

Optimierung der Systemzuverlässigkeit durch Stochastische Neuronale Systemquantifizierung

Zuverlässigkeitsoptimierung in der additiven Fertigung durch Prognose unterstützt durch stochastische Datengenerierung und Systemquantifizierung

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)

Genehmigte Dissertation im Fachbereich Maschinenbau von Sören Wenzel

Tag der Einreichung: 22. Oktober 2024, Tag der Prüfung: 10. Dezember 2024

Erstreferent: Prof. Dr.-Ing. Tobias Melz

Koreferent: Prof. Dr. rer. nat. Oliver Weeger

Darmstadt, Technische Universität Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Maschinenbau

Fachgebiet

Systemzuverlässigkeit,

Adaptronik und

Maschinenakustik SAM

Optimierung der Systemzuverlässigkeit durch Stochastische Neuronale Systemquantifizierung

Zuverlässigkeitsoptimierung in der additiven Fertigung durch Prognose unterstützt durch stochastische Datengenerierung und Systemquantifizierung

Genehmigte Dissertation im Fachbereich Maschinenbau von Sören Wenzel

Tag der Einreichung: 22. Oktober 2024

Tag der Prüfung: 10. Dezember 2024

Darmstadt, Technische Universität Darmstadt

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-314125

DOI: <https://doi.org/10.26083/tuprints-00031412>

Jahr der Veröffentlichung auf TUprints: 2025

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<https://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung 4.0 International

<https://creativecommons.org/licenses/by/4.0/>

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung 4.0 International

<https://creativecommons.org/licenses/by/4.0/>

Wissen und Können

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbstständig verfasst wurde und dass die „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Technischen Universität Darmstadt“ und die „Leitlinien zum Umgang mit digitalen Forschungsdaten an der TU Darmstadt“ in den jeweils aktuellen Versionen bei der Verfassung der Dissertation beachtet wurden.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 22. Oktober 2024

S. Wenzel

Kurzfassung

Die stochastische neuronale Systemquantifizierung (SNSQ) zeichnet sich durch zwei wesentliche Prinzipien aus: Wissenstransfer und Systemquantifizierung. In der additiven Fertigung existieren zahlreiche Variationsmöglichkeiten im Prozess, wie etwa unterschiedliche Materialien, Geschwindigkeiten und Temperaturen. Daher ist der Wissenstransfer von zentraler Bedeutung. Während bekannte Informationen effizient übermittelt werden können, gestaltet sich der Transfer neuer, unbekannter Informationen als komplexer und erfordert gezielte Ansätze. Die Systemquantifizierung definiert, wie diese Informationen genutzt werden sollen. In dieser Arbeit werden Verhaltensvektoren, die das Systemverhalten repräsentieren, den entsprechenden Beobachtungen in Form von Vektoren zugeordnet. Dies erlaubt es, Prognosen zu erstellen, die auf bestimmten Beobachtungen basieren.

Zusammenfassend präsentiert die Dissertation die innovative Methode der SNSQ, die darauf abzielt, die Vorhersagegenauigkeit in dynamischen und unkontrollierbaren Umgebungen erheblich zu steigern. Durch die stochastische Erzeugung eines umfangreichen Raums möglicher Prognosefunktionen, die verschiedene Systemverhalten abbilden, wird es möglich, relevante Verhaltensmuster und die entsprechenden Prognosen gezielt auszuwählen.

Im Gegensatz zu traditionellen Ansätzen, die sich häufig auf statische Trainingsdaten stützen, ermöglicht SNSQ dynamische Anpassungen in Echtzeit. Die praktische Anwendbarkeit dieser Methode wurde in mehreren Prozessen der additiven Fertigung demonstriert, wo sie ihre Fähigkeit zur präzisen Vorhersage in komplexen und sich verändernden Umgebungen eindrucksvoll unter Beweis gestellt hat.

English Summary

The Stochastic Neural System Quantification (SNSQ) is characterized by two essential principles: knowledge transfer and system quantification. In additive manufacturing, numerous variations exist in the process, such as different materials, speeds, and temperatures. Thus, knowledge transfer is of central importance. While known information can be transmitted efficiently, transferring new, unknown information proves to be more complex and necessitates targeted approaches. System quantification defines how this information should be utilized. In this work, behavioral vectors that represent system behavior are assigned to the corresponding observations in the form of vectors. This enables the generation of forecasts based on specific observations.

In summary, this dissertation introduces the innovative method of SNSQ, aimed at significantly improving prediction accuracy in dynamic and uncontrolled environments. By stochastically generating an extensive range of potential predictive functions that reflect various system behaviors, SNSQ facilitates the targeted selection of relevant behavioral patterns and associated forecasts.

Unlike traditional approaches that often rely on static training data, SNSQ allows for real-time dynamic adjustments. The practical applicability of this method has been demonstrated across several additive manufacturing processes, where it has convincingly showcased its ability to make precise predictions in complex and changing environments.

Danksagung

Als erstes möchte ich meinen beiden Söhnen danken. Eurer Eifer, Eure Neugierde und Euer Lächeln hat mich jeden Tag von Neuem angetrieben.

Von tiefsten Herzen möchte ich meiner Frau Anne danken. Ohne deine Unterstützung, deine liebevolle Art und deine Liebe wäre ich nicht so weit gekommen und diese Arbeit wäre nicht möglich gewesen.

Ein Dank geht an meinen Doktorvater, Herrn Professor Dr.-Ing. Tobias Melz. Deine Unterstützung, Motivation und fachliche Anleitung haben mir sehr geholfen, dieses Projekt erfolgreich abzuschließen.

Des Weiteren möchte ich mich herzlich bei meinem Korreferenten, Herrn Prof. rer. nat. Oliver Weeger, für seine wertvollen Kommentare und Anregungen bedanken, die wesentlich zur Verbesserung dieser Dissertation beigetragen haben.

Mein besonderer Dank gilt auch Frau Dr. Elena Slomski-Vetter. Deine Expertise, konstruktiven Ratschläge und kontinuierliche Unterstützung haben entscheidend zum Gelingen dieser Arbeit beigetragen.

Ein weiterer Dank geht an alle Studierenden, die mich bei dieser Arbeit begleitet haben. Eure Unterstützung und Einsatz sind von unschätzbarem Wert. Vielen Dank für eure Zeit und Mühe.

Schließlich möchte ich mich bei meinen Kolleginnen und Kollegen des Fachgebiets und bei Moritz Schäfle und Laura Sun vom PMD bedanken. Eure Zusammenarbeit, die zahlreichen Diskussionen und der stetige Austausch haben mich stets motiviert.

Vielen Dank an alle anderen, die mich auf diesem Weg begleitet und vor allem unterstützt haben.

Vorwort

Vor Beginn meiner Promotion habe ich mich als Student des Maschinenbaus der Herausforderung gestellt, einen 3D-Drucker in Betrieb zu nehmen. Zu dieser Zeit waren preisgünstige 3D-Drucker auf dem Markt erhältlich, aber Einschränkungen dieser Technologie waren den meisten Konsumenten einschließlich mir, nicht klar. Mein Ziel war es, eigene Entwürfe und Prototypen zu realisieren. Doch bald wurde ich mit den zahlreichen Prozessfehlern des 3D-Drucks konfrontiert, sodass die gewünschten Ergebnisse ausblieben.

Diese anfänglichen Hindernisse dienten als Ausgangspunkt für meine tiefgehende Auseinandersetzung mit der Problematik der Zuverlässigkeit in der additiven Fertigung in meiner Dissertation. Nach Abschluss meiner Masterarbeit am Fachgebiet Systemzuverlässigkeit, Adaptronik und Maschinenakustik (SAM), der Technischen Universität Darmstadt, erhielt ich die Gelegenheit zur Promotion an demselben Fachgebiet in der Arbeitsgruppe Systemzuverlässigkeit. Die Herausforderung als Student, einen 3D-Drucker erfolgreich zu nutzen, motivierte mich dazu, eine Methode zu entwickeln, die effizient Fehler an 3D-Druckern vorhersagt und vermeidet.

Parallel zu meinen persönlichen Erfahrungen sind bekannte Probleme der additiven Fertigung auch in der Forschung und Industrie dokumentiert. Der allgemeine Trend zur Lösung dieser Probleme entwickelt sich in Richtung einer genauen Beobachtung des Druckprozesses gepaart mit einer schnellen Reaktion im Prozess. Außerdem werden numerische Simulationen verbessert, um aussagekräftige Prognosen für die additive Fertigung erstellen zu können. Bei der genauen Beobachtung des Prozesses mit einer schnellen Reaktion werden jedoch entweder nur einzelne Fehler betrachtet, oder ein Fehler muss aufgetreten sein, um auf den Fehler zu reagieren. Dieser Weg führt nicht zu einer wirklich zuverlässigen Fertigung, was im Kapitel 2 näher behandelt wird. Der Weg, der in dieser Dissertation verfolgt wird, besteht in einer genauen Prognose der Druckfehler und einer gleichzeitigen Vermeidung dieser Druckfehler.

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 1 |
| 2 | Stand der Technik | 5 |
| 2.1 | Additive Manufacturing (AM) | 5 |
| 2.1.1 | Fused Filament Fabrication (FFF) | 6 |
| 2.1.2 | Laser Powder Bed Fusion (LPBF) | 6 |
| 2.1.3 | Systemzuverlässigkeit in dem Additive Manufacturing (AM) | 7 |
| 2.2 | Maschinelles Lernen (ML) | 8 |
| 2.2.1 | Feedforward Neural Network (FFNN) | 9 |
| 2.2.2 | Autoencoder (AE) | 12 |
| 2.2.3 | Recurrent Neural Network (RNN) | 14 |
| 2.2.4 | Knowledge Transfer (KT) im Maschinelles Lernen (ML) | 16 |
| 2.2.5 | Physics Informed Neural Networks (PINN) | 18 |
| 2.2.6 | Latin-Hypercube-Sampling (LHS) | 19 |
| 3 | Methodik | 21 |
| 3.1 | Referenzsysteme zur Methodenveranschaulichung | 26 |
| 3.1.1 | Definition eines Systems | 26 |
| 3.1.2 | Einführung der Referenzsysteme | 28 |
| 3.2 | Pre-training und Fine-tuning (P&F) mit vergleichbaren Daten | 30 |
| 3.2.1 | Regression zur Erhöhung der Trainingsdatenmenge | 30 |
| 3.2.2 | Training von F | 31 |
| 3.2.3 | Anwendung auf Referenzsysteme | 32 |
| 3.2.4 | Zusammenfassung | 34 |
| 3.3 | Pre-training und Fine-tuning (P&F) mit vergleichbaren Daten aus mehreren Systemen | 34 |
| 3.3.1 | Projektion von i auf A_i aus S_i auf einen Vektor | 35 |
| 3.3.2 | Training von F | 36 |

| | | |
|----------|--|-----------|
| 3.3.3 | Anwendung auf die Referenzsysteme | 37 |
| 3.3.4 | Zusammenfassung | 44 |
| 3.4 | Stochastische Neuronale Systemquantifizierung (SNSQ) | 45 |
| 3.4.1 | Annahmen und Voraussetzungen zur Anwendung | 45 |
| 3.4.2 | Definition der Quantifizierung | 48 |
| 3.4.3 | Praktische Umsetzung der Quantifizierung | 51 |
| 3.4.4 | Synthetische Datengenerierung zur Definition von \mathcal{B} | 52 |
| 3.4.5 | Projektion des multidimensionalen Hyperkubus \mathcal{Z} auf einen endlichen Vektor | 56 |
| 3.4.6 | Anwendung auf die Referenzsysteme | 59 |
| 3.4.7 | Zusammenfassung | 66 |
| 4 | Anwendungsfälle | 67 |
| 4.1 | Untersuchung der Druckbettadhäsion (DBA) beim Fused Filament Fabrication (FFF) | 67 |
| 4.2 | 3D-Druck mit Kupfer im Laser Powder Bed Fusion (LPBF) | 74 |
| 5 | Fazit und zukünftige Arbeiten | 79 |
| | Glossary | 83 |

Symbolverzeichnis

- $(\mathbf{x} \rightarrow \mathbf{z})_i$ Alle Beobachtung von System i
- β_t Bias-Term eines Recurrent Neural Network (RNN)
- $\hat{\mathbf{z}}$ Prognostizierte Werte der Ausgabeparameter
- $(X \rightarrow Z)_i$ Eine Beobachtung von System i
- \mathbb{R} Menge der reellen Zahlen
- \mathbf{x} Matrix der Eingabeparameter
- \mathbf{z} Matrix der Ausgabeparameter
- \mathcal{A} Aktivierungsfunktion
- \mathcal{B} Hyperkubus, der alle möglichen \mathcal{Z}_j an der Stelle B_i enthält
- \mathcal{G} Menge aller stochastisch generierten Systeme
- \mathcal{L} Loss-Funktion
- \mathcal{R} Menge aller relevanten Systeme
- \mathcal{U} Menge aller unbekanntem Systeme
- \mathcal{V} Menge aller vergleichbaren Systeme
- \mathcal{Z}_j Hyperkubus, der alle möglichen Ausgaben von System S_i enthält
- μ Erwartungswert
- ω Gewichte eines neuronalen Netzes
- σ Standardabweichung
- $\min_{\omega}(G)$ Minimierung der Funktion G durch Veränderung von ω

| | |
|---------------|--|
| ε | Fehlerterm oder Abweichung |
| A_F | Architektur und Hyperparameter des neuronalen Netzes F |
| A_i | Systemvektor beinhaltet |
| B_i | Verhaltensvektor von System i |
| D | Decoder, die erste Hälfte des Autoencoder (AE) |
| D_G | Definitionsbereich der Funktion G |
| E | Encoder, die zweite Hälfte des Autoencoder (AE) |
| F | Feedforward Neural Network (FFNN) |
| i | Indexvariable |
| k | Indexvariable |
| L_i | Schicht i eines neuronalen Netzes |
| N_i | Anzahl der Neuronen in Schicht i |
| R | Recurrent Neural Network (RNN) |
| S_i | System i |
| V | Eingabevektor |
| m | Größe des Eingabevektors |
| n | natürliche Zahl |
| p | Größe des Ausgabevektors |
| q | Anzahl der Beobachtungen eines Systems |

1 Einleitung

Die Menschheit steht im 21. Jahrhundert vor zahlreichen technischen Herausforderungen, die Innovationen und wissenschaftliches Arbeiten erfordern. Der Klimawandel, eine der drängendsten globalen Krisen unserer Zeit, erfordert nachhaltige und effiziente Lösungen in allen Bereichen, um eine lebenswerte Zukunft zu sichern. Dabei spielt neben der Politik die Forschung eine der wichtigsten Rollen. Da der einfachste Weg über erzielte technologische Fortschritte geht. Schließlich ist die Menschheit nicht bereit, freiwillig auf die Annehmlichkeiten der modernen Welt zu verzichten. Eine entscheidende Komponente für den Erfolg im Kampf für eine nachhaltige Zukunft ist die Systemzuverlässigkeit. Nur durch die Gewährleistung hoher Zuverlässigkeit in den entwickelten Systemen können wir sicherstellen, dass diese nicht nur funktionieren, sondern auch schnell eingesetzt werden können, und effizient zur Lösung aller globaler Probleme beizutragen. Die vorliegende Dissertation widmet sich daher der Erforschung und Verbesserung der Systemzuverlässigkeit. Erklärtes Ziel ist dabei, einen Beitrag zur Bewältigung der technischen Herausforderungen unserer Zeit zu leisten.

Die Entwicklung zuverlässiger Systeme hängt jedoch nicht nur an der Innovationskraft, sondern auch von der Verfügbarkeit und Qualität relevanter Daten ab. Daten bilden die Grundlage für Entscheidungen, Optimierungen und Anpassungen in technischen Prozessen. Im Ingenieurwesen und speziell im Maschinenbau sind relevante Daten wichtig, um die Leistung und Zuverlässigkeit von Systemen zu überwachen. Diese Notwendigkeit führt uns zu den praktischen Problemen, die aus der Erhebung relevante Daten entsteht.

Eine Herausforderung im Maschinenbau und im Ingenieurbereich allgemein ist die Verfügbarkeit relevanter Daten. Im Gespräch mit Vertretern aus Wissenschaft und Industrie werden verschiedene Gründe für nicht existierende Daten genannt, wie die aktuelle Datenschutzgrundverordnung (DSGVO) [1], unternehmensinterne Bedenken, die Befürchtung der Erstellung eines Performance-Index für Mitarbei-

ter, und die Kosten der Datenerhebung in Experimenten oder Produktion. Hinzu kommen Veränderungen durch modernere Geräte und wandelnden Anforderungen in der Datenerhebung sowie den Experimenten im Laufe der Zeit, die einen Vergleich von historischen Daten und aktuellen Daten erschweren. Als ein weiterer Forschungsschwerpunkt dieser Dissertation hat sich daher auch die Verwendung von Methoden des maschinellen Lernens unter der Prämisse sehr weniger relevanter Daten herausgebildet. Dieser Schwerpunkt wird hauptsächlich mit dem Prinzip des Wissenstransfers oder auch Knowledge Transfer (KT) beantwortet, da im Kern dieses Prinzips auch nicht relevante Daten verwendet werden können und somit sehr effizient hinsichtlich der Nutzung von relevanten Informationen ist.

Die vorliegende Dissertation fokussiert sich auf die Systemzuverlässigkeit in Prozessen der additiven Fertigung oder des Additive Manufacturing (AM). Diese Prozesse sind das Fused Filament Fabrication (FFF) und das Laser Powder Bed Fusion (LPBF). Jeder Prozess des Additive Manufacturing (AM) wird als ein komplexes System betrachtet, bestehend aus mehreren unbekanntem mathematischen Zusammenhängen sowie teilweise unbekanntem Eingabeparametern und Ausgabeparametern. Die Konsequenz dieser Betrachtung ist, dass das "Warum" des Systemverhaltens, also die Frage nach dem unbekanntem mathematischen Zusammenhang, nicht geklärt wird. Stattdessen werden Zusammenhänge identifiziert, die zwischen Ein- und Ausgabeparametern des Systems bestehen. Die Fokus der vorliegenden Arbeit liegt auf der Methode zur Optimierung der Zuverlässigkeit von Systemen, nicht aber auf der Klärung von physikalischen Zusammenhängen zwischen eingehenden Einflussgrößen. Auch wenn die Priorität auf der Zuverlässigkeit liegt, werden

Die zentrale Forschungsfrage lautet: Wie kann eine proaktive Fehlervermeidung in der additiven Fertigung umgesetzt werden? Die proaktive Fehlervermeidung zielt darauf ab, den Prozess so zu optimieren, dass der Druckvorgang gar nicht erst fehleranfällig wird. Dies erfordert eine genaue Analyse des Systems und eine präzise Fehlervorhersage, welche Konfigurationen potenziell fehlerhaft sein könnten. Hierbei steht der grundlegende Systemgedanke im Vordergrund, der die genaue Arbeitsweise definiert. Um die Fehlervermeidung in der additiven Fertigung umzusetzen, wird eine Regressionsmethode des maschinellen Lernens entwickelt, die Fehlerwahrscheinlichkeiten schätzt. Mit einer einfachen Optimierung lässt sich so eine Methode ableiten, die Fehler in der additiven Fertigung vermeidet. Der erste Schritt ist das Training eines Künstliche Neuronale Netze (KNN) mithilfe von Vorwissen. Das können Daten aus der Literatur sein oder bekannte oder vermutete physikalische Zusammenhänge. Im zweiten Schritt werden diese trainierten Künstliche Neuronale Netze (KNN) gezielt verändert, um neue mathematische

Zusammenhänge zwischen Ein- und Ausgabeparametern zu finden. Diese neuen Zusammenhänge sind visuell noch vergleichbar zu der Graphischen Auswertung der originalen Kurven des KNN. Aus diesen neuen Zusammenhängen werden dann im letzten Schritt mithilfe der Systemqualifizierung die Verhaltenskurve geschätzt, die am besten zu dem Systemverhalten passt, für das die Prognose erstellt werden soll.

Die entwickelte Methode unterscheidet sich in zwei grundlegenden Prinzipien von anderen Arbeiten in diesem Bereich: Wissenstransfer und Systemquantifizierung. Wissenstransfer ist wichtig, da es in der additiven Fertigung viele Möglichkeiten bezüglich der Ausführung des eigentlichen Prozesses gibt. Hier können unter anderem Material, Geschwindigkeiten oder Temperaturen variieren. Die einfachste Möglichkeit, Informationen zu vermitteln, besteht darin, einen Verweis auf bereits bekannte Informationen herzustellen. Neue unbekannt Informationen müssen hingegen transferiert werden. Die Art und Weise, wie diese bereits bekannten Informationen verwendet werden sollen, wird durch die Systemquantifizierung definiert. In dieser Arbeit werden anhand des Systemverhaltens, also den Systemausgaben, abhängig von den Systemeingaben, Werte einem System in einem Vektor zugeordnet. Durch die Zuordnung von Werten kann dann eine Prognose, also eine Ausgabe abhängig von Eingaben, geschätzt werden.

Die in dieser Dissertation entwickelten Abläufe der Methode sind vielseitig einsetzbar und können auch auf andere Systeme übertragen werden. Die Anwendung der entwickelten Methode erfordert dafür keine besonderen Kenntnisse abgesehen, von Grundlagen in dem speziellen Anwendungsgebiet und keine Expertise in der Systemzuverlässigkeit.

Diese Arbeit widmet sich der Optimierung der Systemzuverlässigkeit durch eine verbesserte Prognose, die auf stochastischer Datengenerierung und Systemquantifizierung basiert. Der Begriff der Optimierung der Systemzuverlässigkeit bezieht sich auf den Umstand, dass die Zuverlässigkeit des Systems durch alle im Prozess auftretenden Fehler beeinflusst wird. Die Optimierung erfolgt daher über die Fehlerprognose, wodurch ein Arbeitspunkt des Prozesses identifiziert wird, der basierend auf der Prognose eine möglichst geringe Fehlerwahrscheinlichkeit aufweist und somit die Zuverlässigkeit des gesamten Systems erhöht.

2 Stand der Technik

2.1 Additive Manufacturing (AM)

Das AM bietet unterschiedliche neue Fertigungsverfahren, die technologisch sich unterscheidende Ansätze mit sich bringt. Während traditionelle Fertigungsverfahren vor allem aus dem Umformen oder einer Form der subtraktiven Fertigung besteht, beinhaltet die additive Fertigung Möglichkeiten ein Werkstück zu erweitern oder komplett aus einem Rohstoff eine neue Form herzustellen [2]. Ein Vorteil ist es, das beim additiven Fertigen im laufenden Prozess auch Hohlräume verschlossen werden können. So kann bei der additiven Fertigung auch in diesen Hohlräumen gefertigt werden. Das AM, bietet daher völlig neue Verfahren zur Herstellung von Produkten. Die als 3D-Druck bezeichneten Verfahren ermöglichen den schichtweisen Aufbau dreidimensionaler Objekte, indem Materialien schichtweise hinzugefügt oder verfestigt werden [3]. Im Gegensatz zu traditionellen Fertigungsverfahren eröffnet die additive Fertigung neue Möglichkeiten für Designflexibilität, schnelle Prototypenerstellung und personalisierte Massenproduktion [4].

Die Wurzeln der AM reichen bis in die 1980er Jahre zurück [5], als die ersten Prototypen von 3D-Druckern entwickelt wurden. Das Fused Filament Fabrication (FFF) wurde 1988 entwickelt und kam 1991 auf den Markt [6]. Seitdem hat sich das Verfahren rasant weiterentwickelt, wobei Fortschritte in Materialwissenschaften, Hardware und Software die Bandbreite der Anwendungen erweitert haben. Ursprünglich auf Prototypen und kleine Bauteile beschränkt, hat sich das AM nun zu einer etablierten Produktionsmethode in unterschiedlichen Branchen entwickelt [4, 7].

2.1.1 Fused Filament Fabrication (FFF)

Beim FFF wird ein Thermoplast in Form eines Filaments in eine heiße Düse gepresst. Diese Düse ist so heiß, dass das Thermoplast seinen Aggregatzustand ändert und aufgeschmolzen wird. Während kontinuierlich Material in die Düse gepresst wird, tritt auf der anderen Seite heißes aufgeschmolzenes Thermoplast aus. Dieser Vorgang ähnelt dem einer Heißklebepistole [5]. Die Düse bewegt sich so über das Druckbett, dass sich aus dem austretenden Thermoplast ein Produkt ergibt. Dies geschieht in mehreren Schichten die nacheinander gebildet werden. Üblicherweise wird das innere des Produktes dabei mit einer Gitterstruktur ausgefüllt. So wird der Vorteil des FFF Verfahrens gegenüber des konventionellen Kunststoffdruckguss klar, es können verschiedene Geometrien mit dem selben Werkzeug erstellt werden und innere Strukturen oder Hinterschneidungen sind mit sehr viel weniger Einschränkungen herstellbar.

Schon heute werden FFF Maschinen in vielen Bereichen eingesetzt. Es existieren bereits kommerzielle Produkte, die in so geringen Stückzahlen gefertigt werden sollen, dass eine konventionelle Fertigung den Preis dieser Produkte zu groß werden lassen würde. Ebenso werden eine Vielzahl an Prototypen mittels FFF hergestellt [8]. Durch die Produktion ergeben sich gewisse Eigenschaften der Produkte. Am stärksten ausgeprägt ist eine Porosität und damit einen Festigkeitsnachteil gegenüber konventionell gefertigten Kunststoffteilen. Gleichzeitig ermöglicht FFF einen deutlich leichteren Aufbau durch Materialeinsparungen und eine freiere Gestaltung der Geometrie. Die FFF-Methode hat somit das Potenzial, traditionelle Fertigungsmethoden zu ergänzen [9].

Betrachtet man lediglich die Kosten, haben sich die Stückzahlen in einer Produktionsserie von Produkten, ab dem Zeitpunkt, an dem die additive Fertigung im Vergleich zur konventionellen Fertigung in Betracht gezogen wird, durch zahlreiche Verbesserungen des FFF Verfahrens in den letzten Jahren in Richtung größerer Stückzahlen verschoben [10, 11]. Dies ist auf zuverlässigere und günstigere 3D-Drucker, sowie verbesserte Methoden zur Fehlerkontrolle [12] auf der anderen Seite zurückzuführen.

2.1.2 Laser Powder Bed Fusion (LPBF)

Beim Laser Powder Bed Fusion (LPBF) wird ein feines Pulvermaterial, wie z.B. Metall oder Kunststoff, auf einer Bauplattform gleichmäßig verteilt. Anschließend wird mit

einem leistungsstarken Laserstrahl selectiv die Pulverschicht aufgeschmolzen, um die erste Schicht des Bauteils zu erzeugen. Sobald eine Schicht vollständig geschmolzen und verfestigt ist, wird die Bauplattform abgesenkt und eine neue Pulverschicht wird aufgetragen. Dieser Prozess wiederholt sich schichtweise, bis das gesamte Bauteil aufgebaut ist [3].

LPBF Maschinen werden bereits in vielen Industriezweigen eingesetzt, darunter die Luft- und Raumfahrtindustrie, die Automobilindustrie und die Medizintechnik. Wie auch beim FFF profitieren Bauteile, die in kleinen Stückzahlen gefertigt werden, besonders von LPBF, da die Herstellungskosten für konventionelle Werkzeuge und Formen entfallen. Gleichzeitig weist die LPBF Verfahren einige Herausforderungen auf, wie etwa die Notwendigkeit einer sorgfältigen Kontrolle der Prozessparameter und der Nachbearbeitung, um optimale mechanische Eigenschaften und Oberflächenqualitäten zu gewährleisten [13].

Die Weiterentwicklung der LPBF Technologie hat in den letzten Jahren zu einer zunehmenden Verbreitung und Akzeptanz in der Serienproduktion geführt. Verbesserungen in der Prozessstabilität, der Materialvielfalt und den Maschinenkosten haben die Wirtschaftlichkeit der LPBF Methode gesteigert, sodass sich die Technologie zunehmend auch für größere Serien lohnt. Dies ist auf die Fortschritte bei den Lasersystemen, den Pulverbeschichtungsmechanismen und der Prozessüberwachung zurückzuführen [11].

2.1.3 Systemzuverlässigkeit in dem Additive Manufacturing (AM)

Einige Methoden des AM haben sich als serientauglich erwiesen. Viele Firmen bieten mit LPBF, FFF oder anderen Prozessen des AM hergestellte Bauteile an, die schon heute problemlos dem Endkunden ausgeliefert werden können. Dennoch bestehen weiterhin Herausforderungen in der additiven Serienfertigung. Beispielsweise zeigen aktuelle Forschungsarbeiten, dass selbst relativ einfache Probleme, wie die Druckbetthaftung noch nicht vollständig gelöst sind. Dies zeigt sich zum Beispiel daran, dass es Literatur gibt, die mit einfachen Versuchsplänen Druckbetthaftung parametrisieren. [14, 15, 16, 17] Die Ergebnisse dieser Studien sind nur empirisch belegt und zeigen keine allgemeinen Zusammenhänge. Es wird jedoch gezeigt, dass es viele Parameter gibt, um einzelne Fehler zu beeinflussen, doch eine allgemeingültige Lösung zur Gewährleistung konsistenter Qualität wurde bisher noch nicht veröffentlicht.

Die Unternehmen in der Serienfertigung des AM sind zwar in der Lage, qualitativ hochwertige Produkte herzustellen, doch eine verbesserte Kontrolle über Druckfehler und eine höhere Zuverlässigkeit der Prozesse könnten die Herstellungskosten pro Produkt weiter senken. Dies könnte durch die Reduktion von Fehldrucken, den geringeren Einsatz von Fachpersonal oder durch kostengünstigere Maschinen erreicht werden [9].

Ein Trend im Bereich der Zuverlässigkeit des AM ist es, während des Druckprozesses Fehler zu erkennen und darauf zu reagieren. Dies kann visuell[12] oder über die Beobachtung von anderen Parametern im 3D-Drucker[18] geschehen.

Eine weitere Möglichkeit besteht darin die Fehler nach der Herstellung zu analysieren, um proaktiv Prognosen zu erstellen. So sollen Druckfehler im Prozess möglichst ganz verhindert werden [19]. Einige Studien arbeiten mit einer zu geringen Datenbasis und versuchen auf dieser Datenbasis aufbauend, mit Methoden aus dem Maschinelles Lernen (ML) zu arbeiten [20]. Zum einen Betrachten diese Studien nur einige wenige Parameter und zum anderen für diese Parameter auch zu wenige Versuche. Es gibt zahlreiche Beispiele für diese Vorgehensweise im FFF für Modelle der Druckbettadhäsion [14, 15, 16, 17], sowie für Modelle der Zugfestigkeit[20, 21, 22].

