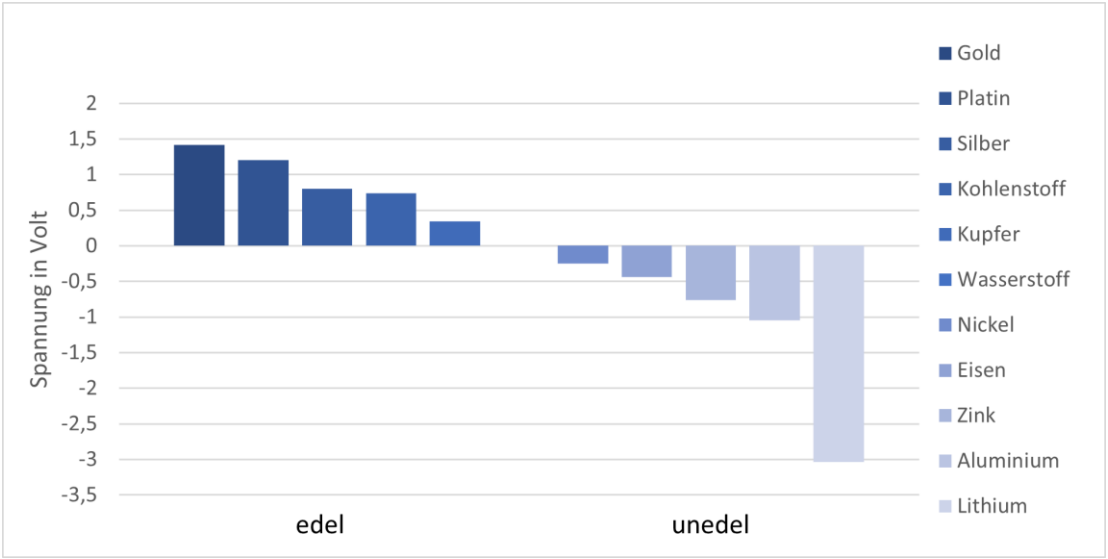


Abbildung 2.2:Auszug aus elektrochemischer Reihe in Anlehnung an [Sch14]

Bei einer Batterie werden Oxidation und Reduktion räumlich getrennt. Die räumliche Trennung wird mit Hilfe von zwei Elektroden ermöglicht und werden als Halbzellen bezeichnet[KD18]. Die zwei Elektroden werden durch eine äußere Verbindung mit einem Verbraucher für den Elektronenfluss verbunden. Die Elektroden bestehen aus dem jeweiligen Ableiter

(Anode/Kathode) und sind mit einem Aktivmaterial umwickelt. Für den Transport der Ionen wird ein Elektrolyt eingesetzt. Der Übergang zwischen Aktivmaterial und Elektrolyt hat eine besondere Bedeutung, da dort alle elektrochemischen Prozesse stattfinden. Das Aktivmaterial muss deshalb hoch porös sein, damit bei Ladungsdurchtritt nur kleine Spannungsabfälle entstehen und zudem muss es sehr leitfähig sein, um den Stromfluss zu gewährleisten. Außerdem werden die beiden Halbzellen durch eine poröse Trennwand voneinander isoliert. Die Trennwand wird als Separator bezeichnet, sie lässt nur Ionen durch und verhindert einen Kurzschluss zwischen den unterschiedlich geladenen Halbzellen. Beim Entladevorgang der Batterie wird die negativ geladene Elektrode als Anode (Hier findet die Oxidation statt) und die positiv geladene als Kathode (Hier findet die Reduktion statt) definiert. Über die äußere Verbindung fließen die Elektronen von der Anode zur Kathode. Die Kationen wandern von der Anode zur Kathode und die Anionen von der Kathode zur Anode. Beim Ladevorgang (Akkumulator) kehren sich diese Vorgänge um. Die Kathode wird zur Anode und die Anode zur Kathode[Sch14]. In der folgenden Abbildung 2.3 ist ein schematischer Aufbau bei einem Entladevorgang einer Batterie dargestellt:

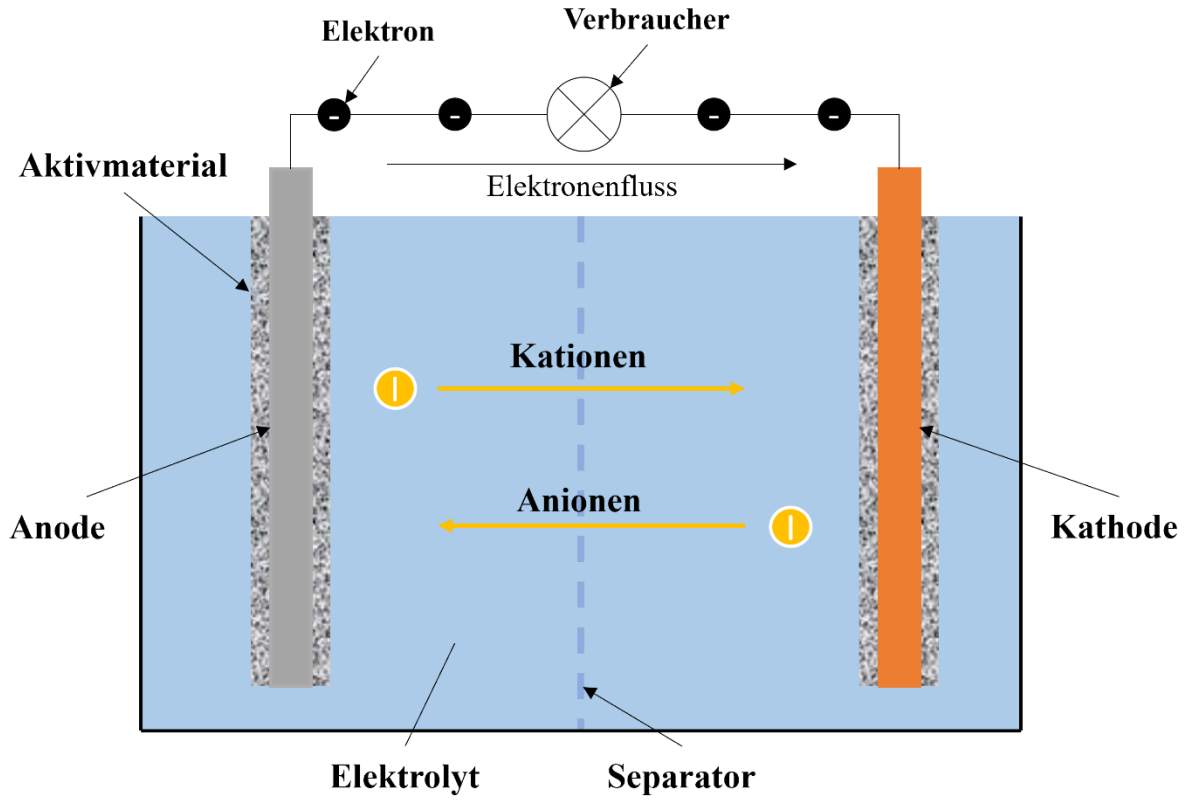


Abbildung 2.3: schematischer Aufbau eine Batterie beim Entladevorgang in Anlehnung an[JW19]

Eine wichtige messbare Größe einer Batterie ist die Klemmspannung. Sie hängt davon ab, ob die Batterie von außen belastet oder unbelastet ist. Im unbelasteten Zustand fließt kein Strom

und man spricht von der Ruhespannung (englisch: „open circuit voltage (OCV)). In der Realität weicht die Ruhespannung von der Gleichgewichtsspannung ab, da durch Nebenreaktionen wie z. B. Selbstentladung, Spannungsabfälle entstehen. In der Praxis wird der Unterscheid zwischen Gleichgewichtsspannung und Ruhespannung in den meistens Fällen vernachlässigt, da nur die Ruhespannung gemessen werden kann. Laut [KD18] besitzt einer Li-Ion-Batterie eine Selbstentladungsrate von etwa 0,5% pro Woche(bei 20°C) , 2% pro Woche (bei 40°C) und 4-10% pro Woche (bei 60°C). Die in der Arbeit verwendet Lithiumionen-Batterie hat z. B. eine geringe Selbstentladungsrate von 3% im Monat [Die13]. Bei einer belasteten Batterie hängt die Klemmspannung zusätzlich noch von der Überspannung ab, weil die kinetischen Vorgänge zwischen den Reaktionspartner nicht ideal verlaufen und die Leitfähigkeiten durch die unterschiedlichen Materialen begrenzt wird. Diese wird laut [JW19] in ohmsche Überspannung, Diffusions-, Durchtritts, Rektions- und Kristallisationsüberspannung eingeteilt. Beim Entladen einer Batterie ergibt sich eine negative Überspannung und beim Laden eine positive Überspannung.

$$OCV = U_{Gleichgewicht} + \sum U_{Überspannungen} \tag{1}$$

- **Ohmsche Überspannung**: Setzt sich aus dem Ableiter, dem Widerstand der Elektroden und dem Widerstand des Elektrolyts zusammen. Zusammen bilden sie den Innenwiderstand der Batterie ab.
- **Diffusionsüberspannung:** Wird durch die begrenzte Geschwindigkeit der Diffusion der an der Elektrodenreaktion beteiligten Teilchen innerhalb des Elektrolyts hervorgerufen.
- **Durchtrittsüberspannung:** Beschreibt die Lade-/ Entladereaktionen an den einzelnen Elektroden. Die Reaktionen finden an der sogenannten Doppelschicht, zwischen Elektrode und Elektrolyt statt, dort wo sich Ionen und Elektronen ein- bzw. auslagern.
- **Rektionsüberspannungen:** Werden durch vor- oder nachgelagerte Reaktionen verursacht.
- **Kristallisationsüberspannung:** Wird durch die Bildung von Kristallisationskeimen verursacht, da Ionen sich nur an bestimmte Stellen auf der Elektrodenoberfläche einlagern können.

Anzumerken ist hierbei, dass die Diffusions- und Reaktionsüberspannungen sehr ähnlich sind und deshalb die Reaktionsüberspannungen nicht immer separat betrachtet werde. Alle aufgelisteten Reaktionen sind zudem von der Temperatur abhängig[JW19].

In Abbildung 2.4 wird der Verlauf der Klemmspannung infolge einer Sprunganregung mit einer definierten Stromstärke entladen. Der Verlauf zeigt zu Beginn, dass durch den ohmschen Widerstand die Spannung nahezu senkrecht abfällt, gefolgt von einem exponentiellen Spannungsabfall durch den Ladungsdurchtritt zwischen Aktivmaterial und Elektrolyt, der Doppelschicht und Diffusion.

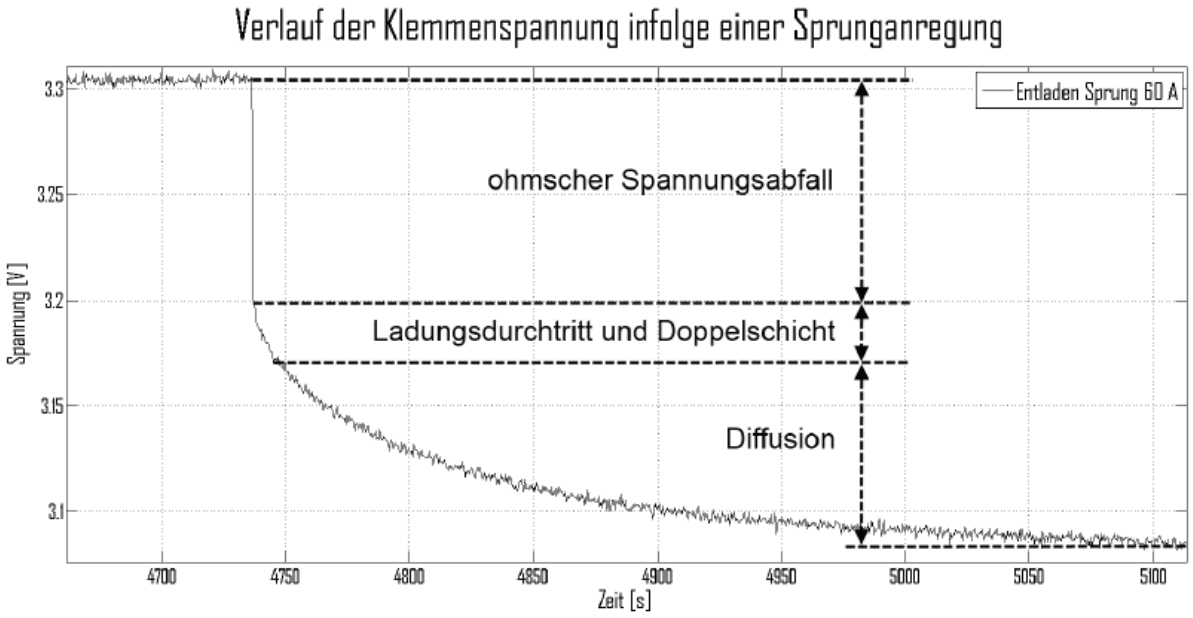


Abbildung 2.4: Spannungsverlauf einer Li-Ion-Zelle bei sprungartiger Entladung [Die13]

Die Doppelschicht ist eine Kapazität, die sich durch die enganliegenden Ladungsträger mit unterschiedlichen Potenzialen im Grenzbereich zwischen Elektrode und Elektrolyt aufbaut. In der Physik spricht man von einem Kondensator und in der Batterietechnik von einer Doppelschichtkapazität. Die sich zeitlich aufbauende Kapazität, welche die Dynamik einer Elektrode widerspiegelt, beeinflusst die Durchtrittsreaktionen und verhält sich wie eine Zeitglied erster Ordnung (RC-Glied). Da eine Zelle aus zwei Elektroden mit unterschiedlichen Oberflächen besteht, ergeben sich auch zwei in Reihe geschaltete RC-Glieder[JW19].

In Abbildung 2.5 ist ein vereinfachtes Ersatzschaltbild einer Batterie dargestellt, mit den vorher erwähnten RC-Gliedern und dem Innenwiderstand der Batterie. Je nachdem, wie genau die Schaltung das dynamische Verhalten der Batterie abbilden soll, können die Anzahl der RC-Glieder variiert werden[KKP20].

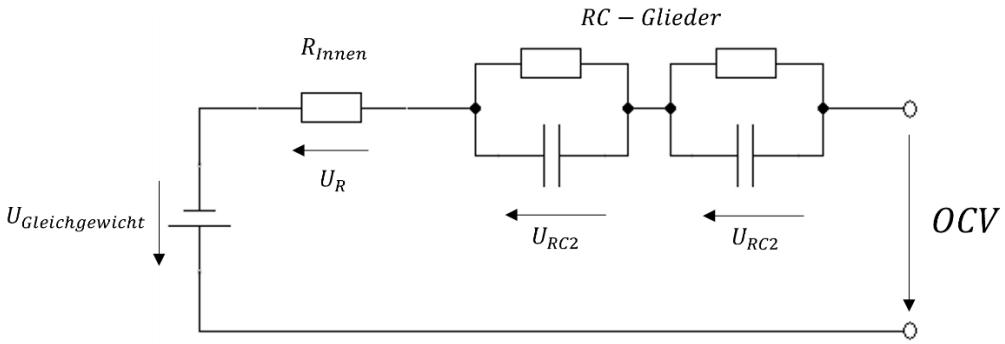


Abbildung 2.5: Vereinfachtes Ersatzschaltbild einer Batterie

Eine weitere wichtige Kenngröße ist die Kapazität [C]. Sie gibt die Ladungsmenge an, die mit einem konstanten Entladestrom über eine gewisse Zeit entnommen werden kann, bis die Entladeschlussspannung erreicht wird. Die Kapazität wird in Ampere pro Stunden [Ah] angegeben. Eine Batterie mit z. B. 60Ah kann mit einem konstanten Strom von 6A bis zu 10h genutzt werden. Die Kapazität ist stark von Temperatur, Entladestrom und Batteriealter abhängig. Die tatsächlich entnehmbare Kapazität sinkt mit steigendem Entladestrom und kann mit der „Peukert-Gleichung“ beschrieben werden, die von Wilhelm Peukert aufgestellt wurde. [KD18]. Eine hohe Temperatur bewirkt zudem, dass die Hauptreaktionen und Nebenreaktionen in einer Batterie schneller ablaufen. Die Leitfähigkeit des Elektrolyten steigt, Diffusionsvorgänge laufen schneller ab, was eine Erhöhung der entnehmbaren Kapazität zur Folge hat. Hohe Temperaturen bewirken zusätzliche negative Effekte, da die unterwünschten Alterungsprozesse verstärkt werden, was die Kapazität langfristig reduziert. Deshalb sollten hohe Temperaturen vermieden werden. Um unterschiedliche Batterien untereinander vergleichbar zu machen, wird die Nennkapazität [$C_N$] verwendet. Die Nennkapazität gibt die unter Nennbedingungen (Nenntemperatur, Nennstrom) entnehmbare Ladungsmenge bei einer neuen, vollgeladenen Batterie an. Es ist zu erkennen, dass die Stromlast einen wesentlichen Einfluss auf die Kapazität und Spannung hat. Um Batterien mit unterschiedlicher Kapazität besser vergleichen zu können, wird der Lade- und Entladestrom in normierter Form verwendet. Die normierte Form wird als Stromrate [C-Rate] bezeichnet. Der Strom wird in Verbindung mit der Nennkapazität gesetzt. Beispielsweise besitzt ein Akku, mit eine Nennkapazität von 6Ah bei einer Belastung von 6A, eine Stromrate von 1C[JW19], [Sch14].

Der Ladezustand (State of Charge (SoC)) bei einem Akku beschreibt die aktuelle zur Verfügung stehende Kapazität als Prozentangabe im Bezug zur Nennkapazität und dem Integral über den effektiven Lade-/Entladestrom:

$$SoC(t) = SoC(t = 0) + \frac{1}{C_{Nenn}} \int_{t=0}^{t} (I_{Batterie} - I_{Verlust}) dt \quad (2)$$

Eine Batterie mit einem SoC von 50% ist zur Hälfte geladen. Umgekehrt spricht man von der Entladetiefe (englisch: Depth of Discharge (DoD)), bei dem die entnommene Ladungsmenge angegeben wird. Die Batterie wäre dann zu 50% entladen. Anzumerken ist hier, dass bei einem SoC von 0% eine Batterie chemisch gesehen nicht leer ist, sondern die Entladeschlussspannung bzw. bei 100% die Ladeschlussspannung erreicht wird[TGP19].

Die Lade- und Entladeschlussspannung dürfen bei Batterien und Akkumulatoren nicht überschritten werden, da beim Überschreiten der Ladeschlussspannung, die Gefahr besteht, dass sich das Elektrolyt zersetzt und Gas freigesetzt wird. Die Batterie entgast und kann sich in seltenen Fällen selbst entzünden. Das Unterschreiten der Entladeschlussspannung bewirkt eine dauerhafte Beschädigung mit dem daraus entstehenden Kapazitätsverlust[TGP19].

Durch die vorher beschriebenen charakteristischen Kenngrößen einer Batterie ist das Laden und Entladen von wichtiger Bedeutung, so dass kritische Zustände vermieden werden. Die Aufgaben der Ladung lassen sich laut [JW19] unterteilen in:

- Vollständige Aufladung der Batterie (in der Regel möglichst schnell)
- Angleichung unterschiedlicher Zellzustände
- Minimierung von Alterungseffekten
- Aufhebung von reversiblen Effekten (z. B. „Memory Effekt“)

Die meisten Ladeverfahren setzen sich aus Konstantstrom- (englisch: constant current(CC))- und Konstantspannungs- (englisch: constant voltage (CV)) Kennlinien zusammen. Für das Umschalten bzw. Abschalten der einzelnen Kennlinien werden als Kriterien: „Zeit, Ladungsmenge, Temperatur und Strom“ verwendet[JW19]. Zusätzlich wird laut [JW19] bei einigen Ladeverfahren zwischen unterschiedlichen Ladephasen unterschieden:

- **Vorladung:** Die Batterie wird durch einen kleinen Strom hinsichtlich ihrer Funktionalität geprüft. Die Batterie muss in einer festgelegten Zeit eine vorgegebene Spannung erreichen, andernfalls wird das Laden abgebrochen. Diese Phase ist besonders wichtig bei tiefenentladenen Li-Ion-Batterien, damit kann überprüft werden,

ob ein Kurzschluss zwischen den Elektroden vorhanden ist und dadurch starke Erwärmungen vermieden werden.

