IAS-Forschungsberichte

Forschungsbericht
Institut für Automatisierungstechnik und Softwaresysteme

Hrsg.: Prof. Dr.-Ing. Dr. h. c. Michael Weyrich

Benjamin Maschler

Eine Architektur für maschinelles Transfer-Lernen in industriellen Automatisierungssystemen

Band 2/2023

Universität Stuttgart

Eine Architektur für maschinelles Transfer-Lernen in industriellen Automatisierungssystemen

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Benjamin Maschler
aus Oranienburg

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. Michael Weyrich
Mitberichter:	Prof. Dr.-Ing. Tobias Meisen

Tag der mündlichen Prüfung: 25. September 2023

Institut für Automatisierungstechnik und Softwaresysteme
der Universität Stuttgart

2023

IAS-Forschungsberichte

Band 2/2023

Benjamin Maschler

Eine Architektur für maschinelles Transfer-Lernen in industriellen Automatisierungssystemen

D 93 (Diss. Universität Stuttgart)

Shaker Verlag
Düren 2023

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2023

ISBN 978-3-8440-9325-4
ISSN 1610-4781

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Wenn es ein Dorf braucht, um ein Kind großzuziehen
– was braucht es, damit das Kind anschließend promoviert?

Die vorliegende Arbeit entstand in weiten Teilen während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungstechnik und Softwaresysteme (IAS) der Universität Stuttgart.

Mein Dank gilt meinem Doktorvater, Institutsleiter Herrn Prof. Dr.-Ing. Dr. h. c. Michael Weyrich, für die Übernahme des Hauptberichts sowie die vielen fruchtvollen Diskussionen, Hinweise und Anregungen.

Herrn Prof. Dr.-Ing. Tobias Meisen danke ich für das Interesse an meiner Arbeit und die Übernahme des Mitberichts.

Allen Kolleg:innen am IAS gilt mein herzlichster Dank für die fortwährende Unterstützung, die sehr gute Zusammenarbeit und die vielen wertvollen und konstruktiven Diskussionen.

Ebenso gilt mein herzlicher Dank den Studierenden, die im Rahmen ihrer Master-, Studien-, Forschungs- und Bachelorarbeiten einen Beitrag zum Gelingen dieser Arbeit geleistet haben, und dem System universitärer Forschung in Deutschland, dass mir die Erstellung dieser Arbeit unter sehr guten Bedingungen ermöglicht hat.

Schließlich möchte ich meiner Partnerin Romy, meinen Freund:innen und meiner Familie von Herzen danken, dass sie mich auf meinem Weg begleiten und – wenn nötig auch mit Kritik – unterstützen.

Stuttgart, im Oktober 2023

Benjamin Maschler

Widmung

Diese Arbeit widme ich den Menschen, die mich inspirier(t)en und herausforder(te)n, für die kleinen und großen Zusammenhänge begeister(te)n und mich ermutig(t)en, die Spielräume des Lebens zu nutzen, mich an ihnen zu erfreuen, aber auch an ihnen zu wachsen und dabei meinen Weg zu finden. Die Welt wäre ein besserer Ort, wenn jeder das Glück hätte, Menschen wie Euch um sich zu haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BCE	**B**inary **C**ross **E**ntropy (dt.: Binäre Kreuzentropie)
BIRCH	**B**alanced **I**terative **R**educing and **C**lustering using **H**ierarchies (dt.: Ausgewogenes iteratives Reduzieren und Gruppieren unter Verwendung von Hierarchien)
CNN	**C**onvolutional **N**eural **N**etwork (dt.: Neuronales Faltungsnetzwerk)
CWRU	**C**ase **W**estern **R**eserve **U**niversity[1]
DBSCAN	**D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (dt.: Dichte-basierte räumliche Gruppierung verrauschter Anwendungen)
DMM	**D**ual **M**emory **M**ethod (dt.: Zwei-Gedächtnis-Methode)
DSR	**D**esign **S**cience **R**esearch (dt.: Wissenschaftliche Entwurfsforschung)
FCNN	**F**ully **C**onnected **N**eural **N**etwork (dt.: Vollvermaschte neuronale Netzwerke)
FEMTO-ST	**F**ranche-Comté **É**lectronique **M**écanique **T**hermique et **O**ptique – **S**ciences et **T**echnologies[1]
FFT	**F**ast **F**ourier **T**ransformation (dt.: Schnelle Fourier-Transformation)
FS	**F**ertigungsvarianten-**S**equenz
GRU	**G**ated **R**ecurrent **U**nit (dt.: Geschlossene, rekurrente Einheit)
IAS	**I**nstitut für **A**utomatisierungstechnik und **S**oftwaresysteme
IMS	**I**ntelligent **M**aintenance **S**ystems[1]
KL	**K**ullback-**L**eibler
LSTM	**L**ong **S**hort-**T**erm **M**emory (dt.: Langes Kurzzeitgedächtnis)
MFPT	Society for **M**achinery **F**ailure **P**revention **T**echnology[1]
MLDA	**M**ulti-**L**ayer **D**omain **A**daptation (dt.: Mehrschicht-Domänen-Adaption)

[1] Bei Eigennamen wird keine Übersetzung vorgenommen.

MMD	**M**aximum **M**ean **D**iscrepancy [between feature distributions] (dt.: Maximale Mittlere Abweichung [zwischen Merkmalsverteilungen])
MSE	**M**ean **S**quared **E**rror (dt.: Gemittelter, quadratischer Fehler)
ReLU	**R**ectified **L**inear **U**nit (dt.: Gleichrichtende Lineareinheit)
RUL	**R**emaining **U**seful **L**ifetime (dt.: Restlebensdauer)
SDA	**S**tacked **D**enoising **A**utoencoder (dt.: Mehrstufiger, entrauschender Autoencoder)
SELU	**S**caled **E**xponential **L**inear **U**nit (dt.: Skaliert-exponentielle Lineareinheit)
SI	**S**ilhouetten-**I**ndex
SoH	**S**tate **o**f **H**ealth (dt.: Gesundheitszustand)
TCA	**T**ransfer **C**omponent **A**nalysis (dt.: Transferkomponenten-Analyse)
UDAB	**U**nsupervised **D**omain **A**daption by **B**ackpropagation (dt.: Unüberwachte Domänen-Adaption durch Fehlerrückführung)

Begriffsverzeichnis

Anomaliedetektion: Erkennung von Abweichungen von einem als normal definierten Zustand einer Anlage(-nkomponente) oder eines Systems [1]

Ausfallvorhersage: Vorhersage des Zeitpunkts des Erreichens eines Zustands der Dysfunktionalität einer Anlage(-nkomponente) oder eines Systems [1]

Automatisierung: Umstellung von Verfahren, Prozessen oder Anlagen auf einen automatischen, üblicherweise Computer-gesteuerten Betrieb ohne Eingriff eines menschlichen Bedieners [2]

Clustering[2]: Gruppieren von Objekten anhand ihrer Proximität (auch: Ähnlichkeit) [3]

Datensilo: Daten, die aufgrund technischer oder organisatorischer Hürden nur einer kleinen Gruppe Anwender*innen zur Verfügung stehen

Datentransfer: Wissenstransfer mittels Quellproben(-vektoren) oder -merkmal(-svektoren) (auch: Instanz- bzw. Merkmalsrepräsentationstransfer)

Deep Learning[2]: Maschinelles Lernen mittels mehrschichtiger, künstlicher neuronaler Netze [4]

Mehrschichtiges Transfer-Lernen (engl. Deep Transfer Learning): Transfer-Lernen auf Basis von Deep-Learning-Methoden [5]

Modelltransfer: Wissenstransfer mittels (Parametern des) auf dem bzw. den Quellproblem(-en) trainierten Lernalgorithmus (auch: Parametertransfer)

Föderales Lernen (engl.: Federated Learning): Klasse von Lernalgorithmen, die Wissenstransfers zwischen dezentralen Clients und zentralem Server nutzen, um gemeinsam verschiedene Varianten eines Problems zu lösen ohne die zugrundliegenden Daten auszutauschen [5]

Industrielles Transfer-Lernen (engl.: Industrial Transfer Learning): Klasse von Lernalgorithmen, die Wissenstransfers von Quellproblem(en) auf ein Zielproblem nutzen, um Lernaufgaben im industriellen Kontext robuster, genauer oder dateneffizienter lösen

---

[2] Obwohl diese Arbeit auf Deutsch verfasst ist und dazu auch üblicherweise englischsprachig verwendete Fachbegriffe übersetzt wurden, wird in Einzelfällen auf eine Übersetzung verzichtet, wenn diese nach Meinung des Autors die Verständlichkeit des Textes reduzieren würde. Derartige Einzelfälle werden im Folgenden nicht weiter gekennzeichnet.

zu können. Es kann dabei sowohl kontinuierliches als auch Transfer-Lernen zum Einsatz kommen [5].

Kontinuierliches Lernen (engl.: Continual Learning): Klasse von Lernalgorithmen, die Wissenstransfers von Quellproblem(en) auf ein Zielproblem nutzen, um anschließend Quell- und Zielprobleme besser lösen zu können [5]

Label: Problem-spezifische Lösungsinformation einer einzelnen Probe [6]

Lernalgorithmus: In Software abgebildetes, mathematisches Verfahren, das aus Daten zielgerichtet allgemeine(-re) Zusammenhänge zur Lösung eines konkreten Problems ableiten kann [5]

Lernen mit wenigen Versuchen (engl.: Few Shot Learning): Sammelbegriff für Lernalgorithmen, deren Ziel es ist die Menge an benötigten Trainingsdaten zu minimieren. Methoden des kontinuierlichen und Transfer-Lernens werden häufig dazu verwendet [5].

Maschinelles Lernen: Fähigkeit technischer Systeme mittels mathematischer Verfahren aus Daten zielgerichtet allgemeine(-re) Zusammenhänge zur Lösung eines konkreten Problems abzuleiten [6]

Merkmal(-svektor): (Satz) Abstrakte(-r) Eigenschaft(-en) zur Beschreibung und Unterscheidung von Proben [7]

Partikularlösung: Lösung, die nur für einen spezifischen Anwendungsfall oder ein spezifisches Transfer-Problem nutzbar ist

Probe(-nvektor, engl.: Sample): Einzelnes Datenobjekt, Bestandteil eines Datensatzes [6]

Problem: Lernaufgabe, die sich durch Format und Art von Ein- und Ausgangsdaten sowie die Ein- und Ausgang verbindenden Zusammenhänge charakterisiert [5]

Prozess: Abfolge von Menschen oder Maschinen unter Einsatz von Ressourcen ausgeführter, aufeinander abgestimmter Teil-Aufgaben, die gemeinsam Materialien, Energie oder Informationen transformieren [8]

Quellproblem: Problem, das dem Lernalgorithmus bereits bekannt ist und von ihm gelöst werden kann [5]

Restlebensdauer: Verbleibende (Betriebs-)Zeit einer Anlage(-nkomponente) oder eines Systems bis zum Erreichen eines Zustands der Dysfunktionalität [1]

Robustheit: Eigenschaft einer Software(-komponente), auch mit ungültigen Eingangswerten oder unter ungünstigen Umgebungsbedingungen korrekt zu funktionieren [9]

Training: Prozess der automatischen Anpassung eines Lernalgorithmus bzw. seiner Parameter zur (Verbesserung der) Lösung eines Problems [10]

Transfer-Lernen (engl.: Transfer Learning): Klasse von Lernalgorithmen, die Wissenstransfers von Quellproblem(en) auf ein Zielproblem nutzen, um das Zielproblem anschließend besser lösen zu können [5]

Transfer-Problem (auch: Transfer Szenario): Beschreibung einer mittels Transfer-Lernen zu lösenden Kombination aus Quell- und Zielproblemen.

Validierung: Überprüfung eines Lösungsartefakts hinsichtlich seiner Erfüllung vorher festgelegter Anforderungen für einen spezifischen Anwendungsfall [7]

Wissenstransfer: Austausch von verknüpften Informationen hoher Komplexität, wobei die Form abstrakt, d.h. nicht menschen-verständlich, sein kann [5]

Zielproblem: Problem, das dem Lernalgorithmus bisher unbekannt ist und von ihm noch nicht gelöst werden kann [5]

Zusammenfassung

Industrielles Transfer-Lernen ermöglicht einen breiteren Einsatz von maschinellem Lernen in der Industrieautomatisierung, indem es die praktischen Herausforderungen konventionellen Deep Learnings überwindet: Lernalgorithmen müssen zwischen Anlagen und über Prozessgrenzen hinweg übertragbar sein und dabei Daten unterschiedlicher Qualität, Dimensionalität und Herkunft verwenden können, um mit geringen Anpassungsaufwänden auf Veränderungen der betrachteten (Teil-)Probleme reagieren zu können.

In dieser Arbeit wurde mittels Design Science Research angelehnt an Dual-Memory-Methoden und den Parametertransfer (Feinabstimmung) im Bereich der visuellen Objekterkennung eine modulare Architektur für industrielles Transfer-Lernen entwickelt: Eine statische, aber Anwendungsfall-spezifische Eingangsdatenverarbeitung führt eine Merkmalsextraktion aus. Die resultierenden Merkmalsvektoren können in der Folge über Clustering mit bereits bekannten Merkmalsvektoren im Transfermodul verglichen und von einem einfach nachtrainierbaren Ausgangsmodul weiterverarbeitet werden.

Zur Evaluation dieser Architektur wurden prototypische Realisierungen in drei Industrie-typischen Anwendungsfällen untersucht: Für die Pumpen einer Hydraulikpresse, die häufig wechselnde Produkte fertigt, wurde auf Basis von Druckverläufen eine Anomaliedetektion implementiert. Über einen Vergleich von Verläufen neuer Produkte mit denen bereits bekannter konnte eine deutliche Reduktion der benötigten Trainingsdatenmenge sowie eine gesteigerte Robustheit des Algorithmus erzielt werden. Für unter unterschiedlichen Einsatzbedingungen genutzte elektromechanische Relais wurde eine Ausfallvorhersage implementiert. Auch hier wurden über Vergleiche zwischen neuen Einsatzbedingungen mit bekannten Einsatzbedingungen deutliche Verbesserungen der Vorhersagegenauigkeit auch bei reduzierter Trainingsdatenmenge erreicht. Für unter unterschiedlichen Einsatzbedingungen genutzte Kugellager wurde auf Basis von Vibrationsdaten eine Ausfallvorhersage implementiert. Mittels Domänen-Adaption konnten hierbei erfolgreich Zusammenhänge aus bekannten, gelabelten Einsatzbedingungen auf neue, ungelabelte Einsatzbedingungen übertragen werden.

Im Rahmen der Evaluation konnte gezeigt werden, dass die vorgeschlagene Architektur auch mit wenigen Trainingsdaten robust gute Vorhersagequalitäten gewährleisten kann und so bspw. auch dynamische Probleme lösbar macht. Sie bietet darüber hinaus Ansätze, um auch auf ungelabelten Daten Regressions- oder Klassifikationsaufgaben ausführen zu können, und ermöglicht mit ihrem modularen Aufbau ein hohes Maß an Wiederverwendbarkeit.

Schlagwörter: Industrieautomatisierung, Lern-Architektur, Mehrschichtiges Lernen, Transfer-Lernen

Abstract

Industrial transfer learning enables a wider use of machine learning in industrial automation by overcoming the practical challenges of conventional deep learning: Learning algorithms must be transferable between assets and across process boundaries, using data of different quality, dimensionality and origin, in order to respond to changes in the (sub-)problems under consideration with little adaptation effort.

In this work, using design science research, a modular architecture for industrial transfer learning was developed based on dual-memory methods and finetuning in the field of visual object recognition: Static but use-case-specific input data processing performs a feature extraction. The resulting feature vectors can subsequently be compared via clustering with already known feature vectors in the transfer module and further processed by an easily retrainable output module.

To evaluate this architecture, prototypical implementations in three typical industrial use cases were investigated: For the pumps of a hydraulic press, which produces frequently changing products, an anomaly detection was implemented based on pressure curves. By comparing new products with known ones, a significant reduction of the required amount of training data as well as an increased robustness of the algorithm could be achieved. A wear prediction was implemented for electromechanical relays used under different operating conditions. Here, too, significant improvements in prediction accuracy were achieved by comparing new operating conditions with known ones, even using a reduced amount of training data. For ball bearings used under different operating conditions, a wear prediction was implemented using vibration data. By means of domain adaptation, correlations from known, labeled operating conditions were successfully transferred to new, unlabeled operating conditions.

In the evaluation, it could be demonstrated that the proposed architecture can robustly provide good prediction qualities even with few training data and thus, for example, also make dynamic problems solvable. Furthermore, it offers approaches to perform regression or classification tasks even on unlabeled data, and its modular structure allows a high degree of reusability.

Keywords: Deep Learning, Industrial Automation, Learning Architecture, Transfer Learning

1 Einleitung

1.1 Motivation und Problemstellung

Die produzierende Industrie ist inmitten eines extern getriebenen Transformationsprozesses hin zu mehr Effizienz, Flexibilität [11–13] und – nicht zuletzt – einer Reduzierung von schädlichen Auswirkungen auf die Umwelt [14–17]. Konzepte wie „Industrie 4.0“ oder das „Industrial Internet of Things“ tragen dem Rechnung, indem sie digitale, hoch-vernetzte Prozesse und damit Kooperation über vormals trennende Systemgrenzen hinweg ermöglichen [18–22].

Die Automatisierungstechnik stellt dies jedoch vor die Herausforderung immer komplexerer zu automatisierender Systeme, deren Funktionalitäten, Komponenten und Beziehungen häufig nicht mehr abschließend erfassbar sind [21–24]. Ein Ansatz der Komplexitätsbeherrschung bzw. Kostenbegrenzung ist hier die Nutzung künstlicher Intelligenz, also eines „Teilgebiet[s] der Informatik, welches versucht, menschliche Vorgehensweisen der Problemlösung auf Computern nachzubilden, um auf diesem Wege neue oder effizientere Aufgabenlösungen zu erreichen,“ [10] zur automatischen Generierung oder Anpassung von Automatisierungs-Funktionalitäten [21, 23, 24]. Dabei kommen verschiedenste Verfahren zum Einsatz.

Seit den 2000er Jahren erzielen Lernalgorithmen auf der Basis mehrschichtiger neuronaler Netzwerke, sogenanntes Deep Learning, bemerkenswerte Erfolge: Entsprechende Algorithmen demonstrieren auf so unterschiedlichen Gebieten wie Sprachverständnis [25], Bilderkennung [26] oder dem Spielen komplexer Strategiespiele [27] übermenschliche Fähigkeiten. Als Form des maschinellen Lernens extrahiert Deep Learning die Wissensbasis für die Intelligenz-Funktionalität automatisch aus den zur Verfügung gestellten Trainingsdatensätzen [4, 28, 29].

Auch in der Automatisierungstechnik werden Deep-Learning-Methoden vermehrt erprobt [30]: Ob in der vorausschauenden Instandhaltung [31], in der Anomaliedetektion [32] oder der Vorhersage von Kurvenverläufen [33] – Deep Learning verspricht daten-getrieben einen höheren Automatisierungsgrad in einer Vielzahl von Anwendungsfällen [34]. Die Übertragung von Konzepten aus anderen Domänen in die Automatisierungstechnik wird dabei jedoch von den im industriellen Kontext vielfach genutzten Zeitreihendaten erschwert, die an anderer Stelle nicht in diesem Ausmaß vorkommen [31, 35].

In der Praxis ist die Anwendung von Deep-Learning-Methoden in der produzierenden Industrie allerdings gehemmt [36–38]. Im folgenden Kapitel werden daher einige der dafür ursächlichen Herausforderungen vorgestellt.

1.2 Herausforderungen

Trotz der in Kapitel 1.1 dargestellten Potentiale von Deep Learning in der Automatisierungstechnik und der zahlreichen publizierten Implementierungsansätze beschränkt sich deren produktive Nutzung bisher vielfach auf Nischen und viele der theoretisch möglichen Anwendungen werden zumindest aktuell nicht in der Breite realisiert [5, 36–42].

Dies liegt im Wesentlichen an dem hohen Bedarf nach Adaptierbarkeit oder Übertragbarkeit der verwendeten Algorithmen: Die erlernten Modelle und Zusammenhänge müssen auch bei leichten Veränderungen des Anwendungsfalls weiter nutzbar und zwischen verwandten Anwendungsfällen austauschbar sein. Dies ist mit konventionellem Deep Learning nicht möglich [5, 40, 41, 43–45].

Konkret stellen sich für die effiziente Nutzung von Deep Learning im Anwendungskontext Industrie daher aus Sicht des Autors die im Folgenden beschriebenen drei Herausforderungen. Sie basieren auf den in Kapitel 3 vorgestellten Literaturrecherchen sowie Gesprächen mit Expert:innen aus Industrie und Wissenschaft. Der Autor hat die darin genannten Teil-Herausforderungen aggregiert und abstrahiert, um eine möglichst breite Problembeschreibung zu gewährleisten – ein Anspruch auf Vollständigkeit besteht jedoch nicht.

Herausforderung 1 (H1): Mangelnde Übertragbarkeit zwischen Anlagen

Industrielle Anlagen zeichnen sich durch starke, oft individuelle Anpassung an den jeweiligen Nutzungskontext aus [46]. Dadurch unterscheiden sich häufig selbst innerhalb desselben Unternehmens ähnliche Prozesse ausführende Anlagen stark, umso mehr jedoch über Unternehmensgrenzen hinweg. Zusammen mit den jeweils unterschiedlichen Umgebungsbedingungen, bspw. hinsichtlich des Klimas am Einsatzort, ergibt sich damit eine sehr heterogene und dynamische Anlagenlandschaft, die eine direkte Übertragung von (Lern-)Ergebnissen von einer Anlage auf eine andere Anlage verhindert [41, 45, 47–52].

Herausforderung 2 (H2): Mangelnde Übertragbarkeit über Prozessgrenzen hinweg

Auch industrielle Prozesse zeichnen sich durch starke, oft individuelle Anpassungen an den jeweiligen Nutzungs- und Anlagenkontext aus [46]. So sorgen schnellere Innovationszyklen, kleinere Los-Größen und ein steigender Bedarf nach individualisierten bzw. personalisierten Produkten für eine erhebliche Verkürzung von Prozess-Lebensdauern [53–56] gepaart mit einem erhöhten Auftreten von Rekonfigurationsmaßnahmen [57–61]. Zusammen mit der großen Vielfalt an Umgebungsbedingungen ergibt sich so nur eine sehr schmale Basis für die Erhebung von Trainingsdatensätzen [41, 62–66].

Herausforderung 3 (H3): Mangelnde Übertragbarkeit zwischen Datensilos

Daten werden im Anwendungskontext Industrie vielfach anwendungsspezifisch erhoben, gespeichert und genutzt [38, 46, 67–70]. Dies begrenzt ihr Nutzungspotential erheblich: Aufgrund der großen Vielfalt von Anwendungsfällen für Deep Learning in der Automatisierungstechnik [34, 71–74] könnten die in einer konkreten Anlagen-Prozess-Konstellation erhobenen Daten in unterschiedlichen Zusammensetzungen zur Lösung ganz unterschiedlicher Probleme eingesetzt werden. Aufgrund der langen Lebensdauer von Industrieanlagen ist zudem zu ihrer Inbetriebnahme nicht absehbar, auf welche Weise die darin erhobenen Daten zukünftig einmal eingesetzt werden könnten [52, 66]. Darüber hinaus fallen Daten unterschiedlicher Art an – vor allem Zeitreihendaten, aber auch Bilder, Videos, Texte oder (statische) Metadaten -, die für die Lösung eines Problems von Bedeutung sein könnten [75] - gängige Deep-Learning-Methoden basieren jedoch üblicherweise auf homogenen Eingangsdaten, sodass die Integration heterogener Datenquellen deren Einsatz erschwert bzw. verhindert [11, 22, 76, 77]. Und schließlich ist das Labeling, also der Prozess Trainingsdatensätze mit Labeln zu versehen, zeitaufwendig und benötigt in vielen Fällen Expertenwissen [78–81], sodass in der industriellen Praxis vielfach darauf verzichtet wird, Datensätze zu labeln [31]. Dies erschwert bzw. verhindert den Einsatz überwachter Lernmethoden trotz deren guter Eignung für die im Anwendungskontext Industrie gängigen Klassifikations- und Regressionsprobleme [28, 29, 82]. Zusammengenommen ergibt sich so eine zersplitterte, vielfach auf einen einzelnen spezifischen Anwendungsfall und -kontext zugeschnittene Datenlandschaft.

Die aus diesen drei Herausforderungen resultierenden, wenig umfang- und variantenreichen Trainingsdatensätze erlauben lediglich eng auf den jeweiligen, schmalen Datenerhebungs-Kontext zugeschnittene und damit **wenig generalisierte Lernergebnisse** [4, 10, 41, 70, 82–84]. Schon kleinere Änderungen dieser Kontexte führt somit zu einer deutlichen Verschlechterung der Anwendbarkeit der Lernalgorithmen [41, 65, 70, 85, 86]. Aufgrund der dynamischen Natur industrieller Prozesse und Anlagen, bspw. in Folge von Verschleiß, wechselnden Rohstoffeigenschaften oder Umgebungsbedingungen, sind derartige Veränderungen der Datenerhebungs-Kontexte jedoch die Regel [65, 70, 86–88]. Dies erfordert ein stetes Neu-Trainieren der Lernalgorithmen auf immer wieder neu erhobenen, möglichst umfangreichen Trainingsdatensätzen, um dauerhaft eine gute Ergebnisqualität zu sichern. Aus ökonomischen wie ökologischen Gründen sind ständige Anpassungen und insbesondere Partikularlösungen allerdings selten praktikabel [4, 70, 89–92].

