DLR-IB-AS-BS-2020-88

Untersuchung von Deep-Learning-Methoden zur hocheffizienten Vorhersage von aerodynamischen Daten

Forschungsbericht

Autor
Philipp Stürmer

DLR Deutsches Zentrum für Luft- und Raumfahrt

Bericht des Instituts für Aerodynamik und Strömungstechnik
Report of the Institute of Aerodynamics and Flow Technology

DLR-IB-AS-BS-2020-88

Untersuchung von Deep-Learning-Methoden zur hocheffizienten Vorhersage von aerodynamischen Daten

Philipp Stürmer

Herausgeber:
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Aerodynamik und Strömungstechnik
Lilienthalplatz 7, 38108 Braunschweig

ISSN 1614-7790

Stufe der Zugänglichkeit: 1
Braunschweig, im Juli 2020

Institutsdirektor:
Prof. Dr.-Ing. habil. C.-C. Rossow

Verfasser:
Philipp Stürmer

Abteilung: Center of Computer Applications in Aerospace Science and Engineering

Abteilungsleiter:
Prof. Dr. S. Görtz

Der Bericht enthält:
91 Seiten
24 Bilder
17 Tabellen
90 Literaturstellen

Abstract

The aim of this thesis is to evaluate the potential of neural networks as alternative surrogate models for increasing the efficiency of aerodynamic data generation.
For this purpose, three different case studies are analysed, which cover both subsonic and transonic flow regimes, as well as steady and unsteady flow behaviour. For each case study a traditional surrogate model consisting of a dimensionality reduction by using proper orthogonal decomposition and a thin plate splines based interpolation as well as a neural network will be used for aerodynamic data predictions. The multilayer perceptron is used for steady flow and a long short-term memory unit based recurrent neural network for unsteady flow. Furthermore, different data preprocessing methods are investigated. To find suitable hyperparameters different strategies based on either randomly chosen or optimized trials are used and compared.
The prediction accuracy of the surrogate models is determined by comparing with the CFD reference solution. Finally, the potential of the neural network is evaluated based on both prediction accuracy and time effort as criteria by comparing with the traditional surrogate model.
It is shown that the optimized trials of different hyperparameter settings lead to more accurate predictions than the randomly chosen trials method. The prediction accuracy of the neural network increases by 39.08% when predicting the pressure distribution based on the pressure coefficients at individual surface points compared to predicting the pressure distribution as a whole. In addition, a clear trend towards more precise predictions with an increasing number of time derivatives of the angle of attack as additional input parameters is apparent in the investigation of the unsteady flow.

The use of a neural network as a surrogate model for predicting the pressure distribution generally requires more time than the use of a traditional surrogate model. However, this additional time effort can be justified by using neural networks as a surrogate model in the transonic flow regime due to a significantly higher prediction accuracy regarding occurring compression shocks, hence the use of the neural networks can be recommended for this flow regime.

Kurzfassung

Das Ziel dieser Arbeit ist es, das Potential von neuronalen Netzen als alternative Ersatzmodelle zur Effizienzsteigerung bei der Erzeugung aerodynamischer Daten zu bewerten.
Zu diesem Zweck werden drei verschiedene Fallbeispiele analysiert, die sowohl sub- als auch transsonische Strömungsbereiche sowie stationäres und instationäres Strömungsverhalten abdecken. Für jedes Fallbeispiel wird sowohl ein klassisches Ersatzmodell, das aus einer Dimensionsreduzierung durch die Proper Orthogonal Decomposition-Methode und einer Thin Plate Splines-basierten Interpolation besteht, als auch ein neuronales Netz für die Erzeugung bzw. Vorhersage von aerodynamischen Daten verwendet, wobei die Druckverteilung hierbei im Fokus steht. Als neuronales Netz wird ein Multilayer Perceptron für stationäre Strömungen bzw. ein Long Short-Term Memory-basiertes rekurrentes neuronales Netz für instationäre Strömungen verwendet. Außerdem werden verschiedene Datenaufbereitungsmethoden untersucht. Zur Bestimmung geeigneter Hyperparameter werden zudem verschiedene Methoden, die entweder auf einer zufälligen oder optimierten Suche basieren, eingesetzt und miteinander verglichen. Die Vorhersagegenauigkeit der Ersatzmodelle wird anhand eines Vergleichs mit der CFD-Referenzlösung ermittelt. Abschließend wird das Potential des neuronalen Netzes anhand der Kriterien Vorhersagegenauigkeit und zeitlicher Aufwand durch einen Vergleich mit dem klassischen Ersatzmodell bewertet.

Es wird gezeigt, dass die optimierte Suche zu genaueren Vorhersageergebnissen als die zufällige Suche führt. Die Vorhersagegenauigkeit des neuronalen Netzes steigt um 39.08%, wenn die Druckverteilung anhand der Druckbeiwerte an einzelnen Oberflächenpunkten vorhergesagt wird im Vergleich zu der Vorhersage der Druckverteilung als Gesamte. Zudem zeigt sich bei der Untersuchung der instationären Strömung ein deutlicher Trend zu genaueren Vorhersagen bei einer steigenden Anzahl an zeitlichen Ableitungen des Anstellwinkels als zusätzliche Eingangsgrößen.

Die Verwendung eines neuronalen Netzes als Ersatzmodell für die Vorhersagen der Druckverteilung erfordert generell einen höheren Zeitaufwand als die Verwendung eines klassischen Ersatzmodells. Dieser zeitliche Mehraufwand kann jedoch durch den Einsatz der neuronalen Netze als Ersatzmodell im transsonischen Bereich durch eine deutlich höhere Vorhersagegenauigkeit bezüglich auftretender Verdichtungsstöße gerechtfertigt und der Einsatz der neuronalen Netze für diesen Bereich empfohlen werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Abkürzung	**Bedeutung**
CFD	Computational Fluid Dynamics
MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
POD	Proper Orthogonal Decomposition
TPS	Thin Plate Splines
DoE	Design of Experiment
SMARTy	Surrogate Modeling for Aero Data Toolbox in Python
tanh	Tangens Hyperbolicus
ReLU	Rectified Linear Unit

1 Einleitung

In diesem Kapitel werden die Hintergründe für die in der Arbeit behandelte Thematik erläutert und die Motivation für den erarbeiteten Lösungsansatz dargelegt. Im Anschluss daran wird ein Überblick der bereits erprobten Methoden und des aktuellen Forschungsstands gegeben. Ferner wird das Ziel dieser Arbeit definiert und die Methodik, die zum Erreichen des Ziels verfolgt wird, sowie die sich daraus ergebende Struktur der Arbeit aufgeführt.

1.1 Motivation

Im Luft- und Raumfahrtbereich werden besonders im Entwurfsprozess von industriellen Produkten eine Vielzahl aerodynamischer Daten benötigt, die zur Beschreibung und Analyse von unterschiedlichen Strömungszuständen dienen. Die Strömungszustände werden meist durch dimensionslose aerodynamische Beiwerte beschrieben und sind stark vom betrachteten Objekt und einigen aerodynamischen Parametern abhängig. Hervorzuheben sind die besonders relevanten aerodynamischen Parameter, die die Anströmbedingungen beschreiben. Hierbei handelt es sich um den Anstellwinkel des Objekts zur Anströmrichtung sowie die Anströmgeschwindigkeit, die meistens dimensionslos als Machzahl beschrieben wird.
Für hochgenaue Ergebnisse werden meist numerische Strömungssimulationen (CFD, engl. *Computational Fluid Dynamics*) verwendet, bei denen einige aerodynamische Parameter vorgegeben und die resultierenden aerodynamischen Beiwerte als numerische Lösung der strömungstechnischen Erhaltungsgleichungen berechnet werden. Diese hochgenauen CFD-Simulationen sind jedoch sehr zeitaufwändig und ressourcenintensiv. Die Wahl eines einfacheren strömungsbeschreibenden Modells für die CFD-Simulationen kann den Zeitaufwand reduzieren, jedoch besteht die Gefahr, wichtige physikalische Effekte dabei nicht zu erfassen. Insbesondere für komplexe industrielle Anwendungen ist daher eine ausschließliche Verwendung von hochauflösenden CFD-Simulationen für die Erzeugung einer Datenbasis, die das aerodynamische Verhalten und die physikalischen Effekte vollständig beschreibt, meist nicht realisierbar. Stattdessen wird momentan zur Effizienzsteigerung bei der Datenerzeugung mit datengetriebenen Ersatzmodellen gearbeitet. Diese alternative Methode ist kein eigenständiger Ersatz für die CFD-Simulationen, sondern basiert auf den CFD-generierten Daten. Es wird jedoch nur ein Teil aller benötigten Daten mittels hochauflösender CFD-Simulationen berechnet, sodass sich der zeitliche Aufwand deutlich reduziert.
Die Forschung auf dem Gebiet der Ersatzmodellierung ist sehr aktiv und zeigt, dass die Ansätze zur Ersatzmodellierung vielzählig sind. Hierbei haben sich insbesondere Verfahren als brauchbar erwie-

sen, die auf einer Dimensionsreduzierung der Daten, um deren Verwendung in folgenden Prozessen zu beschleunigen, und einer anschließenden Verwendung von Interpolations-, Regressions- oder sonstigen Approximationsmodellen basieren und werden daher hier als klassisch angesehen. Basierend auf den dimensionsreduzierten Daten wird das Ersatzmodell trainiert, wodurch die nicht-linearen Beziehungen zwischen den in den CFD-Simulationen vorgegebenen aerodynamischen Parametern - den Eingangsgrößen - und den berechneten relevanten aerodynamischen Beiwerten - den Ausgangsgrößen - erlernt wird, ohne die der Strömung zugrunde liegenden physikalischen Erhaltungsgleichungen zu kennen.
Mit den erlernten nicht-linearen Beziehungen ist es möglich, die restlichen benötigten Daten, die sonst ebenfalls durch hochgenaue CFD-Simulationen generiert werden, mit dem Ersatzmodell vorherzusagen. Die Vorhersageergebnisse der Ersatzmodelle weichen durch die Approximation von den CFD-Simulationen ab. Das Ziel der Ersatzmodellierung ist es daher, möglichst exakte Vorhersageergebnisse bei einem möglichst geringen zeitlichen Rechenaufwand zu generieren.

In dieser Arbeit wird das Potential eines künstlichen neuronalen Netzwerks (kurz: neuronales Netz) als Ersatzmodell anstelle eines klassischen Ersatzmodells untersucht. Neuronale Netze überzeugen bereits in vielen Anwendungsfällen durch ihre Fähigkeit, äußerst komplexe Zusammenhänge erlernen zu können. Besonders komplexe Strömungszustände, die sich beispielsweise aus extremen Anströmbedingungen im transsonischen Bereich ergeben, werden durch die klassischen Ersatzmodelle oft nicht hinreichend genau abgebildet. Aus diesem Grund wird in dieser Arbeit das Potential des neuronalen Netzes als alternatives Ersatzmodell anhand drei verschiedener aerodynamischer Fallbeispiele untersucht und bewertet.
Der Einsatz von sowohl neuronalen Netzen und klassischen Methoden für die Ersatzmodellierung zur Bestimmung aerodynamischer Daten ist sehr vielfältig, sodass im folgenden Kapitel ein Überblick über bereits erprobte Methoden gegeben wird.

1.2 Verwandte Arbeiten

Seit drei Jahrzehnten werden neuronale Netze zur Ersatzmodellierung dynamischer Systeme und strömungstechnischer Probleme eingesetzt. Die ersten Anwendungen neuronaler Netze in der Strömungsmechanik wurden in den frühen 90ern erprobt und standen im Zusammenhang mit der *Particle Image Velocimetry* und dem Erlernen von Lösungen partieller Differentialgleichungen.[1, 2] In der Literatur sind seitdem zahlreiche Methoden zur Verwendung neuronaler Netze unterschiedlichster Topologie für die Ersatzmodellierung in unterschiedlichen aerodynamischen Anwendungen zu finden. Zur Modellierung stationärer Strömungen verwenden Santos et al. [3] und Secco et al. [4] ein *Multilayer-Perceptron* (MLP) für Vorhersagen von skalaren Beiwerten für unterschiedliche Flügelprofile bzw. Flugzeugkonfigurationen bei unterschiedlichen Anströmbedingungen. Zelong et al. [5] und Zhang et al. [6] nutzen für den gleichen Zweck ein konvolutionelles neuronales Netz (CNN, engl.: *Convolutional Neural Network*). Andere verfolgen den Ansatz, das CNN als Ersatzmodell für Vorhersagen von Druck- und Geschwindigkeitverteilungen verschiedener Flügelprofile zu verwen-

den. [7, 8] In jüngster Forschung beschäftigen sich Wu et al. [9] mit den Vorhersagen von Druck- und Geschwindigkeitsfeldern, indem sie mit einem CNN die Geometrie des Flügelprofil erfassen und anschließend mit Hilfe eines *Generative Adversarial Networks* die Feldgrößen für unterschiedliche Anströmbedingungen vorhergesagt werden.

Forschungen zeigen außerdem, dass sich die rekurrenten neuronalen Netze (RNNs) deutlich besser für die Ersatzmodellierung von instationärem Strömungsverhalten eignen als das MLP.[10, 11] Darauf aufbauend wurden unterschiedliche RNN-Topologien untersucht, von denen das *Long Short-Term Memory* (LSTM)-basierte RNN als Ersatzmodell erstmals vor drei Jahren von Wang et al. [12] eingesetzt wurde. Aus ihrer Arbeit geht hervor, dass mit dem LSTM-basierten RNN komplexe Merkmale der Strömungsdynamik genauer erfasst und der zeitliche Verlauf der Strömung besser vorhergesagt wird, als mit den zuvor erforschten neuronalen Netzen. Ein Jahr später führten Mohan und Gaitonde ähnliche Untersuchungen durch, die diesen Fund bestätigten. [13] Dies führte dazu, dass das LSTM-basierte RNN die aktuelle state-of-the-art Netzwerktopologie für die Ersatzmodellierung von instationärem Strömungsverhalten ist. Insbesondere ist hierbei die jüngste Arbeit von Li et al. [14] hervorzuheben, in der bemerkenswerte Ergebnisse für sub- und transsonische Strömungsbereiche bei zusätzlicher Betrachtung der instationären Aeroelastik erzielt werden. Die Literatur beweist, dass die Verwendung neuronaler Netze als Ersatzmodelle in vielen aerodynamischen Fallbeispielen bereits CFD-ähnliche Vorhersageergebnisse bei deutlich geringerem Zeitaufwand erzielt.

Die Forschung im Bereich der klassischen Ersatzmodellierung im aerodynamischen Kontext ist ebenfalls sehr aktiv. Insbesondere im Bereich des Flugzeugentwurfs und der dafür zu bestimmenden, komplexen Aeroelastik wird häufig auf klassische Ersatzmodelle zurückgegriffen. Dabei wird für verschiedene Anströmbedingungen das gekoppelte fluiddynamische und strukturdynamische Verhalten abhängig von unterschiedlichen Flugzeugkonfigurationen oder Flügelprofilen vorhergesagt.[15–17] Auch im Bereich der Strömungsregelung, d.h. bei variabler Anströmung das sich ergebende Strömungsverhalten durch eine Anpassung der Flugzeugkonfiguration oder des Flügelprofils zu kontrollieren, gibt es einige Ansätze.[18, 19] Sowohl der Entwurf als auch die Regelung können optimiert werden, wobei das Ziel eine optimale Konfiguration in Bezug auf ein bestimmtes Leistungsziel - wie beispielsweise ein möglichst geringer Widerstand bei möglichst großem Auftrieb - oder ein optimaler Regler sein kann. Verschiedene Methoden der klassischen Ersatzmodellierung sind zu diesem Zweck untersucht worden.[20–22]
Ein Großteil der für die oben genannten Zwecke verwendeten Ersatzmodelle bestehen dabei aus einer Dimensionsreduzierung der systembeschreibenden multidimensionalen Daten und einem Interpolations- oder Regressionsmodell, welches zur Approximation basierend auf den reduzierten Daten verwendet wird.[23] Die Dimensionsreduzierung erfolgt dabei meist auf Basis von *Balanced Truncations* [24, 25], der Krylov-Projektionsmethode [26] oder der *Proper Orthogonal Decomposition* (POD)-Methode [27]. Die Interpolation oder Regression verläuft anschließend meist basierend auf Gauß'schen Prozessen [28], verschiedenen Ersatzfunktionen wie der radialen Basisfunktion [29], den *Thin Plate Splines* (TPS) [30, 31] oder Kriging-Modellen [32].

Insbesondere die POD-Methode wird für die Dimensionsreduzierung der CFD-Daten weit verbreitet eingesetzt.[33] Auch Trübelhorn [34] hat die POD-Methode zur Dimensionsreduzierung in seiner Arbeit verwendet und drei verschiedene Interpolationsmodelle für die dimensionsreduzierten Daten untersucht. Seine Arbeit zeigt, dass TPS-Interpolation gegenüber der bikubischen Interpolation und der bilinearen Interpolation meist deutlich überlegen ist.

Die Ergebnisse der untersuchten verschiedenen neuronalen Netze bzw. verschiedenen klassischen Ersatzmodelle werden in der Literatur miteinander verglichen, damit zukünftige Untersuchungsergebnisse immer anhand eines Benchmarks bewertet werde können. Es fehlt jedoch der direkte Vergleich der neuronalen Netze mit den klassischen Ersatzmodellen. Da in einer Vielzahl der industriellen Anwendungen diese als Modell herangezogen werden, ist ein direkter Vergleich der neuronalen Netze mit den klassischen Ersatzmodellen notwendig, um das Potential für den Einsatz in der Industrie zu bewerten. Außerdem wird häufig nicht klar, welche Methode zur Auswahl der verwendeten Hyperparametereinstellungen der neuronalen Netze führt. Um diese Lücke zu schließen, wird in dieser Arbeit zum einen der systematische Entwurfsprozess neuronaler Netze beschrieben und zum anderen werden diese mit verschiedenen state-of-the-art Methoden der Ersatzmodellierung verglichen. Dies geschieht anhand der Anwendung beider Modellkonzepte an verschiedenen aerodynamischen Fallbeispielen.

1.3 Methodik und Aufbau der Arbeit

Im folgenden Kapitel 2 werden zunächst die für diese Arbeit relevanten theoretischen Grundlagen und Grundbegriffe der neuronalen Netze und der klassischen Ersatzmodelle erläutert.
Anschließend werden in Kapitel 3 auf Basis dieser theoretischen Hintergründe sowohl neuronale Netze als auch klassische Ersatzmodelle zur Analyse verschiedener Fallbeispiele verwendet. Diese Fallbeispiele umfassen eine stationäre 2D-Profilumströmung im sub- und transsonischen Bereich, eine instationäre 2D-Profilumströmung im subsonischen Bereich und eine stationäre dreidimensionale Umströmung eines Flugzeugmodells im sub- und transsonischen Bereich. Ziel in jedem Fallbeispiel ist es, die resultierende Druckverteilung auf der Oberfläche des jeweiligen Objekts abhängig von den Anströmbedingungen zu bestimmen. Im ersten Fallbeispiel sind zusätzlich der Auftriebs-, der Widerstands- und der Nickmomentenbeiwert als Zielgrößen definiert, welche mit den Ersatzmodellen vorherzusagen sind.

Bei der Untersuchung des ersten Fallbeispiels in Kapitel 3.1 wird zunächst ein besonderer Fokus auf den Einfluss verschiedener Datenaufbereitungsmethoden und verschiedener Methoden der Hyperparameteroptimierung auf die Vorhersageergebnisse der neuronalen Netze gelegt. Als Hyperparameter werden die Einstellgrößen der neuronalen Netze bezeichnet, die vor dem Training gesetzt werden müssen und einen großen Einfluss auf die Qualität der Vorhersagen der neuronalen Netze haben. Durch diese Untersuchung wird die jeweils beste Methode identifiziert und in den folgenden Fallbeispielen verwendet. Damit die Einflüsse vorheriger Ergebnisse auf die Untersuchungsmethoden der konsekutiven Fallbeispiele in Kapitel 3.2 und 3.3 eindeutig nachvollziehbar sind, ist die Arbeit so

aufgebaut, dass die Fallbeispiele chronologisch und als in sich abgeschlossene Kapitel dokumentiert sind, die jeweils aus der angewandten Methodik, den Ergebnissen, der Diskussion dieser Ergebnisse und einem Fazit bestehen.

Für jedes Fallbeispiel wird sowohl ein neuronales Netz als auch ein klassisches Ersatzmodell auf gleicher Datenbasis trainiert und für die Vorhersagen der Daten verwendet. Anhand eines Vergleichs der Vorhersageergebnisse beider Ersatzmodelle mit den Ergebnissen der CFD-Simulationen kann die jeweilige Vorhersagegenauigkeit bewertet werden. Als weiteres Bewertungskriterium wird der zeitliche Aufwand für die Erstellung der verschiedenen Ersatzmodelle definiert. Die Potentialbewertung der neuronalen Netze, die das Ziel dieser Arbeit darstellt, erfolgt für jedes Fallbeispiel separat durch einen finalen Vergleich der Ergebnisse des klassischen Ersatzmodells mit den Ergebnissen des neuronalen Netzes anhand der definierten Kriterien. Kapitel 4 fasst die Ergebnisse der einzelnen Fallbeispiele anschließend zusammen und gibt einen Ausblick über zukünftige Untersuchungsmöglichkeiten.

2 Theoretischer Hintergrund

In diesem Kapitel werden zunächst die notwendigen theoretischen Konzepte und Schlüsselbegriffe der neuronalen Netze erklärt. Verschiedene Arten neuronaler Netze werden vorgestellt und die mathematischen Hintergründe des Trainingsprozesses werden beschrieben. Außerdem werden häufig auftretende Probleme und deren mögliche Lösung sowie systematische Vorgänge zur Entwicklung eines neuronalen Netzes vorgestellt, die im weiteren Verlauf der Arbeit angewendet werden. Anschließend werden die theoretischen Grundlagen des im Rahmen dieser Arbeit verwendeten klassischen Ersatzmodells - bestehend aus einer POD-Dimensionsreduzierung mit anschließender TPS-Interpolation - erklärt. Abschließend werden die Implementierungsdetails beschrieben.

2.1 Neuronale Netze

Die Idee der neuronalen Netze stammt aus der Betrachtung des biologischen Gehirns - grundlegend bestehend aus Neuronen und Synapsen - und der dortigen Informationsübertragung. Diese wurde für ein biologisches Neuron erstmals durch McCulloch und Pitts [35] im Jahre 1943 mathematisch beschrieben. Gemeinsam mit dem von Rosenblatt [36] entwickelten Perzeptron stellt diese Beschreibung den ersten Ansatz für das technische Neuronenmodell dar. Eine weitere wichtige Veröffentlichung wurde durch Hebb [37] getätigt. Dieser beschreibt die erste Lernregel als mathematischer Ansatz zur Beschreibung des biologischen Lernprozesses. Die moderne Forschung neuronaler Netze bezieht sich jedoch kaum noch auf die Modellierung des Gehirns, sondern größtenteils auf den Einsatz neuronaler Netze in mathematischen und ingenieurwissenschaftlichen Disziplinen.[38] Dennoch sind die oben genannten Ansätze zusammengenommen die Grundlagen der heutigen Theorie der neuronalen Netze.[39] Diese wird im Folgenden, startend mit dem fundamentalen Element der neuronalen Netze, dem technischen Neuronenmodell, erläutert.

2.1.1 Technisches Neuronenmodell

Der Grundbaustein eines neuronalen Netzes ist das technische Neuronenmodell, das in Abbildung 2.1 dargestellt ist und im weiteren Verlauf abgekürzt als Neuron bezeichnet wird. Das Neuron erhält dabei N Eingangsgrößen als Eingangsvektor $\vec{x}$ (2.1), die mit den zugehörigen N Gewichten $\vec{W}$ (2.2) multipliziert und aufsummiert werden. Zu dieser gewichteten Summe wird zusätzlich ein skalarer Wert - ein Bias b - addiert und die Gesamtsumme - auch als Nettoeingang bezeichnet - wird über eine sogenannte Aktivierungsfunktion f abgebildet, die in Kapitel 2.1.4 noch ausführlich thematisiert

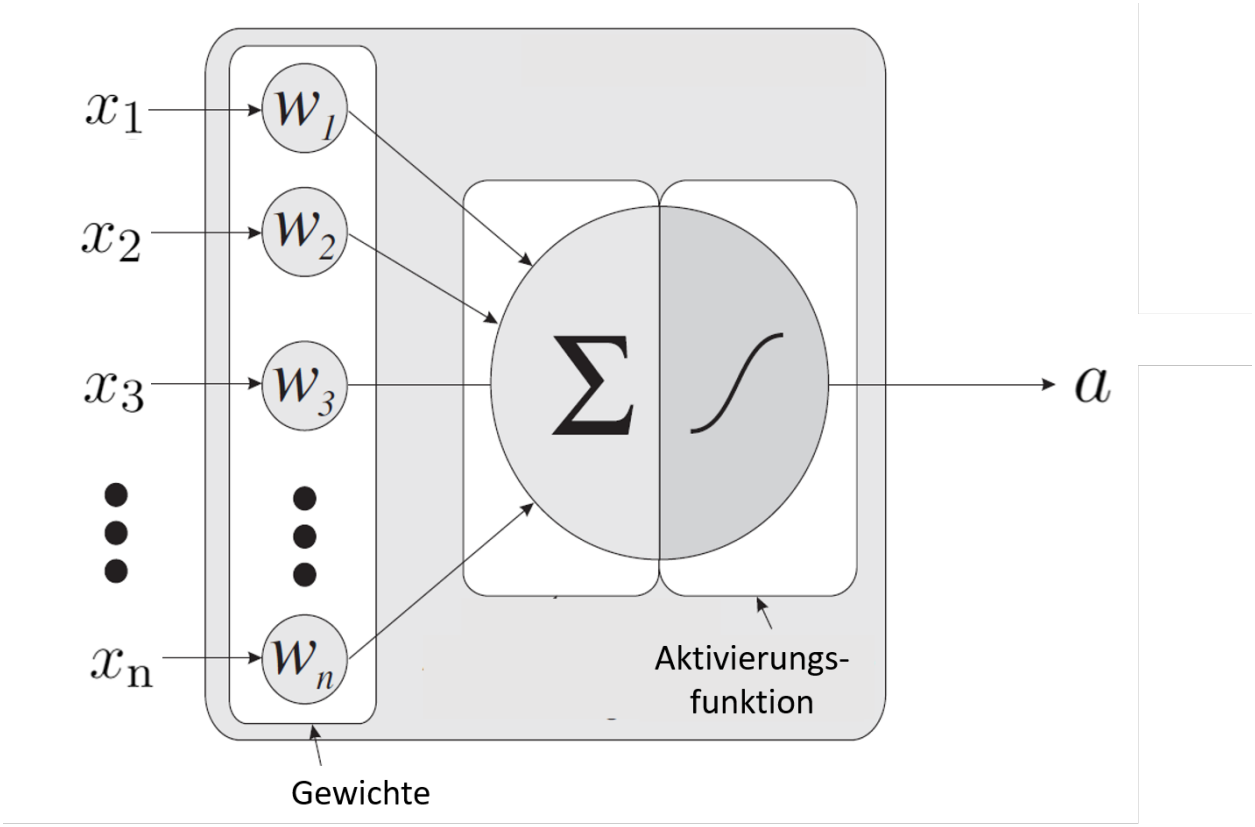


Abbildung 2.1: Technisches Neuronenmodell [40], modifiziert

wird. Die Abbildung der Aktivierungsfunktion wird Aktivierung a genannt und ergibt sich zu (2.3). Die Aktivierung stellt die Ausgangsgröße des Neurons dar.[39, 40]

$$\vec{x} = (x_1, x_2, ..., x_\mathrm{N}) \tag{2.1}$$

$$\vec{W} = (W_1, W_2, ..., W_\mathrm{N}) \tag{2.2}$$

$$a = f(\vec{x}^T \vec{W} + b) \tag{2.3}$$

Durch unterschiedliche Kombinationen und Verbindungen der Neuronen ergeben sich neuronale Netze mit verschiedenen Topologien.

2.1.2 Netzwerktopologie

Die Neuronen eines neuronalen Netzes sind typischerweise als miteinander verknüpfte Schichten von Neuronen angeordnet, die beliebig viele nicht miteinander verbundene Neuronen enthalten. Ein neuronales Netz besteht dabei mindestens aus der Eingangsschicht und der Ausgangsschicht. Alle Schichten dazwischen werden als verdeckte Schichten (engl. *hidden layers*) bezeichnet. Ein Neuron in der Eingangsschicht repräsentiert genau eine Eingangsgröße, sodass die Anzahl an Neuronen in der Eingangsschicht der Anzahl an Eingangsgrößen entspricht. Diese Neuronen enthalten die Identitätsfunktion als Aktivierungsfunktion und den Bias $b = 0$, sodass sie lediglich als Eingabemöglichkeit

der Eingangsgrößen in das neuronale Netz dienen. Analog spiegelt die Anzahl der Neuronen der Ausgabeschicht die Anzahl der vorherzusagenden Ausgangsgrößen wider.[39]

Enthält ein neuronales Netz mehr als eine versteckte Schicht, wird es in der Literatur auch tiefes neuronales Netz genannt, woher auch die englische Bezeichnung *Deep Neural Network* bzw. *Deep Learning* stammt. Je größer ein neuronales Netz ist, d.h. je mehr Schichten und Neuronen es enthält, desto größer ist seine Kapazität. Als Kapazität wird im Deep-Learning die Fähigkeit bezeichnet, eine gewisse Informationsmenge zu speichern und zu verarbeiten. Mit steigender Komplexität der zu lösenden Probleme nimmt somit die erforderliche Kapazität des neuronalen Netzes und dadurch die Anzahl an benötigten Schichten bzw. Neuronen zu.[39]

Die Aktivierungsfunktion der Neuronen ist ein essenzieller Bestandteil für die Ersatzmodellierung nicht-linearer Zusammenhänge. Aus der mathematischen Beschreibung des Neurons geht hervor, dass die Eingangsgrößen ohne die Aktivierungsfunktion linear auf die Ausgangsgröße abgebildet werden. Das neuronale Netz als Ganzes kann somit ohne Aktivierungsfunktionen als Komposition linearer Funktionen beschrieben werden, sodass es ebenfalls ausschließlich linear arbeitet. Erst durch den Einsatz nicht-linearer Aktivierungsfunktionen ist das neuronale Netz in der Lage, nicht-lineare Zusammenhänge abzubilden und somit nicht-lineare Probleme zu lösen. Meist wird für jedes Neuron einer Schicht die gleiche Aktivierungsfunktion gewählt. Als Aktivierungsfunktion kann theoretisch jede beliebige Funktion verwendet werden, von denen zahlreiche in den vergangenen Jahrzehnten erforscht wurden und in Kapitel 2.1.4 aufgeführt sind. Die Wahl geeigneter Aktivierungsfunktionen hängt stark von den zu modellierenden Zusammenhängen und der dafür verwendeten Netzwerktopologie ab, von denen einige zunächst erläutert werden.[41–43]

2.1.2.1 Multilayer Perceptron

Das MLP mit einer beliebigen Anzahl an Schichten und Neuronen pro Schicht ist in Abbildung 2.2 verdeutlicht. Die Struktur des MLPs ist die einfachste und am weitesten verbreitete Topologie der neuronalen Netze. Es ist ein vollständig verbundenes (engl. *fully connected* oder *dense*) neuronales Netz, das aus Schichten besteht, in denen jedes Neuron mit sämtlichen Neuronen der vorherigen und folgenden Schicht verbunden ist. Es ist außerdem ein vorwärts gerichtetes neuronales Netz (engl. *feedforward neural network*), in dem der Informationsfluss beim Berechnen der Vorhersagen nur in eine Richtung verläuft - von der Eingangsschicht zur Ausgangsschicht. Diese Netzwerktopologie weist, basierend auf ihren Eigenschaften, in der Literatur sehr vielfältige Bezeichnungen auf. Sie wird in dieser Arbeit anlehnend an Goodfellow et al. [38] als MLP bezeichnet und kann mathematisch wie folgt definiert werden.

Angenommen, es gibt L Schichten ohne Berücksichtigung der Eingangsschicht. Eine beliebige Schicht ℓ enthält N_ℓ Neuronen $X_1^{(\ell)}$, $X_2^{(\ell)}$, …, $X_{N_\ell}^{(\ell)}$, von denen jedes die gleiche Aktivierungsfunktion $f^{(\ell)}$ aufweist. Diese Neuronen empfangen Eingangsgrößen von den Neuronen in der vorherigen Schicht, $\ell-1$. Das Neuron $X_j^{(\ell)}$ empfängt beispielsweise die Aktivierung $a_i^{(\ell-1)}$ des Neurons $X_i^{(\ell-1)}$ multipliziert mit dem zugehörigen Gewicht $W_{ij}^{(\ell)}$ als Eingangsgröße. Die Gewichte können als eine $N_{\ell-1} \times N_\ell$

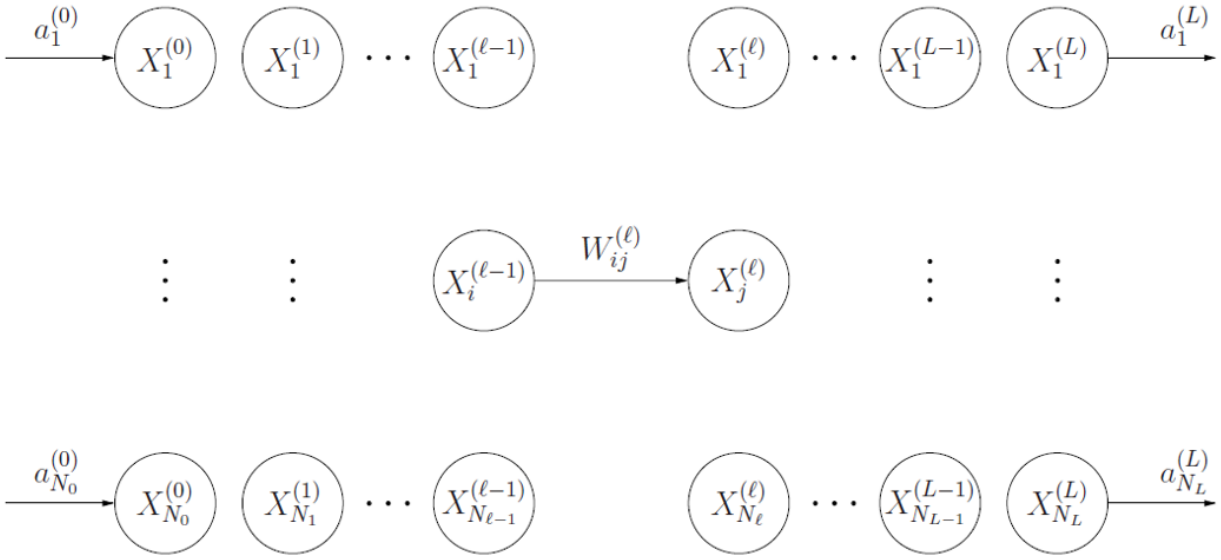


Abbildung 2.2: Allgemeine Struktur eines Multilayer Perceptrons [44]

dimensionale Matrix $\mathbf{W}^{(\ell)}$ dargestellt werden, deren Einträge sich zu $W_{ij}^{(\ell)}$ für $i = 1, 2, \ldots, N_{\ell-1}$ und $j = 1, 2, \ldots, N_\ell$ ergeben. Das Neuron $X_j^{(\ell)}$ enthält außerdem den Bias $b_j^{(\ell)}$ und seine Aktivierung ist $a_j^{(\ell)}$.

Zur Vereinfachung wird die gewichtete Summe mit addiertem Bias als Nettoeingang $n_j^{(\ell)}$ bezeichnet und nach (2.4) definiert, wobei $j = 1, 2, \ldots, N_\ell$. Die Aktivierung $a_j^{(\ell)}$ des Neurons $X_j^{(\ell)}$ ergibt sich dadurch zu (2.5).

$$n_j^{(\ell)} = \sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)} \tag{2.4}$$

$$a_j^{(\ell)} = f^{(\ell)}(n_j^{(\ell)}) = f^{(\ell)}(\sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)}) \tag{2.5}$$

Die nullte Schicht kann als Eingangsschicht betrachtet werden. Wenn der Eingangsvektor $\vec{x}$ aus N Eingangsgrößen besteht, dann ist $N_0 = N$ und die Neuronen in der Eingangsschicht haben die Aktivierungen $a_i^{(0)} = x_i$, mit $i = 1, 2, \ldots, N_0$.