- **Hauptladung**: In dieser Phase wird die Batterie durch den Strom geladen (Hauptreaktionen).
- **Nachladung:** Die Haupt- und Nebenreaktionen werden durch Verminderung der Ladegeschwindigkeit angepasst.
- **Erhaltungsladung:** Kompensation der Selbstentladung

In der folgende Abbildung 2.6 ist ein Ladeverfahren mit drei Phasen dargestellt:

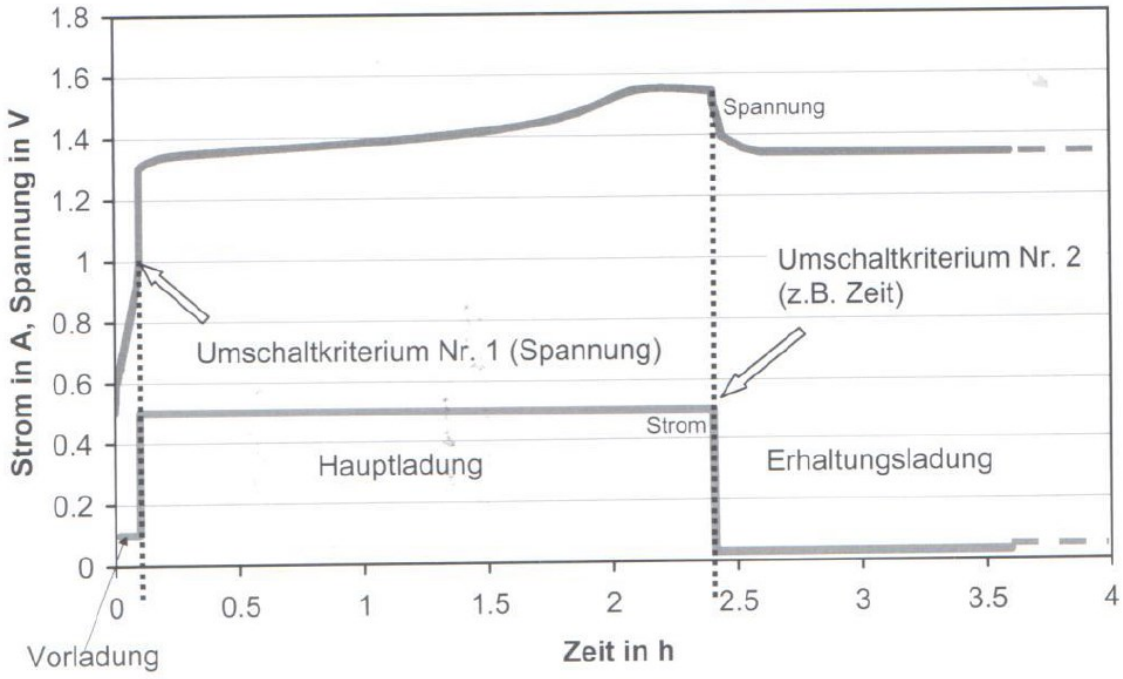


Abbildung 2.6: Beispiel für ein Ladeverfahren mit drei Ladephasen [JW19]

Die treibenden Kräfte in der Entwicklung von Akkumulatoren sind die Leistungs- und Energiewerte. Die Kennwerte können auf die Masse bezogen werden (spezifische Energie [Wh/kg], spezifische Leistung [W/kg]) oder auf das Volumen (Energiedichte [Wh/l], Leistungsdichte[W/l]). In der Elektromobilität wird z. B. in Richtung hoher Energiedichte geforscht. Es wird versucht eine hohe Kapazität bei kleinem Volumen und geringen Gewicht zu erzielen. Zurzeit bewegt man sich bei der Energiedicht zwischen 50 und 750 Wh/l[JW19].

In dieser Arbeit wird eine Li-Ion-Batterie verwendet, deshalb wird im folgendem noch einmal speziell auf die charakteristischen Eigenschaften, Vor- und Nachteile eingegangen.

Die Li-Ion-Batterie wird als Primär- und Sekundärbatterie angeboten. Das Atomgewicht von Lithium beträgt $7\frac{g}{mol}$, welches das leichteste Element im Periodensystem ist. Es besitzt eine Dichte von nur $0.53\frac{g}{m^3}$. Außerdem hat es das höchste chemische Standartpotential von uns bekannten Metallen. Diese Eigenschaften machen es zu einem der geeignetsten Elemente für viele Anwendungen in der Batterietechnik. Der größte Nachteil liegt in der hohen Reaktionsfreudigkeit mit Wasser. Das sich daraus bildende gasförmige Wasserstoff kann in extremen Fällen zur Entzündung führen[JW19]:

$$2\,\mathrm{Li} + 2\mathrm{H_2}0 \rightarrow 2\mathrm{Li0H} + \mathrm{H_2} \qquad (3)$$

Die erste weltweit kommerziell einsetzbare nicht wiederaufladbare Li-Ion-Batterie wurde von der Firma Sanyo aus Japan vor ca. 50 Jahren entwickelt. Sie kommen bis heute noch als Knopfzellen zur Anwendung. Die erste Wiederaufladbare Li-Ion-Batterie wurde in den 1980er Jahren auf den Markt gebracht. Wegen starker Sicherheitsprobleme verschwanden diese aber schnell wieder. 1991 wurde diese von Sony abgelöst. Sony machte die Li-Ion-Batterie durch das Verwenden von amorphem Kohlenstoff als Anodenmaterial und Lithiumkobaltdioxid als Kathodenmaterial sicherer und ebnete den Weg zum Erfolg der Li-Ion-Batterie. Seit dem wurden immer mehr Materialkombinationen erforscht, was die Energiedicht und Zellspannungen erhöhte[JW19]. In der nächsten Abbildung wird ein Überblick über die verschiedenen zur Anwendung kommenden Materialen für Elektroden gegeben:

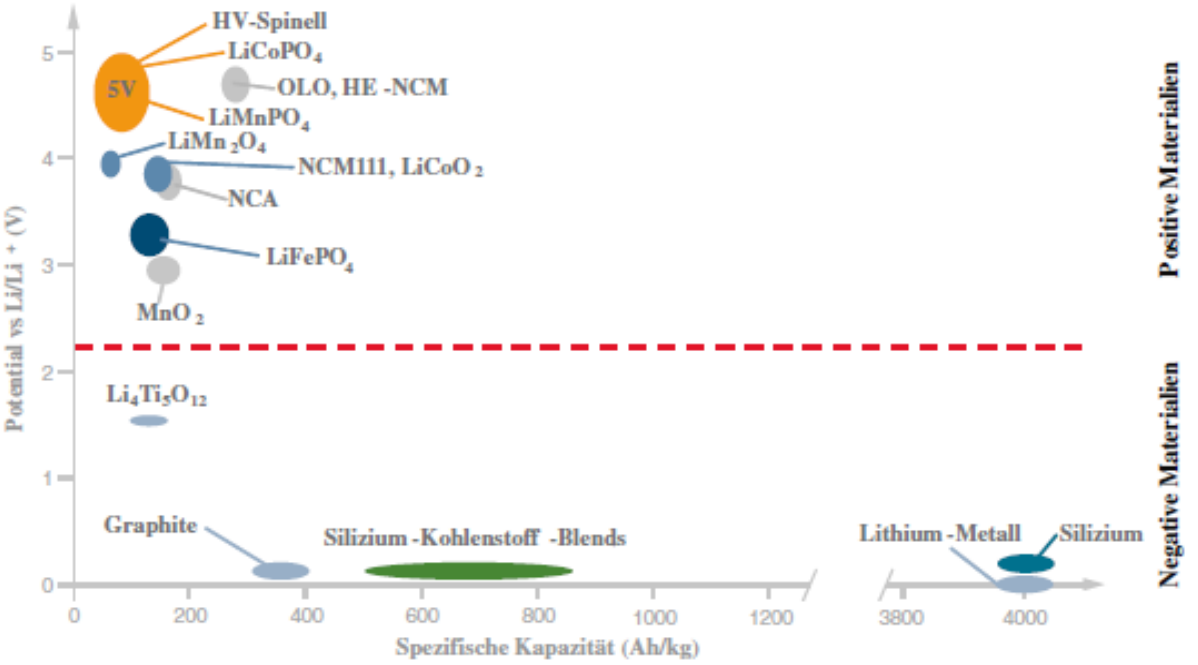


*Abbildung 2.7: Potential und spezifische Kapazität verschiedener positiver und negativer aktiven Materialien für Lithium-Ionen-Zellen*

Auf dem heutigen Markt wird zwischen Konsumanwendungen (bis wenige Ah) - und Traktionsanwendungen, stationären Speicher und Industrieanwendungen (10 bis ca. 500Ah) unterscheiden. Bei Konsumanwendungen steht der Mobiltelefonmarkt an vorderster Stelle mit etwa einer Milliarden Zellen pro Jahr, gefolgt vom Markt für Laptops und Tablets. Der am stärksten wachsenden Markt ist die Elektromobilität. Dort kommt die Li-Ion-Traktionsbatterien in Elektro- und Hybridfahrzeugen zur Anwendung. Dieser Markt wird auch in den nächsten Jahren noch von entscheidender Bedeutung sein[JW19].

Die Li-Ion-Batterie hat eine besondere Funktionsweise durch ihre sogenannten Interkalationselektroden. Die Elektroden besitzen ein hochporöses Aktivmaterial, welches das Wirtsgitter für Gastatome (Lithiumatome) bereitstellt. Die Lithium-Ionen werden dort in feste Plätze reversibel ein(interkalation)- und ausgelagert(deinterkalation). Die strukturellen Eigenschaften der Elektroden werden unwesentlich verändert. Diese Funktionsweise erhöht die Sicherheit und Zyklenlebensdauer[KD18], [Ko13]. Für die negativen Elektrode stehenden viele verschiedene Aktivmaterialien zur Verfügung (siehe Abbildung 2.7). Am Anfang wurde Lithium-Metall eingesetzt, doch es stellte sich schnell heraus, dass bei einem Lade- und Entladezyklus ein Lithiumverlust von ca. 0.3% entsteht. Um diesen Verlust ausgleichen zu können, wurden die die Elektroden überdimensioniert. Durch die überdimensionierten Elektroden sank gleichzeitig die praktische Energiedicht. Außerdem entstanden Nadeln aus Lithium (Dendriten) beim Wiederaufladen durch die Rückabschneidung der Lithium-Ionen an der Oberfläche zum Aktivmaterial. Die Dendriten wachen durch den Separator und führen zu einem Kurzschluss. Aus diesem Grund wird heute bei fast allen Li-Ion-Batterien ein auf Kohlenstoff basierendes Material verwendet, z. B. Graphit. Für die positiven Elektroden wird zum Beispiel Lithiumkobaltdioxid ($LiCoO_2$) benutzt. Es besitzt gute Kapazitätseigenschaften und eine hohe Zyklenstabilität. Diese Verbindung neigt aber bei Überspannungen und hohen Temperaturen zur Zersetzung. Um das zu verhindern, werden die Betriebsgrenzen mit Hilfe von Komponenten überwacht und gegebenenfalls geregelt. Ein weiteres Material ist Lithiummagandioxid ($LiMnO_2$). Es weist eine geringere Energiedichte als $LiCoO_2$ auf, kann aber durch das sichere Verhalten bei Überspannungen und hohen Temperaturen punkten. Zudem ist Mangan kostengünstig, umweltschonend und reichlich verfügbar. Das große Problem liegt in der Tiefenentladung, da es hier zu beschleunigten Alterungsprozessen kommt und dadurch ein erhöhter Kapazitätsverlust entsteht. In dieser Arbeit wird eine Li-Ion-Batterie auf der Grundlage von Lithiumeisenphosphat (LiFePO4) verwendet. Diese weist eine sehr hohe thermische Stabilität, gute Verfügbarkeit (Rohstoffe) und ein exzellentes Alterungsverhalten auf, und ist preisgünstig. Das Hauptproblem besteht in der geringen elektrischen und ionischen

Leitfähigkeit. Um diesem Problem entgegenzuwirken, werden die Aktivmaterialpartikel mit Kohlenstoff beschichtet und die Partikelgröße vom Aktivmaterial wird verkleinert (Nanometer). Durch diese Modifikationen hat das Kathodenmaterial ein gutes Hochstromverhalten und kommt deshalb in vielen Traktionsanwendungen zum Einsatz[JW19]. Für den Transport der Lithium-Ionen wird ein organischer, wasserfreier Elektrolyt verwendet. Durch den Verzicht auf Wasser, leidet die Leitfähigkeit. Um diesen Nachteil auszugleichen, wird ein zusätzliches Leitsalz eingeführt. Diese Leitsalz liefert Lithium-Ionen und stellt die Leitfähigkeit sicher. Zusätzlich werden noch Additive beigemischt, welche die Aufgabe haben, z. B. das Zersetzen des Elektrolyts bei Überladungen zu verringern oder die Entzündbarkeit zu erschweren[JW19]. Eine Besonderheit liegt in der sehr flachen Spannungskennline über dem SOC-Bereich von ca. 20% bis 80%. Die konstante Klemmspannung ist sehr gut für den Betrieb geeignet, jedoch ist die Bestimmung des SOC aufwendiger. Die Klemmspannung kann nicht mehr eindeutig dem jeweiligen Ladungszustand zugeordnet werden.

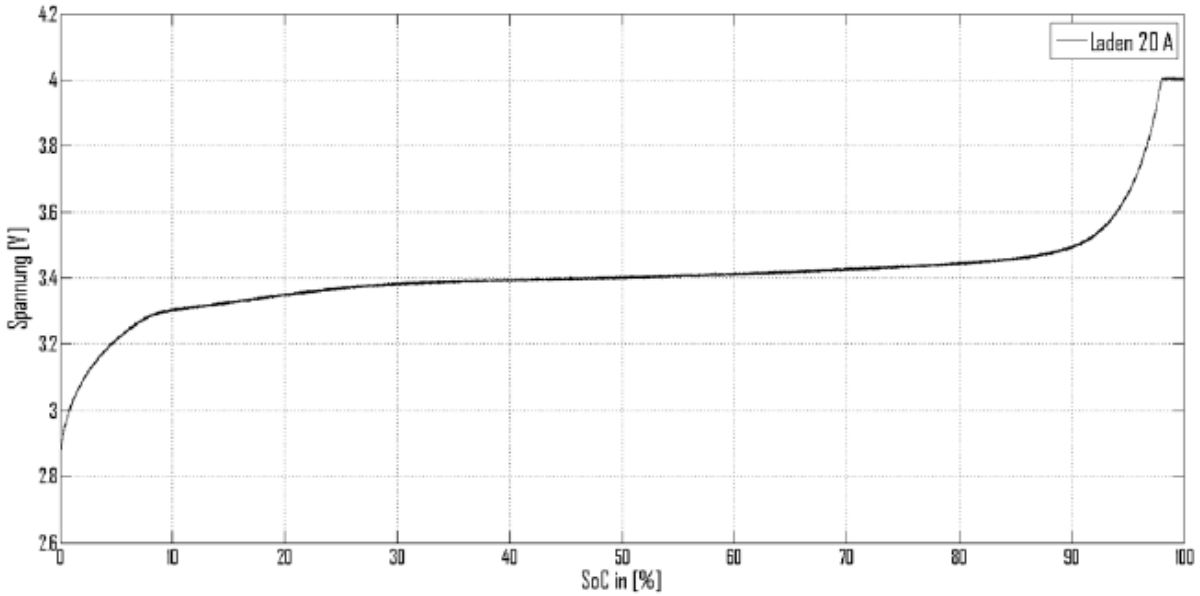


Abbildung 2.8: Verlauf der Klemmspannung bei einem Ladevorgang mit 20A[Die13]

Es gibt verschiedene Methoden zur Bestimmung des SoC. Zu den Methoden gehört zum Beispiel die „OCV-Messung“. Bei dieser Methode macht man sich den linearen Zusammenhang zwischen der Ruhespannung und dem SoC zunutze:

$$U_{OC}(t) = a_1 * SoC(t) + a_0 \quad (4)$$

Der SOC gibt hierbei den aktuellen Ladezustand und $U_{OC}(t)$ die aktuelle OCV an. der Parameter $a_1$ ist die OCV bei SoC(t) = 0% und $a_0$ bei SoC(t) = 100%. Die Messung ist sehr einfach, aber kann nur im unbelasteten Zustand und nicht am Anfang bzw. Ende einer OCV

angewendet werden, da diese Bereiche stark nichtlinear sind (siehe Abbildung 2.8). Außerdem ist es besonders schwer bei Lithiumeisenphosphat-Batterien, wie in dieser Arbeit benutzt wird, da auch, wie schon erwähnt wurde, ein sehr flache OCV im Bereich von ca. 20% und 80% SoC herrscht [Ch13], [JW19].

Die nächste Methode wird „Coulomb Counting“ genannt. Bei dieser Methode wird der Entladestrom gemessen und zusätzlich der Verluststrom geschätzt. Die Differenz wird dann über die Zeit integriert. Am Anfang muss ein Initialwert vom SoC vorhanden sein. Die Berechnung erfolgt mit Hilfe von Gleichung (2). Aufgrund von Ungenauigkeiten bei der Messung/Schätzung durch Nebenreaktion und Temperaturänderungen entsteht ein Fehler. Dieser Fehler wird mit integriert und wird umso größer, je länger die Ladezyklen auseinander liegen[Ch13], [JW19][Ch13]LINK "#_.

Außerdem gibt es noch modellbasierte Anzätze, bei dem ein Ersatzschaltbild(siehe Abbildung 2.5) die Haupt- und Nebenreaktionen der Batterie abbildet und daraus der SoC ermittelt wird. Der „Kalman-Filter“ gehört zu den meist verbreitesten Ansätzen. Bei dieser Methode wird ein zusätzlicher Beobachter(Ersatzschaltbild) zum realen System geschaltet. Die gemessenen Ausgangsgrößen werden mit den berechneten Ausgangsgrößen vom Ersatzschaltbild verglichen. Aus den Abweichungen wird das Modell immer wieder angepasst und der jeweilige SoC berechnet. Die folgende Abbildung zeigt den schematischen Kreislauf eines Kalman-Filter:

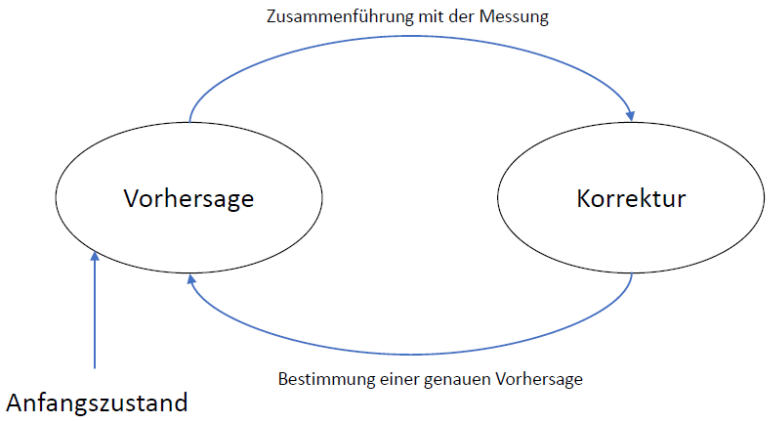


Abbildung 2.9: Kalman-Filter-Kreislauf

Es gibt noch viele weiter Methoden zur Ermitteln des SoC, z. B. im Bereich der künstlichen Intelligenz. In [Ch13] wird eine Übersicht von verscheidenen Methoden/Ansätze vorgestellt.

## 2.2 Neuronale Netze

### 2.2.1 Biologische Grundlagen

Künstliche neuronalen Netze sind dem Zentralnervensystem (Gehirn) von Menschen und Tieren nachempfunden. Das Nervensystem nimmt über das sensorische System, über Körperteile, die Informationen auf und gibt sie an das motorische System weiter, um Bewegungen zu koordinieren. Die Informationsverarbeitung findet durch die Neuronen statt. Im menschlichen Körper existieren ca. 100Milliarden Neuronen, welche teilweise gleichzeitig und miteinander interagieren können[Kr15].

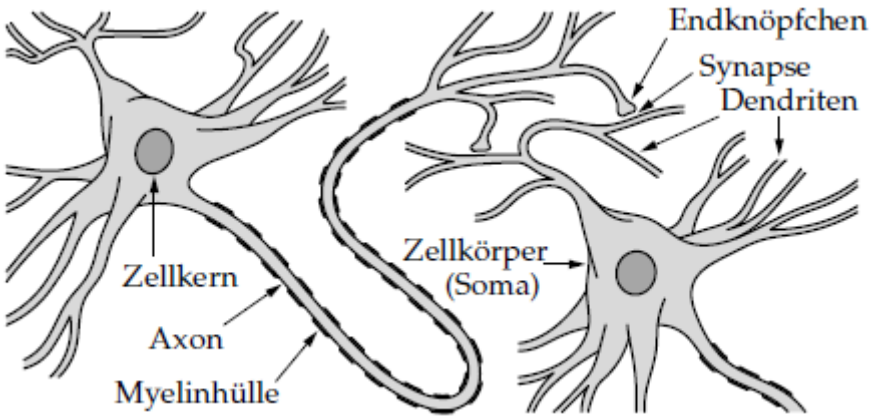


Abbildung 2.10: Prototypischer Aufbau biologischer Neuronen[Kr15]

Die Grundfunktion eins Neurons besteht darin, elektrische Aktivitäten zu akkumulieren und weiterzuleiten. Das Neuron besteht aus einem Zellkern, das durch einen Zellkörper (Soma) umschlossen ist. An dem Zellkern befinden sich ein Axon als Pfad für die Kommunikation mit anderen Neuronen. Am Ende des Axons befindet sich eine starke Verästelung, die an allen enden ein Endknöpfchen besitzt. Die Endknöpfe berühren fast benachbarte Dendriten und sondern Chemikalien (Neurotransmitter) in den Leerbereich (Synapse) ab. Die Neurotransmitter verändern die Polarisation von den Dendriten und können so Informationen weiterleiten. Neuronen können bis zu ca. 10000 von diesen Verbindungen im Menschen haben. Die Konzentration von negativ geladen Ionen ist außerhalb von Neuronen größer als im Inneren. Durch das Ausschütten der Neurotransmitter kann das Spannungspotenzial verringert (exzitatorisch (erregend)) oder erhöht (inhibitorisch (hemmend)) werden. Diese Effekte summieren sich auf. Übersteigt die Spannung in der Zelle einen Schwellwert, wird ein Impuls über das Axon an die verbundenen Neurone verschickt. Das Absenden von einem Impuls wird auch als „feuern“ bezeichnet. Außerdem setzt der Impuls das standartmäßige

Potenzialunterschied von -70mV wieder her. Die Impulsgeschwindigkeit wird über die stärker von der Myelinhülle gesteuert. Die Beschreibung ist stark vereinfacht[Kr15].