Erforderlich sind daher Ansätze, derartige **Lernalgorithmen robuster und adaptierbarer** zu machen, sodass Veränderungen des Datenerhebungs-Kontexts keine Anpassungen des Algorithmus erforderlich machen bzw. dass diese Veränderungen möglichst wenig Aufwand verursachen.

1.3 Methodik

Mithilfe der Methodik der wissenschaftliche Entwurfsforschung (engl.: **Design Science Research**, kurz: DSR) nach Hevner [93, 94] sollen die in Kapitel 1.2 genannten Herausforderungen bei der Entwicklung von Lernalgorithmen für industrielle Automatisierungssysteme überwunden werden. Aufgrund ihres Fokus auf Artefakt-basierte Erforschung konkreter ingenieurwissenschaftlicher Problemstellungen zur Ableitung allgemeinen, übertragbaren und fundierten Gestaltungswissens [95–98] stellt DSR einen optimalen Rahmen zur Lösung der Forschungsfrage dar.

In enger Anlehnung an die Vorgehensweise nach [95] wird der (Lösungs-)Entwurfszyklus (engl.: Design Cycle) in drei Phasen unterteilt:

In der **Untersuchung der Problemstellung** (engl.: Problem Investigation) werden der Anwendungskontext festgelegt, Herausforderungen beschrieben und daraus Anforderungen an das Lösungsartefakt, also das zur Problemlösung entwickelte Objekt, abgeleitet. An der daraus resultierenden Zielsetzung muss sich jeder anschließend entwickelte Lösungsansatz[3] messen lassen.

Im **Lösungsansatzentwurf** (engl.: Treatment Design) werden einerseits vorhandene (Teil-)Ansätze für Lösungsartefakte recherchiert und andererseits basierend darauf (ein Konzept für) das konkret zu untersuchende Lösungsartefakt entworfen.

In der **Lösungsansatzvalidierung** (engl.: Treatment Validation) wird dieses Lösungsartefakt(-konzept) dann hinsichtlich seiner Eignung zur Problemlösung untersucht.

Wie bereits durch die Begrifflichkeit des „Entwurfszyklus“ angedeutet, handelt es sich hierbei um ein iteratives Vorgehen mit Schnittmengen zum agilen Vorgehensmodell [99]. Dies bietet die Möglichkeit der ständigen Überprüfung des Lösungsansatzes und der ggf. nötigen Nachschärfung von Problemstellung und Zielsetzung. Aus Gründen der Stringenz wird in dieser Arbeit auf eine Abbildung aller Iterationszyklen verzichtet. Stattdessen werden lediglich die beiden Iterationen

- Entwurf des abstrakten Artefaktkonzepts und dessen analytische Validierung anhand der formulierten Anforderungen sowie
- Entwurf und prototypische Realisierung mehrerer Artefakte und deren empirische Validierung im konkreten Problemkontext

ausführlich dargestellt.

---

[3] Nach [95] ist der Begriff „Treatment“ dem Begriff „Solution“ vorzuziehen, da er dessen erst aus der iterativen Interaktion zwischen Artefakt und Problemstellung resultierenden Charakter besser verdeutliche. Der Autor entscheidet sich aus Gründen der besseren Verständlichkeit gegen eine wörtliche Übersetzung und verwendet stattdessen den Begriff „Lösungsansatz“, der die Vorläufigkeit und ggf. Unzulänglichkeit des Ansatzes ebenfalls widerspiegelt.

Aufgrund des Forschungscharakters dieser Arbeit werden die in [95] beschriebenen, höheren Technologie-Reifegraden (engl.: Technology Readiness Levels) vorbehaltenen Anwendungstransfer-Phasen der Lösungsansatzimplementierung (engl.: Treatment Implementation) und -evaluierung (engl.: Treatment Evaluation) in der breiten Serienanwendung nicht durchgeführt.

1.4 Anforderungen und Zielsetzung

Die in Kapitel 1.2 genannten Herausforderungen bei der Entwicklung von Lernalgorithmen für industrielle Automatisierungssysteme bilden den Ausgangspunkt dieser Arbeit und die Grundlage der DSR-Phase „Untersuchung der Problemstellung“ (siehe Kapitel 1.3). Die nachfolgenden Anforderungen basieren zudem auf den in Kapitel 3 vorgestellten Literaturrecherchen und formulieren konkrete Erwartungen an mögliche Lösungsansätze. Sie dienen somit deren Validierung im Sinne des DSR.

Anforderung 1 (A1): Robustheit, um die Notwendigkeit des Nachtrainierens zu reduzieren

Trotz vielfach kleiner Trainingsdatensätze und dynamischer (Teil-)Probleme (siehe H1 bis H3) ist es wünschenswert, mittels Maßnahmen zur Steigerung der Robustheit des Lernalgorithmus dessen fortwährende Nutzung ohne Nachtrainieren zu erlauben. Dies kann bspw. über eine breitere Trainingsbasis erreicht werden. Derartige Maßnahmen können den Aufwand des Einsatzes derartiger Lernalgorithmen erheblich senken – ihre Anwendbarkeit ist allerdings stark vom jeweiligen Anwendungsfall abhängig.

Anforderung 2 (A2): Adaptierbarkeit, um den Aufwand des Nachtrainierens zu reduzieren

Wo sich aufgrund vielfach kleiner Trainingsdatensätze und dynamischer (Teil-)Probleme (siehe H1 bis H3) das stetige Nachtrainieren von Lernalgorithmen nicht vermeiden lässt, ist es wünschenswert, den damit verbundenen Aufwand zu reduzieren. Dieser Aufwand umfasst verschiedene Aspekte, bspw. die Erhebung neuer Trainingsdatensätze, das Anpassen der Deep-Learning-Architekturen und -Hyperparameter oder schließlich das eigentliche Training. Potential für eine Reduzierung des Aufwands bietet auch jegliche Reduzierung des Einsatzes gelabelter Daten. Abhängig vom jeweiligen Anwendungsfall sind diese Aspekte unterschiedlich bedeutsam – Strategien zur Reduzierung des Aufwands sollten dem Rechnung tragen.

Anforderung 3 (A3): Wiederverwendbarkeit von Daten zur Anpassung an neue Anwendungskontexte

Da sich industrielle Anwendungsfälle vielfach stark unterscheiden, sich der gewünschte Leistungsumfang von darin eingesetzten Lernalgorithmen a priori schwer abschätzen lässt und die Neu-Erhebung von Trainingsdaten sehr aufwendig ist (siehe H1 bis H3), ist es wünschenswert,

bereits erhobene Daten unterschiedlicher Qualität, Dimensionalität und Herkunft möglichst einfach wiederverwenden zu können. Dies umfasst auch die gleichzeitige Behandlung unterschiedlicher Datenarten innerhalb desselber Lernalgorithmen.

Anforderung 4 (A4): Wiederverwendbarkeit von Code zur Anpassung an neue Anwendungskontexte

Da sich industrielle Anwendungsfälle vielfach stark unterscheiden und sich der gewünschte Leistungsumfang von darin eingesetzten Lernalgorithmen a priori schwer abschätzen lässt (siehe H1 bis H3), ist es wünschenswert, diese Lernalgorithmen so zu entwerfen, dass sie unter möglichst weitgehender Wiederverwendung von Code einfach auf unterschiedliche Problemtypen und Anwendungsfälle angepasst werden können. Dies kann bspw. durch einen modularen Aufbau der Lernalgorithmen erreicht werden.

Es ist festzuhalten, dass diese Anforderungen an eine Architektur für Lernalgorithmen, nicht an einen spezifischen Algorithmus gestellt werden. Sie müssen also lediglich in Abhängigkeit von konkreten Anwendungsfällen prinzipiell erfüllbar sein und nicht in jeder Realisierung der Architektur auch tatsächlich erfüllt werden. Aus den genannten Anforderungen ergibt sich daher in Verbindung mit den in Kapitel 1.2 genannten Herausforderungen die folgende **Zielsetzung für diese Arbeit**:

Entwicklung einer Architektur für übertragbare Lernalgorithmen, die auch auf kleinen Datensätzen und trotz der hohen Dynamik von Automatisierungssystemen robust industrietypische Probleme lösen können

1.5 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich wie in Abbildung 1.1 dargestellt in 6 Kapitel:

In Kapitel 2 werden als Erweiterung der DSR-Phase „Untersuchung der Problemstellung“ die für diese Arbeit relevanten Grundlagen eingeführt. Diese unterteilen sich in eine Betrachtung verschiedener Deep-Learning-Methoden, verschiedener Ansätze zum (Teil-)Problem-übergreifenden Lernen sowie des Clusterings. Abschließend werden für die gesamte Arbeit gültige Charakteristika des Anwendungskontext Industrie abgeleitet.

Als Teil der ersten Iteration der DSR-Phase „Lösungsansatzentwicklung“ gibt Kapitel 3 für die zuvor genannten Themenfelder einen Überblick über den Stand von Wissenschaft und Technik, um daraus den Forschungsbedarf abzuleiten und bereits bestehende Lösungsansätze zu identifizieren. Basierend darauf wird eine konkrete Forschungsfrage formuliert.

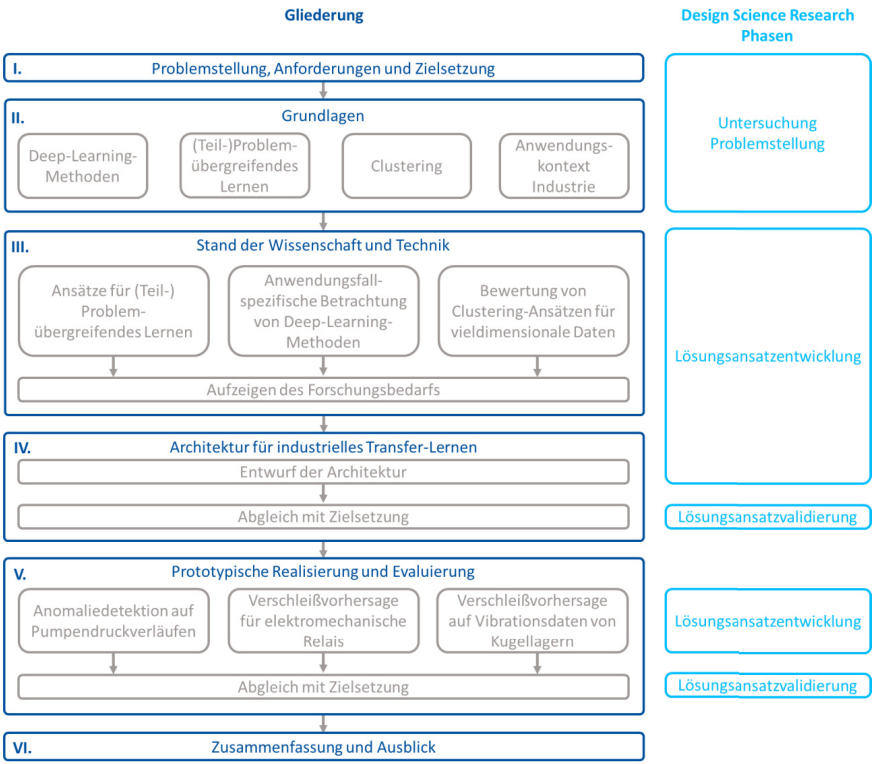

Abbildung 1.1: Aufbau der Arbeit

Der Anfang von Kapitel 4 bildet den zweiten Teil der ersten Iteration der DSR-Phase „Lösungsansatzentwicklung“ und beschreibt die in dieser Arbeit entwickelte Architektur für industrielles Transfer-Lernen. Besonderes Augenmerk wird dabei auf die einzelne Architektur-Komponenten sowie deren Funktionsweise gelegt. Der letzte Abschnitt von Kapitel 4 stellt die erste Iteration der DSR-Phase „Lösungsansatzvalidierung“ in Form eines Abgleichs zwischen dem entwickelten Konzept mit den in Kapitel 1.4 definierten Anforderungen dar.

Kapitel 5 widmet sich im Rahmen der zweiten Iteration der DSR-Phasen „Lösungsansatzentwicklung“ und „Lösungsansatzvalidierung“ der prototypischen Realisierung und Evaluierung der Architektur industriellen Transfer-Lernens anhand dreier, konkreter Evaluierungsfälle: einer Anomaliedetektion auf Pumpendruckverläufen, einer Ausfallvorhersage für elektromechanische Relais sowie einer Ausfallvorhersage auf Vibrationsdaten von Kugellagern. Der erste Evaluierungsfall erfüllt dabei den Zweck einer Referenz-Implementierung, wohingegen der zweite die Übertragbarkeit dieser Referenz-Implementierung auf einen anderen Problemtyp und Anwendungsfall

demonstrieren soll. Der dritte Evaluierungsfall zeigt schließlich an wieder einem anderen Anwendungsfall einen anderen Realisierungsansatz innerhalb der vorgestellten Architektur. Das Kapitel endet mit einem Abgleich des Lösungsartefakts mit den in Kapitel 1.4 definierten Anforderungen hinsichtlich seiner Eignung zur Beantwortung der Forschungsfrage.

Kapitel 6 fasst abschließend die Ergebnisse dieser Arbeit zusammen und bietet einen Ausblick.

2 Grundlagen

In diesem Kapitel werden im Rahmen der Design-Science-Research-Phase „Untersuchung der Problemstellung“ die theoretischen Grundlagen der in dieser Arbeit relevanten Konzepte und Methoden aus den Bereichen des Deep Learnings, des (Teil-)Problem-übergreifenden Deep Learnings sowie des Clusterings eingeführt. Aufgrund deren Breite kann dies jeweils nur in stark verkürzter Form erfolgen, sodass für eine tiefergehende Auseinandersetzung auf entsprechende Fachliteratur verwiesen werden muss. Abschließend werden die Charakteristika des Anwendungskontexts Industrie betrachtet, aus denen für die gesamte Arbeit gültige Eingrenzungen der Aufgabenstellung sowie Abgrenzungen zu verwandten Themen hervorgehen.

2.1 Deep-Learning-Methoden

Maschinelles Lernen ist die Fähigkeit von Computersystemen, aus Daten Zusammenhänge zu extrahieren, indem sie „darauf trainiert werden“. Derartige Systeme können sich somit durch Erfahrung in Form von Trainingsdatensätzen verbessern und mit der Zeit ein Modell generieren, das dazu verwendet werden kann, die Lösungen von Problemen auf der Grundlage des zuvor Gelernten vorherzusagen [3].

Im Allgemeinen werden drei verschiedene Kategorien maschinellen Lernens unterschieden:

- **Überwachtes Lernen** bezeichnet das Lernen mittels gelabelter Trainingsdatensätze. Die „Überwachung“ besteht darin, dass dem Lernalgorithmus im Training also bekannt wird, was das korrekte Ergebnis gewesen wäre – sodass er daraufhin ggf. sein „Verhalten“ bzw. seine Parametrierung anpassen kann. Typische Aufgaben sind **Klassifikation**, d.h. die Verortung von Proben in einem diskreten Wertebereich, und **Regression**, d.h. die Verortung von Proben in einem kontinuierlichen Wertebereich [3, 83].

- **Unüberwachtes Lernen** bezeichnet das Lernen mittels ungelabelter Trainingsdatensätze. Dem Lernalgorithmus kann also im Training nicht bekannt werden, was das korrekte Ergebnis gewesen wäre – er kann somit auch keinen Anpassungsbedarf daraus ableiten. Stattdessen sucht er nach Mustern und Regelmäßigkeiten in den Trainingsdaten, bspw. zur Lösung von **Clustering**-Aufgaben [3, 83].

- **Bestärkendes Lernen** (engl.: Reinforcement Learning) bezeichnet das Lernen aus den aus dem Lernergebnis resultierenden Konsequenzen in Form von (ggf. indirekten) „Belohnungen“. Dazu wird statt Trainingsdatensätzen auf Lernumgebungen zurückgegriffen, die eine direkte Erprobung des Lernergebnisses ermöglichen. Der Lernalgorithmus, üblicherweise ein Software-Agent [28], strebt also danach, sein Verhalten auf Basis dieser

indirekten Rückmeldung zu optimieren [82, 83]. Bestärkendes Lernen wird im weiteren Verlauf dieser Arbeit keine Rolle spielen.

Deep Learning (dt.: Mehrschichtiges Lernen) bezeichnet maschinelles Lernen mittels mehrschichtiger, künstlicher neuronaler Netze. Deren Grundelement, das künstliche Neuron, orientiert sich an seinem biologischen Pendant [100]. Abbildung 2.1 zeigt den Aufbau eines solchen künstlichen Neurons.

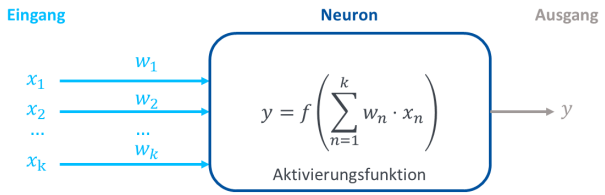

Abbildung 2.1: Aufbau eines künstlichen Neurons

In seiner Grundform kann ein künstliches Neuron mehrere, mit sogenannten **Kantengewichten** $\boldsymbol{w}$ gewichtete **Eingangswerte** $\boldsymbol{x}$ gleichzeitig verarbeiten. Dazu summiert es die gewichteten Eingangswerte auf und führt sie einer **Aktivierungsfunktion** zu. Im Rahmen dieser Arbeit relevante Aktivierungsfunktionen sind die Rectified Linear Unit (dt.: Gleichrichtende Lineareinheit, kurz: ReLU) [101] und die Scaled Exponential Linear Unit (dt.: Skaliert-exponentielle Lineareinheit, kurz: SELU) [102]. Der daraus resultierende Wert wird anschließend als **Ausgangswert** $\boldsymbol{y}$ ausgegeben.

Über die Verknüpfung mehrerer künstlicher Neuronen in zwei oder mehr Schichten (zuzüglich der Eingangsschicht) können beliebig komplexe Verknüpfungen von Eingangs- und Ausgangswerten realisiert werden. Im Rahmen des Trainings können diese Verknüpfungen mittels Anpassung der einzelnen Kantengewichte an den jeweils vorliegenden Trainingsdatensatz automatisch erstellt werden. Die Schichten zwischen Ein- und Ausgangsschicht nennt man versteckte Schichten (engl.: Hidden Layers), da sie von außen nicht direkt zugänglich sind.

Es existiert eine Vielzahl unterschiedlicher Deep-Learning-Methoden, sodass im Folgenden nur auf eine kleine, für diese Arbeit relevante Untermenge eingegangen werden kann: vorwärtsgerichtete Netzwerke in Form von Fully Connected Neural Networks und Convolutional Neural Networks, rekurrente Netzwerke in Form von Long Short-Term Memory und Autoencoder. Für tiefergehende Informationen sei auf [3, 10, 28, 29, 82, 83, 100] verwiesen.

2.1.1 Vorwärtsgerichtete Neuronale Netzwerke

Vorwärtsgerichtete neuronale Netzwerke (engl.: Feed-forward Neural Networks) sind die einfachsten Netzarchitekturen des Deep Learning. In ihnen sind die Neuronen einer Schicht immer nur mit denen der darauffolgenden Schicht verbunden.

2.1.1.1 Fully Connected Neural Networks

Fully Connected Neural Networks (FCNN, dt.: Vollvermaschte neuronale Netzwerke) oder mehrschichtige Perzeptren (engl.: Multi-layer Perceptrons) sind die häufigste Form vorwärtsgerichteter neuronaler Netzwerke [10]. In ihnen sind alle Neuronen einer Schicht mit allen Neuronen der nachfolgenden Schicht verbunden. FCNN sind, ausreichend Neuronen vorausgesetzt, universelle Funktionsapproximatoren und können daher zu unterschiedlichsten Zwecken eingesetzt werden. Ihr Nachteil ist dabei der mit zunehmender Neuronen- und Schichtenzahl stark ansteigende Rechenaufwand [10, 82].

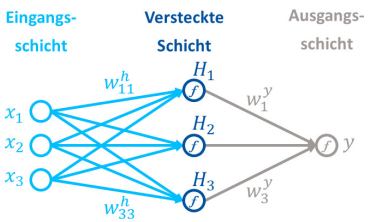

Abbildung 2.2: Aufbau eines FCNN

Abbildung 2.2 zeigt beispielhaft den Aufbau eines zweischichtigen FCNN mit drei Knoten in der versteckten Schicht und einem Ausgangsknoten. Die Eingangsschicht wird üblicherweise nicht mitgezählt. Exemplarisch werden die äußeren Kantengewichte benannt, f steht für die Aktivierungsfunktion eines Knotens.

2.1.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN, dt.: Neuronale Faltungsnetzwerke) basieren auf [103] und stellen eine Sonderform der vorwärtsgerichteten neuronalen Netzwerke dar. Anders als diese kommen sie für Daten in Gitterstrukturen (bspw. Zeitreihen als eindimensionalen Datenpunkt-„Gitter“ oder Bilder als zweidimensionalen Pixel-Gitter) mit einer um Größenordnungen verringerten Anzahl Kantengewichten aus, sodass sie bei weiterhin guten Ergebnissen auf handelsüblicher Hardware trainierbar bleiben [29, 83]. Diese Eigenschaft führt dazu, dass viele der in Kapitel 3.2 untersuchten Ansätze CNN bspw. zur Merkmalsextraktion einsetzen und auch die in Kapitel 5 vorgestellten protoytpischen Implementierungen im Rahmen dieser Arbeit auf diese

zurückgreifen. Im Folgenden werden daher die in CNN gängigerweise zum Einsatz kommenden, unterschiedlichen Schichten und Parameter eingeführt:

Faltungsschichten dienen der Herausarbeitung relevanter Merkmale durch Faltung der umliegenden Werte auf einen Fokuspunkt. Dies geschieht mittels element-weiser Multiplikation einer Faltungsmatrix, des sogenannten **Kernel**s, mit einem Ausschnitt der Eingangsmatrix. Die Elemente des Kernels sind dabei die im Trainingsprozess zu optimierenden Kantengewichte. Die Wahl der Aktivierungsfunktion bestimmt den weiteren Umgang mit dem Ergebnis der Multiplikation. Der Kernel wird schrittweise über die Eingangsmatrix hinwegbewegt, bis die gesamte Eingangsmatrix abgerastert ist. Die Schrittweite wird als **Stride** bezeichnet und ist ein vorher festzulegender Architektur-Parameter. Faltungsschichten können die Dimensionsgröße der Eingangsmatrix reduzieren, müssen dies aber nicht. Für Letzteres wird sogenanntes **Padding** genutzt, das die Eingangsmatrix bspw. mit Nullen umgibt und es dem Kernel so erlaubt, auch die Elemente am Rand der Eingangsmatrix als Fokuspunkt zu nutzen [29, 83].

Auch **Pooling-Schichten** dienen dem Herausarbeiten relevanter Merkmale, jedoch werden dabei alle Elemente innerhalb des jeweiligen Matrix-Ausschnitts gleichbehandelt. Am häufigsten ist das sogenannte Max-Pooling, dass den Maximalwert des Matrix-Ausschnitts weitergibt. Pooling-Schichten führen daher ebenfalls zu einer Reduzierung der Dimensionsgröße der Matrix [29, 83].

Üblicherweise stehen am Ende eines CNN eine FCNN-Schicht (hier auch **Flattening-Schicht** genannt), die die Ausgangsmatrix in einen Vektor überführt und ggf. auch direkt eine Klassifikation oder Regression durchführt.

CNN sind somit in der Lage, potenziell interessante Merkmale unabhängig von deren Position in der Eingangsmatrix mit denselben Gewichten zu lernen. Dabei reagieren sie auch unempfindlich auf verschiedene Formen der geometrischen Transformation [29]. Sie eignen sich damit zur Kompression der Eingangsdaten, da diese nach Verarbeitung durch einige Faltungsschichten als verdichtete Merkmalssätze angesehen und extrahiert für andere Aufgaben verwendet werden können (**Merkmalsextraktion**).

2.1.2 Rekurrente Neuronale Netzwerke

Rekurrente neuronale Netzwerke (RNN, engl.: Recurrent Neural Networks) weisen anders als vorwärtsgerichtete neuronale Netzwerke mindestens einzelne Verbindungen auf, die nicht von einem Neuron einer Schicht zu einem Neuron der darauffolgenden Schicht gehen. Derartige **Rückkopplungen** können auf drei Arten erfolgen: Direkte Rückkopplungen führen vom Ausgang eines Neurons zum Eingang desselben Neurons im nachfolgenden Zeitschritt, laterale Rückkopplungen führen vom Ausgang eines Neurons zum Eingang eines Neurons derselben Schicht und indirekte Rückkopplungen vom Ausgang eines Neurons zu einem Neuron einer vorangegangenen

Schicht. Rückkopplungen müssen dabei nicht unmittelbar ausgeführt, sondern können zeitlich verzögert werden [29].