Die L-te Schicht des Netzwerks ist die Ausgangsschicht. Angenommen, der Ausgangsvektor $\vec{y}$ besteht aus M Ausgangsgrößen, dann gilt $N_L = M$ und die Ausgangsgrößen sind gegeben durch $a_j^{(L)}$, wobei $j = 1, 2, \ldots, M$.
Für beliebige Eingangsgrößen können die obigen Gleichungen verwendet werden, um die Ausgangsgrößen eines MLPs zu berechnen. Diese Berechnung wird sowohl während des Trainings als auch für die Vorhersage von Daten verwendet, was in Kapitel 2.1.3 noch ausführlich beschrieben wird.

Es sei darauf hingewiesen, dass die Definition eines MLP in der Literatur von der hier verwendeten Definition nach [44] abweichen kann.

Neben den ausschließlich vorwärts gerichteten neuronalen Netzen wie den MLPs, gibt es Topologien mit verschiedenen Arten der Rückkopplung.

2.1.2.2 Rekurrente neuronale Netze

Verschiedene Arten der Rückkopplung können in neuronalen Netzen durch rekurrente Verbindungen realisiert werden. Die resultierenden neuronalen Netze sind deshalb allgemein als RNNs (engl. *Recurrent Neural Networks*) bekannt. Sie werden meist für die Modellierung dynamischer Systeme verwendet, deren zeitliche Abfolge von Ereignissen durch zeitlich zusammenhängende Daten beschrieben wird. Der Ansatz besteht darin, diese zeitliche Abhängigkeit der Daten durch die Rückkopplungen zu erfassen. Die Anzahl der verschiedenen Rückkopplungsarten sind dabei unzählig. Als einfache Beispiele sind die Rückkopplungen zwischen verschiedenen Schichten sowie die direkte Rückkopplung des Neuronenausgangs mit dem Neuroneneingang zu nennen.[39] Letztere ist in Abbildung 2.3 dargestellt, wobei sich zur Beschreibung der RNNs in der Literatur andere Bezeichnungen durchgesetzt haben als zur Beschreibung der MLPs. Die hier verwendete Notation bezieht sich auf [45].

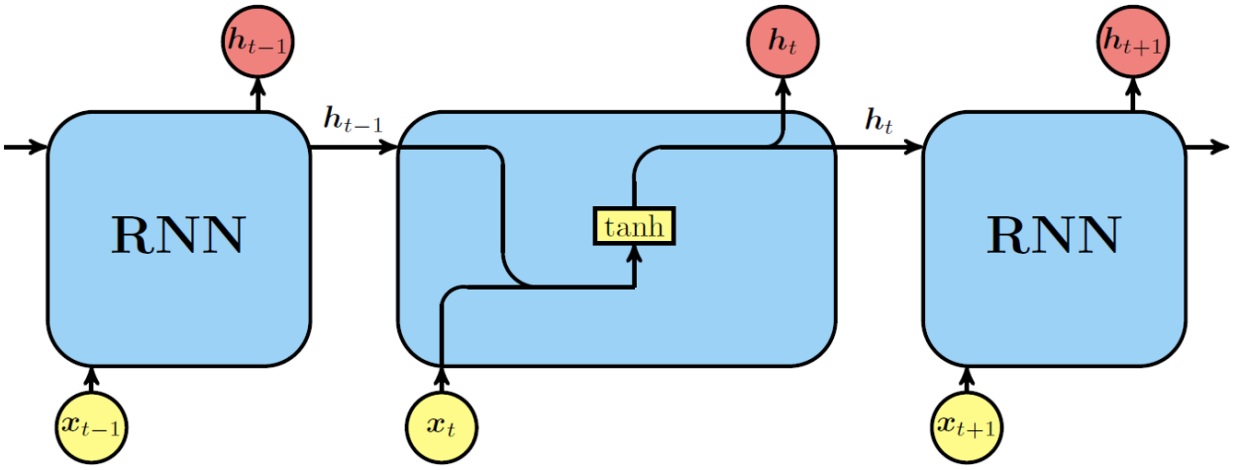


Abbildung 2.3: Rückkopplung eines Neuronenausgangs auf den Neuroneneingang durch eine rekurrente Verbindung [45]

Hierbei stellt x_t den Eingangsvektor und h_t den Ausgangsvektor zum Zeitschritt t dar. Der Index t-1 bzw. t+1 bezeichnet den vorherigen bzw. folgenden Zeitschritt. Es ist zu erkennen, dass der Ausgangsvektor eines Zeitschritts als zusätzlicher Eingangsvektor für den folgenden Zeitschritt eingeht. Die vorhandenen Gewichte und Biases sind in Abbildung 2.3 nicht gezeigt, die grundlegenden Berechnungen sind jedoch analog zu denen des in Kapitel 2.1.1 beschriebenen Neurons. Die Gewichte und Biases eines neuronalen Netzes werden zusammengefasst auch als Netzparameter oder trainierbare Parameter bezeichnet.

Durch diese einfachen Ansätze gelingt es, die Ergebnisse des vorherigen Zeitschritts in der Berechnung des nächsten Zeitschritts und somit deren Ergebnisse indirekt auch für weitere Zeitschritte zu berücksichtigen und die zeitliche Abhängigkeit dadurch zu modellieren. Eine Informationsspeicherung sämtlicher vorheriger Zeitschritte funktioniert jedoch deutlich besser mit komplexeren Rückkopplungen, wie sie beispielsweise durch die LSTM-Einheit realisiert werden. Ihr Funktionsprinzip und die zugrunde liegenden Gleichungen werden im Folgenden erklärt.

Long-Short-Term-Memory

Die im Jahre 1997 von Hochreiter und Schmidhuber [46] entwickelte LSTM-Einheit (engl. *LSTM unit* oder *LSTM cell*) ist in Abbildung 2.4 gezeigt und stellt das Grundelement der state-of-the-art RNNs für die Modellierung von langen Zeitreihen dar. Der Grund dafür ist der, dass im Gegensatz zu vorher erprobten RNNs die LSTM-basierten RNNs deutlich besser in der Lage sind, die Informationen sämtlicher vorheriger Zeitschritte in der Berechnung fortschreitender Zeitschritte zu berücksichtigen.

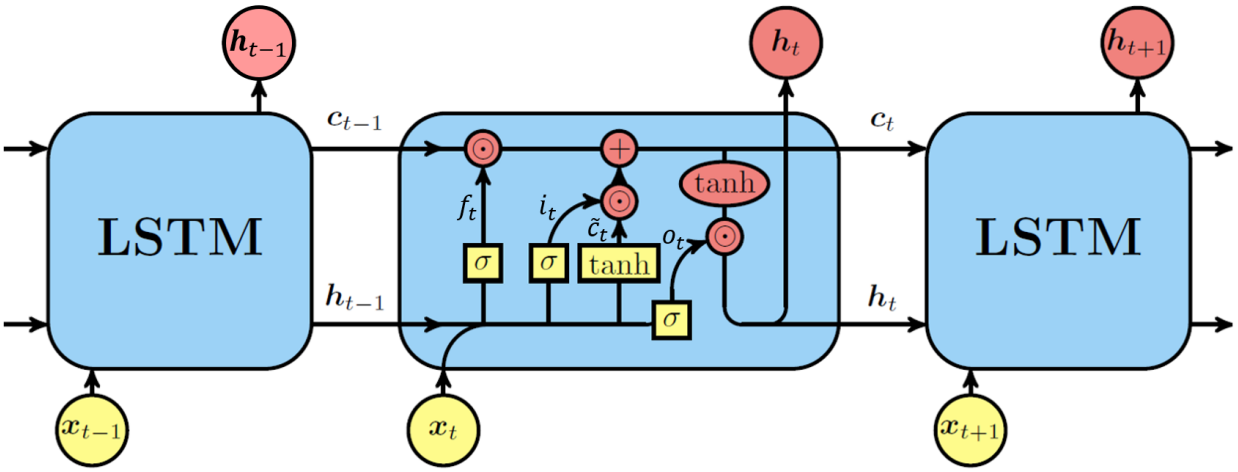


Abbildung 2.4: Struktureller Aufbau einer Long Short-Term Memory-Einheit [45], modifiziert

Analog zur Abbildung 2.3 wird auch bei der LSTM-Einheit die Rückkopplung von Ausgang zu Eingang realisiert, wodurch Informationen des vorherigen Zeitschritts berücksichtigt werden. Hinzu kommt der Zellstatus (engl. *cell state*) c_t als weiterer Eingangsvektor. Dieser aktualisiert sich mit jeder Berechnung eines Zeitschritt nach (2.9) und stellt somit gewissermaßen einen abrufbaren Speicher mit Informationen der vorherigen Zeitschritte zum Zeitpunkt t dar.

Außerdem verfügt die LSTM-Einheit über drei sogenannte *Gates*, die die Aktualisierung des Zellstatus regulieren und in Abbildung 2.4 als rote Kreisflächen mit zentriertem Multiplikationssymbol dargestellt sind. Von links nach rechts stellen sie die folgenden Gates dar, denen unterschiedliche Zwecke zuzuordnen sind:

1. Das Vergessens-Gate (engl. *forget gate*)
   Bestimmt den Einfluss des vorherigen Zellstatus c_{t-1} auf den aktuellen Zellstatus c_t

2. Das Eingangs-Gate (engl. *input gate*)
   Bestimmt den Einfluss des Eingangsvektors x_t auf den Zellstatus c_t

3. Das Ausgangs-Gate (engl. *output gate*)
   Bestimmt den Einfluss des Zellstatus c_t auf den Ausgangsvektor h_t

Die LSTM-Einheit kann allgemein durch (2.6)-(2.12) beschrieben werden, wobei f_t der Vergessens-Gate-Vektor, i_t der Eingangs-Gate-Vektor und o_t der Ausgangs-Gate-Vektor ist sowie W_f, W_i, W_o und W_c die Gewichtsmatritzen und b_f b_i b_o und b_c die Biases sind. Der mathematische Operator $\circ$ bedeutet eine elementweise Multiplikation und σ steht für die Sigmoid-Funktion nach (2.12).[45]

$$f_\mathrm{t} = \sigma(W_\mathrm{f} \cdot [h_\mathrm{t\text{-}1}, x_\mathrm{t}] + b_\mathrm{f}) \tag{2.6}$$

$$i_\mathrm{t} = \sigma(W_\mathrm{i} \cdot [h_\mathrm{t\text{-}1}, x_\mathrm{t}] + b_\mathrm{i}) \tag{2.7}$$

$$\widetilde{c}_\mathrm{t} = tanh(W_\mathrm{c} \cdot [h_\mathrm{t\text{-}1}, x_\mathrm{t}] + b_\mathrm{c}) \tag{2.8}$$

$$c_\mathrm{t} = f_\mathrm{t} \circ c_\mathrm{t\text{-}1} + i_\mathrm{t} \circ \widetilde{c}_\mathrm{t} \tag{2.9}$$

$$o_\mathrm{t} = \sigma(W_\mathrm{o} \cdot [h_\mathrm{t\text{-}1}, x_\mathrm{t}] + b_\mathrm{o}) \tag{2.10}$$

$$h_\mathrm{t} = o_\mathrm{t} \circ tanh(c_\mathrm{t}) \tag{2.11}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.12}$$

Die Sigmoid-Funktion eignet sich aufgrund ihrer Eigenschaft, reelle Zahlen auf den Wertebereich (0, 1) abzubilden, für die mathematische Beschreibung der Aufgaben der verschiedenen Gates sehr gut, was anhand des Vergessens-Gate beispielhaft verdeutlicht wird. Durch das Vergessens-Gate wird der Einfluss des Zellstatus $c_\mathrm{t\text{-}1}$ zum Zeitschritt t-1 auf den Zellstatus c_t des aktuellen Zeitschritts t geregelt. Zu diesem Zweck wird eine Multiplikation von $c_\mathrm{t\text{-}1}$ mit der Abbildung einer Sigmoid-Funktion durchgeführt. Aufgrund ihres Wertebereichs wird daher mit Werten zwischen 0 und 1 multipliziert. Eine Multiplikation mit 0 kann so interpretiert werden, dass die Informationen bzw. Erinnerungen aus vorherigen Zeitschritten komplett vergessen werden, während bei einer Multiplikation mit 1 die gesamten Informationen vorheriger Zeitschritte berücksichtigt werden. Die Einflussregelungen auf den Zellstatus in den anderen Gates können analog interpretiert werden.[46]

2.1.3 Trainingsprozess

Während des Trainingsprozesses des neuronalen Netzes werden die Zusammenhänge zwischen den systembeschreibenden Eingangsvektoren $\vec{x}^{\,(q)}$ nach (2.13) und den Zielvektoren $\vec{t}^{\,(q)}$ nach (2.14) für $q = 1, 2, \ldots, Q$ erlernt.

$$\vec{x}^{\,(q)} = \begin{bmatrix} x_1^{(q)} & x_2^{(q)} & \ldots & x_{N_0}^{(q)} \end{bmatrix} \tag{2.13}$$

$$\vec{t}^{\,(q)} = \begin{bmatrix} t_1^{(q)} & t_2^{(q)} & \ldots & t_{N_L}^{(q)} \end{bmatrix} \tag{2.14}$$

Hierbei gibt Q die Anzahl an vorliegenden Datenpaaren an, die beispielsweise durch CFD-Simulationen erzeugt wurden. Ein Eingangsvektor $\vec{x}^{(q)}$ hat hier N_0 Eingangsgrößen und ein Zielvektor $\vec{t}^{(q)}$ hat N_L Zielgrößen.

Das Training gliedert sich grob in die folgenden drei Schritte:

1. Vorhersageprozess (engl. *forward pass*)
2. Quantifizierung der Abweichung
3. Anpassung der Netzparameter

Als Vorhersageprozess wird die Berechnung eines Ausgangsvektors für einen Eingangsvektor bezeichnet. Der berechnete Ausgangsvektor wird auch Vorhersage genannt und berechnet sich für das MLP nach den Gleichungen (2.1)-(2.5) aus Kapitel 2.1.2.1. Für eine Vorhersage nach t Anpassungen der Netzparameter wird für die folgende Betrachtung der Eingangsvektor durch $\vec{x}(t)$, die Vorhersage durch $\vec{y}(t)$ und der Zielvektor durch $\vec{t}(t)$ beschrieben.

Im zweiten Schritt wird die Abweichung der Vorhersage des neuronalen Netzes von dem Zielvektor quantifiziert, welche indirekt ein Maß für Vorhersagegenauigkeit des neuronalen Netzes ist. Zur Berechnung dieser Abweichung wird eine sogenannte Kostenfunktion E (engl. *cost function* oder *loss function*) definiert. Die gängigste Kostenfunktion für Regressionsprobleme ist die über die Anzahl der Vektoreinträge gemittelte L2-Norm des Differenzenvektors zwischen Vorhersage und Zielvektor, die auch als MSE (engl. *Mean Squared Error*) bekannt und in (2.15) für eine Vorhersage definiert ist.[47] Die Vorhersage und dadurch auch die Kostenfunktion ist dabei direkt abhängig von allen Netzparametern.

$$MSE = \frac{1}{N_L} \sum_{i=1}^{N_L} (t_i - y_i)^2 \tag{2.15}$$

$$MAE = \frac{1}{N_L} \sum_{i=1}^{N_L} |t_i - y_i| \tag{2.16}$$

Eine geeignete Wahl der Kostenfunktion ist für den Trainingsprozess ein wichtiger Aspekt, was beispielhaft anhand des Vergleichs zwischen dem MSE und einer weiteren möglichen Kostenfunktion, dem MAE (engl. *Mean Absolute Error*) nach (2.16), gezeigt wird.

Die Abweichung zwischen der Vorhersage und dem Zielvektor geht also ihren Bezeichnungen entsprechend beim MSE quadratisch und beim MAE linear in die Kostenfunktion ein. Dadurch fallen Ausreißer in den Daten beim MSE deutlich stärker ins Gewicht als beim MAE. Dies hat Auswirkungen auf die vom Gradienten der Kostenfunktion abhängigen Netzparameteranpassungen, die im dritten Trainingsschritt durchgeführt werden.[38]

Das Ziel des dritten Trainingsschritts ist die Optimierung der Vorhersagegenauigkeit des neuronalen Netzes, was gleichbedeutend mit der Minimierung der Kostenfunktion ist, die direkt von allen Netzparametern abhängt. Zu diesem Zweck wird die berechnete Kostenfunktion partiell nach den jeweiligen Netzparametern abgeleitet, um den Gradienten zu bestimmen. Der Gradient enthält Information über den Einfluss des Netzparameters auf die Kostenfunktion. Je größer der Betrag der partiellen Ableitung ist, desto stärker ändert sich die Kostenfunktion durch eine Änderung des Netzparameters. Das Vorzeichen des Gradienten bestimmt dabei die Richtung der Kostenfunktionsänderung abhängig vom Netzparameter. Diese Abhängigkeit wird ausgenutzt, um die Netzparameter abhängig von ihrem Einfluss auf die Kostenfunktion so anzupassen, dass die Kostenfunktion minimiert wird. Für diese Optimierung eignen sich einige Algorithmen. Die heutzutage am häufigsten verwendeten Algorithmen, wie beispielsweise der Adam-Algorithmus [48],beruhen meist auf dem Ansatz des von Rumelhart et al. [49] im Jahre 1986 entwickelten *Back-Propagation*-Algorithmus. Dieser führte zu einer enormen Beschleunigung des Trainings im Vergleich zu vorherigen Algorithmen, sodass die Verwendung von mehrschichtigen neuronalen Netzen erstmals für industrielle Anwendungen interessant wurde.[45] Der Back-Propagation-Algorithmus wird im Folgenden beispielhaft anhand des in Abbildung 2.2 dargestellten MLPs für die Berechnung der Netzparameter nach $t+1$ Anpassungen beschrieben. Dabei wird von der bereits berechneten Kostenfunktion E ausgegangen, die in diesem Beispiel als L2-Norm des Differenzenvektors zwischen einer Vorhersage nach t Anpassungen und zugehörigem Zielvektor gewählt wird.

Die Netzparameter berechnen sich nach (2.17) und (2.18), wobei $lr > 0$ die sogenannte Lernrate ist. Sie stellt einen skalierenden Faktor der Netzparameteranpassungen dar und wird vor dem Training manuell definiert, was in Kapitel 2.1.6 noch detaillierter erläutert wird.

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - lr\ \frac{\partial E(t)}{\partial w_{ij}^{(\ell)}} \tag{2.17}$$

$$b_{j}^{(\ell)}(t+1) = b_{j}^{(\ell)}(t) - lr\ \frac{\partial E(t)}{\partial b_{j}^{(\ell)}} \tag{2.18}$$

Um diese partiellen Ableitungen zu berechnen, wird zunächst die Abhängigkeit von $E(t)$ bezüglich der Netzparameter aufgezeigt. $E(t)$ hängt dabei explizit von der Vorhersage $\vec{y}(t)$ ab, ergo der Aktivierungen $\vec{a}^{\,(L)}(t)$ der letzten Schicht L, wie in (2.5) definiert. Diese hängt dabei vom Nettoeingang $\vec{n}^{\,(L)}(t)$ nach (2.4) ab. $\vec{n}^{\,(L)}(t)$ ist durch die Aktivierungen der vorherigen Schicht und die Netzparameter der Schicht L gegeben. Es ergibt sich demnach für die Kostenfunktion die folgende Gleichung, wobei der Bezug zu t für eine bessere Übersicht zunächst nicht weiter aufgeführt wird.

$$E = \|\vec{y} - \vec{t}\,\|^2 = \|\vec{a}^{\,(L)} - \vec{t}\,\|^2 = \|f^{(L)}(\vec{n}^{\,(L)}) - \vec{t}\,\|^2 \tag{2.19}$$

$$= \left\| f^{(L)} \left(\sum_{i=1}^{N_{L-1}} a_i^{(L-1)} w_{ij}^{(L)} + b_j^{(L)} \right) - \vec{t} \right\|^2 \tag{2.20}$$

Unter Verwendung der Kettenregel berechnen sich die partiellen Ableitungen von E in Bezug auf die Elemente von $\mathbf{W}^{(L)}$ und $\vec{b}^{(L)}$ zu (2.21) und (2.22).

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = \sum_{n=1}^{N_L} \frac{\partial E}{\partial n_n^{(L)}} \frac{\partial n_n^{(L)}}{\partial w_{ij}^{(L)}} \tag{2.21}$$

$$\frac{\partial E}{\partial b_j^{(L)}} = \sum_{n=1}^{N_L} \frac{\partial E}{\partial n_n^{(L)}} \frac{\partial n_n^{(L)}}{\partial b_j^{(L)}} \tag{2.22}$$

Für die Anwendung der Kettenregel wird die Summe in der obigen Gleichung benötigt. Hierzu wird ein sogenannter Empfindlichkeitsvektor für eine allgemeine Schicht ℓ definiert, der aus den Komponenten nach (2.23) für $n = 1, 2, \ldots, N_\ell$ besteht. Er stellt den Einfluss auf die Kostenfunktionsänderung bei einer Änderung des Nettoeingangs des Neurons $X_n^{(\ell)}$ dar und bestimmt maßgeblich zusammen mit der Aktivierung der vorherigen Schicht $\ell - 1$ den Gradienten der Kostenfunktion und somit die Stärke der Anpassung der zugehörigen Netzparameter. Für die Schicht L wird der Empfindlichkeitsvektor durch die Verwendung der Kettenregel zu (2.24) für $n = 1, 2, \ldots, N_L$ bestimmt, wobei $\dot{f}$ die Ableitung der Aktivierungsfunktion f bezeichnet.

$$s_n^{(\ell)} = \frac{\partial E}{\partial n_n^{(\ell)}} \tag{2.23}$$

$$s_n^{(L)} = 2\left(a_n^{(L)} - t_n\right) \dot{f}^{(L)}(n_n^{(L)}) \tag{2.24}$$

$$\frac{\partial n_n^{(L)}}{\partial w_{ij}^{(L)}} = \frac{\partial}{\partial w_{ij}^{(L)}} \left(\sum_{m=1}^{N_{L-1}} a_m^{(L-1)} w_{mn}^{(L)} + b_n^{(L)} \right) = \delta_{nj} a_i^{(L-1)} \tag{2.25}$$

$$\frac{\partial n_n^{(L)}}{\partial b_j^{(L)}} = \delta_{nj} \tag{2.26}$$

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = a_i^{(L-1)}\ s_j^{(L)} \tag{2.27}$$

$$\frac{\partial E}{\partial b_j^{(L)}} = s_j^{(L)} \tag{2.28}$$

Mit (2.25) und (2.26) ergeben sich die partiellen Ableitungen der Kostenfunktion E nach den Netzparametern der Schicht L somit zu (2.27) und (2.28).

Für eine beliebige Schicht ℓ ergeben sich die partiell abgeleiteten Kostenfunktionen zu (2.29) und (2.30).

$$\frac{\partial E}{\partial w_{ij}^{(\ell)}} = \sum_{n=1}^{N_\ell} \frac{\partial E}{\partial n_n^{(\ell)}} \frac{\partial n_n^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \sum_{n=1}^{N_\ell} s_n^{(\ell)} \frac{\partial n_n^{(\ell)}}{\partial w_{ij}^{(\ell)}} \tag{2.29}$$

$$\frac{\partial E}{\partial b_j^{(\ell)}} = \sum_{n=1}^{N_\ell} \frac{\partial E}{\partial n_n^{(\ell)}} \frac{\partial n_n^{(\ell)}}{\partial b_j^{(\ell)}} = \sum_{n=1}^{N_\ell} s_n^{(\ell)} \frac{\partial n_n^{(\ell)}}{\partial b_j^{(\ell)}} \tag{2.30}$$

Die noch fehlende Beschreibung der partiellen Ableitung des Empfindlichkeitsvektor der beliebigen Schicht l, der nach (2.31) für $j = 1, 2, \ldots, N_\ell$ definiert ist, ergibt sich durch eine Vereinfachung mithilfe von (2.32) und (2.33), sodass die partiellen Ableitungen der Kostenfunktion schließlich nach (2.34) und (2.35) definiert werden können.

$$n_n^{(\ell)} = \sum_{m=1}^{N_{\ell-1}} a_m^{(\ell-1)} w_{mn}^{(\ell)} + b_n^{(\ell)} \tag{2.31}$$

$$\frac{\partial n_n^{(\ell)}}{\partial w_{ij}^{(\ell)}} = \delta_{nj} a_i^{(\ell-1)} \tag{2.32}$$

$$\frac{\partial n_n^{(\ell)}}{\partial b_j^{(\ell)}} = \delta_{nj} \tag{2.33}$$

$$\frac{\partial E}{\partial w_{ij}^{(\ell)}} = a_i^{(\ell-1)}\ s_j^{(\ell)} \tag{2.34}$$

$$\frac{\partial E}{\partial b_j^{(\ell)}} = s_j^{(\ell)} \tag{2.35}$$

Ab hier wird der Bezug zu t wieder dokumentiert. Die Anpassungen der Gewichte und Biases werden dann durch (2.36) und (2.37) definiert. Für die Bestimmung der Empfindlichkeitsvektoren $\vec{s}^{(\ell)}$ für $\ell = 1, 2, \ldots, L-1$, die durch (2.38) für $j = 1, 2, \ldots, N_\ell$ definiert sind, muss die partielle Ableitung der Kostenfunktion E nach dem Nettoeingang $n_j^{(\ell)}$ bekannt sein.

$$w_{ij}^{(\ell)}(t+1) = w_{ij}^{(\ell)}(t) - lr\ a_i^{(\ell-1)}(t)\ s_j^{(\ell)}(t) \tag{2.36}$$

$$b_j^{(\ell)}(t+1) = b_j^{(\ell)}(t) - lr\ s_j^{(\ell)}(t) \tag{2.37}$$

$$s_j^{(\ell)} = \frac{\partial E}{\partial n_j^{(\ell)}} \tag{2.38}$$

Der Schlüssel zur Berechnung dieser partiellen Ableitungen liegt in der rekursiven Abhängigkeit der Empfindlichkeitsvektoren, nämlich ist $n_j^{(\ell)}$ wiederum von $n_i^{(\ell-1)}$ für $i = 1, 2, \ldots, N_{\ell-1}$ abhängig, da $n_j^{(\ell)}$ von der Aktivierung $a_i^{(\ell-1)}$ der vorhergehenden Schicht $\ell - 1$ abhängt, die wiederum von $n_i^{(\ell-1)}$

abhängt. Es gilt deshalb (2.39)-(2.40) für $j = 1, 2, \ldots, N_\ell$, sodass die Empfindlichkeitsvektoren der Schicht $\ell - 1$ schließlich durch (2.41)-(2.44) definiert werden können.

$$n_j^{(\ell)} = \sum_{i=1}^{N_{\ell-1}} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)} \tag{2.39}$$

$$= \sum_{i=1}^{N_{\ell-1}} f^{(\ell-1)}(n_i^{(\ell-1)}) w_{ij}^{(\ell)} + b_j^{(\ell)} \tag{2.40}$$

$$s_j^{(\ell-1)} = \frac{\partial E}{\partial n_j^{(\ell-1)}} = \sum_{i=1}^{N_\ell} \frac{\partial E}{\partial n_i^{(\ell)}} \frac{\partial n_i^{(\ell)}}{\partial n_j^{(\ell-1)}} \tag{2.41}$$

$$= \sum_{i=1}^{N_\ell} s_i^{(\ell)} \frac{\partial}{\partial n_j^{(\ell-1)}} \left(\sum_{m=1}^{N_{\ell-1}} f^{(\ell-1)}(n_m^{(\ell-1)}) w_{mi}^{(\ell)} + b_i^{(\ell)} \right) \tag{2.42}$$

$$= \sum_{i=1}^{N_\ell} s_i^{(\ell)} \dot{f}^{(\ell-1)}(n_j^{(\ell-1)}) w_{ji}^{(\ell)} \tag{2.43}$$

$$= \dot{f}^{(\ell-1)}(n_j^{(\ell-1)}) \sum_{i=1}^{N_\ell} w_{ji}^{(\ell)} s_i^{(\ell)} \tag{2.44}$$

Somit hängt die Empfindlichkeit eines Neurons der Schicht $\ell - 1$ von den Empfindlichkeiten aller Neuronen der Schicht ℓ ab. Dadurch entsteht eine rekursive Abhängigkeit für die Empfindlichkeitsvektoren des gesamten Netzes, da zunächst ausschließlich der Empfindlichkeitsvektor der letzten Schicht L berechnet werden kann. Für eine beliebige Schicht können die Empfindlichkeitsvektoren daher unter Ausnutzung der rekursiven Abhängigkeit zwischen den Schichten bestimmt werden, indem die Empfindlichkeitsvektoren ausgehend von der letzten Schicht bis hin zur betrachteten Schicht rückwärts propagiert werden. Aus diesem Grund wird der Algorithmus Back-Propagation-Algorithmus genannt.[49] Heutzutage wird meist ein sogenanntes Mini-Batch-Training durchgeführt. Ein Batch ist dabei ein Subdatensatz aller zur Verfügung stehenden Datenpaare Q. Als Batchgröße bs wird die Anzahl aller Datenpaare genannt, die in einem Subdatensatz enthalten sind. Im obigen Beispiel entspricht $bs \in [1, 2, ..., Q]$. Das Training mit einer Batchgröße von $bs = Q$ wird als Batch-Training bezeichnet, findet jedoch selten Verwendung. Das Mini-Batch-Training zeichnet sich dadurch aus, dass der Back-Propagation-Algorithmus zur Anpassung der Netzparameter nicht aufbauend auf der Vorhersage eines Datenpaares, sondern auf der Vorhersage von bs Datenpaaren durchgeführt wird. Die einzelnen Vorhersagen werden dabei zwischengespeichert, bis für alle bs Eingangsvektoren die Vorhersagen bestimmt sind. Erst dann wird aufbauend auf der über die bs gemittelten Vorhersagen die Kostenfunktion bestimmt und die Netzparameter werden angepasst. Dadurch werden starke Netzparameteranpassungen durch Ausreißer vermieden, es erfordert allerdings einen höheren Arbeitsspeicher. Sobald die Netzparameter aufbauend auf den Vorhersagen aller Q vorhandenen Datenpaare angepasst worden sind, wird von einer Trainingsepoche oder kurz

von einer Epoche gesprochen. Mit zunehmender Anzahl an Epochen steigt daher in der Theorie die Vorhersagegenauigkeit und zugleich der zeitliche Aufwand des Trainings.[50]

Durch die Beschreibung des Back-Propagation-Algorithmus wird ersichtlich, dass die Ableitung der Aktivierungsfunktion $\dot{f}$ einer Schicht ℓ linear in die Berechnung des Empfindlichkeitsvektors eingeht. Damit geht sie auch linear in die Berechnung des Gradienten der Kostenfunktion E der Schicht ℓ ein, der die Netzparameteranpassungen maßgeblich bestimmt. Zusätzlich gehen die Gradienten der tieferen Schichten $\ell_d > l$ durch die rekursive Abhängigkeit multiplikativ in die Berechnung des Gradienten der Schicht ℓ ein. Die Ableitung der Aktivierungsfunktion $\dot{f}$ hat somit einen großen Einfluss auf das Training und wird im folgenden Kapitel genauer erläutert.

2.1.4 Aktivierungsfunktionen

Eine ungeeignete Wahl der Aktivierungsfunktion kann zu instabilen Gradienten der Kostenfunktion und schließlich zu verschiedenen Problemen führen. Eines davon entsteht durch verschwindende Gradienten (engl. *vanishing gradients*), bei dem der Gradient der Kostenfunktion in Richtung 0 strebt. Dieser Effekt tritt wegen der linearen Abhängigkeit des Gradienten der Kostenfunktion von der Ableitung der Aktivierungsfunktion auf, wenn die Ableitung der Aktivierungsfunktion gegen 0 strebt. Folglich werden die betroffenen Netzparameter nicht weiter angepasst und das Training der betroffenen Neuronen oder sogar des gesamten Netzes stagniert.
Das genaue Gegenteil der verschwindenden Gradienten sind die explodierenden Gradienten (engl. *exploding gradients*), bei denen der Gradient der Kostenfunktion sehr groß wird und die Netzparameter folglich sehr stark angepasst werden. Durch die starke Anpassung werden die Gradienten im darauf folgenden Trainingsschritt ebenfalls sehr groß sein und wieder zu einer starken Anpassung führen. Dieser Vorgang wiederholt sich und führt schließlich zu einem instabilen Training.
Beide Probleme sind besonders gravierend, wenn sie in den tiefen Schichten, also Schichten nahe der Ausgangsschicht, entstehen, da die instabilen Gradienten dann durch die rekursive Abhängigkeit bis zur Eingangsschicht rückwärts propagiert werden. Um diese Probleme zu verhindern, muss eine geeignete Aktivierungsfunktion gewählt werden.[51]
In der Forschung haben sich einige Funktionen als sinnvoll erwiesen, die heutzutage in einem Großteil der Anwendungen neuronaler Netze verwendet werden. Grundvoraussetzung für die Realisierbarkeit des gradientenbasierten Trainings ist dabei die stetige Differenzierbarkeit der Aktivierungsfunktion.

$$f_{\text{sigmoid}}(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.45}$$

$$f_{\tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.46}$$

$$f_{\text{ReLU}}(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{2.47}$$

Die Sigmoid-Funktion (2.45) wurde lange Zeit als Aktivierungsfunktion für Regressionsaufgaben verwendet und bildet einen reellen Nettoeingang auf die Aktivierung im Bereich (0, 1) ab. Durch die Begrenzung des Bereichs und die maximale Ableitung an der Stelle 0 mit dem Wert $0.25 < 1$ sind explodierende Gradienten ausgeschlossen. Die Ableitung verläuft jedoch asymptotisch gegen 0 für Nettoeingänge steigender Beträge, sodass verschwindende Gradienten nicht ausgeschlossen sind. Hinzu kommt, dass der Gradient durch die rekursive Abhängigkeit und dadurch mehrfache Multiplikation mit Werten < 0.25 stark fällt, je näher er der Eingangsschicht ist, sodass die Gefahr verschwindender Gradienten insbesondere bei vielschichtigen neuronalen Netzen sehr hoch ist.[52]
Für vielschichtige MLPs hat sich aufgrund der aufgezeigten Schwächen der Sigmoid-Funktion zunächst die Tangens Hyperbolicus (tanh)-Funktion (2.46) als geeigneter herausgestellt und die Sigmoid-Funktion weitestgehend ersetzt.[53, 54] Verschwindende Gradienten sind zwar analog zur Sigmoid-Funktion auch durch den Einsatz der tanh-Funktion nicht ausgeschlossen, jedoch kann ihre Ableitung einen maximalen Wert von 1 annehmen, wodurch die Gefahr des Auftretens und das Ausmaß verschwindender Gradienten bei vielschichtigen MLPs deutlich geringer ist.