2.2 Maschinelles Lernen (ML)

Das ML umfasst Algorithmen, die mit Parametern arbeiten und diese Parameter automatisch anzupassen zu können. Diese Algorithmen ermöglichen es, aus vorhandenen Daten zu lernen, Muster zu erkennen und Prognosen zu erstellen, ohne explizit auf die spezielle Anwendung programmiert zu werden. Maschinelles Lernen wird in einer Vielzahl von Anwendungen angewendet, von der Bilderkennung und Sprachverarbeitung bis hin zur Finanzanalyse und medizinischen Diagnose [23]. Durch die nicht explizite Programmierung auf eine spezielle Anwendung, werden Lösungen für Probleme durch mathematische oder numerische Optimierung und nicht von Menschen bestimmt. Daraus folgt, dass mithilfe des Maschinellen Lernens auch Lösungen gefunden werden können, die für Menschen unintuitiv sind oder Lösungen für Probleme finden, die zu komplex sind, um sie von Menschen lösen zu lassen. Als Beispiele für solche komplexen Probleme können hier Bilderkennung und Textverarbeitung dienen. Menschen sind zwar durchaus in der Lage, beispielsweise Inhalte auf Bildern zu erkennen oder Texte zusammenzufassen, hingegen ist es sehr

viel schwieriger ohne ML einen Computer zu programmieren, sodass dieser Inhalte auf Bildern erkennt oder Texte zusammenfasst.

In der vorliegenden Dissertation werden verschiedene Algorithmen des ML miteinander kombiniert und angewendet. Dabei beschränkt sich die Auswahl der Algorithmen auf die Familie der KNN. Diese KNN werden in der heutigen Form schon seit einigen Jahren verwendet [24]. KNN bieten mehrere Vorteile, darunter die einfache Möglichkeit, die Anzahl der Parameter zu beeinflussen, und somit die Komplexität des resultierenden Algorithmus anzupassen. Eine weitere Besonderheit von KNN ist, dass die Verwendung der Parameter durch die Architektur fast nicht beeinflusst werden kann. Der Optimierungsalgorithmus bestimmt die Werte der Parameter ausschließlich anhand der vorhandenen Daten, was zu einer datengesteuerten Optimierung der Modellparameter führt.

Diese Arbeit untersucht die Anwendung von ML-Methoden zur Verbesserung der Zuverlässigkeit in der additiven Fertigung, insbesondere durch die Entwicklung und Optimierung von KNN-Algorithmen, um Fehlerwahrscheinlichkeiten zu schätzen und zu minimieren.

2.2.1 Feedforward Neural Network (FFNN)

Die variable Architektur der Feedforward Neural Network (FFNN) lässt sich wie folgt beschreiben. Alle FFNN bestehen aus Schichten von Neuronen, bei der die Ausgabe aller Neuronen einer Schicht immer von allen Neuronen der nächsten Schicht zum Berechnen der Ausgabe verwendet wird. Diesen Aufbau ermöglicht, dass alle Neuronen einer Schicht parallel berechnet werden können, da es keine Abhängigkeiten zwischen den Werten der Neuronen in einer Schicht gibt. Jedes FFNN F mit einer vorausgesetzten bekannten Architektur A_F , den vorausgesetzten bekannten Parametern oder Gewichten, ω und den Eingaben \mathbf{x} , $\mathbf{x} \in \mathbb{R}^{(m \times n_x)}$ liefert die Ausgaben $\hat{\mathbf{z}}$, $\mathbf{z} \in \mathbb{R}^{(m \times n_z)}$ Diese Ausgabe von F ist eine Annäherung der gewollten Ausgabe \mathbf{z} mit einem Fehler ε .

$$F(\mathbf{x}, A_F, \omega) = F(\mathbf{x}) = \hat{\mathbf{z}} = \mathbf{z} + \varepsilon$$

Die erste Schicht des KNN F wird Eingabeschicht L_0 genannt. Die Neuronen dieser Schicht nehmen nur die Werte der Eingabe an:

$$L_0(\mathbf{x}, \omega) = \mathbf{x}$$

Die Werte aller anderen Neuronen oder der Neuronen in der Schicht $i \neq 0$, berechnen sich anhand der Werte der Neuronen aus der vorangehenden Schicht

$$L_i(L_{i-1}, \omega_{i-1}) = L_i.$$

Dabei ist die Berechnungsformel für alle Schichten L_i gleich. Die Werte der vorangehenden Schicht werden mit den Gewichten für die Schicht ω_i multipliziert und mit einer Aktivierungsfunktion \mathcal{A} transformiert. Durch die Matrixmultiplikation der Ausgabewerte der letzten Schicht und der Gewichte werden die Ausgaben durch Summen berechnet, da die Summen Teil der Matrixmultiplikation sind.

$$L_i = \mathcal{A}(L_{i-1} \cdot \omega_i) \quad (2.1)$$

Die Werte der Neuronen in der letzten Schicht L_{out} , der sogenannten Ausgabeschicht werden als Ausgabe des kompletten KNN F auf die Eingabe \mathbf{x} interpretiert.

$$F(\mathbf{x}) = L_{\text{out}}$$

Aktivierungsfunktionen

Die Aktivierungsfunktionen \mathcal{A} hat die Aufgabe den Wertebereich der Neuronen zu begrenzen. In den meisten Fällen ist der Wertebereich wie folgt definiert: $\mathcal{A}(\mathbb{R}) \in [0, 1]$ Geläufige Funktionen \mathcal{A} für eine Regression sind beispielsweise die Sigmoidfunktion \mathcal{A}_{sig} , der Tangens hyperbolicus $\mathcal{A}_{\text{tanh}}$ und die Rectifier Funktion $\mathcal{A}_{\text{ReLU}}$:

$$\begin{aligned} \mathcal{A}_{\text{sig}}(x) &= \frac{1}{1 + e^{-ax}} \\ \mathcal{A}_{\text{tanh}}(x) &= \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ \mathcal{A}_{\text{ReLU}}(x) &= \max(0, x) \end{aligned}$$

Der Vorteil der Funktionen \mathcal{A}_{sig} und $\mathcal{A}_{\text{tanh}}$ liegt darin, dass diese durchgängig differenzierbar sind. Hingegen $\mathcal{A}_{\text{ReLU}}$ ist an der Stelle 0 nicht differenzierbar. Diese Eigenschaft ist wichtig für den geläufigen Optimierungsalgorithmus der Gewichte, der sogenannte Backpropagation (BP)-Algorithmus.

Loss-Funktionen \mathcal{L}

Eine weitere wichtige Funktion für die Bestimmung der Gewichte ist die Loss-Funktion \mathcal{L} . Mithilfe dieser Funktion wird aus den Fehlern ε eines Datensatzes ($\mathbf{x} \rightarrow \mathbf{z}$) eine einzelne Zahl bestimmt, die aussagt, wie F die gewollte Ausgabe \mathbf{z} vorhersagt. Die Loss-Funktion bildet auch den Ansatz, für die Optimierung der Gewichte im Training. Geläufige Funktionen sind der Mean Squared Error \mathcal{L}_{MSE} , der Root Mean Square Error (RMSE) $\mathcal{L}_{\text{RMSE}}$, der Mean Absolute Error \mathcal{L}_{MAE} und eine Mischung aus \mathcal{L}_{MSE} und \mathcal{L}_{MAE} , die Huber Loss Funktion $\mathcal{L}_{\text{Huber}}$:

$$\begin{aligned}\mathcal{L}_{\text{MSE}}(\varepsilon) &= \frac{1}{n} \sum \varepsilon^2 \\ \mathcal{L}_{\text{MAE}}(\varepsilon) &= \frac{1}{n} \sum |\varepsilon| \\ \mathcal{L}_{\text{RMSE}}(\varepsilon) &= \sqrt{\frac{1}{n} \sum |\varepsilon|} \\ \mathcal{L}_{\text{Huber}}(\varepsilon) &= \begin{cases} \frac{1}{2}(\varepsilon)^2 & \text{für } |\varepsilon| \leq \delta \\ \delta(|\varepsilon| - \frac{1}{2}\delta) & \text{sonst} \end{cases}\end{aligned}$$

Diese Loss-Funktionen sind alle vollständig differenzierbar bis auf \mathcal{L}_{mae} an der Stelle 0. Diese Eigenschaft ist wichtig für den BP-Trainingsalgorithmus. Also der Algorithmus, der die Gewichte ω festlegt, indem er die Ableitung des Loss \mathcal{L} nach den Gewichten aufbaut.

Backpropagation (BP)

Die grundlegende Idee des Trainingsalgorithmus BP ist es, die Ableitung der Loss-Funktion nach den Gewichten $\frac{\partial \mathcal{L}}{\partial \omega}$ und damit dem Einfluss der Gewichte auf die Genauigkeit des Schätzergebnisses des KNN zu verwenden, um den Fehler ε der

Schätzung zu minimieren. So können schrittweise die Gewichte angepasst werden, um das Ergebnis $\hat{\mathbf{z}}$ dem gewollten Ergebnis \mathbf{z} anzupassen. Wenn die Ableitung nicht durch Differenzenquotienten abgeschätzt werden muss, sondern die Ableitung des gesamten KNN aufgestellt werden kann, ergibt sich ein großer Genauigkeits- und Geschwindigkeitsvorteil.

Das Ziel des Trainingsalgorithmus lässt sich allgemein wie folgt definieren, dabei wird die Loss-Funktion mit den Fehlern der Prognose in Abhängigkeit von den Trainingsdaten ($\mathbf{x} \rightarrow \mathbf{z}$) über eine Variation der Gewichte ω minimiert:

$$\min_{\omega} (\mathcal{L}(\varepsilon)) = \min_{\omega} (\mathcal{L}(F(\mathbf{x}) - \mathbf{z})) \quad (2.2)$$

Im Fall des hier verwendeten BP-Algorithmus wird ausgehend vom Ausdruck $\mathcal{L}(F(\mathbf{x}) - \mathbf{z})$ die Ableitung $\frac{\partial \mathcal{L}}{\partial \omega}$ berechnet. Dabei werden die Gewichte in Abhängigkeit des totalen Loss und der Ableitung angepasst. Diese Trainingschritte werden in einer vorher festgelegten Anzahl durchgeführt. Nach einem erfolgreichen Training ist $\mathcal{L}(\varepsilon) \ll 1$ sehr klein und durch weiteres Training wird es nicht signifikant verkleinert. Somit gilt nach erfolgreichem Aufstellen der Architektur und Training der für Ingenieure wichtige Zusammenhang:

$$F_i(\mathbf{x}) = \hat{\mathbf{z}} \approx \mathbf{z} \text{ für alle erfolgreich trainierten } F_n$$

Die Initialisierung der Gewichte ω_i beim Erstellen der F_i erfolgt zufällig. Infolgedessen können sich selbst erfolgreich trainierte F_i unterscheiden, da der BP-Algorithmus typischerweise in unterschiedlichen lokalen Minima konvergiert. Es gibt Untersuchungen, dass die entstehenden F_i mit wachsender Komplexität der Architektur immer unterschiedlicher werden, was durch die Unsicherheit in den Startgewichten bedingt ist.

2.2.2 Autoencoder (AE)

Im ML unterscheidet man zwischen überwachten und unüberwachtem Lernen. Überwachtes Lernen liegt vor, wenn für einen Algorithmus die Ein- und Ausgabewerte vorgegeben werden. Im Fall von FFNN wird dabei \mathbf{x} und \mathbf{z} für ein Training vorgegeben. Somit sollten alle FFNN mit überwachtem Lernen trainiert werden. Bei unüberwachtem Lernen werden nur die Eingabewerte übergeben und der Algorithmus legt die Bedeutung der Ausgabewerte, sowie die Ausgabewerte selber fest. Um

mit FFNN unüberwachtes Lernen durchführen zu können, kann ein FFNN mit einer bestimmten Architektur verwendet werden, ein Autoencoder (AE). Der AE wird wie folgt verwendet:

$$F(\mathbf{x}) = \hat{\mathbf{x}} = \mathbf{x} + \varepsilon$$

Offensichtlich werden zum Training die Ausgabewerte \mathbf{z} nicht benötigt. Die Architektur des AE muss so gewählt werden, dass die Eingabeschicht L_0 und die Ausgabeschicht L_{2n} deutlich größer als die anderen Schichten sind. Die mittlere Schicht L_n soll dabei die kleinste Menge an Neuronen $N_n < N_i, i \neq n$ aller Schichten haben. Die Werte der Neuronen in dieser mittleren Schicht werden Compressed Feature Vector (CFV) genannt. Durch die Existenz einer mittleren Schicht ist die Anzahl an Schichten $2n + 1$ immer ungerade. In dieser Arbeit nimmt die Anzahl der Neuronen zwischen den äußeren Schichten L_0, L_{2n} und der mittleren Schicht L_n exponentiell ab oder zu, das heißt die Anzahl der Neuronen N_i in L_i mit $i \in [0, n]$ lässt sich wie folgt beschreiben:

$$N_i = \begin{cases} N_0 \left\lfloor \frac{N_0}{N_n} \frac{i}{n} \right\rfloor & \text{für } i \in [0, n] \\ N_{n+i} = N_i & \text{für } i \in (n, 2n] \end{cases}$$

Da das Trainingsziel $\mathbf{x} = \hat{\mathbf{x}}$ ist, müssen die Informationen, die in \mathbf{x} sind, durch die mittlere Schicht L_n hindurch, um die letzte Schicht zu erreichen. Da die mittlere Schicht sehr viel kleiner als die Ein- und Augabeschichten sind mit $N_0 \gg N_n$, werden die Informationen der Eingabe komprimiert oder encodiert. Deshalb nennt man die erste Hälfte des Autoencoders L_0 bis L_n Encoder $E(\mathbf{x})$, da hier die Informationen in den CFV encodiert werden. Aus diesen codierten Informationen wird dann im Decoder $D(\mathbf{x})$, also in der zweiten Hälfte des Autoencoders mit den Schichten L_n bis L_{2n} das Output des Autoencoders errechnet.

$$E(\mathbf{x}) = L_n(L_{n-1}(\dots L_0(\mathbf{x})\dots))$$

$$D(\mathbf{x}) = L_{2n}(L_{2n-1}(\dots L_n(\mathbf{x})\dots))$$

Diese Methode des Autoencoders ist etabliert und findet allgemein Anwendung im ML, da es eine Möglichkeit ist, Informationen zu gewinnen, mit denen sich ein Eingabe \mathbf{x} von den anderen Eingaben im Datensatz unterscheidet. Typischerweise wird

der Autoencoder für Eingabewerte \mathbf{x} mit einer hohen Dimensionalität verwendet. Der Autoencoder wird auch als Methode zur Komprimierung verstanden [25].

2.2.3 Recurrent Neural Network (RNN)

Die in dieser Arbeit verwendeten Recurrent Neural Network (RNN) beschränken sich auf KNN, die eine ähnliche Architektur aufweisen wie FFNN. Der Hauptunterschied liegt in den verwendeten Neuronen. Während die Neuronen im FFNN aus einer Matrixmultiplikation und einer Aktivierungsfunktion bestehen (siehe Gleichung (2.1)) und nur von der Ausgabe der Neuronen in der vorangehenden Schicht abhängig sind, sind die rekurrenten Neuronen in RNN auch von der letzten Ausführung dieser Neuronen abhängig. Dabei werden die rekurrenten Parameter, oder auch der Bias-Term β_t für die Ausgabe berücksichtigt.

$$L_{i,t}(L_{i-1,t}, \beta_{i,t-1}) = \mathcal{A}(L_{i-1} \cdot \omega_i + \beta_{i,t-1} \cdot \theta, i)$$

Hierbei ist \mathbf{x}_t der Eingabewert zum Zeitpunkt t , ω sind die Gewichte für die Ausgabewerte der vorherigen Schicht, und θ_i ist die Gewichtsmatrix für die rekurrenten Verbindungen. $L_{t-1,i}$ ist der Zustand der Neuronen in der Schicht i zum vorherigen Zeitpunkt $t - 1$. Die Aktivierungsfunktion \mathcal{A} transformiert die gewichteten Summen, wie bei FFNN.

Durch die Einbeziehung des Zustands der Neuronen aus vorherigen Zeitschritten L_{t-1} können RNN zeitliche Abhängigkeiten und Sequenzinformationen verarbeiten. Dies macht sie besonders geeignet für Aufgaben wie Zeitreihenanalyse, Sprachverarbeitung und andere Anwendungen, bei denen die Reihenfolge und der Kontext der Eingaben wichtig sind.

Ein Beispiel für die Anwendung eines RNN ist die Vorhersage von Zeitreihenwerten. Hierbei wird der aktuelle Wert \mathbf{x}_t zusammen mit dem Zustand L_{t-1} verwendet, um den nächsten Wert in der Sequenz vorherzusagen. Ein RNN kann so lernen, Muster und Trends in den Daten zu erkennen und zukünftige Werte basierend auf der bisherigen Sequenz vorherzusagen.

Um die Trainingseffizienz und Stabilität von RNN zu verbessern, werden oft spezielle Varianten wie Long Short-Term Memory (LSTM) und Gated Recurrent Unit (GRU) verwendet. Diese Varianten fügen zusätzliche Parameter und Mechanismen hinzu,

um besser mit Langzeitabhängigkeiten und dem Problem des verschwindenden Gradienten umzugehen, welches bei einfachen RNN auftritt.

Long Short-Term Memory (LSTM)

LSTM sind eine spezielle Art von RNN, die entwickelt wurden, um Langzeitabhängigkeiten zu erfassen. LSTM besitzen einen internen Speicher, der es ihnen ermöglicht, Informationen über lange Zeiträume hinweg zu behalten. Jede LSTM-Neurone hat drei Hauptkomponenten: den Eingangskanal, den Ausgangskanal und den Vergessenskanal, die jeweils durch spezielle Gate-Funktionen gesteuert werden. Diese Gates regulieren den Fluss von Informationen innerhalb der Zelle und helfen, relevante Informationen beizubehalten und irrelevante zu vergessen.

Die Funktionsweise einer LSTM-Zelle lässt sich durch folgende Gleichungen beschreiben:

$$\begin{aligned}f_t &= \sigma(\omega_f \cdot [h_{t-1}, \mathbf{x}_t] + \beta_f) \\i_t &= \sigma(\omega_i \cdot [h_{t-1}, \mathbf{x}_t] + \beta_i) \\ \tilde{L}_t &= \tanh(\omega_L \cdot [h_{t-1}, \mathbf{x}_t] + \beta_L) \\L_t &= f_t * L_{t-1} + i_t * \tilde{L}_t \\o_t &= \sigma(\omega_o \cdot [h_{t-1}, \mathbf{x}_t] + \beta_o) \\h_t &= o_t * \tanh(L_t)\end{aligned}$$

Hierbei ist f_t das Vergessensgate, i_t das Eingangsgate, \tilde{L}_t der neue Zellinhalt, L_t der aktuelle Zellinhalt, o_t das Ausgangsgate und h_t der Zellzustand. Die Gewichtsmatrizen $\omega_f, \omega_i, \omega_L, \omega_o$ und die Bias-Vektoren $\beta_f, \beta_i, \beta_L, \beta_o$ werden während des Trainings gelernt [26].

Gated Recurrent Unit (GRU)

GRU sind eine vereinfachte Variante von LSTM, die ebenfalls dazu dienen, Langzeitabhängigkeiten zu erfassen, jedoch mit weniger Parametern und somit weniger Rechenaufwand. Eine GRU-Zelle kombiniert das Eingangsgate und das Vergessensgate in ein einziges Update-Gate und verwendet ein Reset-Gate, um den Fluss von Informationen zu steuern.

Die Funktionsweise einer GRU-Zelle kann durch folgende Gleichungen beschrieben werden:

$$\begin{aligned}z_t &= \sigma(\omega_z \cdot [L_{t-1}, \mathbf{x}_t] + \beta_z) \\r_t &= \sigma(\omega_r \cdot [L_{t-1}, \mathbf{x}_t] + \beta_r) \\ \tilde{L}_t &= \tanh(\omega_L \cdot [r_t * L_{t-1}, \mathbf{x}_t] + \beta_L) \\L_t &= (1 - z_t) * L_{t-1} + z_t * \tilde{L}_t\end{aligned}$$

Hierbei ist z_t das Update-Gate, r_t das Reset-Gate, \tilde{L}_t der neue Kandidatenzustand und L_t der aktuelle Zustand. Die Gewichtsmatrizen $\omega_z, \omega_r, \omega_L$ und die Bias-Vektoren $\beta_z, \beta_r, \beta_L$ werden während des Trainings gelernt [27].

Sowohl LSTM als auch GRU haben gezeigt, dass sie in vielen Anwendungen des maschinellen Lernens, insbesondere in der Verarbeitung von sequentiellen Daten, äußerst effektiv sind.

2.2.4 Knowledge Transfer (KT) im Maschinelles Lernen (ML)

Im traditionellen ML wird zur Lösung eines Problems, eine bestimmte Menge an für genau diesem Problem relevanter Daten benötigt. Wenn sich dieses Problem ändert, werden wieder neue relevante Daten für das neue Problem benötigt. Dies ist nicht immer zielführend, da relevante Daten zur Lösung des neuen Problems möglicherweise nicht vorhanden sind. Knowledge Transfer (KT) nimmt hier einen anderen Ansatz, indem versucht wird, die Daten aus dem ersten Problem, also dem Quellsystem zur Lösung des zweiten Problems oder Zielsystem zu verwenden. Hintergrund ist, dass Menschen mithilfe von schon Gelernten neue Probleme schneller und besser lösen können. [28]. Mithilfe von KT soll dieser Prozess im Bereich des ML ebenfalls Verwendung finden. KT soll dabei verbesserte Prognosen erstellen können, während weniger relevante Daten aus dem Zielsystem benötigt werden.

Um KT über verschiedene Aufgaben hinweg erfolgreich durchzuführen, müssen Gemeinsamkeiten identifiziert werden, die in den bisherigen und auch zukünftigen Aufgaben auftauchen. In der Forschung wurden im Laufe der Zeit viele Ansätzen aufgezeigt, die in der Lage sind, diese Gemeinsamkeiten zu identifizieren oder identifizierte Gemeinsamkeiten zu übertragen. Diese Ansätze unterscheiden sich zum einen in der Art und Weise, wie sie generalisieren, wenn sie mit der ersten

Lernaufgabe konfrontiert werden, und in der Art und Weise, wie ihre Generalisierung beeinflusst wird, wenn zuvor erlerntes Wissen übertragen wird. [29]

Transfer Learning (TL) ist ein Unterbegriff des KT. Beim TL wird ein Modell, das für eine Aufgabe trainiert wurde, für eine verwandte Aufgabe verwendet, während KT allgemein die Übertragung von Wissen thematisiert. Die Verwendung von KT im Allgemeinen bringt einige Vorteile. Im traditionellen ML erfordert die Lösung einer bestimmten Aufgabe, dass ausreichend relevante Daten speziell für diese Aufgabe vorliegen. Mithilfe von KT kann diese Voraussetzung umgangen werden, indem Wissen aus einer anderen, ähnlichen Aufgabe genutzt wird, um die Zielaufgabe zu bewältigen. Der Vorteil von KT ist, dass für die eigentlich relevante Aufgabe weniger Daten benötigt werden. Als Nachteil ist hier aufzuführen, dass der Algorithmus deutlich komplexer und damit fehleranfälliger wird. Außerdem kann es dazu kommen, dass der Prozess des KT komplett fehlschlägt und es keinen Vorteil durch die höhere Komplexität und die vergleichbaren Daten gibt [30].

Für das TL können drei Unterarten unterschieden werden. Inductive, Transductive und Unsupervised TL. Inductive beschreibt den Vorgang des TL wenn Quell- und Zielsysteme in der selben Domain sind. Dabei sind die Daten des Zielsystems in der Regel mit Labeln versehen. Transductive beschreibt TL wenn Quell- und Zielsysteme in unterschiedlichen Domänen sind. Hierbei sind die Daten der Quelldomäne im Normalfall ebenfalls gelabelt. Unsupervised TL beschreibt TL für den Fall, dass die Daten von Quell- und Zielsystem komplett ohne Label auskommen. [31]

Ein weiterer Teil des KT ist das Self Supervised Learning (SSL). Bei diesem unüberwachten Verfahren für Daten ohne Label werden Label für die Daten innerhalb der Methode generiert, um Methoden des überwachten ML anwenden zu können. Dieses neuere Verfahren zeigen eine verbesserte Performance gegenüber TL. [31]

Der Großteil der aktuellen Veröffentlichungen im Bereich KT bezieht sich auf komplexere Daten, wie Text und Bild, bei denen sehr viele Trainingsdaten nötig sind. [32, 33, 31, 30] Allerdings gibt es auch Beispiele und Techniken, bei denen mit einfachen Datenstruktur, wie einer einfachen mathematischen Gleichung gearbeitet wird.

Eine einfache Technik im Bereich des TL ist das Pre-training und Fine-tuning (P&F), bei dem der KT durch zwei separate konventionelle Training von KNN durchgeführt wird. Dabei wird im ersten Schritt das KNN auf das vorhandene Quellsystem trainiert. Das KNN lernt in diesem Schritt die Einflüsse der einzelnen Parameter, sowie die Einflüsse der Interaktionen von Parametern des Quellsystems. Diese Einflüsse werden

beim Training auf das Zielsystem angepasst. Wenn sich Quell- und Zielsystem aber ähnlich sind, müssen diese Einflüsse nur geringfügig angepasst werden und für das Zielsystem gibt es mit sehr wenigen relevanten Daten eine gute Näherung [31].

Model-Agnostic Meta-Learning (MAML) ist eine Technik im Bereich des Meta-Lernens, die darauf abzielt, maschinelle KNN so zu trainieren, dass sie schnell an neue Aufgaben angepasst werden können, mit nur wenigen zusätzlichen Trainingsdaten. Der zentrale Gedanke hinter MAML ist, das Modell nicht direkt für ein bestimmtes System zu optimieren, sondern es so zu trainieren, dass es eine gute Ausgangsbasis für viele verschiedene Systeme bietet. Dadurch kann das Modell bei neuen Systemen schneller lernen. In der Meta-Training-Phase wird das Modell auf mehrere Quellsysteme hinweg trainiert, dabei hat jedes System eigene Trainings- und Testdaten. Das Modell wird so trainiert, dass es nach möglichst wenigen Trainingsschritten eine möglichst gute Prognose für alle Systemen erbringt. Auf diese Weise wird ein KNN erstellt, das die Systeme sehr gut generalisiert. Neue ähnliche Systeme sollen so gut von dem KNN erledigt werden können. Im zweiten Schritt wird das KNN auf das Zielsystem optimiert [34].

In [35] werden verschiedene Methoden des TL verglichen. Außerdem werden Metamodelle dieser Methoden erstellt und ebenfalls verglichen. Über die verschiedenen Szenarien erreichen die Metamodelle aus verschiedenen Methoden konstant einen Loss, der nur ein Fünftel bis ein Zehntel so groß ist, wie der Loss der einzelnen Methoden. Ebenfalls erreicht MAML einen kleineren Loss als P&F, aber der Unterschied beträgt immer weniger als 10%. Diesen bis zu 10% kleineren Loss erreicht P&F gegenüber den anderen Methoden. Allerdings ist der Trainingsaufwand, sowie die Komplexität von MAML ungleich höher. Hier muss in jedem Trainingsschritt des FFNN weitere Trainingsschritte für jedes einzelne Quellsystem durchgeführt werden [34].

2.2.5 Physics Informed Neural Networks (PINN)

Physics Informed Neural Networks (PINN) sind eine innovative Methode im Bereich des maschinellen Lernens, die physikalische Gesetzmäßigkeiten, wie sie durch Differentialgleichungen beschrieben werden, in den Trainingsprozess von neuronalen Netzen integriert. Anders als bei rein datengetriebenen Ansätzen nutzen PINN zusätzlich physikalisches Wissen über das zu modellierende System, um die Modellgenauigkeit zu verbessern und die Generalisierung auf komplexe physikalische Szenarien zu gewährleisten [36]. Mithilfe von Methoden des TL ergibt sich eine

verbesserte Trainierbarkeit von PINN [37].

Im Gegensatz zur anfänglichen Definition [38] entwickelt sich die Bedeutung von PINN mittlerweile zu einer umfassenden Bedeutung. Anfänglich war die Verwendung einer speziellen Loss-Funktion vorgesehen, die eine Differentialgleichung enthält. Mittlerweile beinhaltet der Begriff teilweise alle KNN, die auf irgendeine Weise physikalische Informationen verwenden [39, 40, 41].

Im Zusammenhang mit der additiven Fertigung gibt es verschiedene Möglichkeiten, um ein PINN zu verwenden. Ein häufige verwendeter Ansatz verbindet physikalische Gesetzmäßigkeiten als Differentialgleichungen mit Beobachtungsdaten [42]. Die Beobachtungsdaten machen hier den Ansatz über das PINN effizient [43]. Weitere Möglichkeiten der Anwendung umfassen hybride Modelle aus KNN und physikalischen Modellen, die Vereinfachungen enthalten und somit die Realität nicht genau abbilden [40]. Ein weiterer Ansatz verbindet die Ergebnisse der physikalischen Modelle mit realen Beobachtungen zu einer P&F-Methode [39].

2.2.6 Latin-Hypercube-Sampling (LHS)

Latin-Hypercube-Sampling (LHS) ist eine statistische Sampling-Methode, die zur effizienten Abtastung von hochdimensionalen Parameterbereichen eingesetzt wird. Beim LHS wird jeder Parameterbereich in gleich große Intervalle aufgeteilt. In die Mitte dieser Intervalle wird jeweils ein Sample genommen. Im Gegensatz zu rein zufälligen Ansätzen wie Monte Carlo Sampling sorgt LHS durch die Unterteilung jedes Parameterbereichs in gleich große Intervalle dafür, dass die Proben gleichmäßig über den jeden Parameterraum verteilt sind [44]. Dazu existieren Erweiterungen, um den Versuchsplan möglichst orthogonal zu erstellen, also so, dass die Samples möglichst gleichmäßig im Parameterraum verteilt sind [45].