Durch die Möglichkeit rekursiver Rückführungen kann ein Zusammenhang zwischen dem aktuellen Ausgangswert und vergangenen Eingangswerten hergestellt werden. Dies schafft einen zeitlichen oder räumlichen Kontext, der RNN einen Vorteil im Umgang mit sequenziellen Daten verschafft, indem er eine sequenzielle Verarbeitung ermöglicht. RNN werden daher bspw. für Aufgaben der Spracherkennung und -generierung, maschinelle Übersetzung oder Anomaliedetektion auf Zeitreihendaten eingesetzt und kommen somit in einigen der in Kapitel 3.2 untersuchten Ansätze zum Einsatz. Abhängig von der genutzten Aktivierungsfunktion leiden einfache RNN jedoch unter dem Vanishing oder Exploding Gradient Problem (dt.: Problem gegen Null bzw. Unendlich tendierender Gradienten), die durch wiederholtes Aufmultiplizieren sehr kleiner bzw. sehr großer Werte im Rahmen des Trainings mittels Backpropagation through Time (dt.: Fehlerrückführung durch die Zeit) entstehen [29].

Long Short-Term Memory (LSTM, dt.: Langes Kurzzeitgedächtnis) [104] sind die gängigste Form RNN und leiden nicht unter dem Vanishing Gradient Problem. Statt einer konventionellen Aktivierungsfunktion nutzen sie sogenannte Gates (dt.: Tore), die mittels eigener Berechnungsvorschriften in Verbindung mit einem Neuronen-internen Zustandswert Informationen entweder unverändert oder gar nicht passieren lassen [29, 100, 105]. Dies führt dazu, dass viele der in Kapitel 3.2 untersuchten Ansätze LSTM bspw. zum Fortschreiben von Zeitreihen einsetzen und auch die in Kapitel 5 vorgestellten protoytpischen Implementierungen im Rahmen dieser Arbeit auf diese zurückgreifen. Abbildung 2.3 veranschaulicht daher den inneren Aufbau eines LSTM-Neurons inklusive der drei Gates („Vergessen“, „Hinzufügen“, „Ausgeben“, in der Abbildung jeweils gestrichelt eingerahmt) unter besonderer Berücksichtigung der darin genutzten (üblicherweise direkten) Rückkopplungen.

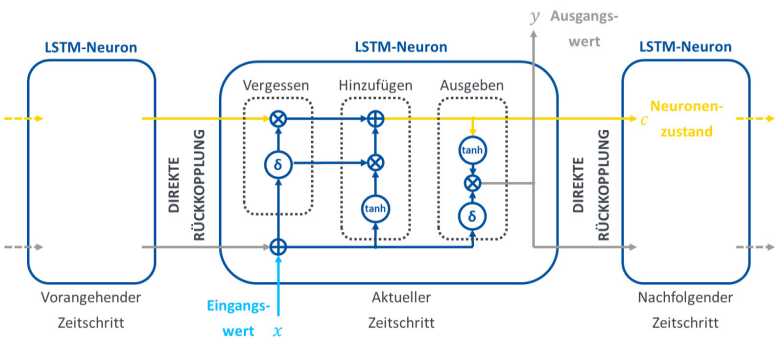

Abbildung 2.3: Aufbau eines LSTM-Neurons inklusive seiner Rückkopplungsverbindungen

2.1.3 Autoencoder

Autoencoder sind neuronale Netzwerke, die versuchen mit ihrem Ausgang ihren Eingang zu reproduzieren [106]. In der bzw. den Zwischenschicht(-en) befinden sich jedoch weniger Neuronen als in der Ein- oder Ausgangsschicht, sodass die Informationen zwischenzeitlich mehr oder weniger stark komprimiert werden. Da ein funktionaler Autoencoder auch aus diesen komprimierten Informationen die Eingangsdaten rekonstruieren kann, können diese als verdichtete Merkmalssätze angesehen und extrahiert für andere Aufgaben verwendet werden (**Merkmalsextraktion**). Darüber hinaus sollten robust trainierte Autoencoder unempfindlich gegenüber verrauschten Eingangssignalen sein, sodass sie zur nichtlinearen **Hauptkomponentenanalyse** (engl.: Principal Component Analysis) eingesetzt werden können. Das Training von Autoencodern ist auch auf ungelabelten Trainingsdatensätzen, d.h. unüberwacht, möglich, da die Abweichung zwischen Ein- und Ausgang, d.h. der Reproduktionsfehler, als zu reduzierende Größe genutzt werden kann und Label damit nicht zur Verlustbestimmung erforderlich sind [29, 100]. Dies führt dazu, dass viele der in Kapitel 3.2 untersuchten Ansätze Autoencoder bspw. zur Merkmalsextraktion einsetzen und auch die in Kapitel 5 vorgestellten protoytpischen Implementierungen im Rahmen dieser Arbeit auf diese zurückgreifen. Abbildung 2.4 veranschaulich daher die Komponenten und Funktionsweise eines Autoencoders mit drei Zwischenschichten (inklusive der Codeschicht).

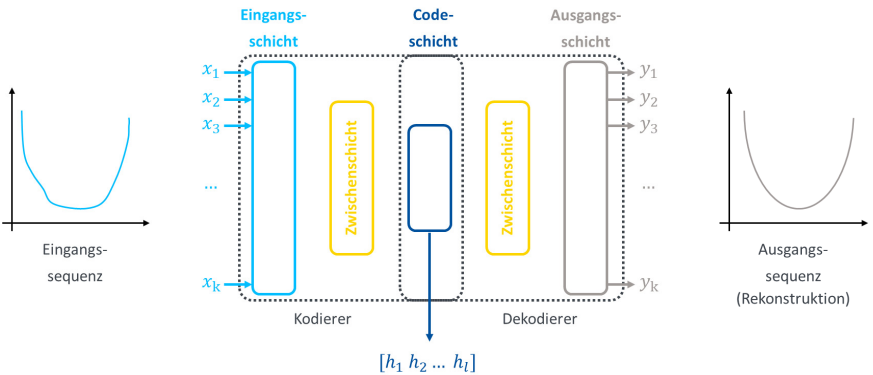

Abbildung 2.4: Funktionsweise eines Autoencoders

2.2 (Teil-)Problem-übergreifendes Deep Learning

Die in Kapitel 2.1 vorgestellten Deep-Learning-Methoden werden auf Trainingsdatensätzen trainiert und können anschließend für bestimmte Eingangswerte passende Ausgangswerte zurückgeben. Training bezeichnet dabei den Prozess, der über eine Anpassung Netzwerk-interner Parameter den jeweils aus den Trainingsdatensätzen resultierenden Zusammenhang zwischen Ein- und

Ausgang herstellt. Ein trainiertes neuronales Netzwerk kann somit als abstraktes Modell und dessen Netzwerk-interne Parameter als abstrakte, d.h. nicht Menschen-verständliche, **Repräsentationen von Wissen** über den jeweiligen Zusammenhang angesehen werden [28, 100, 107, 108].

Trainierte neuronale Netzwerke sind keine allgemein-gültigen Modelle, sondern üblicherweise auf einen engen Problem-Kontext festgelegt. Dieser Problem-Kontext wird durch die im Trainingsdatensatz enthaltenen Proben spezifiziert. Ändert sich das Problem, bspw. hinsichtlich der Häufigkeitsverteilung schon vorhandener Ereignisse oder des Hinzukommens bzw. Wegfallens von Ereignissen, so muss davon ausgegangen werden, dass das trainierte neuronale Netzwerk nicht mehr in der Lage ist, die nun vorliegenden Zusammenhänge korrekt zu berücksichtigen. Es müsste auf Basis eines neu zusammengestellten Trainingsdatensatzes erneut trainiert werden [4, 5, 109–111] (siehe Abbildung 2.5).

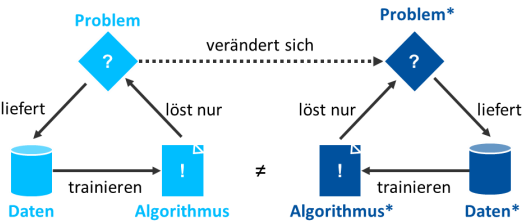

Abbildung 2.5: Deep Learning ohne Wissenstransfer auf veränderlichen Problemen (nach [5])

Derartige Trainings sowie die vorangehende Beschaffung ausreichender Trainingsdatensätze sind Ressourcen-aufwändig, sodass eine Reduzierung des Trainings-Aufwands bzw. der dafür benötigten Daten ökonomisch und ökologisch sinnvoll ist [4, 5, 89–91, 111]. Ein Ansatz dazu ist (Teil-)Problem-übergreifendes Deep Learning, das das bereits vorhandene, im Netzwerk hinterlegte Wissen für das neue bzw. veränderte (Teil-)Problem nutzbar macht. Dies kann je nach Realisierungsstrategie als **Wissenstransfer** angesehen werden und ist durch verschiedene Anpassungen oder Erweiterungen der in Kapitel 2.1 vorgestellten Deep-Learning-Methoden realisierbar.

Zu beachten ist, dass die Anwendung (Teil-)Problem-übergreifenden Deep Learnings wiederum eigene Herausforderungen mit sich bringt: Abhängig vom gewählten Ansatz und Transfer-Szenario, d.h. der Art und Reihenfolge der (Teil-)Probleme sowie der für den Transfer zur Verfügung stehenden Datensätze, kann sich die Problemlösefähigkeit der neuronalen Netze nicht nur verbessern, sondern auch verschlechtern (siehe Kapitel 2.2.1.3). Um eine solche Verschlechterung zu vermeiden bzw. tatsächliche Verbesserungen zu erzielen, müssen bspw. Ansatz und Transfer-Szenario zueinander passen, die Initialisierungen der neuronalen Netze angemessen vorgenommen werden oder zu transferierende Proben aus den zur Verfügung stehenden Datensätzen ausgewählt werden [112].

Es existiert eine Vielzahl unterschiedlicher Konzepte und Methoden im Bereich des (Teil-)Problem-übergreifenden Deep Learning, sodass im Folgenden nur auf eine im Sinne der in Kapitel 1.4 beschriebenen Zielsetzung dieser Arbeit relevante Untermenge näher eingegangen werden kann: Transfer-Lernen und kontinuierliches Lernen. Auf weitere, in diesem Kontext häufiger genannte Begriffe geht das abschließende Unterkapitel „Weitere Konzepte“ gesammelt ein.

2.2.1 Transfer-Lernen

Das Gebiet des Transfer-Lernens untersucht und entwickelt Methoden des maschinellen Lernens, die Wissen aus einem oder mehreren zuvor gelösten Quellproblemen nutzen, um ein neues, bisher unbekanntes Zielprobleme effizienter zu lösen. Im Folgenden soll dieser weite Fokus auf Ansätze aus dem Bereich des Deep Learning eingeschränkt werden. Abbildung 2.6 veranschaulicht den dabei zugrunde liegenden Wirk-Mechanismus.

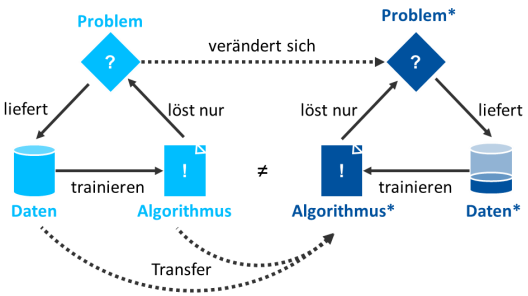

Abbildung 2.6: Transfer-Lernen auf veränderlichen Problemen (nach [5])

Transfer-Lernen wird allgemein auf zwei Arten kategorisiert: Es kann anhand der gelösten Transfer-Probleme oder anhand der dafür genutzten Transfer-Ansätze strukturiert werden [113–115].

2.2.1.1 Problemkategorisierung

Die Problemkategorisierung betrachtet dabei konkret entweder die Verfügbarkeit von gelabelten Daten zu oder die Ähnlichkeit der (Eingangs-)Merkmalsräume von Quell- und Zielproblemen [116].

Hinsichtlich der Verfügbarkeit von gelabelten Daten werden üblicherweise drei verschiedene Kategorien unterschieden:

- **Induktives Transfer-Lernen** adressiert Transferprobleme, in denen gelabelte Trainingsdatensätze des Zielproblems verfügbar sind [113–115]. Abhängig von der zusätzlichen Verfügbarkeit von gelabelten Trainingsdatensätzen des bzw. der Quellproblem(-e) hat

diese Kategorie Ähnlichkeiten mit Multi-Tasking (wenn diese verfügbar sind) [117, 118] oder autodidaktischem Lernen (wenn diese nicht verfügbar sind) [119].

- **Transduktives Transfer-Lernen** adressiert Transferprobleme, in denen nur gelabelte Trainingsdatensätze des bzw. der Quellproblem(-e) verfügbar sind [113–115].
- **Unüberwachtes Transfer-Lernen** adressiert Transferprobleme, in denen gar keine gelabelten Trainingsdatensätze verfügbar sind [113–115].

Hinsichtlich der Ähnlichkeit der (Eingangs-)Merkmalsräume werden üblicherweise zwei Kategorien unterschieden:

- **Homogenes Transfer-Lernen** adressiert Transferprobleme, in denen Quell- und Zielmerkmalsräume identisch sind [113, 114].
- **Heterogenes Transfer-Lernen** adressiert Transferprobleme, in denen sich Quell- und Zielmerkmalsräume unterscheiden [113, 114].

In der Literatur werden die gleichen Kategorien vereinzelt auch auf die Ähnlichkeit von (Ausgangs-)Labelräumen angewandt (siehe z. B. [114, 120]). In dieser Arbeit wird jedoch wie in [116] auf eine solche Unterscheidung verzichtet, da vom Autor keine eindeutigen terminologischen Tendenzen erkannt werden konnten. Obwohl die zuvor beschriebene Terminologie weit verbreitet ist, gibt es in dem noch sehr neuen Forschungsfeld des Transfer-Lernens diesbezüglich weiterhin Uneinheitlichkeiten [116]. In dieser Arbeit werden daher ausschließlich die oben genannten Definitionen verwendet. Eine ausführlichere Diskussion zur Terminologie des Transfer-Lernens findet sich in [114].

2.2.1.2 Lösungskategorisierung

Die Lösungskategorisierung zielt darauf ab, anhand der verwendeten Ansätze verschiedene Kategorien des Transfer-Lernens zu definieren. Sie lassen sich in vier Hauptlösungskategorien einteilen:

Instanztransfer (siehe Abbildung 2.7, Fall I) umfasst Ansätze, die (gewichtete) Proben aus dem bzw. den Quellproblem(-en) zum Trainingsdatensatz des Zielproblems hinzufügen, um das Erlernen des Zielproblem-Lösungszusammenhangs zu verbessern [113–115]. Dadurch wird die Menge der für das Training verfügbaren Daten erhöht, ohne den Algorithmus selbst wesentlich zu verändern [121].

Merkmalsrepräsentationstransfer (siehe Abbildung 2.7, Fall II) umfasst Ansätze, die Proben aus Quell- und Zielproblemen in einen gemeinsamen Merkmalsraum abbilden, um das Erlernen des Zielproblem-Lösungszusammenhangs zu verbessern [113–115]. Dabei kann sowohl der Quell- als auch der Ziel- oder sogar ein komplett neuer Merkmalsraum genutzt werden (abgebildet

ist die Nutzung des Quellmerkmalsraums). Beim mehrschichtigen Transfer-Lernen kann dies durch Domänen-Adaption erreicht werden [120, 121].

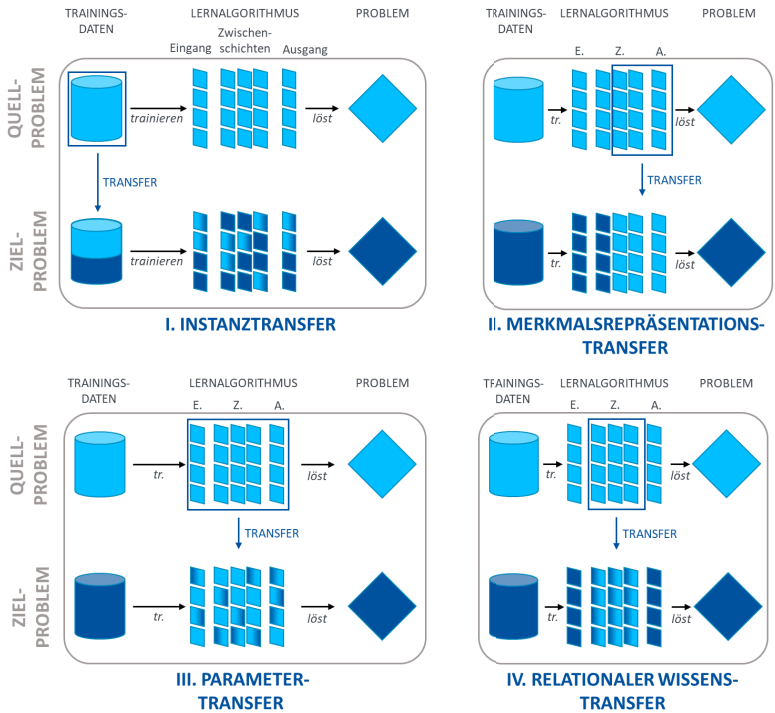

Abbildung 2.7: Schematische Darstellung der Lösungskategorien des mehrschichtigen Transfer-Lernens

Parametertransfer (siehe Abbildung 2.7, Fall III) umfasst Ansätze, die Parameter oder Initialisierungen vom auf dem bzw. den Quellproblem(-en) trainierten Lernalgorithmus an den Zielproblem-Lernalgorithmus weitergeben, um dessen Initialisierung zu verbessern, bevor das eigentliche Training auf dem Zielproblem beginnt [113–115]. Dadurch wird der Algorithmus selbst verändert, wodurch sein Bedarf an Trainingsdaten sinkt. Für mehrschichtiges Transfer-Lernen entspricht dies einer teilweisen Wiederverwendung von neuronalen Netzwerken, die auf dem bzw. den Quellproblem(-en) vortrainiert wurden [121].

Relationaler Wissenstransfer (siehe Abbildung 2.7, Fall IV) umfasst Ansätze, die relationales Wissen zwischen Quell- und Zielproblemen direkt abbilden [113–115], was typischerweise Expertenwissen über die Domänen dieser Probleme erfordert [114]. Mit Deep Transfer Learning kann dies durch generative adversarische Netze erreicht werden [121].

Während die ersten beiden Kategorien primär datengetriebene Ansätze darstellen, d.h. zum Wissenstransfer auf die Veränderung der Trainingsdatensätze für das Zielproblem zurückgreifen, können die letzten beiden als modellgetriebene Ansätze bezeichnet werden, d.h. sie manipulieren oder tauschen zum Wissenstransfer die Modelle (repräsentiert in den jeweiligen Algorithmen) selbst [115]. Modelle bezieht sich in diesem Kontext somit auf abstrakte Modelle, die die Entscheidungsfunktion repräsentieren, d.h. die Gewichte und Topologien neuronaler Netzwerke, während z. B. mechanische, elektrische oder funktionale Modelle nicht gemeint sind [116].

2.2.1.3 Negativer Wissens-Transfer

Wenn mittels des Transfers von Wissen aus bekannten Problemen das Erlernen von Lösungen zu neuen Problemen beeinflusst werden kann, dann muss eine solche Beeinflussung nicht zwangsläufig positiv sein. Ein schädlicher, d.h. das Ergebnis verschlechternder, Wissens-Transfer wird daher negativer Transfer genannt [113–115, 122].

Nach [122] ist der Leistungsunterschied eines gegebenen Lernalgorithmus mit und ohne Nutzung von Daten des Quellproblems als **Transfer-Leistung** anzusehen. Ist diese Leistung ohne Nutzung von Daten des Quellproblems besser, so liegt **negativer Transfer** vor. Ursache negativen Transfers sind demnach Divergenzen in der gemeinsamen Wahrscheinlichkeitsverteilung (engl.: Joint Distribution) von Quell- und Zielmerkmalsräumen bzw. deren Nutzung im Rahmen des Wissens-Transfers. **Positiver Transfer** ist demnach die Ausnutzung systematischer Ähnlichkeiten zwischen den Merkmalsräumen. Die Menge gelabelter Zieldaten hat dabei einen ambivalenten Einfluss auf das Auftreten von negativem Transfer: Einerseits verbessern größere Mengen gelabelter Zieldaten die Leistung des Algorithmus ohne Wissens-Transfer und erhöhen damit relativ die Wahrscheinlichkeit negativen Transfers. Andererseits ermöglichen erst größere Mengen gelabelter Zieldaten die Untersuchung der gemeinsamen Wahrscheinlichkeitsverteilung und damit eine systematische Eindämmung negativen Transfers [122].

In Abwesenheit bekannter Verteilungsfunktionen für Quell- und Zielmerkmalsräume wird eine Ähnlichkeits-basierten Auswahl von Transfer-Wissen empfohlen. Auf diese Weise konnte in verschiedenen Studien eine deutliche Reduzierung des negativen Transfers bis hin zu positivem Transfer erreicht werden [123–126]. Trotzdem wird negativer Wissens-Transfer weiterhin als Herausforderung insbesondere für den praktischen Einsatz von Transfer-Lernen angesehen [114, 115, 122].

2.2.2 Kontinuierliches Lernen

Während der Begriff "Kontinuierliches Lernen" in unterschiedlichen Kontexten und verschiedenen Forschungsbereichen verwendet wird, beschreibt er hier ein Gebiet, das sich auf die Lösung

von (Teil-)Problem-übergreifenden Klassifikationsproblemen mit bekannten Quell- und Ziellabels unter Verwendung von Deep-Learning-Methoden konzentriert. Ziel ist es, Probleme sequenziell zu lernen ohne die Lösungen zuvor gelernter Probleme zu vergessen - ein Phänomen, das als "katastrophales Vergessen" bezeichnet wird [127] – so dass schließlich alle betrachteten Probleme von einem einzigen Deep-Learning-Algorithmus gelöst werden können [118, 128]. Dafür muss eine zweckmäßige Lösung des sogenannten Stabilitäts-Plastizitäts-Dilemmas, welches die Gegensätzlichkeit der gewünschten Eigenschaften Konservierung von Lösungen bekannter (Teil-) Probleme und Anpassung an neue (Teil-)Probleme beschreibt [129], gefunden werden [84]. Obwohl es also auch beim kontinuierlichen Lernen um Wissenstransfer geht, unterscheiden sich Zielsetzung, Methodik und Terminologie deutlich von der des Transfer-Lernens [116]. Abbildung 2.8 veranschaulicht den dem kontinuierlichen Lernen zugrunde liegenden Wirk-Mechanismus.

Die **Problemkategorien** des kontinuierlichen Lernens können als feinere Unterteilungen der Problemkategorie des homogenen induktiven Transfer-Lernens angesehen werden. Dabei ist jedoch zu beachten, dass für kontinuierliche Lernansätze eine bloße Anpassung der Lernalgorithmus-Fähigkeiten auf das Zielproblem nicht ausreichend ist, sondern dass stattdessen eine sowohl Quell- als auch Zielprobleme lösende Erweiterung derselben notwendig ist [116–118].

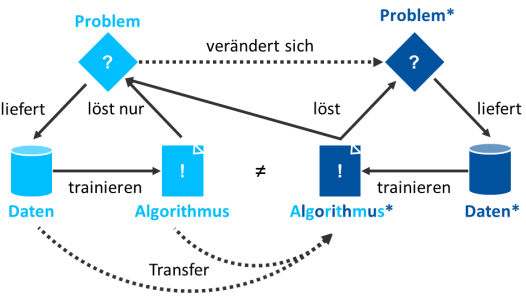

Abbildung 2.8: Kontinuierliches Lernen auf veränderlichen Problemen (nach [5])

Nach [128] werden beim kontinuierlichen Lernen üblicherweise drei Hauptlösungskategorien unterschieden. Sie können mit den **Lösungskategorien** des Transfer-Lernens in Verbindung gebracht werden:

- **Architektur-Strategien** sind Ansätze, die die Architektur, also z. B. die Anzahl oder Art der Schichten, eines neuronalen Netzwerks verändern, um den Effekt des katastrophalen Vergessens abzumildern. Analoge Ansätze im Transfer-Lernen sind im Parameter- oder relationale Wissenstransfer zu finden [116].

- **Regularisierungs-Strategien** sind Ansätze, die die Verlustfunktion eines Algorithmus so erweitern, dass die Veränderung von Parametern, die für das Lösen von zuvor gelernten

Problemen wichtig sind, bestraft wird. Analoge Ansätze im Transfer-Lernen sind im Parametertransfer zu finden [116].

- **Wiederholungs-Strategien** sind Ansätze, die Daten von zuvor trainierten Problemen beim Training neuer Probleme wiederverwenden. Analoge Ansätze im Transfer-Lernen sind im Instanz- oder Merkmalsrepräsentations-Transfer zu finden [116].

Eine detailliertere Diskussion dieser Strategien findet sich in [118, 128, 130].

2.2.3 Abgrenzung von weiteren Konzepten

Im Zusammenhang mit (Teil-)Problem-übergreifendem Deep Learning werden teilweise auch weitere Konzepte aufgebracht [131–134]. Dazu gehören insbesondere die Folgenden:

Föderales Lernen (engl.: Federated Learning) [135] bezeichnet den Wissenstransfer zwischen dezentralen Clients und zentralem Server im Rahmen verteilten Lernens, um gemeinsam verschiedene Varianten eines Problems zu lösen ohne die zugrundliegenden Daten auszutauschen [5, 111].

One-/Single-/Few-Shot Learning (dt.: Lernen mit einem bzw. wenigen Versuchen) ist ein Sammelbegriff für Lernalgorithmen, deren Ziel es ist, die Menge an benötigten Trainingsdaten zu minimieren. Dabei werden häufig Methoden des Transfer- oder kontinuierlichen Lernens eingesetzt [4, 5, 100, 111].