### 2.2.2 Künstliche Neuronale Netze

#### Künstliches Neuron

Künstliche Neuronale Netze versuchen, dass in Kapitel 2.2.1 beschrieben biologische Vorgehen, nachzubilden. Die KNN ist ein Teilgebiet der Künstlichen Intelligenz und erfordert eine interdisziplinare Zusammenarbeit von vielen Fachgebieten, z. B. Informatik, Mathematik, und Ingenieurswissenschaften. Durch gezieltes Training eines KNN auf spezielle Aufgaben, kann es gelernte und auch neue Situationen meistern. Das KNN besteht in den meisten Fällen aus vielen künstlichen Neuronen. In der folgenden Abbildung 2.11 ist ein künstliches Neuron abgebildet:

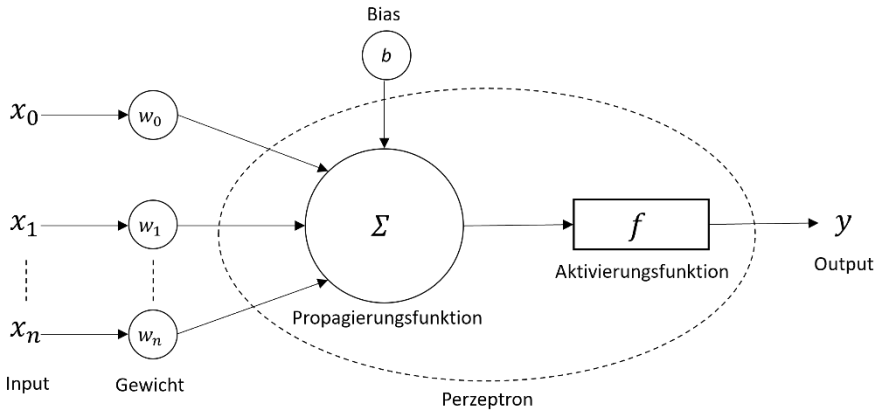


Abbildung 2.11: Künstliches Neuron (Perzeptron)

Die eingehenden Signale sind auf der linken Seite abgebildet x = [$x_1$, $x_1$, …, $x_n$] und sind zeitgleich die Ausgänge von zuvor anderen Neuronen. Die Eingänge werden zusätzlich mit Gewichten w = [$w_1$, $w_1$, …, $w_n$] versehen. Die Gewichte stellen die Wichtigkeit der eingehenden Signale da und können durch einen Lernalgorithmus angepasst werden. Die Funktion entspricht in der Biologie, der von Synapsen, wo die Information überbracht und durch die Neurotransmitter erregt oder gehemmt werden. Im nächsten Schritt werden die gewichteten Eingänge summiert und müssen einen bestimmten Schwellwert überschreiten, damit das Neuron durch die Aktivierungsfunktion aktiviert wird bzw. feuert. Der Schwellwert kann außerdem über ein „Bias-Neuron“ (b) definiert werden. Im echten neuronalen Netz würde der Schwellwert ein Spannungspotenzial darstellen, der überwunden werden muss, bis ein Impuls verschickt wird. Sind all die Informationen im Neuron angekommen, werden die

Informationen durch die Propagierungsfunktion zusammengefasst. Schlussendlich wird durch eine Aktivierungsfunktion ein neuer Zustand ausgegeben[Kr15].

**Propagierungsfunktion**

Die Propagierungsfunktion (Netzeingabe ($net_j$)) übernimmt die eingehenden Inputs ($x_i$), multipliziert sie mit den Gewichten($w_{ij}$) und übergibt die Summe an die Aktivierungsfunktion:

$$net_j = \sum_{i=1} w_{ij} * x_i \tag{5}$$

**Aktivierungsfunktion**

Die Aufgabe der Aktivierungsfunktion besteht darin, die Netzeingabe auf einen bestimmten Bereich zu normieren und ein gewünschtes Aktivitätslevel einzustellen. Die Normierung bewirkt, dass die Ausgabewerte in sinnhaften Intervallen von z. B. [0,1] oder [-1,1] gehalten werden und dadurch sinnlose Rechenoperationen auf Grund von infinitesimal kleinen Veränderungen zu unterbinden. Außerdem wird das Anwachsen der immer weiter aufsummierenden Eingabewerte unterbunden, damit das Netz schneller und genauer konvergiert[SKK10]. Die am häufigsten verwendeten Aktivitätsfunktion sind in der folgenden Abbildung 2.12 dargestellt:

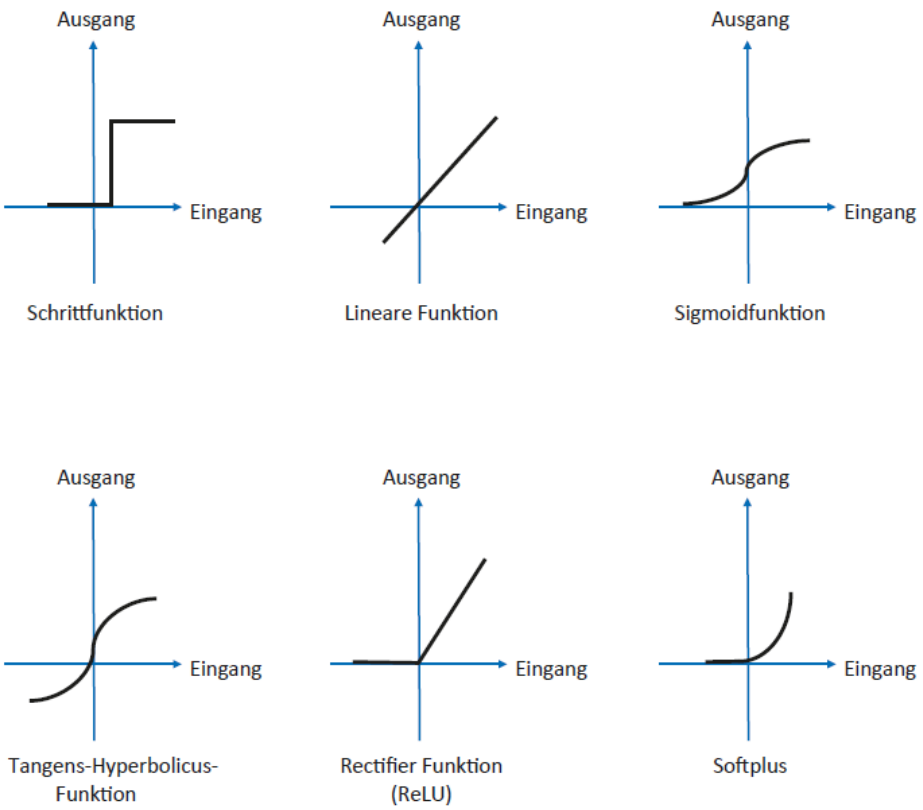


Abbildung 2.12: Aktivierungsfunktionen [Ba21]

Durch die Schrittfunktion kann die Aktivität eines Neurons mit einem bestimmten Schwellwert gesteuert werden. Das Neuron wird erst ab einer bestimmten Summe der Netzeingabe aktiviert:

$$net_j = \begin{cases} 1 \; für \sum_{i=1} w_{ij} * x_i > \; \Theta_j \\ 0 \; sonst \end{cases} \tag{6}$$

Die lineare Funktion erzeugt einen linearen Zusammenhang zwischen Eingang und Ausgang. Durch einen zusätzlichen Schwellwert kann das Aktivitätslevel angehoben werden, um z. B. Rauschen zu unterdrücken. Die Sigmoidfunktion begrenzt das Aktivitätslevel zwischen [0,1] und ist an allen Stellen differenzierbar, was die Durchführbarkeit von weiteren Berechnungen vereinfacht. Die Tangens-Hyperbolicus-Funktion hat einen Wertebereich von [-1,1] und verhält sich wie die Sigmoidfunktion. Die Rectifier-Funktion funktioniert wie ein Gleichrichter in der Elektrotechnik. Sie lässt allen positiven Wert durch und sperrt negative. Im Gegensatz zur ReLU ist die Softplus schnell differenzierbar und in der Nähe von 0 glatt[Ba21].

Für die Auswahl einer geeigneten Aktivierungsfunktion gibt es keine universelle Regel und beruht in den meisten Fällen auf Erfahrungen in den verschiedenen Anwendungsgebieten. Die Lineare Funktion wird oft für Regressionsprobleme verwendet und die ReLU zur Erkennung von Texten und Objekten in Bildern. Die Sigmoid- und Tanh-Funktionen spielt später beim Lernverfahren, durch ihre Differenzierbarkeit und Nichtlinearität, eine wichtige Rolle[Ba21].

**Bias**

Das Bias ermöglich eine Veränderung des Aktivitätslevel eins Neurons und ermöglicht eine höhere Flexibilität beim Erlernen von Problemen. Angenommen das Ergebnis der Propagierungsfunktion ist $net_j = -0.6$ und als Aktivierungsfunktion wird ReLU verwendet. Das Ergebnis ist 0 und das Neuron feuert nicht. Durch das hinzufügen eines Bias mit dem Wert von 1 wird ein neues Aktivierungslevel erreicht (-0,6+1=0,4). Das Neuron ist nun in der Lage, auch bei negativen Werten, Einfluss auf weitere Vorgänge im Netz zu haben, wodurch die Anpassungsfähigkeit flexibler wird.

**Neuronales Netz**

Verbindet man nun mehreren Neuronen, entsteht ein künstliches neuronales Netz. Außerdem werden die einzelnen Neuronen nochmal in Input-, Hidden- und Output-Neuronen unterteilt. Die Neuronen werden auch als Knoten oder Units bezeichnet und in Schichten, den sogenannten Layer angeordnet. Laut [Ba21] besitzen die Layer folgende Aufgaben:

- **Input Layer**: Die Eingabeschicht repräsentiert die Eingabedaten in das neuronale Netz, also die gewünschten Daten
- **Hidden Layer:** Zwischen den Neuronen der Eingangsschicht und deren Ausgangsschicht befinden sich eine oder auch mehrere versteckte Schichten. Gibt es mehr als nur eine versteckte Schicht, handelt es sich um ein sogenanntes tiefes neuronales Netz, auch bekannt unter dem Begriff Deep Neural Network.
- **Output Layer:** Die Ausgangsschicht ist mit den Neuronen der versteckten Schicht verbunden und repräsentiert das Ergebnis der Informationsverarbeitung.

### 2.2.3 Netzstrukturen und -arten

Wie schon im vorigen Kapitel erwähnt, kann durch mehrere Neuronen ein KNN erzeugt werden. Dabei gibt es die verschiedensten Strukturen für die unterschiedlichsten Anwendungen. Die zwei am häufigsten vorkommenden Netzstrukturen sind das vorwärts gerichtete Netz und das Netz mit Rückkopplungen.

#### Das Perzeptron

Das Perzeptron ist ein einschichtiges neuronales Netz und wurde durch die Funktionsweise der Retina (Netzhaut) motiviert. Es kann mit Hilfe von zwei Inputs, binären Operationen (OR, AND, NOT, NAND, NOR) erlernen und einfache lineare Probleme lösen. Die folgende Abbildung 2.13 veranschaulicht diese Logischen Operationen:

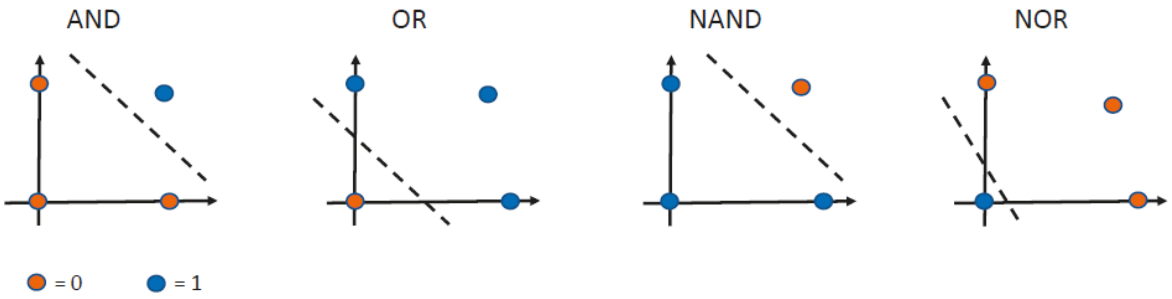


Abbildung 2.13: Logische Operationen für das Perzeptron [Ba21]

Wie zu erkennen, lassen sich die zwei Netzeingaben durch eine Gerade klar voneinander trennen. Die Grundversion besteht aus einem künstlichen Neuron (siehe Abbildung 2.11). In der Praxis sind jedoch viele Probleme nicht genau separierbar und erfordern nichtlineare Operationen. Das Perzeptron gelangt zum Beispiel bei einer XOR-Operation an seine Grenzen, was die Abbildung 2.14 verdeutlichen soll:

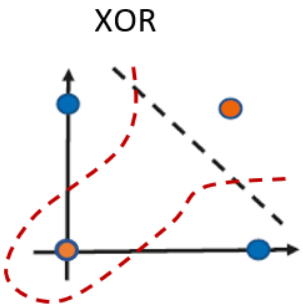


Abbildung 2.14: XOR-Operation

Die zwei unterschiedlichen Daten lassen sich nur mit einer nichtlinearen Funktion separieren. Aus diesem Grund werden mehrschichtige Perzeptronen (Multilayer Perceptron(MLP) zum Lösen von komplexen Problemen verwendet, man spricht dann auch von einem Feed-Forward-Netzwerk[Ba21].

**Feed-Forward-Netzwerk (FFN)**

Die einfachste Struktur beinhaltet eine Eingabeschicht, versteckte Schicht und Ausgabeschicht. Die Anzahl der Neuronen von der Eingabe und Ausgabeschicht werden durch die vorgegebenen Probleme bestimmt. Die Anzahl der versteckten Schicht, sowohl auch die Anzahl der Neuronen in der Schicht können beliebig verändert werden. Die Informationen werden ausschließlich in Vorwärtsrichtung weitergleitet und alle Neuronen sind mit allen darauffolgenden Neuronen verbunden[Ba21]. In folgenden Abbildung 2.15 ein FFN mit einem Input-, Hidden- und Output-Layer.

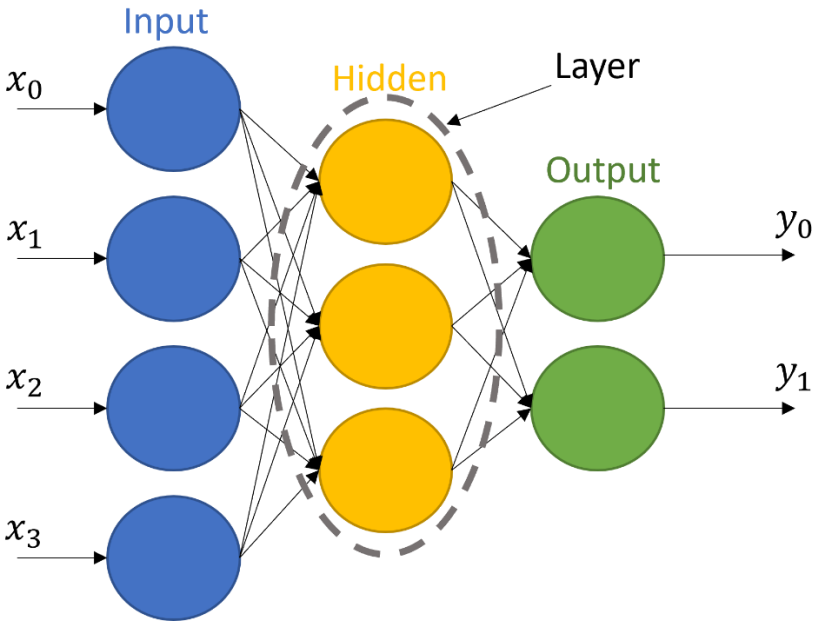


Abbildung 2.15:Beispiel eines Feed-Forward-Netzwerk

**Rekurrentes neuronales Netzwerk (RNN)**

Bei einem RNN können durch Rückkopplungen zwischen Neuronen Informationen im Netz bleiben und in zukünftigen Schritten berücksichtigt werden. Es ist eine Art von Kurzzeitgedächtnis im Netz. Des Weiteren kann die Entwicklungen von Informationen betrachtet und durch erkannte Muster vom Netz, Vorhersagen erzeugt werden. Laut [Ba21] unterscheidet man zwischen den Rückkopplungsarten:

- **Direkte Rückkopplung (direct feedback):** Der Neuronen-Ausgang wird als weiterer Eingang genutzt. Es entsteht eine Verbindung von einem Neuron zu sich selbst.
- **Indirekte Rückkopplung (indirect feedback):** Sie verbindet den Ausgang mit einem Neuron der vorhergehenden Schicht.
- **Seitliche Rückkopplung (lateral feedback):** Der Ausgang des Neurons wird mit einem Neuron derselben Schicht verbunden.
- **Vollständige Verbindung:** Jeder Neuronen-Ausgang hat eine Verbindung zu jedem anderen existierenden Neuron.

In der nächsten Abbildung 2.16 sind drei wichtige Rückkopplungen veranschaulicht:

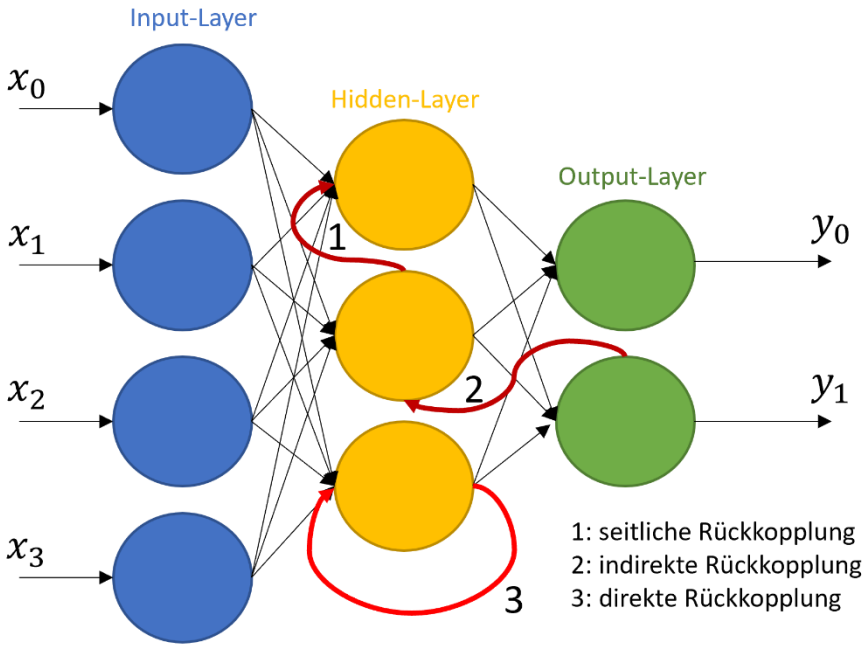


Abbildung 2.16: Beispiel eines Rekurrenten neuralen Netzwerks in Anlehnung an [Ba21]

In dieser Arbeit werden zeitdiskrete Zeitreihen mit dynamischen Vorgängen für das Training eines Neuralen Netzes verwendet. Deshalb wird im Folgenden ein spezielles RNN mit dynamischen Eigenschaften vorgestellt.

**NARX-Netzwerke**

Ein „nonlinear autoregressive network with exogenous inputs (NARX)“ ist eine abgewandelte Form eines RNN und kann für nichtlineare zeitdiskrete Systeme verwendet werden. Durch das Hinzufügen von Zwischenspeichern (tapped delay lines(TDL)) am Eingang und an der Rückkopplung vom Ausgang zum Eingang, können vergangene Zeitschritte dem aktuellen Zeitschritt beigefügt werden. Die Zwischenspeicher repräsentieren somit das charakteristische Zeitverhalten des Systems. Durch die Rückführung (Feedback) von der Systemantwort an den Systemeingang, kann das neuronale Netz die Eigendynamik abbilden. Das NARX-Netzwerk wird mit der allgemeinen Gleichung(6) nach [HDB02] beschrieben:

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u)) \qquad (7)$$

Der Parameter u(t) stellt die Eingangsgrößen dar und $n_u$ gibt an, wie viele Zeitschritte aus der Vergangenheit mit einbezogen werden. Außerdem wird der Output y(t) mit den vergangenen Zeitschritten $n_y$ in den Eingang eingespeist. Die Inputs gelangen über die TDL in das Netz und werden in ein FFN geleitet, verarbeitet und ausgegeben.

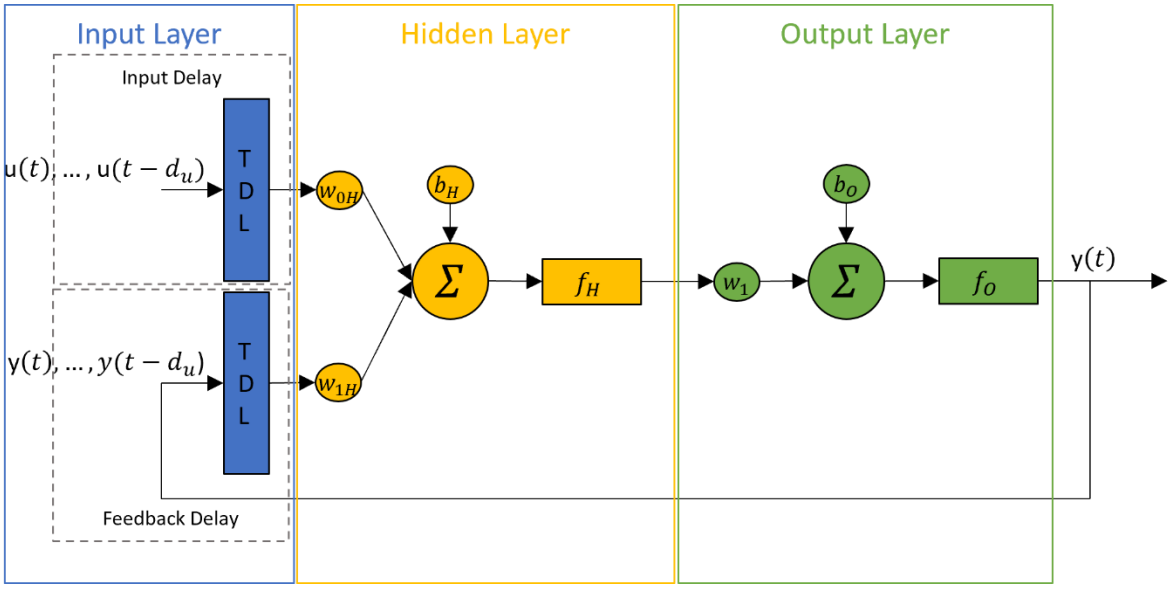


Abbildung 2.17: Vereinfachte Darstellung eines NARX-Netzwerks

Aus dieser Struktur kann die folgende Gleichung (7) für den Ausgang eine Zeitreihe aufgestellt werden:

$$y(t+1) = f_0\left[b_0 + \sum_{n=1}^{H} w_1 f_H\left(b_H + \sum_{n=1}^{d_u} w_{oH} u(t-d_u) + \sum_{n=1}^{d_y} w_{1H} y(t-d_y)\right)\right] \qquad (8)$$

### 2.2.4 Lern- und Trainingsverfahren

Um einem neuronalen Netz bestimmte Aufgaben beizubringen, gibt es unterschiedliche Lernverfahren. Die zwei wichtigsten sind:

- **Überwachtes Lernen**: Das KNN lernt durch einen bekannten Output. Es wird dann auch von sogenannten „gelabelten Daten“ gesprochen.
- **Unüberwachtes Lernen**: Der Output ist unbekannt. Das KNN versucht durch Zusammenhängen in den Daten, sein Output zu definieren.