Die derzeit gängigsten nicht-linearen Aktivierungsfunktionen für Regressionsaufgaben sind die *Rectified Linear Unit* (ReLU)-Funktion (2.47) und verschiedene Erweiterungen ebendieser.[52] Sie setzt sich aus zwei linearen Abschnitten zusammen, sodass sie insgesamt als nicht-linear gilt und die Berechnung der Ableitung trivial ist. Dadurch wird das Training verglichen mit der tanh-Funktion beschleunigt, ohne dabei die Vorhersagegenauigkeit einzuschränken.[55] Durch die lineare Abbildung werden positive Nettoeinträge auf Aktivierungen im Bereich $(0, \infty)$ abgebildet, sodass verschwindende Gradienten für diesen Bereich ausgeschlossen sind. Hierdurch können jedoch beim Vorhersageprozess des Trainings sehr hohe Werte erzielt werden, wenn die Eingangsvektoren große positive Werte aufweisen, was zu Instabilitäten des Trainings führen kann. Um dies zu vermeiden, werden die Eingangsgrößen in der Regel auf einen Bereich von [-1, 1] oder [0, 1] skaliert. Durch die Skalierung wird das Training generell robuster und schneller, sodass die Skalierung ein gängiges Prozedere der Datenaufbereitung darstellt. [56, 57] Für negative Nettoeinträgen ist die Ableitung jedoch immer 0, sodass verschwindende Gradienten für diesen Bereich entstehen können. Dieses Problem kann durch verschiedene Modifikationen der ReLU-Funktion behoben werden. Trottier et al. [58] verfolgen dabei beispielsweise den Ansatz, die Funktion im negativen Bereich durch eine Exponentialfunktion zu beschreiben, sodass die Ableitung im negativen Bereich Werte ungleich 0 aufweist. Ein weiteres Problem der ReLU besteht in der undefinierten Ableitung an der Stelle 0, das jedoch einfach durch die manuelle Definition der Ableitung an dieser Stelle zu 0 oder 1 behoben werden kann.[55]

Insbesondere für die Ausgangsschicht ist die Wahl der Aktivierungsfunktion von entscheidender Bedeutung, da die Aktivierungen dieser Schicht den Ausgangsvektor bilden. Für sinnvolle Vorhersagen ist es daher essentiell, dass die zu erwartenden Ausgangsgrößen im Wertebereich der Aktivierungsfunktion liegen. Ein Beispiel: Für die Vorhersage einer Druckbeiwertsverteilung, die Werte im Bereich [-3, 1] annimmt, eignet sich keine der obigen Funktionen aufgrund ihres begrenzten Wertebereichs als Aktivierungsfunktion in der Ausgangsschicht.
Eine Abhilfe besteht darin, die Ausgangsvektoren auf den Wertebereich der verwendeten Aktivierungsfunktion zu skalieren, die skalierten Vorhersagen zu berechnen und diese anschließend wieder

zurückzuskalieren. Eine deutlich einfachere Abhilfe ist die Wahl der Identitätsfunktion als Aktivierungsfunktion in der Ausgangsschicht, bei der die Aktivierung eines Neurons gleich seines Nettoeingangs ist und somit jeden Wert annehmen kann.[59]

2.1.5 Generalisierungsfähigkeit

Durch das Training eines neuronalen Netzes werden die Netzparameter so angepasst, dass das neuronale Netz den Zusammenhang der systembeschreibenden Eingangs- und Ausgangsdaten möglichst genau abbildet. Anschließend werden mit dem trainierten Netz Vorhersagen für neue Eingangsdaten berechnet, die dem neuronalen Netz während des Trainings nicht vorgelegt wurden. Die Fähigkeit, nicht nur für die Trainingsdaten, sondern auch für neue Daten präzise Vorhersagen berechnen zu können, wird als Generalisierungsfähigkeit bezeichnet. Das Erzeugen eines generalisierenden neuronalen Netzes stellt daher im Kontext der Ersatzmodellierung das Ziel des Trainings dar.[39]

Die Überanpassung (engl. *Overfitting*) der Netzparameter während des Trainings ist eine der häufigsten Ursachen dafür, dass ein neuronales Netz nicht generalisiert. Hierbei wird das neuronale Netz so lange trainiert, dass es den Zusammenhang der Trainingsdaten bis ins kleinste Detail äußerst genau abbilden kann. Diese kleinen, erlernten Details spiegeln dabei jedoch häufig nur spezielle Details der Trainingsdaten wider, die nicht für die Gesamtheit aller systembeschreibenden Daten zutreffend sind. Dadurch sinkt die Generalisierungsfähigkeit des neuronalen Netzes und die Vorhersagen für neue Daten werden ungenau, sodass das eigentliche Ziel des Trainings verfehlt wird. [59] Eine notwendige Bedingung für das Auftreten der Überanpassung ist ein neuronales Netz mit einer zu großen Kapazität, da erst hierdurch das Erlernen der speziellen Details der Zusammenhänge der Trainingsdaten ermöglicht wird. Der häufigste Grund der Überanpassung ist dann das Training mit zu vielen Anpassungen der Netzparameter, was gleichbedeutend mit einer zu großen Anzahl an Trainingsepochen ist.[38] Eine Abhilfe in diesem Fall kann daher durch eine Verringerung der Anzahl an Schichten und Neuronen des neuronalen Netzes geschaffen werden, sodass die Überanpassung gar nicht erst ermöglicht wird. Dadurch besteht jedoch die Gefahr, dass ein Netz mit zu geringer Kapazität gewählt wird und nicht das vollständige Potential eines neuronalen Netzes ausgenutzt wird. In der Regel wird daher meist das Training mit zu vielen Epochen verhindert, indem ein sinnvoller frühzeitiger Abbruch (engl. *Early Stopping*) des Trainings durchgeführt wird.[60] Um diesen frühzeitigen Abbruch zu realisieren, ist eine epochenweise Überwachung der Generalisierungsfähigkeit des neuronalen Netzes während des Trainings nötig. Ein gängiges Prozedere dafür besteht darin, die zur Verfügung stehenden Trainingsdaten in zwei Datensätze - einen tatsächlich für das Training verwendeten Trainingssatz und einen Validierungssatz - aufzuteilen.[39] Der Validierungssatz wird dabei ausdrücklich nicht verwendet, um Anpassungen der Netzparameter durchzuführen, sondern dient lediglich der Überwachung des Lernprozesses. Für beide Datensätze wird dann die Genauigkeit anhand der Kostenfunktion nach jeder Trainingsepoche evaluiert. Die Darstellung der berechneten Kostenfunktion E für beide Datensätze über den Epochen Z ergibt eine sogenannte Lernkurve (engl. *learning curve*), die in Abbildung 2.5 qualitativ dargestellt ist. Solange das Training mit zunehmenden Epochen zu einer Genauigkeitssteigerung der Vorhersagen für beide Datensätze führt, steigt

dadurch auch die Generalisierungsfähigkeit des neuronalen Netzes. Falls eine Überanpassung des neuronalen Netzes stattfindet, sinkt mit zunehmender Anzahl an durchgeführten Trainingsepochen die Kostenfunktion für den Trainingssatz, während diese für den Validierungssatz steigt. Durch dieses Monitoring kann daher diejenige Epoche $Z_{Abbruch}$ identifiziert werden, bis zu der die Generalisierungsfähigkeit des neuronalen Netzes durch das Training steigt.[60]

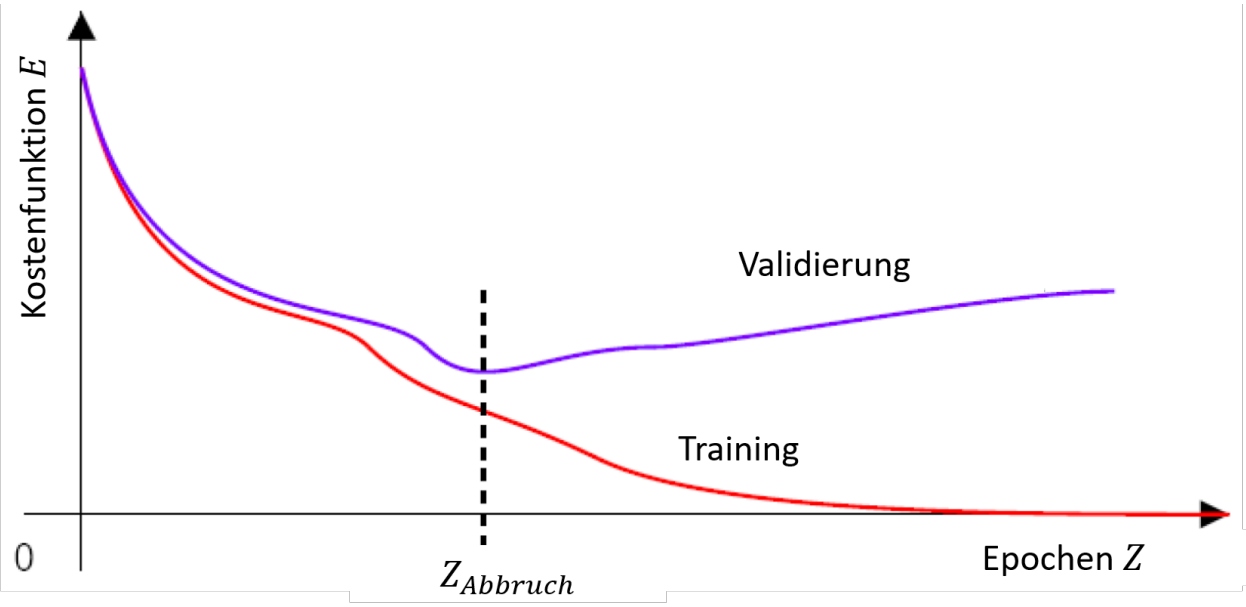


Abbildung 2.5: Lernkurve zur Überwachung der Generalisierungsfähigkeit des neuronalen Netzes während des Trainings [61], modifiziert

Der Verlauf der Kostenfunktion über den Trainingsepochen ist jedoch häufig schwankend und nicht, wie in der Abbildung 2.5 dargestellt, stetig fallend bis zum Erreichen des Optimums. Der frühzeitige Trainingsabbruch wird in der Praxis daher durch ein Abbruchkriterium realisiert: Eine Metrik zur Quantifizierung des Vorhersagefehlers bezogen auf den Validierungssatz muss nach einer definierten Anzahl an Trainingsepochen mindestens um einen definierten Wert gesunken sein. Das neuronale Netz mit der geringsten Metrik wird während des Trainings zwischengespeichert und ständig aktualisiert, sodass das Training bei Verletzen des Abbruchkriteriums gestoppt wird und das zuletzt zwischengespeicherte neuronale Netz mit der besten Generalisierungsfähigkeit vorliegt. Aufgrund der Einfachheit und Effektivität sollte ein frühzeitiger Trainingsabbruch entsprechend dieser Methode nahezu universell eingesetzt werden.[38]

Eine weitere Technik, um die Überanpassung zu reduzieren, ist die Verwendung der *Dropout*-Methode von Srivastava et al. [62], die in Abbildung 2.6 verdeutlicht wird. Sie beruht darauf, einen gewissen Prozentsatz der Neuronen einer Schicht während der Anpassungen der Netzparameter nicht zu berücksichtigen. Die unberücksichtigten Neuronen variieren dabei zufällig für jede Netzparameteranpassung während des Trainings. Srivastava et al. zeigten, dass die Verwendung dieser Methode die Gefahr des Überanpassens reduziert bei gleichzeitiger Steigerung der Trainingsperformance.

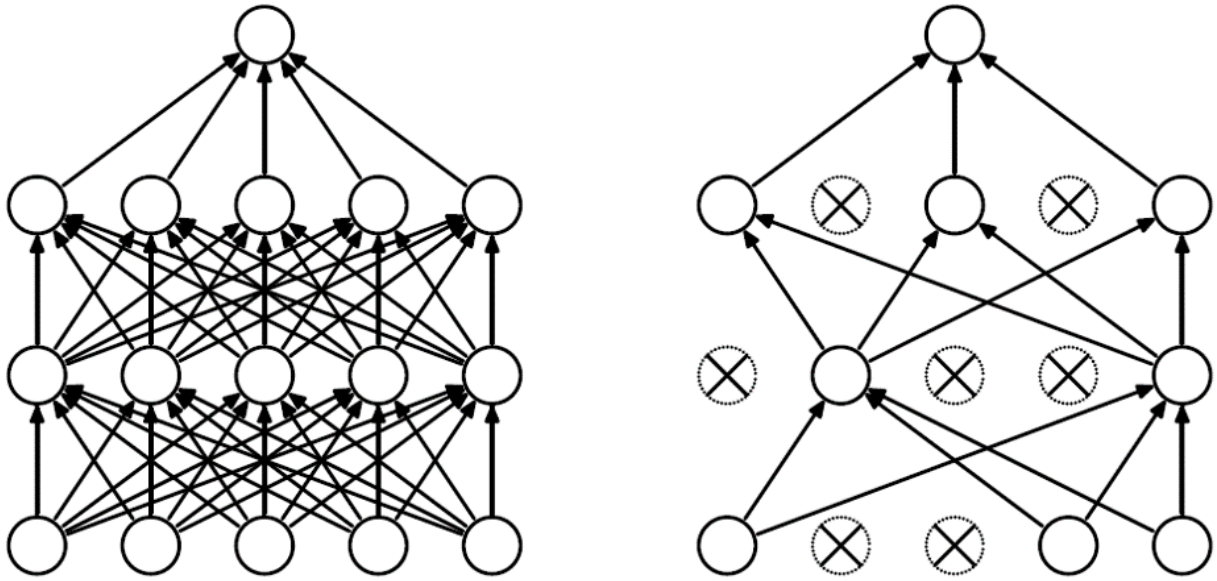


Abbildung 2.6: Vergleich eines MLPs mit zwei verdeckten Schichten (links) mit einem MLP unter Verwendung der Dropout-Methode (rechts) [62]

Das gegensätzliche Pendant zur Überanpassung ist die Unteranpassung (engl. *Underfitting*) der Netzparameter, die meist durch eine zu geringe Anzahl an Trainingsepochen und eine zu geringe Kapazität zustande kommt. Die Gründe sind dabei analog und entgegengesetzt zur Überanpassung. Dieses Problem tritt jedoch sehr selten auf und ist durch sehr hohe Werte der Kostenfunktion erkennbar. Es kann durch eine Erhöhung der Trainingsepochen und der Kapazität des neuronalen Netzes einfach behoben werden.[38]

Durch die Abhängigkeit der Generalisierungsfähigkeit von der Kapazität des neuronalen Netzes, muss letztere sorgfältig gewählt werden. Eine strukturierte Methode für die Bestimmung einer geeigneten Kapazität sowie weiterer Netzeinstellungen wird im folgenden Kapitel erläutert.

2.1.6 Entwicklungsprozess eines neuronalen Netzwerks

Die gesamte Entwicklung eines neuronalen Netzes als Ersatzmodell für ein bestimmtes Problem kann durch verschiedene Phasen beschrieben werden. Für diese Arbeit werden die Entwicklungsphasen eines neuronalen Netzes für die Ersatzmodellierung als modifizierte Version der Definition nach Füser [63] wie folgt definiert.

1. Definition des Problems
2. Datengenerierung
3. Datenaufbereitung
4. Wahl der Netzwerktopologie

5. Entwurfsprozess des neuronalen Netzes
6. Finales Training des neuronalen Netzes
7. Evaluierung des trainierten neuronalen Netzes

In Phase 1 wird dabei zunächst das Problem beschrieben. Für die Ersatzmodellierung eines strömungstechnischen Systems wird dieses insbesondere durch die Eigenschaften der zugrundeliegenden Strömung, das zu untersuchende Objekt und die Eingangs- und Zielgrößen des Systems definiert. Anschließend müssen die systembeschreibenden CFD-Daten in Phase 2 erzeugt werden und sowohl quantitativen als auch qualitativen Anforderungen entsprechen. Letztere werden durch die hochauflösenden numerischen Strömungssimulationen ab einer gewissen Konvergenz garantiert. Bezüglich der Quantität der Daten ist es das Ziel, die Datenpunkte im systembeschreibenden Parameterbereich so zu wählen, dass sie den gesamten Bereich möglichst genau und homogen beschreiben, damit dem neuronalen Netz im gesamten Parameterbereich eine gleiche Informationsdichte zugrunde liegt. Für diesen Zweck ist es sinnvoll, die Datenpunkte mit einer Methode der Versuchsplanung (engl. Design of Experiment) (DoE) zu bestimmen. In Phase 3 werden die erzeugten Daten aufbereitet, sodass der zu modellierende systembeschreibende Zusammenhang von Eingangs- und Ausgangsdaten deutlich wird. Hierfür werden daher die Eingangs- und Ausgangsgrößen klar definiert und die Daten den Definitionen entsprechend aufbereitet, dass sie für das Training des neuronalen Netzes verwendet werden können. Außerdem werden die Daten aus dem in Kapitel 2.1.4 beschriebenen Grund auf einen bestimmten Bereich skaliert. Häufig verwendete Bereiche sind [-1, 1] oder [0, 1].[64] Anschließend werden alle zur Verfügung stehenden Daten in zwei disjunkte Datensätze aufgeteilt, die Trainingsdaten und die Testdaten. Die Testdaten werden ausschließlich in der letzten Entwicklungsphase verwendet, um das trainierte neuronale Netz zu evaluieren. In welchem Verhältnis die Datensätze aufgeteilt werden hängt dabei stark von der zur Verfügung stehenden Datenmenge ab, sodass die Aufteilung meist auf subjektiver Einschätzung des Anwenders vorgenommen wird. Werden zu viele Daten als Testdaten definiert, könnten dem neuronalen Netz wichtige Informationen während des Trainings fehlen, das zu einer schlechten Generalisierung führt. Bei zu wenigen Testdaten besteht Gefahr, dass die finale Evaluierung des neuronalen Netzes nicht aussagekräftig ist. Gängige Aufteilungsverhältnisse von Trainingsdaten zu Testdaten sind 70/30 oder 80/20.[38, 39] Abhängig vom vorliegenden Problem und der Beschaffenheit der Daten wird in Phase 4 eine geeignete Netzwerktopologie ausgewählt. Phase 5 stellt den Entwurfsprozess eines neuronalen Netzes dar, in welchem Hyperparameter so gewählt werden, dass eine gute Generalisierungsfähigkeit mit möglichst genauen Vorhersagen bei geringem Zeitaufwand ermöglicht wird. Als Hyperparameter werden Parameter bezeichnet, die vor dem Training des neuronalen Netzes festgelegt werden müssen und in der Regel nicht während des Trainings optimiert werden, d.h. sämtliche Parameter mit Ausnahme der Gewichte und Biases.[59] Die Lernrate, die Anzahl an Schichten und Neuronen, die Aktivierungsfunktionen und die Batchgröße sind dabei nur wenige Beispiele. Diese Phase stellt wegen der enormen Anzahl an verschiedenen Hyperparametern eine große Herausforderung dar. Die möglichen Kombinationen der Hyperparameter steigen zudem exponentiell mit der Anzahl an betrachteten Hyperparametern an und die möglichen Bereiche der verschiedenen Hyperparameter sind

meist nicht limitiert. Außerdem haben die Einstellungen verschiedener Hyperparameter eine Wechselwirkung auf die Performance des neuronalen Netzes. Aus diesen Gründen werden verschiedene Methoden eines systematischen Entwurfsprozess zur Bestimmung geeigneter Hyperparameter im Abschnitt 2.1.7 detailliert erläutert. In Phase 6 wird das neuronale Netz mit einer hohen Anzahl an Trainingsepochen trainiert, welches in Phase 5 anhand der besten gefundenen Hyperparameter entworfen wurde. Abschließend werden in Phase 7 die Vorhersageergebnisse des trainierten neuronalen Netzes für die Testdaten anhand einer beliebigen Metrik, etwa dem MSE (2.15) oder dem MAE (2.16), evaluiert. Diese Evaluierung stellt gewissermaßen die Generalprobe des neuronalen Netzes dar.[63]

2.1.7 Systematischer Entwurfsprozess des neuronalen Netzes

In diesem Abschnitt werden Methoden für einen systematischen Entwurf des neuronalen Netzes, welcher grundsätzlich auf der Bestimmung geeigneter Hyperparameter basiert, beschrieben. Dafür wird ein neuronales Netz mit einer Hyperparameterkombination erstellt und mit den Daten des Trainingssatzes trainiert. Anschließend wird die Vorhersagegenauigkeit für die Daten des Validierungssatzes durch eine beliebige Metrik evaluiert, um die Eignung der gewählten Hyperparameter zu bewerten. Dieser Vorgang wird für verschiedene Kombinationen an Hyperparametern wiederholt. Verschiedene Art und Weisen, wie die jeweiligen Hyperparameter ausgewählt werden, sind im Folgenden beschrieben.
Es sei an dieser Stelle betont, dass die Testdaten im Entwurfsprozess des neuronalen Netzes nicht verwendet werden. Würden die während der Suche getesteten Hyperparameterkombinationen anhand der Evaluierung der Vorhersagen für die Testdaten bewertet und angepasst, wäre das resultierende neuronale Netz in der finalen Evaluierung anhand der Testdaten bevorteilt und die Bewertung der Generalisierungsfähigkeit wäre verzerrt. Die Methoden zur Bestimmung geeigneter Hyperparameter lassen sich prinzipiell in zwei Kategorien unterteilen, die manuelle und die automatisierte Suche.[38]

2.1.7.1 Manuelle Suche

Bei der manuellen Suche werden neuronale Netze mit händisch eingestellten Hyperparametern erzeugt. Die Eignung der gewählten Hyperparameterkombination wird anschließend wie oben beschrieben bewertet. In einem iterativen Prozess werden die Hyperparameter immer wieder angepasst und die Ergebnisse verglichen, solange bis eine gewünschte Vorhersagegenauigkeit erreicht wird.
Neuronale Netze profitieren meistens von einer Berücksichtigung von vierzig oder mehr Hyperparametern, können jedoch oft schon mit einer kleinen Anzahl an untersuchten Hyperparametern gute Ergebnisse erzielen. In der manuellen Suche werden daher meist nur die wichtigsten Hyperparameter untersucht.[38] Laut Bengio [59] ist der wichtigste Hyperparameter die anfängliche Lernrate, die wie in Kapitel 2.1.3 beschrieben bestimmt, wie stark die Netzparameter durch eine Anpassung geändert werden. Die Lernrate wird häufig während des Trainings mit zunehmender Anzahl an durchgeführten Trainingsepochen automatisiert verringert, um ein anfänglich schnelles, grobes

Lernen der Zusammenhänge zu ermöglichen und im weiteren Verlauf des Trainings durch feinere Netzparameteranpassungen die Genauigkeit zu erhöhen. Die Untersuchungen verschiedener Methoden der Lernratenverringerung von Senior et al. [65] zeigen, dass in ihrem Anwendungsfall eine exponentielle Abnahme der Lernrate zu den Ergebnissen mit der höchsten Vorhersagegenauigkeit führt.
Weitere wichtige Hyperparameter sind die Anzahl an verdeckten Schichten und die Anzahl enthaltener Neuronen, die zusammengenommen die Kapazität des neuronalen Netzes definieren. Eine vergleichende Studie [66] hat gezeigt, dass eine schichtenübergreifende konstante Zahl an Neuronen zu besseren Ergebnissen führt, als eine zu- oder abnehmende Neuronenanzahl. Außerdem sollte die Anzahl an Neuronen in der ersten Schicht größer sein als die Anzahl an Eingangsgrößen, um gute Ergebnisse zu garantieren. Die Wirkungen der verschiedenen Hyperparameter sind jedoch problemspezifisch, sodass dies keine allgemein gültigen Ergebnisse sind.
Ein empfohlenes Vorgehen bei der manuellen Suche ist es, immer nur einen Hyperparameter zu ändern und bei der Untersuchung weiterer Hyperparameter eine Änderung ausgehend von der besten bisher gefundenen Hyperparameterkombination vorzunehmen.[59]

Zusammenfassend lässt sich sagen, dass eine effiziente manuelle Suche ein gewisses Know-How des Anwenders erfordert, das stark auf dessen bisherigen Erfahrung beruht. Um einen Hyperparameter manuell effektiv einzustellen, müssen dem Anwender beispielsweise die Wechselwirkungen zwischen verschiedenen Hyperparametern und der Einfluss der einzelnen Hyperparameter auf die Vorhersagegenauigkeit bekannt sein. Die manuelle Wahl der Hyperparameter ist daher sehr subjektiv, sodass eine automatisierte Suche geeigneter Hyperparameter meist effektiver ist.

2.1.7.2 Automatisierte Suche

Die automatisierte Suche der besten Netzwerkkonfiguration kann mit verschiedenen Methoden durchgeführt werden, welche sich wiederum in zwei Gruppen unterteilen lassen - die uninformierte und die informierte Suche.[38]

Uninformierte Suche

Bei der uninformierten Suche werden einige Hyperparameterkombinationen automatisch getestet und bewertet. Die Informationen der Auswertung einer Hyperparameterkombination beeinflussen dabei nicht die Wahl der als nächstes getesteten Kombination. Eine verbreitete Methode ist die Rastersuche (engl. *Grid Search*), bei der jeder Hyperparameter auf einen definierten Bereich begrenzt wird und eine Anzahl an Einstellungswerten für jeden Hyperparameter in dem jeweiligen begrenzten Bereich definiert wird. Die Bereiche der Hyperparameter spannen somit einen multidimensionalen Hyperparameterraum $\mathbb{R}^d$ auf, in dem die verschiedenen Hyperparameterkombinationen enthalten sind und d die Anzahl an betrachteten Hyperparametern beschreibt. Durch die Rastersuche wird dann jede mögliche Kombination getestet und bewertet. Die Hyperparameterkombinationen sind links in Abbildung 2.7 für zwei Hyperparameter dargestellt. In der Theorie würde bei unendlich

kleinen Abständen der definierten Hyperparameter definitiv die optimale Hyperparameterkombination im aufgespannten Hyperparameterraum gefunden werden.[59] Allerdings leidet die Rastersuche am sogenannten Fluch der Dimensionalität [67], der in diesem Kontext die exponentielle Zunahme der Anzahl zu testender Hyperparameterkombinationen bei Erhöhung der Dimension beschreibt, die durch die Berücksichtigung einer zunehmenden Anzahl an Hyperparametern entsteht.

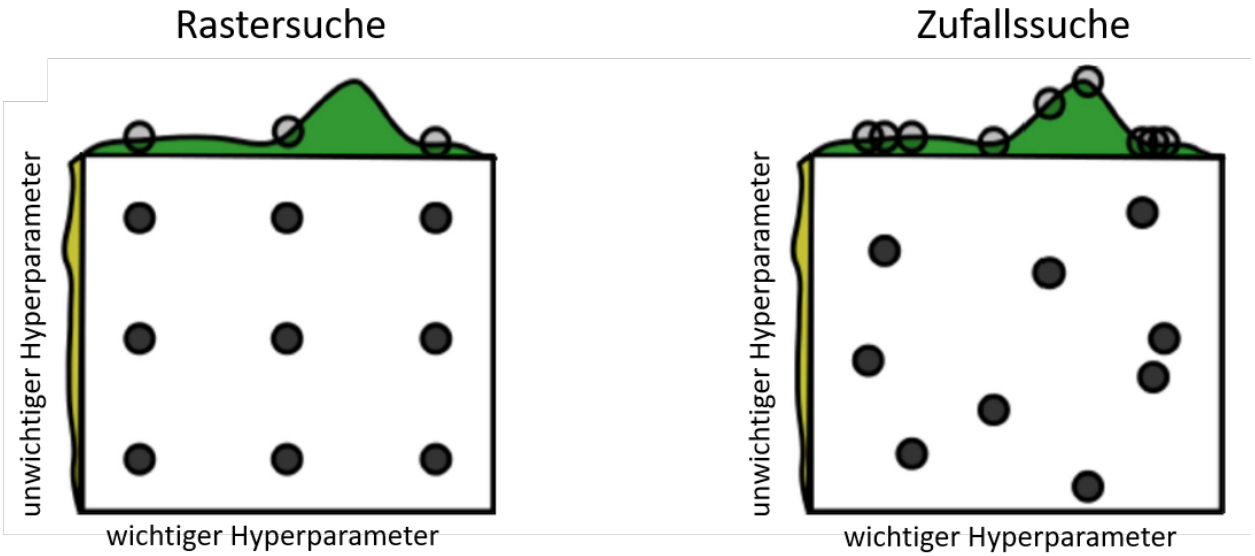


Abbildung 2.7: Vergleich der Rastersuche und der Zufallssuche zur Bestimmung geeigneter Hyperparameter [68], modifiziert

Hinzu kommt, dass einige wichtige Hyperparameter einen größeren Einfluss auf die Vorhersageergebnisse der neuronalen Netze haben als andere Hyperparameter.[38] In der Rastersuche werden daher einige nicht zielführende Kombinationen getestet, bei denen alle wichtigen Hyperparameter, die für tatsächliche Änderung in den Vorhersageergebnissen sorgen, gleich bleiben und nur die unwichtigen geändert werden, die keinen Einfluss auf die Ergebnisqualität der Vorhersage haben. Links in Abbildung 2.7 ist dieses Problem aufgezeigt, wobei auf der vertikalen Achse der Bereich eines unwichtigen und auf der horizontalen Achse der Bereich eines wichtigen Hyperparameter darstellt ist. Ihre Einflüsse auf die Vorhersageergebnisse des neuronalen Netzes sind jeweils durch den gezeigten Verlauf dargestellt.

Eine alternative von Bergsta und Bengio [68] vorgestellte Methode ist die sogenannte Zufallssuche (engl. *Random Search*). Diese führt für gleiche Bereiche der Hyperparameter zu genaueren oder besseren Ergebnissen bei einer geringeren Anzahl an getesteten Hyperparameterkombinationen und dadurch deutlich geringerem Zeitaufwand. Das Prinzip ist analog zur Rastersuche ebenfalls in Abbildung 2.7 rechts gezeigt. Die Überlegenheit der Zufallssuche wird besonders deutlich bei der Berücksichtigung von vielen unwichtigen Hyperparametern. Da die unterschiedliche Wichtigkeit problemspezifisch ist, kann vor der Suche jedoch kein Hyperparameter als unwichtig identifiziert werden. In den meisten Fällen wird die Rastersuche daher bei drei oder weniger betrachteten Hyperparametern verwendet und die Zufallssuche entsprechend bei mehr als drei Hyperparametern.

Aufgrund der gegenseitigen Unabhängigkeit der zu testenden Hyperparameterkombinationen weisen beide uninformierten Suchen eine hohe Parallelisierbarkeit auf. Dadurch besteht die Möglichkeit, die verschiedenen Hyperparameterkombinationen simultan auf geeigneter Computerhardware wie beispielsweise einem Cluster zu testen, um den zeitlichen Aufwand zu reduzieren.

Informierte Suche

Der Ansatz der informierten Suche besteht darin, die Evaluierungsergebnisse der bereits getesteten Hyperparameterkombinationen als Informationsgrundlage für die Wahl einer nächsten, möglichst vielversprechenden Kombination an Hyperparametern zu nutzen.[38]
Hierfür wird die Hyperparametersuche als Optimierungsproblem angesehen und eine Zielfunktion definiert, die es zu optimieren gilt. Die Zielfunktion stellt dabei eine Metrik als direktes Maß für die Vorhersagegenauigkeit des neuronalen Netzes für den Validierungssatz, abhängig von den gewählten Hyperparametern dar. Hierbei fehlen jedoch Informationen über die Gradienten der Metrik bezüglich der Hyperparameter, beispielsweise bedingt durch diskrete kategorische Hyperparameter wie die Aktivierungsfunktion, sodass eine gradientenbasierte Optimierung nicht möglich ist und alternative Optimierungsansätze verwendet werden. Für einen detaillierten Überblick einiger alternativen Ansätze sei auf die Arbeit von Claesen [69] verwiesen. Eine weit verbreitete Methode der Hyperparameteroptimierung ist die Bayesian Optimierung mit Gauß'schem Prozess, die ein globales Optimum X^+ durch Maximieren der Zielfunktion f abhängig von den Hyperparameterkombinationen x nach (2.48) sucht. Dabei ist d die Anzahl der berücksichtigten Hyperparameter.

$$X^+ = \underset{x \in \mathbb{R}^d}{\operatorname{argmax}} f(x) \tag{2.48}$$

Die Bayesian Optimierung ist ein iterativer Prozess. Während einer Iteration, die in Abbildung 2.8 verdeutlicht ist, wird die zu optimierende Zielfunktion zunächst möglichst genau durch eine Ersatzfunktion approximiert. In Abbildung 2.8 ist der Verlauf der in der Realität unbekannten Zielfunktion für demonstrative Zwecke dargestellt.

Zur Approximation wird ein Gauß'scher Prozess verwendet, der aufbauend auf der priori Verteilung zusammen mit den Beobachtungen der bereits getesteten Hyperparameterkombinationen eine posteriori Verteilung der möglichen Ersatzfunktionen berechnet. Diese werden auf ein 95% Konfidenzintervall, also zwei Standardabweichungen, für die weitere Betrachtung eingegrenzt. Für eine ausführliche Erklärung des Gauß'schen Prozesses sei auf [71] und [47] verwiesen. Je breiter das Konfidenzintervall - also je größer die Standardabweichung - für eine bestimmte Hyperparameterkombination ist, desto größer ist die Unsicherheit des Werts der Zielfunktion für diese Kombination. Deshalb ist es für eine Steigerung der Approximationsgenauigkeit der Ersatzfunktion sinnvoll, diese Kombination als nächstes zu untersuchen. Das eigentliche Ziel der Optimierung ist jedoch das Finden des Maximums der Zielfunktion und nicht das Finden einer nahezu perfekt approximierten Ersatzfunktion. Aus diesem Grund wird eine Akquisitionfunktion EI (engl. *Expected Improvement*) nach [72] erstellt, die in (2.49) definiert ist und eine Balance zwischen diesen beiden Zielen schafft.

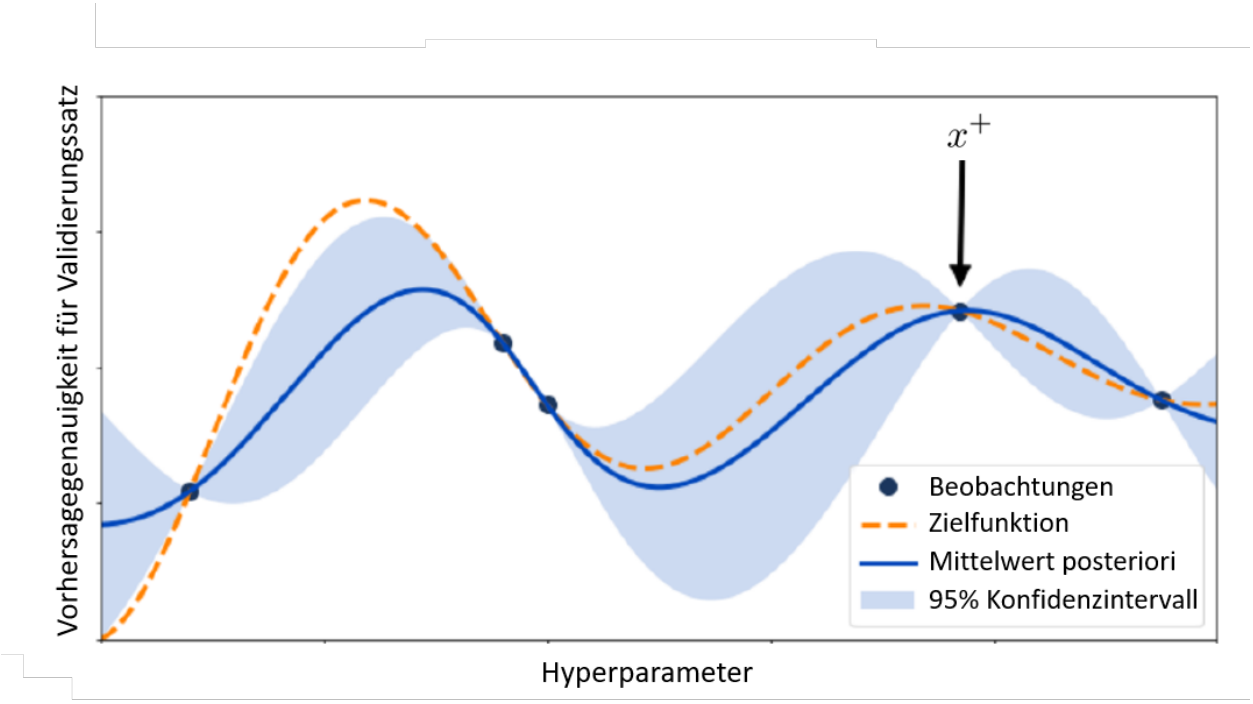


Abbildung 2.8: Veranschaulichung eines Iterationsprozesses der Bayesian-Optimierung nach fünf bereits getesteten Hyperparameterkombinationen [70], modifiziert

Das Maximum der Akquisitionsfunktion liegt an der Stelle der als nächstes zu testenden Hyperparameterkombination x_t nach (2.50).
Hierbei ist zu erwähnen, dass die Ansätze einer geeigneten Akquisitionsfunktion vielfältig sind, die EI-Funktion jedoch die weit verbreitetste darstellt.[56] $D_{1:t-1}$ stellt dabei die $t-1$ bereits getesteten Kombinationen dar und $f(x^+)$ ist der derzeit höchste Zielfunktionswert bei der Hyperparameterkombination x^+.

$$EI(x) = \mathbb{E}max(f(x) - f(x^+), 0) \tag{2.49}$$

$$x_t = \underset{x}{\operatorname{argmax}}\, EI(x|D_{1:t-1}) \tag{2.50}$$

$$EI(x) = \begin{cases} (\mu(x) - f(x^+) - \xi)\Phi(Z) + \sigma(x)\phi(Z), & \sigma(x) > 0 \\ 0, & \sigma(x) = 0 \end{cases} \tag{2.51}$$

$$Z = \begin{cases} \frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}, & \sigma(x) > 0 \\ 0, & \sigma(x) = 0 \end{cases} \tag{2.52}$$

Die Akquisitionsfunktion lässt sich durch den Gauß'schen Prozess auch analytisch als (2.51)-(2.52) schreiben, wobei $\mu(x)$ den Mittelwert und $\sigma(x)$ die Standardabweichung der posteriori Verteilung für eine Hyperparameterkombination x darstellt. ϕ ist dabei die Dichtefunktion der Wahrscheinlichkeiten (engl. *probability density function*) und Φ die kumulierte Dichtefunktion (engl. *cummulative density function*) der Standardnormalverteilung. Der erste Summand der Akquisitionsfunktion in

(2.51) wird auch exploitativer Term genannt und beschreibt den Einfluss, als nächstes eine Hyperparameterkombination zu testen, bei der ein möglichst hoher Zielfunktionswert erreicht wird. Dagegen bestimmt der zweite Summand - der sogenannte explorative Term - den Einfluss, als nächstes eine Kombination mit dem Ziel, eine möglichst hohe Steigerung der Approximationsgenauigkeit der Ersatzfunktion, zu untersuchen. Je höher ξ gewählt wird, desto größer ist der explorative Anteil während der Optimierung. Analog wird der exploitative Anteil größer, je kleiner ξ gewählt wird, sodass ξ einen Parameter darstellt, der die beiden unterschiedlichen Ziele gewissermaßen balanciert.[73]
Ist die nächste zu testende Hyperparameterkombination x_t bestimmt, wird der zugehörige Zielfunktionswert berechnet. Anschließend wird wieder eine neue posteriori Verteilung unter Berücksichtigung der neuen Beobachtung berechnet. Dieser Prozess wird iteriert, sodass die Konfidenzintervalle der posteriori Verteilung sukzessive kleiner und die Zielfunktion dadurch immer besser durch die Ersatzfunktion beschrieben wird. Daher wird sich auch der maximale Zielfunktionswert immer weiter dem globalen Maximum annähern, bis eine zuvor definierte Anzahl an Optimierungsiterationen erreicht und die Bayesian Optimierung abgeschlossen ist.[71] Das Ergebnis ist dann die Hyperparameterkombination, mit der das neuronale Netz während der Optimierung die höchste Vorhersagegenauigkeit für die Validierungsdaten erreicht hat.
Die Bayesian Optimierung ist insbesondere durch die Vernachlässigung von Untersuchungen in nicht vielversprechenden Hyperparameterbereichen meist effizienter als die Raster- und die Zufallssuche.[69] Durch die Abhängigkeit einer Iteration von den vorherigen Iterationen ist sie jedoch prinzipbedingt nicht parallelisierbar. Es existieren jedoch Erweiterung der Bayesian Optimierung, die eine Parallelisierung ermöglichen. Eine dieser Erweiterungen wird beispielsweise von Falkner et al. in [74] vorgestellt.