3 Methodik

Die additive Fertigung hat in den letzten Jahren zunehmend an Bedeutung gewonnen, insbesondere in Bereichen wie der Luft- und Raumfahrt, dem Automobilbau und der Medizintechnik [46]. Trotz ihrer Vorteile, wie der Flexibilität und der Reduktion von Materialverbrauch, steht die Technologie jedoch vor einer zentralen Herausforderung: der Systemzuverlässigkeit. Durch eine verbesserte Systemzuverlässigkeit wird im Produktionsprozess weniger Ausschuss produziert, weniger manuelles Eingreifen nötig und somit die Kosten pro Teil reduziert. [9]

Es wird eine Methode vorgestellt, die darauf abzielt, die Systemzuverlässigkeit in der additiven Fertigung zu verbessern. Dabei steht nicht die detaillierte Analyse eines spezifischen Fertigungsprozesses im Vordergrund, sondern vielmehr wird die Additive Fertigung als ein Gesamtsystem mit Zielwerten betrachtet, das durch bestimmte Parameter beeinflusst wird. Ziel der Methode ist es somit nicht, ein spezielles neues Feature oder eine Entdeckung hervorzubringen. Stattdessen kombiniert die entwickelte Methode vorhandenes Wissen über bestehende Fertigungsprozesse zu Modellen, die mit minimalen Daten möglichst präzise und objektive Prognosen treffen können.

Diese neue Methode basiert auf KT, um mögliche Fehler im Fertigungsprozess frühzeitig vorherzusagen. Indem vorhandene Daten effizient eingesetzt werden, kann die Zuverlässigkeit des Gesamtsystems signifikant verbessert werden, auch wenn nur begrenzte Informationen zur Verfügung stehen. Im Fokus steht somit ein Ansatz, der das KI-Training mit wenigen Daten optimiert und gleichzeitig die Fertigungsprozesse stabilisiert.

Im Kern der vorliegenden Dissertation geht es um das effiziente KI-Training mit wenigen Daten im ingenieurtechnischen Bereich. Denn wie im vorangehenden Kapitel erläutert, können gerade im ingenieurtechnischen Bereich Daten aus der Nutzungsphase nur aufwendig, teuer und zeitintensiv gesammelt werden. Allerdings sind diese Daten wichtig, um Fragen zu beantworten, die die Qualität, Sicherheit und

Funktionalität während der Nutzung betreffen und Schwachstellen am Produkt zu beheben. Ein Grund ist, dass technische Geräte häufig nicht vom Hersteller betrieben werden. Der Hersteller hat durch die kaum vorhandene Kommunikation zwischen ihm und dem Endkunden nur eine sehr begrenzte Möglichkeit vom Endkunden Nutzungsdaten zu erhalten. Ein weiterer Grund ist, dass die vorhandenen Daten, die während der Produktion anfallen und somit im Unternehmen vorhanden sind, nur schwer Rückschlüsse auf die Nutzung zulassen.

Eine weitere Herausforderung kann entstehen, wenn in einem Unternehmen schon seit vielen Jahren verschiedene Versuche durchgeführt werden. In der Versuchsdurchführung ändern sich im Laufe der Jahre Prüfmaschinen, Produktgeometrien, Anforderungen sowie Techniken zur Datenaufzeichnung. In solchen Fällen der iterativen Wissensgenerierung durch Versuche bestehen die Daten aus verschiedenen Versuchsplänen, die unterschiedliche Fragen beantworten sollten. Die eigentliche Herausforderung besteht darin, die vielfältigen Daten zusammenzufassen und Erkenntnisse zu gewinnen. Diese Probleme von ungenügenden oder ungleichmäßigen Datensätzen sollen angegangen werden.

Die Lösung für das Verwenden von Daten aus unterschiedlichen Quellen, die in der vorliegenden Dissertation verwendet wird, ist der Knowledge Transfer (KT). Die Möglichkeiten des KT lassen sich dabei an dem folgenden Beispiel veranschaulichen. Nehmen wir an, wir möchten einer fremden Person das Konzept eines umgangssprachlichen Cabrios erklären. Wenn wir dafür das Wissen um Autos der fremden Person verwenden können, ist es sehr viel leichter ein Cabrio zu erklären, da es umgangssprachlich ein Auto ohne Dach ist. Wenn wir für die Erklärung aber nicht das Konzept von Autos verwenden können, dann wird die Erklärung sehr langwierig. Diese Stärke des KT soll genutzt werden, um die vollständige Analyse eines Systems durch umfangreiche Versuche zu umgehen. Stattdessen wird Wissen transferiert, um ein System mit wenigen Versuchen zu analysieren, basierend auf der Annahme, dass sich die Systeme sehr ähneln. Dies ermöglicht es, Prognosen für technische Systeme im Bereich des AM effizient zu erstellen.

Die vorliegende Dissertation startet mit der Prämisse, mit wenigen relevanten Daten ein Modell für Regressionen auf Basis KNN zu erstellen. Relevante Daten werden dabei als jene Daten definiert, die direkt aus dem System stammen, das mit der Regression vorhergesagt werden soll. Sie sind für das Training und die Prognose des Modells unverzichtbar, da ohne diese Daten nichts Konkretes über das System bekannt ist. Relevante Daten ermöglichen auch den Einsatz verschiedener Regressionsmethoden. KNN bieten sich besonders an, da sie nicht nur komplexe

Zusammenhänge darstellen können, sondern auch modular und flexibel erweiterbar sind. Diese Eigenschaften machen sie zu einer idealen Methode, um das Modell sowohl mit vielfältigen Datenquellen als auch TL-Ansätzen zu kombinieren, um die Vorhersagegenauigkeit zu erhöhen.

Eine Möglichkeit, um mit wenigen Daten ein Neuronales Netz zu trainieren, bietet die P&F Methode. So trainierte Netze verwenden vergleichbare Informationen oder Daten. Diese Daten werden als Datensätze definiert, die aus ähnlichen Systemen oder Prozessen stammen, aber nicht direkt von dem System, das vorhergesagt werden soll. Diese Daten weisen ähnliche Eingangs- und Ausgangsgrößen auf und können dazu verwendet werden, das Modell zu trainieren und mögliche Zusammenhänge des Zielsystems abbilden. Vergleichbaren Daten ergänzen so relevante Daten in einem Regressionsmodell. Diese vergleichbaren Daten, die zum Training verwendet werden, sind zum Beispiel komplett synthetisch oder unter anderen Bedingungen erstellt wurden, wie das eigentliche System. Beispielsweise wurde ein anderer Werkstoff verwendet oder es herrschten andere Umgebungsbedingungen.

Diese P&F-Methode ist identisch zu einer von Kapusuzoglu et al. [40] vorgeschlagen PINN-Methode zur Anwendung beim FFF-Prozess. Der Unterschied liegt hier nur in der Herkunft der Daten. Diese soll bei Kapusuzoglu et al. [40] aus physikalischen Modellen stammen. Generell lässt sich jedoch sagen, dass Informationen nicht mehr nur als relevante Daten vorhanden sein müssen, um beim Training mit KNN verwendet werden zu können. Informationen aus ähnlichen Quellen können mithilfe eines KT-Ansatzes zur Lösung beitragen.

P&F als Methode zur Erstellung von Regressionsmodellen funktioniert unter kontrollierbaren Bedingungen einer kontrollierbaren Umgebung. Im Falle einer dynamisch und unkontrollierten Umgebung, die zu unbekanntem und sogar zeitlich veränderlichen Einflussfaktoren führt, sind die Prognosen eines mit P&F trainierten KNN nicht brauchbar, da relevante Daten schnell zu vergleichbaren Daten werden. Ein anschauliches Beispiel für diese Herausforderung lässt sich anhand einer falsch gestellten Uhr illustrieren. Die Prognosen der Uhr zur Uhrzeit sind nicht brauchbar, da sie dem unbekanntem Einflussfaktor „Differenz zur wahren Uhrzeit“ unterliegen. Aber dennoch ähneln die Prognosen der wahren Uhrzeit. Damit sind die Prognosen zur Uhrzeit zwar Informationen, die das System betreffen, aber nicht relevant. Dieses Problem kann jedoch offensichtlich mit weiteren P&F gelöst werden, indem die korrekte Uhrzeit bekannt ist, allerdings muss so, nach jedem Ereignis, dass die Uhrzeit verstellt ein neues Training durchgeführt werden. Die neu entwickelte Methode der Stochastische Neuronale Systemquantifizierung (SNSQ) lässt

sich ebenfalls innerhalb eines Gedankenexperiments veranschaulichen, in dem verschiedene hypothetische Uhren mit unterschiedlichen Differenzen erdacht werden. Die Methode bestimmt dann dynamisch die passende Differenz zur wahren Uhrzeit, indem sie anhand von Beobachtungen der wahren Uhrzeit die richtige Korrektur ableitet. Die eigentliche Innovation der vorliegenden Arbeit besteht also darin, so viele vergleichbare Informationen zu erzeugen, dass ein Funktionsraum an möglichen Prognosefunktionen erstellt werden kann, in dem sich die relevante Prognosefunktion ebenfalls befindet. Diese relevante Prognosefunktion wird anhand schon aufgetretener Systemantworten ausgewählt und für Prognosen verwendet.

In Abbildung 3.1 ist der Ablauf der im Rahmen der vorliegenden Dissertation erarbeiteten SNSQ-Methode schematisch dargestellt. Auffällig in der Abbildung ist die inhärente Komplexität der Methode. Für den Nachweis eines Proof-of-Concept ist jedoch eine möglichst einfache Herangehensweise erforderlich, um die Methode verständlich und effizient zu testen. Eine übermäßige Komplexität würde die Nachvollziehbarkeit erschweren und die Wahrscheinlichkeit für Fehler erhöhen. Daher wurden bewusst Methoden mit geringer Komplexität für die einzelnen Schritte der SNSQ-Methode gewählt, die es ermöglichen, die grundlegende Funktionsfähigkeit der SNSQ-Methode zu demonstrieren. In zukünftigen Arbeiten können diese Komponenten durch komplexere Methoden ersetzt werden, um die Performance weiter zu optimieren, beispielsweise hinsichtlich Laufzeit oder Prognosegenauigkeit.

Zu Beginn der SNSQ-Methode wird eine Informationsfusion mit der P&F-Methode aus ähnlichen Daten aus der Literatur und Beobachtungen der relevanten Systeme durchgeführt. Dies ist inductive TL. Die genaue Herangehensweise dieser Methode in der vorliegenden Dissertation ist in den Kapiteln 3.2 und 3.3 dargestellt.

Der nächste Schritt umfasst die Anwendung des Stochastische Datengenerierung mit Neuronalen Netzen (SDNN) an den Modellen der P&F-Methode. Dabei werden die Ausgaben der Modelle zufällig verändert, um eine größere Datenbasis für die nächsten Schritte verwenden zu können, auch wenn die absolute Menge an Informationen nicht ansteigt. Diese unsupervised TL-Methode wird im Unterkapitel 3.4.4 erläutert.

Um eine Prognose für ein bestimmtes Systemverhalten aus dieser größeren Datenbasis zu erstellen, wird eine Quantifizierung des Systemverhaltens erstellt. Dies ist eine verbreitete Methode, um Prognosen für unbekannte Systeme zu erstellen, beispielsweise werden User von Online-Plattformen anhand verschiedener Parameter charakterisiert, um ihnen eine Webseite mit personalisierten Inhalten darstellen zu können. In dieser Arbeit werden erfolgte Experimente und damit Beobachtungen

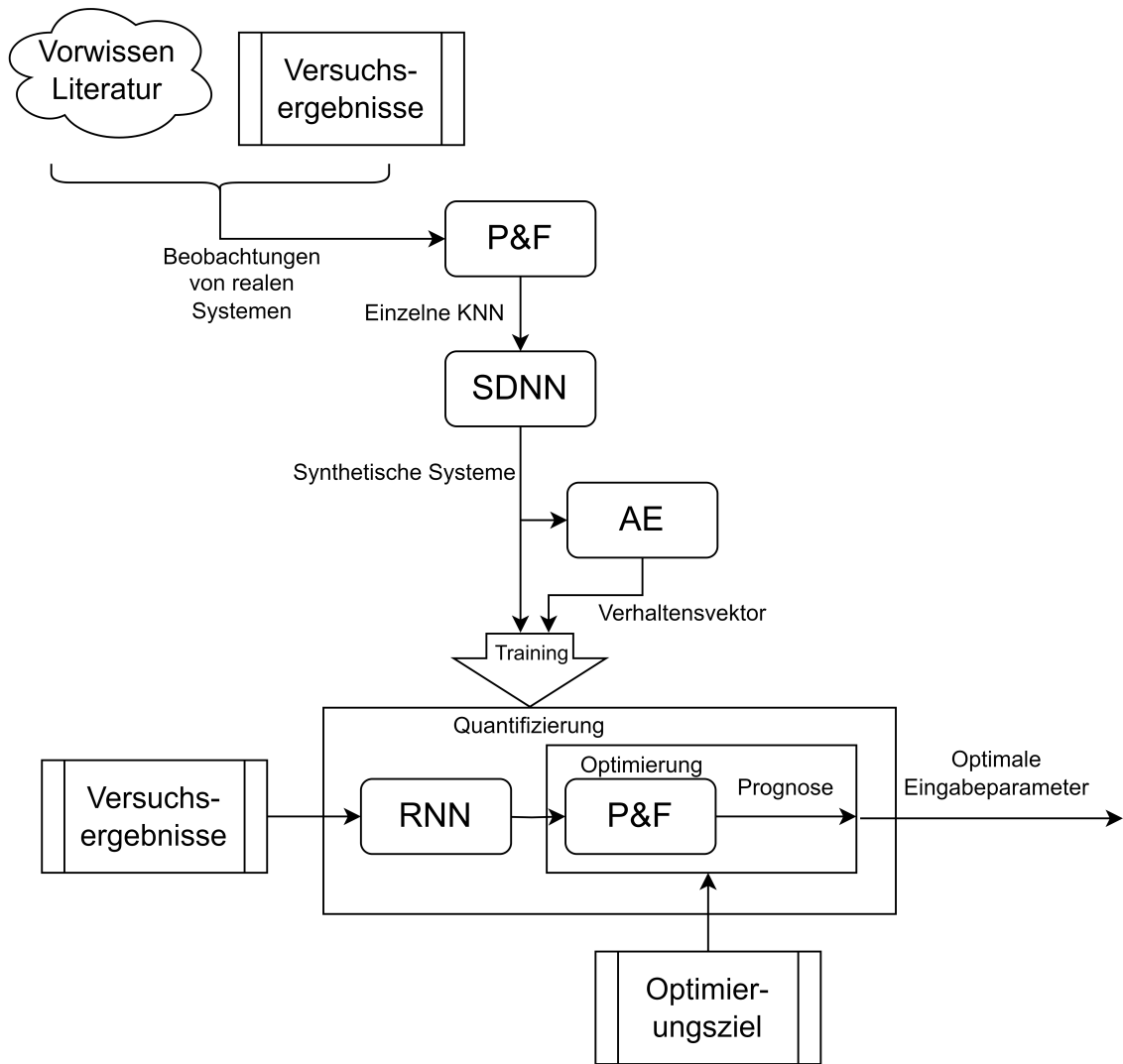


Abbildung 3.1: Die Stochastische Neuronale Systemquantifizierung (SNSQ) dargestellt als Flussdiagramm

eines Systems verwendet, um eine Schätzung des Systemverhaltens zu erstellen. Vergleichbare Informationen können so mit der richtigen Charakterisierung auch für das Training verwendet werden [47, 48]. Diese Quantifizierung wird durchgeführt, indem ein Verhaltensvektor eingeführt wird, der jedes Systemverhalten in komprimierter Form beschreibt. Dieser Verhaltensvektor wird mithilfe eines AE, wie im Unterkapitel 3.4.2 beschrieben ist, definiert.

Die eigentliche Quantifizierung unbekannter Systeme erfolgt von einem RNN, das anhand von vorhandenen Beobachtungen einem unbekanntem System einen Verhaltensvektor zuordnet. Mit diesem Verhaltensvektor wird im nächsten Schritt noch die eigentliche Prognose oder Regression durchgeführt. Diese induktive TL-Methode ist im Unterkapitel 3.4.3 dargestellt.

3.1 Referenzsysteme zur Methodenveranschaulichung

Die im Folgenden vorgestellten Methoden werden nicht nur durch mathematische Erläuterungen, sondern auch anhand von einfachen Referenzsystemen veranschaulicht, die an dieser Stelle eingeführt werden. Dies dient der Nachvollziehbarkeit der vorgestellten Methoden.

3.1.1 Definition eines Systems

Allgemein wird über jedes System S_i der Eingabevektor $X \in \mathbb{R}^m$ dem Ausgabevektor $Z \in \mathbb{R}^p$ zugeordnet. Jede dieser Zuordnungen $(X \rightarrow Z)_i$ für die Eingabe- und Ausgabevektoren bekannt sind, wird im Rahmen dieser Arbeit eine Beobachtung von System S_i genannt.

Der Zusammenhang zwischen X_i und Z_i mit dem System S_i ist also wie folgt:

$$S_i(X_i) = Z_i$$

Alle q Eingabe- und Ausgabevektoren aus den q verwendeten Vektoren lassen sich zusammenfassen zu den Matrizen:

$$\mathbf{x} \in \mathbb{R}^{q \times m} \quad \text{und} \quad \mathbf{z} \in \mathbb{R}^{q \times p}$$

Alle im Rahmen der Methoden verwendeten Beobachtungen von S lassen sich schließlich schreiben als

$$(\mathbf{x} \rightarrow \mathbf{z})_i.$$

Die Methoden, die in dieser Arbeit vorgestellt werden, sind von einer PINN-Methode [39] abgeleitet, die wiederum von einer TL-Methode, dem P&F, abgeleitet sind. Im Gegensatz zur ursprünglichen PINN-Methode wird dabei kein physikalisches Domainwissen verwendet. Bei dieser Methode nach Kapusuzoglu [39] werden Beobachtungen verwendet, die von vergleichbaren Systemen sind. Vergleichbare Systeme sind Systeme S_V , die den gleichen physikalischen Gesetzen und Zusammenhängen unterliegen, sich jedoch in ihren absoluten Parametern oder Skalierungen unterscheiden. Beispielhaft sind etwa Maschinen verschiedener Fabrikate und Größe untereinander vergleichbar, solange sie den selben Prozess durchführen. In vergleichbaren Systemen sind die Strukturen der Zusammenhänge ähnlich, während die absoluten Werte sich unterscheiden. Im Gegensatz dazu stehen die relevanten Systeme S_R . Diese stehen im direktem Bezug zum Zielsystem und stimmen in allen wesentlichen physikalischen und parametrischen Eigenschaften mit diesem überein. Sie bilden die Zielumgebung oder den Anwendungsfall. Bei der Anwendung dieser P&F-Methode werden experimentelle Ergebnisse, also empirische Daten aus vergleichbaren Systemen verwendet. Diese Daten können aus verschiedenen Quellen stammen, wie etwa eine analytische Hypothese oder experimentelle Daten aus einer Veröffentlichung. Auch wenn die Entstehung dieser Daten gut dokumentiert oder nachvollziehbar ist, sind diese Daten nicht immer reproduzierbar für den Anwender dieser Methode.

Für das unbekannte vergleichbare System S_V lassen sich alle verwendeten Beobachtungen als vergleichbare Daten schreiben

$$(\mathbf{x} \rightarrow \mathbf{z})_V$$

Das zweite System, das verwendet wird, ist das tatsächlich relevante System S_R . Alle zu diesem System vorhandenen Beobachtungen lassen sich wie folgt beschreiben:

$$(\mathbf{x} \rightarrow \mathbf{z})_R$$

Für dieses relevante System sollen Prognosen erstellt werden. Zu diesem Zweck, wird FFNN F modelliert, das S_R annähert. Dieses FFNN F , das trainiert werden soll, erstellt Prognosen \hat{Z} mit dem Eingabevektor X :

$$F(X) = \hat{Z}$$

Das Modell ist vom System zu unterscheiden: Während das System ein beobachtbares reales oder simuliertes Verhalten einer Entität beschreibt, dient das Modell als Näherung des Systemverhaltens. In dieser Arbeit kann das Modell grundsätzlich als „Black Box“ aufgefasst werden, da es ohne explizite Kenntnis der inneren Struktur des zugrunde liegenden Systems arbeitet, Das System definiert sich also über das Systemverhalten, während das Modell dieses annähert.

3.1.2 Einführung der Referenzsysteme

Zur Validierung der vorgeschlagenen Methode werden sogenannte Referenzsysteme definiert. Die Referenzsysteme bestehen aus einem vergleichbaren System S_V und einem relevanten System S_R . Durch diesen Aufbau lässt sich das Verhalten der Methode beim Transfer zwischen vergleichbaren, aber nicht identischen physikalischen Systemen untersuchen. Die Referenzsysteme wurden willkürlich gewählt, um die Funktionsweise der Methode zu demonstrieren. Sie sind nicht als Einschränkung der Übertragbarkeit auf andere Anwendungen zu verstehen, sondern dienen ausschließlich der Illustration der vorgestellten Methode. Das vergleichbare System nimmt Eingabevektoren $X \in \mathbb{R}^m$ mit $m = 1$ an, wobei jedem X ein Ausgabevektor $Z \in \mathbb{R}^p$ mit $p = 1$ zugeordnet wird. Das System S_V wird durch folgende Formel beschrieben:

$$S_V(X) = x_0^3 + 0,4x_0^2 - 0,5x_0 + 0,1$$

Von diesem vergleichbaren System sind beliebig viele Beobachtungen im Bereich $[0, 1]$ verfügbar, da es sehr gut bekannt ist. Das relevante Referenzsystem S_R wird ebenfalls mit $m = 1$ und $p = 1$ definiert. Das System hat folgende Gleichung:

$$S_R(X) = 0,5x_0^3 + 0,4x_0^2 - 0,5x_0 + 0,2$$

Im Vergleich zum vergleichbaren System wird der kubische Anteil halbiert und der konstante Anteil verdoppelt. Der ähnliche Kurvenverlauf beider Funktionen ist auch in Abbildung 3.2 gut sichtbar. Beide Systeme haben ein Extrema im Definitionsbereich $D_S = [0; 1]$, sind schlecht linear modellierbar und haben eine große positive Steigung im Bereich $[0,8; 1]$.

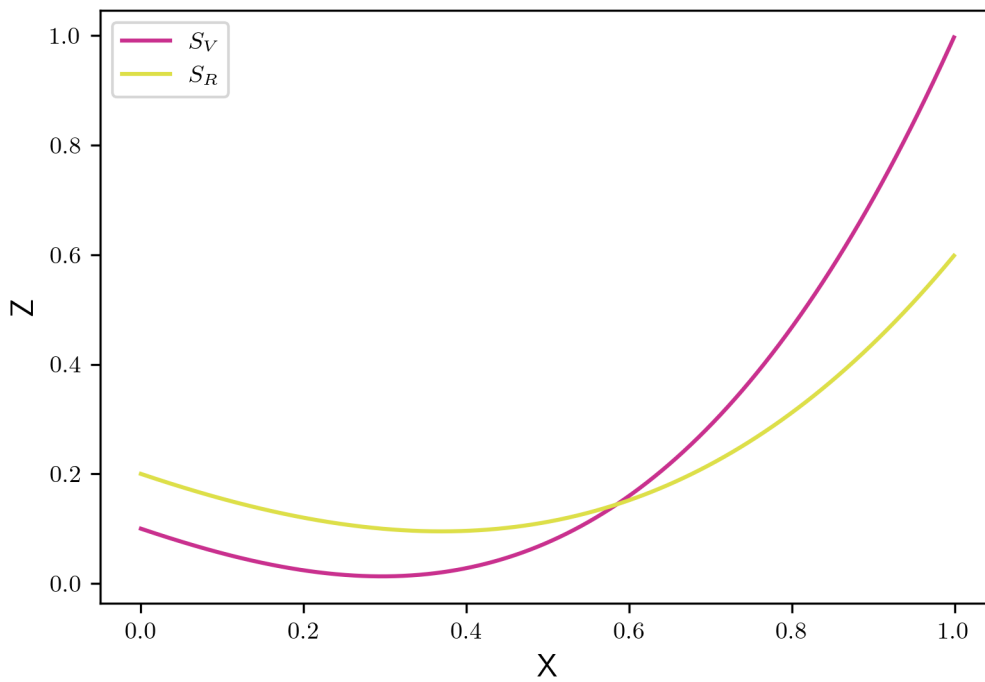


Abbildung 3.2: Darstellung der Referenzsysteme mit dem vergleichbaren System S_V und dem relevanten System S_R .

3.2 Pre-training und Fine-tuning (P&F) mit vergleichbaren Daten

Im ersten Schritt der Anwendung der SNSQ-Methode wird ein einfaches TL-Verfahren eingesetzt. Dabei wird das FFNN F mithilfe von S_V vortrainiert, um anschließend S_R vorherzusagen. Beobachtungen aus S_V werden schließlich verwendet, um die Vorhersage von S_R zu verbessern.

3.2.1 Regression zur Erhöhung der Trainingsdatenmenge

Um die vergleichbaren Daten $(\mathbf{x} \rightarrow \mathbf{z})_V$ für das Training des Modells F verwenden zu können, muss zunächst eine detaillierte Analyse der vorliegenden Daten durchgeführt werden. Dabei ist es wichtig, zwischen verschiedenen Datensätzen und Szenarien zu unterscheiden, um sicherzustellen, dass nur aussagekräftige und konsistente Daten zum Einsatz kommen. Insbesondere bei hochdimensionalen Daten oder komplexen Eingabeverteilungen kann eine einfache Datenzuordnung irreführend sein. Hier müssen Aspekte wie Datenverteilung, Varianz und eventuelle Abhängigkeiten berücksichtigt werden, um ein verzerrtes Modelltraining zu vermeiden. Eine unzureichende Datenmenge pro Eingabeparameter könnte das Modell anfällig für ein Overfitting machen, was die Generalisierungsfähigkeit auf neue, unbekannte Daten stark einschränken würde. Die genannten Herausforderungen sind für die Qualität des Modelltrainings entscheidend, bleiben jedoch außerhalb des Fokus dieser Arbeit und werden daher nicht näher betrachtet.

Ein weiterer entscheidender Faktor ist die Anzahl der verfügbaren Datenpunkte. Dies stellt sicher, dass das Modell genügend Struktur erkennt oder Verallgemeinerungen treffen kann und somit in der Lage ist, auch bei neuen, bisher unbekanntem Eingabewerten sinnvolle Vorhersagen zu treffen. Es gibt keine fixe Regel zur Anzahl der benötigten Datenpunkte, aber je komplexer das System und je mehr Freiheitsgrade vorhanden sind, desto mehr Daten werden benötigt, um das Modell adäquat zu trainieren. Je komplexer das Systemverhalten (beispielsweise durch nicht-lineare Beziehungen oder viele Freiheitsgrade), desto mehr Datenpunkte sind notwendig, um zuverlässige Ergebnisse zu erzielen. Hierbei spielt auch die Art der Datenverteilung eine Rolle. Liegt eine ungleichmäßige Verteilung vor, kann das Modell in bestimmten Bereichen einen größeren Fehler in der Prognose aufweisen.

Falls die Datenmenge nicht ausreicht, sollte ein Regressionsmodell verwendet wer-

den, um zusätzliche Datenpunkte zu generieren. Dies wird insbesondere dann notwendig, wenn die beobachteten Daten nicht direkt für das Training ausreichen, beispielsweise aufgrund einer zu geringen Anzahl oder einer ungleichmäßigen Verteilung. Eine Vielzahl von Regressionsmodellen kann hierfür in Frage kommen, angefangen bei einfachen linearen Modellen bis hin zu komplexeren Ansätzen wie Polynomregressionen oder neuronalen Netzwerken. Die Wahl des Modells hängt dabei stark von der Natur der Daten sowie den zugrundeliegenden Zusammenhängen ab.

In der Literatur wird häufig mit der Veröffentlichung der Beobachtungen eines Systems auch ein Modell veröffentlicht, das dieses System modelliert. Diese Modelle ermöglichen es, auf Basis vorhandener Datenpunkte neue, synthetische Daten zu generieren, die das Training unterstützen. Sollte ein solches Modell nicht verfügbar sein, ist es sinnvoll, ein eigenes Regressionsmodell zu erstellen. Dies kann unter Berücksichtigung der spezifischen Merkmale des Datensatzes erfolgen, um die Abdeckung des Parameterraums zu verbessern. Mit Modellen, die auf simplen Methoden, wie der linearen Interpolation beruhen, konnten gute Ergebnisse erzielt werden, sofern die Daten ausreichend gleichmäßig verteilt sind [41]. Die guten Ergebnisse dieser Methode können allerdings nicht als allgemeingültig betrachtet werden. Letztlich hängt der Erfolg der Datenvermehrung durch Interpolation stark von der Struktur des zugrunde liegenden Modells und den Beziehungen zwischen den Parametern ab.

3.2.2 Training von F

Eine weitere Herausforderung kann auftreten, wenn für die Beobachtungen aus dem vergleichbaren System Parameterangaben fehlen. Es gilt also $m_Y \leq m_R$, mit m als Anzahl der Parameter eines Systems. Diese Parameter sind in der Veröffentlichung nicht angegeben oder sind nicht Teil der analytischen Herleitung. In der praktischen Anwendung des Trainings vom FFNN F , müssen diese Parameter mit Werten versehen werden. Zweckmäßig ist es, diese Parameter mit Zufallswerten zu belegen. So werden diese Parameter vollständig von F ignoriert, da sie keinen Einfluss auf die Zielgröße haben. Im folgenden Training am relevanten System können diese Parameter aber dennoch verwendet werden. Der potentielle Einfluss dieser Parameter kann somit erst im abschließenden Training beachtet werden.