Nachdem man das passende Lernverfahren ausgewählt hat, kann man den dazu passenden Algorithmus auswählen. Beim Überwachten Lernen gibt es zwei Möglichkeiten:

- **Klassifikation:** Die bekannten Daten werden mit Hilfe vom KNN durch erkennbare Merkmale in vordefinierte Klassen geordnet. Unter Klassen versteht man z. B. Bilder (Hund und Katze)[SK19].
- **Regression:** Es werden Zusammenhänge zwischen abhängigen- und unabhängigen Variablen gesucht, um damit eine vorgegebene Funktion zu beschreiben und daraus Vorhersagen zu tätigen.

Der Algorithmus beim Unüberwachten Lernen ist:

- **Clustering:** Die ungekannten Daten werden anhand von verborgenen Mustern automatisch in Kategorien unterteilt und in Gruppen (Clustern) zusammengefasst. Die Cluster werden durch das KNN vorgegeben. Dieser Algorithmus kann z. B. beim Finden von neuen chemischen Verbindungen hilfreich sein[SK19].

Das Lernen eines KNN wird in zwei Phasen unterteilt. In der Trainingsphase werden die Gewichte durch ein Trainingsalgorithmus angepasst. Die Gewichte vom KNN werden so lange optimiert, bis das vorgegebene Problem angemessen nachgebildet wird. In der Testphase wird das KNN auf unbekannte Daten angewendet, um zu schauen, ob es auch diese, wie in der Trainingsphase, nachbilden kann. Der Erfolg in der Testphase hängt davon ab, wie gut die Generalisierungsfähigkeit, also die Anpassungsfähigkeit an unbekannte Daten ist. Das hängt von vielen Faktoren ab. Ist die Anzahl der Trainingsdaten zu gering, besteht die Gefahr einer Überanpassung (engl.: Overfitting). Dabei lernt das KNN die Zusammenhänge der Daten auswendig und verliert seine Generalisierungsfähigkeit bei unbekannten Daten. Andersherum führt eine geringe Anzahl von Daten dazu, dass die Zusammenhänge nicht optimal erlernt werden (Unteranpassung). Ein weiterer Faktor ist die Komplexität der Struktur in einem KNN.

Besteht das KNN aus vielen Neuronen und versteckten Schichten, kann es zu einer Überanpassung führen. Genauso kann es zu einer Unteranpassung kommen, wenn die Struktur eines KNN bei komplexen Problemen zu einfach gewählt wurde. Für die Anpassung der Gewichte gibt es, je nach Komplexität, unterschiedliche Algorithmen. Bei einem Neuronalen Netz mit überwachtem Lernen und versteckten Schichten kommt der Backpropagation-Algorithmus zum Einsatz. Die Gewichte werden durch rekursives Vorgehen optimiert. Dabei wird der entstehende Fehler eines KNN als Funktion, in Abhängigkeit aller anliegenden Gewichte, betrachtet[LÄ20]:

$$E(w_j) = E(w_{1j}, w_{2j}, \dots, w_{nj}) \tag{9}$$

Die daraus entstehende Fehlerfunktion(wird häufig auch als Kostenfunktion bezeichnet [Ba21]) stellt einen mehrdimensionalen Raum(Hyperebene) dar. Die Aufgabe besteht nun darin, die Gewichte so zu optimieren, dass das globale Minimum der Fehlerfunktion erreicht wird. Für dieses Vorhaben wird das Gradientenabstiegsverfahren verwendet. Durch das partielle Ableiten der mehrdimensionalen Fehlerfunktion nach den Gewichtsvariablen, wird ein Gradient erzeugt, der mit Betrag und Richtung den stärksten Anstieg repräsentiert. Im ersten Schritt wird der Fehler aus der Summe von der Differenz zwischen Ist-Output $t_j$ vom Netz und Soll-Output $o_j$ berechnet[LÄ20]:

$$E = \frac{1}{2}\sum_j (t_j - o_j)^2 \tag{10}$$

Dabei wird die Differenz quadriert, damit sich negative und positive Werte nicht aufheben. Das $\frac{1}{2}$ wird zu Vereinfachungen, zwecks späterer Ableitung, eingeführt. Im nächsten Schritt wird die Fehlerfunktion, nach den jeweiligen Gewichten für die Optimierung, partiell abgeleitet [LÄ20]:

$$\Delta w_{ij} = -\lambda \cdot \frac{\partial E}{\partial w_{ij}} \tag{11}$$

Die Variable $\lambda$ gibt die Lernrate an. Sie bestimmt die Änderungsrate der Gewichte. Das Minus konvertiert den eigentlichen Anstieg in einen Abstieg, um das Minimum der Fehlerfunktion zu erreichen. Dieses Vorgehen, wird ausgehend von der letzten Schicht, bei jeder vorigen Schicht durchgeführt. Die Fehlerterme werden also rückfährts durch das KNN geführt („**Back**“-Propagation)[LÄ20].

Durch die Vereinfachung, die Hyperebene auf einen Gradienten zu reduzieren, führt dies zu mehreren Problemen. Die Probleme werden in der nächsten Abbildung 2.18 visualisiert:

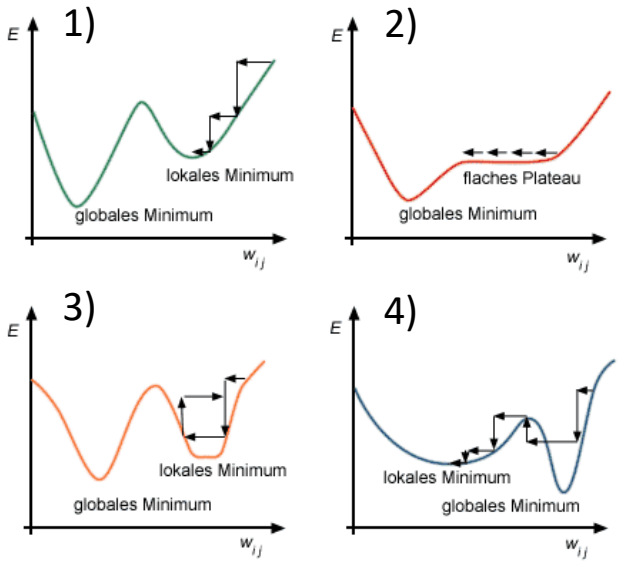


Abbildung 2.18: Probleme bei Backpropagation[CP21]

1. **Lokale Minima**: Beim Optimierungsversuch wird nur ein lokales Minimum erreicht, obwohl ein besseres Ergebnis erzielt werden könnte (globales Minimum).
2. **Flaches Plateau:** Das Plateau führt zu einem Stagnieren der Optimierung. Das Training braucht sehr lang.
3. **Oszillation:** Die Gewichtsänderung schwingt über einem Minimum hin und her und verhindert die optimale Anpassung.
4. **Änderungsbetrag:** Ein zu großer Betrag bei einer Gewichtsänderung kann dazu führen, dass schmale Minima übersprungen und höher liegende erreicht werden.

Diese Probleme können durch die Veränderung der Lernrate, sowie Veränderung und Art der Initialisierung der Gewichte gelöst werden[Ba21]. Es gibt jedoch keine eindeutige Lösung, deshalb gibt es eine Vielzahl von Trainingsalgorithmen, die im Grunde auf dem Prinzip der Backpropagation beruhen, aber für verschiedenste Anwendungsfälle optimiert wurden. Eine Übersicht von verschiedenen Trainingsalgorithmen kann [MW21] entnommen werden.

### 2.2.5 Anwendungsmöglichkeiten für KNN

Die Anwendungsmöglichkeiten von KNN ist sehr vielfältig. Sie kommen bei der Bild-, Sprach-, Muster-, Schrifterkennung, Frühwarnsystemen und Simulation von komplexen Systemen zum Einsatz[Ba21].

In der Industrie werden KNN zur Prozessoptimierung verwendet. Die Anforderungen an die Produkte werden immer komplexer. Die Kunden wollen immer mehr Individualität, sodass auch logistische Anforderungen steigen. Der Informationenfluss zwischen Produktion, Lieferant und Kunde steigt und kann nur noch begrenzt mit gängigen Verfahren gemeistert werden. Das KNN kann die Informationen verarbeiten und damit z. B. Lieferwege, Lagerbestände optimieren und somit auch den gesamten Prozess.

Fehler bei Maschinen können, noch bevor sie entstehen, vorhergesagt werden. Der Verschleiß oder auch der komplette Ausfall kann verhindert werden. Die Produktion kommt nicht zum Stillstand und die daraus entstehenden Kosten können vermieden werden. Außerdem sind weniger Sensoren, und damit weniger Technik erforderlich, da viel Parameter einfach aus älteren Daten vorhergesagt werden können. Zum Beispiel werden Windräder und Stromnetze mit Drohnen auf äußere Auffälligkeiten untersucht und die Daten automatisch ausgewertet.

Eine weitere spannende Anwendung sind die Deepfakes. Dabei werden originale Gesichter in Bildern oder Videos durch künstliche Gesichter (Deepfakes) ersetzt. In der Filmproduktion werden künstliche Bilder mit Hilfe von CGI (Computer Geneated Imagery) erzeugt. Die Deepfakes sind eine kostengünstige Alternative[Ah21]. Diese Technik hat aber auch Schattenseiten. Sie kann für kriminelle Handlungen eingesetzt werden, um vertrauenswürdige Falschinformationen weiterzugeben.

Die KNN finden deshalb auch immer mehr Einzug in der Cypersecurity. Sie wird zur Erkennung von Malware oder Bots, Pishing und Vorhersage von Hackerangriffen genutzt. In China wird die Technik aber auch außerhalb eingesetzt. Die Menschen werden durch Sicherheitskameras in der Öffentlichkeit überwacht und bei Straftaten können direkt Maßnahmen ergriffen werden.

In der Automobilbranche werden KNN für die optimale Straßenführung und das autonome Fahren eingesetzt.

# 3 Allgemeine Methodik/Rahmbedingungen

Für die Vorgehensweise beim Implementieren eines KKN gibt es keine allgemeingültige Vorgehensweise. Es beruht vielmehr auf Erfahrungen und erfordert in vielen Fällen eine experimentelle Herangehensweise für eine gute Performance. Es gibt viele verschiedene Tools und Leitfäden, die beim implementier unterstützen.

## 3.1 Vorgehensweise beim Erstellen eines Neuronalen Netzes

Im ersten Schritt beim Implementieren eines neuronalen Netzes wird, ausgehend von der Problemstellung, eine Netzarchitektur ausgewählt. Dabei wird zwischen überwachtem und unüberwachtem Lernen unterschieden (siehe 2.2.4). In dieser Arbeit wird sich ausschließlich mit dem überwachten Lernen befasst. Außerdem müssen alle Hyperparameter initialisiert werden. Als Hyperparameter werden diejenigen Parameter bezeichnet, die vor Beginn der Trainingsphase initialisiert werden müssen. Dabei kann es sich z. B. je nach Netz um die Anzahl der Neuronen, Hidden-Layer oder Aktivierungsfunktionen handeln. Nachdem eine passende Netzarchitektur ausgewählt wurde, müssen alle erforderlichen Trainingsdaten für das Training angelegt werden. Beim Erstellen der Daten sollte sich ausreichend Zeit genommen werden, denn die Qualität beeinflusst maßgeblich das Ergebnis. Die Trainingsdaten sollten deshalb viele unterschiedliche Betriebszustände beinhalten, sodass das KNN immer neue Informationen erlernen kann und das Auswendiglernen vermieden wird. Dadurch kann eine hohe Generalisierungsfähigkeit bei unbekannten Daten erreicht werden. Neben den Trainingsdaten müssen noch zusätzliche Testdaten erstellt werden, die nicht in den Trainingsdaten vorkommen. Die Testdaten werden für die spätere Validierung des KNN genutzt. Im nächsten Schritt wird das KNN mit den Trainingsdaten trainiert. Der Trainingsalgorithmus passt die Gewichte so lange an, bis das KNN die Daten ausreichend erlernt hat. Im letzten Schritt wird die Generalisierungsfähigkeit anhand der Testdaten getestet. Ist das Ergebnis nicht zufriedenstellend, müssen die Hyperparameter angepasst und weitere Trainingsdurchläufe durchgeführt werden. Dieser Iterationsprozess wird so lange durchgeführt, bis ein optimales Ergebnis erzielt wird[LÄ20]. Die Vorgehensweise ist aber keine Garantie dafür, dass ein KNN das vorliegende Problem optimal lösen kann. In der folgenden Abbildung 3.1 wird der Entwicklungsprozess eines KNN zusammengefasst veranschaulicht:

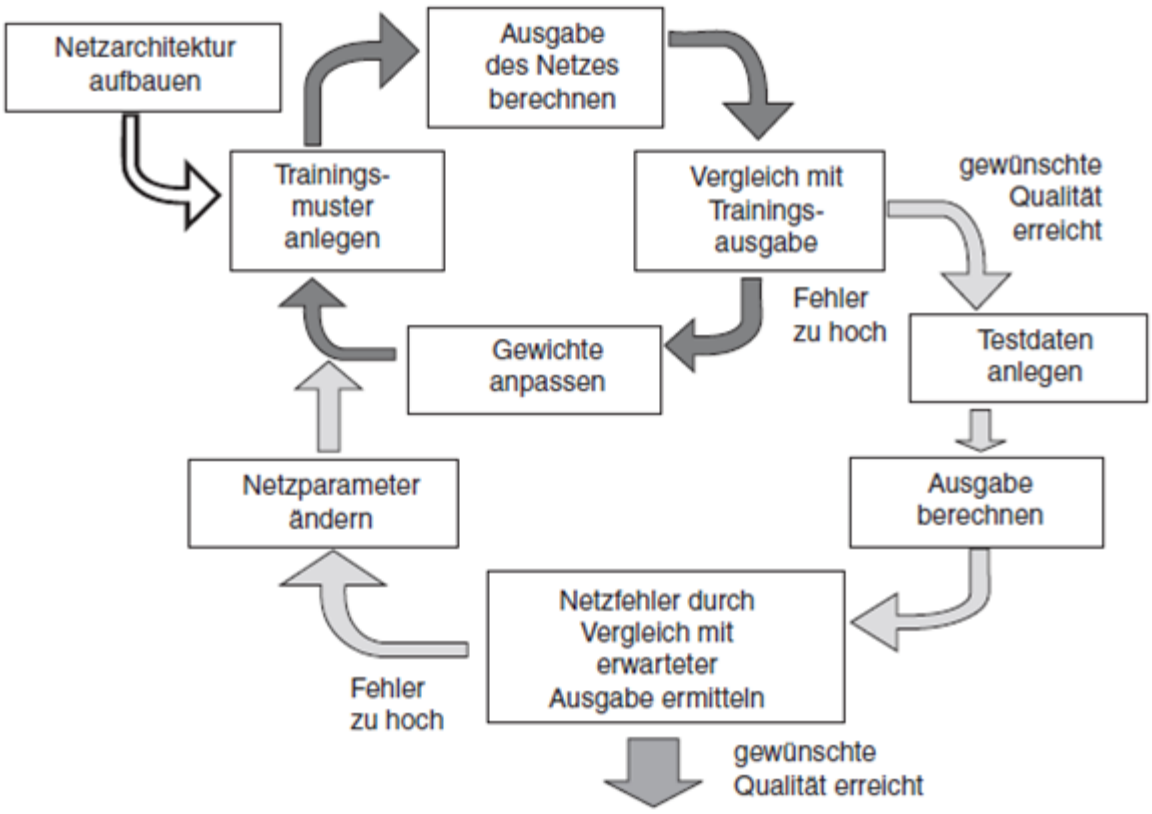


Abbildung 3.1: Entwicklung eines künstlichen neuronalen Netzwerks[LÄ20]

## 3.2 Anforderungen

Die Anforderungen an ein KNN kann je nach Anwendungsfall variieren. In dieser Arbeit sollen mit Hilfe von Zeitreihen Prognosen erstell werden. In diesem Fall ist eine gute Generalisierungsfähigkeit unerlässlich. Dadurch sollen mit zukünftigen unbekannten Daten genaue Vorhersagen ermöglicht werden. Dabei soll das KNN auch bei nicht linearen Vorgängen mit einer hohen Eigendynamik nicht an seine Grenze stoßen. Für die Bewertung der Genauigkeit eines KNN gibt es eine Vielzahl von mathematische Bewertungsmethoden. In Literaturen kommen die nachfolgenden Methoden am häufigsten zum Einsatz:

Der mittlere absolute Fehler (MAE) gibt die gemittelte absolute Abweichung zwischen dem aktuellen $y_1$ und vorhergesagten $\hat{y}$ Output an:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}| \tag{12}$$

Mittlere quadratische Fehler (MSE) gibt die gemittelte quadrierte Differenz an:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{13}$$

Der „Root Mean Square Error (RMSE)“ ist die Wurzel aus dem MSE:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \tag{14}$$

Das Bestimmtheitsmaß oder R-Quadrat ($R^2$) ist ein Maß dafür, wie gut die Varianz des Soll-Outputs($y_1$) durch die Varianz des vorhergesagten Ist-Outputs ($\hat{y}_1$) erklärt werden kann. Das Ergebnis wird zwischen 0 und 1 angeben. Im Nenner steht die aufgeklärte Varianz der Vorhersage und im Zähler die Gesamtvarianz des Soll-Outputs. Dabei wird sich immer um den Mittelwert des Soll-Outputs bezogen. Bei einem Ergebnis von 1 können dann die Varianz des Soll-Outputs zu 100% durch die Ist-Varianz erklärt werden:

$$R^2 = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{15}$$

Bei der Verwendung von MSE und RMSA werden starke Abweichungen stärker bestraft. MSE bietet den Vorteil, dass der Fehler leichter zu interpretieren ist, da es die gleiche Einheit wie vom Output des KNN besitzt. Bei Daten mit starken Ausreißern bietet sich dagegen der MAE an, da die Auswirkungen von Ausreißern kompensiert werden. Die Anforderung an ein KNN ist, dass in der Testphase die Werte von MAE, MSE und RMSE auf das möglichste Minimum gesenkt werden. Wohingegen beim Bestimmtheitsmaß ($R^2$) ein Maximum angestrebt wird. Für die Bewertungen gibt es noch zahlreiche weitere Bewertungsmaße[MRR21]. Eine weitere Anforderung geht an die Trainingsdaten. Diese müssen ein breites Spektrum von Zuständen des zu lösenden Problems abdecken und eine hohe Varianz untereinander aufweisen, um die besagte Generalisierungsfähigkeit für unbekannte Daten zu gewährleisten. Ein gleichbleibendes Verhalten der Daten über einen größeren Bereich sollte deshalb stehts vermieden werden.

# 4 Konzeption der Methode zur Parametereinflussanalyse/Versuchsaufbau

Wie schon zu Beginn erwähnt, besteht die Aufgabe in dieser Arbeit darin, optimale Hyperparameter für ein KNN auszuwählen und damit den SoC eine LiFePO4-Batterie zu schätzen. Die LiFePO4 hat eine sehr flache OCV im Bereich zwischen ca. 20% bis 80% des SoC (siehe Abbildung 2.8). Dies macht die Zuordnung des SoC zum OCV in diesem Bereich sehr ungenau und konventionelle Berechnungsmethoden ermöglichen kein zufriedenstellendes Ergebnis. Der SoC wird deshalb mit Hilfe eines KNN geschätzt. Unter Verwendung eines vorhanden Batteriemodelles werden Daten erstellt und für das Training und Tests verwendet.