Aus Sicht des Autors dieser Arbeit ist hier eine Überschneidung mit (Teil-)Problem-übergreifendem Deep Learning nur zufällig gegeben und nicht konstituierend für die genannten Konzepte. Dies unterscheidet sie von Transfer- und kontinuierlichem Lernen, das grundsätzlich (Teil-)Problem-übergreifend ist.

2.3 Clustering

Clustering ist eine zentrale Technik der Datenanalyse und hat zum Ziel, Proben innerhalb eines oder mehrerer Datensätze anhand ihrer Proximität zu gruppieren. Dabei werden ähnliche Proben jeweils demselben Cluster und unähnliche Proben jeweils unterschiedlichen Clustern zugeordnet. Das Resultat des Clusterings ist somit eine Menge an Clustern, denen alle Proben zugeordnet sind, wobei jede Probe exakt einem Cluster angehört [82, 136].

Clustering ist grundsätzlich deskriptiv in dem Sinne, dass die Zuordnung ausschließlich auf bereits im Datensatz vorhandenen, intrinsischen Informationen beruht. Dies ermöglich die unüberwachte Ausführung von Clustering-Algorithmen und unterscheidet Clustering von der Klassifikation, die eine prädiktive Zuweisung auf Basis von über den Datensatz hinausgehenden, extrinsischen (Klassen-)Informationen erfordert und somit nur überwacht stattfinden kann [3, 82, 136].

Keines der im Folgenden genannten Clustering-Verfahren funktioniert ohne a priori gegebene Abbruchbedingungen bzw. Vorgaben, typischerweise die Anzahl Cluster betreffend. Somit ist es notwendig, die aus einer solchen Vorgabe resultierenden Clustering-Ergebnisse zu validieren [3, 82, 136–140]. Abbildung 2.9 zeigt den vollständigen Ablauf des Clustering-Prozesses unter Einbeziehung der Cluster-Validierung.

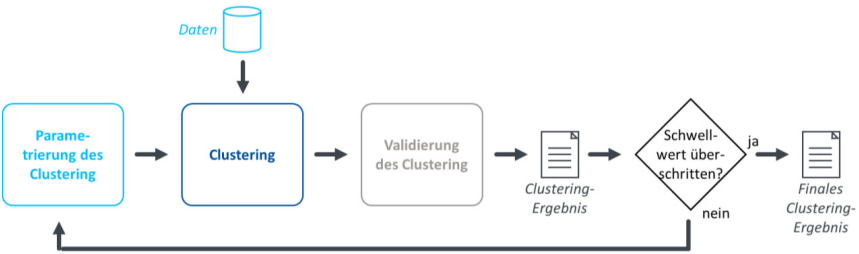

Abbildung 2.9: Schematischer Ablauf des Clustering-Prozesses

2.3.1 Proximitätsmetriken

Da Clustering Zuordnungen basierend auf der Proximität durchführt, muss diese messbar gemacht werden. Dafür können unterschiedliche Proximitätsmetriken eingesetzt werden. Die Wahl einer dieser Metriken beeinflusst das Ergebnis des Clusterings [136, 141].

Allgemein werden Proximitätsmetriken in Abstands- oder Ähnlichkeits-Metriken untergliedert [136, 141]:

Eine funktionale **Abstandsmetrik** ist symmetrisch, ergibt also unabhängig von der Betrachtungsrichtung für zwei Proben x_i, x_j den gleichen Abstand $d(x_i, x_j)$. Darüber hinaus nimmt der Abstand mit zunehmender Proximität ab, sodass er für identische Proben 0 wird. Die bei weitem gängigsten Abstandsmetriken für numerische Daten sind der Euklidische Abstand

$$d_e(x_i, x_j) = \sqrt[2]{\sum_{k=1}^{p} (x_{i,k} - x_{j,k})^2}$$

und der Manhattan- oder Cityblock-Abstand

$$d_m(x_i, x_j) = \sum_{k=1}^{p} |x_{i,k} - x_{j,k}|$$

als Sonderformen des allgemeineren Minkowski-Abstands mit p als der Vektorlänge bzw. Dimensionalität der Proben. Eine spezifische Gewichtung der einzelnen Dimensionen kann mittels eines Vor-Faktors ω erfolgen, setzt jedoch Kenntnisse über deren Relevanz voraus, die vielfach erst mittels des Clusterings ermittelt werden soll [136, 141–143].

Eine funktionale **Ähnlichkeitsmetrik** ist ebenfalls symmetrisch, ergibt also unabhängig von der Betrachtungsrichtung für zwei Proben x_i, x_j die gleiche Ähnlichkeit $s(x_i, x_j)$, nimmt jedoch mit zunehmender Proximität zu und erreicht für identische Proben ihren Maximalwert. Eine gängige Ähnlichkeitsmetrik für nummerische Daten ist die Kosinus-Ähnlichkeit

$$s(x_i, x_j) = \frac{x_i^T \cdot x_j}{|x_i| \cdot |x_j|} = \cos(\Theta)$$

mit Θ als dem Winkel zwischen x_i und x_j [136, 143].

2.3.2 Clustering-Verfahren

Clustering-Verfahren lassen sich in verschiedene, auf ihrem Ansatz basierenden Kategorien einteilen. Dazu zählen neben den im Folgenden aufgeführten hierarchischen, partitionierenden oder Dichte-basierten Clustering-Verfahren, die im weiteren Verlauf dieser Arbeit eine Rolle spielen, noch mindestens Fuzzy und Graphen-basiertes Clustering [136].

Hierarchische Clustering-Algorithmen verfahren Distanz-basiert und schrittweise kontinuierlich, sodass nachfolgende Schritte grundsätzlich nicht die Entscheidungen vorangegangener Schritte korrigieren können [136, 144]. Sie werden hinsichtlich der Vorgehens-Richtung in zwei Unter-Kategorien unterteilt:

- **Agglomerierende Verfahren** fügen einzelne Proben sukzessive zu Clustern zusammen. Sie beginnen mit Proben ohne Cluster-Zuordnung und enden ohne Abbruchbedingung erst, wenn alle Proben demselben Cluster angehören. Gängige Beispiele für agglomerierendes Clustering sind das Nächster- [145] und das Fernster-Nachbar-Verfahren [144].

- **Trennende Verfahren** unterteilen Cluster sukzessive immer feiner. Sie beginnen mit einem einzelnen Cluster, das alle Proben umfasst und enden ohne Abbruchbedingung erst, wenn alle Cluster lediglich aus einer Probe bestehen. Trennende Clustering-Verfahren sind rechnerisch aufwändiger als agglomerierende Verfahren und werden daher seltener eingesetzt [82, 144].

Ebenfalls ein hierarchischer Clustering-Algorithmus, allerdings speziell für große Datenmengen optimiert, ist der Balanced Iterative Reducing and Clustering using Hierarchies (dt.: Ausgewogenes iteratives Reduzieren und Gruppieren unter Verwendung von Hierarchien, kurz: **BIRCH**)

Clustering-Algorithmus [146]. Im weitesten Sinne agglomerierend erstellt BIRCH einen Clustering Merkmalsbaum, wobei die maximale Anzahl Proben im selben Verzweigungsknoten eines Clusters durch den sogenannten Verzweigungsfaktor und die maximale Distanz der Proben in den Verzweigungsknoten auf der untersten Ebene eines Clusters durch einen Distanzschwellenwert vorgegeben werden müssen. BIRCH hat Probleme mit nicht-kreisförmigen oder deutlich unterschiedlich großen Clustern [82].

Partitionierende Clustering-Algorithmen erlauben den Wechsel einzelner Proben zwischen Clustern. Grundlage ist dabei meist die Minimierung eines Cluster-übergreifenden, bspw. Distanz-basierten Fehlerwerts (Fehlerminimierungs-Algorithmen). Aus Performanz-Gründen wird vielfach ein heuristisch-iteratives statt eines mathematisch-vollständigen Vorgehens genutzt. Dadurch sind partitionierende Clustering-Verfahren anfällig für lokale Minima. Sie haben darüber hinaus Probleme mit isoliert liegenden Proben [82, 136]. Der **K-Means Clustering-Algorithmus** [147] ist der einfachste und häufigste Fehlerminimierungs-Algorithmus. Ein Cluster wird hier durch den Schwerpunkt (d.h. Mittelpunkt) der ihm zugeordneten Proben repräsentiert. Vor dem Start des Algorithmus müssen diese Schwerpunkte (ggf. zufällig) initialisiert werden, was erhebliche Auswirkungen auf die Ergebnisse haben kann. Anschließend werden alle Proben dem Cluster mit dem ihnen nächsten Schwerpunkt zugewiesen, woraufhin die Schwerpunkte neukalkuliert werden [82, 136, 148]. Der **Mini-Batch K-Means Clustering-Algorithmus** [149] ist eine Erweiterung des K-Means Clustering-Algorithmus, der an Stelle der gesamten Probenzahl zufällig ausgewählte Untermengen (sogenannte Mini-Batches) nutzt. Es ließ sich zeigen, dass dies die rechnerische Komplexität erheblich reduzierte, ohne das Ergebnis signifikant zu beeinträchtigen [149].

Dichte-basierte Clustering-Algorithmen gruppieren Proben anhand ihrer räumlichen Dichte, also der Anzahl anderer Proben innerhalb einer gewissen Distanz. Sie können damit unregelmäßig geformt sein [82, 148]. Dichte-basiertes Clustering hat jedoch Probleme mit Clustern unterschiedlicher Dichten [136]. Der Density-Based Spatial Clustering of Applications with Noise (dt.: Dichte-basierte räumliche Gruppierung verrauschter Anwendungen, kurz: **DBSCAN**) **Clustering-Algorithmus** [150] ist einer der gängigsten Dichte-basierte Clustering-Algorithmus [82]. Ihm muss eine Mindestanzahl Proben für ein neues Cluster sowie der Distanzschwellenwert, der die maximale Distanz zwischen zwei direkt verbundenen Proben innerhalb desselben Clusters beschreibt, vorgegeben werden [148, 150].

2.3.3 Cluster-Validierung

Ein guter Clustering-Algorithmus sollte Cluster mit geringen inneren (Kompaktheit) und großen äußeren Abständen (Separiertheit) schaffen [82]. Der **Silhouetten-Index** (SI) [139] ermöglicht eine solche Bewertung des Cluster-Ergebnis auf Basis des Unterschiedes des mittleren Abstands

aller Proben zu anderen Proben desselben Cluster und des mittleren Abstands aller Proben zu allen Proben des nächstgelegenen, anderen Clusters. Er ist definiert als

$$SI = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n_i}\frac{\mathrm{b_{j,i}} - \mathrm{a_{j,i}}}{\max\{\mathrm{a_{j,i}}, \mathrm{b_{j,i}}\}}$$

mit n als der Anzahl Proben im Datensatz, n_i als der Anzahl Proben im i-ten Cluster, $a_{j,i}$ als durchschnittlichem Abstand zwischen der j-ten Probe im i-ten Cluster und allen anderen Proben im selben Cluster, $b_{j,i}$ als durchschnittlichem Abstand zwischen der j-ten Probe im i-ten Cluster und allen Proben im nächstgelegenen, anderen Cluster, wobei $SI \in [0;\ 1]$ und ein höherer Wert eine bessere Bewertung darstellt. Der SI ist eher aufwendig zu berechnen, erlaubt jedoch durch die Normalisierung auf Probenzahl innerhalb der Cluster bzw. innerhalb des Datensatzes Vergleiche auch zwischen unterschiedlichen Datensätzen [137].

2.4 Anwendungskontext Industrie

Aus dem Anwendungskontext Industrie folgen mittelbar verschiedene, für diese Arbeit relevante Eigenschaften, die über die reine Verortung des Problems in „Institutionen, deren wirtschaftliche Tätigkeit überwiegend darin besteht, Elektrizität, Gas, Fernwärme und Waren zu erzeugen oder zu gewinnen oder in verschiedener Weise zu be- oder verarbeiten, und zwar mit dem Ziel, andere Waren herzustellen“ [151] hinausgeht.

In Kapitel 1.2 wurde bereits die Heterogenität der Datenarten im Anwendungskontext Industrie festgestellt. Zusätzlich dazu muss auf die zentrale Rolle von Zeitreihendaten verwiesen werden, die in dieser Form in anderen Anwendungskontexten nicht vorliegt [31, 35, 47, 49, 152–154]. Jegliche Lösungsansätze müssen also aktuell insbesondere zur **Nutzung von Zeitreihendaten** geeignet sein. Es ist denkbar, dass sich diese Charakteristik zukünftig zugunsten anderer Datenarten ändert, bspw. durch den verstärkten Einsatz von Ontologien und Wissensgraphen (engl.: Knowledge Graphs) [155].

In Kapitel 2.2.1.1 wurden bereits verschiedene Formen der Problemkategorisierung für Transfer-Lernen eingeführt. Im Anwendungskontext Industrie geht es aktuell um den Transfer zwischen identischen Quell- und Zielmerkmalsräumen, d.h. um **homogenes Transfer-Lernen** [45, 48, 156]. Dies liegt darin begründet, dass die praktische Anwendung von Transfer-Lernen noch ein sehr junges Forschungsfeld ist und auch der einfachere, weil auf größerer Ähnlichkeit von Quell- und Transferproblem basierende Fall des homogenen Transfer-Lernens noch nicht zufriedenstellend gelöst ist.

Die genannten Charakteristika des Anwendungskontext Industrie werden den folgenden Kapiteln zugrunde gelegt, ohne dort erneut explizit genannt zu werden.

3 Stand der Wissenschaft und Technik

In diesem Kapitel wird zum Einstieg in die Design-Science-Research-Phase „Lösungsansatzentwicklung“ ein Überblick über den Stand der Wissenschaft in Bezug auf für den Anwendungskontext Industrie prinzipiell geeignete Methoden (Teil-)Problem-übergreifenden Lernens gegeben. Darauffolgend wird der Stand der Technik in der anwendungsbezogenen Forschung hinsichtlich der beiden Anwendungsfälle „Anomaliedetektion“ und „Ausfallvorhersage“ wiedergegeben. Abschließend erfolgt eine Bewertung von Clustering-Methoden, bevor als Fazit eine konkrete Forschungsfrage identifiziert wird.

3.1 Ansätze für (Teil-)Problem-übergreifendes Deep Learning

In den folgenden Unterkapiteln wird der Stand der Wissenschaft hinsichtlich der im Anwendungskontext Industrie genutzten Methoden kontinuierlichen und Transfer-Lernens beschrieben. Dabei kann aufgrund der Vielfalt unterschiedlicher Verfahren nur eine jeweils sehr knappe Vorstellung erfolgen, für darüberhinausgehende Erläuterungen sei auf die jeweils aufgeführten Quellen verwiesen. Abschließend wird eine Bewertung der vorgestellten Ansätze vorgenommen. Einschätzungen zu deren Verbreitung basieren dabei wesentlich auf den in Kapiteln 3.2.1.3 und 3.2.2.3 vorgestellten Literaturstudien.

3.1.1 Kontinuierliches Lernen

In den letzten Jahren hat die Forschung an Methoden für kontinuierliches Lernen eine breite Auswahl an verschiedenen Ansätzen hervorgebracht. Einen Überblick über die verschiedenen Lösungskategorien gibt Kapitel 2.2.2. Für die praktische Anwendung im in Kapitel 1 beschriebenen Kontext eignet sich jedoch nicht alle davon:

Ansätze aus der Lösungskategorie der **Wiederholungs-Strategien** erfordern große Datenspeicher oder intelligente Auswahlverfahren für repräsentative Proben, die jedoch bei dynamischen Prozessen aufgrund der unbekannten, zukünftigen Entwicklung schwierig zu realisieren sind [63, 64, 87, 157, 158]. Sie werden daher im Folgenden nicht weiter betrachtet.

Innerhalb der **Architektur-Strategien** ist der von komplementären Lern-Systemen (engl.: Complementary Learning Systems) nach [159] inspirierte Ansatz der Zwei-Gedächtnis-Methode (engl.: **Dual Memory Method**, kurz: DMM) [130, 157, 160, 161] hervorzuheben: Ein langsam lernendes Modul (inspiriert vom Neokortex des Säugetiergehirns) und ein schnell lernendes Modul (inspiriert vom Hippocampus des Säugetiergehirns) kombinieren ihre Stärken, um (Teil-)Problem-übergreifendes Lernen zu realisieren. Das langsam lernende Modul dient dazu, allgemeine Informationen aus den Eingabedaten zu extrahieren und zu verallgemeinern. Im Gegensatz

dazu wird das schnell lernende Modul verwendet, um spezifische Informationen aus den Eingabedaten zu extrahieren und sie als neue Erkenntnisse zu speichern. Vom reinen Parametertransfer des Transfer-Lernens unterscheidet sich dieser Ansatz dahingehend, dass bei der DMM kontinuierlich jede neue Probe in das Training einbezogen wird und sich der Algorithmus damit auch dynamischen Prozessen fortwährend anpasst.

Innerhalb der **Regularisierungs-Strategien** existiert eine wachsende Zahl vielversprechender Methoden, wie bspw. elastische Gewichts-Konsolidierung (engl.: Elastic Weight Consolidation) [162], dessen Online-Version [163], synaptische Intelligenz [164] oder Lernen ohne Vergessen (engl.: Learning Without Forgetting) [165]. Sie alle beruhen auf der Annahme, dass mehr als ein Parametersatz eine Lösung eines konkreten (Teil-)Problems darstellen kann, sodass in vielen Fällen andere Parametersätze eine Lösung sowohl des ursprünglichen als auch eines neuen (Teil-)Problems darstellen können sollten (siehe Abbildung 3.1). Dies wird über eine Veränderung der Verlustfunktion erreicht, die die relative Wichtigkeit eines Parameters innerhalb eines Parametersatzes auf die Lösungsqualität für das bzw. die ursprüngliche(-n) (Teil-)Problem(-e) einbezieht und eine Veränderung wichtiger Parameter bestraft. Aufgrund der Art und Weise, mit der diese relative Wichtigkeit bspw. mittels der sogenannten Fisher-Informations-Matrix bestimmt wird [166], eignen sich die zuvor genannten Ansätze nur für Klassifikations- und nicht für Regressionsprobleme [63, 64, 87]. Der Ansatz Gedächtnis-bewusster Synapsen (engl.: Memory Aware Synapses) [167] ist dagegen aufgrund eines anderen Vorgehens auch auf Regressions-Probleme anwendbar: Statt der Wichtigkeit eines Parameters innerhalb eines Parametersatzes für die Lösungsqualität wird sein Einfluss auf die Ausgabewerte des gelernten Modells (unabhängig davon, ob korrekt oder inkorrekt) als Grundlage für den mit seiner Veränderung einhergehenden zusätzlichen Verlust genommen. Ein praktischer Vergleich verschiedener Ansätze findet sich u. a. in [168–170].

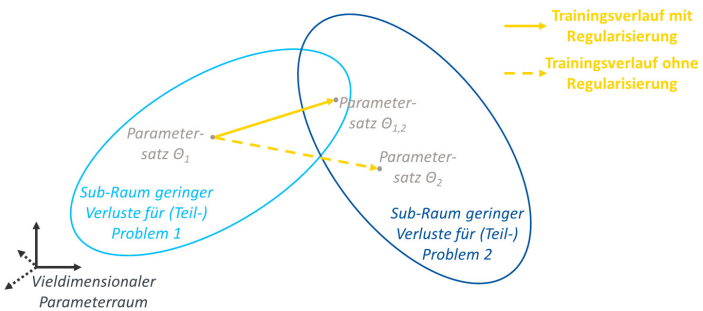

Abbildung 3.1: Grundprinzip Regularisierungs-basierten kontinuierlichen Lernens

3.1.2 Transfer-Lernen

In den letzten Jahren hat die Forschung an Transfer-Lern-Methoden eine breite Auswahl an verschiedenen Ansätzen hervorgebracht. Einen Überblick über die verschiedenen Lösungskategorien gibt Kapitel 2.2.1.2. Für die praktische Anwendung im in Kapitel 1 beschriebenen Kontext eignet sich jedoch nicht alle davon:

Ansätze aus der Lösungskategorie des **Instanztransfers** eignen sich aufgrund der Anforderungen an eine manuelle Auswahl der zu transferierenden Ansätze nicht für einen praktischen Einsatz in (hoch-)automatisierten Umfeldern. Ansätze aus der Lösungskategorie des **relationalen Wissenstransfers** existieren nur in geringer Zahl und dem Autor sind keine unter Verwendung von Deep-Learning-Methoden bekannt.

Im Folgenden wird der aktuelle Stand der Algorithmen-Entwicklung für die in dieser Arbeit relevanten Transfer-Konzepte vorgestellt.

3.1.2.1 Merkmalsrepräsentationstransfer

Aufgrund der in dieser Arbeit zugrunde gelegten Nutzung von Deep Learning, der Nutzbarkeit im Anwendungskontext Industrie sowie der in Kapitel 1.4 formulierten Anforderungen, verbleiben lediglich vier der in [114, 115] beschriebenen, gängigen Merkmalsrepräsentationstransfer-Ansätze. Sie alle führen einen symmetrischen Merkmalsrepräsentationstransfer durch [171] und werden im Folgenden vorgestellt.

Ein zentrales Konzept des Merkmalsrepräsentationstransfers ist die **Domänen-Adaption** [100, 172]. Es wird zwischen unüberwachter, d.h. auf Ziellabel komplett verzichtender, und semi-überwachter, d.h. nur wenige Ziellabel benötigender, Domänen-Adaption unterschieden [173]. Unüberwachte Domänen-Adaption ist transduktives Transfer-Lernen [113]. Domänen-Adaption beruht üblicherweise auf der Minimierung des Abstands zwischen den Merkmalsverteilungen der verschiedenen (Teil-)Probleme. Gängige Metriken für diesen Abstand sind nach [115] die maximale mittlere Abweichung [zwischen Merkmalsverteilungen] (engl.: Maximum Mean Discrepancy [between feature distributions], kurz: MMD) oder die Kullback-Leibler-(KL-)Divergenz [174].

[175] beschreibt die **unüberwachte Domänen-Adaption mittels Fehlerrückführung** (engl.: Unsupervised Domain Adaptation by Backpropagation, kurz: UDAB) und ist dreiteilig aufgebaut: Ein Merkmalsextrakteur verarbeitet sowohl Quell- als auch Zielproben, ein Prädiktor löst basierend auf den extrahierten Merkmalen das eigentliche Problem und ein Diskriminator versucht zwischen den extrahierten Merkmalen von Quell- und Zielproblem zu unterscheiden. Während Prädiktor (nur auf Quellproben) und Diskriminator im Training erlernen sollen, ihr jeweiliges

Teilproblem optimal zu lösen, wird der Merkmalsextrakteur mit dem Ziel, den Prädiktor zu verbessern und den Diskriminator zu verschlechtern, optimiert. Die Abweichungen zwischen den gemeinsamen Merkmalsverteilungen werden also nicht explizit, sondern nur implizit durch den Diskriminator bestimmt. [175] nutzt eine gemeinsam auf ihren jeweiligen Verteilungsfunktionen trainierte Kombination aus CNN als Merkmalsextrakteur und FCNN als Diskriminator und Prädiktor zur Lösung einer Klassifikationsaufgabe. Prinzipiell ist dieser unüberwachte Ansatz jedoch auf andere, ebenfalls auf Fehlerrückführung basierende Architekturen und auch Regressionsaufgaben übertragbar.

[176] beschreibt die **Mehrschicht-Domänen-Adaption** (engl.: Multi-Layer Domain Adaptation, kurz: MLDA). Über eine angepasste Verlustfunktion, die auch die MMD berücksichtigt, wird die Abweichung zwischen den gemeinsamen Merkmalsverteilungen von Quell- und Zieldatensatz über die gesamte Tiefe des Netzwerks hinweg reduziert. [176] nutzt eine gemeinsam auf derselben Verlustfunktion trainierte Kombination aus CNN als Merkmalsextrakteur und FCNN als Klassifikator zur Lösung einer Klassifikationsaufgabe. Prinzipiell ist dieser unüberwachte Ansatz jedoch auf andere Architekturen und auch Regressionsaufgaben übertragbar.

[177] beschreibt die **Transferkomponenten-Analyse** (engl.: Transfer Component Analysis, kurz: TCA). In einem Hilbertraum mit reproduzierendem Kern (engl.: Reproducing Kernel Hilbert Space) werden unter Minimierung der (lediglich marginalen) MMD neue, gemeinsame Merkmale für Quell- und Zieldatensatz gesucht, die jedoch die ursprüngliche Varianz der beiden Datensätze erhalten. Diese Merkmalssuche wird dem eigentlichen Deep Learning mittels neuronaler Netzwerke vorgeschaltet. Damit ist dieser unüberwachte Ansatz mit verschiedenen Netzwerk-Architekturen nutzbar.

[178] beschreibt den **mehrstufigen, entrauschenden Autoencoder** (engl.: Stacked Denoising Autoencoder, kurz: SDA). Einem solchen SDA werden im Training sowohl Quell- als auch Zielproben zugeführt, wodurch er gemeinsame Merkmale finden muss, die in beiden Domänen funktionieren. Die Abweichungen der (lediglich marginalen) Merkmalsverteilungen werden also gar nicht bestimmt, sondern direkt zu reduzieren versucht. Diese Merkmalssuche, die ebenfalls Deep-Learning-basiert sein kann, wird dem Deep Learning zur eigentlichen Problemlösung mittels neuronaler Netzwerke vorgeschaltet. Damit ist dieser unüberwachte Ansatz mit verschiedenen Netzwerk-Architekturen nutzbar, wobei nicht jede SDA auch eine Domänen-Adaption darstellen muss.