Wenn genug Datenpunkte $(\mathbf{x} \rightarrow \mathbf{z})_V$ vorhanden sind, können die Gewichte ω von F im Training optimiert werden nach

$$\min_{\omega} (\mathcal{L}(F(\mathbf{x}_V) - \mathbf{z}_V)). \quad (3.1)$$

Im Anschluss wird das vortrainierte F mit Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$, von dem relevantem System $S_{\mathcal{R}}$, das vorhergesagt werden soll, trainiert.

$$\min_{\omega} (\mathcal{L}(F(\mathbf{x}_{\mathcal{R}}) - \mathbf{z}_{\mathcal{R}})) \quad (3.2)$$

3.2.3 Anwendung auf Referenzsysteme

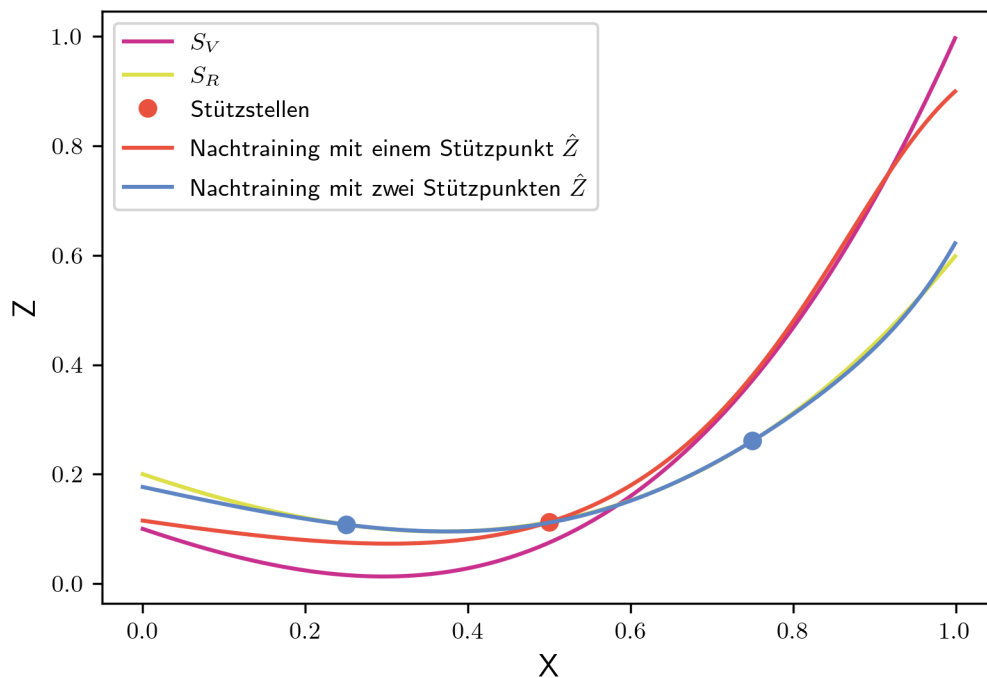


Abbildung 3.3: Ergebnisse des Trainings aus der Gleichung (3.1) und anschließend aus der Gleichung (3.2) mit jeweils einer und zwei Stützstellen. Die Stützstellen sind durch Punkte gekennzeichnet.

Der erste Schritt in der Anwendung besteht in der Klärung der Frage, ob für das vergleichbare System S_V eine Regression benötigt wird, um die Anzahl der Da-

tenpunkte für das Training zu erhöhen. Da für das vergleichbare Referenzsystem beliebig viele Messungen verfügbar sind, wird keine Regression benötigt und es kann direkt mit dem Training gemäß Gleichung (3.1) begonnen werden. Für das Training mit den relevanten Beobachtungen gemäß Gleichung (3.2) werden die Beobachtungen nach dem LHS durchgeführt. Wie in Abbildung 3.3 zu sehen ist, verändert sich durch die Nähe der beiden Systeme bei der einzelnen Beobachtung bei $x = 0.5$ die Kurve mit einer einzelnen Stützstelle nur sehr geringfügig zur Kurve des Referenzsystemes S_V im oberen Bereich. Im unteren Bereich hingegen liegt die Kurve visuell betrachtet dicht bei S_V . Mit zwei Beobachtungen hingegen liegt die blaue Kurve von F deutlich auf der gelben Kurve von S_R . Bei dem LHS wird der Definitionsbereich $[0; 1]$ in gleich große Intervalle aufgeteilt und jeweils die Mitte der Intervalle als Beobachtung ausgewählt. Im Falle von 2 Beobachtungen liegen diese hier bei:

$$X = \begin{bmatrix} \frac{0+0,5}{2} \\ \frac{0,5+1}{2} \end{bmatrix} = \begin{bmatrix} 0,25 \\ 0,75 \end{bmatrix}. \quad (3.3)$$

In der Tabelle 3.1 sind für jeweils eine andere Anzahl an Stützstellen die RMSE aufgetragen. Bemerkenswert ist hierbei, dass der RMSE für 2 Stützstellen niedriger als für eine größere Anzahl an Stützstellen ist. Eine Erklärung könnte sein, dass der KT mit einer größeren Anzahl an Stützstellen schlechter wird. Die Interpretation dazu wäre, dass das FFNN F mit fortschreitendem Training das Vortraining schlicht vergisst. Der visuelle Eindruck aus Abbildung 3.3 bezüglich der Anzahl der Stützstellen und der Abweichung von S_R bestätigt sich. Der RMSE von einer Stützstelle ist mehr als 10 mal größer als mit zwei Stützstellen.

| Anzahl der Stützstellen | Stützstellen (X) | $\mathcal{L}_{\text{RMSE}}$ |
|-------------------------|------------------------------|-----------------------------|
| 1 | (0.5) | 0.299 |
| 2 | (0.25, 0.75) | 0.017 |
| 3 | (0.1667, 0.5, 0.8333) | 0.018 |
| 4 | (0.125, 0.375, 0.625, 0.875) | 0.018 |

Tabelle 3.1: Stützstellen und zugehörige RMSE-Werte für verschiedene Anzahlen von Stützstellen

3.2.4 Zusammenfassung

Nach dem Training mit den vergleichbaren Daten werden während des zweiten Trainings durch den Backpropagation (BP) die Gewichte von F so angepasst, dass die Prognose \hat{Z} von F während des zweiten Trainings auf einem direkten Weg die Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$ für das relevante System enthält. Diese Eigenschaft liegt in der Art und Weise, wie der BP arbeitet. Diese inkrementelle Verschiebung der Ausgabe von F auf dem direkten Weg ist eine Möglichkeit, praktisch ein TL durchzuführen. Dieses TL funktioniert, indem das von F trainierte und damit angeeignete Wissen in Form einer mathematischen Zuordnung $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{V}}$ auf einem direktem Weg so verändert wird, dass es nun $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$ enthält. Eine direkte Folge davon ist, wenn das System $S_{\mathcal{V}}$ sehr viel mehr Beobachtungen hat, dass in der Regel die Prognosen von F für $S_{\mathcal{R}}$ besser werden, also einen kleineren Fehler haben.

Allerdings gibt es zwei negative Aspekte, die bei dieser Art von TL auftreten können:

1. Offensichtlich geht beim zweiten Training in Gleichung (3.2) der Einfluss des ersten Trainings auf F aus Gleichung (3.1) mit größer werdender Menge an Beobachtungen von $S_{\mathcal{R}}$ verloren.
2. Für den Fall, dass es mehr als jeweils ein System im Bereich der vergleichbaren bzw. relevanten Beobachtungen gibt, kann das Training sehr leicht in einem Overfitting enden. Dies geschieht, wenn die Beobachtungen zusammengefasst werden und bei unterschiedlichen Systemen F die Loss-Funktion nur minimieren kann, durch Erkennung des Systems anhand der Eingabewerte.

Diese beiden negativen Aspekte lassen sich beheben, indem die Methode auf beliebig viele Systeme verallgemeinert wird.

3.3 Pre-training und Fine-tuning (P&F) mit vergleichbaren Daten aus mehreren Systemen

Zur Verallgemeinerung sollen mehrere Systeme $\{S_1, S_2, \dots\}$ betrachtet werden. Jedes System ist entweder ein relevantes System, also ein System, für das Prognosen erstellt werden sollen, oder ein vergleichbares System, also ein System aus der Literatur oder auch aus anderen Quellen, für das Beobachtungen bekannt sind. Die Gruppen der Systeme werden demnach wie folgt definiert:

$$\mathcal{V} = \{S_i \mid S_i \text{ ist ein vergleichbares System}\}$$
$$\mathcal{R} = \{S_i \mid S_i \text{ ist ein relevantes System}\}$$

Analog zur Methode aus dem vorangegangenen Kapitel 3.2 mit nur einem vergleichbaren System soll das FFNN F in zwei Schritten trainiert werden. Im zweiten Schritt sollen jedoch möglichst keine Informationen aus den \mathcal{V} verloren gehen. Dafür wird ein zusätzlicher Parameter zum Training eingeführt, der Systemvektor A_i . Dieser Vektor beinhaltet eine projizierte Form von i aus S_i , die für ein KNN leichter zu verarbeiten ist. Das KNN kann so während des Trainings zwischen den einzelnen Systemen unterscheiden. Im zuvor beschriebenen zweistufigen Trainingsprozess werden ausschließlich die Beobachtungsdaten verarbeitet, ohne dass Informationen über deren spezifische Systemzugehörigkeit genutzt werden. Diese Zugehörigkeit wird stattdessen durch den Vektor A_i dargestellt.

3.3.1 Projektion von i auf A_i aus S_i auf einen Vektor

Eine Möglichkeit eine Zahl i zu einem Vektor A_i umzuwandeln, ist die binäre Darstellung von i . Diese Darstellung wird dann als Vektor verwendet, also die Ziffern von i im Zahlensystem mit der Basis 2. Die Basis 2 bietet eine kompakte Darstellung, die gut von KNN interpretiert werden kann [49]. Diese wird wie folgt definiert:

$$A_i = (a_1, a_2, \dots, a_n),$$
$$a_k \in \{0, 1\} \quad \text{mit} \quad i = \sum_{k=1}^n a_k \cdot 2^{n-k}$$

Eine weitere Möglichkeit der Projektion von i auf A_i erfolgt durch das One-hot-Encoding (OHE). Dabei werden alle Elemente des Vektors A_i zu 0 gesetzt, bis auf das i -te Element. Dieses wird zu 1 gesetzt. Diese Möglichkeit ist auch sehr gut von KNN verarbeitbar[50], allerdings nicht ganz so kompakt wie die binäre Darstellung.

$$A_i = (a_1, a_2, \dots, a_n),$$

$$a_k \in \{0, 1\} \quad \text{mit} \quad a_k = \begin{cases} 1, & \text{wenn } k = i \\ 0, & \text{sonst} \end{cases}$$

3.3.2 Training von F

Begrenzte Datenquellen in \mathcal{V} sollen weiterhin mithilfe von Regressionsmodellen erweitert werden, wie es im vorherigen Abschnitt beschrieben ist. Diese Systeme dienen dazu, das FFNN F vorzutrainieren. Dafür gelten weiterhin die beschriebenen Anforderungen an die Datenbasis. Dies umfasst die Anzahl der Beobachtungen, sowie die Verteilung der Beobachtungen im Definitionsbereich von F , sowie eine annähernde Orthogonalität. Dies ist wichtig, da eine ungleichmäßige Verteilung der Trainingsdaten zu einem Modell führen könnte, das in bestimmten Regionen des Definitionsbereichs stark überrepräsentierte Muster lernen könnte. Dadurch wird das Modell weniger generalisierbar und könnte in schlecht abgedeckten Regionen falsche Vorhersagen treffen. Eine fehlende Orthogonalität der Daten kann dazu führen, dass das Modell falsche Korrelationen zwischen den Eingangsparametern lernt.

Das Vortraining von F mit den Beobachtungen der Systeme S_i aus \mathcal{V} erweitert sich mit dem Vektor A damit folgendermaßen:

$$\min_{\omega} (\mathcal{L}(F(\mathbf{x}_{\mathcal{V}}, A_i) - \mathbf{z}_{\mathcal{V}})). \quad (3.4)$$

Im Anschluss wird das vortrainierte F mit Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$ von den Systemen in \mathcal{R} trainiert. Hier wird ebenfalls analog die binäre Schreibweise oder das Ergebnis von OHE von i aus dem dazugehörigen S_i verwendet.

$$\min_{\omega} (\mathcal{L}(F(\mathbf{x}_{\mathcal{R}}, A_i) - \mathbf{z}_{\mathcal{R}})). \quad (3.5)$$

Durch Variation von A_i lassen sich nach dem Training Prognosen für verschiedene relevante Systeme mit dem selben KNN wie folgt erstellen.

$$F(\mathbf{x}, A_i) = \mathbf{z} \quad (3.6)$$

Falls neue Beobachtungen auftauchen oder sich aus anderen Gründen $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$ ändert, lassen sich neue Beobachtungen in \mathcal{R} durch erneutes Training einfügen.

Dieses TL beruht nicht auf demselben direkten Weg wie das TL mit nur jeweils einem vergleichbaren und relevanten System. Zum Verständnis muss hierbei klar sein, dass die Ausgaben von F mit A von einem System aus \mathcal{R} schon im ersten Schritt trainiert werden, ohne dass relevante Trainingsdaten verwendet oder vorhanden sind. Der zentrale Unterschied zur klassischen Form liegt darin, dass mehrere verschiedene Quellen aus \mathcal{V} verwendet werden, um allgemeine Muster zu lernen, die über alle Systeme hinweg gültig sind. Von Vorteil ist hierbei, dass Gemeinsamkeiten für alle Systeme unabhängig von A trainiert werden und Besonderheiten eines Systems immer abhängig von A in F berechnet werden. Die Identifikation dieser Gemeinsamkeiten erfolgt implizit durch das gleichzeitige Training auf mehreren Systemen, wobei das Modell nur solche Strukturen stabil lernen kann, die systemübergreifend auftreten. Als Extremfall existiert natürlich die triviale Lösung: «alle Gemeinsamkeiten gleich null zu setzen und alle Ausgabewerte abhängig von A zu definieren» immer und unabhängig von den verwendeten Daten. Allerdings müssen dann für jedes A die Gewichte ω angepasst werden. Da die gemeinsam verwendeten Gewichte einen Einfluss auf alle Trainingsdaten haben, ist der berechnete Einfluss auf den Loss deutlich größer und sollte dementsprechend stärker vom Trainingsalgorithmus angepasst werden. Die triviale Lösung ist also unwahrscheinlich. Im zweiten Trainingsschritt wird die vortrainierte Ausgabe dann wie in der vorherigen Methode auf einem direkten Weg so verändert, dass sie $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$ enthält. Der Unterschied besteht also darin, dass nur von F antrainierte gemeinsame Muster aus $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{V}}$ zu $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{R}}$ verändert werden.

3.3.3 Anwendung auf die Referenzsysteme

Um diese Methode an den Referenzsystemen anzuwenden, müssen die beiden bisherigen Referenzsysteme durch ein weiteres System ergänzt werden. Dies geschieht dennoch, obwohl die Referenzsysteme als Minimalbeispiel gedacht sind und nur die Funktionsweise der Methode ergänzend gezeigt werden soll. Die beiden bisherigen Systeme werden umbenannt um in S_1 und S_2 . Diese sind jetzt Teil der Gruppe \mathcal{V} :

$$\mathcal{V} = \{S_1(x_0), S_2(x_0)\}, \text{ mit}$$
$$S_1 = x_0^3 + 0,4x_0^2 - 0,5x_0 + 0,1$$
$$S_2 = 0,5x_0^3 + 0,4x_0^2 - 0,5x_0 + 0,2$$

Für die Gruppe \mathcal{R} wird ein System S_3 benötigt:

$$\mathcal{R} = \{S_3(x_0)\} \text{ mit}$$
$$S_3 = 0,25x_0^3 + 0,6x_0^2 - 0,3x_0 + 0,2$$

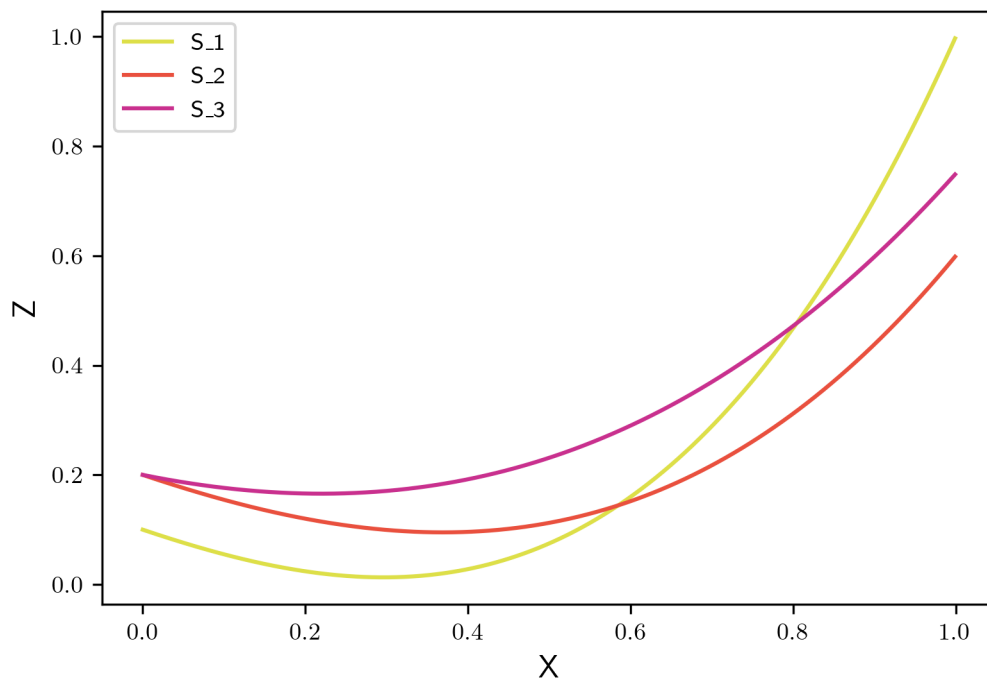


Abbildung 3.4: Grafische Darstellung der drei Systeme des erweiterten Referenzsystems für die P&F-Methode mit mehr als zwei Systemen als funktionaler Zusammenhang

In der Abbildung 3.4 sind die drei Systeme erneut dargestellt. Visuell ausgewertet sind die drei Systeme graphisch sehr ähnlich. Die gelbe Kurve von S_3 verläuft insgesamt etwas oberhalb der beiden Kurven aus \mathcal{V} und zeigt eine geringere Steigung.

Der erste Schritt, der durchgeführt wird, ist die Projektion von i auf den Vektor A_i . Für das Referenzsystem wird eine gleichmäßige Schrittweite von $\frac{1}{1000}$ für x und die binäre Darstellung von i verwendet. Der Trainingsdatensatz für das erste Training mit allen Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{V}}$ sieht wie folgt aus:

$$(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{V}}$$

$$\left(\begin{array}{c} \left[\begin{array}{ccc} 0 & 0 & 1 \\ 0.001 & 0 & 1 \\ 0.002 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 1.000 & 0 & 1 \\ 0 & 1 & 0 \\ 0.001 & 1 & 0 \\ 0.002 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 1.000 & 1 & 0 \end{array} \right] \rightarrow \left[\begin{array}{c} S_1(0) \\ S_1(0.001) \\ S_1(0.002) \\ \vdots \\ S_1(1) \\ S_2(0) \\ S_2(0.001) \\ S_2(0.002) \\ \vdots \\ S_2(1) \end{array} \right] \end{array} \right)_{\mathcal{V}}$$

Die binäre Darstellung von 0 in A kann für das Auslesen von Gemeinsamkeiten verwendet werden und wird daher für das Training nicht verwendet. Hierbei werden alle Anteile von A zu null gesetzt und keine Eigenheiten der Systeme dargestellt. Ein Unterschied zum OHE ist nur schwer zu erkennen, da $(1)_{10} = (01)_2 = (001)_{\text{OHE}}$ und $(2)_{10} = (10)_2 = (010)_{\text{OHE}}$ gilt. Zur Unterscheidung der Darstellungsarten wird ein Index verwendet: numerische Indizes stehen für Zahlensysteme, der Index OHE für die Darstellung mittels OHE. Erst bei $(3)_{10} = (11)_2 = (100)_{\text{OHE}}$ wird der Unterschied klar. Der Vollständigkeit halber sind dennoch alle Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{\mathcal{V}}$ mit dem OHE wie folgt dargestellt:

$$\left(\begin{array}{c} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.001 & 0 & 0 & 1 \\ 0.002 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \\ 1.000 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0.001 & 0 & 1 & 0 \\ 0.002 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \\ 1.000 & 0 & 1 & 0 \end{bmatrix} \\ \rightarrow \\ \begin{bmatrix} S_1(0) \\ S_1(0.001) \\ S_1(0.002) \\ \vdots \\ S_1(1) \\ S_2(0) \\ S_2(0.001) \\ S_2(0.002) \\ \vdots \\ S_2(1) \end{bmatrix} \end{array} \right)_{\mathcal{V}}$$

Im nächsten Schritt wird das Training mit den Beobachtungen aus den vergleichbaren Systemen durchgeführt gemäß Gleichung (3.4). Der TL kann über $i = 0$ ausgelesen werden. Da dieser durch das Training von F nicht immer gleich ausfällt, wird F für dieses Diagramm insgesamt 10 mal trainiert und der TL ausgelesen. Die Kurven des TL werden mit den beiden Systemen aus \mathcal{V} in der Abbildung 3.5 dargestellt.

In den speziellen Systemen für unser Referenzsystem scheint das TL häufig an der X -Achse gespiegelt und in den positiven Bereich verschoben zu sein. Was dazu führt, dass das TL nicht komplett falsch ist, aber dennoch passt die Steigung des TL nicht zu der Steigung der Systeme. Einige wenige Kurven des TL scheinen nicht diesem Muster zu folgen, sondern folgen den Kurven der Systeme. Da beim Training keine Daten für $i = 0$ vorgegeben wurden, handelt es sich bei dem TL um unüberwachtes Lernen. Die Darstellung der Ergebnisse muss also für einen Menschen nicht intuitiv sein und kann trotzdem verwendbares Wissen enthalten.

In Abbildung 3.6 wird eine alternative Methode zur Darstellung des TL verwendet. Dabei wird der Vektor A komplett mit der Zahl 1 gefüllt. In der binären Darstellung entspricht das $i = 2$. Bei Verwendung von $i = 2$ zur Bestimmung des TL zeigt sich ein komplett anderes Bild als für $i = 0$ in Abbildung 3.5. Allerdings sind die Kurven in Abbildung 3.5 mit denselben F erzeugt worden, wie die Kurven in Abbildung 3.6. Durch die binäre Darstellung können die beiden Vektoren A_1 und A_2 addiert werden, um A_3 zu erhalten. Dabei werden im Wesentlichen die gemeinsamen Eigenschaften der beiden Systeme S_1 und S_2 kombiniert. Insgesamt sind die 2. Ableitungen der Kurven, sowie die Werte visuell deutlich besser. Allerdings sind die Steigungen der Kurven S_1 und S_2 in einer anderen Größenordnung. Eine endgültige Antwort auf

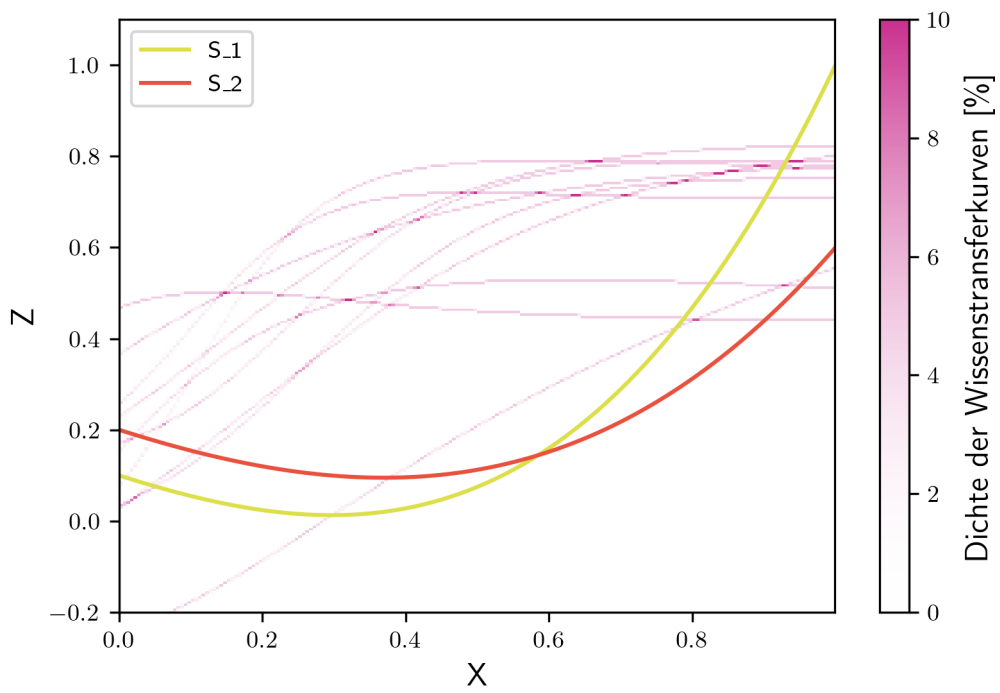


Abbildung 3.5: Darstellung der beiden Systeme aus \mathcal{V} dem TL von 10 aufgestellten FFNN F .

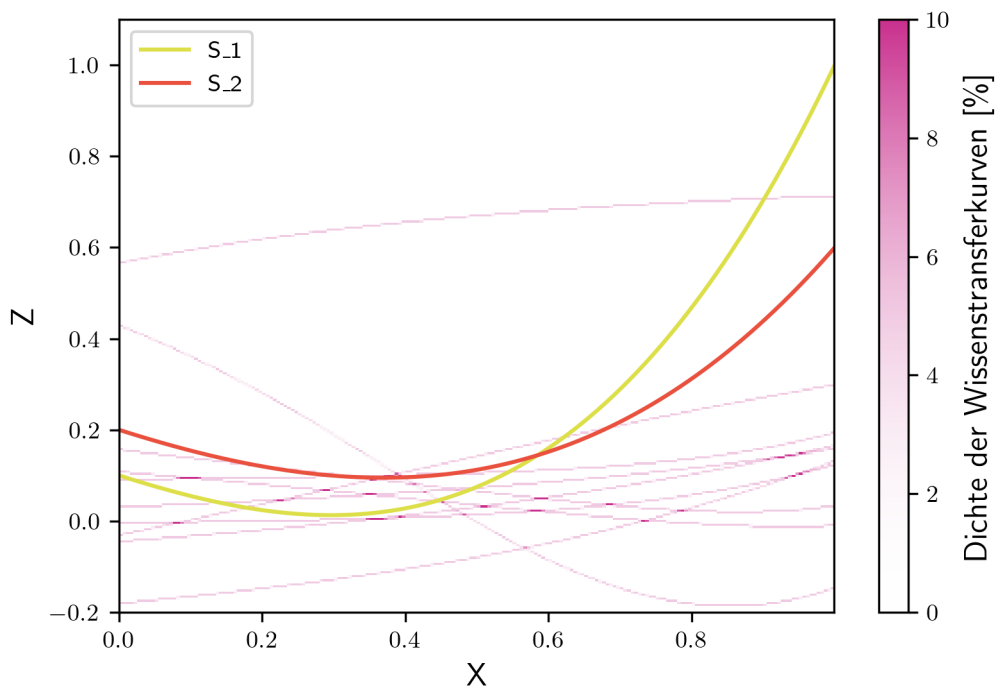


Abbildung 3.6: Darstellung der beiden Systeme aus \mathcal{V} mit 2D-Histogramm des alternativen TL von 10 aufgestellten FFNN F .

den TL, den von den FFNN F vollzogen wird, durch das Training kann mit diesen Abbildungen nicht gegeben werden. Allerdings ist der TL stark vom Zufall abhängig, wie in den Abbildungen ersichtlich.

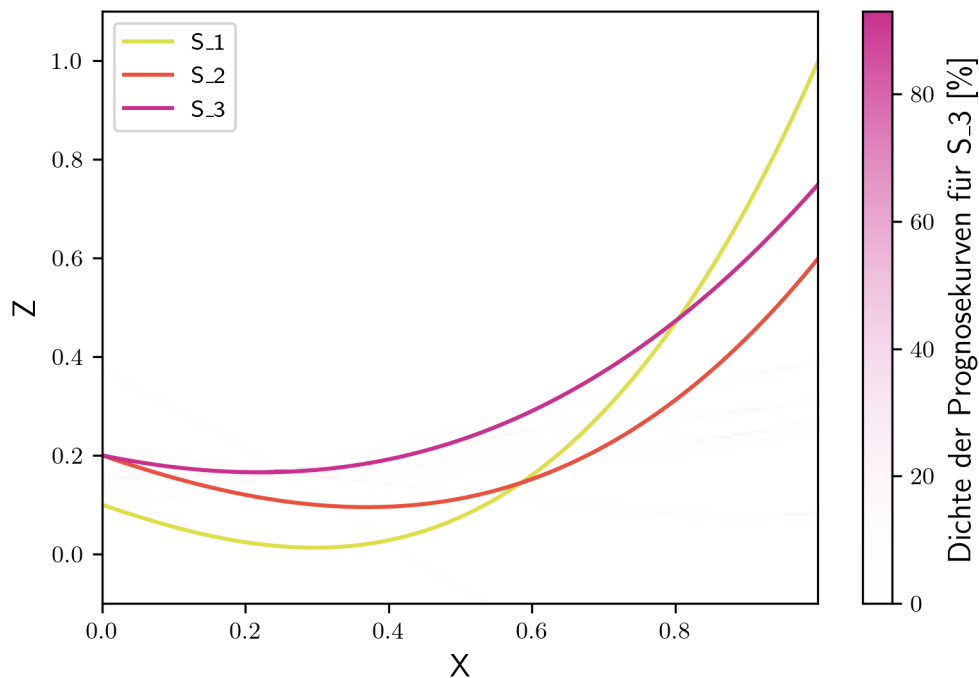


Abbildung 3.7: Darstellung der Ausgabe von FFNN F mit 2D-Histogramm nach dem 2. Training mit 2 Beobachtungen

Im anschließenden Training mit den beiden Beobachtungen aus \mathcal{R} vom System S_3 ergibt sich dann folgendes Histogramm. Die für Abbildung 3.7 trainierten 10 FFNN F werden nach der Gleichung (3.5) erneut trainiert mit den folgenden zwei Beobachtungen, dabei leitet sich X von den angesprochenen LHS aus Gleichung (3.3) ab. Ebenfalls wird hier die binäre Darstellung von i verwendet.

$$\left(\begin{array}{c} \left[\begin{array}{ccc} 0,25 & 1 & 1 \\ 0,75 & 1 & 1 \end{array} \right] \rightarrow \left[\begin{array}{c} S_3(0,25) \\ S_3(0,75) \end{array} \right] \end{array} \right)_v$$

In der Abbildung 3.7 sind die Ausgabekurven von F nicht gut sichtbar, da diese fast komplett mit der Kurve S_3 überlagert werden. Bei genauerer Betrachtung fällt auf,

das einzelne Ausgaben von F nicht von S_3 überlagert werden. Insgesamt zeigt sich wieder, dass das TL nicht eindeutig ist und dass sich gerade im Zusammenhang mit TL das Training von KNN zu einer Herausforderung werden kann. Diese entstehen durch die zufällige Initialisierung der Gewichte eines KNN vor dem Training. Der Backpropagation-Algorithmus passt die Gewichte zwar an, um ein lokales Optimum zu finden, jedoch wird das globale Optimum nicht erreicht. Dadurch ergibt sich keine eindeutige Lösung, was zu unterschiedlichen Trainingsergebnissen je nach Startbedingungen führen kann.