Es ist zu betonen, dass der Entwurfsprozess der neuronalen Netze generell durch das ausprobieren einiger Hyperparameterkombinationen meist sehr zeitaufwändig ist, da für jede Kombination ein neuronales Netz trainiert wird. Es eignet sich daher, den Entwurfsprozess mit einer geringen Anzahl an Trainingsepochen durchzuführen und anschließend ein neuronales Netz mit den gefundenen Hyperparametern mit einer höheren Anzahl an Epochen zu trainieren. Außerdem sollte eine Überanpassung unbedingt vermieden werden, etwa durch die Verwendung des frühzeitigen Abbruchkriteriums des Trainings, da die Optimierung ansonsten auf falschen Beobachtungen aufbaut und zu verzerrten Ergebnissen führt.[59]

2.2 Klassische Ersatzmodellierung

Das Ziel der datengetriebenen Ersatzmodellierung ist es, die komplexen Zusammenhänge zwischen Eingangs- und Ausgangsdaten eines System möglichst genau und mit geringem Zeitaufwand abzubilden. Zu diesem Zweck werden die systembeschreibenden Daten in der klassischen Ersatzmodellierung meist einer Dimensionsreduzierung unterzogen, um den zeitlichen Aufwand der anschließenden Approximation zu reduzieren. Diese beruht häufig auf verschiedenen Arten der Interpolation oder Regression. Es wird eine mathematische Funktion erzeugt, deren Parameter so optimiert werden, dass die Beziehung zwischen Eingangs- und Ausgangsdaten des Systems möglichst genau approximiert wird.[75] In diesem Kapitel werden die POD-basierte Dimensionsreduzierung und die TPS-basierte Interpolation als gängige und häufig zusammen verwendete Verfahren der klassischen Ersatzmodellierung erläutert.

2.2.1 POD-basierte Dimensionsreduzierung

Die Dimensionsreduzierung von systembeschreibenden hochdimensionalen Datensätzen zielt darauf ab, diese Datensätze auf einen niedrigdimensionalen Unterraum zu projizieren. Dadurch können die Daten im Unterraum mit einer geringeren Anzahl an Variablen beschrieben werden. Daraufhin können die Zusammenhänge des im niedrigdimensionalen Unterraum dargestellte Systems beispielsweise durch Interpolationsansätze approximiert werden. Hierdurch entsteht eine enorme Steigerung der Effizienz der Ersatzmodellierung, da der zeitliche Aufwand mit zunehmender Dimensionsanzahl exponentiell steigt.[76]

Die POD-Dimensionsreduzierung ist weit verbreitet im Bereich der Ersatzmodellierung strömungstechnischer Probleme.[75] Bei der POD-Methode wird ein Datensatz in Form einer hochdimensionalen Matrix in orthogonale Hauptkomponenten zerlegt. Durch eine Singulärwertzerlegung der Matrix werden die benötigten Basisvektoren berechnet, mit denen eine niedrigdimensionale Darstellung des hochdimensionalen Raumes ermöglicht wird.[77] Ein Verfahren zur effizienten Bestimmung der Basisvektoren bei großen Datenmengen ist die von Sirovich [27] entwickelte Methode der Snapshots (engl. *method of snapshots*), die weitgehend in der auf CFD-Daten basierenden Ersatzmodellierung benutzt wird. Ein Snapshot stellt dabei eine CFD-Lösung des Strömungszustands dar, der durch aerodynamische Daten beschrieben wird und sich aus einer bestimmten Konfiguration an aerodynamischen Parametern, wie beispielsweise den Anströmbedingungen, ergibt.

Die sogenannte 'Snapshotmatrix' $\mathbf{S} = (\vec{S}_1, ..., \vec{S}_m) \in \mathbb{R}^{n \times m}$ besteht aus m Snapshots, die jeweils einen n-elementigen Vektor mit den aerodynamischen Daten enthalten - beispielsweise für die Darstellung einer Druckverteilung.

Die Snapshotmatrix $\mathbf{S}$ lässt sich in zwei orthogonale Matrizen und eine diagonale Matrix zerlegen, sodass (2.53) erfüllt wird.

$$\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.53}$$

Hierbei ist $\mathbf{U} \in \mathbb{R}^{n\times n}$ orthogonal und die Spalten von $\mathbf{U}$ sind die Eigenvektoren von $\mathbf{S}\mathbf{S}^T$. Weiterhin ist $\mathbf{V} \in \mathbb{R}^{m\times m}$ orthogonal und die Spalten von $\mathbf{V}$ sind die Eigenvektoren von $\mathbf{S}^T\mathbf{S}$. Die diagonale Matrix $\mathbf{\Sigma} = diag(\sigma_1, ..., \sigma_\mathrm{p}, 0, ..., 0) \in \mathbb{R}^{n\times m}$ enthält die nach ihrer Größe geordneten Singulärwerte $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_\mathrm{p} \geq 0$ auf der Hauptdiagonalen, wobei $\mathrm{p} = min(n, m)$ der Rang der Snapshotmatrix $\mathbf{S}$ ist.[75]

Die Singulärwerte stellen den Skalierungsfaktor der Daten auf die jeweiligen Hauptkomponenten dar. Je kleiner dieser Faktor ist, desto geringer ist der Einfluss auf die Abbildung der Daten auf den dimensionsreduzierten Unterraum. Diese Tatsache wird zur Reduzierung der Dimension ausgenutzt, indem lediglich die ersten, und damit größten k Singulärwerte der Hauptdiagonalen verwendet werden und alle übrigen Einträge auf der Hauptdiagonalen zu 0 gesetzt werden. Hierdurch kann die Darstellung der Snapshotmatrix auf k Dimensionen reduziert werden.[78]

Die Darstellung eines beliebigen Snapshots $\vec{S}_\mathrm{i}$ ist direkt durch die zugehörige Spalte i der Matrix $\mathbf{V}^T$ nach (2.54) bzw. vereinfacht als Summe dargestellt nach (2.55) definiert.

$$\vec{S}_\mathrm{i} = \mathbf{U}\mathbf{\Sigma}(\mathbf{V}^T)_\mathrm{i} \tag{2.54}$$

$$\vec{S}_\mathrm{i} = \sum_{\mathrm{j}=1}^{k} (\sigma_\mathrm{j}\mathbf{V}_\mathrm{j}^\mathrm{i})\vec{U}_\mathrm{j} \tag{2.55}$$

$$a_\mathrm{j}^\mathrm{i} = (\sigma_\mathrm{j}\mathbf{V}_\mathrm{j}^\mathrm{i}) \tag{2.56}$$

$$\vec{a}^\mathrm{i} = (a_1^\mathrm{i}, a_2^\mathrm{i}, ..., a_\mathrm{k}^\mathrm{i}) \tag{2.57}$$

Hierbei stellt $\{\vec{U}_1, ..., \vec{U}_k\}$ die orthogonale Basis des k-dimensionalen Unterraums dar, wobei die enthaltenen Basisvektoren POD-Moden genannt werden. Der POD-Koeffizient a_j^i wird nach (2.56) und der Koeffizientenvektor nach (2.57) definiert. Der Snapshot $\vec{S}_\mathrm{i}$ hängt somit direkt von dem Koeffizientenvektor ab. Durch die Wahl von k, also der Anzahl an POD-Moden bzw. der Dimension des Koeffizientenvektors, kann eine Balance zwischen Genauigkeitsverlust und zeitlicher Reduktion des Rechenaufwands erreicht werden. Der Genauigkeitsverlust, der durch vernachlässigte Singulärwerte ungleich 0 entsteht, steigt dabei mit sinkender Anzahl an berücksichtigten POD-Moden bei gleichzeitiger Reduktion des Rechenaufwands und vice versa. Bei einer Wahl von $k = \mathrm{p}$ entsteht dabei kein Genauigkeitsverlust, da die restlichen Singulärwerte ohnehin 0 sind. Durch die Dimensionsreduzierung kann ein folgendes Approximationsmodell des Systems aufbauend auf dem Koeffizientenvektor im k-dimensionalen Unterraum deutlich schneller erzeugt werden. Die Vorhersagen des Modells ergeben damit ebenfalls Koeffizientenvektoren für das dimensionsreduzierte System. Diese können anschließend durch die direkte Abhängigkeit der Snapshotmatrix vom Koeffizientenvektor

rückdimensioniert werden, um die Vorhersage auf den ursprünglichen hochdimensionalen Raum abzubilden.

2.2.2 TPS-basierte Interpolation

Die TPS-Methode basiert auf dem Grundansatz, eine Oberfläche mit möglichst geringer Krümmung anhand sogenannter Stützpunkte zu erzeugen, die die zugrunde liegenden Zusammenhänge von Eingangs- und Ausgangsdaten möglichst gut approximiert. Dabei wird meist eine Regularisierung durch einen Regularisierungsterm angewendet, um Abweichungen der Oberfläche von den Stützpunkten zu ermöglichen. Dadurch gilt die TPS-Methode als sehr robust gegenüber Ausreißern und erzeugt Vorhersagen mit hohen Approximationsqualitäten.[56] Sie basiert auf der sogenannten Spline-Funktion [79], die m-dimensionale Eingangsvektoren auf n-dimensionale Ausgangsvektoren anhand einer beliebigen Anzahl an Stützpunkten p als Interpolationsgrundlage abbildet und in (2.58) definiert ist.[34] Für p Stützpunkte stellt $x_\mathrm{i} \in \mathbb{R}^m$ mit $i = 1, 2, \ldots, \mathrm{p}$ die jeweiligen Eingangsvektoren der Stützpunkte dar und R ist dabei die radiale Basisfunktion nach (2.59).

$$f(x) = a_1 + x^T \mathbf{A}_\mathrm{r} + \sum_{\mathrm{i}=1}^{\mathrm{p}} w_\mathrm{i} R(||x - x_\mathrm{i}||) \tag{2.58}$$

$$R = \begin{cases} r^2 log(r), & r > 0 \\ 0, & r = 0 \end{cases} \tag{2.59}$$

Die Lösung des in (2.60) definierten linearen Gleichungssystems ergibt die Stützpunktmatrizen $\mathbf{W}$ und $\mathbf{A}$, sodass die Spline-Funktion zur Bestimmung der Interpolationsoberfläche vollständig definiert ist.

$$\begin{bmatrix} \mathbf{K} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{W} \\ \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{Y} \\ 0 \end{bmatrix} \tag{2.60}$$

Der Ausgangsvektor wird als $y_\mathrm{i} \in \mathbb{R}^n$ mit $i = 1, 2, \ldots, \mathrm{p}$ und die sogenannten Stützpunktmatrizen als $\mathbf{W} \in \mathbb{R}^{p \times p}$ und $\mathbf{A} \in \mathbb{R}^{(m+1) \times n}$ definiert. $\mathbf{W}$ - bestehend aus den einzelnen Werten w_i - und $\mathbf{A}$ beschreiben dabei die Lage und die Krümmung der Oberfläche. Die erste Zeile von $\mathbf{A}$ wird zu a_1 definiert und $\mathbf{A}_\mathrm{r}$ entspricht der Matrix $\mathbf{A}$ ohne die erste Zeile.[30]

In dem linearen Gleichungssystem 2.60 besteht $\mathbf{K} \in \mathbb{R}^{p \times p}$ nach (2.61) und (2.62) mit i,j $\in [1, 2, \ldots, \mathrm{p}]$ aus der Abbildung der euklidischen Distanzen zwischen den Vektorelementen der Stützpunkte durch die in (2.59) definierte radiale Basisfunktion und einem zusätzlichen Regularisierungsterm. $\mathbf{P} \in \mathbb{R}^{p \times (m+1)}$ enthält die Eingangsgrößen der Stützpunkte nach (2.63) und $\mathbf{Y} \in \mathbb{R}^{p \times n}$ die zugehörigen Ausgangsgrößen nach (2.64).[30]

$$K_{\mathrm{ij}} = R(||x_{\mathrm{j}} - x_{\mathrm{i}}||) + \alpha^2 \lambda \tag{2.61}$$

$$\alpha = \frac{1}{p} \sum_{\mathrm{i}=1}^{\mathrm{p}} \sum_{\mathrm{j}=1}^{\mathrm{p}} ||x_{\mathrm{j}} - x_{\mathrm{i}}|| \tag{2.62}$$

$$\mathbf{P} = \begin{bmatrix} 1 & x_{11} & x_{12} & ... & x_{1\mathrm{m}} \\ 1 & x_{21} & x_{22} & ... & x_{2\mathrm{m}} \\ ... & & & & \\ 1 & x_{\mathrm{p}1} & x_{\mathrm{p}2} & ... & x_{\mathrm{pm}} \end{bmatrix} \tag{2.63}$$

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & ... & y_{1\mathrm{n}} \\ y_{21} & y_{22} & ... & y_{2\mathrm{n}} \\ ... & & & \\ y_{\mathrm{p}1} & y_{\mathrm{p}2} & ... & y_{\mathrm{pn}} \end{bmatrix} \tag{2.64}$$

Bei der Anwendung der TPS-basierten Interpolation auf durch die POD-Methode dimensionsreduzierte Snapshots ergeben sich die Ausgangsvektoren y_{i} der Stützpunkte dementsprechend zu den k-dimensionalen POD-Koeffizientenvektoren nach (2.57) der jeweiligen Snapshots. Die Eingangsvektoren x_{i} ergeben sich durch die betrachteten aerodynamischen Parameter wie beispielsweise den Anströmbedingungen, aus denen die Strömungszustände resultieren. Dadurch können aerodynamische Daten vorhergesagt werden, indem die Spline-Funktion die aerodynamischen Parameter auf die POD-Koeffizientenvektoren abbildet und durch die Rückdimensionierung in den ursprünglichen hochdimensionalen Raum der Strömungszustand berechnet wird.

2.3 Praktische Implementierungsdetails

Die gesamte Implementierung ist in Python v3.7 geschrieben. Der Algorithmus zur Erstellung der neuronalen Netze verwendet die Python-Bibliothek Keras v2.3.1 [80] und TensorFlow v2.2.0 [81] als Back-End. Für die Hyperparameteroptimierungen wird zum einen die in der Python-Bibliothek scikit-learn [82] enthaltene Klasse RandomizedSearchCV für die Zufallssuche, als auch das Python-Paket bayesian-optimization [83] für die Bayesian-Optimierung verwendet. Hierbei ist für zukünftige Implementierungen zu erwähnen, dass in der Keras-Bibliothek mittlerweile ein dezidiert für die Hyperparameteroptimierung vorgesehenes Paket, der Keras-Tuner, enthalten ist. Ein schneller Vergleich verschiedener Optimierungsmethoden mit annähernd gleichem Code ist dabei möglich, sodass eine enorme Zeiterstparnis erzielt werden kann. Die Erstellung der klassischen Ersatzmodelle wird mit Hilfe der DLR-Toolbox für Ersatzmodellierung SMARTy (Surrogate Modeling for Aero Data Toolbox in Python) [78, 84] realisiert. In dieser sind sowohl die POD-Methode der Snapshots als auch die TPS-Interpolation nach der Definition in Kapitel 2.2 implementiert.
Alle verwendeten hochgenauen CFD-Daten werden mit dem DLR-internen Strömungslöser TAU [85] generiert, der die Reynolds-gemittelten Navier-Stokes-Gleichungen in Verbindung mit dem Spalart-Allmaras-Turbulenzmodell [86] löst.

3 Untersuchung der Ersatzmodelle anhand aerodynamischer Fallbeispiele

In diesem Kapitel wird das Potential von neuronalen Netzen als Ersatzmodell untersucht, indem diese mit den klassischen Ersatzmodellen anhand von drei verschiedenen Fallbeispielen miteinander verglichen werden. In jedem dieser Fallbeispiele werden die Genauigkeit der Vorhersagen und der zeitliche Rechenaufwand als Kriterien definiert, um die Leistung der neuronalen Netze zu bewerten und mit den klassischen Ersatzmodellen zu vergleichen. Bei den Fallbeispielen handelt es sich um unterschiedliche, typische aerodynamische Herausforderungen, denen eine zweidimensionale stationäre, eine zweidimensionale instationäre bzw. eine dreidimensionale stationäre Strömung zugrunde liegen. Im Fokus steht die Vorhersage der Druckverteilung auf den untersuchten Modellen basierend auf den beschriebenen Strömungen. Die Herausforderung der Fallbeispiele mit stationärer Strömung liegt darin, dass sowohl der sub- als auch transsonische Strömungsbereich mit einem Ersatzmodell untersucht wird, während im Fallbeispiel mit zugrunde liegender instationären Strömung die zeitliche Änderung der Druckverteilung eine besondere Herausforderung darstellt.
Die Fallbeispiele werden als in sich abgeschlossene Kapitel behandelt, da gewonnene Erkenntnisse in die Untersuchungsmethodik der konsekutiven Fallbeispiele einfließen. Im ersten Fallbeispiel wird ein besonderer Fokus auf verschiedene Methoden der Datenaufbereitung und der Hyperparameteroptimierung gelegt. Die als am besten identifizierten Methoden werden im Anschluss bei den beiden anderen Fallbeispielen angewendet.

3.1 Fallbeispiel 1: Stationäre 2D-Profilumströmung

In diesem Kapitel wird eine stationäre Umströmung eines transsonischen NLR7301 Flügelprofils für verschiedene Machzahlen Ma und Anstellwinkel α untersucht. Das Profil ist in Abbildung 3.1 dargestellt und wird mit 597 Oberflächenpunkten beschrieben.

Ziel ist es sowohl die Druckverteilung auf der Profiloberfläche, als auch die skalaren Auftriebs-, Widerstands- und Nickmomentenbeiwerte mithilfe der neuronalen Netze und der klassischen Ersatzmodelle für verschiedene Kombinationen aus Machzahl und Anstellwinkel zu bestimmen und die Ergebnisse zu vergleichen, um das Potential des neuronalen Netzes bewerten zu können.

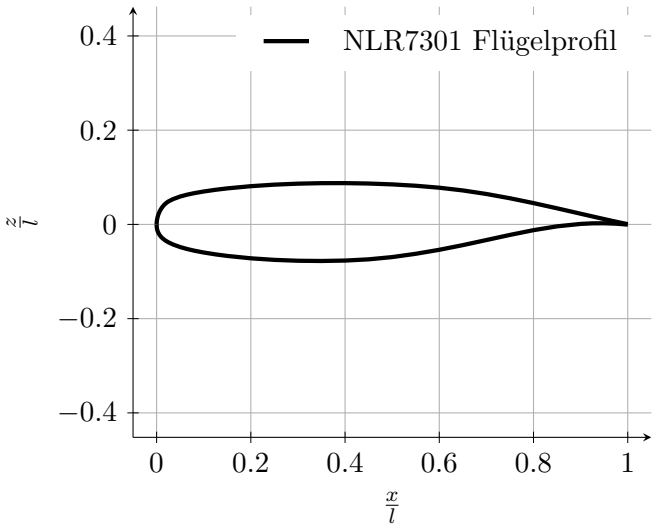


Abbildung 3.1: Fallbeispiel 1: NLR7301 Flügelprofil

3.1.1 Methodik

Im ersten Schritt wird ausschließlich die Druckverteilung betrachtet und basierend auf den zur Verfügung stehenden Daten drei verschiedene Methoden der Datenaufbereitung definiert. Außerdem werden zwei Optimierungsmethoden - die Zufallssuche und die Bayesian Optimierung - für den Entwurfsprozess der neuronalen Netze für die Bestimmung geeigneter Hyperparameter verwendet und bewertet. Hierbei ist zu erwähnen, dass der Zufallssuche zwar kein Optimierungsalgorithmus zugrunde liegt und diese somit keine tatsächliche Optimierungsmethode als solche darstellt, für eine einfache Dokumentation in diesem Kapitel aber dennoch als eine Methode der Hyperparameteroptimierung angesehen wird.
Um die Ergebnisse der neuronalen Netze mit denen der klassischen Ersatzmodelle zu vergleichen, wird zunächst sowohl die beste Optimierungsmethode als auch die beste Datenaufbereitungsmethode identifiziert. Für die Untersuchung der beiden Optimierungsmethoden ist es irrelevant, durch welche der drei Methoden die Daten aufbereitet sind, da es für die Evaluierung ausschließlich wichtig ist, dass beide Optimierungsmethoden auf identisch aufbereiteten Daten basieren. Zu diesem Zweck werden zwei neuronale Netze, die auf derselben zufällig gewählten Datenaufbereitungsmethode beruhen, mit jeweils einer der beiden Optimierungsmethoden erstellt. Die Ergebnisse dieser beiden neuronalen Netze werden miteinander verglichen, um die beste Optimierungsmethode der Hyperparameter zu identifizieren. Anschließend werden die Datenaufbereitungsmethoden untersucht. Hierfür wird für jede Methode der Datenaufbereitung ein separates neuronales Netz erstellt, bei welchem die Hyperparameter mit der Variante optimiert werden, die im vorherigen Schritt als die Effektivere identifiziert worden ist. Die so entstehenden drei neuronalen Netze werden anhand der definierten Kriterien miteinander verglichen und so die beste Methode der Datenaufbereitung herausgefunden.

Für den finalen Vergleich des neuronalen Netzes mit dem klassischen Ersatzmodell wird ein neuronales Netz verwendet, welches auf der besten Datenaufbereitungsmethode kombiniert mit der

besten Optimierungsmethode der Hyperparameter basiert. Das klassische Ersatzmodell besteht aus der POD-Dimensionsreduzierung mit anschließender TPS-Interpolation.

Im Anschluss daran werden die skalaren Beiwerte betrachtet. Diese werden einzeln untersucht, allerdings ist die Methodik für jeden Beiwert dieselbe. Für jede Machzahl-Anstellwinkel-Kombination liegt jeweils genau ein Wert vor, sodass lediglich eine einzige sinnvolle Methode der Datenaufbereitung möglich ist. Für jeden der skalaren Beiwerte wird ein neuronales Netz erstellt, dessen Hyperparameter mit der Optimierungsmethode bestimmt werden, die im Vorfeld als die Beste identifiziert worden ist.
Zusätzlich zu der direkten Berechnung durch das neuronale Netz wird jeder der Beiwerte auf indirektem Wege durch die Integration der Druckverteilung entlang der Oberfläche berechnet, die im vorherigen Schritt von dem besten neuronalen Netz vorhergesagt wurde.
Außerdem wird ein klassisches Ersatzmodell erstellt, was ausschließlich aus der TPS-Interpolation besteht. Die POD-Dimensionsreduzierung erübrigt sich, da der jeweilige skalare Beiwert als Zielgröße bereits eindimensional ist.

Abschließend werden die Vorhersageergebnisse und der zeitliche Rechenaufwand der neuronalen Netze sowohl für die Druckverteilung als auch für die skalaren Beiwerte mit denen der klassischen Ersatzmodelle verglichen, um das Potential der neuronalen Netze als Ersatzmodell zu bewerten.

3.1.1.1 Beschreibung der Daten

Die Daten bestehen aus den sich ergebenden Beiwerten c_l, c_d, c_my und c_p für 98 verschiedene Kombinationen aus Anstellwinkel α und der Machzahl Ma. Sie spannen einen Parameterbereich von $0.3 \leq Ma \leq 0.75$ und $-3 \leq \alpha \leq 5$ auf, womit sowohl der sub- als auch transsonische Strömungsbereich abgedeckt wird. Die Auswahl der Daten erfolgt in diesem Fall durch ein DoE basierend auf der sogenannten *Halton Sequence*, da sich diese Methode besonders gut bei geringen Datenmengen eignet.[87] Die Daten werden anschließend in 80% Trainings- und 20% Testdaten aufgeteilt. Hierbei dienen die Trainingsdaten den Ersatzmodellen zum Erlernen der nicht-linearen Zusammenhänge zwischen der Machzahl-Anstellwinkel Kombination und den sich ergebenden aerodynamischen Beiwerten. Hierfür werden die Trainingsdaten wiederum in ein Trainingssatz und einen Validierungssatz aufgeteilt, um Kapitel 2.1.5 entsprechend die Generalisierungsfähigkeit des neuronalen Netzes während des Trainings zu überwachen und das frühzeitige Abbruchkriterium anwenden zu können. Diese erlernten Zusammenhänge werden auf die Testdaten angewandt und so die Vorhersagegenauigkeit der Ersatzmodelle basierend auf einem Vergleich mit der CFD-Lösung ermittelt. Die resultierenden 98 CFD-Lösungen und ihre Unterteilung in 79 Trainings- und 19 Testdaten sind in Abbildung 3.2 dargestellt.

Für eine sinnvolle Darstellung der skalaren Beiwerte im aerodynamischen Kontext bietet es sich an, ihren Verlauf über dem Anstellwinkel für konstante Machzahlen zu zeigen. Daher werden für zufällig gewählte Machzahlen $Ma = 0.36$ und $Ma = 0.58$ sowie ganzzahlige Anstellwinkel im Bereich von $-5 \leq \alpha \leq 8$ ebenfalls CFD-Daten generiert, die lediglich der genannten visuellen Darstellung der

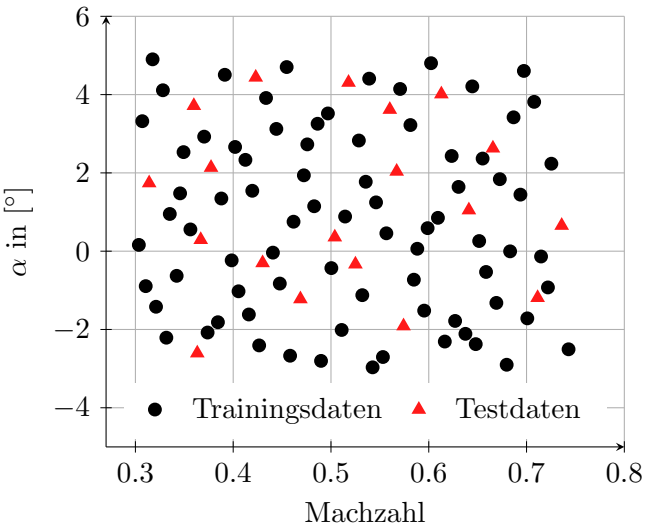


Abbildung 3.2: Fallbeispiel 1: Trainings- und Testdaten

Ergebnisse dienen und kein Bestandteil der Trainingsdaten sind. Außerdem werden hierdurch die Extrapolationsfähigkeiten der Ersatzmodelle getestet, da der Anstellwinkel einiger Daten außerhalb des Trainingsbereichs liegt. Die Vorhersagegenauigkeit im Vergleich zu den klassischen Ersatzmodellen wird jedoch lediglich anhand der im Trainingsbereich befindlichen Testdaten evaluiert, da die Ersatzmodelle im Normalfall nicht zur Extrapolation benutzt werden und eine entsprechend geringe Vorhersagegenauigkeit außerhalb des Trainingsbereichs erwartet wird.[45]

3.1.1.2 Datenaufbereitung

Wie in Kapitel 2.1.4 erwähnt, ist es für ein robustes und zügiges Training wichtig, die Daten zu skalieren. Daher werden die Eingangsgrößen für die neuronalen Netze hier auf den Bereich [-1, 1] skaliert. Basierend auf den CFD-Daten werden für die Bestimmung der Druckverteilung auf dem Profil drei unterschiedliche Datenaufbereitungsmethoden und somit drei verschiedene Kombinationen der Eingangs- (engl. *input*) und Ausgangsvektoren (engl. *output*) des neuronalen Netzes untersucht, die im Folgenden definiert werden.

Methode A

Die Daten werden so aufbereitet, dass für jeden Eingangsvektor, bestehend aus den Eingangsgrößen Anstellwinkel und Machzahl, die gesamte Druckverteilung - d.h. die Druckbeiwerte an allen Oberflächenpunkten - mit einer einzigen Vorhersage bestimmt werden. Dadurch ergeben sich die Ein- und Ausgangsvektoren zu (3.1) und (3.2), wobei N die Anzahl der Oberflächenpunkte des Profils ist. Für dieses Fallbeispiel ist daher N = 597.

$$input_{\mathrm{A}} = (Ma, \alpha) \tag{3.1}$$

$$output_{\mathrm{A}} = (c_{\mathrm{p}_1}, c_{\mathrm{p}_2}, ..., c_{\mathrm{p}_{\mathrm{N}}}) \tag{3.2}$$

Methode B

Diese Methode ermöglicht die Bestimmung des Druckbeiwerts für jeden einzelnen Oberflächenpunkt. Hierfür wird die jeweilige Position des Oberflächenpunktes als zusätzliche Information im *input* berücksichtigt, was durch x und z im Bezug auf die Profillänge l als kartesischen Koordinaten realisiert wird. Da sich durch diese Aufteilung für jede Machzahl-Anstellwinkel-Kombination N separate Ein- und Ausgangsvektoren ergeben, steigt die Gesamtanzahl an Ein- und Ausgangsvektoren im Vergleich zur der Methode A um den Faktor N auf $N \cdot 79 = 47163$. Es ist zu erwarten, dass aufgrund der Erhöhung der Datensätze der zeitliche Aufwand, aber auch die Genauigkeit der Vorhersagen der neuronalen Netze durch die Zusatzinformation der Position steigt. Die Ein- und Ausgangsgrößen ergeben sich zu (3.3) und (3.4) mit $\mathrm{n} \in [1, \mathrm{N}]$.

$$input_{\mathrm{B}} = (Ma, \alpha, \frac{x}{l}_{\mathrm{n}}, \frac{z}{l}_{\mathrm{n}}) \tag{3.3}$$

$$output_{\mathrm{B}} = (c_{\mathrm{p}_{\mathrm{n}}}) \tag{3.4}$$

Methode C

Diese Methode unterliegt dem gleichen Ansatz der Datenaufteilung wie Methode B und unterscheidet sich nur insofern, dass der zugehörige Oberflächenpunkt durch die kurvilineare Koordinate s beschrieben wird, wodurch sich (3.5) und (3.6) mit $\mathrm{n} \in [1, \mathrm{N}]$ ergeben. Die Koordinate beginnt von der Hinterkante ausgehend mit $s = 0$ entlang der Oberfläche des Profils gegen den Uhrzeigersinn, bis sie wieder an der Hinterkante für $s = 1$ ankommt. Dadurch wird wie in Methode B auch die Position des Oberflächenpunktes als zusätzliche Information hinzugefügt, jedoch reduziert sich die Anzahl der Eingangsgrößen von 4 auf 3, weshalb eine ähnliche Vorhersagegenauigkeit, aber ein geringerer zeitlicher Rechenaufwand der neuronalen Netze im Vergleich zu Methode B erwartet wird.

$$input_{\mathrm{C}} = (Ma, \alpha, s_{\mathrm{n}}) \tag{3.5}$$

$$output_{\mathrm{C}} = (c_{\mathrm{p}_{\mathrm{n}}}) \tag{3.6}$$

Für die Bestimmung der skalaren Beiwerte werden die beiden verschiedene Methoden folgendermaßen definiert:

Methode 1

Für jeden skalaren Beiwert wird jeweils ein neuronales Netz erstellt und mit der als beste identifizierten Methode optimiert. Der Eingangsvektor pro Machzahl-Anstellwinkel-Kombination ist daher für alle skalaren Beiwerte wie in (3.7) beschrieben. Die Ausgangsvektoren der Netze ergeben sich abhängig vom jeweiligen skalaren Beiwert zu (3.8) - (3.10).

$$input = (Ma, \alpha) \tag{3.7}$$

$$output_{\mathrm{l}} = (c_{\mathrm{l}}) \tag{3.8}$$

$$output_{\mathrm{d}} = (c_{\mathrm{d}}) \tag{3.9}$$

$$output_{\mathrm{my}} = (c_{\mathrm{my}}) \tag{3.10}$$

Methode 2

Die skalaren Beiwerte lassen sich durch die Integration des Druckbeiwerts c_{p} und des Wandreibungsbeiwerts c_{f} entlang der Profiloberfläche berechnen. Da die Druckverteilung im Allgemeinen mehr Informationen bereitstellt als die skalaren Beiwerte - wie beispielsweise bei der Beobachtung eines Verdichtungsstoßes - und somit für den Ingenieur interessanter ist, wird die Druckverteilung meistens ohnehin bestimmt. Durch das Einbeziehen des Ergebnisses aus dem integrativen Ansatz in den finalen Vergleich kann erörtert werden, ob die Notwendigkeit eines neuronalen Netzes dezidiert für die Bestimmung der skalaren Beiwerte überhaupt besteht. Für die Wandreibungsbeiwerte wird im Rahmen dieser Arbeit kein Ersatzmodell untersucht, weshalb in allen aufgeführten skalaren Beiwerten lediglich die Druckkräfte und ausdrücklich nicht die Wandreibungskräfte berücksichtigt sind.

3.1.1.3 Entwurfsprozess des neuronalen Netzes

Die optimalen Hyperparameter eines neuronalen Netzes hängen stark von der Komplexität des Problems ab. Das Ziel der Hyperparameteroptimierung ist es, ein neuronales Netz mit Hyperparametern zu finden, die eine möglichst hohe Vorhersagegenauigkeit bei möglichst geringem zeitlichen Rechenaufwand ermöglichen. Zur Bestimmung dieser Hyperparameter eignen sich einige Methoden, wie in Kapitel 2.1.7 beschrieben.