3.1.2.2 Parametertransfer

Aufgrund der in dieser Arbeit zugrunde gelegten Nutzung von Deep Learning und der Homogenität der in ihr betrachteten, industriellen Transfer-Probleme (siehe Kapitel 2.4) können drei Ansätze des Parametertransfers unterschieden werden (siehe Abbildung 3.2). Da Parametertransfer

üblicherweise induktives Transfer-Lernen ist [113], kann hier zumeist nicht auf Label des Zieldatensatzes verzichtet werden. Sofern der Parametertransfer jedoch die Menge zum Training benötigter Zieldaten reduziert, reduziert er gleichzeitig den Labeling-Aufwand.

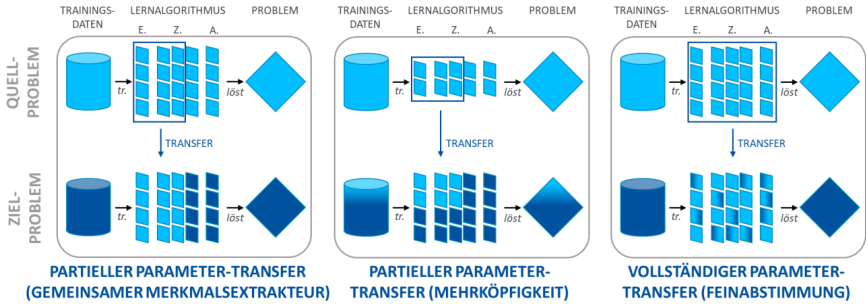

Abbildung 3.2: Parametertransferansätze

Der partielle Parametertransfer umfasst Ansätze, die lediglich die Parameter des Merkmalsextrakteurs vom auf dem Quellproblem trainierten Lernalgorithmus an den Zielproblem-Lernalgorithmus weitergeben [32]. Der Merkmalsextrakteur ist dann nicht Gegenstand des Trainings auf dem Zielproblem, sondern wird währenddessen fixiert. Eine derartige Nutzung eines **gemeinsamen Merkmalsextrakteurs** reduziert den Trainings-Aufwand auf dem Zielproblem, da lediglich ein kleiner Teil des gesamten Lernalgorithmus noch trainiert werden muss [179].

Eine Abwandlung des partiellen Parametertransfers ist die sogenannte **Mehrköpfigkeit** (engl.: Multi-Headedness). Hier wird für (bspw. aus verschiedenen Sensor-Werten) zusammengesetzte Proben pro Teilprobe ein separater Merkmalsextrakteur genutzt [49, 180, 181]. Im Transferfall werden die Parameter der Merkmalsextrakteure vom auf dem Quellproblem trainierten Lernalgorithmus ebenfalls an den Zielproblem-Lernalgorithmus weitergeben – sofern auch das Zielproblem entsprechende Teilproben aufweist [49, 181]. Der Vorteil ist, dass die Merkmalsextraktion auf diesem Weg besser zum jeweiligen (Teil-)Problem passt und nicht notwendige Teil-Algorithmen verworfen werden.

Der vollständige Parametertransfer umfasst Ansätze, die alle Parameter und Initialisierungen des auf dem Quellproblem trainierten Lernalgorithmus an den Zielproblem-Lernalgorithmus weitergeben [32]. Der transferierte Lernalgorithmus wird anschließend auf dem Zielproblem weitertrainiert. Dieses auch **Feinabstimmung** (engl.: Finetuning) genannte Verfahren reduziert den Trainings-Aufwand auf dem Zielproblem, da der Lernalgorithmus bereits vortrainiert ist [179].

3.1.3 Abstrakte Bewertung der Ansätze

Kontinuierliches Lernen ist derzeit primär im Kontext der Bilderkennung, der Verarbeitung natürlicher Sprache (engl.: Natural Language Processing) und im bestärkenden Lernen weit verbreitet [87, 182, 183].

DMM wird in Einzelfällen bereits im Anwendungskontext Industrie eingesetzt [157, 160]. Auch Methoden aus dem Bereich der **Regularisierungs-Strategien** kommen hier vereinzelt schon zum Einsatz (siehe Kapitel 3.2.1.3 und 3.2.2.3). Nachteilig ist jedoch Prinzip-bedingt, dass kein Vergessen vorgesehen ist, bei dynamisch veränderlichen Prozessen also bspw. der Urzustand langfristig weiter Kapazität belegt, ohne absehbar wieder benötigt zu werden. Für die Regularisierungs-Methoden gilt darüber hinaus, dass sie bisher unverstandene Abhängigkeiten von Ähnlichkeit, Richtung, Reihenfolge und Anzahl von Quell- und Zielproblemen aufweisen, was einen produktiven Einsatz aufgrund mangelnder Determiniertheit weiter erschwert [63, 64, 184].

Transfer-Lernen wird bereits sehr breit erforscht. Merkmalsrepräsentationstransfer ist bspw. in der Ausfallvorhersage weitverbreitet (siehe Kapitel 3.2.2.3). Dabei kommen vor allem **Domänen-Adaptions-Varianten** zum Einsatz. Dies ist darin begründet, dass derartige Ansätze in einem Schritt sowohl die Merkmalsräume anpassen als auch das eigentliche Problem lösen. Gleichzeitig wird in UDAB und MLDA nicht nur die marginale Verteilung, sondern auch die bedingte Verteilung betrachtet, was sich vorteilhaft auf die Lösungsqualität auswirkt [185, 186]. Nachteilig ist jedoch, dass Domänen-Adaption meist nur zwischen zwei (Teil-)Problemen stattfindet und eine breitere Mischung unterschiedlicher Quellprobleme nicht vorgesehen ist. **Parametertransfer** ist bspw. in der Bild- und Objekterkennung sowie in der Verarbeitung natürlicher Sprache weit verbreitet: State-of-the-art Lernalgorithmen, wie bspw. VGG-16 [26] oder ResNet50 [187] in der Bilderkennung bzw. GPT-3 [25] oder BERT [188] in der Verarbeitung natürlicher Sprache, wurden auf umfangreichen Trainingsdatensätzen vortrainiert und können nun mit geringem Aufwand und unter Verwendung kleiner Datensätze auf neue (Teil-)Probleme angepasst werden – ohne, dass sie an Leistungsfähigkeit verlieren [4]. Dabei kommt sowohl partieller als auch vollständiger Parametertransfer zum Einsatz.

Beide Ansätze haben jedoch Prinzip-bedingte Nachteile: Die gemeinsame Merkmalsextraktion reduziert den Trainingsaufwand, lässt jedoch die Lösungs-zentralen letzten Schichten des Netzwerks außen vor. Die Feinabstimmung wiederum legt für jedes neue (Teil-)Problem einen vollständigen, neuen Parametersetz an, der komplett neu trainiert werden muss – sie ist also Parameter-ineffizient [189, 190].

Allgemein ist festzustellen, dass eine strenge Unterscheidung zwischen kontinuierlichem und Transfer-Lernen nicht zweckdienlich ist [116]. Die gemeinhin als Transfer-Lern-Ansatz angese-

hene Domänen-Adaption erfüllt bspw. auch die Definition des kontinuierlichen Lernens. Im Folgenden wird daher auf eine Unterscheidung zwischen kontinuierlichem Lernen und Transfer-Lernen verzichtet und allgemein von (Teil-)Problem-übergreifendem Deep Learning gesprochen.

Für eine spezifische Bewertung der im Anwendungskontext Industrie genutzten Methoden kontinuierlichen und Transfer-Lernens hinsichtlich der Erfüllung der in Kapitel 1.4 definierten Anforderungen sei auf das folgende Kapitel verwiesen, indem derartige Deep-Learning-Methoden Anwendungsfall-spezifisch betrachtet werden.

3.2 Anwendungsfall-spezifische Betrachtung von Deep-Learning-Methoden

In den folgenden Unterkapiteln wird der Stand der Technik hinsichtlich der in den Anwendungsfällen Anomaliedetektion und Ausfallvorhersage eingesetzten Deep-Learning-Methoden mit besonderem Fokus auf (Teil-)Problem-übergreifendes Lernen beschrieben. Abschließend wird eine Bewertung der vorgestellten Ansätze vorgenommen. Für eine Erläuterung der Auswahl der Anwendungsfälle sei auf Kapitel 5.1 verwiesen.

3.2.1 Anwendungsfall Anomaliedetektion

3.2.1.1 Überblick über den Anwendungsfall

Anomalien sind Abweichungen von einem als normal definierten Zustand [32, 191–194]. In technischen Systemen sind sie Ausdruck von unerwünschten Betriebszuständen jenseits des intendierten Systemverhaltens, die zu Ineffizienzen oder teilweisen bis vollständigen Systemausfällen führen können [32]. Die Anomaliedetektion ist daher ein intensiv beforschtes Anwendungsfeld – sowohl im Kontext industrieller Anwendungen als auch darüber hinaus [32, 191–194].

Üblicherweise unterscheidet man drei verschiedene Arten von Anomalien: **Punktuelle Anomalien** sind Abweichungen einer einzelnen Probe relativ zum Vergleichsdatensatz. **Kontextuelle Anomalien** sind Abweichungen, die nur aufgrund ihres Kontexts anormal sind. Zur Beurteilung eines solchen Kontextes müssen entweder kontextuelle oder behaviorale Zusammenhänge für den Datensatz bekannt sein. **Kollektive Anomalien** sind Anomalien, die erst durch das Zusammentreffen mehrerer entsprechender Proben anormal sind [32, 191].

Anomaliedetektion mittels maschinellen Lernens kann in Abhängigkeit des konkreten Anwendungsfalls auf verschiedene Arten durchgeführt werden: **Überwachte Anomaliedetektion** nutzt gelabelte Trainingsdatensätze, um die Charakteristika normaler und (potenziell verschiedener Arten) anormaler Proben im Sinne eines Klassifikationsproblems zu erlernen. Herausfordernd ist

dabei üblicherweise der geringe Anteil anormaler Daten sowie deren ausreichende Repräsentativität bzw. Vollständigkeit und sicheres Labeling. **Semi-überwachte Anomaliedetektion** adressiert diese Herausforderungen und nutzt lediglich normale Proben zum Training. Dadurch reduziert sich das Problem zu einem sogenannten Ein-Klassen-Klassifikationsproblem (engl.: One-Class Classification) [195, 196], dass auch unbekannte Anomalien aufgrund ihrer Abweichung vom erlernten Normalzustand erkennbar macht. **Unüberwachte Anomaliedetektion** wiederum kommt völlig ohne gelabelte Trainingsdatensätze aus und basiert auf der Annahme, dass normale Proben viel häufiger sein werden als anormale [191–194].

3.2.1.2 Allgemeines Deep Learning

Verschiedene Deep-Learning-Methoden können zur Anomaliedetektion eingesetzt werden. Unterschieden werden dabei grundsätzlich zwei unterschiedliche Herangehensweisen:

Diskriminative Ansätze detektieren Anomalien, indem sie den Unterschied zwischen normalen und anormalen Klassen (Mehr-Klassen-Klassifikation) bzw. zwischen der normalen Klasse und nicht dazugehörigen Proben (Ein-Klassen-Klassifikation) erlernen. Die Anomaliedetektion wird somit als Klassifikationsproblem behandelt. Zu den diskriminativen Ansätzen gehören daher prinzipiell alle gängigen Klassifikations-Algorithmen, insbesondere aber FCNN und RNN [192, 194]. Letztere erlauben die Berücksichtigung zeitlicher Kontexte in der Anomaliedetektion und sind damit geeignet für die Detektion kontextueller Anomalien [32]. Vorteilhaft für diskriminative Mehr-Klassen-Ansätze sind Datensätze mit einem hohen Anteil anormaler Daten. Für diskriminative Ein-Klassen-Ansätze sind hingegen Datensätze mit einem geringen Anteil anormaler Daten vorteilhaft [192].

Generative Ansätze detektieren Anomalien, indem sie die Proben zu rekonstruieren versuchen. Da sie (primär) mit normalen Proben trainiert wurden, können sie anormale Proben schlechter rekonstruieren. Zu den generativen Ansätzen gehören insbesondere Autoencoder und Deep Belief Networks, die ihrerseits aus Restricted Boltzmann Machines bestehen [192, 194]. In jüngster Vergangenheit geht der Trend in Richtung variationaler Autoencoder [197], die aufgrund ihres zusätzlichen Erlernens einer Verteilungsfunktion für die Eingangsdaten Anomalien sicherer erkennen [192]. Für Zeitreihendaten sind darüber hinaus auch alle gängigen Regressions-Algorithmen geeignet [198]. Vorteilhaft für generative Ansätze sind Datensätze mit einem niedrigen Anteil anormaler Daten [192].

Trotz vielfach sehr vielversprechender Ergebnisse auf isolierten Anwendungsfällen bei ausreichender Verfügbarkeit entsprechender Trainingsdatensätze [154, 199–203] weisen diese Ansätze jedoch grundsätzliche Lücken hinsichtlich des Aufwands beim Nachtrainieren (A2, siehe Kapitel 1.4) und der Wiederverwendbarkeit (A4, siehe Kapitel 1.4) der Lernalgorithmen auf. Dieser Umstand wird teilweise auch von den Autoren der zuvor genannten Veröffentlichungen anerkannt,

so wird bspw. in [199, 200, 202] eine Erweiterung des vorgestellten Ansatzes um die Fähigkeit zum Transfer-Lernen angeregt.

Nach [193, 194] kann der eigentlichen Anomaliedetektion eine Merkmalsextraktion oder eine Dimensionsreduzierung vorgeschaltet werden, um den Rechen- und Speicheraufwand zu begrenzen bzw. die Detektions-Qualität zu erhöhen.

3.2.1.3 (Teil-)Problem-übergreifendes Deep Learning

Methodik

Zur Identifizierung relevanter Publikationen für (Teil-)Problem-übergreifenden Deep Learning im Anwendungsfall Anomaliedetektion wurde eine zweistufige, systematische Literaturstudie (engl.: Systematic Literature Review) durchgeführt:

Im ersten Schritt wurde auf Google Scholar wissenschaftliche Veröffentlichungen identifiziert, die den in Tabelle 3.1 aufgelisteten Kombinationen von Suchbegriffen entsprechen. Für jede Suchanfrage wurde dazu jeweils ein Begriff der Kategorie „Suchbegriff 1“ mit dem einen Begriff der Kategorie „Suchbegriff 2“ kombiniert.

Tabelle 3.1: Suchbegriffe für die erste Stufe der systematischen Literaturstudie zum (Teil-)Problem-übergreifenden Deep Learning im Anwendungsfall Anomaliedetektion

Suchbegriff 1	Suchbegriff 2
Transfer Learning	Anomaly Detection
Continual Learning	

Im zweiten Schritt wurden die identifizierten wissenschaftlichen Veröffentlichungen manuell gefiltert. Nur englischsprachige[4], im Volltext vorliegende Forschungsbeiträge, die Deep-Learning-basierte Methoden und eine Form von Transfer- oder kontinuierlichem Lernen im Anwendungsfall Anomaliedetektion einsetzen und bis einschließlich 2021 erschienen sind, werden im Folgenden berücksichtigt.

Ergebnisse

Tabelle 3.2 gibt einen Überblick über die aus der zuvor beschriebenen systematischen Literaturstudie hervorgegangenen Ansätze (Teil-)Problem-übergreifenden Deep Learnings im Anwendungsfall Anomaliedetektion.

[4] Aufgrund der dominierenden Rolle des Englischen in der wissenschaftlichen Kommunikation im Bereich der KI-Forschung wurde auf die Einbeziehung deutsch- oder anderssprachiger Veröffentlichungen verzichtet. Daher sind auch die Suchbegriffe englisch.

Tabelle 3.2: Beispiele für die Nutzung von (Teil-)Problem-übergreifendem Deep-Learning im Anwendungsfall Anomaliedetektion

Quelle	Lern-Kategorie	Detektions-Ansatz	Datenart	Transfer-Methode
Canizo et al., 2019 [49]	Überwacht	Diskriminativ	Multivar. Zeitreihen	Mehrköpfiger Merkmalsextrakteur
Hsieh et al., 2019 [204]	Unüberwacht	Generativ	Multivar. Zeitreihen	Feinabstimmung
Liang et al., 2019 [205]	Überwacht	Generativ	Univ. Zeitreihen	Feinabstimmung
Müller et al., 2020 [206]	Überwacht	Diskriminativ	Univar. Zeitreihen (als Bild)	Gemeinsamer Merkmalsextrakteur
Xu et al., 2020 [48]	Semi-überwacht	Diskriminativ	Univar. Zeitreihen	Domänen-Adaption
Baireddy et al., 2021 [207]	Semi-überwacht	Generativ	Univar. Zeitreihen	Feinabstimmung / gemeinsamer Merkmalsextrakteur
Maschler et al., 2021 [63]	Überwacht	Diskriminativ	Univar. Zeitreihen	Regularisierung
Michau et al., 2021 [45]	Unüberwacht	Diskriminativ	Multivar. Zeitreihen / Bilder	Domänen-Adaption
Ullah et al., 2021 [208]	Überwacht	Diskriminativ	Metadaten	Gemeinsamer Merkmalsextrakteur
Wang et al., 2021 [209]	Unüberwacht	Diskriminativ	Multivar. Zeitreihen	Domänen-Adaption

In den letzten Jahren wurden mehrere Deep-Learning-basierten Ansätzen zur (Teil-)Problem-übergreifenden **diskriminativen Anomaliedetektion** mittels **Domänen-Adaption** im Anwendungskontext Industrie veröffentlicht:

[48] nutzt Cluster-basierte Domänen-Adaption für semi-überwachte, diskriminative Anomaliedetektion im Sinne einer Mehr-Klassen-Klassifikation. Der beschriebene Prototyp verwendet eine Kombination aus CNN und FCNN als Klassifikator. Eine Evaluation auf einem industriellen univariaten Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem. Eine Besonderheit des Ansatzes ist in Abgrenzung zu einer konventionellen Domänen-Adaption die zuvor auf Basis der Trainings-Klassifikationen vorgenommene Auslese der in die Domänen-Adaption eingehenden Proben, sodass Ausreißer und im Zielproblem nicht vorhandene Klassen

ausgeschlossen werden können. Dies verbessert nachweislich das Resultat der Domänen-Adaption. [45] nutzt adversarische Domänen-Adaption für unüberwachte, diskriminative Anomaliedetektion im Sinne einer Ein-Klassen-Klassifikation. Der beschriebene Prototyp verwendet als Merkmalsextraktor ein FCNN und als Klassifikator ein FCNN in Form einer extremen Lern-Maschine (engl.: Extreme Learning Machine) [210]. Eine Evaluation auf zwei industriellen Zeitreihen-Datensätzen und einem Benchmark-Bild-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem. Dafür notwendig ist eine annähernd gleiche Anzahl von Quell- und Zielproben sowie unterschiedliche Betriebsbedingungen in Quell- und Zieldaten. [209] nutzt Ähnlichkeits-basierte Domänen-Adaption für unüberwachte, diskriminative Anomaliedetektion im Sinne einer Ein-Klassen-Klassifikation. Der beschriebene Prototyp verwendet FCNN als Merkmalsextraktor und Klassifikator. Eine Evaluation auf zwei industriellen Zeitreihen-Datensätzen soll den positiven Transfer zwischen Quell- und Zielproblem belegen, wobei als Metrik nicht die Definition gemäß Kapitel 2.2.1.3 genutzt wird, sondern das Verhältnis der Leistung auf dem Quell- zum Zielproblem. Da gemäß dieser Definition ein negativer Transfer nicht möglich ist, erscheint diese Herangehensweise wenig zweckdienlich. Weitere Vergleiche werden nicht vorgenommen. In ihrer Zusammenfassung geben die Autoren selbst zu, dass sie dem Titel ihrer Studie dahingehend nicht gerecht werden, dass sie keine Ausfallvorhersage betreiben, sondern lediglich vorliegende Ausfälle als Anomalien detektieren.

Nur zwei bzw. drei Deep-Learning-basierte Ansätze zur (Teil-)Problem-übergreifenden, **überwachten**, **diskriminativen Anomaliedetektion** mittels **partiellem Parametertransfer** konnten im Anwendungskontext Industrie identifiziert werden. Der Klassifikator kann dabei auch als nicht-Deep-Learning-Methode ausgeführt sein:

[206] nutzt einen gemeinsamen Merkmalsextrakteur für überwachte, diskriminative Anomaliedetektion im Sinne einer Ein-Klassen-Klassifikation. Der beschriebene Prototyp verwendet eine Kombination aus CNN und FCNN als Merkmalsextrakteur und verschiedene nicht-Deep-Learning-Methoden als Klassifikator. Eine Besonderheit des Ansatzes ist die vorgeschaltete Umwandlung von Zeitreihen in Bilddaten, genauer in sogenannte Mel-Spektrogramme, um die hohe Merkmalsextraktionsleistung von CNN auf Bilddaten zu nutzen. Eine Evaluation auf einem industriellen univariaten Zeitreihen-Datensatz belegt die höhere Genauigkeit des auf beschriebenen Prototypen verglichen mit einem einfachen und einem komplexen Autoencoder-basierten Ansatz ohne Merkmalsextraktion. Als problematisch wird jedoch die starke, bisher nicht näher untersuchte Abhängigkeit der Genauigkeit von Maschinenart und -modell beschrieben. Ein direkter Vergleich der Klassifikationsgenauigkeit mit und ohne Transfer-Funktionalität wird nicht angestellt, sodass hier keine Aussage zur Transfer-Leistung möglich ist. Darüber hinaus ist die Eigenbezeichnung als „unüberwacht“ verwunderlich, da zum Training ausschließlich normale Proben verwendet werden, sodass das Training als mit gelabelten Daten durchgeführt angesehen muss. [208] nutzt einen gemeinsamen Merkmalsextrakteur für überwachte, diskriminative Anomaliedetektion im

Sinne sowohl einer Ein- als auch einer Mehr-Klassen-Klassifikation. Der beschriebene Prototyp verwendet CNN als Merkmalsextrakteur und FCNN als Klassifikator. Eine Evaluation auf verschiedenen Internet-der-Dinge-Metadatensätzen belegt die sehr hohe Klassifikationsgenauigkeit des Algorithmus, weist jedoch hinsichtlich dessen Transfer-Leistung erhebliche Defizite auf: Für die Mehr-Klassen-Klassifikation wirkt sich der Transfer negativ aus, für die Ein-Klassen-Klassifikation fehlen Vergleichswerte ohne Transfer, sodass hier keine Aussage zur Transfer-Leistung möglich ist. Grundsätzlich ist problematisch, dass Quell- und Zielprobleme in weiten Teilen identisch sind (Mehr-Klassen-Klassifikation) bzw. sich in weiten Teilen lediglich durch eine Zusammenfassung mehrerer unterschiedlicher Label zu einem gemeinsamen Label unterscheiden (Ein-Klassen-Klassifikation). Unklar ist darüber hinaus, wie bei teilweise deutlich weniger als 64 Eigenschaften pro Probe ein Eingangsvektor von 64 für die CNN genutzt werden konnte.

Eine Abwandlung des Konzepts einer gemeinsamen Merkmalsextraktion stellt die **mehrköpfige Merkmalsextraktion** in [49] dar. Ziel ist die überwachte, diskriminative Anomaliedetektion im Sinne einer Ein-Klassen-Klassifikation. Die sehr ausführliche Studie beschreibt Prototypen aus Kombinationen von CNN als Merkmalsextrakteuren und verschiedenen RNN als Klassifikatoren. Pro univariater Zeitreihe in einem multivariaten Zeitreihendatensatz werden wird ein eigener Merkmalsextrakteur genutzt, der bei Veränderungen auf die Herkunft anderer, von ihm unabhängiger Bestandteile des Datensatzes unverändert beibehalten wird. Eine Evaluation auf einem industriellen, multivariaten Zeitreihendatensatz belegt in machen Transferszenarien einen deutlich positiven Transfer bei stark reduzierter Trainingszeit.

In der Literatur können drei Deep-Learning-basierte Ansätze zur (Teil-)Problem-übergreifenden, **generativen Anomaliedetektion** mittels **vollständigem Parametertransfer** im Anwendungskontext Industrie identifiziert werden:

[204] nutzt Feinabstimmung für unüberwachte, generative Anomaliedetektion. Der beschriebene Prototyp verwendet den Rekonstruktionsfehler eines mehrschichtigen, LSTM-basierten Autoencoder als Prädiktor. Nach einem Vortraining des Algorithmus auf einem Quelldatensatz findet eine Feinabstimmung der jeweils letzten Schicht des Kodierers und Dekodierers auf dem Zieldatensatz statt. Eine Evaluation auf einem industriellen, multivariaten Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem, wobei die Komplexität des Datensatzes unrealistisch gering ist. [205] nutzt ebenfalls Feinabstimmung für überwachte, generative Anomaliedetektion. Der beschriebene Prototyp verwendet den Rekonstruktionsfehler eines SDA als Prädiktor. Angaben zur Art der verwendeten Knoten fehlen. Nach einem unüberwachten Vortraining des Algorithmus auf einem Quelldatensatz findet eine überwachte Feinabstimmung auf dem Zieldatensatz statt. Eine Evaluation auf einem industriellen, univariaten Zeitreihen-Datensatz belegt den positiven Transfer in Bezug auf die Zeitreihenvorhersage zwischen Quell- und Zielproblem, wobei unklar bleibt, ob der Vergleichsalgorithmus wirklich keine weiteren Unterschiede als

nur die fehlende Transfer-Funktionalität aufweist. Eine ausführliche Analyse und Diskussion hinsichtlich der Anomaliedetektionsleistung erfolgen ebenfalls nicht. [207] nutzt Parametertransfer für semi-überwachte, generative Anomaliedetektion. Ziel des Transfers ist die Reduzierung des Aufwands für das Training von Anomalie-Detektor-Algorithmen für potenziell tausende mehr oder weniger unterschiedliche, jeweils univariate Sensor-Kanäle. Der beschriebene Prototyp verwendet den Rekonstruktionsfehler einer Kombination aus LSTM und FCNN als Prädiktor. Eine Evaluation auf zwei univariaten Zeitreihen-Datensätzen aus der Raumfahrt belegt bei minimal negativer Transfer-Leistung eine deutliche Reduktion der Trainingsdauer. Dabei wurde sowohl partieller (gemeinsamer Merkmalsextrakteur) als auch vollständiger (Feinabstimmung) Parametertransfer untersucht.