3.3.4 Zusammenfassung

Die von P&F verallgemeinerte Methode ermöglicht die Verwendung und Kombination verschiedener Quellen von Domänenwissen. Die Art des TL und damit wie sich der TL darstellt, beziehungsweise wie er für den Menschen aussieht, ist stark abhängig von den Hyperparametern der Methode. Erweiterbar ist die Methode, indem TL nicht vollständig unüberwacht durchgeführt wird. Vorab könnten Gemeinsamkeiten identifiziert werden und für das Training genutzt werden. Außerdem könnte der Einfluss des Parameters i als Residuum interpretiert werden und analog zur Regression ein Bestimmtheitsmaß r^2 optimiert werden.

Daneben tauchen mit dieser Verallgemeinerung ebenfalls noch zwei neue Herausforderungen auf, die eine Überarbeitung der Methode erfordern:

1. Durch die Verwendung einer binären Darstellung können ungewollte oder gewollte Zusammenhänge entstehen. F kann Gemeinsamkeiten von S_i der binären Darstellung von i zuordnen. Bei einer großen Menge an Systemen ist das OHE aber nicht mehr zu empfehlen, durch die große Menge an zusätzlichen Eingangsparametern für F .
2. Durch die Verwendung von Daten für bekannte Systeme können unbekannte Systeme nicht vorhergesagt werden. Die vorgestellte Methode kann TL nur im Training, was eine Vielzahl negativer Folgen mit sich bringt. Diese Folgen sind ein mögliches Vergessen und generell Veränderung von F , sowie ein erhöhter Zeitbedarf und die nötige Verfügbarkeit von sämtlichen Daten für das Training. Außerdem muss F für jedes neue System S trainiert werden, auch wenn es nicht genug Beobachtungen für das System gibt, sodass F neue Gemeinsamkeiten durch das Training definieren könnte.

Diese beiden Herausforderungen lassen sich lösen, indem Systeme nicht mit einem

rein durch einen willkürlichen Index definierten Vektor A unterschieden werden, sondern mithilfe von einem deskriptiven Vektor B , der das Systemverhalten repräsentiert. Nach dem Training mit B ist kein erneutes Training für die Prognose von einem neuen System S_{n+1} erforderlich, solange das neue System innerhalb des mit B beschreibbaren Systemverhalten bleibt. Im Folgenden wird die im Rahmen dieser Dissertation entwickelte Methode vorgestellt.

3.4 Stochastische Neuronale Systemquantifizierung (SNSQ)

Die SNSQ-Methode bietet gezielte Prognosen in komplexen und dynamischen Umgebungen, in denen herkömmliche Ansätze wie die bisher vorgestellten P&F-Methoden durch ihre statische Definition an ihre Grenzen stoßen. Insbesondere die Kombination aus SDNN und Systemquantifizierung bietet variable und spontane Prognosen für unbekannte Systeme. In diesem Kapitel wird zuerst auf die Systemquantifizierung eingegangen und anschließend auf die Datengenerierung.

Die Quantifizierung von Systemverhalten ist eine Methode für KNN, die die latente Struktur der Daten automatisch entdeckt und diese in einem numerischen Vektor oder einer Darstellung codiert. Diese Daten können Beobachtungen von Systemen n , beispielsweise Menschen sein, die anhand von Parametern charakterisiert werden. Durch diese Benutzersegmentierung können individuelle Werbung im E-Commerce [51] und Vorschläge zum Inhalt in Social-Media [52] erstellt werden. Allerdings stehen im Unterschied zu den typischen Fragestellungen in den Ingenieurwissenschaften sehr viele Trainingsdaten bereit, um die benötigten KNN zu trainieren. Ziel ist es dabei, nicht nur ein einzelnes System zu modellieren, sondern ein Modell zu entwickeln, das die Gesamtheit aller möglicher Systeme erfassen kann [53].

3.4.1 Annahmen und Voraussetzungen zur Anwendung

Für die Anwendung der vorgestellten Methode müssen bestimmte Voraussetzungen erfüllt und Annahmen getroffen werden. Auf diese wird im Folgenden eingegangen, um die Umstände der Methode näherzubringen und Entscheidungen nachvollziehbar zu machen.

1. Alle Systeme S sind genau einer der vier folgenden Gruppen an Systemen

zugeordnet.

$$\mathcal{V} = \{S_i \mid S_i \text{ ist ein vergleichbares oder ähnliches System}\}$$

$$\mathcal{G} = \{S_i \mid S_i \text{ ist ein stochastisch generiertes System}\}$$

$$\mathcal{R} = \{S_i \mid S_i \text{ ist ein relevantes System für das Training und Prognose}\}$$

$$\mathcal{U} = \{S_i \mid S_i \text{ ist ein unbekanntes System für die Prognose}\}$$

Durch diese Aufteilung wird die Verwendung jedes Systems klar festgelegt.

2. Alle verwendeten Informationen sind Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_i$, die dem System S_i zugeordnet werden können. Für Systeme aus den Gruppen \mathcal{R} und \mathcal{U} wird ebenfalls die Reihenfolge der Beobachtungen zum Training verwendet.¹ Über das System werden keine weiteren Erkenntnisse verwendet, außer den vollständigen Beobachtungen, bestehend aus $(\mathbf{x} \rightarrow \mathbf{z})$. Diese Bedingung hat ihre Ursache in der Verwendung von FFNN im ersten Schritt, die mit P&F wie in den Kapiteln 3.2 und 3.3 beschrieben, erstellt werden. Durch Anpassungen der Methode an dieser Stelle lassen sich auch andere Informationen verwenden.
3. In der Gruppe \mathcal{R} ist mindestens ein System vorhanden, da hiermit die FFNN erstellt werden, mit denen die stochastische Datengenerierung durchgeführt wird. Die anderen Gruppen dürfen für die Methode leer sein. Systeme der Gruppe \mathcal{U} werden nicht für das Training verwendet, sondern nur für die Prognose. Ein relevantes System aus \mathcal{R} kann ebenfalls zur Prognose verwendet werden und bei der Erstellung des FFNN F kann auch auf das Vortraining und damit auf Systeme der Gruppe \mathcal{V} verzichtet werden. Systeme aus Gruppe \mathcal{G} werden erst im Rahmen der Methode erstellt.
4. Die Ein- und Ausgabevektoren $X \in \mathbb{R}^m; Z \in \mathbb{R}^p$ haben für alle Beobachtungen der Systeme aus \mathcal{R} und \mathcal{U} dieselben Größen m und p . X und Z , die Systemen in \mathcal{V} zugeordnet sind, dürfen kleinere Vektoren haben $m > m_{\mathcal{V}}; p > p_{\mathcal{V}}$. Zusammengefasst ergeben alle Eingabevektoren X die Eingabematrix $\mathbf{x} \in \mathbb{R}^{m \times q}$, sowie alle Ausgabevektoren Z die Ausgabematrix $\mathbf{z} \in \mathbb{R}^{p \times q}$ bestehend aus allen n Beobachtungen. Die unterschiedlichen Dimensionen der Systeme aus \mathcal{V} stellen in der Pretrain-Phase der P&F-Methode keinen Hinderungsgrund dar, da die fehlenden Dimensionen mit Zufallszahlen gefüllt werden.
5. In den vorhandenen Daten eines Systems S in \mathcal{V} sollte die Eingabematrix \mathbf{x} annähernd orthogonal sein, also die Eingabedaten sollten einigermaßen

¹Für den Fall, dass keine chronologische Reihenfolge vorhanden ist, kann eine Reihenfolge geschätzt oder zufällig angenommen werden.

gleichmäßig auf den Definitionsbereich von S verteilt sein. Dies ist wichtig, um mit einfachen Modellen Werte zu interpolieren, wie in Kapitel 3.2 beschrieben ist. Die annähernde Orthogonalität von \mathbf{x} sollte auch gegeben sein, wenn die Daten des Systems direkt zum Training verwendet werden und nicht interpoliert werden. Ansonsten hat das FFNN F im Trainingsalgorithmus keine gleichmäßige Verbesserung des Fehlers über die kompletten Eingabewerte als Optimierungsziel.

6. Der Unterschied zwischen den Systemen in \mathcal{R} und \mathcal{U} besteht darin, dass sich die Beobachtungen für ein System in \mathcal{R} zum Training mit der P&F-Methode eignen. Dies kann über die Verteilung der X im Eingaberaum oder die Anzahl der Beobachtungen entschieden werden.² Auch für Systeme aus \mathcal{R} führt eine starke nicht-Orthogonalität von \mathbf{x} im Training zu keiner gleichmäßigen Verbesserung des Fehlers über die kompletten Eingabewerte.
7. Die Systeme sind sich für KNN erkennbar untereinander ähnlich. Für diese Eigenschaft gibt es keine Definition aber zum Verständnis und als Daumenregeln kann folgendes gelten. Für zwei Systeme gilt, sie sind für ein KNN ähnlich, wenn die Methode im Kapitel 3.2 einen kleineren Loss-Fehler eines Test-Datensatzes hat, als mit einem einfachen Training, wie in Gleichung (2.2) beschrieben. Diese Behelfsregel kann nicht für Systeme aus \mathcal{U} angewendet werden, da sich die vorhandenen Beobachtungen nicht für ein Training mit KNN eignen.
8. Die Systemantworten dürfen einen zeitabhängigen Anteil, sowie einen zufälligen Anteil haben. Sich mit der Zeit ändernde Systemantworten sind typisch für reale Systeme und damit auch eine reale Herausforderung in den Ingenieurwissenschaften. Die zeitliche Änderung der Systemantwort sollte durch einen sich ändernden Quantifizierungsvektor B_i sichtbar werden. Um dem KNN zu helfen mit zeitabhängigen Anteilen, ist zu überlegen, ob ein Parameter hinzugefügt wird, der die Zeit widerspiegelt oder ob ein System aufgeteilt werden kann.

²Die Anzahl der Beobachtungen für ein System in \mathcal{R} ist typischerweise deutlich größer, als für Systeme in \mathcal{U} .

3.4.2 Definition der Quantifizierung

Da unter der Prämisse gearbeitet wird, dass wenig Informationen vorhanden sind und dass nach der Informationstheorie[54] keine Informationen in der Methode erzeugt werden können, muss eine weitere Methode entwickelt werden, um eine Quantifizierung durchführen zu können. Als Ausgangspunkt für die folgende Quantifizierung dienen die im Kapitel 3.3 trainierten FFNN F . Wie dort beschrieben, wird F in zwei Schritten mit Beobachtungen aus \mathcal{V} und \mathcal{R} trainiert. Das erfolgreich³ trainierte F bildet mit jeweils einem trainierten Wert A_i einen multidimensionalen Hyperkubus mit m Dimensionen, der alle möglichen Ausgaben $Z \in \mathcal{Z}$ von allen möglichen Eingaben X beinhaltet. Dabei steht X für eine Koordinate in \mathcal{Z} , während Z für den Wert von \mathcal{Z} an der Stelle X steht. Die Funktion F_i , die schon den Parameter A_i übergeben bekommen hat, hat folgende Eigenschaften bezüglich des Definitionsbereiches D_{F_i} und des Wertebereiches W_{F_i} :

$$\begin{aligned} F_i(X) &= Z, \\ D_{F_i} &= \mathbb{R}^m \mid 0 \leq B_j \leq 1 \text{ für alle } j = 1, 2, \dots, m\} \\ W_{F_i} &= \mathcal{Z} \end{aligned}$$

Dieser Hyperkubus \mathcal{Z} stellt den kompletten Wertebereich des Regressionmodells F mit einem trainierten A_i dar und nähert damit das Systemverhalten zum dazugehörigen S_i an. Um eine Quantifizierung des Systemverhaltens von S_i zu finden, wird ein weiterer Hyperkubus \mathcal{B} mit der Kantenlänge 1 definiert. Jeder Hyperkubus \mathcal{Z} bildet einen Punkt mit den Koordinaten $B \in \mathbb{R}^{n_B} \mid 0 \leq B_j \leq 1$ für alle $j = 1, 2, \dots, m\}$ im Raum \mathcal{B} . Somit gilt $\mathcal{Z} \in \mathcal{B}$. Die Anzahl der Dimensionen von \mathcal{B} entspricht somit der Anzahl der Werte in B . Es gibt eine unbekannte Funktion D , die mit den dazugehörigen Definitionsbereich D_D und Wertebereich W_D wie folgt beschrieben werden kann:

³Zur Vereinfachung wird hier $\mathcal{L}(F(\mathbf{x}, A) - \mathbf{z}) \ll 1$ für normierte X und Z angenommen. Allerdings hängt der genaue Wert der Loss-Funktion zu einem erfolgreichen Training auch von der Versuchungenauigkeit ab. Die gegebene Definition ist für den Begriff „erfolgreich trainiert“ im Bezug auf ein KNN nicht eindeutig und liegt im Auge des Anwenders.

$$\begin{aligned}
D(B) &= \mathcal{Z}, \\
D_D &= \mathbb{R}^{n_B} \mid 0 \leq B_j \leq 1 \text{ für alle } j = 1, 2, \dots, m\} \\
W_D &= \mathcal{B}
\end{aligned}$$

Es ist naheliegend, den Raum \mathcal{B} über die unbekannte Funktion D zu definieren. Allerdings ist für die Durchführung der Systemquantifizierung wichtig, die Umkehrfunktion $E = D^{-1}$ zur Definition von \mathcal{B} zu verwenden. Dies liegt vor allem daran, dass \mathcal{Z}_i bekannt ist und die genaue Definition der Vektoren B eine untergeordnete Rolle spielt.

$$\begin{aligned}
E(\mathcal{Z}) &= B, \\
D_B &= \mathcal{B} \\
W_D &= \mathbb{R}^{n_B} \mid 0 \leq B_k \leq 1 \text{ für alle } k = 1, 2, \dots, m\}
\end{aligned}$$

Um die Systemquantifizierung durchführen zu können, wird eine Definition des Raumes \mathcal{B} gesucht. Diese Definition geschieht mit der Funktion E , die folgende Kriterien erfüllen soll:

1. \mathcal{B} soll möglichst wenige Dimensionen enthalten. Dies hat den Grund, dass so die einzelnen Werte von B eine größere Aussagekraft haben. Für den trivialen Extremfall, dass B gleich der Gewichte ω von F_i ist, würden alle möglichen \mathcal{Z} , die ein KNN mit der Architektur von F erstellen könnten, ein Teil von \mathcal{B} . In Folge davon werden wieder alle Informationen benötigt und mit der Systemquantifizierung kann kein TL stattfinden.
2. Zu jeder Koordinate B soll der dazugehörige Hyperkubus \mathcal{Z} mit einem möglichst kleinen Fehler bestimmt werden können. Andersherum soll aus jedem \mathcal{Z} , das in \mathcal{B} enthalten ist, eine Koordinate B definiert werden. Diese Forderung hat praktische Gründe, da ohne die Bestimmung der Funktion D keine Prognose des Systemverhaltens durchgeführt werden kann.
3. \mathcal{B} soll so diverse \mathcal{Z} enthalten, dass diese alle \mathcal{Z}_i zu allen Systemen S_i und möglichst auch \mathcal{Z}_{i+1} für unbekannte aber vergleichbare Systeme S_{n+1} annähern.
4. Nicht ähnliche Systeme zu den verwendeten Systemen S_i sollen einen möglichst kleinen Anteil an \mathcal{B} haben.

Um alle Kriterien zu erfüllen und den Raum \mathcal{B} , sowie die dazugehörige Funktion E zu definieren, wird ein AE ED verwendet. Die angesprochenen Kriterien werden dabei wie folgt erfüllt.

1. Das erste Kriterium lässt sich beim Training des AE ED über einen zusätzlichen Term in der Hyperparameteroptimierung umsetzen. Dabei wird die Größe des CFV als kleiner positiver Wert dem Optimierungsziel, also der Loss-Funktion hinzugefügt. Dies sorgt für eine möglichst kleine Anzahl an Dimensionen von \mathcal{B} .
2. Mit dem Training des AE ED sind beide angesprochenen Funktionen D und E vorhanden, da der AE geteilt werden kann. Die genaue Erklärung dieser Teilung folgt in diesem Kapitel.
3. Nicht nur die Definition von \mathcal{B} läuft über E ab, sondern auch das Training von ED enthält alle verwendeten \mathcal{Z} . Daher sind alle verwendeten \mathcal{Z} auch im Hyperkubus \mathcal{B} enthalten.
4. Durch den schon angesprochenen zusätzlichen Term in der Hyperparameteroptimierung werden die Bereiche von \mathcal{B} , die nicht-ähnliche \mathcal{Z} enthalten ebenfalls minimiert. Da die Größe von \mathcal{B} stark von der Dimension von \mathcal{B} abhängt, hat die Hyperparameteroptimierung das implizite Ziel, die Größe von \mathcal{B} so klein wie möglich zu definieren.

Der AE ED muss also trainiert werden, den Hyperkubus \mathcal{Z} zu encodieren und wieder zu decodieren. Das Training des AE ist wie folgt:

$$\min_{\omega} (\mathcal{L}(ED(\mathcal{Z}) - \mathcal{Z})) \quad (3.7)$$

Nach dem Training lässt sich ED in zwei KNN aufteilen, den Encoder E und den Decoder D . Dadurch verändern sich die Funktionen E und D leicht, da sie nur eine Schätzung ausgeben können.

$$\begin{aligned} ED(\mathcal{Z}_i) &= D(E(\mathcal{Z}_i)) \\ E(\mathcal{Z}_i) &= \hat{B}_i \\ D(\hat{B}_i) &= \hat{\mathcal{Z}}_i \end{aligned}$$

Im nächsten Schritt wird $\mathcal{Z} \rightarrow B$ über die Funktion D definiert. Es wird jedoch nicht der gesamte Raum \mathcal{B} betrachtet, sondern nur die Bereiche, in denen \mathcal{Z} vorhanden ist. Aus praktischen Gründen wird also nur der Teil von \mathcal{B} verwendet, der auch \mathcal{Z} enthält. Folgendes wird definiert:

$$E(\mathcal{Z}_i) = B_i \quad (3.8)$$

Damit kann B für folgendem Zusammenhang ausgerechnet werden:

$$\mathcal{Z}_i \rightarrow B_i$$

3.4.3 Praktische Umsetzung der Quantifizierung

Zur Quantifizierung von Systemen fehlt noch die Schätzung der Koordinaten B_{n+1} aus beliebigen Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{n+1}$, sodass einem unbekanntem System ein Hyperkubus \mathcal{Z} zugeordnet werden kann. Diese Schätzung von B_{n+1} wird mit einem RNN R mit vorhandenen Beobachtungen $(\mathbf{x} \rightarrow \mathbf{z})_{n+1}$ des unbekanntem Systems S_{n+1} ausgeführt.

$$R((\mathbf{x} \rightarrow \mathbf{z})_{n+1}) = \hat{B}_{n+1}$$

Das Training von R sieht wie folgt aus:

$$\min_{\omega} (\mathcal{L}(R((\mathbf{x} \rightarrow \mathbf{z})_i) - B_i)) \quad (3.9)$$

Schließlich wird F analog zu Gleichung (3.6) mit dem quantifizierten Verhaltensvektor B trainiert, um Schätzungen von $\hat{\mathbf{z}}$ zu berechnen.

$$F(\mathbf{x}, B) = \hat{\mathbf{z}}$$

Hier zeigen sich die Gemeinsamkeiten zu anderen Regressionsmethoden. Zum Beispiel wird bei einer linearen Regression die resultierende Funktion im Vorfeld alleine durch die Auswahl der Methode auf einen speziellen Bereich an resultierenden Regressionsfunktionen beschränkt. Bei der linearen Regression wird die resultierende Regressionsfunktion auf den Bereich der linearen Funktionen beschränkt. Analog

dazu wird das Modell F auf die Funktionen \mathcal{Z} in \mathcal{B} beschränkt, die sich durch die vage Eigenschaft der Ähnlichkeit zu den Systemantworten aus \mathcal{V} und \mathcal{R} auszeichnen.

Noch ein letzter Schritt muss durchgeführt werden, um nicht nur eine Prognose $\hat{\mathbf{z}}$ für \mathbf{x} zu bekommen. Um ein X zu bestimmen, für das eine bestimmte Systemantwort Z_{soll} auftritt, wird folgendes Optimierungsproblem gelöst:

$$\min_X (\mathcal{L}(F(X, B) - Z_{\text{soll}})) \quad (3.10)$$

3.4.4 Synthetische Datengenerierung zur Definition von \mathcal{B}

Die zentrale Herausforderung bei der Quantifizierung besteht in der systematischen Definition des Raumes \mathcal{B} . Um den AE ED , wie in Gleichung (3.7) zu trainieren, sind mit den Systemen aus \mathcal{V} und \mathcal{R} nicht genügend Systeme für ein erfolgreiches Training vorhanden. Die zentrale Herausforderung besteht zusätzlich in der begrenzten Datenmenge, die für sich genommen ein Training von ED verhindert. Um dieses Problem zu umgehen, werden in dieser Arbeit synthetische Daten erstellt, die vergleichbar zu den Systemen in \mathcal{R} sind. Im Gegensatz zu anderen Methoden zur synthetischen Datengenerierung werden dabei keine Anforderungen an das Endergebnis gestellt. Da in der Anwendung ein optimales \mathcal{Z} ausgesucht wird, stören synthetische Daten von schlechter Qualität nicht, solange es auch synthetische Daten von guter Qualität gibt.

Die Datengenerierung erfolgt dabei durch ein geschicktes Vorgehen. Ein FFNN F wird trainiert, wie es im Abschnitt Kapitel 3.3 beschrieben ist. Dafür werden Daten von allen Systemen in \mathcal{V} und \mathcal{R} verwendet. Nach dem erfolgreichen Training wird F noch mit einem oder mehreren zufälligen Datenpunkten $A_{\text{RAN}}, (\mathbf{x} \rightarrow \mathbf{z})_{\text{RAN}}$ weiter trainiert.

$$\min_{\omega} (\mathcal{L}(F(\mathbf{x}_{\text{RAN}}, A_{\text{RAN}}) - \mathbf{z}_{\text{RAN}})) \quad (3.11)$$

Während A_{RAN} komplett zufällig bestimmt werden kann, besteht für $(\mathbf{x} \rightarrow \mathbf{z})_{\text{RAN}}$ ein Grund, nicht komplett zufällige Werte und damit beliebig große oder kleine Werte anzunehmen. Durch eine zu große Entfernung der zufälligen Punkte zur echten Ausgabe \mathcal{Z} wird \mathcal{Z} möglicherweise zu sehr verändert und verliert damit jede Ähnlichkeit zu anderen \mathcal{Z} . Für diesen Zweck wird die Standardabweichung σ_{RAN} definiert, die den Abstand von Z_{RAN} zu \mathcal{Z} an der Stelle X_{RAN} beeinflusst. Die Funktion $\text{Unif}(0, 1)$

bezeichnet eine Gleichverteilung im Intervall $[0, 1]$. Mit geschweiften Klammern $\{\dots\}$ bezeichnet die Funktion $\text{Unif}\{0, 1\}$, das eine Auswahl der in den Klammern enthaltenen Elementen mit der jeweils gleichen Wahrscheinlichkeit getroffen wird. $\text{Norm}(\mu, \sigma^2)$ bezeichnet die Normalverteilung mit dem Mittelwert μ und der Standardabweichung σ^2 . Die beiden Funktionen $\text{Unif}(0, 1, k)$ und $\text{Norm}(\mu, \sigma^2, k)$ haben die selbe Funktion Zufallszahlen zu beschreiben, nur beschreiben sie einen Vektor der Länge k :

$$\text{label} : \text{unif} \text{unif}(0, 1, k) = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{pmatrix}, \quad u_i \sim \text{Unif}(0, 1) \quad \text{für alle } i = 1, 2, \dots, k$$

Analog dazu die Definition der Funktion $\text{Norm}(\mu, \sigma^2, n)$:

$$\text{Norm}(\mu, \sigma^2, k) = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix}, \quad v_i \sim \text{Norm}(\mu, \sigma^2) \quad \text{für alle } i = 1, 2, \dots, k$$

Schließlich noch die mathematische Beschreibung der oben genannten Vektoren \mathbf{A}_{RAN} , \mathbf{X}_{RAN} und \mathbf{Z}_{RAN} mit n als Anzahl aller Systeme.

$$\begin{aligned} \mathbf{A}_{\text{RAN}} &= A_i \quad \text{mit } S_i \sim \text{Unif}\{\mathbb{R}\} \\ \mathbf{X}_{\text{RAN}} &\sim \text{Unif}(0, 1, m) \\ \mathbf{Z}_{\text{RAN}} &\sim \text{Norm}(F(\mathbf{X}_{\text{RAN}}, \mathbf{A}_{\text{RAN}}), \sigma_{\text{RAN}}^2, p) \end{aligned} \tag{3.12}$$

Allerdings zeigt sich, dass verschiedene KNN mit identischer Architektur, die auf dieselben Daten trainiert wurden, sich dennoch stark in den Gewichten unterscheiden. Diese Unterschiede nehmen mit größerer Komplexität zu und entstehen, da die KNN mit zufälligen, also unterschiedlichen Gewichten starten. Diese Unterschiede sind praktisch unsichtbar nach einem erfolgreichen Training, aber kommen deutlich zum Vorschein, wenn das KNN mit zufälligen Werten im Folgenden trainiert wird.

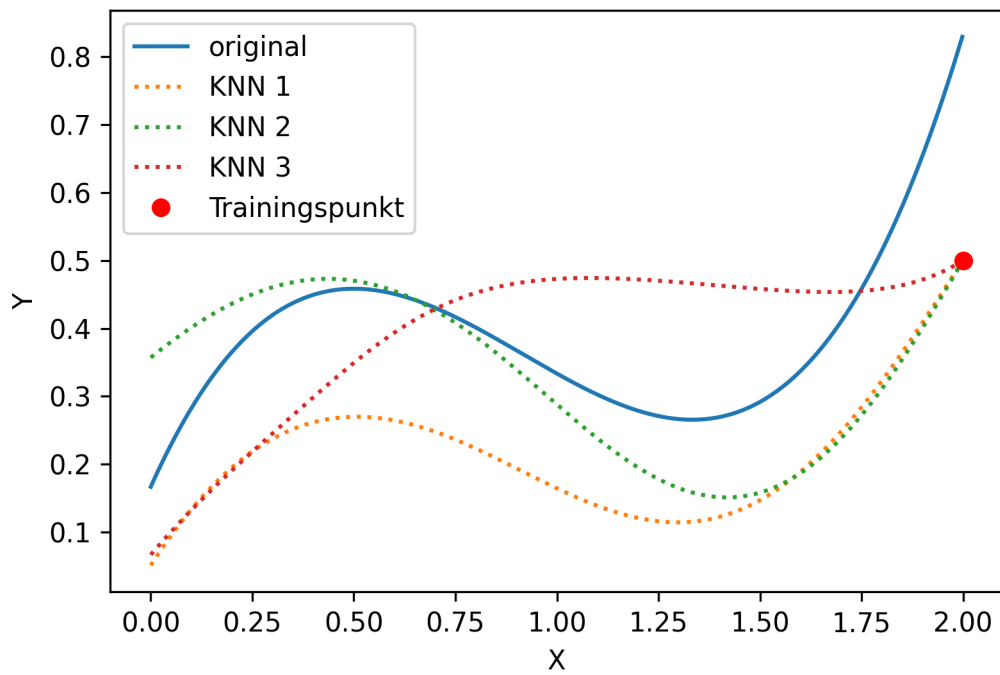


Abbildung 3.8: Vergleich von auf die selbe Kurve trainierten KNN, die mit einem Datenpunkt nachtrainiert sind [55].

In der Abbildung 3.8 ist so ein Training mit einem zufälligen Punkt, aber identischer Architektur dargestellt. Die blaue Kurve entspricht der Originalkurve, auf die alle KNN im ersten Schritt trainiert werden. Im zweiten Schritt werden alle KNN auf den einzelnen roten Punkt trainiert. Dank des BP Algorithmus werden die Kurven auf direktem Weg in Richtung des roten Punktes verschoben. Durch die unterschiedliche Berechnungen im Inneren der KNN entstehen dabei unterschiedliche Ausgangskurven. Bei KNN 3 scheinen sich die Extrema zu verschieben, um die Ausgabe anzupassen, damit sie den roten Punkt enthält. Bei KNN 2 scheint sich eher nur der linke Teil der Ausgabe abzusenken und bei KNN 3 ist am ehesten die ganze Ausgabe im Training mit dem zufälligen Punkt abgesenkt worden. Durch dieses Training mit zufälligen Datenpunkten bleibt die Gestalt der Ausgabekurve weitestgehend erhalten.

Im Gegensatz zur probabilistischen synthetischen Datengenerierung durch Generative Adversarial Networks (GAN) oder Large Language Models (LLM), die auf globalen Wahrscheinlichkeitsverteilungen basieren, und der Datenaugmentation, die existierende Daten durch Transformationen erweitert, verwendet diese Methode das zufällige Ergebnis eines trainierten KNN als Zufall, der der Datengenerierung dient. Hierbei werden gezielt durch eine Veränderung der Gewichte synthetische Daten erzeugt.

Dieser Ansatz nutzt das stochastische Verhalten der trainierten KNN, indem die Gewichte minimal auf ein stochastische Ziel hin verändert werden. Auf diese Weise entstehen gezielt synthetische, aber vergleichbare Ausgabekuben. Diese Technik wird in dieser Arbeit synthetische Datengenerierung genannt. Damit unterscheidet sich die Methode auch im Namen von zwei anderen populären Methoden der Datengenerierung im Bereich des ML, nämlich der schon erwähnten generativen Modellierung eines Large Language Model (LLM) oder eines Generative Adversarial Network (GAN) [56] und der Augmentierung von Daten, die starren mathematischen Transformationen folgt.

Eine Interpretation des nachfolgenden Trainings mit einem einzelnen Punkt, lässt dieses als Frage nach der möglichen Veränderung der Originalkurve interpretieren, unter der Voraussetzung, dass sich die Originalkurve verändert hat und jetzt den zufälligen Punkt $(\mathbf{x}|\mathbf{z})^{\text{RAN}}$ enthält.