Es werden zwei Methoden der Hyperparameteroptimierung, die Zufallssuche und die Bayesian-Optimierung untersucht, um im Anschluss die bessere Methode zu identifizieren. Zu diesem Zweck werden zwei neuronale Netze mit der jeweils besten Hyperparameterkombination der unterschiedlichen Optimierungsmethoden erstellt. Hierbei wird beiden neuronalen Netzen dieselbe Datenaufbereitungsmethode B zugrunde gelegt. Durch einen Vergleich der Ergebnisse der beiden neuronalen Netze wird die geeignetere Optimierungsmethode der Hyperparameter identifiziert. Da der Bereich der optimalen Hyperparameter an dieser Stelle noch nicht eingegrenzt ist, werden vorläufige Tests mit manueller Wahl der Hyperparameter mit einer geringen Anzahl an Trainingsepochen durch-

geführt. Dadurch kann der Hyperparameterbereich - insbesondere die Anzahl an Schichten und Neuronen - sinnvoll eingegrenzt werden.

Um Vergleichbarkeit zu gewährleisten, wird der Hyperparameterbereich bei beiden Optimierungsmethoden gleich gewählt. In Tabelle 3.1 ist dieser Bereich für die einzelnen Hyperparameter, abhängig von den Datenaufbereitungsmethoden, dargestellt.
Die zunehmende Komplexität der nicht-linearen Beziehung zwischen *input* und *output* im Falle der Datenaufbereitungsmethode A, in der die Druckbeiwerte aller Oberflächenpunkte auf einmal vorhersagt werden, verlangt eine höhere Kapazität des neuronalen Netzes, als die der neuronalen Netze, denen Methode B und C unterliegen. Der hier gewählte Ansatz, um dies zu berücksichtigen, ist eine Erhöhung der maximal möglichen Anzahl an Neuronen pro Schicht auf 1000 für das neuronale Netz, dem die mit Methode A aufbereiteten Daten zugrunde liegen.

Wie in Kapitel 2.1.7 erwähnt, ist eine exponentielle Abnahme der Lernrate über die Trainingsepochen vielversprechend. Der Verlauf der Lernrate lr wird durch (3.11) beschrieben.

$$lr = lr_0 \cdot e^{-k \cdot Z} \tag{3.11}$$

Hierbei ist lr_0 die Anfangslernrate, k der Abnahmefaktor und Z die Anzahl an bereits durchlaufenen Trainingsepochen. lr_0 und k sind vor dem Training zu wählen und zählen somit zu den Hyperparametern, welche hier ebenfalls optimiert werden.

Als Aktivierungsfunktion wird die ReLU-Funktion nach (2.47) verwendet, die Gewichtsfaktoren während des Trainings werden durch den Adam-Algorithmus optimiert und als Kostenfunktion wird der MSE verwendet. Diese genannten Hyperparameter werden zuvor gewählt und bleiben während der Optimierung konstant. Alle weiteren, nicht genannten Hyperparameter unterliegen den Standarteinstellungen der Keras-Bibliothek und können in der Dokumentation [80] nachgelesen werden.

Die Anzahl an unterschiedlichen Hyperparameterkombinationen, die während der Optimierungen getestet werden, ist in jeder Methode gleich. Aufbauend auf den Erkenntnissen der vorläufigen manuellen Untersuchungen wird diese auf 140 Iterationen und die maximale Anzahl an Trainingsepochen auf 3000 gesetzt. Es wird das in Kapitel 2.1.5 beschriebene frühzeitige Abbruchkriterium verwendet, um die Überanpassung der neuronalen Netze zu verhindern und zugleich Rechenzeit zu sparen. Das Abbruchkriterium wird so definiert, dass der Vorhersagefehler gemessen als MAE der neuronalen Netze im Bezug auf die Validierungsdaten um mindestens 0.002 innerhalb von 50 Epochen. Diese Werte haben sich in den manuellen Voruntersuchungen als sinnvoll erwiesen hat.

Nach dem Abschluss der beiden Optimierungsmethoden wird jeweils ein neuronales Netz mit den resultierten besten Hyperparametern der jeweiligen Optimierungsmethode erstellt und trainiert.
Anders als bei der Optimierung selbst werden dabei die Abbruchkriterien des Trainings abgeschwächt, sodass der Vorhersagefehler um lediglich 0.0005 anstatt 0.002 nach der genannten Anzahl

Hyperparameter	Bereich Methode A	Bereich Methoden B & C
Batchgröße bs	[16, 512]	[16, 512]
Dropoutrate	[0.0, 0.5]	[0.0, 0.5]
Anfangslernrate lr_0	[0.001, 0.1]	[0.001, 0.1]
Abnahmefaktor k	[1e-6, 1e-2]	[1e-6, 1e-2]
Anzahl verdeckter Schichten	[1, 11]	[1, 11]
Neuronenanzahl pro Schicht	[6, 1000]	[6, 120]

Tabelle 3.1: Fallbeispiel 1: In der Optimierung berücksichtigte Hyperparameter und deren zulässiger Bereich

an Trainingsepochen sinken muss. Hierdurch werden die neuronalen Netze mit einer größeren Anzahl an Epochen trainiert. Das führt zu einem zeitaufwändigerem Training, allerdings ist es wahrscheinlich, dass eine Genauigkeitssteigerung der Vorhersagen erreicht wird. Die gesetzten Werte des Abbruchkriteriums spigeln sich daher direkt in der Vorhersagegenauigkeit und dem zeitlichen Aufwand wider, sodass die gesetzten Werte im Folgenden nicht weiter explizit angegeben werden.
Die Vorhersageergebnisse der beiden trainierten neuronalen Netze werden anschließend verglichen, um die beste Methode der Hyperparameteroptimierung zu identifizieren. Abschließend wird für den Vergleich mit dem klassischen Ersatzmodell das neuronale Netz basierend auf der besten Datenaufbereitungsmethode und der besten Hyperparameteroptimierung verwendet.

3.1.2 Ergebnisse und Diskussion

Die Abweichung der vorhergesagten Druckverteilung von der CFD-Lösung, gemessen als MAE und MSE, der aus den unterschiedlichen Optimierungsansätzen entstandenen besten neuronalen Netze, denen die mit Methode B aufbereiteten Daten zugrunde liegen, sind in Tabelle 3.2 gezeigt.

Methode	MAE $[\cdot 10^{-3}]$	MSE $[\cdot 10^{-3}]$	Optimierungszeit [h]
Zufallssuche	7.43	0.67	1.7
Bayesian-Optimierung	6.15	0.44	3.4

Tabelle 3.2: Fallbeispiel 1: Ergebnisse der Bayesian-Optimierung und der Zufallssuche zur Bestimmung geeigneter Hyperparameter

Die Bayesian-Optimierung findet Hyperparameter für die neuronalen Netze, die mit einer durchschnittlichen absoluten Abweichung von $MAE = 6.15 \cdot 10^{-3}$ zu genaueren Ergebnissen führen als die Zufallssuche mit $MAE = 7.43 \cdot 10^{-3}$, sie braucht hierzu jedoch doppelt so lange.
Generell wird die Vorhersagegenauigkeit als Bewertungskriterium zunächst als relevanter eingestuft, da der Einsatz neuronaler Netze nur dann sinnvoll ist, unabhängig von dem zeitlichen Aufwand, wenn die Ergebnisse ähnlich oder akkurater sind, als die der klassischen Ersatzmodelle.
Aufgrund der höheren Genauigkeit wird die Bayesian-Optimierung als bessere Optimierungsmethode der Hyperparameter identifiziert. Zur Bestimmung geeigneter Hyperparameter der neuronalen Netze werden aufbauend auf diesem Ergebnis im weiteren Verlauf der Arbeit ausschließlich die Bayesian-Optimierung verwendet.

Die gefundenen besten Hyperparameter der beiden Methoden sind in Tabelle 3.3 gezeigt.

Hyperparameter	**Zufallssuche**	**Bayesian-Optimierung**
Batchgröße bs	300	443
Dropoutrate	0.0	0.0
Anfangslernrate lr_0	0.0016	0.001
Abnahmefaktor k	1e-2	1e-2
Anzahl verdeckter Schichten	7	11
Neuronenanzahl pro Schicht	101, 93, 40, 81, 91, 112, 69	11x120

Tabelle 3.3: Fallbeispiel 1: Ergebnisse der durch die Zufallssuche und die Bayesian-Optimierung bestimmten besten Hyperparameter

Besonders auffällig ist die große Abweichung der Kapazität der neuronalen Netze. Während die Bayesian-Optimierung das größte mögliche neuronale Netz mit 11 verdeckten Schichten und jeweils 120 Neuronen als bestes identifiziert, findet die Zufallssuche ein deutlich kleineres neuronales Netz mit 7 verdeckten Schichten und Neuronen im Bereich von 40-112. Es könnte sein, dass durch die zufällige Wahl der getesteten Hyperparameter in der Zufallssuche keine größeren neuronalen Netze getestet wurden, was ebenfalls die kürzere Optimierungszeit der Zufallssuche erklären würde. Außerdem werden durch die Bayesian-Optimierung sowohl für die Anfangslernrate, den Abnahmefaktor und die Größe des Netzwerks jeweils die Randwerte des möglichen Bereichs als Optimum identifiziert, weshalb eine erneute Optimierung mit größeren Bereichen wahrscheinlich zu besseren Ergebnissen führen würde.

Die Ergebnisse der Bayesian-Optimierung für die unterschiedlichen Datenaufbereitungsmethoden A, B und C sind in Tabelle 3.4 aufgelistet. Auch hier ist es auffällig, dass für alle drei Methoden das jeweils größte mögliche neuronale Netz als Bestes identifiziert wird. Außerdem werden für die Anfangslernrate und den Abnahmefaktor ebenfalls die Randwerte des möglichen Bereichs als Beste identifiziert. Dies lässt wieder die Vermutung zu, dass der Hyperparameterbereich wahrscheinlich zu klein gewählt ist und eine weitere Optimierung mit erweitertem Hyperparameterbereich durchgeführt werden sollte, um noch bessere Ergebnisse zu erhalten, was für eine erste hier durchgeführte Untersuchung jedoch nicht zwingend erforderlich ist.

Hyperparameter	**Methode A**	**Methode B**	**Methode C**
Batchgröße bs	182	443	300
Dropoutrate	0.0	0.0	0.0
Anfangslernrate lr_0	0.01	0.001	0.001
Abnahmefaktor k	1e-6	1e-2	1e-2
Anzahl verdeckter Schichten	11	11	11
Neuronenanzahl pro Schicht	11x1000	11x120	11x120

Tabelle 3.4: Fallbeispiel 1: Ergebnisse der durch die Bayesian-Optimierung bestimmten besten Hyperparameter der neuronalen Netze, die auf Daten der Aufbereitungsmethoden A, B und C beruhen

Die Vorhersagen des Druckbeiwerts der neuronalen Netze, basierend auf den verschiedenen Methoden A, B und C der Datenaufbereitung, deren Hyperparameter jeweils mit der Baysian Optimierung bestimmt sind, sind für ausgewählte Machzahl-Anstellwinkel-Kombinationen der Testdaten in Abbildung 3.3 dargestellt. Die Genauigkeit der Ergebnisse ist für alle drei Methoden bei Testdaten im subsonischen Bereich, bei denen keine Verdichtungsstöße auf dem Profil auftreten, sehr ähnlich. Sind allerdings im transsonischen Bereich auftretende Verdichtungsstöße vorhanden, schneidet die Methode A, bei denen die gesamte Druckverteilung auf einmal vorhergesagt wird, im Vergleich zu beiden anderen Methoden deutlich schlechter ab. Anstelle eines Stoßes wird mit Methode A eher eine kontinuierliche Abnahme des Druckbeiwerts vorhergesagt, wodurch alle Druckbeiwerte unmittelbar vor und nach dem Stoß stark von der CFD-Lösung abweichen. Die Vorhersagen der neuronalen Netze der Methode B sind nur geringfügig genauer als die der Methode C. Die Ergebnisse für die unterschiedlichen Datenaufbereitungsmethoden sind in Tabelle 3.5 aufgeführt.

Die Vorhersagegenauigkeit der neuronalen Netze, denen die Datenaufbereitungmethode B unterliegt, ist gemessen am MAE bzw. MSE für die Testdaten am genauesten. Hinzu kommt, dass die mittels CFD-Simulationen erzeugten Oberflächendaten ohnehin in kartesischem Format vorliegen, wodurch eine Umwandlung in die kurvilineare Darstellung für Methode C einen zusätzlichen zeitlichen Aufwand bedeutet.
Weiterhin wird erkenntlich, dass die Methode A eine knapp um den Faktor $N = 597$ geringere Vorhersagezeit aufweist als Methode B und C. Das liegt daran, dass für eine Vorhersage der Druckverteilung in Methode A nur ein Vorhersageprozess des neuronalen Netzes durchlaufen werden muss, während in den beiden anderen Methoden 597 Vorhersageprozesse erforderlich sind. Allerdings weist die Methode A starke Defizite in der Genauigkeit auf. Aus diesen Gründen wird die Methode B als beste Datenaufbereitungsmethode identifiziert und zum finalen Vergleich mit den klassischen Ersatzmodellen verwendet.

Methode	**MAE** $[\cdot\mathbf{10^{-3}}]$	**MSE** $[\cdot\mathbf{10^{-3}}]$	**Trainingszeit [s]**	**Vorhersagezeit [s]**
A	10.03	0.86	333	0.053
B	6.11	0.44	458	34.45
C	8.98	0.58	363	39.40

Tabelle 3.5: Fallbeispiel 1: Vorhersagefehler gemessen als MAE und MSE der neuronalen Netze für alle Testdaten, denen die verschiedenen Datenaufbereitungsmethoden zugrunde liegen

Die für den finalen Vergleich relevanten Vorhersagen des neuronalen Netzes, welches der Bayesian-Optimierung für die Hyperparameterbestimmung und dem $input_{\mathrm{B}}$ der Datenaufbereitungsmethode B unterliegt, wie auch die Vorhersagen des klassischen Ersatzmodells sind in Abbildung 3.4 ebenfalls für beispielhafte Testdaten im sub- und transsonischen Bereich dargestellt. Auch hier ist zu erkennen, dass es von hoher Relevanz ist, ob ein Verdichtungsstoß vorliegt. Die Genauigkeit der Ersatzmodelle ist für die Testdaten ohne Verdichtungsstoß im subsonischen Bereich sehr ähnlich. Für transsonische Testdaten, bei denen ein Verdichtungsstoß vorliegt, ist das neuronale Netz deutlich genauer als das klassische Ersatzmodell. Dies spiegelt sich ebenfalls in Tabelle 3.6 in dem um Faktor 0.58

kleineren MAE der neuronalen Netze gegenüber dem des POD+TPS-Ersatzmodells bezogen auf alle Testdaten wider.

Ein weiteres Kriterium für die Potentialbewertung der neuronalen Netze als Ersatzmodell ist der zeitliche Rechenaufwand, der für beide Ersatzmodelle ebenfalls in Tabelle 3.6 aufgelistet ist. Die Vorhersagezeit des neuronalen Netzes ist, auch wenn es mit den Daten, die nach Methode B oder C aufbereitet sind, trainiert ist, wodurch eine Vorhersage genau einen Druckbeiwert bei einem bestimmten Oberflächenpunkt beschreibt, immer bezogen auf die Vorhersage der Druckbeiwerte auf der gesamten Oberfläche, sodass die Vorhersagezeiten der verschiedenen Ersatzmodelle direkt miteinander vergleichbar sind. Eine deutliche zeitliche Überlegenheit des klassischen Ersatzmodells mittels POD-Dimensionsreduzierung und anschließender TPS-Interpolation wird ersichtlich. Das Training des neuronalen Netzes dauert mit 8 Minuten knapp 260-mal so lange wie das Training des POD+TPS-Ersatzmodells. Die Vorhersage der Druckverteilung für eine beliebige Machzahl-Alpha-Kombination dauert mit dem neuronalen Netz mit 34.35 Sekunden mehr als 200-mal so lange wie mit dem klassischen Ersatzmodell.

Für einen Gesamtvergleich der Ersatzmodelle sollte die Zeit für die Optimierung der Hyperparameter ebenfalls mit in die Bewertung einfließen, da diese sehr zeitintensiv ist. Analog sollte die Zeit für die Identifikation des besten Ersatzmodells aller in der SMARTy Toolbox enthaltenen klassischen Ersatzmodelle mit bester Konfiguration in die zeitliche Bewertung einfließen. Durch die viel kürzere Trainingszeit der klassischen Ersatzmodelle ist abzuschätzen, dass für die Bestimmung des besten klassischen Ersatzmodells ebenfalls eine deutlich geringere Optimierungszeit benötigt wird. Dieser zeitliche Vorteil kann hier allerdings nicht explizit gezeigt werden, da in der SMARTy Toolbox noch keine solche Identifikationsmethode der Ersatzmodelle für die Bestimmung der Druckverteilung existiert. Folglich wird im Vergleich ausschließlich die Trainings- und Vorhersagezeit der Ersatzmodelle berücksichtigt.

Ersatzmodell	**MAE** $[\cdot 10^{-3}]$	**MSE** $[\cdot 10^{-3}]$	**Trainingszeit [s]**	**Vorhersagezeit [s]**
NN	6.11	0.44	458	34.45
POD+TPS	10.48	2.11	1.71	0.17

Tabelle 3.6: Fallbeispiel 1: Vorhersagefehler gemessen als MAE und MSE und zeitlicher Rechenaufwand des neuronalen Netzes und des klassischen POD+TPS-Ersatzmodells

Die Vorhersageergebnisse für den Auftriebs-, den Widerstands- und den Nickmomentenbeiwert, die durch Methode 1 mittels separaten neuronalen Netzen, Methode 2 auf integrativem Wege indirekt oder durch das klassische Ersatzmodell, welches hier ausschließlich aus der TPS-Interpolation besteht, berechnet worden sind, sind in Abbildung 3.5 beispielhaft für zufällig gewählte konstante Machzahlen von $Ma = 0.36$ und $Ma = 0.58$ für unterschiedliche Anstellwinkel gezeigt. Die Druckverteilung, die der integrativen Berechnung zugrunde liegt, sind mit dem selben neuronalen Netz berechnet, welches für den finalen Vergleich mit dem POD+TPS-Ersatzmodell verwendet wurde.
Es ist zu erkennen, dass die Ergebnisse der integrativen Berechnung für beide gezeigten Machzahlen für den Auftriebs- und Widerstandsbeiwert der CFD-Lösung am nächsten sind, während die direk-

ten Vorhersagen der jeweiligen neuronalen Netze am stärksten abweichen. In Tabelle 3.7 sind die Vorhersagefehler der Verfahren für die verschiedenen skalaren Beiwerte gezeigt. Es wird ersichtlich, dass die TPS-Methode für den Widerstandsbeiwert insgesamt berechnet für alle Testdaten einen um Faktor 4 höheren MAE als die indirekte Methode aufweist und sich dementsprechend für den Widerstandsbeiwert nicht eignet. Diese Methode eignet sich dafür am besten für den Nickmomentenbeiwert, der allerdings bei $Ma = 0.58$ und hohen Anstellwinkeln durch kein Ersatzmodell genau vorhergesagt wird.
Es ist zu erwähnen, dass alle Vorhersagefehler auf die integrativ berechneten skalaren Beiwerte der CFD-Lösung bezogen sind. Auch beim indirekten Berechnungsverfahren durch Druckbeiwertsvorhersagen des neuronalen Netzes ist der Fehler anhand der integrativ berechneten skalaren Beiwerte gemessen und nicht etwa anhand eines Vergleichs der vorhergesagten Druckverteilungen, damit die verschiedenen Verfahren vergleichbar sind.
Das indirekte Bestimmungsverfahren ergibt zwar die genausten extrapolativen Vorhersagen, jedoch ist die Extrapolationsgenauigkeit wie erwartet für alle Ersatzmodelle deutlich schlechter als die Vorhersagegenauigkeit innerhalb des Trainingsbereichs, was die normalerweise ausschließliche Verwendung der Ersatzmodelle im Trainingsbereich bestärkt.

Verfahren	**MAE c_l**	**MAE c_d**	**MAE c_{my}**	**MSE c_l**	**MSE c_d**	**MSE c_{my}**
NN direkt	0.07337	0.02566	0.01259	0.01787	0.00180	0.00056
NN indirekt	0.04434	0.01713	0.01416	0.00654	0.00109	0.00067
TPS	0.04557	0.06892	0.01181	0.00760	0.01549	0.00055

Tabelle 3.7: Fallbeispiel 1: Vorhersagefehler gemessen als MAE und MSE der unterschiedlichen Bestimmungsverfahren für die verschiedenen skalaren Beiwerte

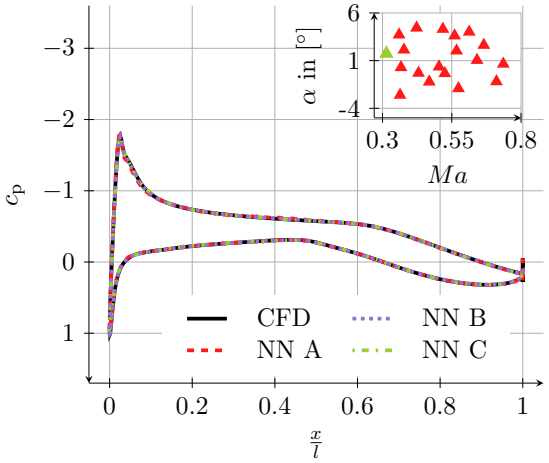
3.1.3 Fazit

Generell lohnt sich der hohe zeitliche Aufwand der neuronalen Netze für Vorhersagen im transsonischen Bereich, da die Vorhersagegenauigkeit deutlich höher beim Auftreten von Verdichtungsstößen ist als die der klassischen Ersatzmodelle. Dagegen lohnt es sich nicht, neuronale Netze für ausschließlich subsonische Bereiche bzw. subsonische Profile zu verwenden, da die Vorhersageergebnisse der klassischen Ersatzmodelle bei subsonischen Strömungsphänomenen ähnlich gut oder besser bei deutlich geringerem zeitlichen Rechenaufwand sind.

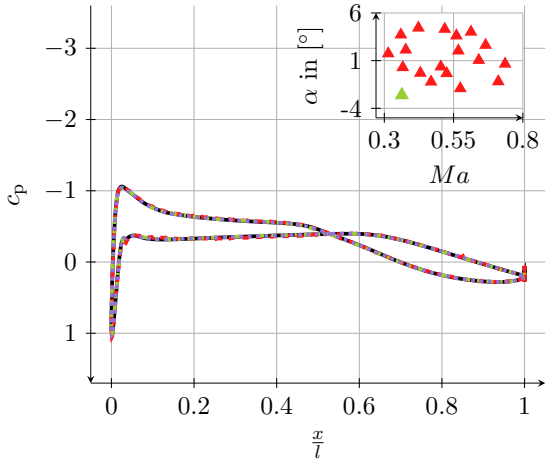
Der Einsatz eines neuronales Netzes dezidiert für die Bestimmung skalarer Beiwerte ist im Allgemeinen weder genauer noch effizienter als der Einsatz des klassischen Ersatzmodells, weshalb sich das neuronales Netz für diese Aufgabe nicht eignet. Der integrative Ansatz aus vorhergesagter Druckverteilung zur Bestimmung der skalaren Beiwerte ist relativ genau möglich, allerdings müsste für die wahren skalaren Beiwerte ebenfalls die Wandreibungskoeffizienten c_f bestimmt und in der Integration berücksichtigt werden. Dafür müsste ein separates neuronales Netz optimiert und trainiert werden, wodurch sich der zeitliche Aufwand für die Ergebnisse mittels neuronaler Netze

verdoppelt. Deutlich praktikabler ist es, die ohnehin vorliegenden wahren skalaren Beiwerte aus der CFD-Lösung zu nutzen, um ein klassisches Ersatzmodell zu erstellen.

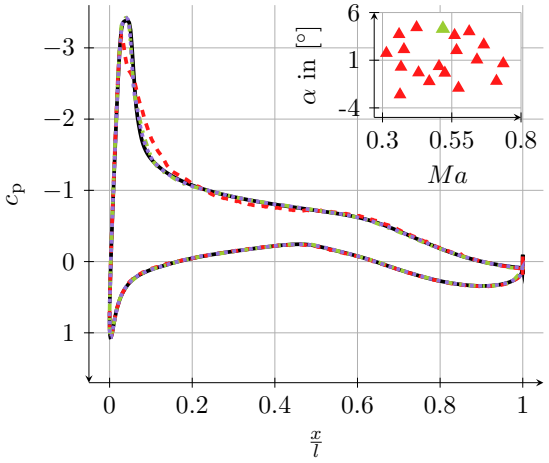
Aufgrund der ungenauen Vorhersageergebnisse außerhalb des Trainingsbereichs sollte generell keins der untersuchten Ersatzmodelle zur Extrapolation eingesetzt werden.



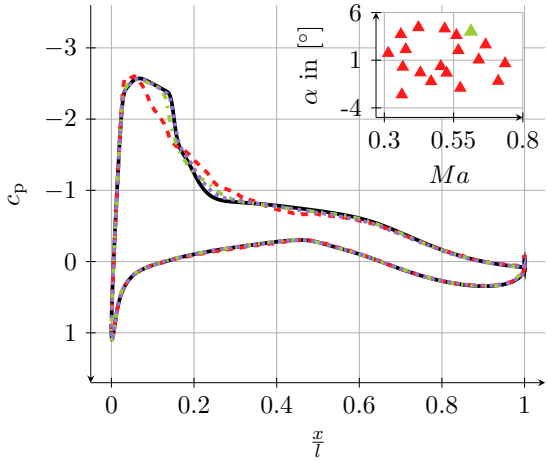
(a) $Ma = 0.31406$ und $\alpha = 1.74074$



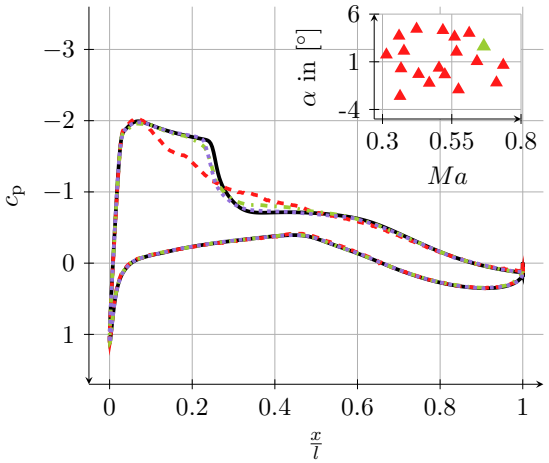
(b) $Ma = 0.36328$ und $\alpha = -2.60494$



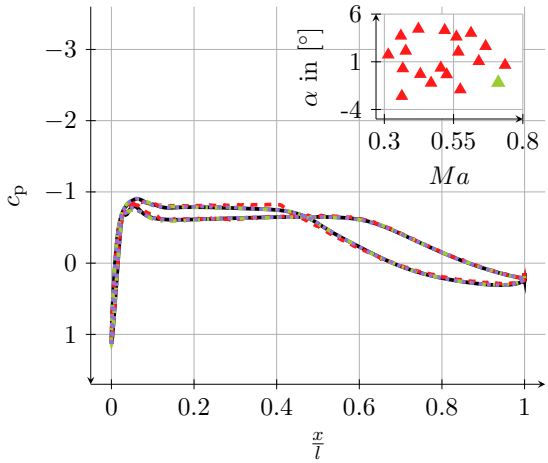
(c) $Ma = 0.51797$ $\alpha = 4.30864$



(d) $Ma = 0.61289$ und $\alpha = 4.01235$

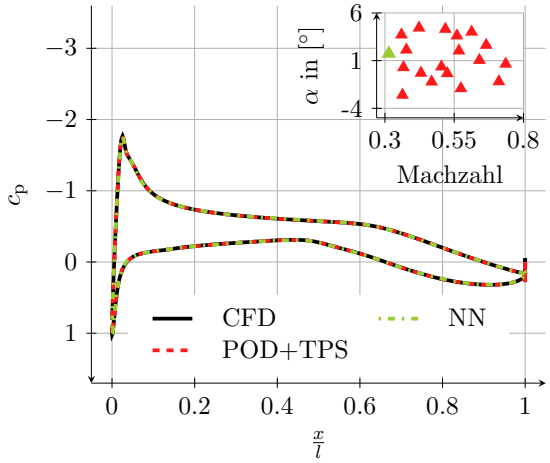


(e) $Ma = 0.66562$ und $\alpha = 2.62963$

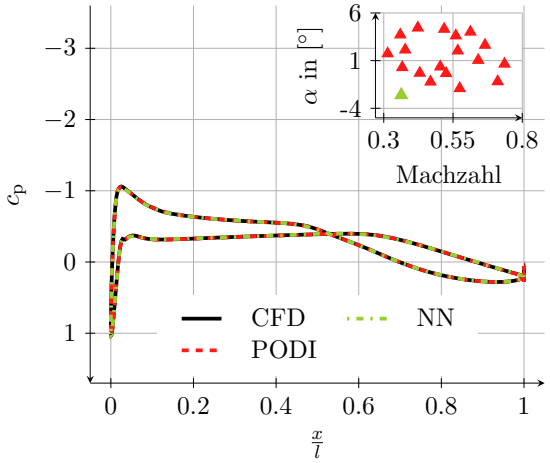


(f) $Ma = 0.71133$ und $\alpha = -1.1893$

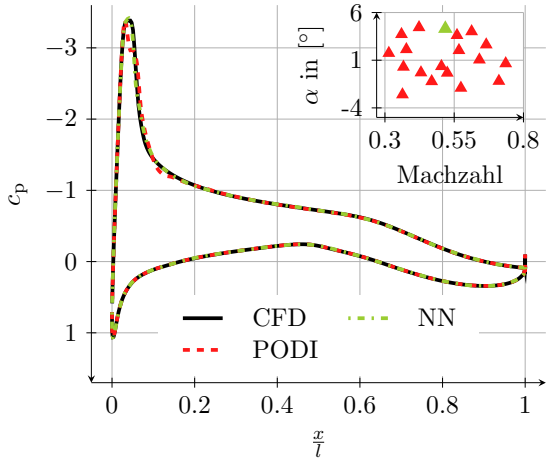
Abbildung 3.3: Fallbeispiel 1: Vorhersagen des Druckbeiwerts für ausgewählte Machzahl-Anstellwinkel-Kombinationen der neuronalen Netze, denen jeweils die mit einer der drei Methoden A, B und C aufbereiteten Daten zugrunde liegen und deren Hyperparameter jeweils durch die Bayesian-Optimierung bestimmt sind



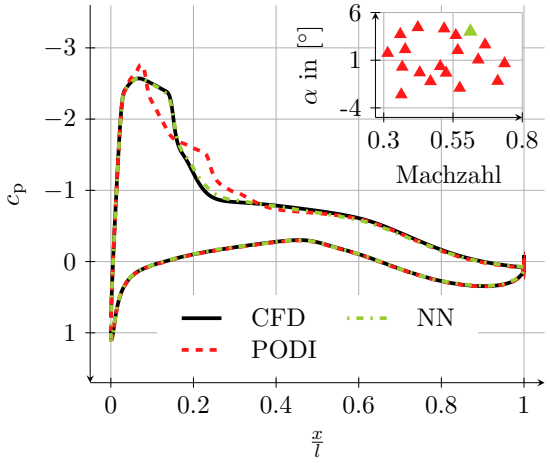
(a) $Ma = 0.31406$ und $\alpha = 1.74074$



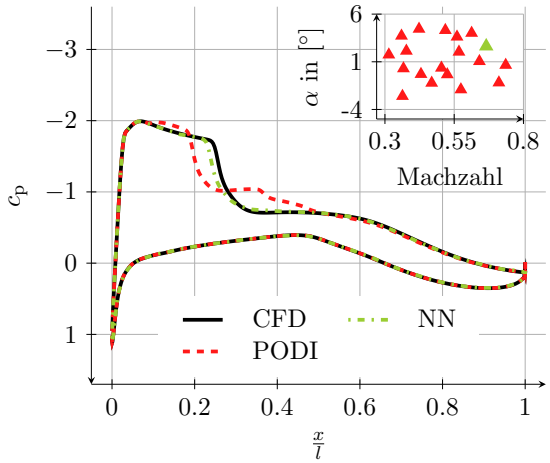
(b) $Ma = 0.36328$ und $\alpha = -2.60494$



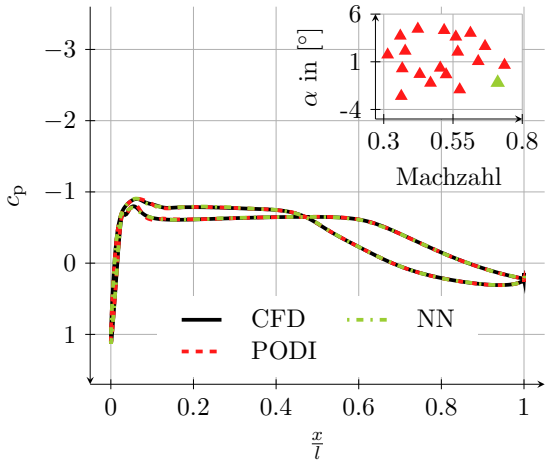
(c) $Ma = 0.51797$ $\alpha = 4.30864$



(d) $Ma = 0.61289$ und $\alpha = 4.01235$

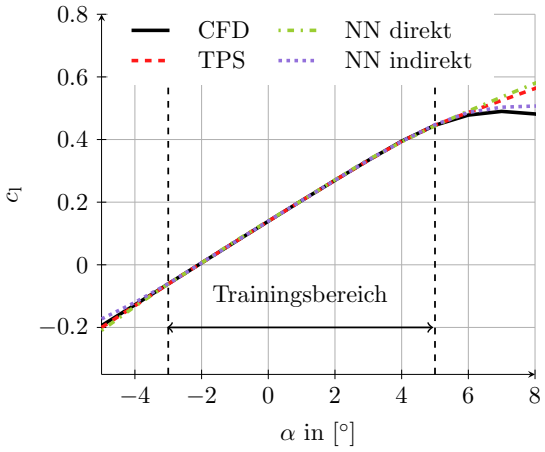


(e) $Ma = 0.66562$ und $\alpha = 2.62963$

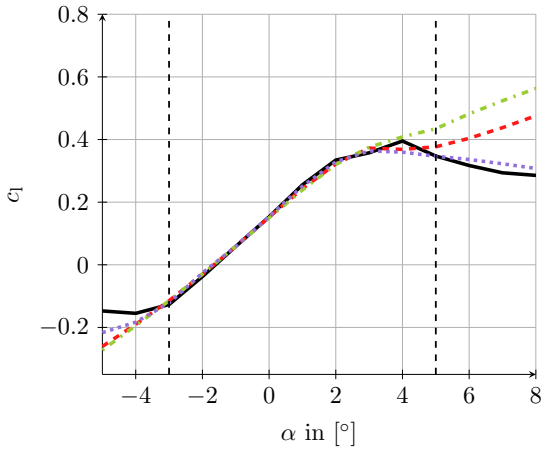


(f) $Ma = 0.71133$ und $\alpha = -1.1893$

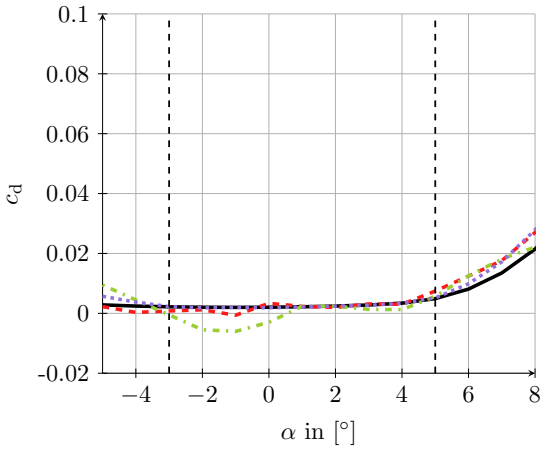
Abbildung 3.4: Fallbeispiel 1: Druckbeiwertsvorhersagen des mit $input_B$ trainierten neuronalen Netzes, deren Hyperparameter mit der Bayesian-Optimierung bestimmt sind, sowie des klassischen Ersatzmodells POD+TPS für ausgewählte Machzahl-Anstellwinkel-Kombinationen



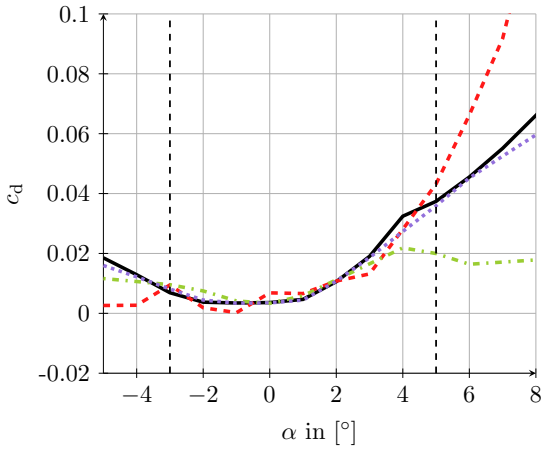
(a) Auftriebsbeiwert für $Ma = 0.36$



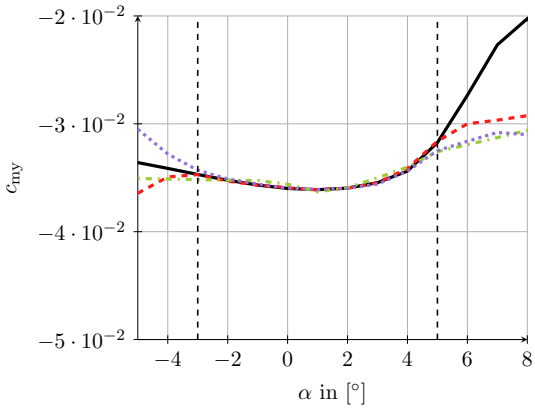
(b) Auftriebsbeiwert für $Ma = 0.58$



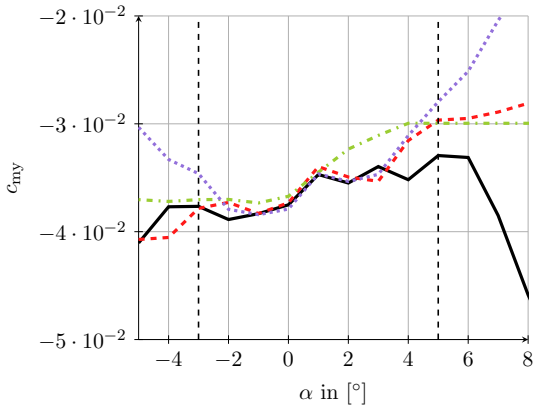
(c) Widerstandsbeiwert für $Ma = 0.36$



(d) Widerstandsbeiwert für $Ma = 0.58$



(e) Nickmomentenbeiwert für $Ma = 0.36$



(f) Nickmomentenbeiwert für $Ma = 0.58$

Abbildung 3.5: Fallbeispiel 1: Vergleich der skalaren Beiwerte aus indirektem Ansatz durch Integration der vorhergesagten Druckverteilung, durch direkte Vorhersagen der jeweiligen neuronale Netze und durch das TPS-Ersatzmodell

3.2 Fallbeispiel 2: Instationäre 2D-Profilumströmung

In diesem Kapitel wird eine zweidimensionale instationäre Umströmung des gleichen Flügelprofils wie in Kapitel 3.1, dem NLR7301, untersucht. Die Strömung zeichnet sich durch eine konstante Machzahl von $Ma = 0.5$ aus und ist dementsprechend ausschließlich subsonisch.
Das Ziel ist es, den zeitlichen Verlauf des Druckbeiwerts auf der Oberfläche des Profils, der sich abhängig von dem aufgeprägten, zeitlich variablen Anstellwinkel ergibt, mit einem neuronalen Netz und einem klassischen Ersatzmodell zu bestimmmen und das Potential des neuronalen Netzes durch einen Vergleich der Ergebnisse zu bewerten.