## 4.1 Auswahl der Netzart

Die Auswahl eins Netzes wurde durch die zur Verfügung stehenden Möglichkeiten in Matlab begrenzt. Durch Literaturrecherche zu ähnlichen Problemstellungen mit Vorhersage von zeitdiskreten dynamischen Zeitreihen wurde das NARX-Netzwerk ausgewählt. Das NARX erweist sich durch seinen einfachen Aufbau und der Betrachtung von Dynamik am Output/Input durch die Rückkopplung als optimale Kandidat. Im Kapitel 2.2.3 wurde schon das grundlegende Funktionsprinzip erläutert. Das Training eines neuronalen Netzes erfolgt in zwei wesentlichen Phasen ab. In der ersten Phase wird das Netz im Openloop trainiert. Im Openloop ist das Netz ein reines FFN. Die Rückkopplung wird durch einen reinen Input ersetz, welches zusätzlich den Soll-Output ins Netz eingibt. Durch die entstehende Architektur wird immer nur ein Schritt in die Zukunft vorhergesagt. Die dabei berechneten Gewichte durch den Trainingsalgorithmus stabilisieren das Netz. Danach wird dem Netz die Rückkopplung hinzugeführt (Closeloop) und die Gewichte aus dem Openloop werden als Initialisierung verwendet[HDB02]. In der nächsten Abbildung werden die zwei unterschiedlichen Architekturen veranschaulicht:

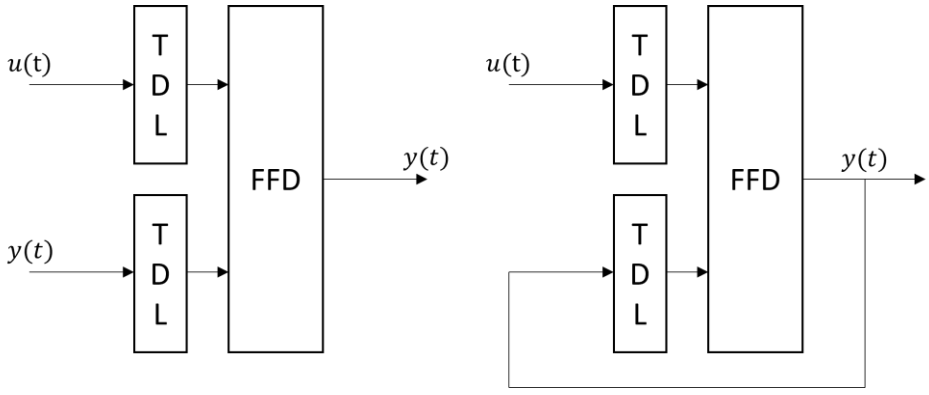


Abbildung 4.1: Links: Openloop; Rechts: Closeloop

## 4.2 Vorauswahl der Hyperparameter

Für das Training des NARX -Netzwerks müssen im Vorfeld die erforderlichen Hyperparameter festgelegt werden. Dazu gehören die Aktivierungsfunktionen für die Hidden-Layer und dem Output-Layer. Im Kapitel 2.2.2 wurden schon die gängigen Funktionen aufgelistet. In den folgenden Schritten wird die Aktivierungsfunktion für die Hidden-Layer ausgewählt. Theoretisch können alle Funktionen für das Netz verwendet werden. In den Literaturen werden standartmäßig nichtlineare Aktivierungsfunktionen verwendet. Deshalb fällt die Wahl auf die Tangens-Hyperbolicus-Funktion, da diese differenzierbar ist und der Wertebereich zwischen -1 und 1 begrenzt wird. Der Output wird standartmäßig mit einer linearen Funktion versehen. In Matlab wird diese mit dem Namen „Pureline“ bezeichnet. Der Wertebereich ist nicht begrenzt, sodass der Output alle beliebigen Werte annehmen kann[HDB02]. Als Trainingsalgorithmus wird der Levenberg-Marquardt-Algorithmus (LM) verwendet. Dieser erwies sich schon in vielen Literaturen als zuverlässiger Backpropagation-Algorithmus durch sein schnelles und gutes Konvergenzverhalten[HM94], [Qi19], [ZZZ]. Die wichtigsten Parameter für die Performance für das NARX-Netzwerk sind die Zeitverzögerungen für die TDL des Inputs (Inputdelay (ID) und der Rückführung (Feedbackdelay (FD)), sowie die Anzahl der Neuronen in Hidden- und Outputlayer. Zusätzlich können noch die Anzahl der Hiddenlayer erhöht werden. Diese Paramater sind vorher nicht festlegbar und müssen experimentell ermittelt werden. Dazu wird im nächsten Kapitel eine Versuchsplanung zur Ermittlung erstellt. Um die Wertebereiche der Parameter einzugrenzen, werden anhand von Literaturen, vergleichbare Lösungsansätze verglichen. Der ID wird vorläufig auf den Bereich von 0 bis 5 und der FD auf 1 bis 5 festgelegt[KS17], [WQ17], [LIP18]. Die Anzahl der Neuronen und Layer sollte so gering wie möglich gehalten werden, da größere Netze eine höhere Tendenz zum Overfitting haben [KSC19]. In der Praxis gibt es kaum Probleme, die mehr als zwei Hidden-Layer erfordern. Es besteht vielmehr die Gefahr zum Overfitting, da bereits im Training das Netz die Daten perfekt auswendig lernen könnte und die Generalisierungsfähigkeit auf unbekannte Daten verloren geht.

Ein Netz mit zwei Hidden-Layern kann laut [He05] jede Funktion mit beliebiger Form darstellen:

Tabelle 4.1: Determining the Number of Hidden Layers[He05]

| Number of Hidden Layers | Result |
|---|---|
| 0 | Only capable of representing linear separable functions or decisions. |
| 1 | Can approximate arbitrarily with any functions which contains a continuous mapping from one finite space to another. |
| 2 | Represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. |

Deshalb wird erstmal die Anzahl der Hidden-Layer auf maximal drei begrenzt. Zusätzlich wird aus Formel (8) ersichtlich, dass durch die weiteren Terme ein höherer Rechenaufwand beim Training vermieden wird und letztendlich auch der Backpropagation-Algorithmus wenige verarbeiten muss. Außerdem sollten deshalb auch immer erst die Anzahl der Neuronen im Hidden-Layer Schrittweis erhöht werden, bis ein akzeptables Ergebnis erreicht wird. Die Anzahl wird vorerst auf maximal 20 begrenzt, welcher sich als guter Anhaltswert durch Literaturrecherchen erwies[KSC19], [Qi19].

## 4.3 Versuchsplanung

Vor Beginn des eigentlichen Trainings muss die Anzahl der Trainingsversuche pro Trainingsdurchlauf ermittelt werden. Dies ist erforderlich, da die Initialisierung der Gewichte zufällig erfolgt. Dadurch wird vermieden, dass der Trainingsalgorithmus beim Optimierungsversuch in einem lokalen Minimum stecken bleibt. Es werden dafür mehrere Versuche mit einem gleichbleibenden Netz durchgeführt. Die Trainingsversuche werden dann, beginnend mit 10, in 10er-Schritten erhöht. Die Trainingsversuche werden je Schritt 5-mal wiederholt, damit Ausreißer kompensiert werden. Mit Hilfe des Mittelwerts der jeweiligen Versuche wird dann die optimale Anzahl der Trainingsversuche ermittelt. Darauffolgend werden die Faktoren (ID, FD und N) mit Hilfe eines vollfaktoriellen Versuchsplans gegenübergestellt und die beste Zusammenstellung für das NARX-Netzwerk ausgewählt:

Tabelle 4.2: Faktoren für den vollfaktoriellen Versuchsplan

| Bezeichnung | Faktor | Min | Max |
|---|---|---|---|
| ID | Input-Delay | 0 | 5 |
| FD | Feedback-Delay | 1 | 5 |
| N | Neuronen (Hidden-Layer) | 1 | 20 |
| H | Hidden-Layer | 1 | 3 |

Die vier Faktoren mit den jeweiligen Wertebereichen würde zu einer Anzahl von $6 * 5 * 20 * 3 = 1800$ führen. Um den Umfang zu verringern, werden bei der Versuchsreihe erstmal die Faktoren ID, FD und N berücksichtigt. Zusätzlich wird die Anzahl der Neuronen in 2er-Schritten erhöht. Daraus ergibt sich ein Versuchsumfang von 300 Versuchen. Außerdem kommen noch die ermittelten Trainingsversuche je Versuch dazu. Diese werden aber zu einem späteren Zeitpunkt ermittelt. Zu guter Letzt werden die Ergebnisse aus den Versuchen mit Hilfe des RMSE verglichen. Der RMSE hat einen hohen Bekanntheitsgrad bei der Analyse von KNN und besitzt die gleiche Einheit wie vom Output, sodass die Interpretation der Ergebnisse erleichtert wird. Außerdem werden Abweichungen durch die quadratische Fehlergewichtung stärker bestraft (siehe 3.2).

## 4.4 Automatisierte Versuchsdurchführung

Das manuelle Einstellen der einzelnen Hyperparameter für die jeweiligen Versuche und die spätere Auswertung würde sehr viel Zeit in Anspruch nehmen. Deshalb wird der komplette Ablauf von Initialisieren der Hyperparameter bis hin zum Training automatisiert. Der vollfaktorielle Versuchsplan wird mit der „Statistik Toolbox“ in Matlab erstellt. Zu Beginn werden die Daten eingelesen und müssen für die Weiterverarbeitung in ein standardmäßiges Zellenarray für das Netz konvertiert werden. Daraufhin werden die einzelnen Versuche mit den entsprechenden Werten der Faktoren mit Hilfe einer For-Schleife einer Funktion übergeben, die das Training des NARX-Netzwerks enthält. Dort werden die festgelegten Hyperparameter und die erforderliche Netzeinstellung initialisiert. Anschließen wird das Netz als erstes im Openloop und anschließend im Closeloop trainiert. Das Zwischenergebnis sowie das Gesamtergebnis werden nach den Durchläufen gespeichert.

Das folgende Flussdiagramm zeigt noch einmal den schematischen Ablauf:

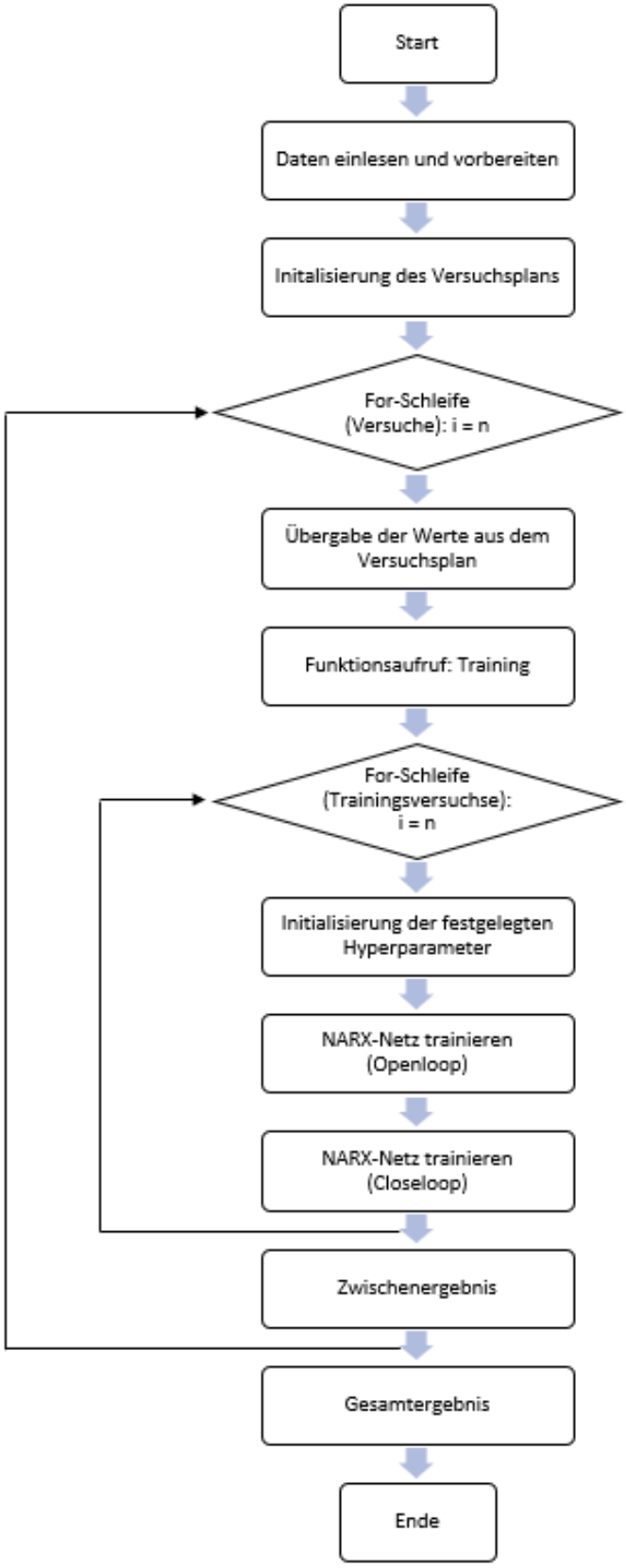


Abbildung 4.2: Der automatisierte Trainingsablauf eines NARX-Netzwerks

## 4.5 Auswahl von Trainings- und Testprofil

Für die Profile der Daten werden die Parameter Spannung(V), Strom(I) und SoC verwendet. Die Erstellung erfolgt mit dem bereitgestellten Batteriemodell von der Ostfalia. Die dabei verwendete Batteriezelle von der Winston Battery (jetzt:Thunder Sky Winston Battery) ist vom Typ WB-LYP60AHA. Die Kathode besteht aus LiFeYP$O_4$, eine weiterentwickelte Form der LiFeYP$O_4$. Die zusätzliche Dotierung mit Yttrium sorgt für ein höheren Schutz an den Betriebsgrenzen der maximalen Belastbarkeit der Batterie[OA17]. In der folgenden Tabelle sind die technischen Spezifikationen der Batterie aufgelistet:

Tabelle 4.3: Technische Spezifikationen der Batterie WD-LYP60AHA[OA17]

| **Parameter** | **Wert** | |
|---|---|---|
| Typ | TSWB-LYP60AHA | |
| Nennkapazität | 60Ah | |
| Spannungsbereich | 2.8V …. 4.0V | |
| Maximaler Entladestrom | Konstant | ≤ 3C (180A) |
| | Impuls | ≤ 10C (600A) |
| Standard Lade-/ Entladestrom | 0.5C (30A) | |
| Betriebstemperatur | -45°C …. 85°C | |
| Gewicht | 2.3kg | |
| Abmessungen ($B\ x\ H\ x\ T$) | $(115\ x\ 203\ x\ 61)\ mm$ | |

Basierend auf dem Batteriemodell kann anhand eines Stromlastprofils das dazugehörige Spannungs- und SoC-Profil simuliert werden. Für das Stromprofil werden viele verschiedene Profile für die Trainingsdaten kombiniert, um möglichst alle Betriebsbereiche der Batterie abzubilden. Zum Abbilden der Betriebsbereiche werden Fahrzyklen aus der Automobilbranche verwendet. Durch Recherchen im Internet konnte nur eine begrenzte Anzahl von Profilen gefunden werden, daher wird mit Simulink und der Erweiterung „Powertrain-Blockset“ das Stromprofil erstellt. Die Erweiterung beinhaltet viele unterschiedliche Fahrzyklen. Das Problem besteht aber darin, dass die Fahrzyklen in Abhängigkeit der Beschleunigung oder Leistung angegeben sind. Deshalb ist eine Skalierung der Fahrzyklen auf den Arbeitsbereich der Batterie erforderlich. Wichtig ist dabei, dass das gesamte Trainingsprofil sehr stark variiert, damit das Netz keine Bereiche auswendig lernt. Zusätzlich wird bei der Erstellung der Trainingsprofile darauf geachtet, dass die Lade- und Entladephase sowie auch größere Sprünge zwischen einigen Bereichen im SoC abgebildet werden. Durch die Sprünge soll das

Konvergenzverhalten bei schlagartiger SoC-Änderung beobachtet werden. Dies kann besonders in den SoC-Bereichen von 20% oder 80% durch das exponentielle Verhalten der OCV auftreten (siehe Abbildung 2.8). Abschließend wird die Generalisierungsfähigkeit mit zwei unbekannten Fahrzyklen getestet: der „Dynamik-Stress-Test" (DST) und der „Worldwide Harmonized Light-Duty Vehicles Test Procedure" (WLTP):

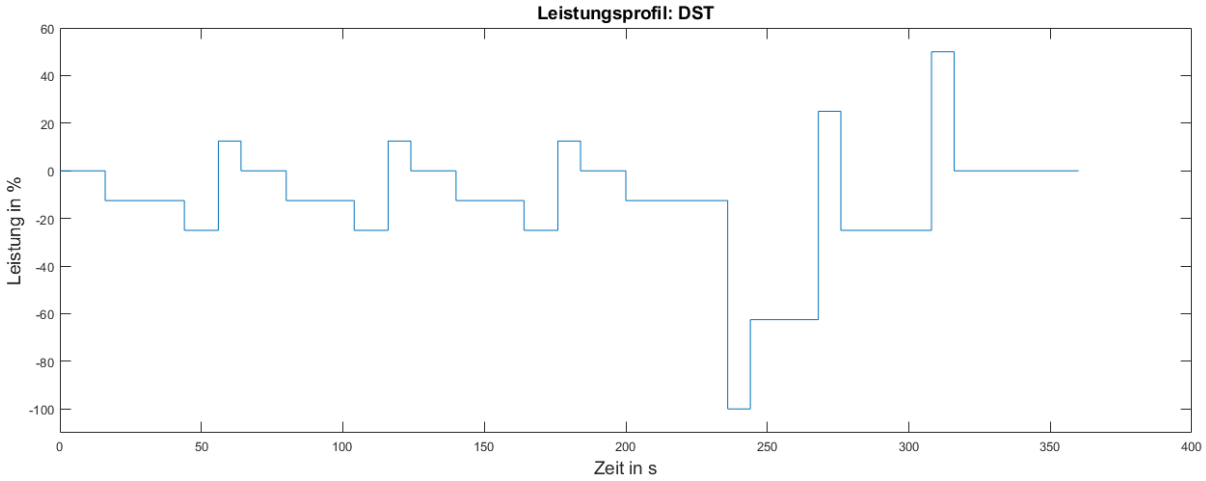


Abbildung 4.3: Leistungsprofil: DST

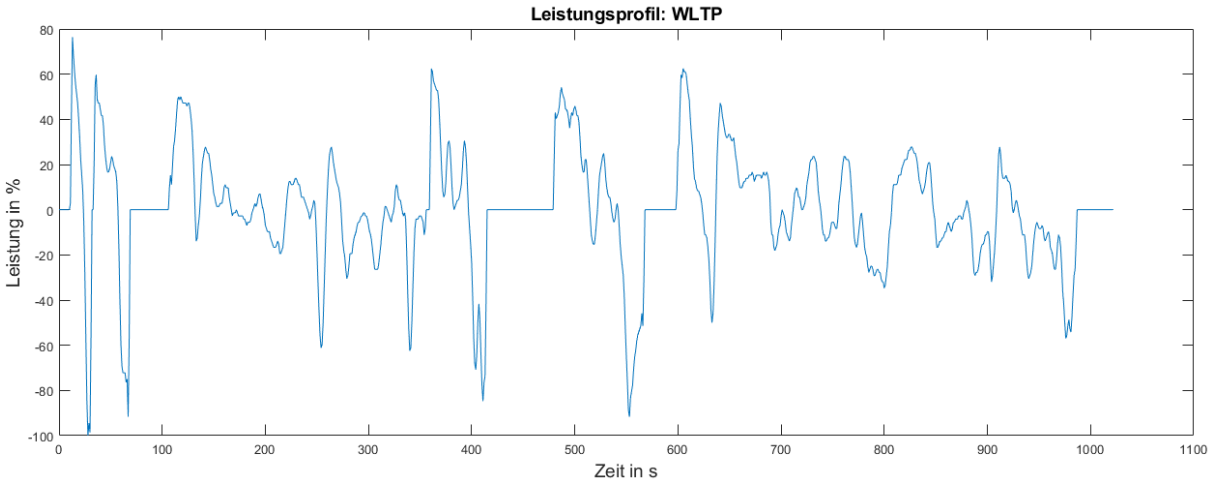


Abbildung 4.4: Leistungsprofil: WLTP

Wie zu erkennen ist, weisen die Fahrzyklen untereinander sehr unterschiedliche dynamische Verhalten auf, sodass die Generalisiungsfähigkeit ausreichend getestet werden kann.

# 5 Versuchsdurchführung und Auswertung

In diesem Kapitel wird des erarbeitete Konzept aus dem vorigen Kapitel praktisch durchgeführt. Die Trainings- und Testprofile werden erstellt, sodass das NARX-Netzwerk anhand dieser trainiert und getestet werden kann. Außerdem werden immer wieder Auswertungen vorgenommen, um zusätzliche erforderliche Anpassungen vorzunhemen.