Für die Methode ist es vorteilhaft, dass sich die Antwort der unterschiedlichen KNN unterscheiden, da so die Diversität der \mathcal{Z} in \mathcal{B} vergrößert wird. Dies entspricht der Anforderung 3 an die Definition von \mathcal{B} . Diese Interpretation als Frage der Veränderung der Originalkurve passt gut zu diesem Verhalten, dass sich die Antworten der

verschiedenen KNN unterscheiden, da es auf diese Frage der Veränderung keine eindeutige Antwort geben kann.

Die Voraussetzung der Anwendung dieser stochastischen Datengenerierungsmethode im Bereich der Quantifizierung ist erfüllt, da die entstehenden Hyperkuben \mathcal{Z} vergleichbar mit den relevanten Systemen in \mathcal{R} sind. Schließlich leiten sie sich direkt von diesen ab.

Das Ziel ist es nicht, die mit dieser stochastischen Generierung von Daten verbunden ist, dass alle entstehenden Hyperkuben \mathcal{Z} relevant sein könnten, also dass es ein reales System geben kann, das genau dieses Systemverhalten widerspiegelt. Es ist aber das Ziel, dass möglichst so viele verschiedene Hyperkuben vorhanden sind, dass sich möglichst die realen Systeme aus \mathcal{U} darin befinden, ohne dass sich alle möglichen Kurven in diesen Hyperkuben befinden. Sollten sich alle möglichen Ausgabekurven innerhalb dieser Hyperkuben befinden, könnte die ganze Methode nicht besser werden, als andere Regressionsmethoden, die ebenfalls alle möglichen Ausgabekurven sehr gut annähern können. Nur durch die Beschränkung der möglichen Ausgabekurven kann diese Methode mit weniger Informationen genauer werden, als andere Regressionsmethoden ohne Beschränkungen.

3.4.5 Projektion des multidimensionalen Hyperkubus \mathcal{Z} auf einen endlichen Vektor

Eine Herausforderung in der Anwendung zeigt sich unter anderem bei der Umsetzung der Gleichung (3.9). Die in dieser Dissertation verwendeten KNN akzeptieren nur Vektoren als Eingabe und keine multidimensionalen Hyperkuben. Daher werden die Hyperkuben \mathcal{Z} in Vektoren Z umgewandelt. Um \mathcal{Z}_i in einen Vektor umzuwandeln, werden in dieser Arbeit zwei Sampling-Methoden verwendet. Zum einen wird ein LHS durchgeführt, sodass für jeden Hyperkubus \mathcal{Z}_i Beobachtungen in der Form $(\mathbf{x}^{\text{LHS}} \rightarrow \mathbf{z}^{\text{LHS}})_i$ erstellt werden. Da $\mathbf{x}_i^{\text{LHS}}$ unabhängig von \mathcal{Z}_i identisch für alle \mathcal{Z} ist, reicht die Verwendung von $\mathbf{z}_i^{\text{LHS}}$ für das Training aus. Anschließend wird $\mathbf{z}_i^{\text{LHS}}$ mithilfe der Vektorisierungsoperation `vec` in einen Vektor umgewandelt.

$$\text{vec}(\mathbf{z}_i^{\text{LHS}}) = \begin{bmatrix} \mathbf{z}_{1,1} \\ \mathbf{z}_{2,1} \\ \vdots \\ \mathbf{z}_{m,1} \\ \mathbf{z}_{1,2} \\ \vdots \\ \mathbf{z}_{m,n} \end{bmatrix} = \mathbf{z}_i^{\text{LHS}} \quad (3.13)$$

Das angesprochene Training des RNN aus Gleichung (3.9) wird dann zu folgendem Ausdruck, wobei alle $\mathbf{z}_i^{\text{LHS}}$ zu \mathbf{z}^{LHS} zusammengefasst werden.

$$\min_{\omega} (\mathcal{L} (ED(\mathbf{z}^{\text{LHS}}) - \mathbf{z}^{\text{LHS}}))$$

Die zweite Möglichkeit der Umwandlung von \mathcal{Z}_i in einen Vektor besteht in dem Sampling mit Zufallszahlen. Der Vorteil gegenüber LHS ist, dass die Anzahl der Samples beliebig erweiterbar ist. Es können immer wieder neue Zufallszahlen erstellt werden. In den erstellten Samples $(\mathbf{x}^{\text{RAN}} \rightarrow \mathbf{z}^{\text{RAN}})_i$ ist der Term $\mathbf{x}_i^{\text{RAN}}$ zufällig erstellt und damit abhängig von \mathcal{Z}_i . Daher müssen die vollständigen Samples verwendet werden, um eine Projektion von \mathcal{Z} auf einen Vektor zu ermöglichen. In der anschließenden Anwendung der Sample ergibt sich ein weiterer Unterschied. Entweder reicht es, \mathbf{z}^{LHS} zu verwenden, da \mathbf{x}^{LHS} konstant ist, oder aber es müssen die kompletten Samples $(\mathbf{x}^{\text{RAN}} \rightarrow \mathbf{z}^{\text{RAN}})_i$ verwendet werden. Um beide Matrizen zu verwenden, werden diese zunächst mit der Verkettungsoperation concat verbunden:

$$\text{concat}(\mathbf{x}_i^{\text{RAN}}, \mathbf{z}_i^{\text{RAN}}) = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \\ \hline z_{1,1} & z_{1,2} & \dots & z_{1,n} \\ z_{2,1} & z_{2,2} & \dots & z_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{p,1} & z_{p,2} & \dots & z_{p,n} \end{bmatrix}$$

Anschließend wird diese Sample-Matrix $\text{concat}(\mathbf{x}_i^{\text{RAN}}, \mathbf{z}_i^{\text{RAN}})$ noch mit der Verkettungsoperation vec in einen Vektor überführt:

$$\text{vec}(\text{concat}(\mathbf{x}_i^{\text{RAN}}, \mathbf{z}_i^{\text{RAN}})) = \begin{bmatrix} x_{1,1} \\ x_{2,1} \\ \vdots \\ x_{m,1} \\ z_{1,1} \\ z_{2,1} \\ \vdots \\ z_{p,1} \\ x_{1,2} \\ x_{2,2} \\ \vdots \\ x_{m,2} \\ z_{1,2} \\ z_{2,2} \\ \vdots \\ z_{p,2} \\ \vdots \\ x_{1,n} \\ x_{2,n} \\ \vdots \\ x_{m,n} \\ z_{1,n} \\ z_{2,n} \\ \vdots \\ z_{p,n} \end{bmatrix} = \mathbf{Z}_i^{\text{RAN}}$$

Anschließend werden alle $\mathbf{Z}_i^{\text{RAN}}$ zu \mathbf{z}^{RAN} zusammengefasst und das Training des RNN aus Gleichung (3.9) wird wie folgt ausgeführt:

$$\min_{\omega} (\mathcal{L}(\text{ED}(\mathbf{z}^{\text{RAN}}) - \mathbf{z}^{\text{RAN}}))$$

In der praktischen Umsetzung geschieht diese Projektion genauso mit den anderen Gleichungen in denen der Hyperkubus \mathcal{Z} ein Teil der Berechnungen ist. Für das Training von dem AE ED , ist die Projektion mit dem Sampling über Zufallszahlen

keine Option. x_i^{RAN} sind Zufallszahlen, abhängig von i . Diese Zahlen als Ausgabe von ED im Training zu verwenden, bedeutet, dass der Autoencoder in seinem Training auch genau diese Zahlen lernen muss. Und da diese Zahlen zufällig sind, kann ED diese Zahlen nur replizieren, wenn sie Teil des CFV werden oder wenn die Zahlen in Abhängigkeit des CFV auswendig gelernt werden. Dies sollte nicht Teil des Trainings von ED sein, da der CFV vor allem abhängig von \mathbf{z} sein sollte.

3.4.6 Anwendung auf die Referenzsysteme

Zur Anwendung der Methode werden die in Kapitel 3.3.3 trainierten FFNN F als Basis verwendet. Diese F werden anhand von Vorwissen, Hypothesen und Literatur zu vergleichbaren Systemen, sogenannten Domainwissen sowie Versuchsergebnissen des relevanten Systems nach der P&F-Methode trainiert. Mit diesen FFNN wird im nächsten Schritt die stochastische Datengenerierung SDNN durchgeführt. Danach wird mit diesen Daten eine Quantifizierung erstellt, also mit einem AE, einem speziellen KNN ein Verhaltensvektor definiert. Zuletzt wird ein weiteres KNN R erstellt, das aus Beobachtungen, also Versuchsergebnissen eines relevanten Systems einen quantifizierten Verhaltensvektor schätzt. Das RNN wird im letzten Trainingsschritt mit den synthetischen Daten aus dem SDNN und den passenden Verhaltensvektoren aus dem AE trainiert. Dann können für das relevante System anhand des Verhaltensvektors Prognosen erstellt werden. In Abbildung 3.1 ist dieser Ablauf nochmals dargestellt.

Im ersten Schritt wird die SDNN durchgeführt. Da die Referenzsysteme aus kubischen Polynomen mit jeweils einem Eingabe- und Ausgabeparameter sind, reicht ein zufälliger Trainingspunkt, der nah an der Originalkurve liegt, vollkommen aus. Die Vektoren aus der Gleichung (3.12) ergeben sich für das Training gemäß Gleichung (3.11) dabei wie folgt:

$$\mathbf{A} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad (3.14)$$

$$\mathbf{X}_{\text{RAN}} \sim \text{Unif}(0, 1)$$

$$\mathbf{Z}_{\text{RAN}} \sim \text{Norm}(F(\mathbf{X}_{\text{RAN}}, \mathbf{A}_{\text{RAN}}), \sigma_{\text{RAN}} = 0.2)$$

Mit diesen Vektoren wird also das FFNN F umtrainiert. F wurde zunächst auf die Ausgabe von den Systemen in \mathcal{V} trainiert worden. Damit waren alle anderen

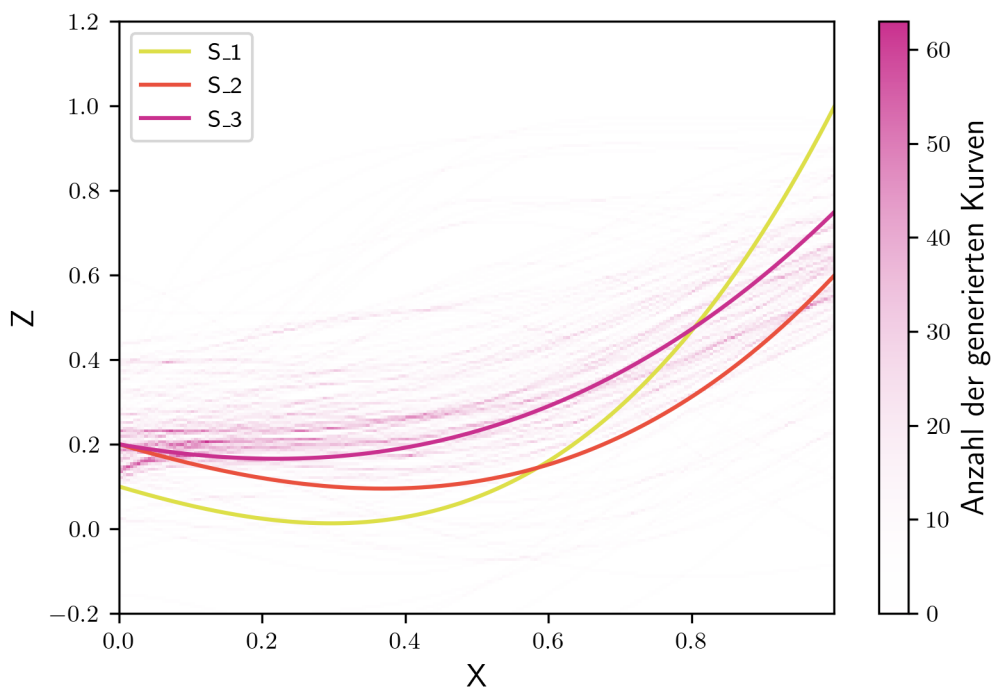


Abbildung 3.9: 2D-Histogram des SDNN an den Referenzsystemen

Hyperkuben \mathcal{Z}_i für alle $S_i \notin \mathcal{V}$ nicht trainiert, und deren Ausgabe ist mehr oder weniger zufällig. Nach dem Training von F auf alle $S_j \in \mathcal{R}$ sind alle \mathcal{Z}_i für alle $S_i \in \mathcal{V}$ nicht mehr trainiert und deren Ausgabe ist mehr oder weniger zufällig. Nach dem Training mit den Vektoren aus den Gleichungen (3.14) wird ein trainierter \mathcal{Z}_j stochastisch verändert. Zu diesem Zeitpunkt ist nicht klar, was genau mit den anderen \mathcal{Z}_i geschieht, die durch einen zufälligen Trainingspunkt mit A_j verändert werden. Die Verwendung aller anderen \mathcal{Z}_i , die nicht direkt verändert wurden, führt zu einer großen Unsicherheit. Eine Aussage über Ähnlichkeit oder Vergleichbarkeit kann nicht getroffen werden, da während des Trainings mit BP auch nicht die Ausgabe als \mathcal{Z}_i optimiert wird.

In Abbildung 3.9 ist das Ergebnis dieses zufälligen Trainings für die Referenzsysteme als 2D-Histogramm dargestellt. Verglichen mit Abbildung 3.4 zeigt sich, dass die Originalkurve S_3 im Histogramm nicht besonders hervorsteht. Die Mehrheit der Kurven im Diagramm scheint dabei eine positive Änderung der Steigung zu haben, also ganz ähnlich zu den Funktionen des Referenzsystems sind, die alle eine positive Änderung der Steigung haben. Auffällig ist auch, dass die Referenzsysteme bei Eingaben bis circa 0,7 keine größeren Ausgaben als 0,3 aufweisen. Somit ist der Wertebereich über einen großen Teil des Definitionsbereichs stark eingeschränkt und die generierten Kurven weichen nur selten von diesem Bereich ab.

| σ_{RAN} aus Gleichung (3.14) | resultierendes gemitteltes σ_{SDNN} der Verteilung |
|--|--|
| 0.01 | 0.053 |
| 0.1 | 0.104 |
| 0.2 | 0.183 |

Tabelle 3.2: Zusammenhang zwischen der Standardabweichung σ_{RAN} der zufälligen Trainingspunkte mit der über x gemittelten Standardabweichung σ_{SDNN} der resultierendem synthetischen Daten aus SDNN

In Tabelle 3.4.6 ist der Einfluss von der Standardabweichung der zufälligen Trainingspunkte σ_{RAN} auf die SDNN-Methode zur Orientierung dargestellt. Diese Trainingspunkte werden in der SDNN-Methode verwendet, um mit den trainierten FFNN neue ähnliche Systemverhalten zu generieren. Diese Trainingspunkte werden mit der Standardabweichung von σ_{RAN} zur Originalkurve ausgewählt. Um das gemittelte σ_{SDNN} der resultierenden Funktionen zu ermitteln, wird das σ_j für einige Werte von X_j über alle Systeme $S_i \in \mathcal{G}$ ermittelt und anschließend gemittelt. Für den Extremfall, dass $S_i = S_k \quad \forall i, k$ mit $S_i, S_k \in \mathcal{G}$ dann gilt $\sigma_{\text{RAN}} = 0$ unabhängig vom spezifischen Kurvenverlauf von S_j . Für die ermittelten Werte in Tabelle 3.4.6

wurden 1000 gleichverteilte X_j über den kompletten Definitionsbereich von S_i verwendet.

Um die Anwendung am Referenzsystem abzuschließen, wird noch ein weiteres Referenzsystem $S_4 \in \mathcal{U}$ eingeführt:

$$S_4(X) = 1 - \cos(x)$$

Dieses System dient ausschließlich dem Test der Methode. Es ist anhand der Formel nicht ähnlich zu den bereits eingeführten Referenzsystemen. Visuell ist es aber in einem ähnlichen Wertebereich, wie in Abbildung 3.10 dargestellt ist.

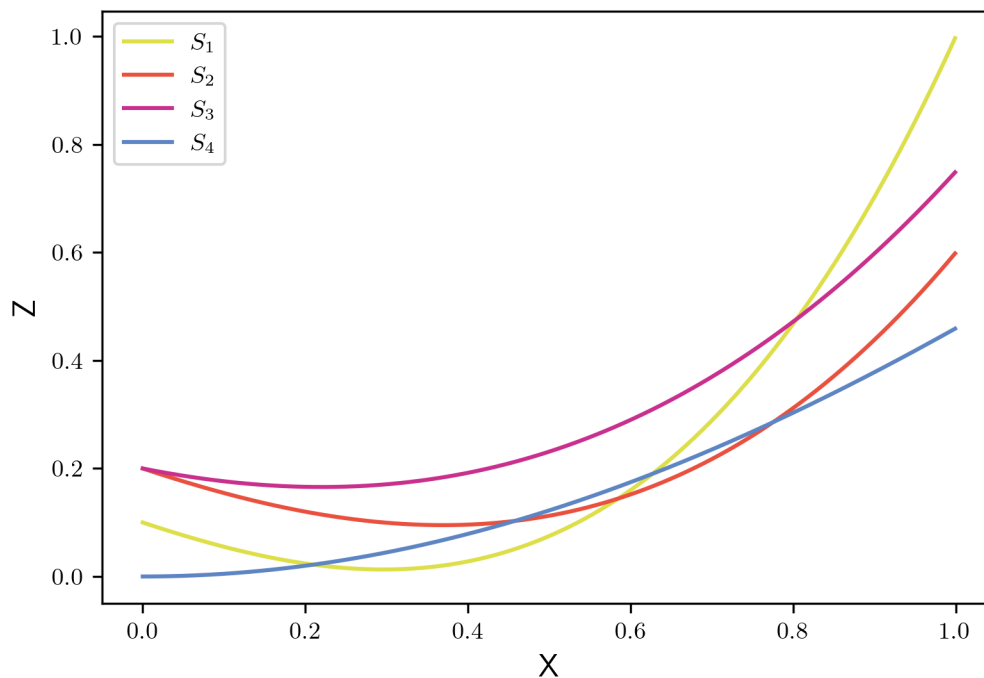


Abbildung 3.10: Darstellung der Referenzsysteme S_1 bis S_4

Im nächsten Schritt geht es um die Definition des Raumes \mathcal{B} über den Verhaltensvektor B . Es wird ein AE ED trainiert. Dafür werden die Vektoren Z_i^{LHS} nach Gleichung (3.13) gebildet und zusammengefasst zu \mathbf{z}^{LHS} . Schließlich folgt das Training von

ED nach Gleichung (3.7). Nach dem erfolgreichen Training von ED wird nun der AE ED aufgeteilt in die beiden FFNN E und D . Danach werden B_i berechnet mithilfe von E nach Gleichung (3.8). Nun kann mit B_i und $(\mathbf{x} \rightarrow \mathbf{z})_{\text{LHS}}$ das RNN R nach Gleichung (3.9) trainiert werden. Da für die Referenzsysteme die genaue Berechnung bekannt ist, können mit E auch die genauen Verhaltensvektoren B für die Referenzsysteme bestimmt werden. Anhand dieser lässt sich der Fehler bestimmen, der in Abhängigkeit der Anzahl der Stützstellen bei der Prognose von B mit R auftritt. In Abbildung 3.11 ist der Fehler als RMSE dargestellt. Diese Ergebnisse sollten als Test betrachtet werden, da die Referenzsysteme Teil von \mathcal{R} und \mathcal{U} sind und nur die Systeme in \mathcal{G} zum Training verwendet werden.

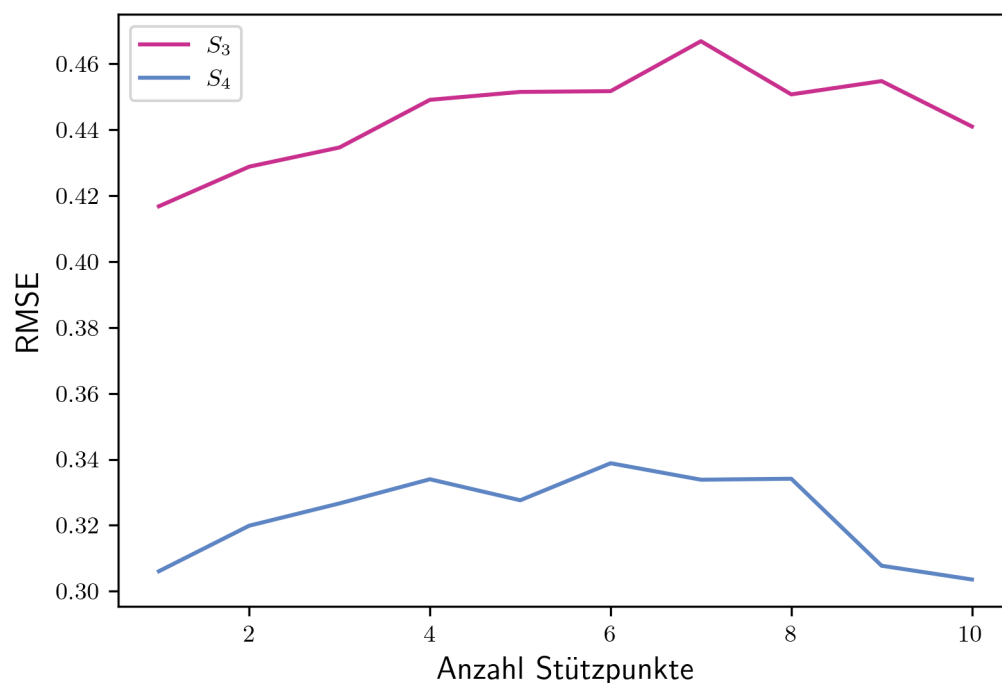


Abbildung 3.11: RMSE der Prognose von B mit dem RNN R im Vergleich zum Encoder E in Abhängigkeit von der Anzahl der Stützstellen.

In Abbildung 3.11 ist der RMSE über die Anzahl der Stützpunkte aufgestellt, die zum Schätzen von dem Verhaltensvektor B verfügbar waren. Der RMSE wird hier berechnet, zwischen der Schätzung von \hat{B} mit dem RNN R und dem B , das mithilfe

des Encoders E bestimmt wird. Interessanterweise geschieht die Schätzung B durch das RNN R mit einem insgesamt hohen Fehler. Da die Bedeutung der Werte von B von dem AE ED im Training festgelegt wurde, ist von den hohen Werten für den RMSE leider nicht wirklich etwas für die eigentliche Schätzung ableitbar.

Die Schätzung von S_4 ist unabhängig, von der Anzahl der Stützstellen, immer mit einem kleineren Fehler verbunden als die Schätzung von B für S_3 . Dies ist ein interessantes Ergebnis, da S_3 mitten durch die Verteilung der generierten Systeme aus \mathcal{G} verläuft. Ebenfalls waren bisher alle Referenzsysteme Polynome. Dass der Fehler von S_4 dabei so viel niedriger ist, könnte daran liegen, dass S_4 fast durchgängig einen Abstand von 0,2 zu S_3 hat. Von S_3 wurden die Systeme aus \mathcal{G} generiert und die resultierende Verteilung hat eine gemittelte Standardabweichung von ebenfalls ungefähr 0,2.

Eine Weitere Beobachtung in Abbildung 3.11 ist, dass der Fehler mit einer größer werdenden Zahl an Stützstellen nicht direkt kleiner wird. Der Fehler wird erstmal größer und mit einer Stützstellenzahl von 7 bzw. 8 verringert sich der Vorhersagefehler von B wieder auf oder knapp über das Niveau von einer Stützstelle. Dies ist unerwartet, da das System S_3 mit zwei Stützstellen sehr gut von einem KNN prognostiziert wurde.

Als nächstes wird FFNN F trainiert. Dafür werden die selben Daten verwendet wie für das Training von R nur mit einer anderen Zielsetzung. Das Training von F erfolgt nach Gleichung (3.10). Nun kann mit dem trainierten F der Fehler auf \mathcal{Z} berechnet werden, der durch den Prognosefehler von B ausgelöst wird. Die Abbildung 3.12 verwendet die gleichen Prognosen für B wie die Abbildung 3.11.

In Abbildung 3.12 sehen wir den RMSE der gesamten Systeme S_3 und S_4 der Prognose durch FFNN F mit den prognostizierten B aus Abbildung 3.11, ebenfalls über der Anzahl der Stützstellen. Die Vorhersagefehler für diese einfachen Funktionen der Referenzsysteme weist mit etwa 0,04 bis 0,11 vergleichsweise hohe Werte auf. Im Vergleich zum Vorhersagefehler von B auf dem die Vorhersage von Z basiert, das von etwa 0,3 bis 0,45 reicht, sind die Fehler allerdings bemerkenswert klein. Der Grund dafür, liegt an dem relativ kleinen Bereich, in dem die generierten Systeme aus \mathcal{G} zu finden sind. Dieser Bereich ist in Abbildung 3.9 dargestellt. Ein großer Vorhersagefehler in diesem Bereich wird zu einem deutlich kleineren Fehler im gesamten Wertebereich der Vorhersagefunktion.

Weiterhin fällt auf, dass die Schätzung des Verhaltensvektors B in Abbildung 3.11 von System S_3 mit einem deutlich größeren Fehler versehen ist, als die Schätzung von

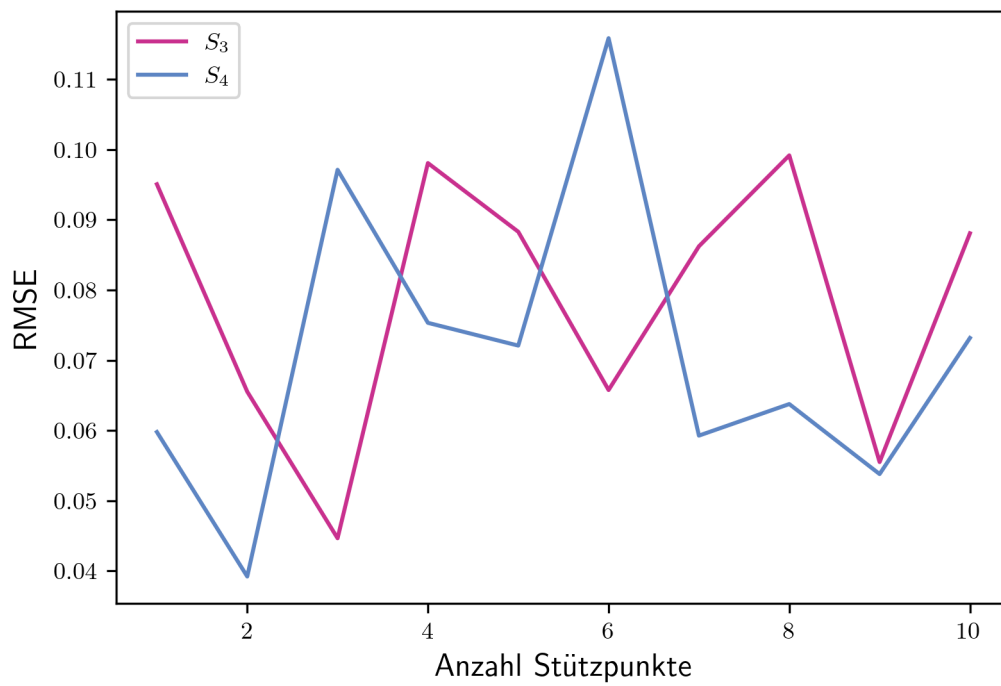


Abbildung 3.12: RMSE der Prognose von Z durch das FFNN F im Vergleich zu den wahren Z -Werten von S_3 und S_4 in Abhängigkeit von der Anzahl der Stützstellen.

System S_4 . Allerdings ist der Vorhersagefehler von Z für beide Systeme annähernd gleich. Dies könnte daran liegen, dass der Bereich um das System S_3 viel genauer von dem Verhaltensvektor B definiert wird. Schließlich befinden sich hier die meisten stochastisch generierten Systeme.

Auffällig an dem RMSE der Prognose von Z ist ein fast komplementär verlaufender Prognosefehler zwischen S_3 und S_4 bis zu 8 Stützpunkten. Der zudem schwankende Fehler könnte ein Hinweis auf die statistische Ungenauigkeit des Vorhersagefehlers sein.

Im Vergleich zu der relativ einfachen $P\&F$ -Methode, die wie in Tabelle 3.1 vermerkt einen Fehler von etwa 0,02 mit mehr als einer Stützstelle und von etwa 0,3 mit einer Stützstelle erreicht, sind die Vorhersagefehler mit einer Stützstelle deutlich kleiner und damit besser, aber mit mehreren Stützstellen etwas größer.

3.4.7 Zusammenfassung

Im vorangehenden Kapitel 3.4 ist der Ablauf der SNSQ-Methode, ebenso wie Überlegungen zu dieser Methode genau erläutert. Durch den durchgängigen Systemgedanken, also die Anwendbarkeit auf generalisierte Systeme ergibt sich ein möglicher Transfer auf andere Modelle beziehungsweise Anwendungen.

Zwar wurde bei der beispielhaften Anwendung der durchweg eindimensionalen Referenzsysteme noch kein guter Vorhersagefehler erreicht, allerdings ist dies auf ein Problem der Vorhersage des Verhaltensvektors B mit R zurückzuführen. Trotz dieser sehr schlechten Vorhersage, waren die darauf aufbauenden Vorhersagen der eigentlichen Systeme verhältnismäßig gut.

Mit dem Referenzsystem S_4 ist ein gänzlich unbekanntes aber dennoch ähnliches System spontan prognostiziert und bewertet worden. Die Prognosegenauigkeit ist in einem ähnlichen Rahmen wie die von dem Referenzsystem S_3 . Da die KNN ED , R und F mit von S_3 stochastisch veränderten Systemen trainiert sind, überrascht diese ähnliche Prognosegenauigkeit und zeigt die Fähigkeit der SNSQ-Methode zur Extrapolation.

4 Anwendungsfälle

In dieser Dissertation sollen zwei Anwendungsfälle für die neue Methode detailliert dargestellt werden, um Vor- und Nachteile zu beleuchten. Dafür wurden zwei verschiedene Anwendungen innerhalb des AM ausgesucht. Der erste Anwendungsfall betrifft die ganzheitliche Optimierung von Fehlern im FFF. Im zweiten Fall geht es um eine Optimierung der Dichte im LPBF.