Lediglich ein Deep-Learning-basierter Ansatz lässt sich zur (Teil-)Problem-übergreifenden, überwachten, **diskriminativen Anomaliedetektion** im Sinne einer Mehr-Klassen-Klassifikation im Anwendungskontext Industrie finden. Er nutzt **Regularisierungs-basiertes kontinuierliches Lernen**:

[63] verwendet eine Kombination aus LSTM und FCNN als Klassifikator. Eine Evaluation auf einem industriellen univariaten Zeitreihen-Datensatz belegt den positiven, auch mehrfachen Transfer zwischen Quell- und Zielproblemen. Als problematisch wird jedoch die starke, bisher nicht näher untersuchte Abhängigkeit der Transfer-Leistung von Ähnlichkeit, Richtung, Reihenfolge und Anzahl von Quell- und Zielproblemen beschrieben.

3.2.1.4 Bewertung der Ansätze

Die vorgestellten Ansätze entstammen allesamt den Lösungskategorien des Merkmalsrepräsentations- und Parametertransfers des Transfer-Lernens bzw. den Regularisierungs-Strategien des kontinuierlichen Lernens, die jedoch methodisch eine Form des Parametertransfers darstellen (siehe Kapitel 2.2.2). Zum Einsatz kommen verschiedene Deep-Learning-Methoden, von einfachen FCNN über LSTM bis hin zu Autoencodern oder komplexen, vortrainierten CNN wie AlexNet oder ResNet.

Hinsichtlich der Erfüllung der in Kapitel 1.4 formulierten Anforderungen durch die vorgestellten Ansätze lassen sich folgende Bewertungen vornehmen:

Theoretische Aussagen zu verbesserter Robustheit im Sinne von **Anforderung 1** trifft nur [207]. Inwieweit der beschriebene, generalisierte und vortrainierte Prädiktor diese Anforderung tatsächlich erfüllt, ist jedoch nicht Gegenstand der Studie.

Hinsichtlich des eigentlichen Wissens-Transfers im Sinne von **Anforderung 2** muss festgestellt werden, dass die Ergebnisse einiger Ansätze nicht sauber genug dokumentiert sind, um eine ein-

deutige Bewertung der Leistungsfähigkeit der Ansätze zuzulassen. Dies kann in mangelnden Vergleichswerten [205, 206, 208, 209] oder auch unzureichend komplexen Anwendungsfällen [204] begründet liegen. Da Anforderung 2 eigentlich im Fokus aller vorgestellten Studien liegt, werden diese Lücken nicht als „keine Erfüllung“, sondern als „nicht beantwortbar“ gewertet.

Als Eingangsdatenart werden in allen Studien Industrie-typisch primär Zeitreihendaten genutzt. Nur vereinzelt werden diese zuerst in Bilddaten transformiert oder direkt Bild- oder Metadaten genutzt. Multivariate Zeitreihen verarbeiten nur ein Teil der vorgestellten Ansätze und kein Ansatz nutzt parallel unterschiedliche Datenarten. Lediglich [49] und in eingeschränkter Form auch [45] gehen zumindest theoretisch auf diese Möglichkeit ein und gewährleisten somit eine gewisse Wiederverwendbarkeit von Daten im Sinne von **Anforderung 3**.

Die meisten vorgestellten Studien nutzen nur einen einzigen Datensatz zur Evaluation. Aussagen zur Übertragbarkeit im Sinne von **Anforderung 4** sind damit für die darin präsentierten Algorithmen nur sehr eingeschränkt möglich. Lediglich [207–209] evaluieren ihren Algorithmus auf verschiedenen, ähnlichen Datensätzen und nur [45] greift auf Evaluationsdatensätze mit unterschiedlichen Datenarten zurück.

Tabelle 3.3: Analyse der Ansätze für (Teil-)Problem-übergreifendem Deep-Learning im Anwendungsfall Anomaliedetektion hinsichtlich der Erfüllung der Anforderungen

Quelle	A1	A2	A3	A4
Canizo et al., 2019 [49]	○	◕	◕	◔
Hsieh et al., 2019 [204]	○	?	○	◔
Liang et al., 2019 [205]	○	?	○	◔
Müller et al., 2020 [206]	○	?	○	◔
Xu et al., 2020 [48]	○	◕	○	◔
Baireddy et al., 2021 [207]	◔	◕	○	◕
Maschler et al., 2021 [63]	○	◕	○	◔
Michau et al., 2021 [45]	○	◔	◔	◕
Ullah et al., 2021 [208]	○	?	○	◕
Wang et al., 2021 [209]	○	?	○	◕

Legende: ? = nicht beantwortbar; ○ = keine Erfüllung; ◔ = geringe Erfüllung; ◕ = mittlere Erfüllung; ● = hohe Erfüllung

Tabelle 3.3 gibt einen Überblick über die Erfüllung der in Kapitel 1.4 definierten Anforderungen durch die vorgestellten Ansätze. Es wird ersichtlich, dass keiner der untersuchten Ansätze für (Teil-)Problem-übergreifendes Deep Learning im Anwendungsfall Anomaliedetektion die im Rahmen dieser Arbeit gestellten, über Partikularlösungen hinausgehenden Anforderungen erfüllt.

Für eine tiefergehende Analyse hinsichtlich der daraus zu ziehenden Schlüsse sei auf Kapitel 3.4 verwiesen.

Für diese Arbeit lassen sich aus den vorgestellten Studien insbesondere die folgenden Schlüsse ziehen: Mehrköpfigkeit ist ein vielversprechender Ansatz für industrielles Transfer-Lernen [49] und eine leicht negative Transfer-Leistung kann mit deutlichen Vorteilen hinsichtlich Trainingszeit oder benötigter Trainingsdatenmenge einhergehen [207]. Außerdem zeigt [48], dass eine geeignete Auswahl der im Transfer zu nutzenden Quelldaten die Transfer-Leistung erhöht, während [45] deutliche Einschränkungen in der Nutzbarkeit der Domänen-Adaption durch den Bedarf an gleichgroßen Datensätzen von Quell- und Zielproblem sieht. Regularisierungs-Ansätze erscheinen aufgrund ihrer komplexen Abhängigkeiten nicht geeignet [63].

3.2.2 Anwendungsfall Ausfallvorhersage

3.2.2.1 Überblick über den Anwendungsfall

Unter einem Ausfall wird das Erreichen eines Zustands der Dysfunktionalität verstanden. Bezogen auf industrielle Komponenten kann damit der Ausfall weiterer Komponenten bzw. übergeordneter Systeme einhergehen. Üblicherweise unterscheidet man vier verschiedene Ursachen für Ausfälle:

Abnutzung beschreibt den zumeist mechanischen Verlust von Material über den Nutzungszeitraum. **Korrosion** beschreibt den chemischen Verlust von Material über den Nutzungszeitraum. **Bruch** beschreibt die physische Trennung verschiedener, vormals ungetrennter Komponenten. **Verformung** beschreibt die Veränderung der Form einer oder mehrerer Komponenten. Grundsätzlich können die Ursachen auch in Kombination auftreten, bspw. indem Korrosion eine chemische Umwandlung von Werkstoffen bewirkt, die eine mechanische Abnutzung erleichtert und schließlich zum Bruch führt [211].

Aufgrund der mit ungeplanten Ausfällen verbundenen hohen Kosten ist die Ausfallvorhersage ein wichtiges Forschungsfeld. Ihr Gegenstand ist die Vorhersage des Zeitpunkts eines Ausfalls – üblicherweise ohne weitere Berücksichtigung von dessen Ursache. Ihr Ziel ist unter anderem die Ermöglichung proaktiver Instandhaltung, die Erhöhung der Betriebssicherheit sowie die Reduzierung von Ausfall(-folge-)kosten [211–214].

Üblicherweise unterscheidet man drei verschiedene Ansätze für die Ausfallvorhersage:

Modell-basierte Ansätze weiter unterteilt in Physik- und Experten-basierte Ansätze [212], beschreiben Verschleißprozesse mittels mathematischer Modelle und ausgehend von den Ursachen von Ausfällen sowie den sie beeinflussenden Faktoren. Derartige Ansätze sind sehr effizient und genau, setzen jedoch ein möglichst vollständiges Systemverständnis voraus. Die Anpassung an

neue bzw. veränderte Szenarien muss zumeist manuell erfolgen und lohnt sich daher nur für statische und teure Szenarien [213, 214]. **Daten-basierte Ansätze**, weiter unterteilt in numerische bzw. statistische und Maschinelles-Lernen-basierte Ansätze [212], beschreiben Verschleißprozesse anhand historischer Daten, die entweder aus den realen Anlagen, aus Testaufbauten oder Simulationen gewonnen werden können. Derartige Ansätze sind in der Entwicklung meist kostengünstig, lassen sich einfach anpassen und setzen kein bzw. nur geringes Systemverständnis voraus. Demgegenüber stehen ihre hohen Anforderungen an Vielfalt, Menge und Qualität der Daten sowie der teilweise hohe Trainingsaufwand [213, 214]. **Hybride Ansätze** vereinigen Modell- und Daten-basierte Methoden mit dem Ziel, die Qualität Daten-basierter Ansätze zu steigern ohne den hohen Aufwand ausschließlich Modell-basierter Ansätze zu generieren [213, 214].

3.2.2.2 Allgemeines Deep Learning

Verschiedene Deep-Learning-Methoden können zur Ausfallvorhersage eingesetzt werden. Unterschieden werden dabei grundsätzlich zwei unterschiedliche Herangehensweisen:

Die **Vorhersage der Restlebensdauer** (engl.: Remaining Useful Lifetime, kurz: RUL) als kontinuierlicher Prozentangabe bezogen auf die Gesamtlebensdauer ist der übliche Ansatz zur Ausfallvorhersage [1, 31, 215, 216]. Es handelt sich dabei um ein Regressions-Problem. Zum Einsatz kommen dabei vor allem CNN, RNN, Autoencoder und Deep Belief Networks [1, 31, 214]. Vereinzelt wird zur RUL-Vorhersage auch auf die Zeitreihenvorhersage mittels LSTM zurückgegriffen [217].

Eine Alternative zur RUL-Vorhersage ist die **Bestimmung des Gesundheitszustands** (engl.: State of Health, kurz: SoH) in Form einer Mehr-Klassen-Klassifikation. Während der Begriff SoH ursprünglich aus dem Bereich der Batterie-Forschung stammt und dort ebenfalls eine kontinuierliche Prozentangabe $SoH_{\%}$ bezogen auf das Verhältnis unterschiedlicher, jeweils aktueller Leistungsparameter auf ihre jeweiligen Nennwerte war [215, 216, 218], so wird der Begriff inzwischen auch für eine diskrete Klassifizierung der Restlebensdauer $SoH_{Klassen}$ verwendet – sowohl im Batterie-Kontext [64, 219, 220], als auch darüber hinaus [158, 221–223]. Die $SoH_{Klassen}$-Bestimmung kommt primär dann zum Einsatz, wenn die RUL-Vorhersage bspw. aus methodischen Gründen oder aufgrund unzureichender Trainingsdaten nicht möglich ist. Teilweise wird auch bewusst auf die genauere RUL-Vorhersage verzichtet, weil die $SoH_{Klassen}$-Bestimmung weniger aufwändig und für den vorliegenden Anwendungsfall ausreichend ist. Vereinzelt wird der $SoH_{Klassen}$ auch als Vorstufe einer nachgeschalteten RUL-Vorhersage genutzt [224, 225].

Abbildung 3.3 stellt RUL, $SoH_{\%}$ und $SoH_{Klassen}$ in Abhängigkeit von der bereits vergangenen Lebenszeit beispielhaft dar. Es wird ersichtlich, dass $SoH_{\%}$ im Kontext des stark nicht-linearen Verhaltens von bspw. Batterie-Kapazität gegenüber dem linearen RUL vorteilhaft ist [215, 216] – in anderen Bereichen mit stärker linearem Verhalten bzw. einem Fokus auf der bis zu einem Ausfall

verbleibenden Zeit gegenüber der verbleibenden Funktionalität bringt es jedoch keinen Vorteil. Im weiteren Verlauf dieser Arbeit wird SoH daher im Sinne von $SoH_{Klassen}$ verwendet.

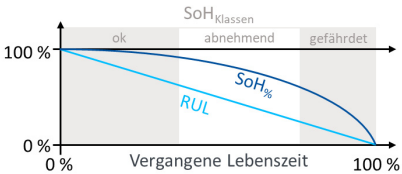

Abbildung 3.3: Beispielhafte Darstellung von RUL, $SoH_{\%}$ und $SoH_{Klassen}$ in Abhängigkeit von der bereits vergangenen Lebenszeit

Obwohl Deep-Learning-basierte Methoden der Ausfallvorhersage in der Literatur gute Ergebnisse erzielen, besteht weiterhin erheblicher Forschungsbedarf. Die vorgestellten Ansätze sind in aller Regel Anwendungsfall-spezifisch, Betrachtungen der Übertragbarkeit fehlen. Dies kann auch der geringen Verbreitung frei verfügbarer, Benchmarking-geeigneter Datensätze geschuldet sein [1, 31].

3.2.2.3 (Teil-)Problem-übergreifendes Deep Learning

Methodik

Zur Identifizierung relevanter Publikationen für (Teil-)Problem-übergreifenden Deep Learning im Anwendungsfall Ausfallvorhersage wurde eine zweistufige, systematische Literaturstudie durchgeführt:

Im ersten Schritt wurde auf Google Scholar wissenschaftliche Veröffentlichungen identifiziert, die den in Tabelle 3.4 aufgelisteten Kombinationen von Suchbegriffen entsprechen. Für jede Suchanfrage wurde dazu jeweils ein Begriff der Kategorie „Suchbegriff 1“ und ein Begriff der Kategorie „Suchbegriff 2“ kombiniert.

Tabelle 3.4: Suchbegriffe für die erste Stufe der systematischen Literaturstudie zum (Teil-)Problem-übergreifenden Deep Learning im Anwendungsfall Ausfallvorhersage

Suchbegriff 1	Suchbegriff 2	
Transfer Learning	Fault Prognostics	Fault Prognosis
Continual Learning	Remaining Useful Lifetime	State of Health

Im zweiten Schritt wurden die identifizierten wissenschaftlichen Veröffentlichungen manuell gefiltert. Nur englischsprachige[5], im Volltext vorliegende Forschungsbeiträge, die Deep-Learning-basierte Methoden und eine Form von Transfer- oder kontinuierlichem Lernen im Anwendungsfall Ausfallvorhersage einsetzen und bis einschließlich 2021 erschienen sind, werden im Folgenden berücksichtigt.

Ergebnisse

Tabelle 3.5 gibt einen Überblick über die aus der zuvor beschriebenen systematischen Literaturstudie hervorgegangenen Ansätze (Teil-)Problem-übergreifenden Deep Learnings im Anwendungsfall Ausfallvorhersage.

Tabelle 3.5: Beispiele für die Nutzung von (Teil-)Problem-übergreifendem Deep-Learning im Anwendungsfall Ausfallvorhersage

Quelle	Lern-Kategorie	Problem-Kategorie	Datenart	Transfer-Methode
Zhang et al., 2018 [226]	Überwacht	RUL	Multivar. Zeitreihen	Feinabstimmung
Sun et al., 2019 [227]	Unüberwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Da Costa et al., 2020 [156]	Unüberwacht	RUL	Multivar. Zeitreihen	Domänen-Adaption
Maschler et al., 2020 [158]	Überwacht	SoH	Multivar. Zeitreihen	Regularisierung
Ragab et al., 2020 [228]	Unüberwacht	RUL	Multivar. Zeitreihen	Domänen-Adaption
Russell et al., 2020 [44]	Semi-überwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Tan et al., 2020 [229]	Überwacht	RUL	Merkmale	Gemeinsamer Merkmalsextrakteur plus Feinabstimmung
Zhang et al., 2020 [230]	Überwacht	RUL	Bilder	Feinabstimmung

[5] Aufgrund der dominierenden Rolle des Englischen in der wissenschaftlichen Kommunikation im Bereich der KI-Forschung wurde auf die Einbeziehung deutsch- oder anderssprachiger Veröffentlichungen verzichtet. Daher sind auch die Suchbegriffe englisch.

Quelle	Lern-Kategorie	Problem-Kategorie	Datenart	Transfer-Methode
Cao et al., 2021 [225]	Unüberwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Cheng et al., 2021 [231]	Unüberwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Cheng et al., 2021 [232]	Unüberwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Ding et al., 2021 [233]	Unüberwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Gribbestad et al., 2021 [234]	Überwacht	RUL	Multivar. Zeitreihen	Parametertransfer
Marei et al., 2021 [235]	Überwacht	RUL (indirekt)	Bilder	Gemeinsamer Merkmalsextrakteur plus Feinabstimmung
Maschler et al., 2021 [64]	Überwacht	SoH	Merkmale	Regularisierung
Zeng et al., 2021 [236]	Unüberwacht	RUL	Univar. Zeitreihen	Domänen-Adaption
Zhang et al., 2021 [237]	Überwacht	RUL	Zeitreihen	Domänen-Adaption

In den letzten Jahren wurde eine Vielzahl von Deep-Learning-basierten Ansätzen zur (Teil-)Problem-übergreifenden **RUL-Vorhersage** mittels **Domänen-Adaption** im Anwendungskontext Industrie veröffentlicht:

Von diesen Domänen-Adaptions-Ansätzen beschäftigen sich lediglich zwei mit **semi-überwachtem** Lernen. [228] verwendet dabei eine Kombination aus LSTM als Merkmalsextrakteur, einem nicht näher spezifiziertem neuronalen Netzwerk als Diskriminator und FCNN als Regressor. Eine Evaluation auf einem multivariaten, industriellen Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem, wobei der Ansatz ohne Transfer-Funktionalität lediglich auf dem Quellproblem trainiert wurde, da für das Zielproblem keine Label zum Training notwendig sein sollen. Vergleiche mit anderen Domänen-Adaptions-Algorithmen aus [156] ergaben zudem eine bessere Leistungsfähigkeit des vorgestellten Algorithmus. Der in [44] beschriebene Prototyp verwendet eine Kombination aus CNN als Merkmalsextrakteur, FCNN als Diskriminator und FCNN als Regressor. Eine Evaluation auf einem multivariaten, industriellen Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem insbesondere bei

nur geringer Anzahl Zielproben. Da jedoch nur ein einzelnes Transfer-Szenario betrachtet wird, ist die Aussagekraft der Studie begrenzt.

Die meisten Domänen-Adaptions-Ansätze beschäftigen sich folglich mit **unüberwachtem** Lernen. [227] nutzt dazu eine Kombination aus SDA als Domänen-Adapter und FCNN als Regressor, wobei nur der Merkmalsextrakteur an das Zielproblem angepasst wird und der Regressor unverändert bleibt. Somit ist eine unüberwachte Anpassung des überwacht vortrainierten Algorithmus an das Zielproblem möglich. Eine Evaluation auf einem univariaten, industriellen Zeitreihen-Datensatz bestehend aus Vibrationsdaten belegt den positiven Transfer zwischen Quell- und Zielproblem. Der in [156] beschriebene Prototyp verwendet eine Kombination aus LSTM als Merkmalsextrakteur und FCNN als Regressor und Diskriminator. Eine Evaluation auf einem multivariaten, industriellen Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem, wobei für jedes Transfer-Szenario eine eigene Hyperparameteroptimierung durchgeführt wird. Ein Vergleich mit anderen unüberwachten Domänen-Adaptions-Algorithmen (u.a. [177]) ergibt, dass der vorgestellte Algorithmus eine höhere Genauigkeit erzielt, ein Vergleich mit dem überwachten Ansatz nach [226] fällt zu deren Gunsten aus. Zusätzlich wird die alternative Nutzung von FCNN, CNN oder RNN als Merkmalsextrakteur untersucht, wobei auf LSTM-Basis insgesamt die besten Ergebnisse erzielt werden. [225] verwendet demgegenüber eine Kombination aus FCNN und bidirektionalen geschlossenen, rekurrenten Einheiten (engl.: Gated Recurrent Unit, kurz: GRU) als Regressor. Diesen vorgeschaltet ist eine Merkmalsextraktion, die aus den Zeitreihensignalen zuvor definierte Merkmale generiert, die vor weiterer Nutzung hinsichtlich ihrer Domänen-Invarianz geprüft werden. Eine Evaluation auf einem univariaten, industriellen Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem. Ein Vergleich mit anderen Algorithmen (u.a. [177, 231]) ergibt, dass der vorgestellte Algorithmus eine höhere Genauigkeit erzielt und insbesondere die besondere Merkmalsextraktion dazu beitragen soll. [231] beschreibt eine Kombination aus CNN und FCNN als Merkmalsextrakteur, Multi-Kernel MMD als Adaptions-Zielfunktion und FCNN als Regressor. Auf einem multivariaten, aber univariat genutzten industriellen Zeitreihen-Datensatz wird eine umfangreiche Evaluation durchgeführt: Mittels einer Betrachtung nur des beschriebenen Prototyps wird die optimale Nutzung des Diskriminators untersucht. Es zeigt sich, dass die MMD auf möglichst weit extrahierten Merkmalen, d.h. möglichst auf dem Ergebnis der Merkmalsextraktion, bestimmt werden sollte. Mittels eines Vergleichs mit Lernalgorithmen derselben Architektur aber ohne Transfer-Funktionalität und trainiert auf gelabelten Quell-, Ziel- oder Quell- und Zielproben wird gezeigt, dass der vorgestellte Algorithmus einen positiven Transfer bewirkt und zudem besser generalisiert. Ein abschließender Vergleich mit anderen Algorithmen, die jedoch nur vereinzelt mehrschichtiges Transfer-Lernen nutzen, ergibt, dass der vorgestellte Algorithmus eine höhere Genauigkeit erzielt. Die Autoren von [231] präsentieren in [232] zwei weitere Ansätze zur unüberwachten RUL-Vorhersage mittels Domänen-Adaption – einen adversarischen Ansatz und einen nicht-adversari-

schen. Beide beschriebenen Prototypen verwenden eine Kombination aus CNN als Merkmalsextrakteur und FCNN als Regressor. Der nicht-adversarische Ansatz basiert auf einer erweiterten Verlustfunktion, die die marginale Wahrscheinlichkeitsverteilung mittels Multi-Kernel MMD und die bedingte Wahrscheinlichkeitsverteilung auf Basis einer Fuzzy-Klasseneinteilung (analog zu $SoH_{Klassen}$) in Kombination mit MMD berücksichtigt. Der adversarische Ansatz nutzt einen modularen Diskriminator für die marginale und bedingte Wahrscheinlichkeitsverteilung, wobei auf deren Aufbau nicht näher eingegangen wird. Im Rahmen einer Quellproben-Auswahl basierend auf umgekehrter Validierung werden vor Beginn des eigentlichen Adaptionsvorgangs geeignete Quellproben identifiziert. Auf zwei multivariaten, aber univariat genutzten industriellen Zeitreihen-Datensatz wird eine umfangreiche Evaluation durchgeführt: Mittels Vergleiche mit Lernalgorithmen verschiedener Architekturen mit und ohne Transfer-Funktionalität und unterschiedlichen Trainings-Strategien wird gezeigt, dass beide vorgestellten Algorithmen sowohl einen positiven Transfer bewirken als auch (vielfach deutlich) besser funktionieren. Auf dem einen Datensatz liegt dabei der nicht-adversarische Ansatz vorne, auf dem anderen der adversarische. Der in [233] beschriebene Prototyp verwendet eine Kombination aus FCNN als Merkmalsextrakteur, Multi-Kernel MMD als Adaptions-Zielfunktion und Kernel-Regression als Regressor. Eine Evaluation auf einem univariaten, industriellen Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem, wobei die Vorhersagegenauigkeit insgesamt eher gering ist. Ein Vergleich mit anderen Algorithmen (u. a. [156] und [231]), ergibt, dass der vorgestellte Algorithmus eine höhere Genauigkeit erzielt. Es bleibt jedoch unklar, auf welchen Zahlen dieser Vergleich konkret beruht – auch die für den eigenen Ansatz angegebenen Durchschnittswerte lassen sich aus der Publikation nicht entnehmen. [236] beschreibt eine Kombination aus CNN als Merkmalsextrakteur, FCNN als Diskriminator und FCNN als Regressor. Eine Evaluation auf einem univariaten, industriellen Zeitreihen-Datensatz belegt die bessere Vorhersagequalität des Algorithmus verglichen mit verschiedenen anderen Algorithmen (u. a. [231]). Auch wenn zu diesen anderen Algorithmen einer ohne Transfer-Funktionalität gehört, so ist eine direkte Beurteilung der Transfer-Leistung aufgrund dessen anderer Architektur nicht möglich. Stattdessen wird eine Untersuchung der Auswirkung unterschiedlicher Lernraten und Kernel-Größen vorgenommen. [237] verwendet schließlich eine Kombination aus CNN und FCNN als Merkmalsextrakteur und FCNN als Regressor. Über Erweiterungen der Verlustfunktionen für „gesunde“ und Ausfall-gefährdete Proben (ähnlich SoH-Klassen) wird für diese beiden Kategorien eine separate Domänen-Adaption durchgeführt. Eine Evaluation auf zwei industriellen Zeitreihen-Datensätzen – einer univariat, einer multivariat – belegt den positiven Transfer zwischen Quell- und Zielproblemen. Dabei wird sowohl der Einfluss der Transfer-Funktionalität bei einer konventionellen Domänen-Adaption als auch bei der vorgeschlagenen Domänen-Adaption mit einer separaten Behandlung verschiedener SoH-Klassen untersucht, wobei der vorgeschlagene Ansatz die besten Ergebnisse erzielt, allerdings auch eine längere Trainingszeit benötigt.