3.2.1 Methodik

Aufbauend auf den Untersuchungen des ersten Fallbeispiels werden die Daten mit der Methode B aufbereitet und die Hyperparameteroptimierung durch die Bayesian-Optimierung durchgeführt.
Das instationäre Strömungsverhalten wird durch zeitlich voneinander abhängige Daten beschrieben, weshalb das LSTM-basierte RNN verwendet wird. Außerdem ergeben sich durch die Zeitabhängigkeit weitere Möglichkeiten zur Aufbereitung der Daten. Es werden drei mögliche Methoden der Datenaufbereitung, die alle auf Methode B basieren, definiert.
Analog zu der Methodenuntersuchung in Fallbeispiel 1 wird für jede Methode ein neuronales Netz erstellt, deren Hyperparameter mit der Bayesian-Optimierung gefunden werden. Anschließend wird die beste Methode identifiziert, indem die Ergebnisse der auf den verschiedenen Methoden basierenden neuronalen Netze verglichen werden.

Abschließend wird das beste neuronale Netz, das auf der zuvor als beste identifizierte Datenaufbereitungsmethode beruht und dessen Hyperparameter durch die Bayesian-Optimierung gefunden werden, mit dem klassischen Ersatzmodell anhand der definierten Kriterien bewertet. Als solches wird wieder die POD-Dimensionsreduzierung mit nachgeschalteter TPS-Interpolation verwendet.

3.2.1.1 Beschreibung der Daten

Alle verwendeten Daten dieses Fallbeispiels unterliegen einer Machzahl von $Ma = 0.5$, demnach handelt es sich ausschließlich um den subsonischen Bereich. Der Anstellwinkel des Profils ändert sich mit der Zeit, wodurch die Umströmung des Profils instationär ist und ein völlig neues Problem vorliegt. Deshalb macht es Sinn dieses Fallbeispiel zu untersuchen, obwohl in dem ersten Fallbeispiel festgestellt wurde, dass sich neuronale Netze für Probleme im ausschließlich subsonischen Bereich nicht lohnen.

Für die Trainingsdaten wird die Änderung des Anstellwinkels durch eine lineare Chirp-Schwingung um den mittleren Anstellwinkel $\alpha_0 = 0.4°$ beschrieben, welche sich durch eine kontinuierliche lineare Frequenzänderung auszeichnet. Abbildung 3.6 zeigt sowohl den zeitlichen Verlauf des Anstellwinkels

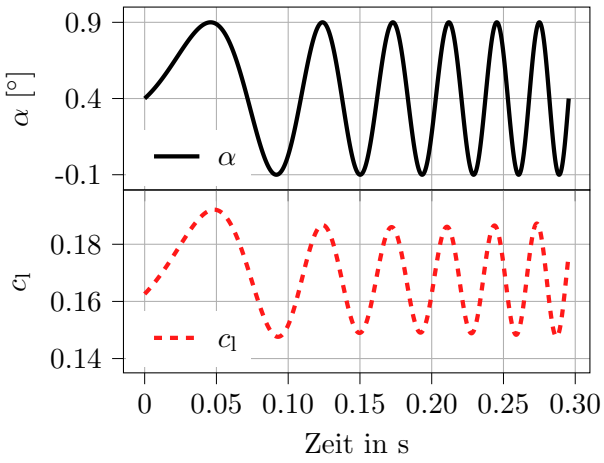


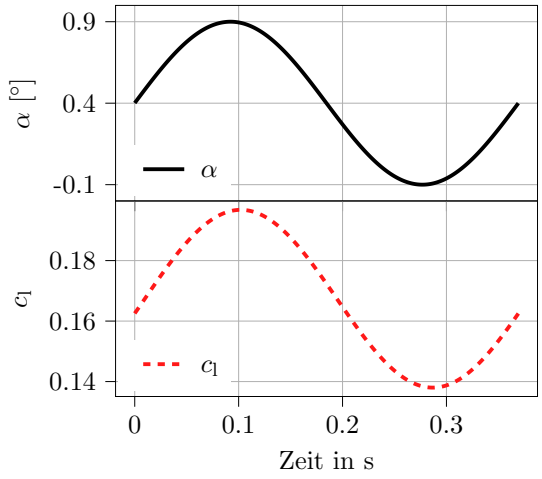
Abbildung 3.6: Fallbeispiel 2: Zeitlicher Verlauf des Anstellwinkels des Trainingssignals und zugehöriger zeitlicher Verlauf des Auftriebsbeiwerts, der für jeden Zeitschritt integrativ aus dem Druckbeiwert ohne Berücksichtigung des Wandreibungsanteils berechnet ist

als auch den sich ergebenden zeitlichen Verlauf des Auftriebsbeiwerts, der auch in diesem Fallbeispiel integrativ aus der Druckverteilung ohne Berücksichtigung des Wandreibungsanteils berechnet ist.

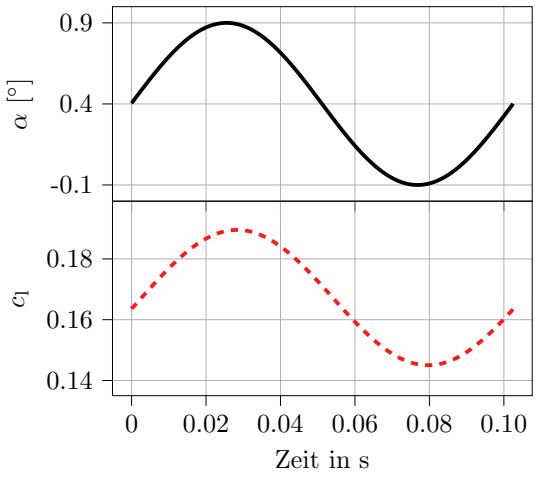
Die dimensionslose reduzierte Frequenz, die in (3.12) definiert ist, wobei c die Sehnenlänge des Profils und u_∞ die Anströmgeschwindigkeit ist, ändert sich beim Trainingssignal zeitlich von 0.05 bis 0.7. Das entspricht den Frequenzen von $f = 2.708$Hz bis 37.91Hz. Die Amplitude des Trainingssignals ist konstant $\hat{A} = 0.5°$ und die Zeitschrittweite beträgt $\Delta t = 1.319 \cdot 10^{-4}$s, welche so gewählt ist, dass eine Periode der harmonischen Schwingung mit der reduzierten Frequenz von 0.7, also der höchsten im linearen Chirp-Signal vorkommenden Frequenz, durch $T = 200$ Zeitschritte beschrieben wird.

$$k = \frac{c\pi f}{u_\infty} \tag{3.12}$$

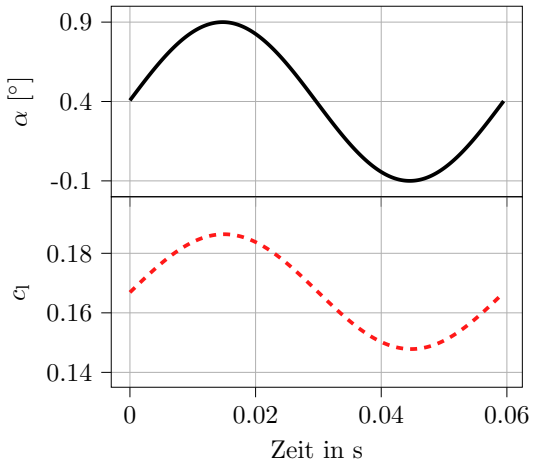
Bei den Testdaten werden die Änderungen des Anstellwinkels als harmonische Schwingungen beschrieben. Es liegen sechs Testsignale mit unterschiedlichen reduzierten Frequenzen von $k = 0.05$ bis 0.7 vor, wobei die Amplitude bei allen Testsignalen gleich der Amplitude des Trainingssignals $\hat{A} = 0.5°$ ist. Die Zeitschrittweite ist für alle Testsignale identisch mit der des Trainingssignals, was eine essentielle Voraussetzung für sinnvolle Vorhersagen mit dem neuronalen Netz ist. Das liegt daran, dass den LSTM-basierten RNNs keine explizite Information über die Weite des Zeitschritts vorliegt. Vorhersagen für Signale mit abweichenden Zeitschrittweiten sind folglich verzerrt und nicht sinnvoll. Die Eigenschaften der einzelnen Signale sind in Tabelle 3.8 aufgelistet und der zeitliche Anstellwinkel- mit resultierendem Auftriebsbeiwertsverlauf der Testsignale in Abbildung 3.7 dargestellt.



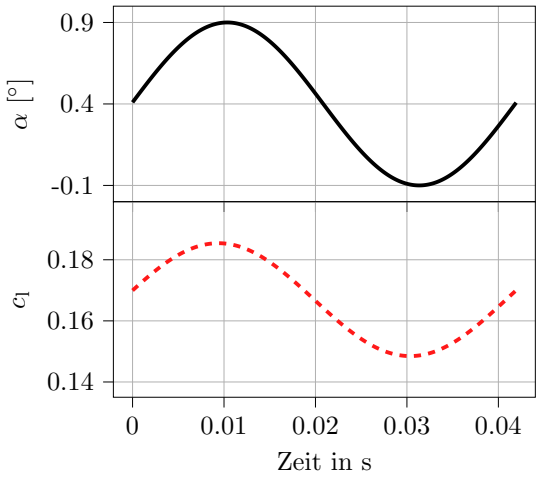
(a) Testsignal 0



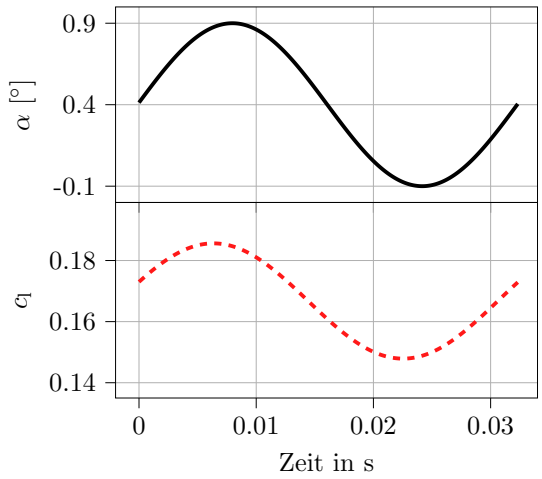
(b) Testsignal 1



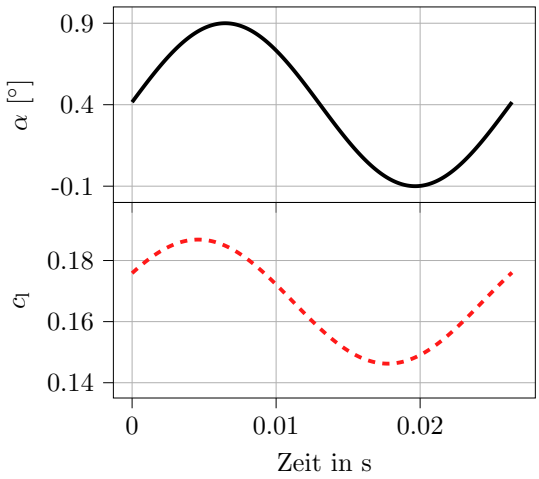
(c) Testsignal 2



(d) Testsignal 3



(e) Testsignal 4



(f) Testsignal 5

Abbildung 3.7: Fallbeispiel 2: Zeitlicher Verlauf des Anstellwinkels der Testsignale und zugehöriger zeitlicher Verlauf des Auftriebsbeiwerts, der für jeden Zeitschritt integrativ aus dem Druckbeiwert ohne Berücksichtigung des Wandreibungsanteils berechnet ist

Signal	α_0[°]	**Â**[°]	**k**	**f**[Hz]	**Zeitschrittanzahl T**
Training	0.4	0.5	0.05-0.7	2.71-37.91	2400
Test 0	0.4	0.5	0.05	2.71	2800
Test 1	0.4	0.5	0.18	9.75	778
Test 2	0.4	0.5	0.31	16.79	452
Test 3	0.4	0.5	0.44	23.83	319
Test 4	0.4	0.5	0.57	30.87	246
Test 5	0.4	0.5	0.70	37.91	200

Tabelle 3.8: Fallbeispiel 2: Eigenschaften des Trainingssignals und der Testsignale

Das Trainingssignal und die Testsignale sind ebenfalls in Abbildung 3.8 als Anstellwinkeländerung über dem Anstellwinkel dargestellt. Durch diese Darstellung wird die resultierende Dichte der abgebildeten Frequenzen des Trainingssignals, die sich durch die zeitliche Frequenzänderung ergibt, und die Einordnung der Testsignale in das Frequenzspektrum des Trainingssignals ersichtlich.

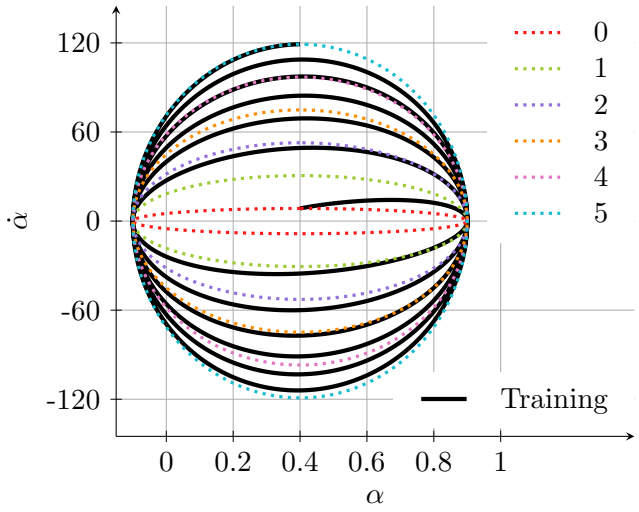


Abbildung 3.8: Fallbeispiel 2: Frequenzdichtediagramm des Trainingssignals und der Testsignale

Eine vollständige Periode einer harmonischen Schwingung mit einer bestimmten Frequenz wird in der Abbildung als eine geschlossene Ellipse dargestellt, wie es für die Testsignale der Fall ist. Es ist zu erkennen, dass kein vollständiges Testsignal durch das Trainingssignal abgebildet wird. Das Testsignal 0 wird am schlechtesten repräsentiert und liegt größtenteils sogar außerhalb des Trainingsbereichs, wodurch die Vorhersagen der Ersatzmodelle einer Extrapolation unterliegen und dies nach den Erkenntnissen des ersten Fallbeispiels höchstwahrscheinlich zu ungenauen Ergebnissen führen wird. Es ist weiterhin zu erkennen, dass die Frequenzdichte zu höheren Frequenzen hin steigt und das Testsignal 5 ebenfalls nicht besonders gut durch das Trainingssignal repräsentiert wird. Somit sind die Vorhersagen teilweise extrapolativ. Aufgrund dieser frequenzabhängigen Informationsdichte des Trainingssignals ist zu erwarten, dass die Vorhersagen der Ersatzmodelle für den mittleren Frequenzbereich präziser sind als für die Testsignale 0, 1 und 5.

3.2.1.2 Datenaufbereitung

Auch in diesem Fallbeispiel werden für das neuronale Netz alle Eingangsgrößen der Daten auf den Bereich [-1, 1] skaliert. Die in Kapitel 3.1.1.2 beschriebene Methode B der Datenaufbereitung, bei der die Oberflächenkoordinaten in kartesischer Darstellung in den Eingangsgrößen für die neuronalen Netze enthalten sind und der Druckbeiwert für jeden Oberflächenpunkt einzeln vorhergesagt wird, wird als Fundament der Datenaufbereitung für dieses Fallbeispiel verwendet.

Weitere Möglichkeiten der Datenaufbereitung ergeben sich durch die Tatsache, dass das instationäre Verhalten durch die zeitlichen Verläufe des Anstellwinkels beschrieben wird, dessen zugrunde liegenden Funktionen sowohl für das Trainingssignals als auch für die Testsignale bekannt, stetig und stetig differenzierbar sind und folglich zeitliche Ableitungen des Anstellwinkels berechnet werden können. Die Idee ist es, die zeitlichenen Verläufe dieser Ableitungen, welche zusätzliche Informationen über das physikalische Problem bieten, als weitere Eingangsgrößen zu berücksichtigen, um die Vorhersagegenauigkeit zu steigern.
Hierfür werden in (3.13) - (3.15) die auf einen Zeitschritt bezogenen *inputs* definiert, die die drei unterschiedlichen Datenaufbereitungsmethoden dieses Fallbeispiels repräsentieren. Bei $input_0$ sind die Eingangsgrößen pro Zeitschritt identisch mit denen der Datenaufbereitungsmethode B des ersten Fallbeispiels. Beim $input_1$ wird zusätzlich die erste Ableitung - die Anstellwinkelgeschwindigkeit - als weitere Eingangsgrößen pro Zeitschritt berücksichtigt, während für $input_2$ sowohl die erste, als auch die zweite Ableitung - die Anstellwinkelbeschleunigung - berücksichtigt wird. Der *output* bezogen auf einen Zeitschritt ist für die drei *inputs* gleich und entspricht dem zugehörigen Druckbeiwert des Oberflächenpunktes n nach (3.16).

$$input_0 = (\alpha, (\frac{x}{l})_{\mathrm{n}}, (\frac{z}{l})_{\mathrm{n}}) \tag{3.13}$$

$$input_1 = (\alpha, \dot{\alpha}, (\frac{x}{l})_{\mathrm{n}}, (\frac{z}{l})_{\mathrm{n}}) \tag{3.14}$$

$$input_2 = (\alpha, \dot{\alpha}, \ddot{\alpha}, (\frac{x}{l})_{\mathrm{n}}, (\frac{z}{l})_{\mathrm{n}}) \tag{3.15}$$

$$output = (c_{\mathrm{p_n}}) \tag{3.16}$$

mit $\mathrm{n} \in [1, \mathrm{N}]$, wobei das Profil wie im ersten Fallbeispiel mit $\mathrm{N} = 597$ Oberflächenpunkten beschrieben wird.

Bei den klassischen Ersatzmodellen ist es anders als bei den LSTM-basierten RNNs nicht möglich, einen Bezug zu zeitlich vorherigen Daten herzustellen. Für eine eindeutige Zuordnung der Eingangsgrößen zum Zeitschritt des Signals ist es eine Voraussetzung, mindestens den Anstellwinkel und die erste Ableitung in den Eingangsgrößen zu berücksichtigen. Für den Vergleich mit den neuronalen Netzen wird bei der Wahl der Eingangsgrößen für das klassische Ersatzmodell mindestens diese Voraussetzung erfüllt. Werden zusätzlich die in $input_2$ definierten Eingangsgrößen für das neuronale

Netz als Beste identifiziert, wird dem klassischen Ersatzmodell ebenfalls die zweite Ableitung als weitere Eingangsgröße zur Verfügung gestellt, um Vergleichbarkeit zu gewährleisten.

3.2.1.3 Entwurfsprozess des neuronalen Netzes

Für dieses Fallbeispiel wird ein LSTM-basiertes rekurrentes neuronales Netz verwendet, da sich dieses besonders gut für die Vorhersage zeitlich abhängiger Daten eignet. Das liegt daran, dass für die Berechnung der Ausgangsgrößen eines bestimmten Zeitschritts die Ausgangswerte vorheriger Zeitschritte berücksichtigt werden, wie in Kapitel 2.1.2.2 beschrieben.
Diese Tatsache bedingt jedoch äußerst ungenaue Vorhersagen der neuronalen Netze für die anfänglichen Zeitschritte eines Signals. Beim ersten Zeitschritt liegen dem neuronalen Netz keinerlei zeitlich zurückliegende Informationen vor, während es beispielsweise beim dritten Zeitschritt bereits die Informationen der ersten beiden Zeitschritte in der Vorhersage berücksichtigt. Die Vorhersagegenauigkeit steigt dadurch mit zunehmender Anzahl an Zeitschritten. Dieses Verhalten wird in Abbildung 3.9 deutlich, die beispielhaft Vorhersagen eines neuronalen Netzes für zwei Schwingungsperioden des Testsignals 3 zeigt.

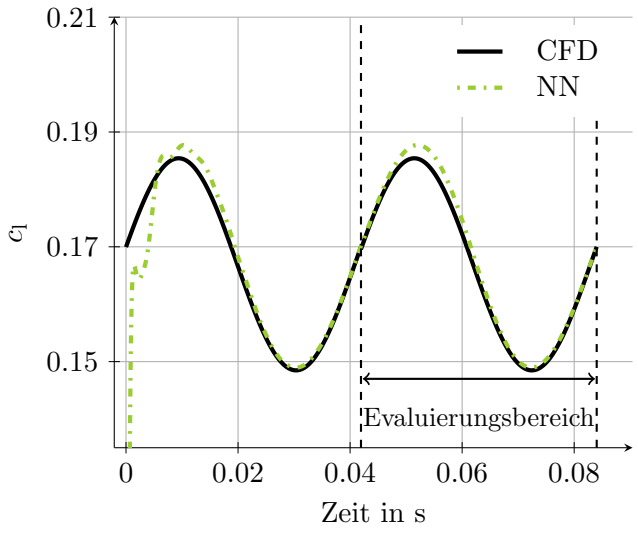


Abbildung 3.9: Fallbeispiel 2: Vorhersage eines neuronalen Netzes für zwei Schwingungsperioden des Testsignals 3

Dieses Problem kann allgemein behoben werden, indem das vorherzusehende Signal geklont wird, wie das Testsignal 3 in Abbildung 3.9. Dadurch werden die Daten verdoppelt und es werden in der Evaluierung lediglich die Vorhersagen der zweiten Hälfte des Signals berücksichtigt. In der Abbildung ist der Evaluierungsbereich für das geklonte Testsignal 3 beispielhaft gekennzeichnet. Im weiteren Verlauf dieses Kapitels wird diese Methode angewendet. Die dadurch entstehende Verdopplung der Vorhersagezeit der neuronalen Netze für ein Signal ist in den aufgeführten Tabellen ebenfalls berücksichtigt, sodass ein direkter Vergleich mit dem klassischen Ersatzmodell möglich ist.

Durch die unterschiedlichen Informationsmengen, welche in den verschiedenen *inputs* enthalten sind und den neuronalen Netzen zur Verfügung gestellt werden, sind die optimalen Hyperparameter für jede Variante höchstwahrscheinlich unterschiedlich. Daher ist es für einen aussagekräftigen Vergleich der *inputs* nötig, für jede dieser Varianten eine separate Hyperparameteroptimierung durchzuführen, die aufbauend auf den Ergebnissen von Fallbeispiel 1 mit der Bayesian-Optimierung durchgeführt werden. Die Optimierungsbereiche der Hyperparameter sind für die Netze der unterschiedlichen *inputs* gleich gewählt und aus vorläufigen manuellen Untersuchungen entstanden. Sie sind in Tabelle 3.9 aufgelistet.

Hyperparameter	**Optimierungsbereich**
Batchgröße bs	[16, 128]
Anfangslernrate lr_0	[0.001, 0.1]
Abnahmefaktor k	[1e-6, 1e-2]
Anzahl verdeckter Schichten	[1, 2]
Neuronenanzahl pro Schicht	[10, 100]

Tabelle 3.9: Fallbeispiel 2: In der Optimierung berücksichtigte Hyperparameter und deren zulässiger Bereich

Da die LSTM-Einheiten sehr komplex sind, reicht zur Lösung der meisten Probleme bereits eine einzelne verdeckte Schicht mit einer Neuronenanzahl im unteren zweistelligen Bereich.[88, 89] Für die Optimierung wurden dennoch zwei verdeckte Schichten und 100 Neuronen pro Schicht ermöglicht. Da das Training aufgrund der hohen Komplexität der LSTM-Einheiten und dadurch auch die Hyperparameteroptimierung äußerst lange dauert, werden lediglich 20 Iterationen während jeder Optimierung mit einer Anzahl von 20 Trainingsepochen durchgeführt. Außerdem wird die Dropoutrate in diesem Fallbeispiel in der Optimierung nicht berücksichtigt, sondern aufbauend auf den Ergebnissen des ersten Fallbeispiels zu Null gesetzt, um die Optimierungszeit zu verkürzen. Alle anderen eingestellten Hyperparameter sind während der Optimierung aufbauend auf den Ergebnissen der vorläufigen manuellen Untersuchungen und des ersten Fallbeispiels konstant gehalten und in Tabelle 3.10 aufgeführt. Obwohl die ReLU-Funktion generell vielversprechender ist als die tanh-Funktion, ist in diesem Fallbeispiel letztere aufgrund genaueren Ergebnissen in manuellen Voruntersuchungen als Aktivierungsfunktion gewählt. Alle nicht aufgeführten Hyperparameter unterliegen den Standardeinstellungen der Keras Bibliothek [80]. Die Lernrate ändert sich wie im ersten Fallbeispiel exponentiell nach (3.11).

Hyperparameter	**Einstellung**
Aktivierungsfunktion	tanh
Optimierer	Adam
Kostenfunktion	MSE
Dropoutrate	0.0

Tabelle 3.10: Fallbeispiel 2: Manuell eingestellte während der Optimierung konstante Hyperparameter

Anschließend werden drei neuronale Netze mit den jeweils besten Hyperparametern, die durch die jeweilige Optimierung gefunden wurden, mit 500 Epochen trainiert. Die Vorhersageergebnisse der drei neuronalen Netze werden anschließend verglichen, um die besten *inputs* zu identifizieren.

Aufbauend auf dem Ergebnis wird ein neuronales Netz basierend auf den besten *inputs*, dessen Hyperparameter mit der Bayesian-Optimierung gefunden wurden, für den finalen Vergleich mit dem klassischen Ersatzmodell mit maximal 3500 Epochen trainiert.

3.2.2 Ergebnisse und Diskussion

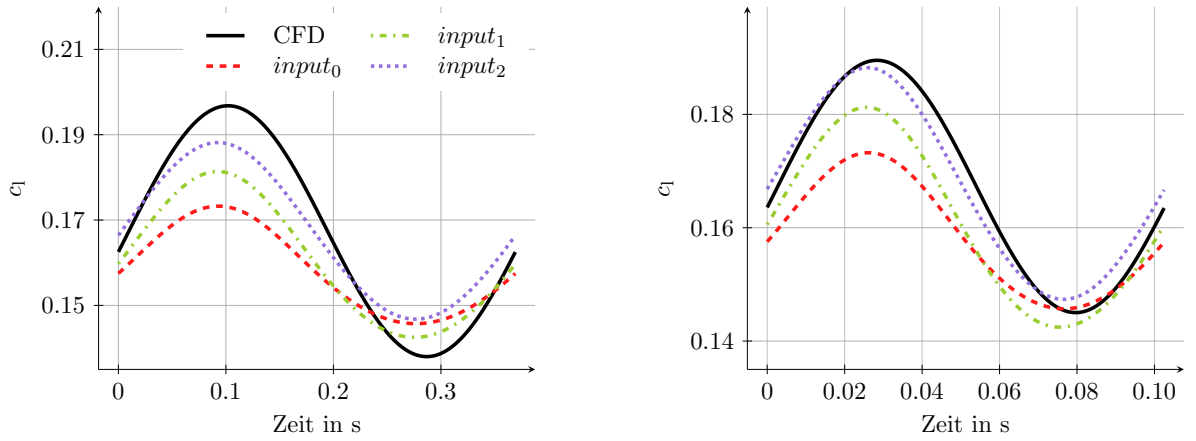
Die durch die Bayesian-Optimierung gefundenen Hyperparameter der neuronalen Netze basierend auf den drei verschiedenen *inputs* sind in Tabelle 3.11 aufgelistet.

Hyperparameter	$\mathbf{input}_0$	$\mathbf{input}_1$	$\mathbf{input}_2$
Batchgröße bs	58	102	72
Anfangslernrate lr_0	0.001	0.001	0.001
Abnahmefaktor k	1e-4	1e-4	1e-4
Anzahl verdeckter Schichten	2	1	2
Neuronenanzahl pro Schicht	95, 69	100	71, 57

Tabelle 3.11: Fallbeispiel 2: Ergebnisse der durch die Bayesian-Optimierung gefundenen besten Hyperparameter für die unterschiedlichen *inputs*

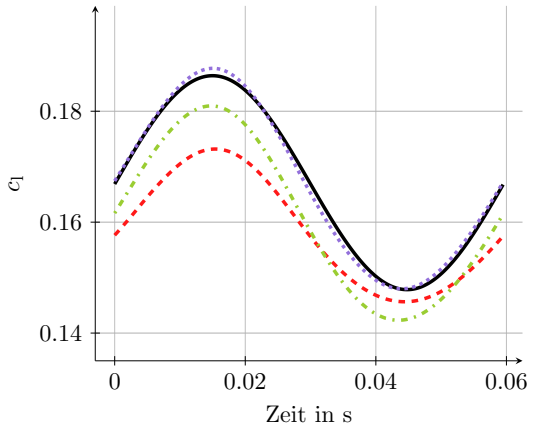
Wider der Erwartungen führen die geeignetsten Hyperparameter für den $input_0$ zu einem Netz mit größerer Kapazität, als für $input_1$ bzw. $input_2$. Eine mögliche Erklärung dafür wäre, dass es dem neuronalen Netz basierend auf der geringeren Informationsmenge des $input_0$ nur durch eine Kapazitätserhöhung möglich ist, die nicht-lineare Beziehung zu lernen. Entgegen dieser Erklärung könnte es ein Indiz dafür sein, dass die Optimierung mit einer höheren Iterations- und Epochenanzahl durchgeführt werden sollte, um die erwarteten Ergebnisse zu erhalten. Im Rahmen dieser Arbeit wird aufgrund begrenzter zeitlicher Ressourcen von der beschriebenen erweiterten Optimierung abgesehen. Diese ist als weitere Untersuchungen für die Erklärung des Ergebnisses erforderlich.
Die Vorhersagegenauigkeit der auf den verschiedenen *inputs* basierenden neuronalen Netze ist gemessen anhand des MAE und des MSE in Tabelle 3.12 gezeigt. Die Abweichungen hierbei sind nicht auf die integrativ berechneten Auftriebsbeiwerte, sondern direkt auf die vorhergesagten Druckverteilungen bezogen. Dies ist eine genauere Auswertung der Vorhersagen der Druckverteilung, in der vermieden wird, dass sich durch die Integration der Druckverteilung positive mit negativen Abweichungen aufheben und sich dadurch die Vorhersagegenauigkeit verzerrt. Die integrative Berechnung des Auftriebsbeiwerts dient ausschließlich der indirekten visuellen Darstellung der Vorhersageergebnisse der Druckverteilung über der Zeit. Diese sind für die drei neuronalen Netze mit unterschiedlichen *inputs* für alle Testsignale in Abbildung 3.10 dargestellt.