4.1 Untersuchung der Druckbettadhäsion (DBA) beim Fused Filament Fabrication (FFF)

Die Druckbettadhäsion (DBA) ist ein entscheidender Faktor im FFF. Eine zuverlässige Haftung des Produkts auf dem Druckbett ist eine Voraussetzung für den qualitativ hochwertigen Druckprozess. Mangelnde Adhäsion kann zu einer Vielzahl von Fehlern führen, darunter ungewollte Verformungen des Produktes oder im schlimmsten Fall zum kompletten Abbruch des Druckvorgangs. [57]

Aufgrund dieser zentralen Rolle der Druckbetthaftung im FFF-Prozess soll die DBA untersucht und vorhersagbar werden. Das Ziel ist es, mit möglichst wenigen relevanten Daten ein zuverlässige Vorhersage der DBA zu ermöglichen und somit die Qualität und Effizienz der FFF-Produktion zu steigern. In Verbindung mit einer einfachen Optimierungsmethode kann die Regressionsmethode zur Vorhersage von DBA zur Bestimmung von Druckparametern verwendet werden.

Nach Wenzel (2019) existieren 40 Druckfehler im Prozess des FFF 3D-Drucks. Diese Druckfehler werden durch 18 verschiedene Druckparameter beeinflusst, wobei der Eingabeparameter Drucktemperatur auf 22 Druckfehler einen Einfluss haben soll. Außerdem bestehen zwischen verschiedenen Parametern Wechselwirkungseffekte.

Zusammenfassend handelt es sich bei dem FFF 3D-Druck mit seinen Druckfehlern und Eingabeparametern um einen komplexen Prozess [58].

Der in dieser Anwendung verwendete Datensatz heißt „SAM-TUDa 3D Printer Bed Adhesion Measurements“[59]. Dabei wurden 1273 3D-gedruckte Würfel mit 1 cm Kantenlänge auf 4 baugleichen 3D-Druckern senkrecht von einer Druckplatte gezogen. Weitere Erläuterungen und grafische Darstellungen sind der zitierten Quelle zu entnehmen. Die resultierende Kraft ist der Zielwert der Messungen. Für die Messungen wurden 125 Eingabeparameter notiert. Die Eingabeparameter wurden mithilfe eines LHS systematisch variiert. Um diese Anzahl an Messungen durchführen zu können, wurden über mehrere Monate hinweg in einer nicht klimatisierten Umgebung Messungen durchgeführt. In dieser Zeit haben sich aber nicht nur Umgebungsvariablen verändert. Die vier verwendeten FFF-3D-Drucker, deren Zielgruppe die Verbraucher sind, wurden regelmäßig gereinigt, gewartet und kalibriert. Bei den Druckbettadhäsionsmessungen handelt es sich deshalb um dynamisch veränderliche Systeme.

An diesen Messdaten wurden verschiedene Studien durchgeführt, um ein Modell aufzustellen, um die DBA zu prognostizieren. Diese sind in Tabelle 4.1 aufgeführt. Die ersten Modelle sind von Wenzel et al.(2022) dokumentiert. Ein einfaches FFNN erreicht dabei eine Prognosegenauigkeit mit einem RMSE von 0,2240. Mit einem einfachen FFNN das mit der P&F-Methode trainiert wurde, also dem vorgeschlagenen Ansatz aus Kapitel 3.2, wird ein RMSE der Testdaten von 0,2057 erreicht. Dabei zu beachten ist, dass für das FFNN keine Hyperparameteroptimierung durchgeführt wurde und das Training von dem FFNN mehrfach wiederholt wurde, um einen Erwartungswert für das Ergebnis angeben zu können. In der Arbeit wurde ebenfalls die prinzipielle Idee eines Verhaltensvektors und damit einer Quantifizierung angesprochen, aber noch nicht umgesetzt [41].

Ein weiteres Modell wurde von Zhang (2022) [60] im Rahmen einer Masterthesis umgesetzt. Dabei ging es um die Anwendung eines Bayesian Neural Network. Das Modell ist mit einer Hyperparameteroptimierung entstanden und besteht aus einem FFNN, dessen Neuronen der letzten Schicht den probabilistischen Ansatz eines Bayesian Neural Networks verwenden. Das Modell von Zhang (2022) hatte mit 0,1562 eine deutlich verbesserte Prognosegenauigkeit der DBA.

Das nächste Modell wurde von Wenzel et al. (2023) aufgestellt. Dabei wurde ein Modell mithilfe einer primitiven Form der hier vorgestellten SNSQ-Methode aufgestellt. Im Vordergrund stand dabei noch die generelle Durchführbarkeit und Vorstellung der Methode. Daher wurde auch hier nicht mit einer automatischen

Hyperparameteroptimierung gearbeitet. Auch wurde in der Arbeit nicht die Problematik einer dynamischen Umgebung angegangen. Es wurde lediglich untersucht, wie viele Beobachtungen nötig sind, um das RNN R und FFNN F zu trainieren.

Das letzte zum Zeitpunkt dieser Arbeit veröffentlichte Modell wurde von Dou (2024) ebenfalls im Rahmen einer Masterarbeit aufgestellt. Dabei wurden diverse Regressionsmethoden zu sogenannten Ensemble-Methoden kombiniert. In dieser Arbeit ging es darum ein Modell mit einem möglichst geringen RMSE-Wert zu erstellen. Für das finale Modell wurde ein Ensemble-Modell aus 15 Regressionsmethoden ohne zeitlichen Kontext, das mit Voting den Prognosewert aussucht, mit einem Bootstrap Aggregated Artificial Neural Networks und einer Gaussian Process Regression kombiniert. Auf dieses Modell wird an dieser Stelle nicht weiter eingegangen, es kann aber in der Arbeit von Dou (2024) nachgelesen werden. Dieses Modell erreicht einen RMSE von 0,0768.

| Quelle | Methode | RMSE |
|---------------------------|----------------------------------|--------|
| Wenzel et al. (2022) [41] | FFNN | 0,2240 |
| Wenzel et al. (2022) [41] | P&F mit FFNN | 0,2057 |
| Zhang (2022) [60] | Bayesian Neural Network | 0,1562 |
| Wenzel (2023) [55] | SNSQ | 0,1126 |
| Dou (2024) [61] | ML Ensemble Methode | 0,0768 |
| | SNSQ über alle Beobachtungen | 0,0988 |
| | SNSQ jeweils nächste Beobachtung | 0,0898 |

Tabelle 4.1: Vergleich der erreichten RMSE für den Datensatz „SAM-TUDa 3D Printer Bed Adhesion Measurements“[59]

Für die Anwendung der SNSQ-Methode in dieser Arbeit wird auf die detaillierte Architektur der verschiedenen KNN nicht eingegangen. Diese ist mittels einer Hyperparameteroptimierung bestimmt worden und die genauen Werte bringen somit keinen Mehrwert. Alle speziell ausgewählten Details werden jedoch erwähnt. Im ersten Schritt der SNSQ-Methode wird ein KNN analog zu Wenzel et al. (2022) und Kapitel 3.2 trainiert. Dabei wird ein FFNN mithilfe von P&F mit Daten aus der Literatur trainiert. Aufgrund der geringen Datenmenge wird eine lineare Interpolation vorgenommen, um das Training zu ermöglichen.

Im nächsten Schritt wird das SDNN mit Zufallszahlen gemäß der Normalverteilung und einem σ_{RAN} von 0,2 analog zum Unterkapitel 3.4.4 durchgeführt. Die verwendeten Drucker sind baugleich und sollten sich dementsprechend ähnlich

verhalten. Hierbei wird jedes trainierte FFNN drei mal stochastisch mit einem einzelnen zufälligen Punkt umtrainiert und für das SDNN verwendet. Durch die Mehrfachverwendung der FFNN kann vor allem Rechenzeit gespart werden. Insgesamt wurden 3000 Systeme und deren Systemantworten \mathcal{Z} für die Definition von \mathcal{B} stochastisch generiert. Der Grund hierfür ist ebenfalls wieder eine Minimierung der Rechenzeit. Mit 3000 generierten Systemen ist das nachfolgende Training des AE erfolgreich. Der CFV und damit die zentrale und kleinste Schicht des AE ED besteht aus 12 Neuronen. Dieser Wert ist von einer Hyperparameteroptimierung ermittelt worden. Dabei ist in dem Optimierungsziel dieser Hyperparameteroptimierung entsprechend der 1. Forderung aus Kapitel 3.4.2 mit einem Term der CFV berücksichtigt worden. Das Optimierungsziel ist wie folgt, mit $\mathcal{L}_{\text{val BP}}$ als \mathcal{L} der Validierung des BP-Trainingsalgorithmus und n_{CFV} als Länge des CFV:

$$\min \left(\frac{\mathcal{L}_{\text{val BP}}}{n_{\text{CFV}}} \right)$$

Für die Projektion der Räume \mathcal{Z} auf einen Vektor wird ein LHS-Sampling mit 256 Punkten gewählt. Dieses Sampling wird für die nachfolgenden Schritte ebenfalls verwendet und es wird kein Zufallszahlen-Sampling durchgeführt. Die Auswahl der Hyperparameter von RNN R und FFNN F erfolgt mit einer einfachen Hyperparameteroptimierung alleine über $\mathcal{L}_{\text{val BP}}$.

In Abbildung 4.1 ist der RMSE über der Anzahl der Testbeobachtungen aufgetragen. Für das gezeigte Diagramm wird eine bestimmte Anzahl an Beobachtungen verwendet, um mithilfe des RNN R einen Verhaltensvektor B zu prognostizieren. Anschließend wird auf Basis dieses Verhaltensvektors und zusätzlichen Beobachtungen eine Prognose mit dem FFNN F erstellt. Für diese Prognose wird anschließend der entsprechende RMSE berechnet. In Gelb ist der RMSE aufgetragen, der ausschließlich mit jenen Beobachtungen berechnet wurde, die an der Prognose von B beteiligt sind. In Rot ist hingegen der RMSE aufgetragen, der über alle Testbeobachtungen ermittelt wird. Auffällig ist hier in der roten Kurve die plötzlich wechselnde RMSE, die eine untere Grenze bei circa 0,9 zu haben scheint. Dies ist nicht verwunderlich, da es sich um ein dynamisch veränderliches System handelt und die 3D-Drucker gewartet und neu kalibriert wurden. Die Beobachtungen sind chronologisch sortiert, die Veränderungen am Drucker können also nachvollzogen werden. Die rote Kurve erreicht nach 6 Beobachtungen erstmals einen RMSE kleiner 0,1. Dies entspricht in etwa der Anzahl an Beobachtungen, die erforderlich ist, um für einen baugleichen unbekanntem 3D-Drucker Prognosen zur DBA mit einem

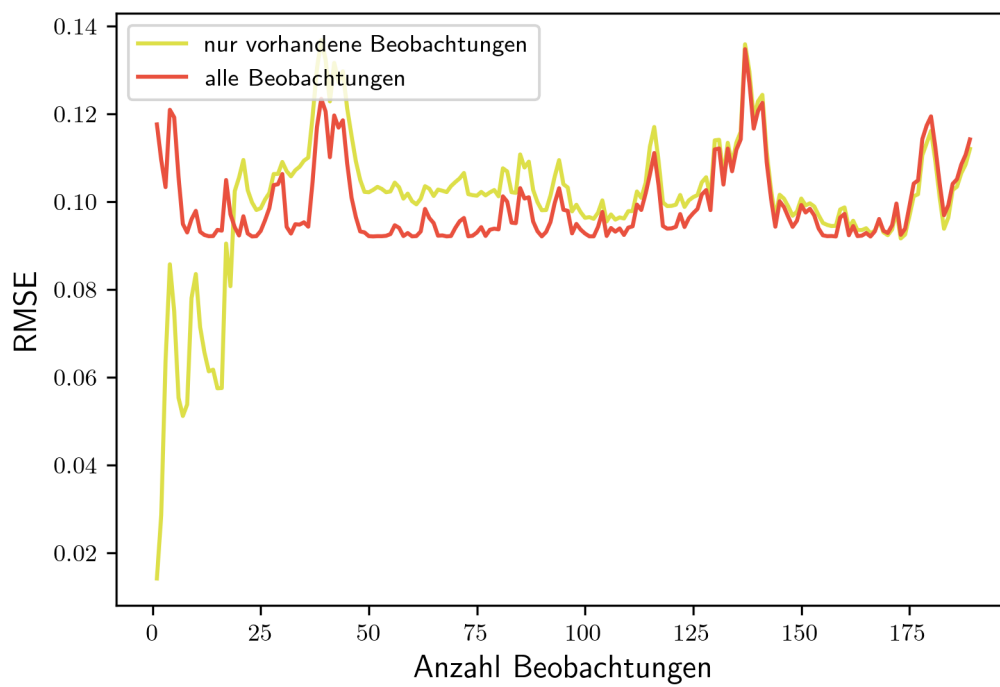


Abbildung 4.1: RMSE in Abhängigkeit von der Anzahl der Testbeobachtungen. In gelb den RMSE für die Beobachtungen, die bei der Prognose von B beteiligt sind, während die rote Kurve den RMSE über alle Testbeobachtungen darstellt.

RMSE von circa 0,1 zu stellen. Den Ergebnissen aus Tabelle 4.1 zufolge ist dieser Fehler, der mit einem einfachen FFNN erreicht wurde vergleichbar mit dem Fehler, der mit einer Ensemblemethode erreicht wurde. Insgesamt erreicht die Methode einen RMSE von 0,0988.

Die gelbe Kurve startet bei einem sehr niedrigen Wert von unter 0,02. Dies ist die aggregierte Ungenauigkeit aus dem Training der beiden KNN. Hier wird eine einzelne Beobachtung verwendet, um ein B zu prognostizieren und dieses B wird auch für die Vorhersage verwendet. Die gelbe Kurve stabilisiert sich vergleichsweise früh in einem Bereich zwischen 0,05 und 0,08. Bei 17 Beobachtungen ist der RMSE das erste Mal über dem RMSE von 0,1. Hier lässt sich vermuten, dass möglicherweise eine Wartung oder Reinigung durchgeführt wurde. Dennoch bleibt die Vorhersagegenauigkeit über kurze Abschnitte deutlich unterhalb von 0,1. Insgesamt ergibt sich für die Methode so ein RMSE von 0,0997.

In der Anwendung des SNSQ-Methode ist jedoch nicht die gleichzeitige Vorhersage aller Beobachtungen relevant. Da es sich um ein dynamisches System handelt, ist nur die Vorhersage der jeweils nächsten Beobachtung oder einer kurzen Folge von Beobachtungen insteressant. In Abbildung 4.2 ist der RMSE für die jeweils nächsten Beobachtungen dargestellt. Der Verhaltensvektor B für die Vorhersage wird über alle vorangegangenen Beobachtungen erstellt. Die gelbe Kurve, bei der der RMSE nur für die jeweils nächste Beobachtung bestimmt wird, hat die deutlichsten Ausreißer. Dabei bleibt zu vermuten, dass die Ausreißer nach oben jeweils einen Grund in der Anwendung haben. Die 3D-Drucker wurden, wie schon zuvor erwähnt, während der Versuche regelmäßig gewartet und gereinigt. Dieses Vorgehen war in erster Linie notwendig für den Betrieb der 3D-Drucker. Allerdings hilft es auch, um einen möglichst realistischen Datensatz im Sinne von realistischen unbekanntem Einflüssen zu bekommen. Schließlich müssen 3D-Drucker von Zeit zu Zeit gereinigt und gewartet werden. Daher ist die Erwartungshaltung der Anwender nach einer solchen Wartung nicht, dass die Prognosemethode sehr gute Ergebnisse liefert. Für die alleinige Prognose der nächsten Beobachtung, erzielt die Methode einen RMSE von 0,0898.

Wird zudem jeder deutliche Ausreißer, bei dem der Vorhersagefehler plötzlich über 0,05 ansteigt, erklärbar und ausgeschlossen, verbessert sich der RMSE sogar auf 0,0667. Dieser Wert ist allerdings dadurch verfälscht, dass besonders schlechte Schätzungen nicht verwendet worden sind. Allerdings kann dieser Wert einen Trend aufzeigen, der möglicherweise in der Anwendung für ein Problem erreicht wird, das schwer vorherzusagen ist, weil es ein hochdimensionales Problem in einer

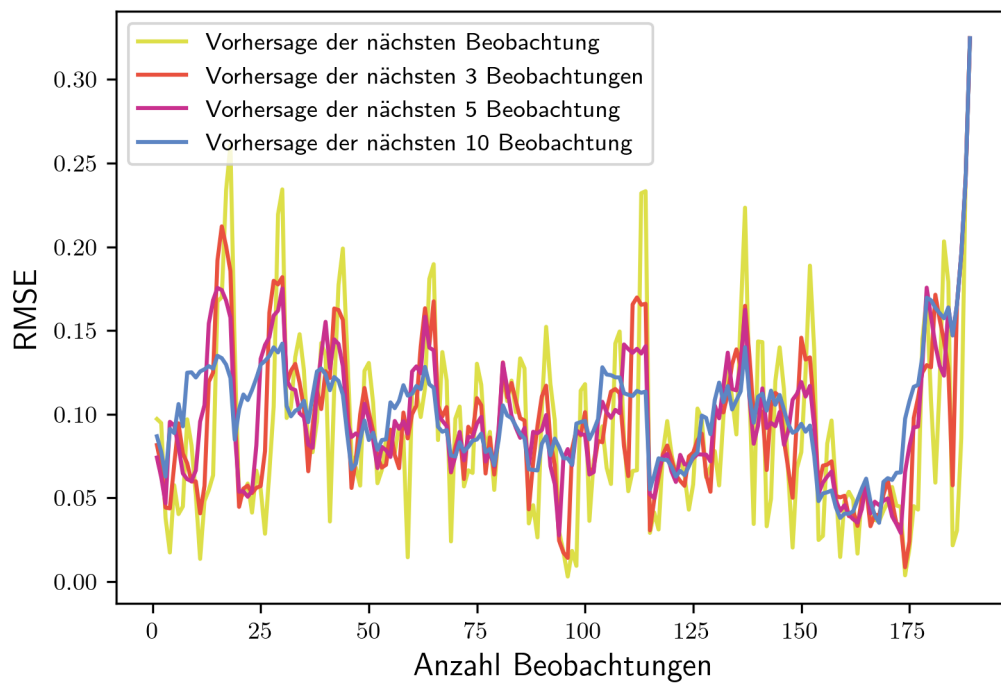


Abbildung 4.2: RMSE in Abhängigkeit von der Anzahl der Testbeobachtungen zur Prognose vom Verhaltensvektor B . Dargestellt ist der RMSE für die jeweils nächsten Beobachtungen

dynamisch veränderlichen Umgebung ist. Für die anderen Methoden in Tabelle 4.1 macht eine solche Betrachtung keinen Sinn, da sie keine dynamische Anpassung der Prognose anhand von vergangenen Beobachtungen vornehmen.

Zusammenfassend zeigt die neue SNSQ-Methode eine höhere Prognosegenauigkeit in diesem Datensatz. Der verwendete Datensatz lässt sich mit herkömmlichen Methoden nur sehr schwer vorhersagen, da er zum einen sehr viele Eingangsparameter hat und zum anderen sich zeitlich verändert. Besonders hervorzuheben ist die deutliche Verbesserung der Prognosegenauigkeit eines einfachen FFNN durch die Verwendung der SNSQ-Methode von 0,2057 auf 0,0898. Sollte das deutlich komplexere Modell von Dou (2024) ebenfalls den Verhaltensvektor B zur Prognose verwenden, ist hier auch eine deutliche Verbesserung zu erwarten.

4.2 3D-Druck mit Kupfer im Laser Powder Bed Fusion (LPBF)

Die Motivation hinter dieser Arbeit liegt im wachsenden Interesse an der Verwendung von reinem Kupfer im LPBF, vor allem aufgrund der hervorragenden thermischen und elektrischen Eigenschaften des Materials, die es besonders für die Anwendung in der Luft- und Raumfahrt interessant machen [62]. Bisherige Forschungen haben sich jedoch fast ausschließlich auf die Optimierung der relativen Dichte durch zahlreiche Experimente konzentriert, was mit einem hohen zeitlichen und materiellen Aufwand verbunden ist. Die Anwendung der SNSQ-Methode hat das Ziel diesen experimentelle Aufwand zu reduzieren, indem die relative Dichte der Kupferproben auf Basis vorhandener Datensätze aus der Literatur prognostiziert wird. Durch die Anwendung des KT eröffnen sich zukünftige Potenziale, gezielt Mikrostrukturen wie Korngröße und -form zu beeinflussen. Dies könnte dazu führen, dass Bauteile in verschiedenen Bereichen unterschiedliche funktionale Eigenschaften aufweisen und die Effizienz und Nutzbarkeit erheblich gesteigert werden [63].

Der 3D-Druck mit Kupfer im Laser Powder Bed Fusion (LPBF)-Prozess wurde im Rahmen einer Masterarbeit von Sun (2024) gestartet, wobei Daten erhoben und erste Anwendungen der SNSQ-Methode durchgeführt wurden. Die vorliegende Arbeit knüpft an diese Vorarbeiten an und konzentriert sich auf die umfassende Auswertung und Optimierung des Prozesses. Das Ziel der Masterarbeit ist es, die Methodik in einem weiteren Kontext anzuwenden, um detaillierte Erkenntnisse über die relevanten Prozessparameter und deren Einfluss auf die Porosität der

hergestellten Kupferprodukte zu gewinnen. Die Ergebnisse der Anwendung der SNSQ-Methode werden im Kontext eines weiteren Beispiels präsentiert.

Die für das Training der P&F-Methode verwendeten Daten stammen aus verschiedenen Literaturquellen. Hierzu wurde eine umfassende Literaturrecherche durchgeführt, um geeignete Datensätze zu identifizieren. Ein wichtiges Kriterium zur Auswahl der Datensätze ist die Orthogonalität der angewendeten Versuchspläne, sowie eine ausreichende Anzahl an Experimenten. Insgesamt wurden Daten aus zehn verschiedenen Quellen verwendet. Wie in Abschnitt 3.2.1 beschrieben, wurde anschließend eine Regressionsanalyse durchgeführt, um ein FFNN zu trainieren. Nach dem Vergleich verschiedener Regressionsmethoden fiel die Wahl schließlich auf eine Random-Forest-Regression, einem Ensemble Lernverfahren, das Vorhersagen durch Mittelung vieler Entscheidungsbäume trifft. [63].

Mit den Daten dieser Regression wurde schließlich das FFNN F mit der P&F-Methode trainiert. Diese trainierten F bieten die Grundlage für das darauf folgende SDNN und auch für den in Abbildung 4.3 gezeigten Vergleich. Die Trainingspunkte für das SDNN wurden nicht mit einer Normalverteilung bestimmt, sondern unabhängig von dem trainierten FFNN mit der Gleichverteilung $\text{Unif}\{0,1\}$ nach Gleichung (3.14). Dadurch wird die gesamte Komplexität der Methode verkleinert, sowie die generierten Systeme sind nicht auf einen kleineren Bereich um die Originalkurve beschränkt. Für das SDNN wurde jedes trainierte Modell F jeweils dreimal mit einem einzelnen, zufälligen Punkt umtrainiert, wodurch insgesamt 10.000 neue synthetische Systeme generiert wurden. Diese Systeme wurden mithilfe LHS mit 32 Samples auf einen Vektor projiziert. Anschließend wurde die Größe des CFV zu 5 gesetzt und der AE ED trainiert. Anschließend wurde noch das RNN R und das FFNN F trainiert.

Die in Abbildung 4.3 verwendeten Testdaten zeigen mit der P&F-Methode einen fast durchweg kleineren Fehler. Insgesamt wurde ein RMSE von 0,157 erreicht. Mit der SNSQ-Methode wurde ein RMSE von 0,273 erreicht. Insgesamt hat also der Verhaltensvektor B der von R berechnet wird, dem FFNN F nicht bei der Prognose geholfen, sondern nur behindert. Der Grund dafür könnte in der Stabilität des LPBF-Prozesses liegen. Im Gegensatz zu den 3D-Druckern im ersten Anwendungsfall, sind LPBF-Maschinen nicht an Verbraucher als Zielgruppe gerichtet. Sie verfügen über einen temperierten und mit Schutzgas gefüllten Bauraum. Ebenfalls ist das Druckmaterial für einen LPBF-Prozess deutlich besser überwacht und bekannt, als für den FFF-Prozess [58].

Zusammenfassend zeigt sich keine gute Leistung der SNSQ-Methode in dieser



MSE of Relative Density in Percentage Points: Section 1 vs. Section 2

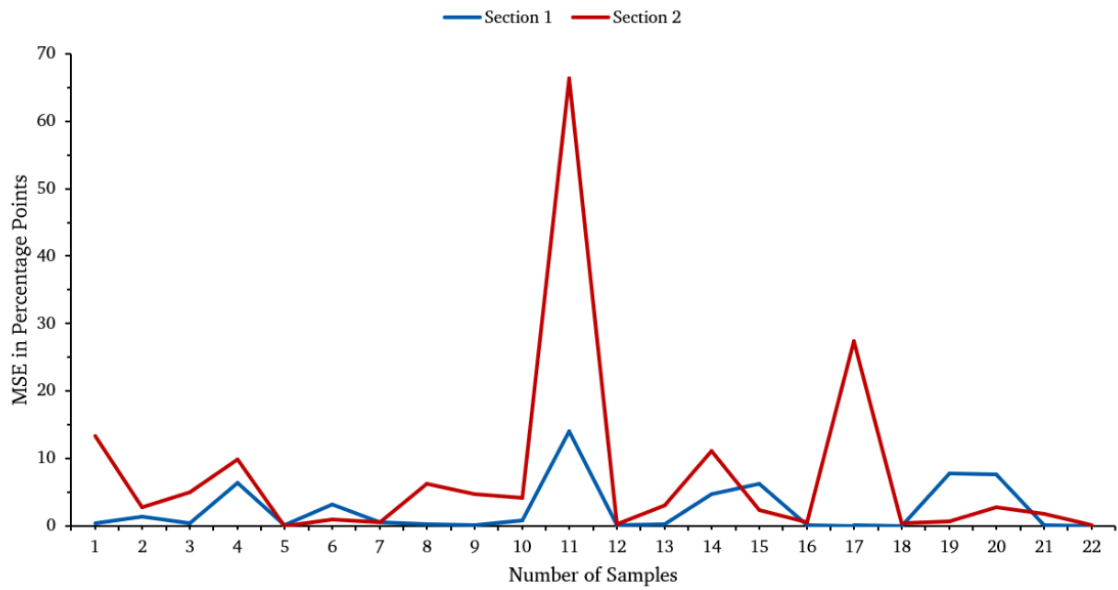


Abbildung 4.3: Vergleich der Prognosegenauigkeit mit dem MSE zwischen der P&F-Methode (Section 1) und der SNSQ-Methode (Section 2) für die Porosität im Kupfer in Produkten des LPBF-Prozesses aus Sun (2014) [63]

Anwendung des AM. Allerdings hat Sun (2024) ebenfalls gezeigt, dass herkömmliche Regressionsmethoden keine guten Ergebnisse mit diesem Datensatz erzeugen. Durch den fehlenden dynamischen Anwendungsfall war die SNSQ-Methode einfach nicht nötig angewendet zu werden. Die P&F-Methode aus Kapitel 3.3 hat hingegen sehr viel besser funktioniert, als die anderen von Sun (2024) verwendeten Regressionsmethoden.

5 Fazit und zukünftige Arbeiten

Diese Arbeit widmet sich der Optimierung der Systemzuverlässigkeit von Prozessen, die sich in verschiedenen Kontexten des FFF und des LPBF dynamisch verändern. Ein Fokus liegt dabei im Sinne der Systemzuverlässigkeit auf der Betrachtung des Gesamtsystems, anstatt ausschließlich Einzelkomponenten zu analysieren. Der gewählte Ansatz fokussiert auf die Ein- und Ausgangsparameter und weniger auf interne Prozesse, was im Gegensatz zur gängigen wissenschaftlichen Arbeitsweise steht. Damit wird weder nach Zusammenhängen oder Struktur außerhalb der mathematischen Beziehungen gesucht, um ein umfassendes Verständnis zu erlangen. Im Gegensatz dazu liegt das Hauptaugenmerk auf den vorhandenen Beobachtungen und somit auf dem tatsächlichen Systemverhalten gemäß der Messdaten. Die Priorität liegt dabei eine konsistente Leistung für einen definierten Zweck gewährleisten zu können.

Diese Differenzierung ist insbesondere im Bereich des 3D-Drucks von Bedeutung, da seit langem ähnliche Herausforderungen im 3D-Druck bekannt sind. Obwohl die Ursachen vieler Druckfehler gut bekannt sind, fehlen bislang dauerhafte Lösungen zu ihrer vollständigen Beseitigung. Hier setzt die vorliegende Dissertation an und bietet einen innovativen Lösungsansatz zur Bewältigung dieser Problematik.

Die zentrale Forschungsfrage, wie proaktive Fehlervermeidung in der additiven Fertigung realisiert werden kann, wurde in dieser Arbeit am Beispiel der Druckbettadhäsion erfolgreich untersucht. Zwar beschränkt sich die Anwendung der Methodik bisher nur auf diesen spezifischen Fehler, jedoch zeigt die entwickelte Regressionsmethode, dass bereits mit begrenzten relevanten Daten, also Daten, die das Problem direkt betreffen, aussagekräftige Prognosen erstellt werden können. Durch die Nutzung von KNN-Modellen und stochastischer Datengenerierung konnte der Erwartungswert der Druckbettadhäsion vor dem eigentlichen Druckprozess prognostiziert werden.

Die Ergebnisse im Kapitel 4.1 deuten darauf hin, dass durch gezielte Vorhersage der

Druckbettadhäsion die Systemzuverlässigkeit verbessert werden kann, da eine Optimierung der Druckparameter hinsichtlich der Druckbettadhäsion möglich ist. Diese Arbeit zeigt vielversprechende Ansätze zur Optimierung der Druckfehlervermeidung in der additiven Fertigung und demonstriert das Potenzial der vorgestellten SNSQ-Methode zur Anwendung auf ähnliche Fragestellungen in dynamischen Systemen.

Ein Beitrag zum Stand der Technik lässt sich mit der SNSQ-Methode durch die neuartige Verbindung von stochastischer Datengenerierung und Systemquantifizierung im Kontext von Regressionen ableiten. Diese Methodik ermöglicht es, die Systemzuverlässigkeit auch bei knappen oder unvollständigen Daten signifikant zu verbessern, unter anderem indem das große Feld der synthetischen Datengenerierung im Bereich des ML verwendet und mit einer Systemquantifizierung kombiniert wird. Durch diese Systemquantifizierung wird eine gezielte Auswahl aus den synthetischen und beobachteten Daten getroffen. Durch diese Auswahl werden die Anforderungen an die Qualität der Datengenerierung deutlich gesenkt und vergleichsweise einfache Möglichkeiten der Datengenerierung können genutzt werden, da erst mit der Systemquantifizierung die relevanten Daten aus dem Pool der verfügbaren, generierten Daten ausgewählt werden. Die darauf aufbauende Prognose mit flexiblem Systemverhalten hebt sich dadurch ab, durch ein großes Maß an TL und sogar Spontanität, da nicht die Notwendigkeit der Anwendung eines Trainingsalgorithmus besteht, wie beispielsweise beim P&F. Durch die Steigerung der Prozesszuverlässigkeit leistet diese Arbeit einen wichtigen Beitrag durch die erfolgreiche Anwendung der vorgestellten Methode.