In der Literatur können drei Deep-Learning-basierte Ansätze zur (Teil-)Problem-übergreifenden, überwachten **RUL-Vorhersage** mittels **Feinabstimmung** im Anwendungskontext Industrie identifiziert werden:

Der in [226] beschriebene Prototyp verwendet eine Kombination aus bidirektionalen LSTM als Merkmalsextrakteur und FCNN als Regressor. Eine Evaluation auf einem multivariaten, industriellen Zeitreihen-Datensatz belegt den positiven Transfer zwischen Quell- und Zielproblem. Es wird festgestellt, dass auch bei größeren Unterschieden zwischen Quell- und Zielproblemen bspw. bezüglich der Anzahl Betriebsbedingungen oder Fehlerzustände der Transfer zumeist positiv ausfiel. [230] verwendet eine Kombination aus vortrainiertem CNN als Merkmalsextrakteur und eigenem, bidirektionalem LSTM und FCNN als Regressor. Eine Evaluation auf zwei univariaten, industriellen Zeitreihen-Datensätzen, die jedoch in Bilder konvertiert werden, belegt den positiven Transfer zwischen (generischem) Quell- und Zielproblem. Darüber hinaus ist die Trainingszeit mit Transfer-Funktionalität kleiner als ohne. Ein Vergleich mit anderen Algorithmen ergibt, dass der vorgestellte Algorithmus eine höhere Genauigkeit erzielt. Es bleibt jedoch unklar, warum der Merkmalsextrakteur nicht stärker an den vorliegenden Anwendungsfall angepasst wird – so wird dasselbe Bild dreimal verarbeitet, weil der genutzte Algorithmus drei Eingangskanäle aufweist. [234] nutzt verschiedene Formen des Parametertransfers zur überwachten RUL-Vorhersage. Basierend auf einer Untersuchung verschiedener Deep-Learning-Methoden ohne Transfer-Funktionalität wird ein Prototyp unter Verwendung einer Kombination aus LSTM und FCNN als Regressor beschrieben. Eine Evaluation auf zwei multivariaten, industriellen Zeitreihen-Datensätzen belegt den positiven Transfer zwischen Quell- und Zielproblem. Dabei wurden verschiedene Parametertransfers sowohl von im Sinne einer Feinabstimmung weiter trainierbaren als auch von nicht weiter trainierbaren Teil-Algorithmen untersucht.

In der Literatur können nur zwei Deep-Learning-basierte Ansätze zur (Teil-)Problem-übergreifenden **RUL-Vorhersage** unter Nutzung eines **gemeinsamen Merkmalsextrakteurs mit Feinabstimmung** im Anwendungskontext Industrie identifiziert werden:

Der in [229] beschriebene Prototyp verwendet eine Kombination aus LSTM als Merkmalsextrakteur und FCNN als Regressor zur überwachten RUL-Vorhersage[6], wobei der Merkmalsextrakteur unverändert bleibt und nur der Regressor ggf. an das Zielproblem angepasst wird. Diese Anpassung ist abhängig vom Ergebnis der Grau-Verhältnis-Analyse (engl.: Gray Relational Analysis) [238] der Merkmale von Quell- und Zieldatensatz. Eine Evaluation auf zwei Batterieverschleiß-Merkmals-Datensätzen belegt den positiven Transfer zwischen Quell- und Zielproblemen insbesondere bezogen auf den Zeitraum kurz vor Ausfalleintritt. Darüber hinaus ist die Trainingszeit

[6] Tatsächlich wird $SoH_{\%}$ vorhergesagt. Aufgrund der Ähnlichkeit zwischen RUL und $SoH_{\%}$ wird hier auf eine Unterscheidung verzichtet (siehe auch Kapitel 3.2.2.2).

mit Transfer-Funktionalität kleiner als ohne. Ein Vergleich mit anderen Algorithmen ergibt, dass der vorgestellte Algorithmus eine höhere Genauigkeit erzielt als andere Deep-Learning-basierte Algorithmen aber eine niedrigere als andere nicht-Deep-Learning-basierte Algorithmen. Der in [235] beschriebene Ansatz basiert auf einer um die Reduzierung des MMD zwischen den Wahrscheinlichkeitsverteilungen von Quell- und Zielproblem erweiterten Verlustfunktion zur überwachten, indirekten RUL-Vorhersage. Unmittelbar ermittelt wird die Abnutzung der Schnittkante von Schneidwerkzeugen, die einen direkten Rückschluss auf die RUL erlauben soll. Der beschriebene Prototyp verwendet eine Kombination aus vortrainiertem CNN als Merkmalsextrakteur und eigenem FCNN als Regressor, wobei nur der Regressor an das Zielproblem angepasst wird und der Merkmalsextrakteur unverändert bleibt. Eine Evaluation auf einem industriellen Bild-Datensatz belegt die hohe Genauigkeit des Algorithmus, bietet jedoch keinerlei Vergleichswerte. Somit ist weder eine Bewertung der absoluten Genauigkeit noch der Transfer-Leistung möglich.

Wiederum nur zwei Ansätze lassen sich zur (Teil-)Problem-übergreifenden, diskreten **SoH-Bestimmung** mittels Deep Learning im Anwendungskontext Industrie finden. Sie wenden allesamt **Regularisierungs-basiertes kontinuierliches Lernen** an:

Der in [158] beschriebene Prototyp verwendet eine Kombination aus LSTM und FCNN als Klassifikator zur überwachten Ausfallvorhersage. Eine Evaluation auf einem multivariaten, industriellen Zeitreihen-Datensatz belegt den positiven, auch mehrfachen Transfer zwischen Quell- und Zielproblemen. Dabei wird eine starke Abhängigkeit der Transfer-Leistung von der Ähnlichkeit von Quell- und Zielproblemen beschrieben. [64] erweitert diesen Ansatz und nutzt eine ähnliche Architektur in einer anderen Anwendungsdomäne. Die Evaluation auf einem Batterieverschleiß-Merkmals-Datensatz belegt auch hier den positiven, mehrfachen Transfer zwischen Quell- und Zielproblemen. Als problematisch wird jedoch die starke, bisher nicht näher untersuchte Abhängigkeit der Transfer-Leistung von Ähnlichkeit, Richtung, Reihenfolge und Anzahl von Quell- und Zielproblemen beschrieben.

3.2.2.4 Bewertung der Ansätze

Die vorgestellten Ansätze entstammen allesamt den Lösungskategorien des Merkmalsrepräsentations- und Parametertransfers des Transfer-Lernens bzw. den Regularisierungs-Strategien des kontinuierlichen Lernens, die jedoch methodisch eine Form des Parametertransfers darstellen (siehe Kapitel 2.2.2). Auch im Anwendungsfall Ausfallvorhersage kommen verschiedene Deep-Learning-Methoden zum Einsatz, von einfachen FCNN über RNN verschiedener Form bis hin zu Autoencodern oder komplexen, vortrainierten CNN wie AlexNet oder ResNet.

Nur zwei der vorgestellten Studien behandeln die SoH-Bestimmung [64, 158], die zusätzlich ohne Deep Learning auch in [223] und ohne Transfer-Lernen auch in [221] zum Einsatz kommt. Die direkte RUL-Vorhersage ist jedoch deutlich prominenter vertreten.

Hinsichtlich der Erfüllung der in Kapitel 1.4 formulierten Anforderungen durch die vorgestellten Ansätze lassen sich folgende Bewertungen vornehmen:

Konkrete Aussagen zu verbesserter Robustheit im Sinne von **Anforderung 1** treffen nur [231, 232, 234]. Alle drei beinhalten Untersuchungen zur Erhöhung der Generalisierung ihrer Algorithmen und reduzieren damit den Bedarf nach Nachtrainings.

Hinsichtlich des eigentlichen Wissens-Transfers im Sinne von **Anforderung 2** muss festgestellt werden, dass die Ergebnisse einiger Ansätze nicht sauber genug dokumentiert sind, um eine eindeutige Bewertung der Leistungsfähigkeit der Ansätze zuzulassen. Dies kann in mangelnden Vergleichswerten [44, 235, 236] oder deren unzureichender Dokumentation [233] begründet liegen. [230] wirkt aufgrund der nicht an den Anwendungsfall angepassten, generischen Eingangsdatenbehandlung unfertig.

Als Eingangsdatenart werden Industrie-typisch primär Zeitreihendaten genutzt. Nur vereinzelt werden direkt Merkmale, Bild- oder Metadaten genutzt. Multivariate Zeitreihen verarbeiten nur ein Teil der vorgestellten Ansätze und kein Ansatz nutzt parallel unterschiedliche Datenarten. Aussagen zur Wiederverwendbarkeit von Daten im Sinne von **Anforderung 3** sind damit nur sehr eingeschränkt möglich.

Die meisten vorgestellten Studien nutzen nur einen einzigen Datensatz zur Evaluation. Aussagen zur Übertragbarkeit im Sinne von **Anforderung 4** sind damit für die darin präsentierten Algorithmen nur sehr eingeschränkt möglich. Lediglich [229, 230, 232, 234, 237] evaluieren ihren Algorithmus auf verschiedenen, ähnlichen Datensätzen und keine Studie greift auf Evaluationsdatensätze mit unterschiedlichen Datenarten zurück.

Tabelle 3.6 gibt einen Überblick über die Erfüllung der in Kapitel 1.4 definierten Anforderungen durch die vorgestellten Ansätze. Es wird ersichtlich, dass keiner der untersuchten Ansätze für (Teil-)Problem-übergreifendes Deep Learning im Anwendungsfall Ausfallvorhersage die im Rahmen dieser Arbeit gestellten, über Partikularlösungen hinausgehenden Anforderungen erfüllt. Für eine tiefergehende Analyse hinsichtlich der daraus zu ziehenden Schlüsse sei auf Kapitel 3.4 verwiesen.

Für diese Arbeit lassen sich aus den vorgestellten Studien insbesondere die folgenden Schlussfolgerungen ziehen: Eine geeignete Auswahl der im Transfer zu nutzenden Quelldaten erhöht die Transfer-Leistung [225, 232]. Außerdem zeigt [237], dass eine separate Behandlung verschiedener Proben-Cluster die Transfer-Leistung erhöht. Regularisierungs-Ansätze erscheinen aufgrund ihrer komplexen Abhängigkeiten nicht geeignet [64, 158].

Tabelle 3.6: Analyse der Ansätze für (Teil-)Problem-übergreifendem Deep-Learning im Anwendungsfall Ausfallvorhersage hinsichtlich der Erfüllung der Anforderungen

Quelle	A1	A2	A3	A4
Zhang et al., 2018 [226]	○	◔	○	◔
Sun et al., 2019 [227]	○	◔	○	◔
Da Costa et al., 2020 [156]	○	◔	○	◔
Maschler et al., 2020 [158]	○	◔	○	◔
Ragab et al., 2020 [228]	○	◔	○	◔
Russell et al., 2020 [44]	○	?	○	◔
Tan et al., 2020 [229]	○	◔	○	◕
Zhang et al., 2020 [230]	○	◔	○	◕
Cao et al., 2021 [225]	○	◕	○	◔
Cheng et al., 2021 [231]	◕	◕	○	◔
Cheng et al., 2021 [232]	◕	●	○	◕
Ding et al., 2021 [233]	○	?	○	◔
Gribbestad et al., 2021 [234]	◕	◕	○	◕
Marei et al., 2021 [235]	○	?	○	◔
Maschler et al., 2021 [64]	○	◔	○	◔
Zeng et al., 2021 [236]	○	?	○	◔
Zhang et al., 2021 [237]	○	◕	○	◕

Legende: ? = nicht beantwortbar; ○ = keine Erfüllung; ◔ = geringe Erfüllung; ◕ = mittlere Erfüllung; ● = hohe Erfüllung

3.3 Bewertung von Ansätzen zum Clustering vieldimensionaler Daten

Die in Kapitel 2.3.2 vorgestellten Clustering-Verfahren haben jeweils Vor- und Nachteile, die einerseits von der jeweiligen Implementierung abhängen. Zugrunde gelegt werden daher die Clustering-Verfahren in der in der verbreiteten Python-Bibliothek scikit-learn 0.23.2 [239] vorliegenden Implementierung.

Andererseits hängt die Bewertung der Clustering-Verfahren von den Anforderungen des jeweils vorliegenden Anwendungsfalls ab. Im Rahmen dieser Arbeit werden dafür die folgenden Kriterien für relevant erachtet:

Speicherkomplexität beschreibt die durch die Ausführung des Clustering-Verfahrens hervorgerufene Auslastung des Arbeitsspeichers in Abhängigkeit von der Anzahl der zu clusternden Proben. Eine geringere Speicherkomplexität ist ein Vorteil, weil so größere Datensätze bei gleichbleibendem Speicherbedarf verarbeitet werden können [112].

Rechenkomplexität beschreibt die durch die Ausführung des Clustering-Verfahrens hervorgerufene Auslastung des Prozessors in Abhängigkeit von der Anzahl der zu clusternden Proben. Die Rechenkomplexität lässt damit einen Rückschluss auf die zur Ausführung benötigte Rechenzeit zu. Eine geringere Rechenkomplexität ist ein Vorteil, weil so größere Datensätze in gleicher Zeit verarbeitet werden können [112].

Funktionsweise bezieht sich auf die theoretische Übertragbarkeit des Clustering-Resultats auf neue, bisher unbekannte Proben. Nach [240] lassen sich induktive, d.h. für den Betrachtungsraum vollständige, und transduktive, d.h. nur das Clustering der bekannten Proben erlaubende, Ansätze unterscheiden. In [241] wird von deskriptiven statt transduktiven Ansätzen gesprochen, die Definition ist jedoch vergleichbar. Die Fähigkeit zum **Online-Lernen** überträgt dieses Prinzip schließlich in die Praxis und berücksichtigt über die Induktivität des Clustering-Verfahrens hinaus auch, ob die Funktion zum nachträglichen Clustern bisher neuer, bisher unbekannter Proben von der zu nutzenden Python-Bibliothek scikit-learn [239] tatsächlich angeboten wird [112].

Aus dem Vergleich der verschiedenen Clustering-Verfahren ergeben sich die Algorithmen BIRCH und K-Means Mini-Batch als auf den ersten Blick gleichwertig und den anderen Verfahren überlegen: Beide weisen eine sehr geringe Speicher- sowie Rechenkomplexität auf und ermöglichen als induktive Clustering-Verfahren nicht nur theoretisch Online-Lernen, sondern auch in der in scikit-learn enthaltenen Implementierung. Die für das K-Means Mini-Batch-Verfahren notwendige Vorgabe einer Clusterzahl erweist sich in dieser Arbeit jedoch als Ausschlusskriterium, da langfristiges Online-Lernen auch neuer (Teil-)Probleme durchaus zu einer Veränderung der Clusterzahl führen kann. Folglich wird der BIRCH-Algorithmus als in dieser Arbeit zu verwendendes Clustering-Verfahren ausgewählt.

Tabelle 3.7 gibt einen Überblick darüber, wie die einzelnen Clustering-Verfahren in den verschiedenen Kriterien-Kategorien abschneiden. Zusätzlich ist die jeweils zu definierende Abbruchbedingung aufgeführt.

Tabelle 3.7: Bewertung von Ansätzen zum Clustering vieldimensionaler Daten

Clustering-Verfahren	Abbruchbedingung	Speicher-komplexität	Rechenkom-plexität	Funkti-onsweise	Online-Lernen
Hierarchisch					
Agglomera-tiv	Clusterzahl oder Distanzschwellen-wert	$O(n^2)$ [82]	$O(n^2)$ [136, 242]	Transd.	Nein
Trennend	Clusterzahl oder Distanzschwellen-wert	$O(n^2)$ [82]	$O(n^2)$ [136, 242]	Transd.	Nein
BIRCH	Verzweigungsfaktor und Distanzschwel-lenwert	$< O(n)$ [146]	$O(n)$ [82, 136]	Ind.	Ja
Partitionierend					
K-Means	Clusterzahl	$O(n + c)$ [136]	$O(n \cdot i \cdot c)$ [136]	Ind.	Nein
K-Means Mini-Batch	Clusterzahl	$O(b + c)$ [136]	$O(b \cdot i \cdot c)$ [149]	Ind.	Ja
Dichte-basiert					
DBSCAN	Mindestprobenzahl für Kernpunkt und Distanzschwellen-wert	$O(n^2)$ [150]	$O(n^2)$ [150]	Transd.	Nein

Legende: n = Anzahl Proben, i = Anzahl Iterationsschritte, c = Anzahl Cluster, b = Anzahl Proben in Batch

3.4 Aufzeigen des Forschungsbedarfs

In den Kapiteln 1 und 2 konnte im Rahmen der DSR-Phase „Untersuchung der Problemstellung“ herausgearbeitet werden, dass aktuelle Ansätze zur Nutzung von Deep Learning im Anwendungskontext Industrie an mangelnder Übertragbarkeit zwischen Anlagen (**H1**), über Prozessgrenzen hinweg (**H2**) und von Daten (**H3**) kranken (siehe Kapitel 1.2).

Lösungsansätze, die diese Herausforderungen überwinden, sollten über Partikularlösungen hinausgehen und sowohl Robustheit (**A1**) sowie Adaptierbarkeit (**A2**) des Lernalgorithmus als auch

Wiederverwendbarkeit von Daten (**A3**) und Code (**A4**) ermöglichen. Diese Anforderungen spiegeln sich in der in Kapitel 1.4 spezifizierten Zielsetzung wider. Ergänzt wird diese in Kapitel 2.4 um zwei weitere Einschränkungen, die sich aufgrund des Anwendungskontexts Industrie ergeben: Lösungsansätze müssen zur Nutzung von Zeitreihendaten geeignet sein und homogenes Transfer-Lernen umsetzen.

Aufbauend auf den in Kapitel 2.1 bis 2.3 beschriebenen Grundlagen wird in Kapitel 3.1 im Rahmen der DSR-Phase „Lösungsansatzentwicklung“ aufgezeigt, dass der Stand der Wissenschaft verschiedene, prinzipiell geeignete Methoden für (Teil-)Problem-übergreifendes Deep Learning bereithält. Diese umfassen DMM und Regularisierungs-Strategien sowie Domänen-Adaption, gemeinsame Merkmalsextraktion, Mehrköpfigkeit und Feinabstimmung.

In einer Betrachtung zweier gängiger Anwendungsfälle des Anwendungskontext Industrie in den Kapiteln 3.2.1 und 3.2.2 wird der Stand der Technik gesichtet. Zwar existiert eine größere Anzahl verschiedener Lösungsansätze, jedoch erfüllt keiner davon die in Kapitel 1.4 definierten Anforderungen an Lösungsartefakte gemäß der beschriebenen Zielsetzung. Wie eingangs beschriebenen, herrschen Partikularlösungen vor und beschäftigen sich nur wenige Autoren überhaupt mit allgemeiner anwendbaren Ansätzen. Es lassen sich jedoch geeignete (Teil-)Konzepte zur Entwicklung eines eigenen Ansatzes aus den untersuchten Studien entnehmen.

Dem Autor ist darüber hinaus auch kein kommerziell verfügbares System bekannt, welches die oben genannten Anforderungen erfüllen würde. Einschlägige Anbieter werben zwar mit Generalisierbarkeit und der Wiederverwendbarkeit sowohl von Daten als auch von (parametrisierten) Algorithmen, jedoch erfolgt dieser Prozess nicht als Teil des integrierten Lernalgorithmus, sondern im Sinne eines manuellen „Copy and Paste“, also der unmittelbaren und nicht an das neue Problem angepassten Wiederverwendung von Daten bzw. Algorithmen. Die gemäß den in Kapitel 2.2 dargelegten Herausforderungen bei (Teil-)Problem-übergreifendem Lernen anschließend notwendigen Anpassungen muss der Anwender anschließend selbst vornehmen. Dieser Eindruck wird von einschlägigen, Anwender-nahen Einrichtungen wie bspw. der *Plattform Industrie 4.0* bestätigt [36, 37, 243].

Kapitel 3.3 gibt schließlich Aufschluss über zum Einsatz im (Teil-)Problem-übergreifenden Transfer-Lernen im Anwendungskontext Industrie geeignete Clustering-Verfahren.

Aus der Untersuchung der Problemstellung sowie einer Recherche des Stands der Wissenschaft und Technik ergibt sich somit, dass hinsichtlich der in Kapitel 1.2 formulierten Herausforderungen tatsächlich eine Forschungslücke besteht, die jedoch mithilfe der zuvor genannten Ansätze und Konzepte zu schließen ist. Konkret ergibt sich damit für diese Arbeit folgende Forschungsfrage:

Wie kann eine Architektur für übertragbare Lernalgorithmen aufgebaut sein, um auch auf kleinen Datensätzen und trotz der hohen Dynamik von Automatisierungssystemen robust Industrie-typische Probleme zu lösen?

Es bleibt festzuhalten, dass im Rahmen dieser Arbeit keine weitere Partikularlösung, sondern ein generalistisches Konzept für den Einsatz von Deep Learning im Anwendungskontext Industrie entwickelt und anschließend auf verschiedenen konkreten Anwendungsfällen evaluiert werden soll.

4 Architektur für industrielles Transfer-Lernen

In diesem Kapitel wird im Rahmen der Design-Science-Research-Phase „Lösungsansatzentwicklung“ die in dieser Arbeit entwickelte Architektur für industrielles Transfer-Lernen als erstes Lösungsartefakt eingeführt und anschließend in Form eines abstrakten Abgleichs mit den Anforderungen validiert („Lösungsansatzvalidierung“).

Die der Forschungsfrage zugrunde liegenden Anforderungen hinsichtlich einer hohen Wiederverwendbarkeit von Code (**A4**) und einer Nutzbarkeit von Daten unterschiedlicher Qualität, Dimensionalität und Herkunft (**A3**) angesichts stark unterschiedlicher Probleme und Anwendungsfälle im Anwendungskontext Industrie machen einen über Partikularlösungen hinausgehenden Ansatz notwendig. Dies deckt sich mit dem in Kapitel 3.4 beschriebenen Forschungsbedarf, der Defizite nicht (nur) im Bereich von Partikularlösungen für die verschiedensten, auch industriellen Problemtypen sieht, sondern vor allem einen **Mangel an allgemeingültigen Lösungsansätzen** feststellt.

Im Rahmen dieser Arbeit soll daher eine **Architektur für industrielles Transfer-Lernen** vorgestellt werden, die über die verschiedenen Probleme und Anwendungsfälle hinweg eine robuste Struktur für Lösungen unter Berücksichtigung der in Kapitel 1.2 beschriebenen Herausforderungen anbietet.

Nach ISO 42010 ist eine **(Software-)Architektur** definiert als „grundlegende Konzepte oder Eigenschaften eines Systems in seiner Umgebung, die in seinen Elementen, Beziehungen und in den Grundsätzen seiner Gestaltung und Entwicklung verkörpert sind“ [244].

In diesem Sinn werden in den folgenden Unterkapiteln zuerst die Gestaltungs-Grundsätze der Transfer-Lern-Architektur hergeleitet, anschließend vorgestellt und dann daraus Beschreibungen der einzelnen Elemente und deren Beziehungen zueinander abgeleitet. Abschließend findet eine Bewertung der Architektur als erstem Lösungsansatz im Sinne des DSR mittels eines Abgleichs mit den Anforderungen statt.

4.1 Herleitung der Konzeptidee

Ausgangspunkt der Konzeptidee ist die Erkenntnis, dass **Transfer-Lernen** in verschiedenen Anwendungsfeldern bereits produktiv eingesetzt wird bzw. auch im Anwendungskontext Industrie in einer größeren Zahl von Studien einen positiven Beitrag zu verschiedenen Partikularlösungen leistet (siehe Kapitel 3.1.3, 3.2.1.4 und 3.2.2.4). Hervorzuheben sind hier die Anwendungsfelder Bilderkennung und Verarbeitung natürlicher Sprache, in denen Lern-Algorithmen auf umfangrei-

chen Trainingsdatensätzen vortrainiert und anschließend mit geringem Aufwand und kleinen Datensätzen auf neue (Teil-)Probleme angepasst werden können – ohne, dass sie an Leistungsfähigkeit verlieren [4].