Das neuronale Netz mit $input_2$, in der die erste und zweite Ableitung des Anstellwinkels als Eingangsgrößen berücksichtigt sind, hat mit einer mittleren absoluten Abweichung von $MAE = 8.86 \cdot 10^{-3}$ entsprechend Tabelle 3.12 das genauste Ergebnis erzielt, während das neuronale Netz

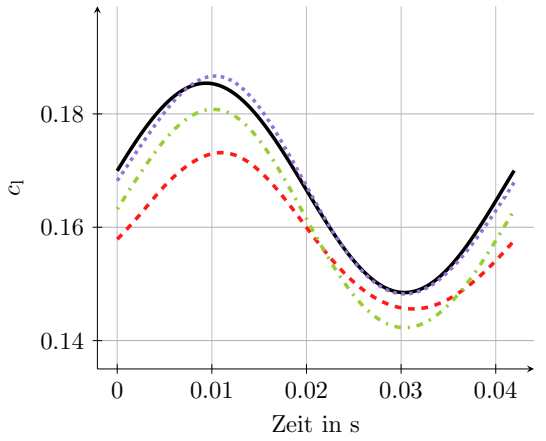


(a) Testsignal 0

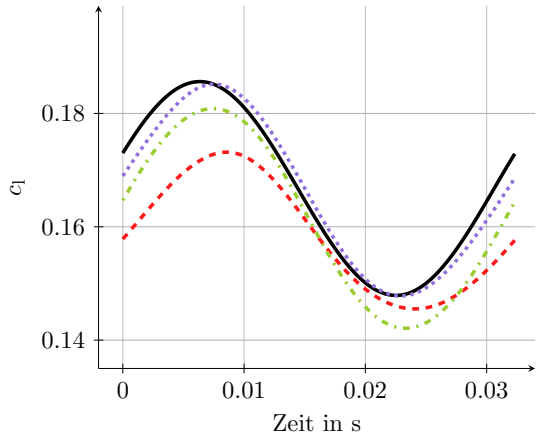
(b) Testsignal 1



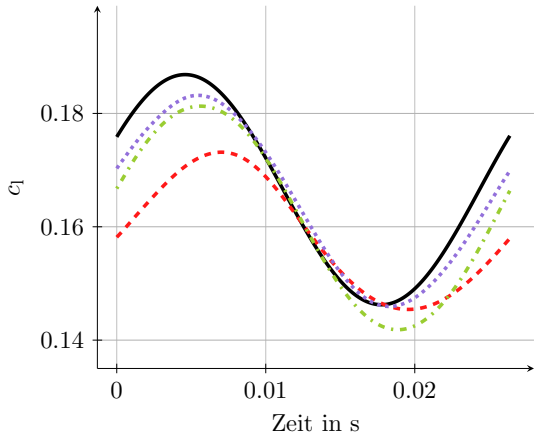
(c) Testsignal 2



(d) Testsignal 3



(e) Testsignal 4



(f) Testsignal 5

Abbildung 3.10: Fallbeispiel 2: Zeitlicher Verlauf des integrativ aus vorhergesagter Druckverteilung berechneten Auftriebsbeiwerts für alle Testsignale für die drei mit 500 Epochen trainierten neuronalen Netze, die auf den verschiedenen *inputs* basieren

mit $input_0$ die ungenausten Vorhersagen mit einer knapp 6-mal so großen Abweichung von $MAE = 59.67 \cdot 10^{-3}$ erzielt und die $input_1$ Methode mit einem $MAE = 14.73 \cdot 10^{-3}$ dazwischen liegt. Dies spiegelt sich auch in der Abbildung 3.10 wieder. Außerdem ist der zeitliche Aufwand für das Training

inputs	**MAE** $[\cdot\mathbf{10^{-3}}]$	**MSE** $[\cdot\mathbf{10^{-3}}]$	**Trainingszeit [h]**	**Vorhersagezeit [s]**
$input_0$	59.67	14.02	6.67	0.0021
$input_1$	14.73	0.60	1.77	0.0007
$input_2$	8.86	0.23	5.5	0.0018

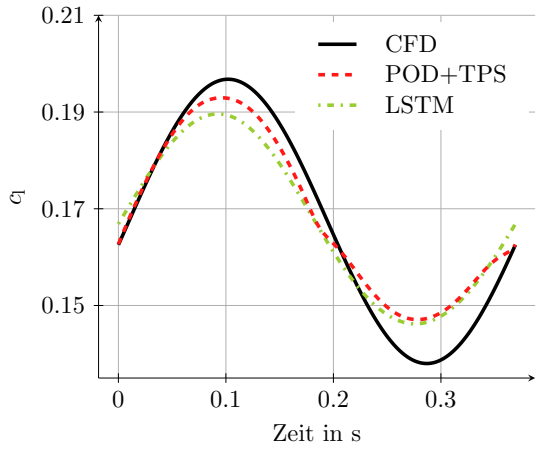
Tabelle 3.12: Fallbeispiel 2: Vorhersagefehler der Druckverteilung gemessen als MAE und MSE der mit 500 Epochen trainierten neuronalen Netze mit unterschiedlichen *inputs* für alle Tests gemittelt und zeitlicher Rechenaufwand unterteilt als Trainings- und Vorhersagezeit für eine vollständige Druckverteilung zu einem Zeitschritt

und die Vorhersagen trotz größerer Anzahl an Eingangsgrößen für das neuronale Netz mit $input_2$ geringer als mit $input_0$, was daran liegt, dass die als beste identifizierte Kapazität des neuronalen Netzes für $input_2$ ebenfalls geringer ist. Das neuronale Netz mit $input_1$ weist die geringste Kapazität und dementsprechend den mit Abstand geringsten zeitlichen Rechenaufwand auf.

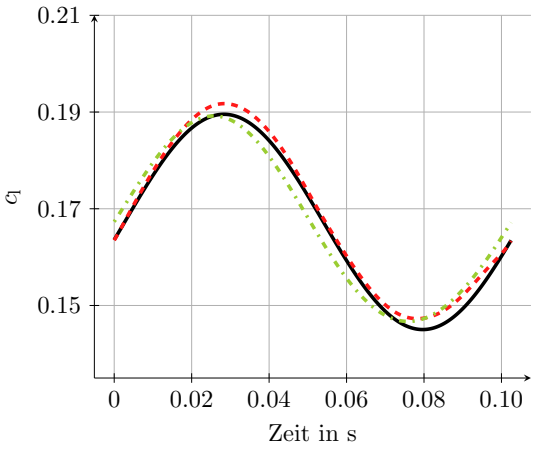
Zusammenfassend lässt sich für die Untersuchung der verschiedenen *inputs* sagen, dass das neuronale Netz mit $input_2$ mit einem knapp 3-fach höheren zeitlichen Rechenaufwand als mit $input_1$ zu einer deutlich höheren und der insgesamt besten Vorhersagegenauigkeit der drei *inputs* führt, weshalb $input_2$ als bester bewertet wird. Für die Potentialanalyse der neuronalen Netze wird den Ergebnissen entsprechend das neuronale Netz mit $input_2$ anhand der definierten Kriterien mit dem klassischen Ersatzmodell verglichen.

Zusätzlich lässt sich - wie dieses Fallbeispiel zeigt - die allgemeine hypothetische Aussage treffen, dass eine Steigerung der Informationen, die in den Eingangsgrößen enthalten sind, zu einer Steigerung der Vorhersagegenauigkeit führt. Hierdurch ergeben sich weitere Untersuchungsmöglichkeiten. Das Verhaltens der Vorhersagegenauigkeit der neuronalen Netze in Abhängigkeit von der Anzahl an in dem Eingangsvektor enthaltenen Ableitungen könnte untersucht werden, indem sukzessiv weitere Ableitungen höherer Ordnung als zusätzliche Eingangsgrößen definiert werden. Hierbei ist zu beachten, dass die Informationen der Ableitungen verschieden hoher Ordnungen durch die Skalierung nicht redundant sind.
In diesem Fallbeispiel sind die Testsignale harmonische Schwingungen, bei denen der zeitliche Verlauf der $(n+4)$-ten Ableitung die gleiche Phasenverschiebung wie die n-te Ableitung für $n \in \mathbb{N}_0$ aufweist. Die Informationen dieser Ableitungen sind durch die in diesem Fallbeispiel angewendete Skalierung der Eingangsgrößen auf den gleichen Bereich identisch.
Diese Untersuchung ist dementsprechend abhängig von den Signaleigenschaften nur begrenzt möglich, aber dennoch interessant.

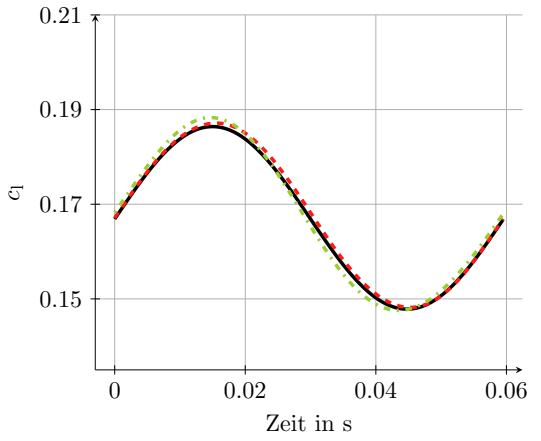
Die Vorhersageergebnisse der Druckverteilung des $input_2$ basierten, mit 3327 Epochen trainierten neuronalen Netzes, dessen Hyperparameter mit der Bayesian-Optimierung gefunden wurden, und des klassischen POD+TPS-Ersatzmodells ist in Abbildung 3.11 als zeitlicher Verlauf des integrativ berechneten Auftriebsbeiwerts für alle Testsignale dargestellt. Es ist in Abbildung 3.11a ersichtlich, dass die POD+TPS-Vorhersagen genauer sind, als die des neuronalen Netzes. Aus Tabelle 3.13 wird dies für alle Testsignale durch die geringeren Abweichungen der vorhergesagten Druckverteilungen von den CFD-Lösungen gemessen als MAE bzw. MSE ebenfalls bestätigt.



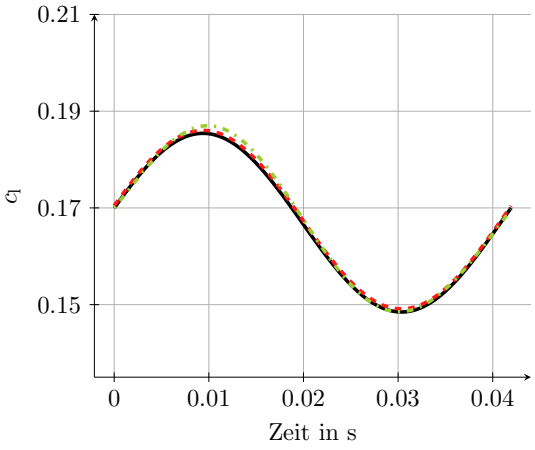
(a) Testsignal 0



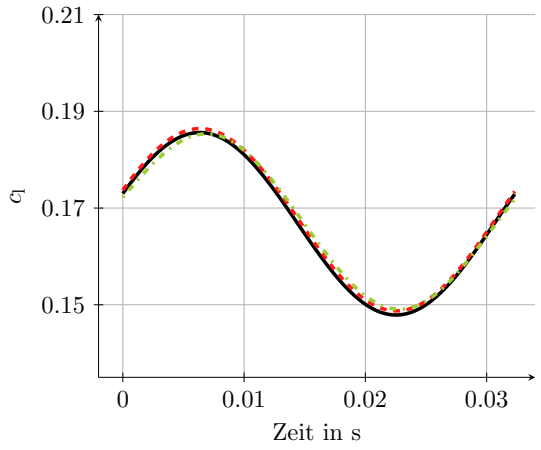
(b) Testsignal 1



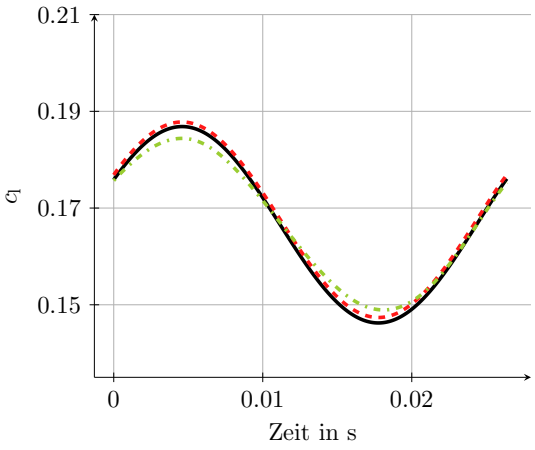
(c) Testsignal 2



(d) Testsignal 3



(e) Testsignal 4



(f) Testsignal 5

Abbildung 3.11: Fallbeispiel 2: Zeitlicher Verlauf des integrativ aus der Druckverteilung berechneten Auftriebsbeiwerts für alle Testsignale. Die Druckverteilungen stammen dabei jeweils aus der CFD-Lösung, den Vorhersagen des mit 3327 Epochen trainierten $input_2$ basierten neuronalen Netzes und den Vorhersagen des klassischen POD+TPS-Ersatzmodells

Ersatzmodell	**Testsignal**	**MAE** $[\cdot\mathbf{10^{-3}}]$	**MSE** $[\cdot\mathbf{10^{-3}}]$
NN	0	7.51	0.17
	1	4.88	0.10
	2	3.81	0.09
	3	3.69	0.08
	4	4.06	0.10
	5	5.55	0.14
	gemittelt	4.92	0.11
POD+TPS	0	4.83	0.0659
	1	1.76	0.0072
	2	0.89	0.0026
	3	0.82	0.0020
	4	1.04	0.0027
	5	1.26	0.0035
	gemittelt	1.77	0.0140

Tabelle 3.13: Fallbeispiel 2: Vorhersagegenauigkeiten für die Druckverteilung für jedes Testsignal einzeln und gemittelt gemessen am MAE und MSE für das mit 3327 Epochen trainierte $input_2$ basierte neuronale Netz und das klassische POD+TPS-Ersatzmodell

Es wird außerdem sehr deutlich, dass die Vorhersagegenauigkeit beider Ersatzmodelle für das Testsignal 0 am geringsten ist. Diese steigt stetig an bis einschließlich Testsignal 3, das durch beide Ersatzmodelle am besten vorhergesagt wird. Anschließend steigt die Abweichung der Vorhersagen bis einschließlich zum Testsignal 5 wieder an. Die Ergebnisse entsprechen exakt der in Kapitel 3.2.1.1 formulierten Erwartungen, die aufgrund der frequenzabhängigen Informationsdichte des Trainingssignals (Abbildung 3.8) aufgestellt wurden.

Da die Bewertung der Ersatzmodelle anhand der Vorhersagen der Druckverteilung erfolgen und aus dieser detailliertere Informationen hervorgehen können, ist diese beispielhaft für denjenigen Zeitschritt jedes Testsignals, bei dem der MAE der Vorhersagen des neuronalen Netzes am größten ist, in Abbildung 3.12 dargestellt. Die Einordnung des Zeitschritts ins gesamte Testsignal wird jeweils oben rechts gezeigt. Es wird deutlich, dass die Abweichungen der vorhergesagten einzelnen Druckbeiwerte an den Oberflächenpunkten von der CFD-Lösung für beide Ersatzmodelle so gering sind, dass sie in der abgebildeten Druckverteilung nicht direkt erkennbar sind. Erst durch die Integration des Druckbeiwerts zur Bestimmung des Auftriebsbeiwerts, wodurch die Abweichungen gewissermaßen aufsummiert werden, und durch die sehr geringe Skalierung der y-Achse wird die Abweichung in Abbildung 3.11 deutlich. Daraus folgt, dass sich in Hinblick auf die sehr hohe Vorhersagegenauigkeit der Druckverteilung beider Ersatzmodelle für dieses Fallbeispiel gut eignen. Der Tabelle 3.14 ist zu entnehmen, dass der zeitliche Rechenaufwand des Trainings des besten neuronalen Netzes mit 36.47 Stunden knapp 1.5-mal so groß ist wie der des POD+TPS-Ersatzmodells, wohingegen die Vorhersagezeit für die Druckverteilung eines Zeitschritts im Durchschnitt deutlich geringer ist. Der extrem viel größere Zeitaufwand zur Erstellung des klassischen Ersatzmodells in diesem Fallbeispiel verglichen mit dem Zeitaufwand in Fallbeispiel 1 liegt größtenteils an dem enormen Unterschied der Datenmengen und der zeitlichen Skalierung der POD-Methode mit $\mathcal{O}(Q^2)$, wobei Q die Anzahl an

Snapshots beschreibt.[90] Da in diesem Fallbeispiel für jeden Zeitschritt ein Snapshot vorliegt, wird das Trainingssignal dementsprechend durch 2400 Snapshots beschrieben, während nur 79 Snapshots im ersten Fallbeispiel vorliegen. Aufgrund der quadratischen zeitlichen Skalierung abhängig von der Anzahl der Snapshots ergibt sich hierdurch der deutliche Unterschied des Zeitaufwands der POD-Methode für Fallbeispiel 1 und 2.

Wie bereits erwähnt, wird bei der Vorhersage des klassischen Ersatzmodelles für einen bestimmten Zeitschritt keine Beziehung zu Ergebnissen anderer Zeitschritte berücksichtigt. Hierdurch ergibt sich der Vorteil, dass mit dem klassischen Ersatzmodell für einen einzigen dezidierten Zeitschritt eine - im Rahmen der in Tab 3.13 beschriebenen Vorhersagefehler - akkurate Vorhersage erzielt werden kann. Außerdem können hierdurch Vorhersagen für Testsignale mit Zeitschrittweiten berechnet werden, die vom Trainingssignal abweichen.
Die Vorhersage für einen einzelnen Zeitschritt oder für Signale mit nicht identischer Zeitschrittweite führt mit dem neuronalen Netz hingegen zu unbrauchbaren Ergebnissen.

Ersatzmodell	**Trainingszeit [h]**	**Vorhersagezeit [s]**
NN	36.47	0.002
POD+TPS	24.39	0.534

Tabelle 3.14: Fallbeispiel 2: Zeitlicher Rechenaufwand des mit 3327 Epochen trainierte $input_2$ basierte neuronalen Netzes und des klassischen POD+TPS-Ersatzmodells unterteilt als Trainings- und Vorhersagezeit für die Druckverteilung eines Zeitschritts

3.2.3 Fazit

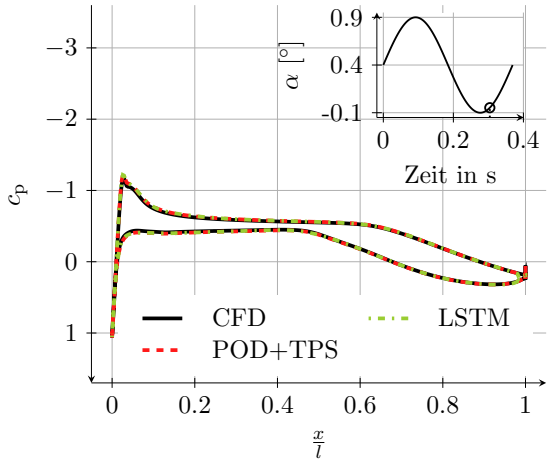
Sowohl die klassischen Ersatzmodelle als auch die LSTM-basierten RNNs können die Abhängig von zeitlicher Anstellwinkeländerung und sich ergebender zeitlich ändernder Druckverteilung erlernen und für nie zuvor vorliegende Anstellwinkeländerung sehr genau vorhersagen. Hierbei liefert das klassische Ersatzmodell genauere Ergebnisse bei geringerem zeitlichen Aufwand, weshalb das POD+TPS-Ersatzmodell sich insgesamt für dieses spezielle Fallbeispiel besser eignet. Für das neuronale Netz eignet es sich am besten, die Daten als $input_2$ aufzubereiten. Die Möglichkeit, die Druckverteilung für einzelne Zeitschritte oder Signale mit Zeitschrittweiten, die vom Trainingssignal abweichen, vorherzusagen spricht ebenfalls für die Verwendung des klassischen POD+TPS-Ersatzmodells.

Wie sich bereits durch die Untersuchungen im ersten Fallbeispiel, bei der die Eignung der neuronalen Netze zur Vorhersage der relativ einfachen skalaren Beiwerte bewertet wurde, herausgestellt hat, lohnt sich der zeitliche Aufwand der neuronalen Netze nicht, um einfache Beziehungen zwischen *input* und *output* zu erlernen. Dies wurde ebenfalls durch einen Vergleich der Vorhersagen zwischen subsonischem (einfachen) und transsonischem (komplexen) Bereich ersichtlich. Für beide Bereiche war das klassische Ersatzmodell mit deutlich geringerem zeitlichen Rechenaufwand verbunden, jedoch lohnte sich der größere zeitliche Aufwand der neuronalen Netze für den komplexen Bereich, da ihre Vorhersagen dort deutlich genauer waren. Dieses zweite aerodynamische Fallbeispiel

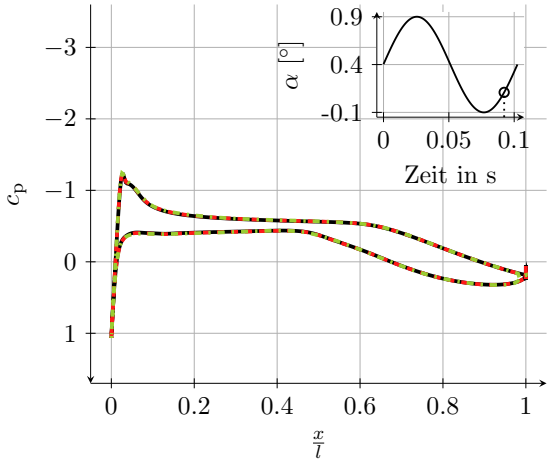
ist aus folgenden Gründen möglicherweise zu einfach, als dass das Potential der neuronalen Netze ausgeschöpft wird und diese sich hier als Ersatzmodell lohnen:

Sowohl der mittlere Anstellwinkel als auch die Amplitude des Anstellwinkels sind sowohl konstant als auch sehr klein, wodurch der gesamte Bereich des sich zeitlich ändernden Anstellwinkels insgesamt mit $-0.1^\circ \leq \alpha \leq 0.9^\circ$ ebenfalls sehr gering ist.
Die Machzahl ist mit $Ma = 0.5$ ausschließlich im subsonischen Bereich und sowohl für das Trainingssignal als auch für alle Testsignale zeitlich konstant. Es liegt zu keinem Zeitpunkt in keinem der Signale ein Verdichtungsstoß vor.
Als zeitlich variable Änderungen ist ausschließlich die Nickbewegung durch die Anstellwinkeländerung berücksichtigt.

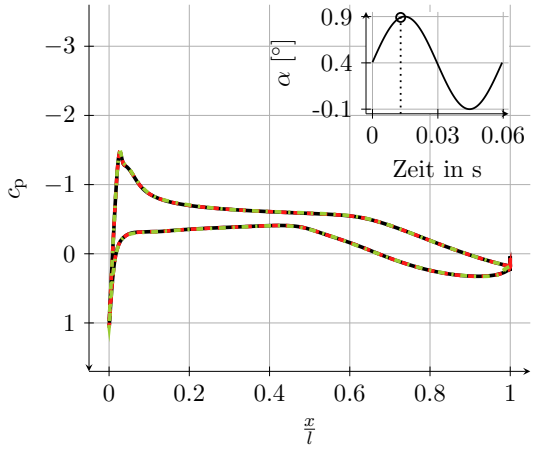
Das Potential der neuronalen Netze als Ersatzmodell sollte daher unbedingt für komplexere instationäre Fallbeispiele in weiteren Untersuchungen analysiert werden. Hierbei sollte die Komplexität schrittweise erhöht werden, um präzise analysieren zu können, für welche Problemstellungen sich die neuronalen Netze im Bezug auf eine instationäre Strömung eignen bzw. nicht eignen.



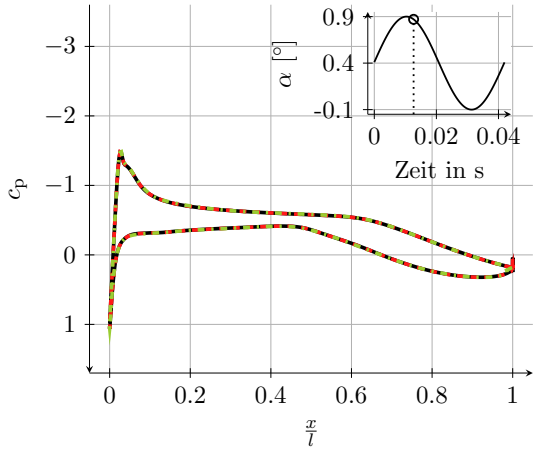
(a) Testsignal 0



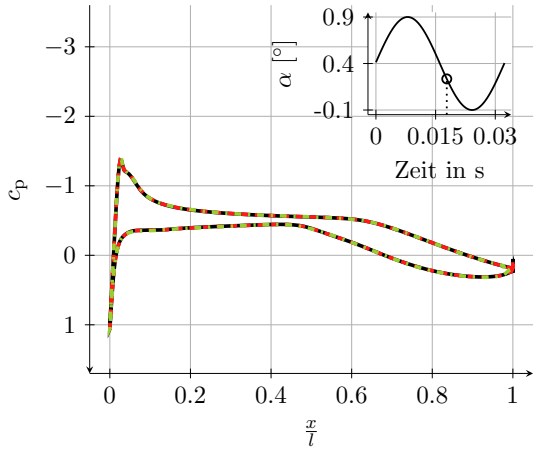
(b) Testsignal 1



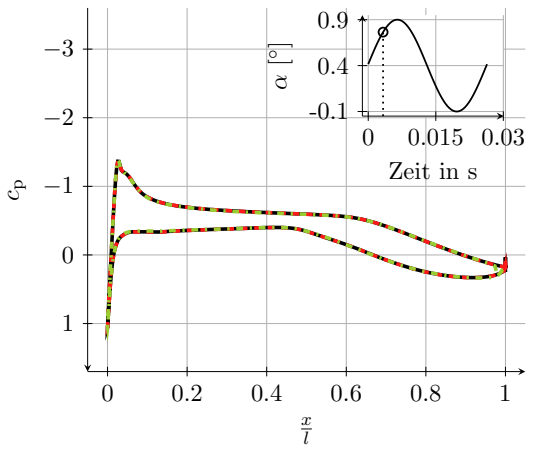
(c) Testsignal 2



(d) Testsignal 3



(e) Testsignal 4



(f) Testsignal 5

Abbildung 3.12: Fallbeispiel 2: Vorhergesagte Druckverteilung der verschiedenen Testsignale mit dem 3327 Epochen trainierten $input_2$ basierten neuronalen Netz und dem klassischen POD+TPS-Ersatzmodell im Vergleich mit der CFD-Lösung an dem Zeitschritt, bei dem die Vorhersage des neuronalen Netzes gemessen am MAE am ungenausten ist

3.3 Fallbeispiel 3: Stationäre 3D-Umströmung eines vollständigen Flugzeugmodells

Dieses Fallbeispiel handelt von der dreidimensionalen stationären Umströmung des allgemeinen Forschungsmodells CRM (Common Research Model) der NASA, welches in Abbildung 3.13 gezeigt ist. Analog zum ersten Fallbeispiel wird sowohl der sub- als auch transsonische Bereich untersucht. Ziel ist es, die Druckverteilung auf der Oberfläche des CRMs für verschiedene Machzahl-Anstellwinkel-Kombinationen vorherzusagen.

3.3.1 Methodik

Basierend auf den Ergebnissen des ersten Fallbeispiels wird die Datenaufbereitung mit Methode B durchgeführt, wobei diese um die zusätzliche dritte Dimension erweitert wird, und geeignete Hyperparameter werden mit der Bayesian-Optimierung gesucht. Aufbauend auf den Ergebnissen der Bayesian-Optimierung wird ein neuronales Netz mit den besten Hyperparametern mit einer größeren Anzahl an Epochen erstellt. Anschließend wird das neuronale Netz, was die höchste Vorhersagegenauigkeit aufweist, mit dem klassischen POD+TPS-Ersatzmodell anhand der definierten Kriterien verglichen und das Potential des neuronalen Netzes für dieses Fallbeispiel bewertet.

3.3.1.1 Beschreibung der Daten

Es liegen CFD-Daten für 48 verschiedene Machzahl-Anstellwinkel-Kombinationen vor. Die Machzahl deckt den sub- und transsonischen Bereich ab und liegt im Bereich $0.50 \leq Ma \leq 0.90$, während der Anstellwinkel im Bereich von $-3 \leq \alpha \leq 8$ Grad liegt.

Die Kombinationen sind wie im ersten Fallbeispiel durch ein DoE basierend auf der Halton Sequence berechnet. Die Daten sind in Abbildung 3.14 sortiert nach 40 Trainings- und 8 Testdaten dargestellt. Die Oberfläche des Modells wird durch 227675 Punkten beschrieben. Dementsprechend liegt die Druckverteilung für jede Machzahl-Anstellwinkel-Kombination durch 227675 Druckbeiwerte vor.

3.3.1.2 Datenaufbereitung

Analog zu den vorherigen Fallbeispielen werden die Eingangsgrößen für die neuronalen Netze auf den Bereich [-1, 1] skaliert. Die Daten werden für das neuronale Netz nach dem gleichen Ansatz, der Methode B zugrunde liegt, aufbereitet. Die kartesischen Koordinaten des jeweiligen Oberflächenpunktes werden als Eingangsgröße definiert. Die Position eines Oberflächenpunktes wird analog zum zweidimensionalen nun mit der zusätzlichen y-Koordinate beschrieben, wobei die Koordinaten in diesem Fall anders als im zweidimensionalen Fallbeispiel nicht relativ zur Sehnenlänge l, sondern absolut beschrieben werden. Der Eingangs- und Ausgangsvektor für das neuronale Netz ergibt sich somit zu (3.17) und (3.18) für $\mathrm{n} \in [1, \mathrm{N}]$, wobei für dieses Fallbeispiel $N = 227675$ entsprechend der

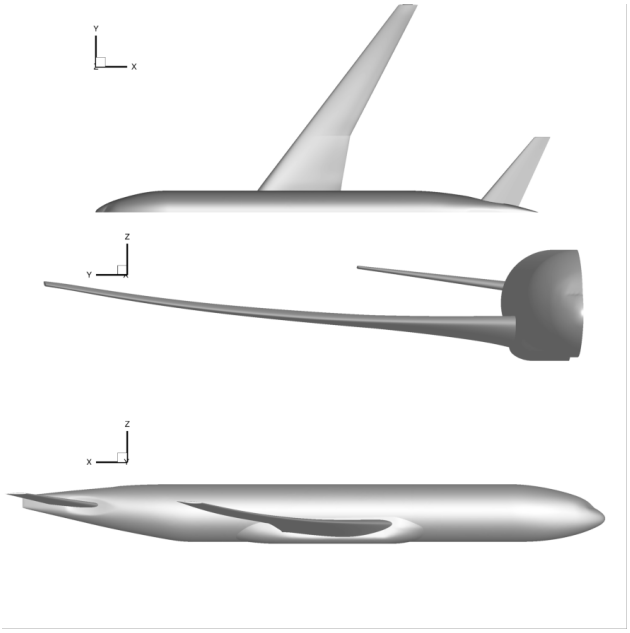


Abbildung 3.13: Fallbeispiel 3: NASA Common Research Model

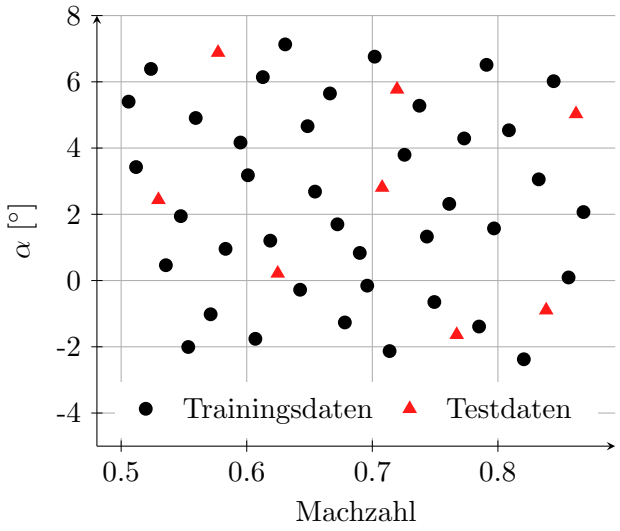


Abbildung 3.14: Fallbeispiel 3: Trainings- und Testdaten

Oberflächenpunkte gilt. Durch die Aufteilung der Oberflächenpunkte ergibt sich eine Gesamtanzahl der Daten für das Training von $227675 \cdot 40 = 9107000$.

$$input = (Ma, \alpha, x_{\mathrm{n}}, y_{\mathrm{n}}, z_{\mathrm{n}}) \tag{3.17}$$

$$output = (c_{\mathrm{p_n}}) \tag{3.18}$$

Hyperparameter	Bereich	Ergebnis
Batchgröße bs	[256, 4096]	3060
Dropoutrate	[0.0, 0.5]	0.0
Anfangslernrate lr_0	[0.001, 0.1]	0.001
Abnahmefaktor k	[1e-6, 1e-2]	1e-6
Anzahl verdeckter Schichten	[1, 11]	11
Neuronenanzahl pro Schicht	[25, 500]	11x500

Tabelle 3.15: Fallbeispiel 3: In der Optimierung berücksichtigte Hyperparameter, deren zulässiger Optimierungsbereich und das Ergebnis der Optimierung

3.3.1.3 Entwurfsprozess des neuronalen Netzes

Die Hyperparameteroptimierung des neuronalen Netzes wird auch hier mit der Bayesian-Optimierung durchgeführt. Hierfür werden 40 Iterationen mit jeweils 30 Epochen angesetzt. Da sich bereits beim ersten Fallbeispiel das neuronale Netz mit der größten Kapazität als Bestes herausgestellt hat und dieses Fallbeispiel durch die dritte Dimension komplexer ist als das erste Fallbeispiel, wird die maximal mögliche Kapazität des neuronalen Netzes erhöht, indem eine maximal mögliche Anzahl an Neuronen pro Schicht von 500 ermöglicht wird.

Aufbauend auf den Ergebnissen vorläufiger manueller Untersuchungen werden die Hyperparameterbereiche der Bayesian-Optimierung gewählt. Diese und die besten Hyperparameter als Ergebnis der Optimierung sind in Tabelle 3.15 aufgelistet.

Das neuronale Netz mit den durch die Bayesian-Optimierung bestimmten Hyperparametern wird anschließend mit einer maximalen Anzahl von 700 Epochen trainiert. Als klassisches Ersatzmodell wird auch hier das POD+TPS-Ersatzmodell verwendet.

3.3.2 Ergebnisse und Diskussion

Die manuellen Voruntersuchungen der Hyperparameter zur Identifikation eines sinnvollen Bereichs für die Optimierung führen zu einem neuronalen Netz, mit dem genauere Vorhersagen erzielt werden als mit dem neuronalen Netz, dessen Hyperparameter durch die Bayesian-Optimierung gefunden werden. Dieser Befund kann durch eine oder mehrere der folgenden Ursachen begründet werden:

- Die Iterationsanzahl der Optimierung war zu gering für den gewählten Hyperparameterbereich.
- Die Netze wurden mit zu wenigen Epochen trainiert. Der Lernprozess der neuronalen Netze mit unterschiedlichen Hyperparametern kann am Anfang des Trainings sehr ähnlich sein, sodass eine gewisse Grenze an Trainingsepochen überschritten werden muss, um das wahre Potential der neuronalen Netze mit verschiedenen Hyperparametern zu identifizieren. Wo diese Grenze liegt ist abhängig von der Komplexität des Problems und sollte optimalerweise bereits durch die manuellen Voruntersuchungen herausgefunden werden.

- Der Hyperparameterbereich wurde zu klein gewählt.

Die hier aufgeführten Ergebnisse für den Vergleich mit dem klassischen Ersatzmodell sind dem mit 700 Epochen trainierten besten neuronalen Netz aus den Voruntersuchungen der Hyperparameter zuzuordnen. Die zugehörigen Hyperparameter sind in Tabelle 3.16 aufgelistet.

Hyperparameter	**Einstellung**
Batchgröße bs	512
Dropoutrate	0.0
Anfangslernrate lr_0	0.001
Abnahmefaktor k	6e-3
Anzahl verdeckter Schichten	4
Neuronenanzahl pro Schicht	4x500

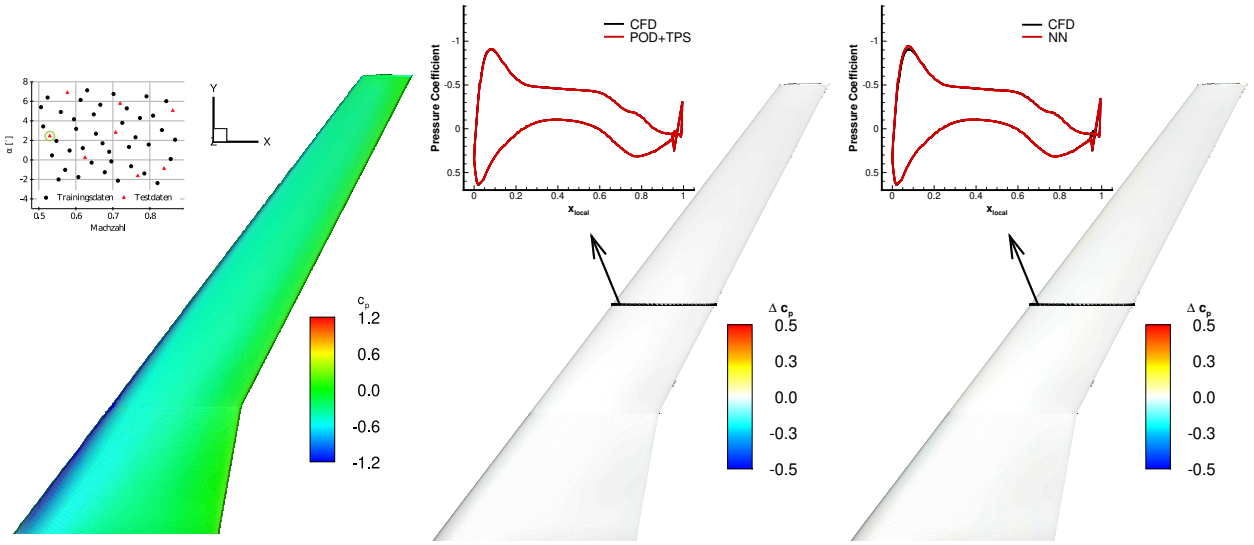
Tabelle 3.16: Fallbeispiel 3: In der Voruntersuchung getestete Hyperparameterkombination, mit denen das neuronale Netz die genausten Vorhersagen erzielt

Die Vorhersageergebnisse der Ersatzmodelle sind beispielhaft für je eine Machzahl-Anstellwinkel-Kombination des subsonischen (Abbildung 3.15) und transsonischen Bereichs (Abbildung 3.16) als Draufsicht des Flugzeugflügels dargestellt. Die Einordnung der Anströmbedingungen in den Datenbereich ist jeweils links oben zu erkennen. Links ist die Druckverteilung der CFD-Lösung abgebildet, während die Abweichung der Vorhersage der Ersatzmodelle von der CFD-Lösung jeweils rechts gezeigt sind. Zudem ist die Druckverteilung der CFD-Lösung und des jeweiligen Ersatzmodells für eine Schnittebene des Flügels bei $y = 17$ exemplarisch abgebildet.