Die Einschränkungen für die SNSQ-Methode ergeben sich aus den Randbedingungen, die im Unterkapitel 3.4.1 vorgestellt werden. Die erste Einschränkung, die sich aus diesen Randbedingungen ergibt, ist, dass nur Informationen verarbeitet werden, die als Beobachtungen ($X \rightarrow Z$) eines Systems vorhanden sind. Durch den Einsatz alternativer Methoden zum Beispiel aus dem Bereich des TL im ersten Schritt der SNSQ-Methode lassen sich zusätzliche Informationen verwenden. Je nach Art der verfügbaren Informationen lässt sich ein anderer Teilschritt der SNSQ-Methode anpassen, um die verfügbaren Informationen zu nutzen. Beispielsweise kann das RNN zur Systemidentifikation angepasst werden, um auch echte bekannte Eingabeparameter des Systems, wie etwa eine Materialeigenschaft eines verwendeten Materials, als Eingabeparameter zu berücksichtigen. Damit fließt die Information des Eingabeparameters in die Systemidentifikation mit ein.

Eine weitere Herausforderung besteht im angewendeten KT. Dieser basiert auf der

Annahme, dass das System mit ähnlichen Daten zum relevanten System überhaupt ähnlich ist. Diese Annahme wird im Rahmen der erarbeiteten Methode auch nicht überprüft und für den Fall, dass diese Annahme überprüft wird, würde sich die Menge der benötigten Daten deutlich vergrößern.

Wie bei der Anwendung der SNSQ beim LPBF im Kapitel 4.2 zu sehen war, ergab sich durch die erhöhte Komplexität der SNSQ-Methode keine verbesserte Prognose im Vergleich zur einfachen P&F-Methode. Ein Grund dafür könnte sein, dass der Prozess des LPBF keine dynamische und unkontrollierbare Umgebung darstellt. Zwar wird durch die Verkleinerung des Raumes möglicher Systemverhalten die Anzahl der Parameter verringert, die für eine Prognose nötig sind. Allerdings kann dieser Effekt den Informationsverlust nicht ausgleichen, den die Information zwangsläufig erleiden, wenn sie mehrfach durch KNN abgebildet werden und mit diesen Abbildungen weiter gearbeitet wird. Somit bietet der Einsatz der komplexeren SNSQ-Methode für im Vergleich stabilere Prozesse wie das LPBF keinen zusätzlichen Nutzen.

Ein Vorteil der SNSQ-Methode zeigt sich, wenn sie außerhalb der additiven Fertigung oder der Systemzuverlässigkeit angewendet werden soll. Aufgrund des methodischen Ansatzes lassen sich schnell neue Anwendungsfelder erschließen, da sich nahezu alle Prozesse als Systeme darstellen lassen. Da die Methode an Systemen im Sinne der Systemzuverlässigkeit angewendet werden kann, ist die SNSQ-Methode fast universell einsetzbar. Die in der vorliegenden Dissertation dargestellten Anwendungsbeispiele sind zwar auf die AM-Verfahren FFF und LPBF beschränkt, jedoch lässt sich die Methode auf andere Prozesse anwenden, wie an den abstrakten Referenzsystemen in Kapitel 3.4.6 gezeigt wurde.

Die in der vorliegenden Dissertation erarbeitete SNSQ-Methode zur Optimierung der Systemzuverlässigkeit weist eine hohe Komplexität aufgrund der großen Anzahl an Anwendungsschritten und trainierten KNN auf. Einen Überblick über den Ablauf verschafft Abbildung 3.1. Um dieser Komplexität gerecht zu werden, werden konsequent Lösungen der einzelnen Anwendungsschritte verwendet, die eine möglichst geringe Komplexität haben. Der anfängliche KT wird mit einem einfachen P&F-Ansatz auf Basis eines FFNN gelöst. Die Datengenerierung wird mit dem BP-Trainingsalgorithmus und einem zufälligen Punkt stochastisch durchgeführt. Die Definition der Systemquantifizierung wird mit einem AE und die Identifizierung der Systemquantifizierung mit einem RNN bestehend aus zwei rekurrenten Schichten durchgeführt. Die eigentliche Prognose hingegen wird mit einem FFNN durchgeführt. Bei der Auswahl dieser Methoden wurde das Ziel verfolgt, die zu-

sammengefasste Komplexität der gesamten Methode möglichst gering zu halten. Dieses Ziel ist zum Nachweis der Machbarkeit dieser Methode notwendig. Daher wurde jeder Teilschritt der Methode bewusst so gelöst, dass möglichst Methoden mit geringer Komplexität verwendet werden.

Aufbauend auf dieser Arbeit lassen sich mehrere Ansätze für zukünftige Forschungsarbeiten identifizieren. Dabei lassen sich die Teilschritte durch komplexere Methoden ersetzen. Beispielsweise lässt sich die Anfängliche Regression durch einen MAML-Ansatz oder einen Ensemble-Ansatz nach [35] ersetzen. Die darauf folgende SDNN lässt sich mithilfe eines GAN [56] oder sogar mit einer weniger komplexeren Methode der Datengenerierung nutzen, indem während der Prognose der Dropout verwendet wird. Zur Verringerung der Rechenzeit kann eine Kombination mit Datenaugmentierung basierend auf mathematischen Transformationen erprobt werden. Zur Definition der Systemquantifizierung können Transformator-basierte Modelle [64] oder ein Variational Autoencoder [65] verwendet werden. Dadurch verändert sich auch entsprechend die Identifikation der Systemquantifizierung für unbekannte Systeme. Die eigentliche Prognose kann probabilistisch mit einem Bayesian Neural Network [66] ergänzt werden oder durch eine Ensemble-Regression ersetzt werden. So ergeben sich für zukünftige Arbeiten in jedem Anwendungsschritt der SNSQ Möglichkeiten eine Methode zu ersetzen.

Zusammenfassend stellt die vorliegende Dissertation die neuartige Methode SNSQ vor, welche darauf abzielt, Prognosen in dynamischen und unkontrollierbaren Umgebungen signifikant zu verbessern. Durch die stochastische Generierung eines breiten Prognosefunktionsraums, der unterschiedliche Systemverhalten integriert, ermöglicht SNSQ eine spontane Auswahl relevanter Systemverhalten und dazugehörigen Prognosefunktionen.

Diese Methode geht über herkömmliche Ansätze hinaus, indem sie nicht nur auf statischen Trainingsdaten basiert, sondern auch dynamische Anpassungen in Echtzeit erlaubt. Die praktische Anwendbarkeit von SNSQ wurde anhand verschiedener Prozesse im Bereich AM gezeigt, bei denen die Methode ihre Fähigkeit zur präzisen Vorhersage in komplexen, sich ändernden Umgebungen unter Beweis stellt.

Glossar

- AE Autoencoder** Ein neuronales Netzwerk, das verwendet wird, um effiziente Codierungen von Daten zu erstellen. 13, 26, 50, 52, 58, 59, 62, 63, 64, 70, 75, 81
- AM Additive Manufacturing** Ein Verfahren zur Herstellung dreidimensionaler Objekte durch das typischerweise schichtweise Hinzufügen von Material. 2, 5, 7, 8, 22, 67, 77, 81, 82
- BP Backpropagation** Der allgemein verwendete Trainingsalgorithmus von Künstlichen Neuronalen Netzen. Dabei wird die Ableitung der Loss-Funktion erstellt und die Loss-Funktion iterativ minimiert. 11, 12, 34, 55, 61, 70, 81
- CFV Compressed Feature Vector** Eine komprimierte Darstellung von Merkmalen in einem vektoriellen Format. 13, 50, 59, 70, 75
- DBA Druckbettadhäsion** Das Phänomen der Haftung von geschmolzenem Material auf dem Druckbett in der additiven Fertigung. 67, 68, 70
- FFF Fused Filament Fabrication** Ein Verfahren der additiven Fertigung, bei dem Filamente aus Kunststoff geschmolzen und schichtweise aufgebaut werden. 5, 6, 7, 8, 23, 67, 68, 75, 79, 81
- FFNN Feedforward Neural Network** Ein einfacher Typ eines neuronalen Netzwerks, bei dem Informationen nur in eine Richtung durch das Netzwerk fließen. 9, 12, 13, 14, 18, 27, 30, 31, 33, 35, 36, 41, 42, 43, 46, 47, 48, 52, 59, 61, 63, 64, 65, 68, 69, 70, 72, 74, 75, 81
- GAN Generative Adversarial Network** Ein neuronales Netzwerk, das in der Lage ist, neue Daten zu generieren, die den Trainingsdaten ähneln. 55, 82

-
- GRU Gated Recurrent Unit** Eine vereinfachte Version von LSTM, die auch in RNNs verwendet wird, um sequentielle Daten zu verarbeiten. 14, 15, 16
- KNN Künstliche Neuronale Netze** Netzwerke, die inspiriert sind von biologischen neuronalen Netzwerken und in der Lage sind, Muster zu erkennen und zu lernen. 2, 3, 9, 10, 11, 12, 14, 17, 18, 19, 22, 23, 35, 36, 44, 45, 47, 48, 49, 50, 53, 54, 55, 56, 59, 64, 66, 69, 72, 79, 81, 85
- KT Knowledge Transfer** Dies beschreibt den allgemeinen Prozess, in dem Wissen von einer Domäne oder einem System auf eine andere übertragen wird. 16, 17, 21, 22, 23, 33, 74, 80, 81
- LHS Latin-Hypercube-Sampling** Eine statistische Methode zur Probenahme multidimensionaler Variablenräume. 19, 33, 43, 56, 57, 68, 70, 75
- LLM Large Language Model** Große neuronale Modelle, die zum Verständnis und zur Erzeugung von Text in natürlicher Sprache trainiert werden. 55
- LPBF Laser Powder Bed Fusion** Ein Verfahren der additiven Fertigung, bei dem ein Laser Metallpulver schmilzt und das Bauteil Schicht für Schicht aufgebaut wird. 6, 7, 67, 74, 75, 76, 79, 81
- LSTM Long Short-Term Memory** Eine spezielle Form eines RNN, das lange Abhängigkeiten in Sequenzen speichern und verarbeiten kann. 14, 15, 16
- MAML Model-Agnostic Meta-Learning** Eine Methode im Bereich des TL, die darauf abzielt, ein Modell so zu trainieren, dass es sich schnell an neue Aufgaben anpassen kann. 18, 82
- ML Maschinelles Lernen** Ein Teilgebiet der künstlichen Intelligenz, das Algorithmen verwendet, um aus Daten zu lernen und Vorhersagen zu treffen. 8, 9, 12, 13, 16, 17, 55, 69, 80, 84, 85
- OHE One-hot-Encoding** Eine Methode zur Projektion von kategorischen Variablen als Binärvektoren. 35, 36, 39, 44
- P&F Pre-training und Fine-tuning** Eine Methode im ML, bei der ein Modell zunächst auf einem großen Datensatz vortrainiert und anschließend mit einem kleineren Datensatz feinabgestimmt wird. 17, 18, 19, 23, 24, 27, 38, 44, 45, 46, 47, 59, 66, 68, 69, 75, 76, 77, 80, 81

PINN Physics Informed Neural Networks Neuronale Netzwerke, die physikalische Gesetze in die Architektur des Netzwerks integrieren. 18, 19, 23, 27

RMSE Root Mean Square Error Ein Maß für den Unterschied zwischen vorhergesagten Werten und den tatsächlichen Werten. 11, 33, 63, 64, 65, 66, 68, 69, 70, 71, 72, 73, 75

RNN Recurrent Neural Network Ein neuronales Netzwerk, das für die Verarbeitung von Sequenzen von Daten entwickelt wurde. 14, 15, 26, 51, 57, 58, 59, 63, 64, 69, 70, 75, 80, 81

SDNN Stochastische Datengenerierung mit Neuronalen Netzen Eine Datengenerierung die darauf abzielt, der Ausgabe eines KNN eine gewisse Zufälligkeit hinzuzufügen, um eine veränderte Ausgabe zu erhalten. 24, 45, 59, 60, 61, 69, 70, 75, 82, 85

SNSQ Stochastische Neuronale Systemquantifizierung Die Verbindung einer SDNN mit einer Identifikation eines Systems anhand von Beobachtungen zum Zweck einer spontanen Prognose mit TL. 23, 24, 30, 45, 66, 68, 69, 72, 74, 75, 76, 77, 80, 81, 82

SSL Self Supervised Learning Eine unüberwachte Technik im Bereich des ML, bei der ein Modell lernt, indem es eigene Labels für die Trainingsdaten generiert, um die zugrunde liegenden Strukturen und Muster in den Daten zu erkennen. 17

TL Transfer Learning Im Bereich des ML ein Modell oder Teile eines Modells, das für ein Quellsystem trainiert wurde, als Ausgangspunkt für eine anderes, verwandtes Zielsystem verwendet wird. 17, 18, 23, 24, 26, 27, 30, 34, 37, 40, 41, 42, 43, 44, 49, 80, 84, 85

Literatur

- [1] Verordnung (EU) 2016/679. *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung) (Text von Bedeutung für den EWR)*. 2016.
- [2] *DIN EN ISO/ASTM 52900:2022-03, Additive Fertigung_ - Grundlagen_ - Terminologie (ISO/ASTM 52900:2021); Deutsche Fassung EN_ISO/ASTM 52900:2021*. Berlin. DOI: 10.31030/3290011.
- [3] I. Gibson. *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. 2nd ed. 2015. Springer eBook Collection Engineering. New York, NY: Springer, 2015. ISBN: 9781493921133. DOI: 10.1007/978-1-4939-2113-3.
- [4] C. Y. Foo u. a. „Three-Dimensional Printed Electrode and Its Novel Applications in Electronic Devices“. In: *Scientific reports* 8.1 (2018), S. 7399. DOI: 10.1038/s41598-018-25861-3.
- [5] M. Ponsford und N. Glass. „The night I invented 3D printing“. In: *CNN* (13.02.2014). URL: <https://edition.cnn.com/2014/02/13/tech/innovation/the-night-i-invented-3d-printing-chuck-hall/index.html>.
- [6] M. Popp und A. Gruska. „3D-Druckverfahren erklärt: FDM, FLM und FFF“. In: *Industry of Things* (22.03.2021). URL: <https://www.industry-of-things.de/3d-druckverfahren-erklart-fdm-flm-und-fff-a-95e2b6e85fea37eb1870f35e5bba259a/>.
- [7] G. Costabile u. a. „Cost models of additive manufacturing: A literature review“. In: *International Journal of Industrial Engineering Computations* (2017), S. 263–283. ISSN: 19232926. DOI: 10.5267/j.ijiec.2016.9.001.

-
- [8] W. Gao u. a. „The status, challenges, and future of additive manufacturing in engineering“. In: *Computer-Aided Design* 69 (2015), S. 65–89. ISSN: 00104485. DOI: 10.1016/j.cad.2015.04.001.
- [9] D. Thomas. „Costs, Benefits, and Adoption of Additive Manufacturing: A Supply Chain Perspective“. In: *The International journal, advanced manufacturing technology* 85.5-8 (2016), S. 1857–1876. ISSN: 0268-3768. DOI: 10.1007/s00170-015-7973-6.
- [10] S. A. M. Tofail u. a. „Additive manufacturing: scientific and technological challenges, market uptake and opportunities“. In: *Materials Today* 21.1 (2018), S. 22–37. ISSN: 13697021. DOI: 10.1016/j.mattod.2017.07.001.
- [11] A. L. Abeliensky, I. Martinez-Zarzoso und K. Prettnner. „3D printing, international trade, and FDI“. In: *Economic Modelling* 85 (2020), S. 288–306. ISSN: 0264-9993. DOI: <https://doi.org/10.1016/j.econmod.2019.10.014>. URL: <https://www.sciencedirect.com/science/article/pii/S0264999319304924>.
- [12] M. Farhan Khan u. a. „Real-time defect detection in 3D printing using machine learning“. In: *Materials Today: Proceedings* 42 (2021), S. 521–528. ISSN: 22147853. DOI: 10.1016/j.matpr.2020.10.482.
- [13] H. Gong u. a. „Influence of defects on mechanical properties of Ti–6Al–4V components produced by selective laser melting and electron beam melting“. In: *Materials & Design* 86 (2015), S. 545–554. ISSN: 02641275. DOI: 10.1016/j.matdes.2015.07.147.
- [14] M. Kujawa. „The influence of first layer parameters on adhesion between the 3D printer’s glass bed and ABS“. In: *Interdyscyplinarność badań naukowych 2017*. Hrsg. von Oficyna Wydawnicza Politechniki Wrocławskiej. Wrocław, Poland, 2017.
- [15] M. A. Nazan u. a. „An exploration of polymer adhesion on 3D printer bed“. In: *IOP Conference Series: Materials Science and Engineering* 210 (2017), S. 012062. ISSN: 1757-8981. DOI: 10.1088/1757-899X/210/1/012062.
- [16] M. Spoerk u. a. „Optimisation of the Adhesion of Polypropylene-Based Materials during Extrusion-Based Additive Manufacturing“. In: *Polymers* 10.5 (2018). DOI: 10.3390/polym10050490.
- [17] M. Spoerk u. a. „Effect of the printing bed temperature on the adhesion of parts produced by fused filament fabrication“. In: *Plastics, Rubber and Composites* 47.1 (2018), S. 17–24. ISSN: 1465-8011. DOI: 10.1080/14658011.2017.1399531.
- [18] G. Dellino und C. Meloni, Hrsg. *Uncertainty Management in Simulation-Optimization of Complex Systems*. Operations Research/Computer Science

-
- Interfaces Series. Boston, MA: Springer US, 2015. ISBN: 978-1-4899-7546-1. DOI: 10.1007/978-1-4899-7547-8.
- [19] G. L. Chen und K. Yanamandra N. Gupta. „Artificial Neural Networks Framework for Detection of Defects in 3D-Printed Fiber Reinforcement Composites“. In: *JOM* 73.7 (2021), S. 2075–2084. ISSN: 1543-1851. DOI: 10.1007/s11837-021-04708-9. URL: <https://link.springer.com/article/10.1007/s11837-021-04708-9>.
- [20] S. Deshwal, A. Kumar und D. Chhabra. „Exercising hybrid statistical tools GA-RSM, GA-ANN and GA-ANFIS to optimize FDM process parameters for tensile strength improvement“. In: *CIRP Journal of Manufacturing Science and Technology* 31 (2020), S. 189–199. ISSN: 17555817. DOI: 10.1016/j.cirpj.2020.05.009.
- [21] A. Dey und N. Yodo. „A Systematic Survey of FDM Process Parameter Optimization and Their Influence on Part Characteristics“. In: *Journal of Manufacturing and Materials Processing* 3.3 (2019), S. 64. DOI: 10.3390/jmmp3030064.
- [22] L. F. C. S. Dur ao u. a. „Optimizing additive manufacturing parameters for the fused deposition modeling technology using a design of experiments“. In: *Progress in Additive Manufacturing* 4.3 (2019), S. 291–313. ISSN: 2363-9512. DOI: 10.1007/s40964-019-00075-9.
- [23] C. C. Aggarwal. „Training Deep Neural Networks“. In: *Neural Networks and Deep Learning*. Hrsg. von Charu C. Aggarwal. Cham: Springer International Publishing, 2018, S. 105–167. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0\$\\backslash\$textunderscore.
- [24] *DARPA neural network study: Neuronale Netze. Studie des DARPA (U.S. Defense Advanced Research Project Agency)*. Fairfax: AFCEA International Press, 1988. ISBN: 0-916159-175. URL: <https://www.tib.eu/de/suchen/id/tema%3ATEMAE90043122022>.
- [25] K. Cho u. a. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. URL: <http://arxiv.org/pdf/1406.1078v3>.
- [26] S. Hochreiter und J. Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997), S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.
- [27] Junyoung Chung u. a. „Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling“. In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: <http://arxiv.org/abs/1412.3555>.

-
- [28] S. J. Pan und Q. Yang. „A Survey on Transfer Learning“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), S. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191.
- [29] S. Thrun und L. Pratt. *Learning to Learn*. Boston, MA: Springer US, 1998. ISBN: 978-1-4613-7527-2. DOI: 10.1007/978-1-4615-5529-2.
- [30] S. J. Pan und Q. Yang. „A Survey on Transfer Learning“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), S. 1345–1359. DOI: 10.1109/TKDE.2009.191.
- [31] Z. Zhao u. a. „A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations“. In: *Expert Systems with Applications* 242 (2024), S. 122807. ISSN: 09574174. DOI: 10.1016/j.eswa.2023.122807.
- [32] A. Ebbehøj u. a. „Transfer learning for non-image data in clinical research: A scoping review“. In: *PLOS digital health* 1.2 (2022), e0000014. DOI: 10.1371/journal.pdig.0000014.
- [33] S. Ahn u. a. „Variational Information Distillation for Knowledge Transfer“. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 62019, S. 9155–9163. ISBN: 978-1-7281-3293-8. DOI: 10.1109/CVPR.2019.00938.
- [34] C. Finn, P. Abbeel und S. Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. 2017. arXiv: 1703.03400 [cs.LG]. URL: <https://arxiv.org/abs/1703.03400>.
- [35] W. F. Satrya und J. Yun. „Combining Model-Agnostic Meta-Learning and Transfer Learning for Regression“. In: *Sensors (Basel, Switzerland)* 23.2 (2023). DOI: 10.3390/s23020583.
- [36] S. Cuomo u. a. *Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What’s next*. URL: <https://arxiv.org/pdf/2201.05624.pdf>.
- [37] S. Monaco und D. Apiletti. „Training physics-informed neural networks: One learning to rule them all?“ In: *Results in Engineering* 18 (2023), S. 101023. ISSN: 25901230. DOI: 10.1016/j.rineng.2023.101023.
- [38] M. Raissi, P. Perdikaris und G.E. Karniadakis. „Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations“. In: *Journal of Computational Physics* 378 (2019), S. 686–707. ISSN: 0021-9991. DOI: <https://doi.org/10.1016/j.jcp.2018.10.045>. URL: <https://www.sciencedirect.com/science/article/pii/S0021999118307125>.

-
- [39] Berkcan Kapusuzoglu und Sankaran Mahadevan. „Physics-Informed and Hybrid Machine Learning in Additive Manufacturing: Application to Fused Filament Fabrication“. In: *JOM* 72.12 (2020), S. 4695–4705. ISSN: 1543-1851. DOI: 10.1007/s11837-020-04438-4.
- [40] Berkcan Kapusuzoglu u. a. „Process Optimization Under Uncertainty for Improving the Bond Quality of Polymer Filaments in Fused Filament Fabrication“. In: *Journal of Manufacturing Science and Engineering* 143.2 (2021). ISSN: 1087-1357. DOI: 10.1115/1.4048073.
- [41] Sören Wenzel, Elena Slomski-Vetter und Tobias Melz. „Optimizing System Reliability in Additive Manufacturing Using Physics-Informed Machine Learning“. In: *Machines* 10.7 (2022), S. 525. DOI: 10.3390/machines10070525.
- [42] Y. Zhu u. a. „Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data“. In: *Journal of Computational Physics* 394 (2019), S. 56–81. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.05.024. URL: <https://www.sciencedirect.com/science/article/pii/S0021999119303559>.
- [43] Q. Zhu, Z. Liu und J. Yan. „Machine learning for metal additive manufacturing: predicting temperature and melt pool fluid dynamics using physics-informed neural networks“. In: *Computational Mechanics* 67.2 (2021), S. 619–635. ISSN: 1432-0924. DOI: 10.1007/s00466-020-01952-9. URL: <https://link.springer.com/article/10.1007/s00466-020-01952-9>.
- [44] J. C. Helton und F. J. Davis. „Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems“. In: *Reliability Engineering & System Safety* 81.1 (2003), S. 23–69. ISSN: 09518320. DOI: 10.1016/S0951-8320(03)00058-9.
- [45] C. W. Hansen, J. C. Helton und C. J. Sallaberry. „Use of replicated Latin hypercube sampling to estimate sampling variance in uncertainty and sensitivity analysis results for the geologic disposal of radioactive waste“. In: *Reliability Engineering & System Safety* 107 (2012), S. 139–148. ISSN: 09518320. DOI: 10.1016/j.ress.2011.12.006.
- [46] Gokulakrishnan Jothibabu und Saravana Kumar Gurunathan. „Surrogate Based Sensitivity Analysis of Part Strength due to Process Parameters in Fused Deposition Modelling“. In: *Procedia Computer Science* 133 (2018), S. 772–778. ISSN: 18770509. DOI: 10.1016/j.procs.2018.07.120.
- [47] D. J. Hopkins und G. King. „A Method of Automated Nonparametric Content Analysis for Social Science“. In: *American Journal of Political Science* 54.1

-
- (2010), S. 229–247. ISSN: 0092-5853. DOI: 10.1111/j.1540-5907.2009.00428.x.
- [48] W. Gao und F. Sebastiani. „From classification to quantification in tweet sentiment analysis“. In: *Social Network Analysis and Mining* 6.1 (2016). ISSN: 1869-5450. DOI: 10.1007/s13278-016-0327-z.
- [49] G. E. Hinton. „A Practical Guide to Training Restricted Boltzmann Machines“. In: *Neural networks: tricks of the trade*. Hrsg. von Montavon u. a. Bd. 7700. Lecture Notes in Computer Science. Berlin und Heidelberg: Springer, 2012, S. 599–619. ISBN: 978-3-642-35288-1. DOI: 10.1007/978-3-642-35289-8{\textunderscore}32.
- [50] T. Mikolov u. a. *Efficient Estimation of Word Representations in Vector Space*. URL: <http://arxiv.org/pdf/1301.3781>.
- [51] V. Katragadda. „Dynamic Customer Segmentation: Using Machine Learning To Identify and Address Diverse Customer Needs In Real-Time - IRE Journals“. In: *IRE Journals* 5.10 (2024), S. 278–286. URL: <https://www.irejournals.com/paper-details/1703349>.
- [52] H. Taherdoost. „Enhancing Social Media Platforms with Machine Learning Algorithms and Neural Networks“. In: *Algorithms* 16.6 (2023), S. 271. DOI: 10.3390/a16060271.
- [53] L. van der Maaten und G. E. Hinton. „Visualizing Data using t-SNE“. In: *Journal of Machine Learning Research* 9 (2008), S. 2579–2605. URL: <https://api.semanticscholar.org/CorpusID:5855042>.
- [54] C. E. Shannon. „A Mathematical Theory of Communication“. In: *Bell System Technical Journal* 27.3 (1948), S. 379–423. DOI: <https://doi.org/10.1002/j.1538-7305.1948.tb01338.x>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1948.tb01338.x>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x>.
- [55] S. Wenzel, E. Slomski-Vetter und Tobias Melz. „Zuverlässigkeitsoptimierung in der additiven Fertigung durch physical informed neural networks (PINN) – Effektive Optimierung der Druckparameter für eine hohe Bauteilqualität“. In: *Technische Zuverlässigkeit*. 2023, S. 283–298. DOI: 10.51202/9783181024096-283.
- [56] I. Goodfellow u. a. *Generative Adversarial Networks*. URL: <http://arxiv.org/pdf/1406.2661>.
- [57] R. Devicharan und R. Garg. „Optimization of the Print Quality by Controlling the Process Parameters on 3D Printing Machine“. In: *3D Printing and Additive Manufacturing Technologies*. Hrsg. von L. Jyothish Kumar, Pulak M. Pandey

-
- und David Ian Wimpenny. Singapore: Springer Singapore, 2019, S. 187–194. ISBN: 978-981-13-0304-3. DOI: 10.1007/978-981-13-0305-0.
- [58] S. Wenzel, E. M. Slomski und T. Melz. „Zuverlässigkeitsanalyse mittels KI in der additiven Fertigung“. In: *Smarte Strukturen und Systeme, Symposium für smarte Strukturen und Systeme - 4Smarts, Darmstadt*. Darmstadt: Shaker Verlag GmbH, 2019. ISBN: 978-3-8440-6425-4.
- [59] Sören Wenzel, Elena Slomski-Vetter und Tobias Melz. *SAM-TUDa 3D Printer Bed Adhesion Measurements*. 2022. DOI: 10.6084/m9.figshare.19854967.v1. URL: https://figshare.com/articles/software/Print_bed_adhesion/19854967.
- [60] M. Zhang. „Comparison of Methods of Uncertainty Estimation in Machine Learning: Methodenvergleich der Unsicherheitsabschätzung beim Maschinellen Lernen“. Masterarbeit. Darmstadt: Technische Universität Darmstadt, Fachgebiet Systemzuverlässigkeit, Adaptronik und Maschinenakustik (SAM), 2022.
- [61] T. Dou. „Assessing the Usability of AI to Increase Reliability in Additive Manufacturing“. Masterarbeit. Darmstadt: Technische Universität Darmstadt, Fachgebiet Systemzuverlässigkeit, Adaptronik und Maschinenakustik (SAM), 2024.
- [62] S. Jadhav u. a. „Laser-based powder bed fusion additive manufacturing of pure copper“. In: *Additive Manufacturing* 42 (2021), S. 101990. ISSN: 2214-8604. DOI: <https://doi.org/10.1016/j.addma.2021.101990>. URL: <https://www.sciencedirect.com/science/article/pii/S221486042100155X>.
- [63] L. Sun. *In-situ Tailoring of Functional Properties for the LPBF-Process of Pure Copper*. Darmstadt, 2024. DOI: Laura. URL: <https://tuprints.ulb.tu-darmstadt.de/28404/>.
- [64] A. Vaswani u. a. *Attention Is All You Need*. URL: <http://arxiv.org/pdf/1706.03762>.
- [65] D. P. Kingma und M. Welling. *Auto-Encoding Variational Bayes*. URL: <http://arxiv.org/pdf/1312.6114>.
- [66] R. M. Neal. *Bayesian learning for neural networks: Zugl.: Toronto, Univ., Diss., 1995*. Bd. 118. Lecture notes in statistics. New York, Berlin und Heidelberg: Springer, 1996. ISBN: 978-0-387-94724-2. DOI: 10.1007/978-1-4612-0745-0.