## 5.1 Erstellung von Trainings- und Testprofil

Die folgende Abbildung 5.1 zeigt die erstellten Trainingsprofile für den Strom, die Spannung und den SoC:

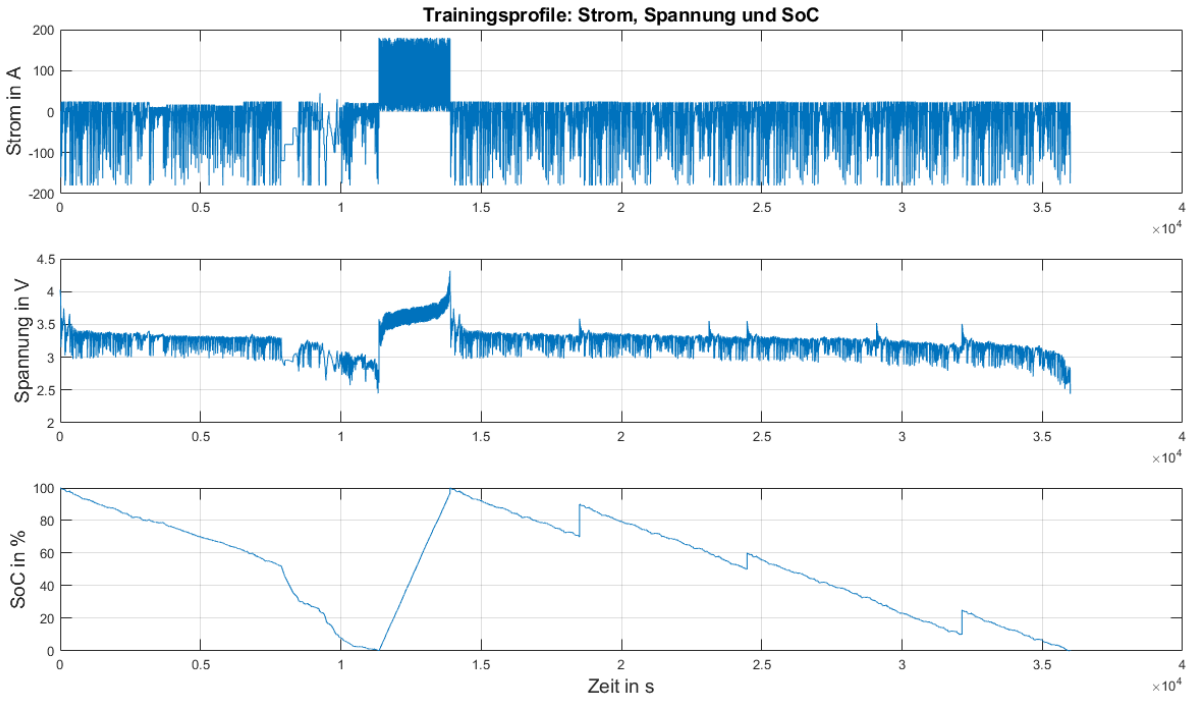


Abbildung 5.1: Trainingsprofile: Strom, Spannung und SoC

Das Stromprofil besteht aus vielen unterschiedlichen Fahrzyklen der „Powertrain Blockset“ -Erweiterung von Simulink. Der Strom wurde auf einen absoluten Lade- (Positiv) und -Entladestrom (Negativ) von 3C(180A) skaliert. Die Zeit wurde auf 4500s reduziert, damit die Trainingsdauer überschaubar bleibt. Die Spannung zeigt ebenfalls den charakteristischen Verlauf einer LiFeYP$O_4$ und bleibt im Betriebsbereich von 2.8V… 4V. Der SoC zeigt am Anfang eine komplette Entladung von 100% auf 0%. Darauffolgend steigt der SoC konstant auf 100%. Zum Ende hin sinkt der SoC wieder auf 0% und beinhaltet währenddessen drei Sprünge für die Beobachtung des Konvergenzverhalten. Die Testprofile mit dem DST und WLTP sind in der folgenden Abbildung 5.2 dargestellt:

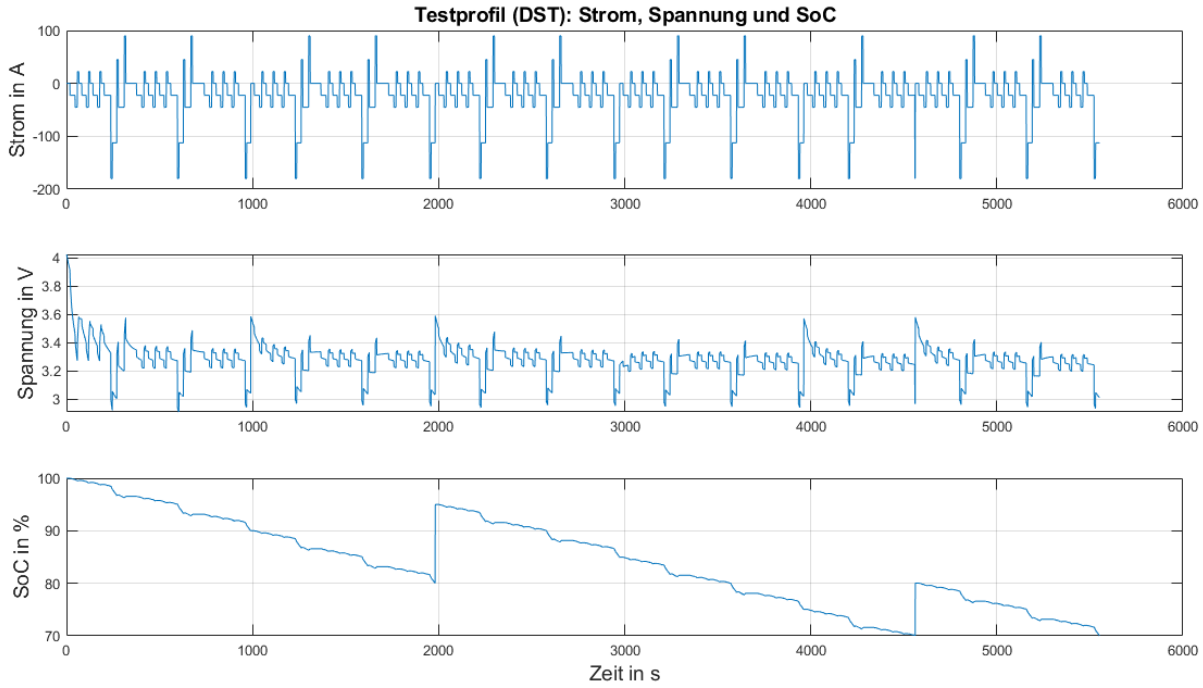


Abbildung 5.2: Testprofil (DST): Strom, Spannung und SoC

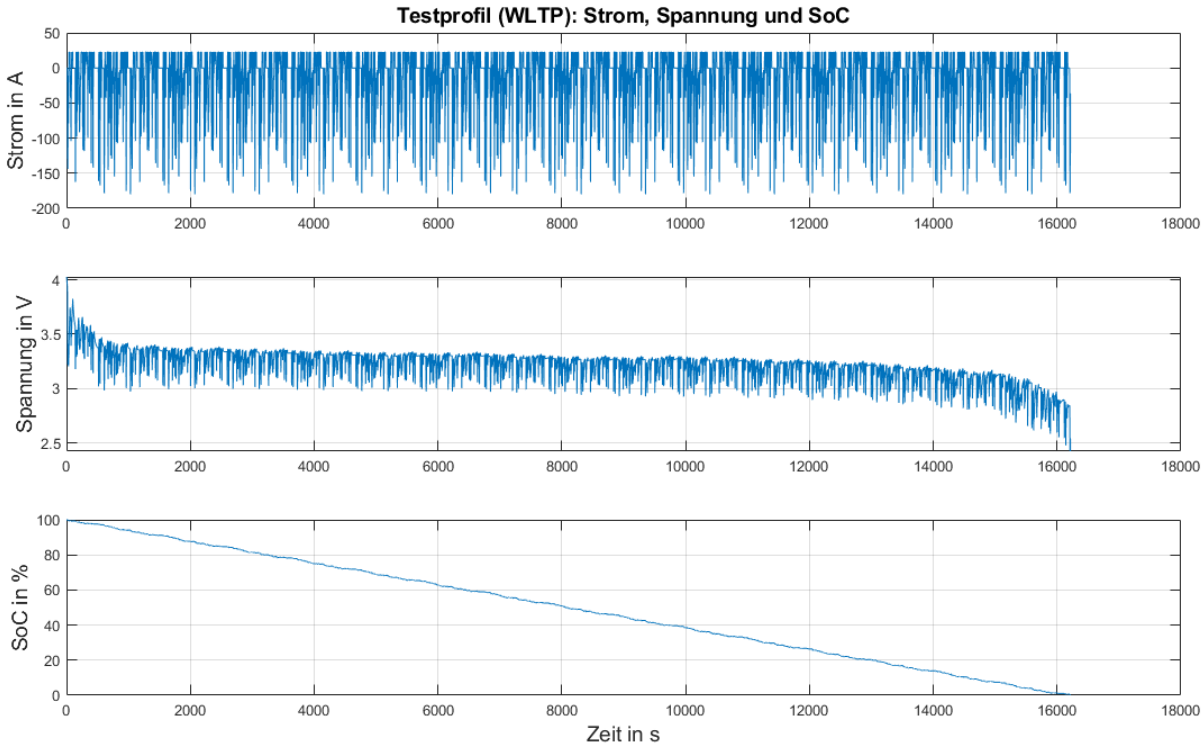


Abbildung 5.3: Testprofil (WLTP): Strom, Spannung und SoC

Die Profile in der ersten Abbildung 5.2 bestehen aus mehreren hintereinander geschalteten DST -Zyklen. Außerdem wurden zusätzlich Sprünge beim SoC eingebaut wie schon bei den Trainingsprofilen. In der zweiten Abbildung wird das Batteriemodell durch mehrere WLTP-Zyklen komplett entladen. Durch das unterschiedliche Verhalten sollte das NARX-Netzwerk ausreichend getestet sein.

## 5.2 Versuchsdurchführung mit Trainingsdaten

Zu Beginn werden die Trainingsversuche je Versuch ermittelt, wie im Unterkapitel 4.3 beschrieben wurde:

Tabelle 5.1: Benötigte Trainingsversuche pro Durchlauf

| Versuch | Trainingsversuche | Wiederholungen | MW(RMSE) |
|---|---|---|---|
| 1 | 10 | 5 | 8,2912 |
| 2 | 20 | 5 | 8,2420 |
| 3 | 30 | 5 | 7,7920 |
| 4 | 40 | 5 | 7,9581 |
| 5 | 50 | 5 | 7,9274 |
| 6 | 60 | 5 | 7,8921 |
| 7 | 70 | 5 | 7,8883 |

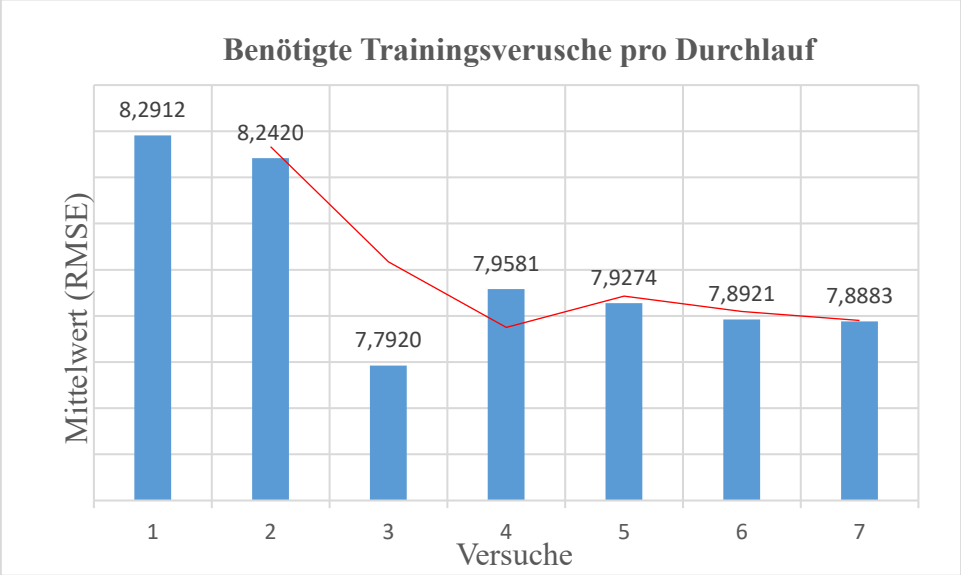


Abbildung 5.4: Benötigte Trainingsversuche pro Durchlauf

Wie zu erkennen, ergibt sich nahezu eine exponentielle Verbesserung des Fehlers (RMSE) bei zunehmenden Trainingsversuchen. Dadurch wird die Gefahr der fehlerhaften Optimierungsversuche durch die zufällig initialisierten Gewichte mit zunehmenden Trainingsversuchen sichtlich reduziert. Auf den ersten Blick fällt die Differenz der Fehler zwischen den Versuchen gering aus, dennoch können sich diese negativ auf das Training und die Generalisierungsfähigkeit auswirken (siehe Abbildung 2.18). Da der dritte Versuch ein unerwartetes Ergebnis zeigt, wurde zusätzlich noch eine Trendlinie (Gleitender Mittelwert) hinzugefügt. Natürlich muss man dabei bedenken, dass sich auch die Trainingszeit erhöht. Um einen Kompromiss zwischen Genauigkeit und Trainingszeit zu finden, wird die Anzahl der Trainingsversuche auf 30 begrenzt.

Im nächsten Schritt wird das NARX-Netzwerk trainiert. Die dabei verwendeten Hyperparameter sind noch einmal in der der nächsten Tabelle zusammengefasst:

Tabelle 5.2: Hyperparameter für das NARX-Netzwerk

| **Bezeichnung** | **Methode/Funktion/Wert** | |
|---|---|---|
| Aktivierungsfunktion | Hidden-Layer | Tangens-Hyperbolicus |
| | Output-Layer | Pureline |
| Trainingsalgorithmus | Levenberg-Marquardt | |
| Neuronen (N) | 0 …. 20 | |
| Inputdelay (ID) | 0 …. 5 | |
| Feedbackdelay (FD) | 1 …. 5 | |
| Hidden-Layer (H) | 1 | |

Das Training wurde mit Hilfe des vollfaktoriellen Versuchsplans durchgeführt, der in Tabelle 4.2 aufgeführt ist. Die Anzahl der Hidden-Layer wurde wegen des zu hohen Trainingsaufwands erstmal vernachlässigt. In der nächsten Tabelle ist ein Auszug von den wichtigsten Ergebnissen aufgelistet:

Tabelle 5.3: Ausschnitt aus Versuchsreihe

| **Versuch** | **ID** | **FD** | **N** | **H** | **RMSE** |
|---|---|---|---|---|---|
| 61 | 0 | 1 | 3 | 1 | 1,3681 |
| 121 | 0 | 1 | 5 | 1 | 0,9638 |
| 181 | 0 | 1 | 7 | 1 | 0,8902 |
| 187 | 0 | 2 | 7 | 1 | 0,9988 |
| 211 | 0 | 1 | 8 | 1 | 0,7564 |
| 217 | 0 | 2 | 8 | 1 | 0,9674 |
| 241 | 0 | 1 | 9 | 1 | 0,8553 |
| 247 | 0 | 2 | 9 | 1 | 0,8601 |
| 271 | 0 | 1 | 10 | 1 | 0,8045 |
| 277 | 0 | 2 | 10 | 1 | 0,8521 |

Daraus geht hervor, dass es charakteristische Kombinationen von ID und FD gibt, wo der RMSE sein Minimum aufweist, unabhängig von der Anzahl der Neuronen. Das hat den großen Vorteil, dass sich durch die Reduzierung der Versuche der Trainingsaufwand enorm verringert. Deshalb wird im nächsten Training auch die Anzahl der Hidden-Layer mit in den Versuchsplan aufgenommen. Folgende Tabelle 5.4 zeigt die Ergebnisse der Versuchsreihe mit Neuronen und Hidden-Layer:

Tabelle 5.4: Ergebnisse der Versuchsreihe mit RMSE

| H \ N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 7,7 | 6,9 | 6,6 | 3,4 | 4,7 | 6,1 | 5,8 | 5,0 | 3,0 | 3,3 | 4,7 | 4,2 | 3,6 | 4,6 | 3,0 | 3,4 | 4,3 | 5,7 | 3,7 | 2,6 |
| **2** | 7,3 | 7,4 | 4,3 | 5,2 | 4,1 | 4,8 | 3,4 | 3,3 | 3,1 | 2,7 | 4,7 | 4,2 | 3,4 | 4,0 | 2,9 | 2,8 | 3,4 | 3,7 | 2,3 | 5,4 |
| **3** | 7,3 | 6,5 | 6,5 | 7,1 | 6,1 | 4,2 | 5,4 | 2,1 | 3,4 | 2,2 | 2,7 | 2,9 | 3,2 | 3,2 | 3,2 | 3,7 | 2,6 | 3,7 | 3,9 | 2,9 |

Die Ergebnisse zeigen, dass sich das Netz an keine eindeutigen Muster hält. Sie weisen zwischen den Kombinationen starke Unterschiede auf. Die farblich hinterlegten Ergebnisse weisen den geringsten RMSE auf. Die Kombination mit H=3 und N = 8 ist mit einem RMSE von 2.1% das beste Ergebnis. In der nächsten Abbildung 5.5 ist das graphische Ergebnis zu sehen. Die Plots sind in folgender Reihenfolge von oben nach unten dargestellt: Bestes-, Schlechtes und mittleres Ergebnis. Diese Darstellung wird auch bei allen weiteren Abbildungen zum Training verwendet.

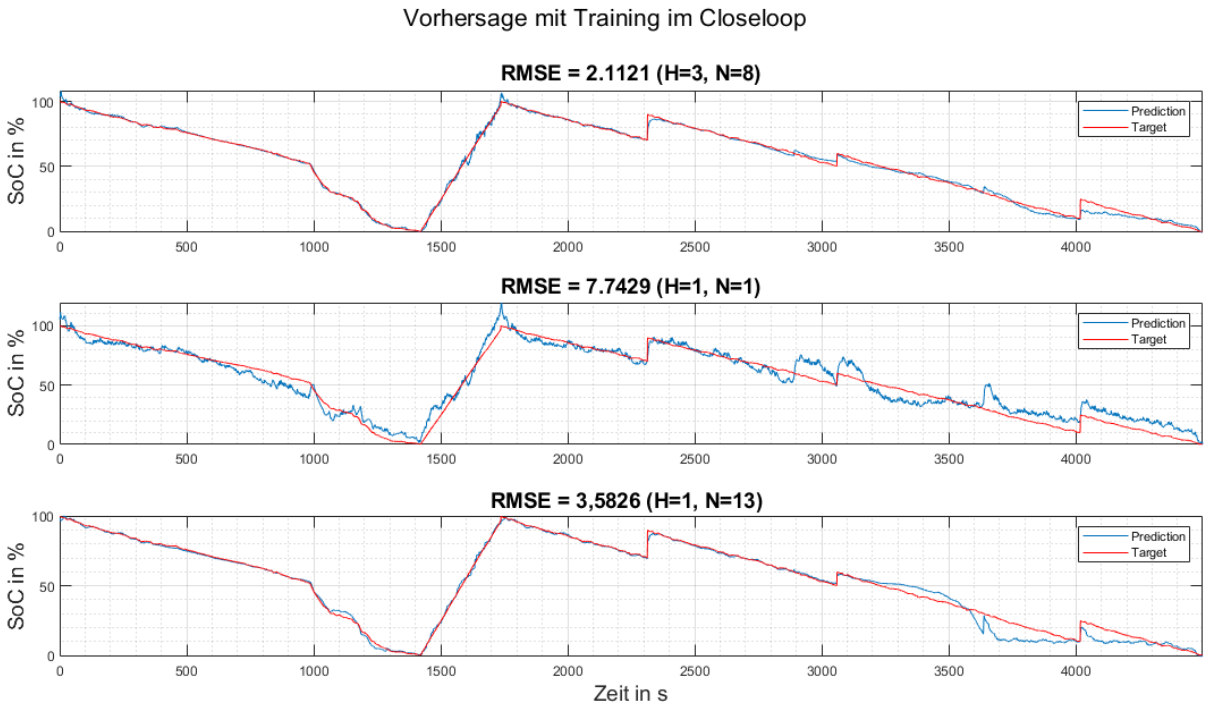


Abbildung 5.5: Vorhersage mit Training im Closeloop (RMSE)

In den Bereichen der starken Änderungen des SoC sind größere Abweichungen der Vorhersage zum Sollwert erkennbar. Besonders bei den eingebauten Sprüngen zeigt das Netz ein schlechtes Konvergenzverhalten. Außerdem nimmt die Genauigkeit bei allen Vorhersagen zum Ende hin ab. Das schlechteste Ergebnis zeigt trotz niedrigem RMSE eine sehr stark verrauschte Vorhersage, was für eine genaue Schätzung des SoC stets vermieden werden sollte und auch folgenschwere Auswirkungen auf die Generalisierungsfähigkeit bei unbekannten Daten hat.