Es ist zu erkennen, dass sowohl das neuronale Netz als auch das POD+TPS-Ersatzmodell im subsonischen Bereich sehr genaue Vorhersageergebnisse liefern. Im transsonischen Bereich tritt - wie sowohl die CFD-Lösung als auch die Druckverteilung in der Schnittebene zeigt - ein Verdichtungsstoß auf der Oberseite des Profils auf. Während die Vorhersage der Druckverteilung auf der Unterseite des Profils beider Ersatzmodelle der CFD-Lösung stark ähnelt, wird der Verdichtungsstoß durch das neuronale Netz deutlich genauer vorhergesagt als durch das klassische Ersatzmodell. Dieses klassische Ersatzmodell sagt eine generell zu kleine Saugspitze auf der Oberfläche voraus und teilt den Verdichtungsstoß in mehrere kleine Verdichtungsstöße auf, deren Positionen deutlich zu weit in Richtung Vorderkante vorhergesagt werden, wodurch große Abweichungen zur CFD-Lösung entstehen. Die Aufteilung in mehrere kleine Verdichtungsstöße mit abweichenden Positionen könnte daher kommen, dass sich die Position des Stoßes abhängig von der Machzahl-Anstellwinkel-Kombination verschiebt und diese Informationen einen Einfluss auf die interpolationsbasierten Vorhersagen hat.

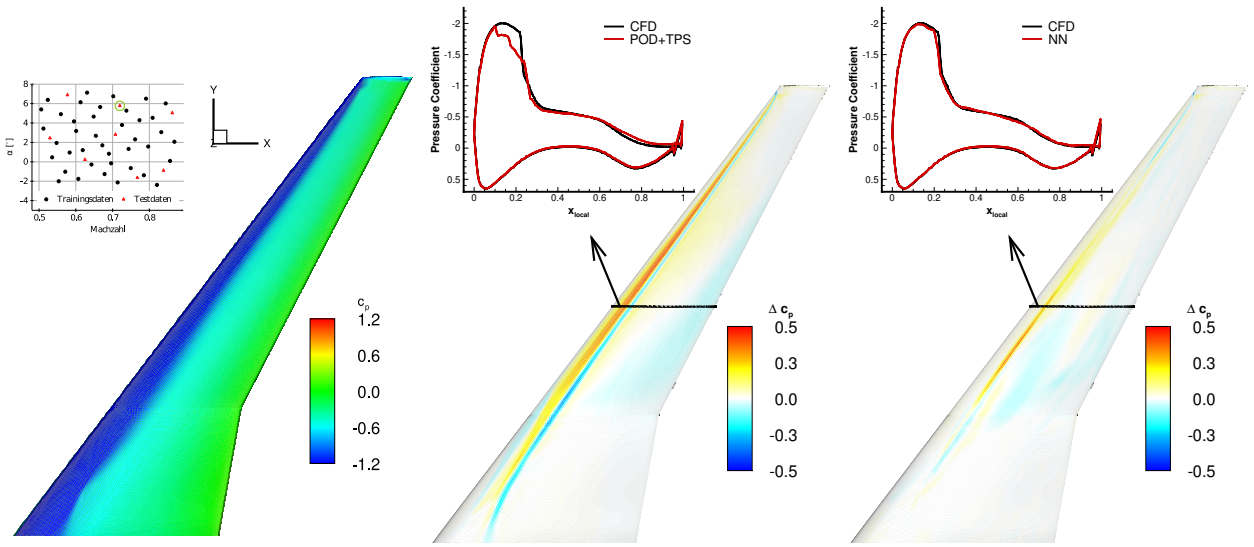
Die generellen Abweichungen zur CFD-Lösung für alle Vorhersagen der Testdaten sowie der zeitliche Aufwand beider Ersatzmodelle können Tabelle 3.17 entnommen werden. Der generelle Vorhersagefehler ist mit $MAE = 12,74 \cdot 10^{-3}$ bei dem neuronalen Netz geringfügig größer als der $MAE = 10.17 \cdot 10^{-3}$ des POD+TPS-Ersatzmodells, wobei das Verhältnis der MSE der Ersatzmodelle zueinander umgekehrt ist. Dies lässt sich dadurch begründen, dass große Abweichungen einzelner Punkte den MSE mehr als den MAE beeinflussen. Das POD+TPS-Ersatzmodell erzielt demnach an einem Großteil der Oberflächenpunkte genauere Ergebnisse als das neuronale Netz, die

aber an einigen Punkten deutlich stärker von der CFD-Lösung abweichen, als die der neuronalen Netze. Dies deckt sich mit den zuvor beschriebenen ungenauen Vorhersagen des Verdichtungsstoßes durch das POD+TPS-Ersatzmodell und den genaueren Vorhersagen des neuronalen Netzes.



(a) CFD **(b)** POD+TPS **(c)** Neuronales Netz

Abbildung 3.15: Fallbeispiel 3: Mit neuronalem Netz und klassischem POD+TPS-Ersatzmodell vorhergesagte Druckverteilung für den Flügelbereich des CRMs im Vergleich mit der CFD-Lösung für $Ma = 0.52$ und $\alpha = 2.43$



(a) CFD **(b)** POD+TPS **(c)** Neuronales Netz

Abbildung 3.16: Fallbeispiel 3: Mit neuronalem Netz und klassischem POD+TPS-Ersatzmodell vorhergesagte Druckverteilung für den Flügelbereich des CRMs im Vergleich mit der CFD-Lösung für $Ma = 0.72$ und $\alpha = 5.77$

Das Training des POD+TPS-Ersatzmodells dauert lediglich wenige Sekunden und ist damit mehr als 16500-mal so schnell wie das Training des neuronalen Netzes, welches knapp 18.5 Stunden benö-

Ersatzmodell	**MAE** $[\cdot\mathbf{10^{-3}}]$	**MSE** $[\cdot\mathbf{10^{-3}}]$	**Trainingszeit**	**Vorhersagezeit**
Neuronales Netz	10.80	0.80	18.5 h	3.1 s
POD+TPS	10.17	1.15	4.01 s	0.02 s

Tabelle 3.17: Fallbeispiel 3: Vorhersagefehler gemessen als MAE und MSE des neuronalen Netzes und des klassischen POD+TPS-Ersatzmodells bezogen auf alle Testdaten sowie der zeitliche Rechenaufwand unterteilt als Trainings- und Vorhersagezeit für die Druckverteilung für eine Machzahl-Anstellwinkel-Kombination

tigt. Auch die Vorhersage einer Machzahl-Anstellwinkel-Kombination dauert mit dem klassischen Ersatzmodell lediglich 0.02 Sekunden, während das neuronale Netz mit 3.1 Sekunden deutlich mehr Zeit benötigt.

3.3.3 Fazit

Mit dem POD+TPS-Ersatzmodell sind, verglichen mit dem neuronalen Netz, insgesamt für den sub- und transsonischen Bereich gemittelt genauere Vorhersagen möglich bei einer deutlich geringeren Trainingszeit. Im subsonischen Bereich liefern beide Ersatzmodelle sehr genaue Ergebnisse. Durch die deutlich genaueren Vorhersagen des Verdichtungsstoßes im transsonischen Bereich durch die neuronalen Netze im Vergleich mit dem POD+TPS-Ersatzmodell wird der Einsatz neuronaler Netze trotz damit verbundener längerer Trainingszeit in diesem Strömungsbereich gerechtfertigt, was sich mit dem Fazit des ersten Fallbeispiels deckt.
Die Untersuchung hat außerdem gezeigt, dass eine nahezu zufällige manuelle Wahl der Hyperparameter des neuronalen Netzes zu insgesamt ähnlich genauen Vorhersageergebnissen führt wie mit dem klassischen Ersatzmodell. Es wird vermutet, dass eine Wiederholung der Bayesian-Optimierung mit angepassten Einstellungen zu neuronalen Netzen mit genaueren Vorhersagen führt.

4 Zusammenfassung und Ausblick

Ziel dieser Arbeit ist die Potentialanalyse eines neuronalen Netzes als datengetriebenes Ersatzmodell zur Effizienzsteigerung bei der Erzeugung aerodynamischer Daten. Zu diesem Zweck werden unterschiedliche Datenaufbereitungsmethoden und Methoden der Hyperparameteroptimierung untersucht, um ein neuronales Netz als Ersatzmodell zu erzeugen, das im Rahmen dieser Arbeit die genausten Vorhersagen erzielt. Außerdem wird das häufig verwendete klassische Ersatzmodell bestehend aus POD-Dimensionsreduzierungs- und TPS-Interpolationsmethode erstellt. Hierfür wird zunächst ein detaillierter Einblick in die für die Verwendung der Ersatzmodelle notwendigen theoretischen Grundlagen gegeben. Insbesondere werden die Grundkonzepte eines systematischen Vorgehens für den Entwurfsprozess eines neuronalen Netzwerks erläutern. Anschließend werden sowohl neuronale Netze als auch klassische Ersatzmodelle für verschiedene aerodynamische Fallbeispiele eingesetzt. Die Ergebnisse beider Ersatzmodelle werden anschließend anhand der Vorhersagegenauigkeit, die aus einem Vergleich mit der CFD-Lösung berechnet wird, und anhand des benötigten zeitlichen Aufwands miteinander verglichen, um das Potential der neuronalen Netze zu bewerten.

Im ersten Fallbeispiel, in dem eine sub- und transsonsiche Strömung um ein Flügelprofil untersucht wurde, sind zunächst die Bayesian-Optimierung und die Random-Search-Methode als Optimierungsansätze für die Bestimmung geeigneter Hyperparameter auf gleicher Datenbasis untersucht worden. Dafür wurden die Ergebnisse der neuronalen Netze, denen die jeweils als Beste identifizierten Hyperparameter beider Optimierungsansätze zugrunde liegen, anhand der definierten Kriterien bewertet. Die Bayesian-Optimierung stellte hierbei die geeignetere Methode dar. Anschließend wurden für die Vorhersage der Druckverteilung drei Datenaufbereitungsmethoden analysiert, die auf unterschiedlichen Ansätzen basieren. Der erste Ansatz realisiert die Vorhersage der kompletten Druckverteilung auf der Oberfläche auf einmal, die ausschließlich abhängig von den Anströmbedinungen als Eingangsgrößen ist. Bei den beiden anderen Ansätzen werden die Druckbeiwerte der Oberflächenpunkte einzeln vorhergesagt, was eine Integration der Position des jeweiligen Oberflächenpunktes als zusätzliche Eingangsgröße erfordert. Diese wird dabei in einem Fall durch die kartesischen und im anderen Fall durch die kurvilinearen Koordinaten beschrieben. Die Ergebnisse belegen, dass die Vorhersagen bei einer Aufteilung der Druckverteilung in die einzelnen Druckbeiwerte deutlich genauer sind. Weiterhin führt die kartesische Beschreibung der Position zu genaueren Ergebnissen als die kurvilineare Darstellung.

Es werden sowohl neuronale Netze als auch klassische Ersatzmodelle, basierend auf ausschließlich einer TPS Interpolation, zur direkten Vorhersage des Auftriebs-, Widerstands- und Nickmomentenbeiwerts verwendet. Hierbei wird für jeden skalaren Beiwert sowohl ein neuronales Netz als auch

ein klassisches Ersatzmodell verwendet. Zusätzlich zu der direkten Berechnung durch das neuronale Netz wird jeder der Beiwerte auf indirektem Wege durch die Integration der Druckverteilung berechnet, die im vorherigen Schritt von dem besten neuronalen Netz vorhergesagt wurde. Außerdem werden anhand der skalaren Beiwerte die Extrapolationsfähigkeiten der Ersatzmodelle getestet.

Die integrative Bestimmung der skalaren Beiwerte führte zu deutlich geringeren Abweichungen von der CFD-Lösung als die Verwendung der dezidiert für direkte Vorhersagen der skalaren Beiwerte erstellten Ersatzmodelle. Die neuronalen Netze für die direkten Vorhersagen erzielen dabei weder genauere Vorhersagen noch sind sie effizienter als die klassischen Ersatzmodelle, sodass der Einsatz von neuronalen Netzen dezidiert für die Vorhersagen der skalaren Beiwerte nicht zielführend ist und der Einsatz der klassischen Ersatzmodelle hierfür besser geeignet ist. Außerdem erzielt kein Ersatzmodell eine hohe Vorhersagegenauigkeit für extrapolative Vorhersagen.
Durch den abschließenden Vergleich der Ergebnisse des neuronalen Netzes mit denen des klassischen Ersatzmodells für die Vorhersagen der Druckverteilung wurde festgestellt, dass neuronale Netze einen deutlich größeren zeitlichen Aufwand erfordern. Die insgesamt höhere Vorhersagegenauigkeit, insbesondere beim Auftreten von Verdichtungsstößen im transsonischen Strörmungsbereich, rechtfertigen diesen höheren Zeitaufwand jedoch.

Bei der Untersuchung des zweiten Fallbeispiels, dem eine zweidimensionale instationäre subsonische Strömung zugrunde liegt, wurden aufbauend auf den zielführendsten Methoden des ersten Fallbeispiels weitere Datenaufbereitungsmethoden untersucht. Diese ergeben sich speziell für die Beschreibung des instationären Verhaltens, das durch einen zeitlich variablen Anstellwinkel erzeugt wurde. Hierfür wurde zunächst ausschließlich der zeitliche Verlauf des Anstellwinkels, anschließend zusätzlich die erste und schließlich die erste und zweite Ableitung dieses Verlaufs als Eingangsgrößen definiert. Es zeigte sich ein deutlicher Trend zu genaueren Vorhersagen bei zunehmender Anzahl an Eingangsgrößen. Der finale Vergleich des neuronalen Netzes mit dem klassischen Ersatzmodell zeigte, dass die Vorhersagen beider Modelle den CFD-Lösungen stark ähneln. Das klassische Ersatzmodell erzielt insgesamt genauere Vorhersagen bei einem geringeren zeitlichen Aufwand als das neuronale Netz. Es wird basierend auf den Ergebnissen des ersten Fallbeispiels vermutet, dass die Vorhersagegenauigkeit des neuronalen Netzes im transsonischen Strömungsbereich deutlich höher ist als die des klassischen Ersatzmodells.

Im dritten Fallbeispiel wird die dreidimensionale stationäre sub- und transsonische Umströmung des CRM-Flugzeugmodells untersucht. Aufbauend auf den Ergebnissen aus dem ersten Fallbeispiel sind die geeignetsten Methoden der Hyperparameteroptimierung und der Datenaufbereitung - modifiziert durch die Erweiterung um die dritte Dimension - gewählt worden. Der abschließende Vergleich der Ersatzmodelle anhand der Vorhersagegenauigkeiten und dem zeitlichen Aufwand führen zur gleichen Potentialbewertung des neuronalen Netzes wie im ersten Fallbeispiel: Der Zeitaufwand zur Erstellung der klassischen Ersatzmodelle ist signifikant geringer als der zur Erstellung der neuronalen Netze benötigte zeitliche Aufwand. Aufgrund der ähnlichen Vorhersagegenauigkeiten der Ersatzmodelle im subsonischen Bereich, jedoch deutlich genauerer Vorhersagen der neuronalen Netze im transsonischen Bereich, ist der zeitliche Mehraufwand zur Erstellung der neuronalen Netze

für Vorhersagen im transsonischen aber nicht im subsonischen Strömungsbereich gerechtfertigt.
Die Hyperparameter, die zu den genausten Vorhersageergebnissen des neuronalen Netzes führen, wurden in diesem Fallbeispiel nicht durch die Hyperparameteroptimierung, sondern durch die manuellen vorläufigen Untersuchungen gefunden. Aufgrund dieser Tatsache ist durch eine detailliertere Hyperparameteroptimierung eine Steigerung der Vorhersagegenauigkeit des neuronalen Netzes zu erwarten.

Abschließend lässt sich sagen, dass der Zeitaufwand der neuronalen Netze verglichen mit dem der klassischen Ersatzmodellen für alle Fallbeispiele deutlich höher ist, jedoch - abhängig von der zu generierenden Datenmenge - immer noch signifikant geringer ist als der für die CFD-Simulationen benötigte Zeitaufwand.
Der Einsatz von neuronalen Netzen als Ersatzmodell besitzt daher ein großes Potential zur Effizienzsteigerung bei der Generierung aerodynamischer Daten bei einer sehr hohen Vorhersagegenauigkeit. Diese hebt sich besonders deutlich im transsonischen Strömungsbereich, d.h. bei auftretenden komplexen Strömungsphänomenen wie beispielsweise einem Verdichtungsstoß, von den Vorhersagegenauigkeit des klassischen Ersatzmodells ab. Aufgrund dieses großen Potentials sollen zukünftig weitere Untersuchungen durchgeführt werden, die zur Verbesserung der Ergebnisse und weiteren Einsichten des Verhaltens der neuronalen Netze führen.

In Zukunft sollte insbesondere versucht werden, den geringen zeitlichen Aufwand des klassischen Ersatzmodells mit der hohen Vorhersagegenauigkeit des neuronalen Netzes im transsonischen Bereich zu kombinieren. Zu diesem Zweck wird die Dimensionsreduzierung des klassischen Ersatzmodells mit dem neuronalen Netz zur Approximation der nicht-linearen Beziehungen vereint. Hierfür können verschiedene Methoden der Dimensionsreduzierung, wie beispielsweise die POD-Methode oder die Verwendung eines Autoencoder neuronalen Netzes untersucht und verglichen werden. Dazu würden die Faktoren Zeitersparnis und Genauigkeitsverlust miteinander ins Verhältnis gesetzt und so bewertet werden, inwieweit die Kombination der jeweiligen Dimensionsreduzierungsmethode mit neuronalen Netzen vorteilhaft ist. Außerdem kann zukünftig die Optimierung der Hyperparameter beschleunigt werden, indem die während des Trainings berechnete Lernkurve des neuronalen Netzes bei jeder Optimierungsiteration extrapoliert und bewertet wird, um einen frühzeitigen Abbruch des Trainings mit aussichtslosen Hyperparametern zu ermöglichen. Für eine weitere deutliche Reduzierung des zeitlichen Aufwands kann die hochgradige Parallelisierbarkeit der neuronalen Netze ausgenutzt werden. Hierfür sollte zumindest das Training, welches bezogen auf den zeitlichen Aufwand den größten Anteil bei der Erstellung neuronaler Netze ausmacht, auf einem state-of-the-art Grafikprozessor oder einem Grafikprozessor-Cluster durchgeführt werden.

Ein weiteres Ziel kann sein, den Prozess der Erstellung des neuronalen Netzes, beginnend mit der Datenaufbereitung bis hin zu einem trainierten neuronalen Netz, welches genauere Vorhersageergebnisse erzielt als die klassischen Ersatzmodelle, zu automatisieren. Die größte Herausforderung dabei besteht darin, eine geeignete Methode zu entwickeln, die automatisch eine sinnvolle Wahl des Bereichs der einzelnen Hyperparameter für die Optimierung trifft. Diese Wahl basiert in dieser Arbeit auf händischen Voruntersuchungen und wird dementsprechend vor der Optimierung manuell gesetzt.

Ein Ansatz ist beispielsweise, einen weiterhin manuell gesetzten, aber äußerst großen initialen Bereich zu wählen, der durch den Optimierungsalgorithmus zunächst grob untersucht und anschließend sukzessive verringert wird. Gleichermaßen würden die Untersuchungen der Hyperparameter durch eine Erhöhung der Trainingsepochen oder der Anzahl an getesteten Hyperparameterkombinationen mit jeder Reduktion des Bereichs schrittweise verfeinert werden. Zudem können einige Hyperparameter aufbauend auf bestehender Literatur in sinnvolle Bereiche eingegrenzt werden. Diese sind allerdings problemspezifisch und sollten daher als Richtlinien gesehen und mit Vorsicht verwendet werden.

Bei der Generierung der CFD-Lösungen wurden die Datenpunkte im ersten und dritten Fallbeispiel für den gesamten - also sowohl sub- als auch transsonischen - Datenbereich durch ein einziges DoE ausgewählt. Wenn die Ausmaße der beiden Bereiche, die sich über die jeweilige Machzahl definieren, unterschiedlich groß sind, wird einer der beiden Bereiche durch eine größere Anzahl an Daten beschrieben. Die Inhomogenität der Informationsmengen bezogen auf die beiden Bereiche kann einen Trend zu genaueren Vorhersagen der Ersatzmodelle zu dem Bereich erzeugen, der durch die größere Datenmenge beschrieben wird. Um diesen verzerrenden Effekt zu vermeiden, sollten die Daten beider Bereiche mit je einem DoE ausgewählt werden, um in beiden Bereichen eine gleiche Menge an Datenpunkten zu erhalten.

Das zweite Fallbeispiel ist zukünftig komplexer zu gestalten und erneut mit den verschiedenen Ersatzmodellen zu untersuchen. Von besonderem Interesse wäre es, die Eignung der Ersatzmodelle für die zusätzliche Betrachtung des transsonischen Bereichs zu prüfen, da sich die Vorhersagegenauigkeiten der verschiedenen Ersatzmodelle für diesen Bereich in den anderen untersuchten Fallbeispielen stark unterscheiden. Es gibt unzählige weitere Möglichkeiten, eine zunehmende Komplexität der aerodynamischen Herausforderungen zu erzielen, von denen im Folgenden einige beispielhaft aufgeführt werden. Zum einen kann nicht nur die Anstellwinkeländerung, sondern auch die zeitabhängige vertikale Verschiebung des Profils berücksichtigt werden. Zum anderen können für den mittleren Anstellwinkel und die Amplitude sowohl die Werte erhöht als auch eine zeitliche Änderung aufgeprägt werden. Darüber hinaus können komplexere Signale für das Training verwendet werden, sodass auch die Testsignale entsprechend komplexer gestaltet werden können.

Schließlich sind einige Möglichkeiten zur Steigerung der Vorhersagegenauigkeit der neuronalen Netze zu nennen, die jedoch mit einem zeitlichen Mehraufwand verbunden sind. Eine davon realisiert die Bestimmung geeigneterer Hyperparameterkombinationen durch die Erhöhung der Iterationszahl der Hyperparameteroptimierung. Weitere Möglichkeiten bestehen darin, die Trainingsepochen zu erhöhen oder zusätzliche in dieser Arbeit nicht betrachtete Hyperparameter in der Optimierung zu berücksichtigen.

Literatur

[1] Grant, I. et al.: *An investigation of the performance of multi layer, neural networks applied to the analysis of PIV images*. In: Experiments in Fluids 19.3 (1995), S. 159–166.

[2] Dissanayake, M. et al.: *Neural-network-based approximations for solving partial differential equations*. In: communications in Numerical Methods in Engineering 10.3 (1994), S. 195–201.

[3] Santos, M. et al.: *Aerodynamic coefficient prediction of airfoils using neural networks*. In: *46th AIAA aerospace sciences meeting and exhibit*. 2008, S. 887.

[4] Secco, N. R. et al.: *Artificial neural networks to predict aerodynamic coefficients of transport airplanes*. In: Aircraft Engineering and Aerospace Technology (2017).

[5] Zelong, Y. et al.: *Aerodynamic Coefficient Prediction of Airfoils with Convolutional Neural Network*. 2019.

[6] Zhang, Y. et al.: *Application of convolutional neural network to predict airfoil lift coefficient*. In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018, S. 1903.

[7] Bhatnagar, S. et al.: *Prediction of aerodynamic flow fields using convolutional neural networks*. In: Computational Mechanics 64.2 (2019), S. 525–545.

[8] Thuerey, N. et al.: *Deep learning methods for Reynolds-averaged Navier-Stokes simulations of airfoil flows*. In: AIAA Journal 58.1 (2020), S. 25–36.

[9] Wu, H. et al.: *A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils*. In: Computers & Fluids 198 (2020), S. 104–393.

[10] Winter, M. et al.: *Monte-Carlo-Based Training and Application Framework for Enhanced Nonlinear Aerodynamic Reduced-Order Modeling*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2017.

[11] Ignatyev, D. I. et al.: *Neural network modeling of unsteady aerodynamic characteristics at high angles of attack*. In: Aerospace Science and Technology 41 (2015), S. 106–115.

[12] Wang, Z. et al.: *Model identification of reduced order fluid dynamics systems using deep learning*. In: International Journal for Numerical Methods in Fluids 86.4 (2018), S. 255–268.

[13] Mohan, A. T. et al.: *A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks.* In: arXiv preprint arXiv:1804.09269 (2018).

[14] Li, K. et al.: *Deep neural network for unsteady aerodynamic and aeroelastic modeling across multiple Mach numbers.* In: Nonlinear Dynamics 96.3 (2019), S. 2157–2177.

[15] Amsallem, D. et al.: *Interpolation method for adapting reduced-order models and application to aeroelasticity.* In: AIAA journal 46.7 (2008), S. 1803–1813.

[16] Lieu, T. et al.: *Adaptation of aeroelastic reduced-order models and application to an F-16 configuration.* In: AIAA journal 45.6 (2007), S. 1244–1257.

[17] Lieu, T. et al.: *Reduced-order fluid/structure modeling of a complete aircraft configuration.* In: Computer methods in applied mechanics and engineering 195.41-43 (2006), S. 5730–5742.

[18] Hay, A. et al.: *Local improvements to reduced-order models using sensitivity analysis of the proper orthogonal decomposition.* In: Journal of Fluid Mechanics 629 (2009), S. 41–72.

[19] Mathelin, L. et al.: *Robust control of uncertain cylinder wake flows based on robust reduced order models.* In: Computers & fluids 38.6 (2009), S. 1168–1182.

[20] Arian, E. et al.: *Trust-region proper orthogonal decomposition for flow control.* Techn. Ber. Institute for computer applications in science und engineering Hampton VA, 2000.

[21] Fahl, M. et al.: „Reduced order modelling approaches to PDE-constrained optimization based on proper orthogonal decomposition“. In: *Large-scale PDE-constrained optimization.* Springer, 2003, S. 268–280.

[22] Yue, Y. et al.: *Accelerating optimization of parametric linear systems by model order reduction.* In: SIAM Journal on Optimization 23.2 (2013), S. 1344–1370.

[23] Benner, P. et al.: *A survey of projection-based model reduction methods for parametric dynamical systems.* In: SIAM review 57.4 (2015), S. 483–531.

[24] Gugercin, S. et al.: *A survey of model reduction by balanced truncation and some new results.* In: International Journal of Control 77.8 (2004), S. 748–766.

[25] Penzl, T.: *Algorithms for model reduction of large dynamical systems.* In: Linear algebra and its applications 415.2-3 (2006), S. 322–343.

[26] Grimme, E.: „Krylov projection methods for model reduction“. Diss. 1997.

[27] Sirovich, L.: *Turbulence and the dynamics of coherent structures. Part 1: Coherent structures.* In: Quart Appl Math 45 (1987), S. 561–571.

[28] Kennedy, M. C. et al.: *Bayesian calibration of computer models.* In: Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63.3 (2001), S. 425–464.

[29] Wild, S. M. et al.: *ORBIT: Optimization by radial basis function interpolation in trust-regions.* In: SIAM Journal on Scientific Computing 30.6 (2008), S. 3197–3219.

[30] Donato, G. et al.: *Approximation methods for thin plate spline mappings and principal warps.* Citeseer, 2003.

[31] Keller, W. et al.: *Thin plate spline interpolation.* In: Journal of Geodesy 93.9 (2019), S. 1251–1269.

[32] Simpson, T. W. et al.: *Kriging models for global approximation in simulation-based multidisciplinary design optimization.* In: AIAA journal 39.12 (2001), S. 2233–2241.

[33] Bui-Thanh, T. et al.: *Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications.* In: AIAA journal 46.10 (2008), S. 2520–2529.

[34] Trübelhorn, S.: *Ein POD-ROM-Verfahren für stationäre Strömungsprobleme.* Springer, 2016.

[35] McCulloch, W. S. et al.: *A logical calculus of the ideas immanent in nervous activity.* In: The bulletin of mathematical biophysics 5.4 (1943), S. 115–133.

[36] Rosenblatt, F.: *The perceptron: a probabilistic model for information storage and organization in the brain.* In: Psychological review 65.6 (1958), S. 386.

[37] Hebb, D. O.: *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[38] Goodfellow, I. et al.: *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016.

[39] Krüger, T.: „Zur Anwendung neuronaler Netzwerke in adaptiven Flugregelungssystemen“. Diss. 2012.

[40] Tagscherer, M.: *Dynamische neuronale Netzarchitektur für kontinuierliches Lernen.* In: (2000).

[41] Crone, S.: *Neuronale Netze zur Prognose und Disposition im Handel.* Bd. 60. Springer-Verlag, 2010.

[42] Rojas, R.: *Neural networks: a systematic introduction.* Springer Science & Business Media, 2013.

[43] Haykin, S. et al.: *A comprehensive foundation.* In: Neural networks 2.2004 (2004), S. 41.

[44] Fausett, L. V.: *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India, 2006.

[45] Brunton, S. L. et al.: *Machine learning for fluid mechanics*. In: Annual Review of Fluid Mechanics 52 (2020), S. 477–508.

[46] Hochreiter, S. et al.: *Long short-term memory*. In: Neural computation 9.8 (1997), S. 1735–1780.

[47] Murphy, K. P.: *Machine learning: a probabilistic perspective*. MIT press, 2012.

[48] Kingma, D. P. et al.: *Adam: A method for stochastic optimization*. In: arXiv preprint arXiv:1412.6980 (2014).

[49] Rumelhart, D. E. et al.: *Learning representations by back-propagating errors*. In: nature 323.6088 (1986), S. 533–536.

[50] Rojas, R.: „The backpropagation algorithm“. In: *Neural networks*. Springer, 1996, S. 149–182.

[51] Bengio, Y. et al.: *Learning long-term dependencies with gradient descent is difficult*. In: IEEE transactions on neural networks 5.2 (1994), S. 157–166.

[52] Nwankpa, C. et al.: *Activation functions: Comparison of trends in practice and research for deep learning*. In: arXiv preprint arXiv:1811.03378 (2018).

[53] Neal, R. M.: *Connectionist learning of belief networks*. In: Artificial intelligence 56.1 (1992), S. 71–113.

[54] Karlik, B. et al.: *Performance analysis of various activation functions in generalized MLP architectures of neural networks*. In: International Journal of Artificial Intelligence and Expert Systems 1.4 (2011), S. 111–122.

[55] Glorot, X. et al.: *Deep sparse rectifier neural networks*. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, S. 315–323.

[56] Bishop, C. M. et al.: *Neural networks for pattern recognition*. Oxford university press, 1995.

[57] Stein, R.: *Selecting data for neural networks*. In: AI expert 8 (1993), S. 42–42.

[58] Trottier, L. et al.: *Parametric exponential linear unit for deep convolutional neural networks*. In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2017, S. 207–214.

[59] Bengio, Y.: „Practical recommendations for gradient-based training of deep architectures“. In: *Neural networks: Tricks of the trade*. Springer, 2012, S. 437–478.

[60] Caruana, R. et al.: *Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping.* In: *Advances in neural information processing systems.* 2001, S. 402–408.

[61] Arvetti, M. et al.: *Classification of EMG signals through wavelet analysis and neural networks for controlling an active hand prosthesis.* In: *2007 IEEE 10th international conference on Rehabilitation Robotics.* IEEE. 2007, S. 531–536.

[62] Srivastava, N. et al.: *Dropout: a simple way to prevent neural networks from overfitting.* In: The journal of machine learning research 15.1 (2014), S. 1929–1958.

[63] Füser, K.: *Neuronale Netze in der Finanzwirtschaft: Innovative Konzepte und Einsatzmöglichkeiten.* Springer-Verlag, 2013.

[64] Rafiq, M. et al.: *Neural network design for engineering applications.* In: Computers & Structures 79.17 (2001), S. 1541–1552.

[65] Senior, A. et al.: *An empirical study of learning rates in deep neural networks for speech recognition.* In: *2013 IEEE international conference on acoustics, speech and signal processing.* IEEE. 2013, S. 6724–6728.

[66] Larochelle, H. et al.: *Exploring strategies for training deep neural networks.* In: Journal of machine learning research 10.1 (2009).

[67] Ernest, B. R. et al.: *Adaptive control processes: a guided tour.* 1961.

[68] Bergstra, J. et al.: *Random search for hyper-parameter optimization.* In: The Journal of Machine Learning Research 13.1 (2012), S. 281–305.

[69] Claesen, M. et al.: *Hyperparameter search in machine learning.* In: arXiv preprint arXiv:1502.02127 (2015).

[70] Kiss, O.: *Bayesian Optimization for machine learning algorithms in the context of Higgs searches at the CMS experiment.* In: arXiv preprint arXiv:1911.02501 (2019).

[71] Snoek, J. et al.: *Practical bayesian optimization of machine learning algorithms.* In: *Advances in neural information processing systems.* 2012, S. 2951–2959.

[72] Mockus, J. et al.: *The application of Bayesian methods for seeking the extremum.* In: Towards global optimization 2.117-129 (1978), S. 2.

[73] Brochu, E. et al.: *A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.* In: arXiv preprint arXiv:1012.2599 (2010).

[74] Falkner, S. et al.: *Practical hyperparameter optimization for deep learning.* In: (2018).

[75] Bui-Thanh, T. et al.: *Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics*. In: *21st AIAA Applied Aerodynamics Conference*. 2003, S. 4213.

[76] Chen, L.: „Curse of dimensionality“. In: *Encyclopedia of Database Systems*. 2009.

[77] Wang, Q. et al.: *Techniques for Improving Neural Network-based Aerodynamics Reduced-order Models*. In: *AIAA Scitech 2019 Forum*. 2019, S. 1849.

[78] Franz, T.: „Reduced-order modeling for steady transonic flows via manifold learning“. Diss. DLR, Deutsches Zentrum für Luft-und Raumfahrt, 2016.

[79] Friedman, J. et al.: *The elements of statistical learning*. Bd. 1. 10. Springer series in statistics New York, 2001.

[80] Chollet, F. et al.: *Keras*. `https://keras.io`. 2015.

[81] Martin Abadi et al.: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[82] Pedregosa, F. et al.: *Scikit-learn: Machine Learning in Python*. In: Journal of Machine Learning Research 12 (2011), S. 2825–2830.

[83] Nogueira, F.: *Bayesian Optimization: Open source constrained global optimization tool for Python*. 2014. URL: `https://github.com/fmfn/BayesianOptimization`.

[84] Zimmermann, R. et al.: *Non-linear reduced order models for steady aerodynamics*. In: Procedia Computer Science 1.1 (2010), S. 165–174.

[85] Gerhold, T. et al.: *Calculation of complex three-dimensional configurations employing the DLR-TAU-code*. In: *35th aerospace sciences meeting and exhibit*. 1997, S. 167.

[86] Spalart, P. et al.: *A one-equation turbulence model for aerodynamic flows*. In: *30th aerospace sciences meeting and exhibit*. 1992, S. 439.

[87] Chi, H. et al.: *On the optimal Halton sequence*. In: Mathematics and computers in simulation 70.1 (2005), S. 9–21.

[88] Graves, A. et al.: *Speech recognition with deep recurrent neural networks*. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, S. 6645–6649.

[89] Pascanu, R. et al.: *How to construct deep recurrent neural networks*. In: arXiv preprint arXiv:1312.6026 (2013).

[90] Elgamal, T. et al.: *Analysis of PCA algorithms in distributed environments*. In: arXiv preprint arXiv:1503.05214 (2015).

DLR-IB-AS-BS-2020-88

Untersuchung von Deep-Learning-Methoden zur hocheffizienten Vorhersage von aerodynamischen Daten

Philipp Stürmer

Verteiler:

Institutsbibliothek	1 Exemplar
Verfasser/Co-Autoren	4 Exemplare
Institutsleitung	1 Exemplar
Abteilungsleiter	1 Exemplar
Deutsche Bibliothek in Frankfurt/Main	2 Exemplare
Niedersächsische Landesbibliothek Hannover	1 Exemplar
Techn. Informationsbibliothek Hannover	1 Exemplar
Zentralbibliothek BS	2 Exemplare
Zentralarchiv GÖ	1 Exemplar
Reserve	5 Exemplare