Deshalb wird zum Vergleich noch eine Versuchsreihe mit dem Bestimmtheitsmaß durchgeführt, um die Robustheit der Bewertungsmaßstäbe zu vergleichen. Die Ergebnisse sind in der nächsten Tabelle 5.5 dargestellt:

Tabelle 5.5: Ergebnisse der Versuchsreihe mit Bestimmtheitsmaß

| N / H | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0,939 | 0,947 | 0,952 | 0,991 | 0,972 | 0,974 | 0,983 | 0,974 | 0,977 | 0,980 | 0,963 | 0,987 | 0,995 | 0,968 | 0,968 | 0,987 | 0,983 | 0,991 | 0,982 | 0,989 |
| **2** | 0,939 | 0,955 | 0,984 | 0,982 | 0,968 | 0,987 | 0,969 | 0,984 | 0,987 | 0,986 | 0,988 | 0,982 | 0,988 | 0,991 | 0,988 | 0,986 | 0,997 | 0,988 | 0,992 | 0,987 |
| **3** | 0,939 | 0,880 | 0,952 | 0,990 | 0,988 | 0,989 | 0,976 | 0,984 | 0,977 | 0,992 | 0,976 | 0,993 | 0,987 | 0,991 | 0,995 | 0,993 | 0,986 | 0,996 | 0,995 | 0,992 |

Alle Ergebnisse ab einem Bestimmtheitsmaß von 0,990, was eine Übereinstimmung der Vorhersage mit dem Sollwert von 99% aussagt, sind farblich markiert. Die Ergebnisse zeigen, dass die Genauigkeit (hohes Bestimmtheitsmaß) mit zunehmender Komplexität (Neuronen, Hidden-Layer) des Netzes genauer wird. In der Abbildung 5.6 ist die dazugehörige graphische Darstellung zu sehen:

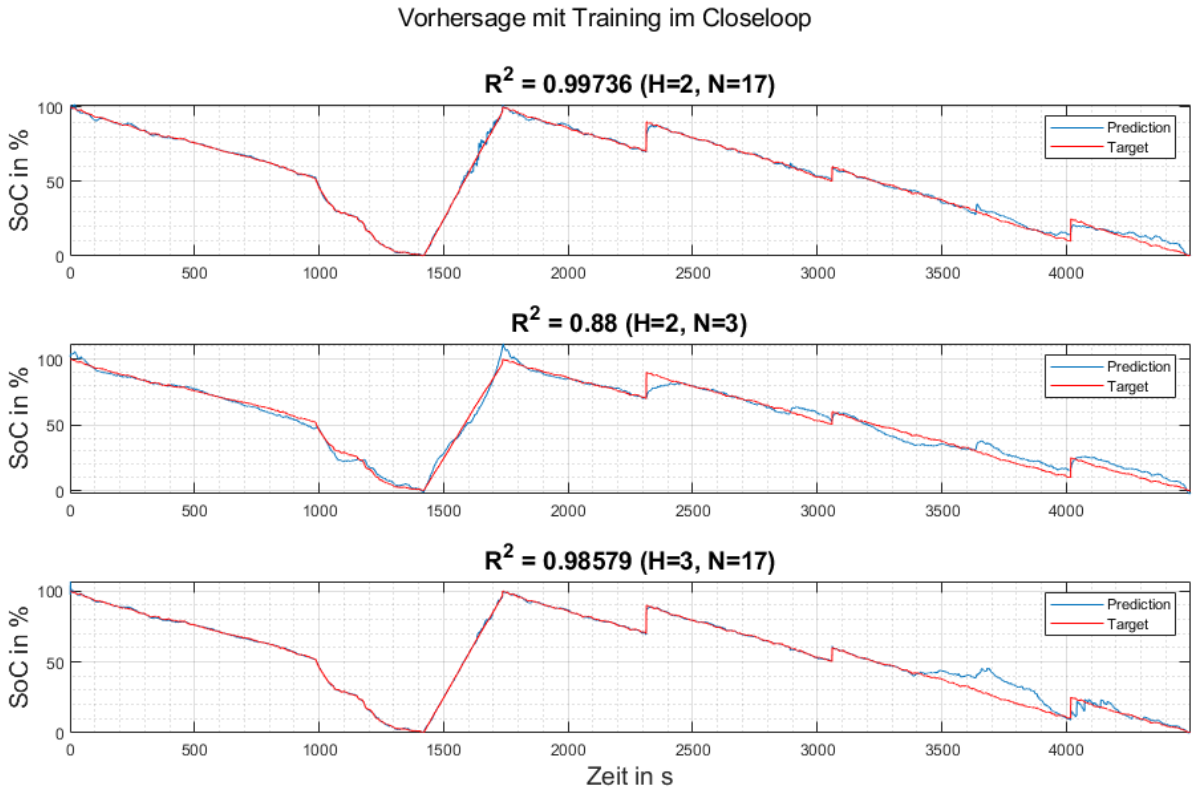


Abbildung 5.6:Vorhersage mit Training im Closeloop (R2)

Die Vorhersage zeigt in etwa das gleiche Verhalten wie in der Versuchsreihe mit dem RMSE. Die Verläufe sind aber geringer verrauscht und schwanken weniger um den Sollwert. Daraus kann man schlussfolgern, dass bei den Trainingsversuchen pro Versuch auch wirklich die beste Vorhersage, die den Sollwert beschreiben soll, ausgewählt wird. Deshalb scheint das

Bestimmtheitsmaß für die Bewertung die bessere Methode zu sein. Eine eindeutige Entscheidung lässt sich aber erst mit den Testdaten sicherstellen. Außerdem wird noch eine Trainingsreihe ohne die eingebauten Sprünge durchgeführt. Die Sprünge wurden bei der Erstellung der Testdaten auch nur künstlich erzeugt und spiegeln den eigentlichen Verlauf der Fahrzyklen nicht wieder. Zudem wurden noch mehr Zeitschritten in die Daten aufgenommen, um ein mögliches Overfitting zu verhindern.

Tabelle 5.6: Trainingsergebnisse ohne Sprünge und mehr Zeitschritte

| N / H | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,996 | 0,999 | 0,998 | 0,999 | 0,992 | 0,995 | 0,999 | 0,993 | 0,993 | 0,980 | 0,994 | 0,991 | 0,998 | 0,992 | 0,979 | 0,991 | 0,982 | 0,957 | 0,992 | 0,921 |
| 2 | 0,986 | 0,996 | 0,993 | 0,995 | 0,976 | 0,993 | 0,986 | 0,983 | 0,950 | 0,910 | 0,954 | 0,981 | 0,929 | 0,841 | 0,923 | 0,901 | 0,837 | 0,769 | 0,773 | 0,842 |
| 3 | 0,982 | 0,996 | 0,989 | 0,974 | 0,993 | 0,923 | 0,934 | 0,948 | 0,882 | 0,833 | 0,925 | 0,853 | 0,855 | 0,789 | 0,889 | 0,827 | 0,885 | 0,771 | 0,919 | 0,781 |

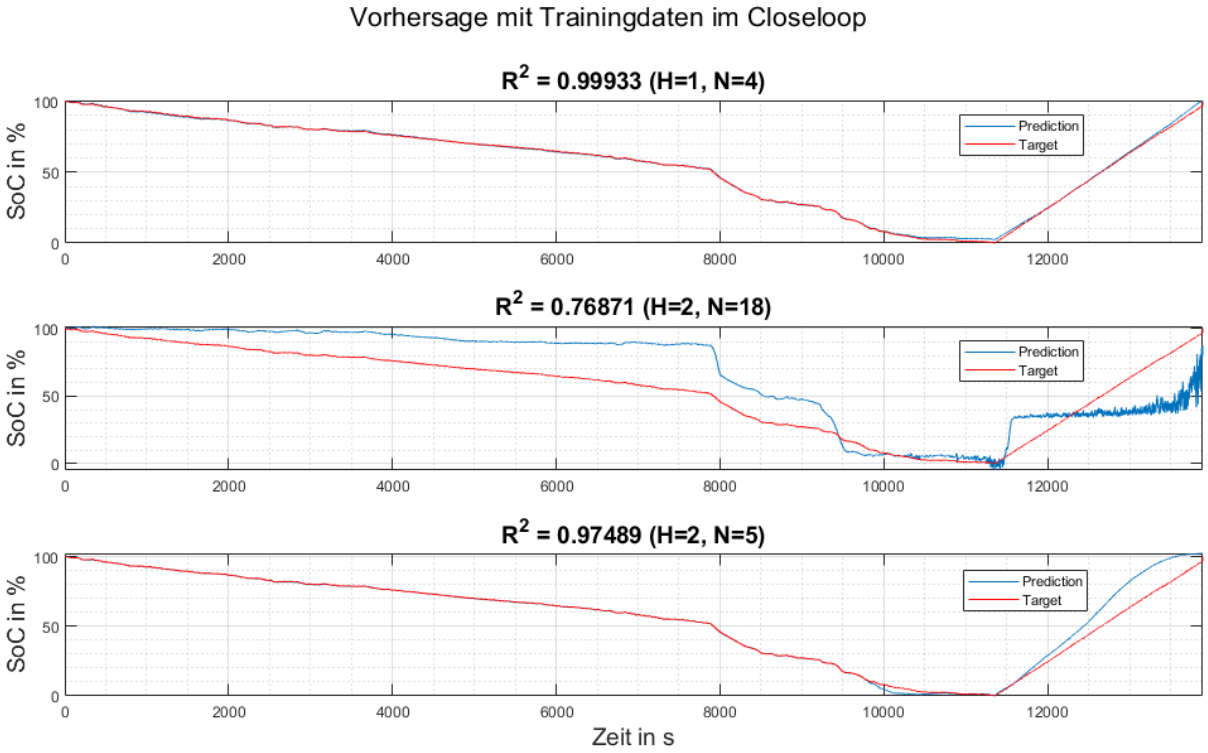


Abbildung 5.7: Trainingsergebnisse ohne Sprünge und mehr Zeitschritte

Ohne die eingebauten Sprünge verhält sich das Netz bei den Trainingsversuchen wesentlich stabiler und es können anhand der Kombinationen von Neuronen und Layern Verhaltensmuster identifiziert werden. Dazu im nächsten Hauptkapitel mehr.

## 5.3 Erprobung mit unbekannten Testdaten

In diese Unterkapitel werden nun die zuvor trainierten NARX-Netzwerke mit unbekannten Daten auf die Generalisierungsfähigkeit getestet. Das ist sehr wichtig, da dadurch erst das wahre Potenzial für die praktische Anwendbarkeit ersichtlich wird. In der folgenden Tabelle/Abbildung sind die Ergebnisse der ersten Testreihe festgehalten. Die Zeitreihe wurden im ersten Versuch auf 460s reduziert.

Tabelle 5.7: Ergebnisse mit Testdaten (460s)

| N / H | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,494 | 0,523 | 0,457 | 0,801 | 0,759 | 0,818 | 0,782 | 0,849 | 0,801 | 0,779 | 0,806 | 0,799 | 0,837 | 0,761 | 0,731 | 0,781 | 0,772 | 0,782 | 0,684 | 0,597 |
| 2 | 0,495 | 0,800 | 0,814 | 0,814 | 0,772 | 0,847 | 0,852 | 0,771 | 0,787 | 0,774 | 0,850 | 0,641 | 0,650 | 0,858 | 0,566 | 0,869 | 0,673 | 0,510 | 0,679 | 0,716 |
| 3 | 0,495 | 0,798 | 0,690 | 0,824 | 0,720 | 0,831 | 0,871 | 0,749 | 0,833 | 0,825 | 0,546 | 0,009 | 0,804 | 0,667 | 0,545 | 0,797 | 0,702 | 0,863 | 0,649 | 0,619 |

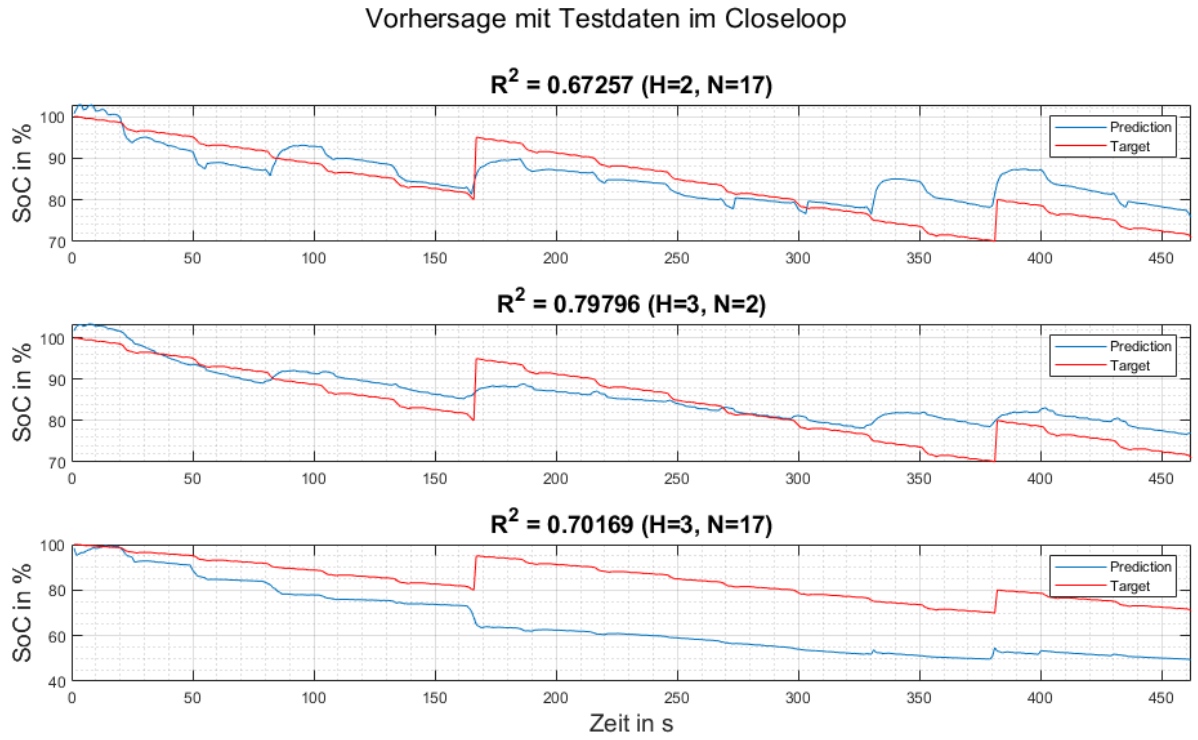


Abbildung 5.8: Ergebnisse mit Testdaten (460s)

Es ist sofort erkennbar, dass die Generalisierungsfähigkeit auf die unbekannten Testdaten sehr schlecht ausfällt. Das Beste Ergebnis aus dem Training mit R = 0.997 ist jetzt sogar schlechter als das schlechteste Trainingsergebnis mit R = 0.88. Andererseits brachte das schlechteste Trainingsergebnis die beste Generalisierungsfähigkeit auf die Testdaten. Zudem streuen die Ergebnisse sehr stark, was auf kein eindeutiges Verhaltensmuster schließen lässt. Trotzdem besteht die Vermutung, dass das Netz ins Overfitting gerät, da sich die Vorhersagefehler stark verschlechtert haben. Deshalb wurde schon zuvor beim Training die Anzahl der Trainingsdaten

erhöht, um genau diesen Fall auszuschließen. Zusätzlich werden die zuvor eingeführten Sprünge entfernt, da sich das Netz in diesen Bereichen schon beim Training sehr stark von den Sollwerten wegbewegte.

Tabelle 5.8: Vorhersage mit Testdaten ohne Sprünge und mehr Zeitschritte

| H \ N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,991 | 0,999 | 0,998 | 0,992 | 0,999 | 1,000 | 1,000 | 0,993 | 0,991 | 0,992 | 0,999 | 0,997 | 0,996 | 0,994 | 0,996 | 0,993 | 0,992 | 0,994 | 0,990 | 0,995 |
| 2 | 0,971 | 0,994 | 0,996 | 0,999 | 0,995 | 0,986 | 0,991 | 0,986 | 0,985 | 0,979 | 0,993 | 1,000 | 0,984 | 0,991 | 0,956 | 0,995 | 0,185 | 0,186 | 0,170 | 0,993 |
| 3 | 0,960 | 0,993 | 0,999 | 0,994 | 0,922 | 0,995 | 0,996 | 0,998 | 0,988 | 0,991 | 0,989 | 0,177 | 0,493 | 0,613 | 0,634 | 0,219 | 0,384 | 0,108 | 0,810 | 0,282 |

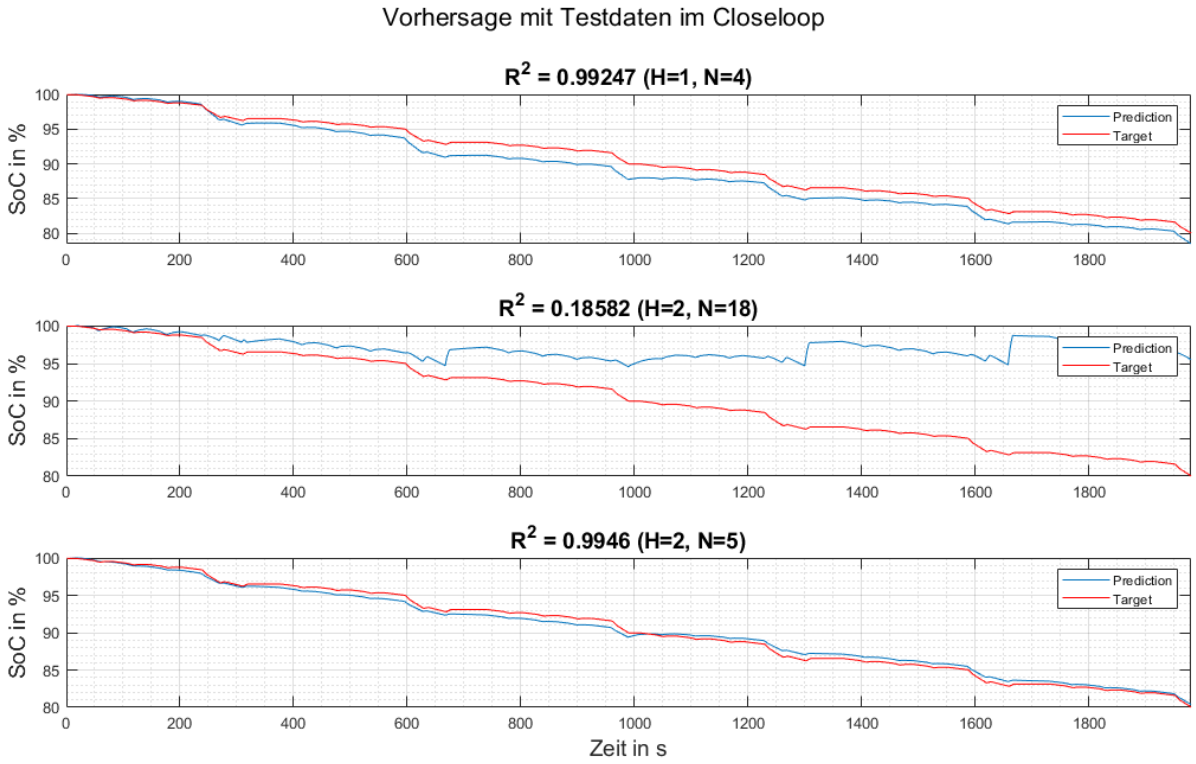


Abbildung 5.9: Vorhersage mit Testdaten ohne Sprünge und mehr Zeitschritten (DST)

Die Ergebnisse zeigen eine sehr gute Vorhersage der unbekannten Daten. Sowohl die besten und mittlere Kombinationen der Parameter aus dem Training (siehe Abbildung 5.7) erzeugen gute Ergebnisse. Hingegen die schlechten Kombinationen eine schlechte Generalisierungsfähigkeit aufweisen. Das mittele Ergebnis zeigt ein besseres Ergebnis als das Beste, was auf ein Overfitting hindeutet. Im nächsten Schritt wird nun das Netz mit den WLTP-Testdaten getestet. Die Ergebnisse zeigen ein ähnliches verhalten wie mit den Testdaten vom DST (siehe Abbildung 5.7).

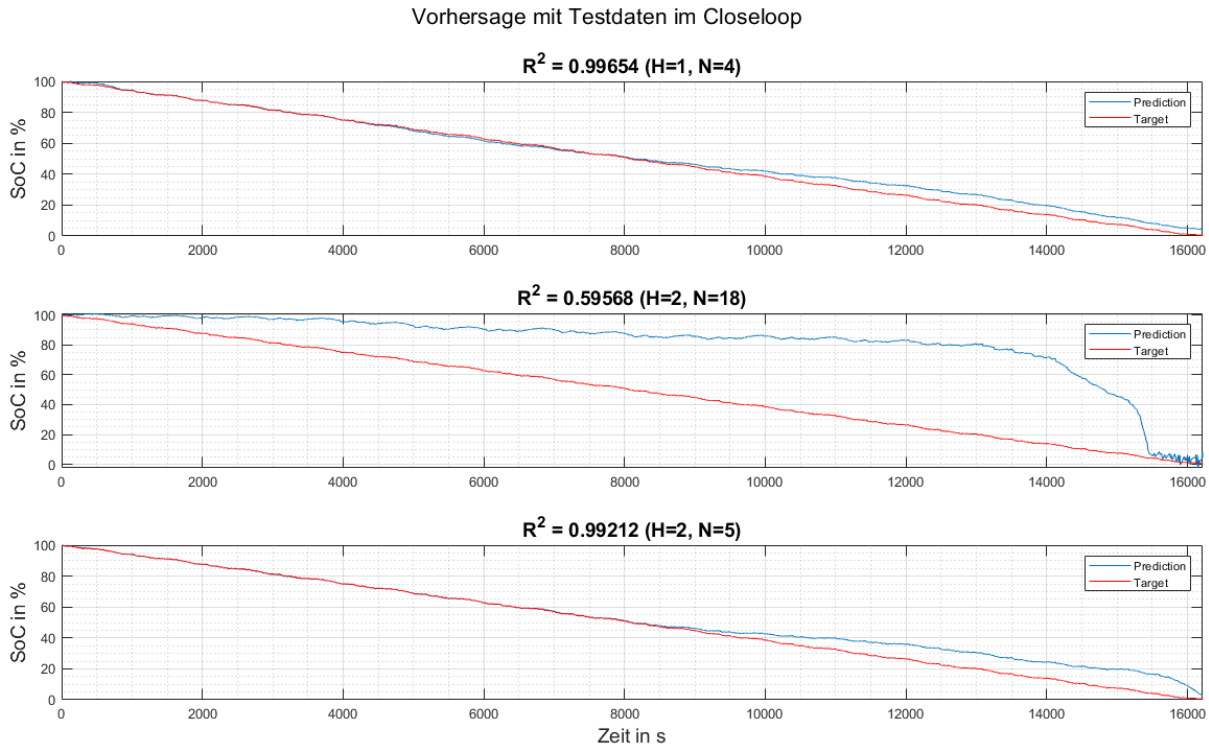


Abbildung 5.10: WLTP-Testdaten

Die Ergebnisse zeigen ein ähnliches Verhalten wie mit den Testdaten vom DST (siehe Abbildung 5.7). Lediglich zum Ende hin scheint das Netz an Genauigkeit zu verlieren.

# 6 Analyse der Ergebnisse

Durch die eingebauten Sprünge in den Daten wurde das NARX-Netzwerk instabil und führte dazu, dass die Ergebnisse bei Training und Test verfälscht bzw. schlechter wurden. Nachdem die Sprünge entfernt waren, stabilisierten sich die Ergebnisse. Zwischen den Kombinationen der Neuronen und Hidden-Layer wurden Abhängigkeiten zur Netzqualität klarer. Aus den Trainingsergebnissen aus Abbildung 5.9 kann folgende Schlussfolgerung erzielt werden:

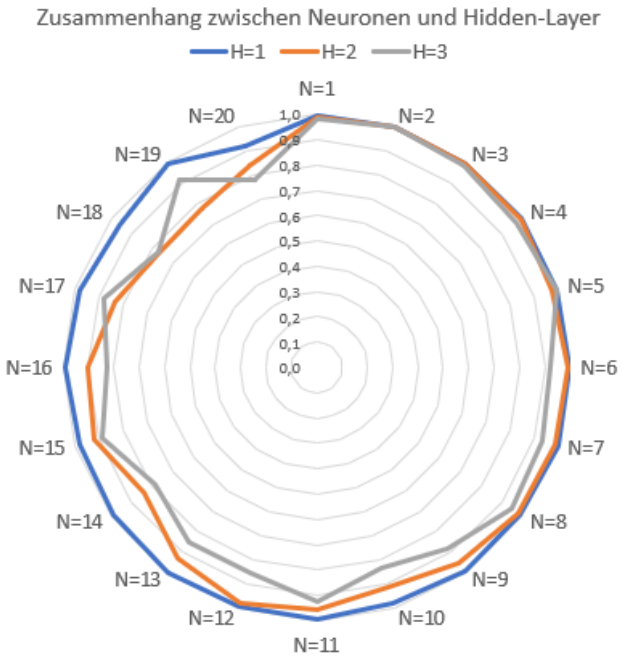


Abbildung 6.1: Zusammenhang zwischen Neuronen und Hidden-Layer (Training)

Aus der Abbildung 6.1 geht hervor, dass bei einem Hidden-Layer mit steigender Anzahl der Neuronen, der Fehler über die gesamten Trainingsversuche auf einem konstanten Niveau bleibt. Gleichzeitig nimmt die Genauigkeit mit zwei Hidden-Layern und acht Neuronen ab. Bei drei Hidden-Layern verschlechterte sich das Netz schon ab fünf Neuronen. Im Bereich zwischen 1-5 Neuronen konnten durch das Netz sehr gute Ergebnisse unabhängig der Anzahl von Neuronen und Layern erzielt werden. Infolgedessen ist eine Erhöhung der Hidden-Layer in den Bereichen mit wenig Neuronen beim Training sinnvoll, um ein optimales Ergebnis zu erzielen. Natürlich muss man auch noch die Trainingsdauer berücksichtigen, die mit steigender Anzahl von Hidden-Layern steigt. Außerdem spielt die Anzahl von Daten eine wichtige Rolle, denn umso höher die Genauigkeit im Training ist, desto höher ist die Wahrscheinlichkeit des Overfitting. Die Folge wäre eine schlechte Generalisierungsfähigkeit auf unbekannte Daten. Da aber in diesem Fall die Genauigkeit mit Erhöhung der Hyperparameter abnimmt, kann man ein Overfitting eher ausschließen.

In der nächsten Abbildung 6.2 wird nun auf den Zusammenhang der Neuronen und Hidden-Layer bei den Testergebnissen eingegangen.

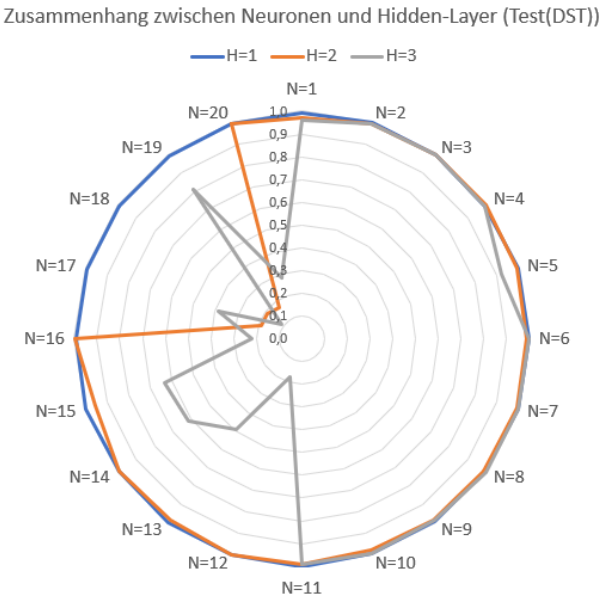


Abbildung 6.2: Zusammenhang zwischen Neuronen und Hidden-Layer (Test (DST))

Die Zusammenhänge zwischen Neuronen und Hidden-Layer zeigen nahezu das gleiche Muster wie aus den Trainingsversuchen (siehe Abbildung 6.1). Das Netz mit einem Hidden-Layer hat über den gesamten Bereich der Neuronen eine sehr gute Vorhersagegenauigkeit. Bei zwei Hidden-Layer wird die Genauigkeit ab 14 Neuronen schlecht. Bei drei Hidden-Layer sogar schon bei 11 Neuronen. Das zeigt, dass das Netz beim Training durch die höhere Komplexität die Daten auswendig lernt und eine schlechte Generalisierungsfähigkeit auf die unbekannten Daten hat (Overfitting). Schlussfolgernd kann man sagen, dass das Ergebnis sehr gut ist, da die ermittelte Kombination aus dem Training (H=1, N=4) eine sehr genau Vorhersage bei den Testdaten liefert und sogar noch mehr Potenzial für weitere Vorhersageschritte besteht. Wären die Ergebnisse wesentlich schlechter im Bereich dieser Kombination, würde das auf ein Overfitting hindeuten und mehr Daten wären erforderlich. Die folgenden Abbildung 6.3 zeigt den Zusammenhang der Neuronen und Hidden-Layern mit den WLTP-Zyklen, welche deutlich mehr Zeitschritte besitzt (DST: 1981s, WLTP: 16216s).

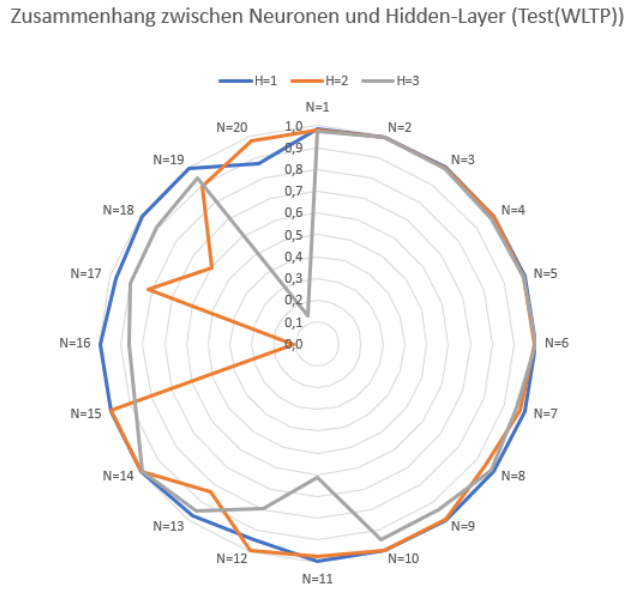


Abbildung 6.3: Zusammenhang zwischen Neuronen und Hidden-Layer (Test (WLTP)):

Die aufgestellte Vermutung bei den DST-Testdaten mit wenige Zeitschritten wird hiermit bestätigt. Durch die Überanpassung beim Training mit erhöhter Anzahl von Neuronen und Hidden-Layern wird die Vorhersagegenauigkeit bei mehr Zeitschritten instabiler. Trotzdem ist das Ergebnis sehr gut, da die ermittelte Kombination beim Training auch hier noch ein genaues Ergebnis erzielt. Möchte man in Zukunft noch mehr Zeitschritte Vorhersagen, so wäre eine Erhöhung der Trainingsdaten sinnvoll. Zusammenfassend kann man sagen, dass das NARX-Netzwerk eine sehr genaue Generalisierungsfähigkeit bei einer geringen Anzahl von Neuronen und Hidden-Layer liefert.

Die Besten Ergebnisse wurden mit folgenden Hyperparametern durch das Training erzielt:

Tabelle 6.1: Beste Ergebnisse mit Testdaten

| **Bezeichnung** | **Methode/Funktion/Wert** | |
|---|---|---|
| Aktivierungsfunktion | Hidden-Layer | Tangens-Hyperbolicus |
| | Output-Layer | Pureline |
| Trainingsalgorithmus | Levenberg-Marquardt | |
| Inputdelay (ID) | 0 | |
| Feedbackdelay (ID) | 1 | |
| Neuronen (N) | 4 | |
| Hidden-Layer (H) | 1 | |

# 7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde eine Parametereinflussanalyse für ein NARX-Netzwerk am Beispiel einer Ladezustandsschätzung für eine Lithium-Ionen-Batterie durchgeführt. Um diesen Vorhaben zu lösen, wurden zu Beginn die Grundlagen der Batterietechnik und künstlichen neuronalen Netze zusammengefasst. Durch die Grundlagen konnte dann mit Hilfe einer Methodik ein Konzept erarbeitet werden. Die Einflüsse der Hyperparamter wurden mit einem vollfaktoriellen Versuchsplan experimentell ermittelt. Zu den Hyperparametern gehörten: ID, FD, N und H. Damit die Parameter nicht jedes Mal vor Beginn des Trainings manuell eingestellt werden mussten, wurde der Prozess in Matlab automatisiert.

Wie schon erwähnt stand das Training und die Einflüsse der Hyperparameter im Fokus. Besonders beim Training des NARX-Netzwerks kamen einige Probleme auf, die seitens der MathWorks-Dokumentation nicht genau beschrieben wurden. Laut Mathwork war nur das Training im Openloop und nicht das zusätzliche Training im Closeloop beschrieben. Durch das eigenständige Implementieren des Trainings im Closeloop wurde eine viel bessere Generalisierungsfähigkeit bei den Testdaten erzielt. Außerdem ist eine zufällige Datenteilung der Trainingsdaten standartmäßig vorgesehen. Das führt dazu, dass das charakteristische Zeitverhalten der Daten zerstört wird und eine genau Ermittlung von ID und FD nicht möglich ist. Deshalb muss die Funktion (divideFcn) zur Aufteilung der Daten von „dividerand“ in „divideblock“ geändert werden.

Durch die Versuche wurde herausgefunden, dass das ID und FD, unabhängig von der Anzahl der Neuronen und Hidden-Layern, ermittelt werden kann. Außerdem wurde festgestellt, dass bis zu einer maximalen Anzahl von fünf Neuronen und zwei Hidden-Layern das NARX-Netzwerk eine ausreichende Generalisierungsfähigkeit auf die Testdaten hat. Bei mehr Vorhersageschritten ist eine Erhöhung der Trainingsdaten erforderlich.

Auf Grundlage dieser Arbeit kann eine sehr genau Schätzung des SoC mit Hilfe des NARX-Netzwerks ausgelegt werden. In zukünftigen Untersuchungen könnten noch die Auswirkung von unterschiedlichen Aktivierungsfunktionen und Trainingsalgorithmen untersucht werden. Zusätzlich wäre die Verwendung von Messdaten aus realen Situationen für eine leistungsfähige SoC-Schätzung sinnvoll. Dadurch sollte die Genauigkeit noch weiter verbessert werden.

# Abkürzungsverzeichnis

CC.....constant current
CV .....*constant voltage*
DoD .....*Depth of Discharge*
DST ..... *Dynamic Stress Test*
FD.....*Feedbackdelay*
FFN.....*Feed Forward Netzwerk*
H .....*Hidden-Layer*
ID.....*Inputdelay*
KNN ..... *Künstliche Neuronale Netze*
Li-Ion..... *Lithium-Ionen*
MLP..... *Multilayer Perceptron*
N .....*Neuron im Hidden-Layer*
NARX.....*Nonlinear autoregressive neural network with external input*
OCV ..... *Open-Circuit Voltage*
R^2 .....*Bestimmtheitsmaß*
ReLU .....*Rectified Linear Units*
RMSE.....*Root Mean Square Error*
RNN ..... *Rekurrentes neuronales Netzwerk*
SoC .....*State of Charge*
Tanh..... *Tangens Hyperbolicus*
TDL .....*Tapped delay Lines*
WLTP ..... *Worldwide Harmonized Light-Duty Vehicles Test Procedure*

# Tabellenverzeichnis

Tabelle 4.1: Determining the Number of Hidden Layers[He05] ..... 31
Tabelle 4.2: Faktoren für den vollfaktoriellen Versuchsplan ..... 32
Tabelle 4.3: Technische Spezifikationen der Batterie WD-LYP60AHA[OA17] ..... 34
Tabelle 5.1: Benötigte Trainingsversuche pro Durchlauf ..... 38
Tabelle 5.2: Hyperparameter für das NARX-Netzwerk ..... 39
Tabelle 5.3: Ausschnitt aus Versuchsreihe ..... 39
Tabelle 5.4: Ergebnisse der Versuchsreihe mit RMSE ..... 40
Tabelle 5.5: Ergebnisse der Versuchsreihe mit Bestimmtheitsmaß ..... 41
Tabelle 5.6: Trainingsergebnisse ohne Sprünge und mehr Zeitschritte ..... 42
Tabelle 5.7: Ergebnisse mit Testdaten (460s) ..... 43
Tabelle 5.8: Vorhersage mit Testdaten ohne Sprünge und mehr Zeitschritte ..... 44
Tabelle 6.1: Beste Ergebnisse mit Testdaten ..... 48

# Abbildungsverzeichnis

Abbildung 2.1: Weltweite Marktaufteilung nach Batteriesystemen auf Kostenbasis in Anlehnung an [JW19] ..... 2
Abbildung 2.2:Auszug aus elektrochemischer Reihe in Anlehnung an [Sch14] ..... 3
Abbildung 2.3: schematischer Aufbau eine Batterie beim Entladevorgang in Anlehnung an[JW19] ..... 4
Abbildung 2.4: Spannungsverlauf einer Li-Ion-Zelle bei sprungartiger Entladung [Die13] ..... 6
Abbildung 2.5: Vereinfachtes Ersatzschaltbild einer Batterie ..... 7
Abbildung 2.6: Beispiel für ein Ladeverfahren mit drei Ladephasen [JW19] ..... 9
Abbildung 2.7: Potential und spezifische Kapazität verschiedener positiver und negativer aktiven Materialien für Lithium-Ionen-Zellen ..... 10
Abbildung 2.8: Verlauf der Klemmspannung bei einem Ladevorgang mit 20A[Die13] ..... 12
Abbildung 2.9: Kalman-Filter-Kreislauf ..... 13
Abbildung 2.10: Prototypischer Aufbau biologischer Neuronen[Kr15] ..... 14
Abbildung 2.11: Künstliches Neuron (Perzeptron) ..... 15
Abbildung 2.12: Aktivierungsfunktionen [Ba21] ..... 16
Abbildung 2.13: Logische Operationen für das Perzeptron [Ba21] ..... 18
Abbildung 2.14: XOR-Operation ..... 19
Abbildung 2.15:Beispiel eines Feed-Forward-Netzwerk ..... 19
Abbildung 2.16: Beispiel eines Rekurrenten neuralen Netzwerks in Anlehnung an [Ba21] ..... 20
Abbildung 2.17: Vereinfachte Darstellung eines NARX-Netzwerks ..... 21
Abbildung 2.18: Probleme bei Backpropagation[CP21] ..... 24
Abbildung 3.1: Entwicklung eines künstlichen neuronalen Netzwerks[LÄ20] ..... 27
Abbildung 4.1: Links: Openloop; Rechts: Closeloop ..... 29
Abbildung 4.2: Der automatisierte Trainingsablauf eines NARX-Netzwerks ..... 33
Abbildung 4.3: Leistungsprofil: DST ..... 35
Abbildung 4.4: Leistungsprofil: WLTP ..... 35
Abbildung 5.1: Trainingsprofile: Strom, Spannung und SoC ..... 36
Abbildung 5.2: Testprofil (DST): Strom, Spannung und SoC ..... 37
Abbildung 5.3: Testprofil (WLTP): Strom, Spannung und SoC ..... 37
Abbildung 5.4: Benötigte Trainingsversuche pro Durchlauf ..... 38
Abbildung 5.5: Vorhersage mit Training im Closeloop (RMSE) ..... 40
Abbildung 5.6:Vorhersage mit Training im Closeloop (R2) ..... 41
Abbildung 5.7: Trainingsergebnisse ohne Sprünge und mehr Zeitschritte ..... 42

Abbildung 5.8: Ergebnisse mit Testdaten (460s) ..... 43
Abbildung 5.9: Vorhersage mit Testdaten ohne Sprünge und mehr Zeitschritten (DST)..... 44
Abbildung 5.10: WLTP-Testdaten ..... 45
Abbildung 6.1: Zusammenhang zwischen Neuronen und Hidden-Layer (Training)..... 46
Abbildung 6.2: Zusammenhang zwischen Neuronen und Hidden-Layer (Test (DST)) ..... 47
Abbildung 6.3: Zusammenhang zwischen Neuronen und Hidden-Layer (Test (WLTP)):..... 48

# Literaturverzeichnis


[Ah21] Ahmed, M. F. B. et al.: Awareness to Deepfake: A resistance mechanism to Deepfake: 2021 International Congress of Advanced, 2021, S. 1–5.

[Ba21] Basler, D.: Neuronale Netze mit C# programmieren. Mit praktischen Beispielen für Machine Learning im Unternehmenseinsatz. Hanser, München, 2021.

[Ch13] Chang, W.-Y.: The State of Charge Estimating Methods for Battery: A Review. ISRN Applied Mathematics, S. 1–7, 2013.

[CP21] Neuronale Netze - Eine Einführung - Chemgapedia. http://www.chemgapedia.de/vsengine/tra/vsc/de/ch/13/trajektorien/nn_ein.tra/Vlu/vsc/de/ch/13/vlu/daten/neuronalenetze/backpropagation.vlu/Page/vsc/de/ch/13/anc/daten/neuronalenetze/snn8_5.vscml.html, Stand: 27.08.2021.

[Die13] Diehl, W.: Entwicklung eines Batteriemanagementsystems für Lithium-Batterien, Wolfenbüttel, 2013.

[HDB02] Hagan, M. T.; Demuth, H. B.; Beale, M. H.: Neural network design. Campus Publishing Service, Boulder, Colorado, 2002.

[He05] Heaton, J.: Introduction to neural networks with Java. Heaton Research, St. Louis, 2005.

[HM94] Hagan, M. T.; Menhaj, M. B.: Training feedforward networks with the Marquardt algorithm: IEEE Transactions on Neural Networks Nov. 1994, S. 989–993.

[JW19] Jossen, A.; Weydanz, W.: Moderne Akkumulatoren richtig einsetzen. Cuvillier Verlag, Göttingen, 2019.

[KD18] Kurzweil, P.; Dietlmeier, O.: Elektrochemische Speicher. Superkondensatoren, Batterien, Elektrolyse-Wasserstoff, rechtliche Rahmenbedingungen. Springer Vieweg, Wiesbaden, 2018.

[KKP20] Kamrueng, C.; Kittiratsatcha, S.; Polmai, S.: A Number of RC Pairs Consideration of Electrical Equivalent Circuit Model of Li-ion Battery: 2020 6th International Conference, 2020, S. 1–4.

[Ko13] Korthauer, R. Hrsg.: Handbuch Lithium-Ionen-Batterien. Springer Vieweg, Berlin, 2013.

[Kr15] Kruse, R. et al.: Computational intelligence. Eine methodische Einführung in künstliche neuronale Netze, evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze. Springer Vieweg, Wiesbaden, 2015.

[KS17] Kumari, S. A.; Srinivasan, S.: NARX models for prediction of reheater fouling: 23rd International Conference, 2017, S. 1–4.

[KSC19] Kim; Seong; Choi: Cooling Load Forecasting via Predictive Optimization of a Nonlinear Autoregressive Exogenous (NARX) Neural Network Model. Sustainability 23/11, S. 6535, 2019.

[Ku20] Kurzweil, P.: Chemie. Grundlagen, technische Anwendungen, Rohstoffe, Analytik und Experimente. Springer Fachmedien Wiesbaden; Imprint: Springer Vieweg, Wiesbaden, 2020.

[LÄ20] Lämmel, U.: Künstliche Intelligenz. Wissensverarbeitung - Neuronale Netze. Carl Hanser Verlag GmbH & Co. KG, 2020.

[LIP18] Lipu, M. S. H.; Hannan, M. A.; Hussain, A.; Saad, M. H. M.; Ayob, A.; Blaabjerg, F.: State of Charge Estimation for Lithium-Ion Battery Using Recurrent NARX Neural Network Model Based Lighting Search Algorithm, 2018.

[MRR21] MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better? | by Akshita Chugh | Analytics Vidhya | Medium. https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e, Stand: 26.08.2021.

[MW21] Choose a Multilayer Neural Network Training Function - MATLAB & Simulink - MathWorks Deutschland. https://de.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html, Stand: 28.08.2021.

[OA17] Or Aviv, Y.: Modellbasierter Entwurf eines skalierbaren Batteriemanagementsystems für Lithium-Ionen-Batterien. Bachelorarbeit, Ostfalia Hochschule für angewandte Wissenschaften, 2017.

[Qi19] Qin, X. et al.: State of Charge Estimation for Lithium-Ion Batteries Based on NARX Neural Network and UKF: 2019 IEEE 17th International Conference, 2019, S. 1706–1711.

[Sch14] Dieter Schulz: Akkus und Ladetechniken. Das Praxisbuch für alle Akku-Typen, Ladegeräte und Ladeverfahren. Franzis Verlag GmbH, München, 2014.

[SK19] Spektrum der Wissenschaft: Spektrum Kompakt – Künstliche Intelligenz, 2019.

[SKK10] Schmidt, J.; Klüver, C.; Klüver, J.: Programmierung naturanaloger Verfahren. Soft Computing und verwandte Methoden. Vieweg + Teubner, Wiesbaden, 2010.

[TGP19] Tschöke, H.; Gutzmer, P.; Pfund, T.: Elektrifizierung des Antriebsstrangs. Grundlagen - vom Mikro-Hybrid zum vollelektrischen Antrieb. Springer Berlin Heidelberg; Imprint: Springer Vieweg, Berlin, Heidelberg, 2019.

[WQ17] Wang, Q.; Gu, H.; Ye, M.; Wei, M.; Xu, X.: State of Charge Estimation for Lithium-Ion Battery Based on NARX Recurrent Neural Network and Moving Window Method, 2017.

[ZZZ] Zhang, Y.; Zhao, C.; Zhu, S.: State-of-Charge Estimation of Li-ion Batteries Based on A Hybrid Model Using Nonlinear Autoregressive Exogenous Neural Networks: 2018 IEEE PES Asia-Pacific Power, 2018, S. 772–777.

# Anhang

- Dokumentation
- Literatur
- Skripte für Matlab (Versuchsdurchführung, Training, Auswertung)
- Ergebnisse