Auch im **kontinuierlichen Lernen** gibt es Ansätze, die über eine Unterteilung des Lernalgorithmus in ein schnelles und ein langsam lernendes Teilsystem das Dilemma, wichtige Informationen potenziell dauerhaft zu behalten und dennoch neues Wissen aufnehmen zu können, auflösen [130, 161, 245]. Diesem Ansatz folgend sieht die im folgenden vorgestellte Architektur für industrielles Transfer-Lernen eine Unterteilung des Lern-Algorithmus in drei unterschiedlich dynamische Module mit jeweils unterschiedlichen Rollen vor:

Das **Eingangsmodul** soll weitgehend statisch bleiben und eine Merkmalsextraktion durchführen, um im weiteren Verlauf Ressourcen zu sparen. Dabei soll mittels Mehrköpfigkeit wie in [49] die Integration von heterogenen Datenquellen und die Wiederverwendbarkeit von Code ermöglicht werden.

Das **Transfermodul** soll mittels Clustering eine geeignete Auswahl von zum Transfer genutzten Daten oder Parametern wie in [48, 225, 237] ermöglichen. Eine derartige Ähnlichkeits-basierte Auswahl des Transfer-Wissens ist bei der im Anwendungskontext Industrie vorliegenden Unkenntnis der Verteilungsfunktionen für Quell- und Zielmerkmalsräume erstrebenswert, um einen positiven Transfer zu gewährleisten bzw. das Transfer-Resultat zu verbessern [123–126, 225, 232]. Darüber hinaus erlaubt dieser Ansatz die Wiederverwendung von Daten, reduziert den Aufwand zum Nachtrainieren und stellt eine Basis für die Reduzierung der zum Training benötigten Datenmenge dar. Der konkrete Transfer-Ansatz soll dabei frei wählbar bleiben, um hier zukünftiger Forschung nicht vorzugreifen.

Das **Ausgangsmodul** soll die letztendliche Problemlösung bewerkstelligen. Es ist daher stark durch den jeweiligen Anwendungsfall geprägt, im Kern Deep-Learning-basiert. Es ist Gegenstand regelmäßigen Nachtrainings mittels verschiedener Transfer-Lern-Ansätze und sollte daher von begrenzter Komplexität sein.

Insgesamt soll die im Folgenden vorgestellte Architektur vielfältige Realisierungen hinsichtlich der verwendeten Methoden und Konzepte in Einklang mit den in Kapitel 1.4 definierten Anforderungen ermöglichen, da aus der Literatur keine eindeutig überlegenen Ansätze hervorgehen und die Architektur somit zwar Grundlage für weiterführende, vergleichende Evaluationen möglicher Transfer-Ansätze sein kann, nicht jedoch deren Resultat. Es ist außerdem festzuhalten, dass es immer einen Zielkonflikt zwischen möglichst hoher Vorhersagequalität und möglichst geringer Trainingszeit bzw. Trainingsdatenmenge gibt [207], und eine konkrete Festlegung daher im Kontext des vorliegenden Anwendungsfalls getroffen werden muss und nicht durch die Architektur vorweggenommen werden sollte.

4.2 Überblick über die Architektur

Im Rahmen dieser Arbeit wird die in Kapitel 4.1 hergeleitete Konzeptidee basierend auf dem in Kapitel 3 vorgestellten Stand der Wissenschaft und Technik zu einer Architektur für industrielles Transfer-Lernen weiterentwickelt. Für einen Abgleich dieses Lösungsartefakts und seiner Eigenschaften mit den in Kapitel 1.4 definierten Anforderungen sei auf Kapitel 4.6 verwiesen.

Abbildung 4.1 gibt einen ersten Überblick über den Aufbau der Architektur. Die einzelnen Module können folgendermaßen beschrieben werden:

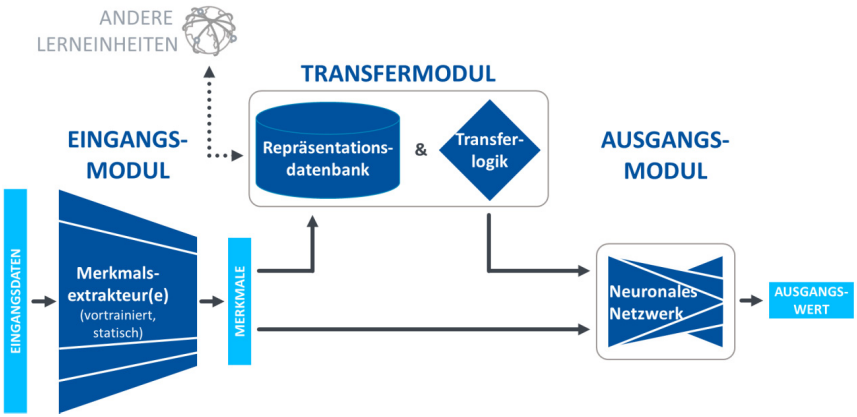

Abbildung 4.1: Schematische Darstellung der Architektur für industrielles Transfer-Lernen

Das **Eingangsmodul** komprimiert die Eingangsdaten mittels eines Merkmalsextrakteurs. Die Eingangsdaten können dabei aus verschiedenen Teil-Segmenten bestehen, die zu einem gemeinsamen Vektor zusammengefügt werden. Ein besonderes Augenmerk wird auf die Verarbeitung von in der Industrie sehr gängigen Zeitreihendaten [31, 35] gelegt. Das Eingangsmodul wird einmalig trainiert und verhält sich anschließend statisch. Kapitel 4.3 bietet eine detaillierte Erläuterung von Aufbau und Funktionsweise des Eingangsmoduls.

Das **Transfermodul** speichert Informationen über bereits bekannte (Teil-)Probleme ab und stellt diese abhängig von den Transfer-Lern-Erfordernissen des jeweiligen (Teil-)Problems zur Verfügung. Kapitel 4.4 bietet eine detaillierte Erläuterung von Aufbau und Funktionsweise des Transfermoduls.

Das (Teil-)Problem-spezifisches **Ausgangsmodul** generiert auf Basis der vom Eingangsmodul bereitgestellten Merkmale einen Ausgangswert. Das Ausgangsmodul wird dabei mittels Transfer-Lernens immer wieder an Veränderungen des (Teil-)Problems angepasst. Kapitel 4.5 bietet eine detaillierte Erläuterung von Aufbau und Funktionsweise des Ausgangsmoduls.

Die gesamte Lern-Architektur soll eine Ausführung abseits der Cloud oder großer Rechenzentren, d.h. mindestens an den Standorten der betrachteten Prozesse und Anlagen oder sogar in diesen selbst ermöglichen.

4.3 Eingangsmodul

Die Aufgabe des Eingangsmoduls ist es, die Eingangsdaten zu komprimieren. Dies senkt einerseits den Speicherbedarf in der Repräsentationsdatenbank und andererseits den Rechenaufwand der Weiterverarbeitung im Transfer- und im Ausgangsmodul. Darüber hinaus ist in diesem Zusammenhang eine Verfremdung, bspw. zur Anonymisierung, der Eingangsdaten möglich – diese Option wird aber im Rahmen dieser Arbeit nicht weiterverfolgt. Dabei ist sicherzustellen, dass bei der Kompression möglichst keine relevanten Informationen verloren gehen.

Das Eingangsmodul wird daher im Kern als ein bzw. mehrere Merkmalsextrakteur(-e) realisiert. Über eine enge Überwachung des Rekonstruktionsverlustes während deren Trainings sowie periodisch während der Nutzung auch mit neuen, bisher unbekannten Eingangsdaten wird sichergestellt, dass während der darin durchgeführten Kompression möglichst keine relevanten Informationen verloren gehen. Vollständig vermeiden lässt sich ein solcher Verlust jedoch nicht, da im Sinne der Problemlösung relevante Informationen auch irrelevant hinsichtlich der Rekonstruktion der Eingangsdaten aus deren extrahierten Merkmalen sein können. Bei schlechten Transfer-Leistungen kann daher die Reduzierung der Kompression angeraten sein.

Weiterhin wird durch die Separierung der Kompression je Eingangsdatenquelle zwar die Kompressionsleistung reduziert, gleichzeitig aber eine Eingangsdatenquellen-übergreifende Abwägung der Relevanz von Teilinformationen unterschiedlicher Herkunft bereits in dieser frühen Phase verhindert. Dies ist eine notwendige Grundlage für die Anwendungsfall-übergreifende Übertragung von Daten einzelner Eingangsdatenquellen.

Abbildung 4.2 veranschaulicht die Funktionsweise des Eingangsmoduls:

Schritt 1: Der Eingangsdatenvektor $\vec{X}_{\text{Gesamt}}$ wird zuerst in seine Teile zerlegt. Dabei werden entsprechend der Eingangsdatenvektor-Zusammensetzung i Teilvektoren $\vec{X}_{\text{i}}$ gebildet bzw. j Skalare X_j isoliert. Zeitreihendaten, die von unterschiedlichen Sensortypen stammen, mit unterschiedlichen Sensorparametern erhoben wurden oder unterschiedliche Sachverhalte darstellen, werden dabei in unterschiedliche Teilvektoren überführt. Dafür notwendige Parameter können entweder auf Basis mitzuliefernder Metadaten automatisch oder von einem Entwickler manuell bestimmt werden.

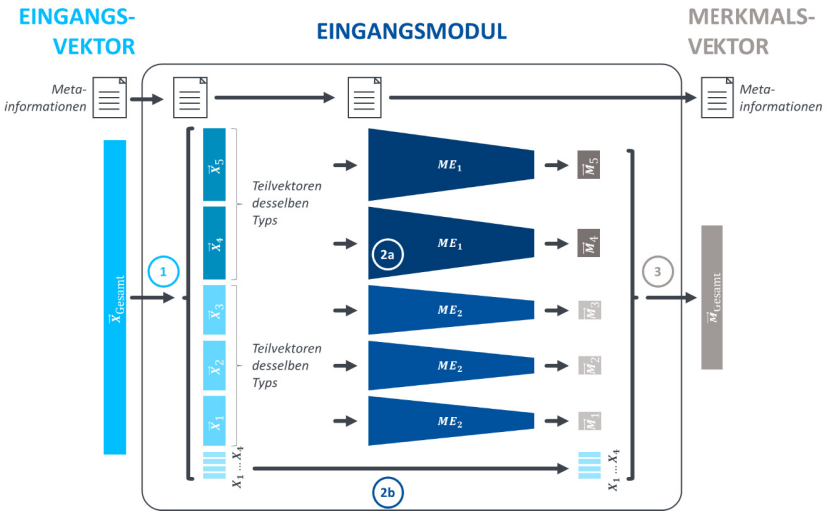

Abbildung 4.2: Schematische Darstellung des Eingangsmoduls

Schritt 2: Anschließend werden die Teilvektoren den eigentlichen Merkmalsextrakteuren ME_i zugeführt. Jeder Teilvektor durchläuft eine separate Merkmalsextraktion, um die Wiederverwendbarkeit der extrahierten Merkmale sicherzustellen (**Schritt 2a**). Abhängig vom Typ des Teilvektors, der bspw. durch den Sensortyp, einen bestimmen Sensorparametersatz oder den Messkontext charakterisiert werden kann, können unterschiedliche Merkmalsextrakteure zum Einsatz kommen – auch diese Zuordnung kann entweder auf Basis mitzuliefernder Metadaten automatisch oder von einem Entwickler manuell vorgenommen werden. Die Merkmalsextrakteure werden einmalig für einen Teilvektortyp trainiert und bleiben anschließend statisch – auch über verschiedene Anwendungsfälle, Standorte etc. hinweg, sodass die resultierenden i Merkmals-Teilvektoren $\vec{M}_i$ vergleichbar bleiben. Die j Skalare X_j werden unverändert überführt (**Schritt 2b**).

Schritt 3: Abschließend werden die Merkmals-Teilvektoren $\vec{M}_i$ und die Skalare Y_j zu einem gemeinsamen Merkmalsvektor $\vec{Y}_{\text{Gesamt}}$ konkateniert.

Die Unterteilung des Eingangsvektors in seine Bestandteile, die dann jeweils einzeln einer Merkmalsextraktion zugeführt werden, orientiert sich am Ansatz von [49]. Eine Domänen-Adaption, d.h. eine Überführung von Eingangsdaten verschiedener Merkmalsräume in einen gemeinsamen Merkmalsraum, findet bewusst nicht statt, da dies die Wiederverwendbarkeit der extrahierten Merkmale gefährden würde.

Für die Merkmalsextrakteure können prinzipiell verschiedene Methoden verwendet werden (siehe u. a. Kapitel 3.1.2). Die konkrete Auswahl ist abhängig vom betrachteten Anwendungsfall. Aufgrund der grundlegenden Bedeutung der extrahierten Merkmale für die Gesamtleistung der Transfer-Lern-Architektur ist auf eine hohe Extraktionsqualität zu achten. Auch die Rechen- und Speicherkomplexität in der Ausführungsphase ist von Bedeutung, wenn man Realisierung in der Edge anstrebt. Der Trainingsaufwand ist hingegen weniger relevant, da er nur einmalig anfällt.

4.4 Transfermodul

Die Aufgabe des Transfermoduls ist es, Merkmalsvektoren, Netzwerkparameter und jeweils dazugehörige Metainformationen (Teil-)Problem-übergreifend zu speichern und bedarfsgerecht verfügbar zu machen. Ersteres wird von der Repräsentationsdatenbank gewährleistet, für Letzteres ist im Wesentlichen die Transferlogik zuständig.

4.4.1 Repräsentationsdatenbank

Die Aufgabe der Repräsentationsdatenbank ist es, Merkmalsvektoren, Netzwerkparameter und jeweils dazugehörige Metainformationen (Teil-)Problem-übergreifend zu speichern und der Transferlogik oder einem Ausgangsmodul auf Anfrage zur Verfügung zu stellen.

Die Repräsentationsdatenbank muss primär Vektoren mit numerischen Einträgen performant verwalten können. Dies kann über dedizierte Datenbankmanagementsysteme erfolgen, abhängig von der Datenmenge können jedoch auch einfache Datenformate wie bspw. CSV-Dateien ggf. in Verbindung mit den Dateiverwaltungssystemen des jeweiligen Betriebssystems ausreichend sein.

Die Repräsentationsdatenbank kann um Verwaltungsfunktionen bspw. zur Bereinigung von redundanten Informationen erweitert werden. Da dieser Aspekt keine unmittelbaren Auswirkungen auf die Performanz des Lernalgorithmus hat, wird er im Rahmen dieser Arbeit nicht weiter untersucht.

Die Repräsentationsdatenbank kann darüber hinaus auch verteilt vorliegen bzw. über den Austausch zwischen verschiedenen Repräsentationsdatenbank-Entitäten Informationen erst auf Anfrage lokal zugänglich machen. Da auch dieser Aspekt keine unmittelbaren Auswirkungen auf die Performanz des Lernalgorithmus hat, wird er im Rahmen dieser Arbeit nicht weiter untersucht.

4.4.2 Transferlogik

Die Aufgabe der Transferlogik ist es, geeignete Transfer-Informationen auszuwählen. Da die konkret übertragenen Wissenselemente, d.h. Daten- oder Parametersätze, einen erheblichen Einfluss

auf den Erfolg des Wissenstransfers im Kontext des jeweiligen (Teil-)Problems haben [123–126, 225, 232], sollte dieser Wissenstransfer selektiv erfolgen.

Abbildung 4.3 veranschaulicht die Funktionsweise des Transfermoduls:

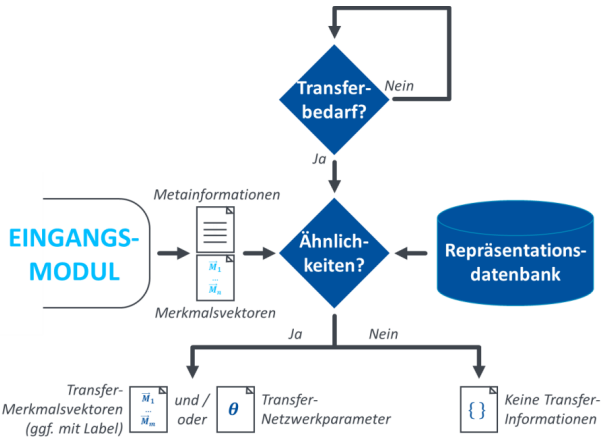

Abbildung 4.3: Schematische Darstellung des Transferlogik

Insbesondere für hochgradig automatisierte Lernalgorithmen ist eine automatische **Transfer-Bedarfs**-Prüfung sinnvoll. In diesen Fällen werden die Betriebs-Ergebnisse des Ausgangsmoduls kontinuierlich anhand zu definierender Kriterien überwacht. Legt eine Verschlechterung dieser Ergebnisse eine hinreichende Veränderung des (Teil-)Problems nahe, wird eine Anpassung mittels Transfer-Lernen ausgelöst. Alternativ kann der Transfer-Bedarf auch manuell festgestellt bzw. das Transfer-Lernen manuell ausgelöst werden – dies bspw. im Fall von Produkt- oder Maschinenwechseln, die eine Anpassung des Lernalgorithmus, wenigstens im Sinne einer Rekalibrierung, unmittelbar nötig machen.

Der selektive Wissenstransfer wird im Kern über einen Vergleich der Merkmalsvektoren des neuen (Teil-)Problems mit denen in der Repräsentationsdatenbank realisiert. Diese **Ähnlichkeits**-Prüfung kann bspw. auf Basis von Clustering erfolgen und Metainformationen berücksichtigen. Abhängig vom gewählten Transfer-Verfahren, also bspw. Daten- bzw. Instanz- oder Modell- bzw. Parameter-Transfer, werden anschließend ausreichend ähnliche, für überwachtes Lernen gelabelte Merkmalsvektoren bekannter (Teil-)Probleme oder die zugehörigen Netzwerkparameter bereitgestellt. Werden keine ausreichend ähnlichen Merkmalsvektoren gefunden, so ist Transfer-Lernen in diesem Fall nicht möglich.

Die Ähnlichkeits-Prüfung muss in der Lage sein, innerhalb kurzer Zeit und mit geringen Anforderungen an Speicher- und Rechenleistung neue Merkmalsvektoren mit der stetig wachsenden, vieldimensionalen Merkmalsvektorensammlung in der Repräsentationsdatenbank zu vergleichen und dabei induktiv vorgehen, um neue Merkmalsvektoren ohne eine komplette Neubehandlung des gesamten Datensatzes eingliedern zu können. Für eine Realisierung mittels Clustering lässt diese Anforderung nur eine Realisierung mittels des BIRCH-Algorithmus zu (vgl. Kapitel 3.3).

Die Transferlogik muss in ihrer Funktionsweise und Parametrierung auf die Erfordernisse des Ausgangsmoduls im jeweiligen Problem-Kontext abgestimmt sein. Es müssen somit beide aufeinander abgestimmt werden.

4.5 Ausgangsmodul

Die Aufgabe des Ausgangsmoduls ist es, das jeweils vorliegende (Teil-)Problem zu lösen. Damit wird für jedes neue (Teil-)Problem auch ein neues Ausgangsmodul gebildet. Abhängig davon, wie stark sich die (Teil-)Probleme unterscheiden, können sich auch die Ausgangsmodule unterscheiden – bis hin zu völlig anderen Netzwerkarchitekturen. Auch die Art des Wissenstransfers kann sich von (Teil-)Problem zu (Teil-)Problem unterscheiden.

Abbildung 4.4 veranschaulicht eine mögliche Funktionsweise des Ausgangsmoduls im Fall von Instanz-Transfer:

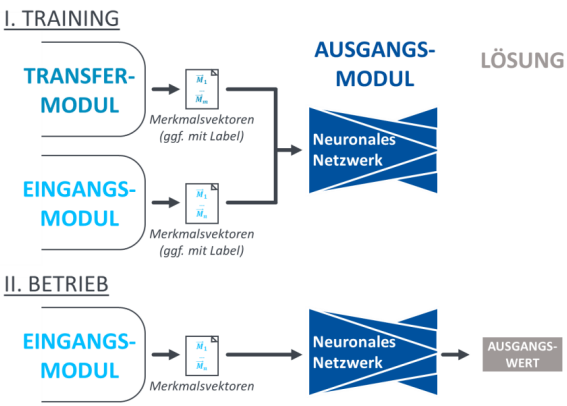

Abbildung 4.4: Schematische Darstellung von Training und Betrieb eines Ausgangsmoduls im Fall von Instanz-Transfer

In der **Trainingsphase** wird das Ausgangsmodul mit vom Transfermodul bereitgestellten Merkmalsvektoren bekannter (Teil-)Probleme sowie Merkmalsvektoren des neuen (Teil-)Problems

trainiert. Im Fall überwachten Lernens müssen diese Merkmalsvektoren gelabelt sein, im Falle unüberwachten Lernens ist dies nicht notwendig.

In der **Betriebsphase** werden dem Ausgangsmodul ausschließlich ungelabelte Merkmalsvektoren des neuen (Teil-)Problems zugeführt, auf deren Basis ein Ausgangswert inferiert wird.

Abbildung 4.5 veranschaulicht demgegenüber eine mögliche Funktionsweise des Ausgangsmoduls im Fall von Parameter-Transfer:

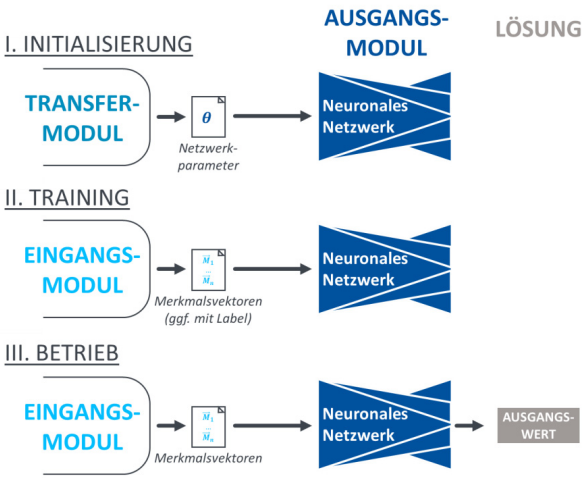

Abbildung 4.5: Schematische Darstellung von Training und Betrieb eines Ausgangsmoduls im Fall von Parameter-Transfer

In der **Initialisierungsphase** wird das Ausgangsmodul mittels vom Transfermodul bereitgestellten Netzwerkparametern initialisiert. Es wird also eine Kopie eines bereits zuvor eingesetzten Ausgangsmoduls erstellt.

In der **Trainingsphase** wird das zuvor initialisierte Ausgangsmodul mit vom Eingangsmodul bereitgestellten Merkmalsvektoren des neuen (Teil-)Problems weitertrainiert. Im Fall überwachten Lernens müssen diese Merkmalsvektoren gelabelt sein, im Falle unüberwachten Lernens ist dies nicht notwendig.

In der **Betriebsphase** werden dem Ausgangsmodul ausschließlich ungelabelte Merkmalsvektoren des neuen (Teil-)Problems zugeführt, auf deren Basis ein Ausgangswert inferiert wird.

Prinzipiell muss nicht der unmittelbare Ausgabewert des im Ausgangsmodul verwendeten neuronalen Netzwerks als Ausgangswert des Ausgangsmoduls genutzt werden. Es kann stattdessen

auch eine Weiterverarbeitung dieses unmittelbaren Ausgabewerts erfolgen, bspw. über einen Vergleich mit einem Schwellenwert im Fall einer Anomaliedetektion mittels Autoencodern.

Aufgrund der ggf. häufigen Neu-Erstellung der Ausgangsmodule sollten diese schnell zu trainieren sein. Übermäßig komplexe neuronale Netzwerke sind somit nicht geeignet. Davon abgesehen können prinzipiell alle in Kapitel 3.1 aufgeführten Methoden verwendet werden.

4.6 Abgleich mit Anforderungen

Die zuvor beschriebene Architektur für industrielles Transfer-Lernen stellt einen ersten Lösungsansatz im Sinne des DSR dar. In der folgenden ersten Lösungsansatzvalidierung soll er als abstraktes Konzept mit den in Kapitel 1.4 an ein solches Lösungsartefakt gestellten Anforderungen hinsichtlich seiner Eignung zur Beantwortung der Forschungsfrage abgeglichen werden.

Robustheit, um die Notwendigkeit des Nachtrainierens zu reduzieren (A1)

Das Transfermodul vergrößert durch die Möglichkeit, passende Daten oder passende Parameter vergangener (Teil-)Probleme zu übernehmen, die Trainingsbasis der Lernalgorithmen und damit die Wahrscheinlichkeit des erfolgreichen Umgangs mit ungünstigen, weil bspw. stärker abweichenden, Eingangsdaten. Der in der vorgestellten Architektur manifestierte, modulare Aufbau des Gesamtalgorithmus reduziert den Trainingsdatenbedarf weiter, da bereits vortrainierte (Teil-)Komponenten einfach übernommen werden können, was wiederum die Wahrscheinlichkeit robusten Lösungsverhaltens erhöht. Darüber hinaus lässt die offene, modulare Architektur viel Freiheit bei der Anpassung an den konkreten Anwendungsfall, sodass aus ihr kaum bzw. keine Zwänge resultieren, die die Auswahl eines suboptimalen und damit mit hoher Wahrscheinlichkeit weniger robusten Ansatzes bspw. der Problemlösung im Ausgangsmodul erzwingen würden.

Adaptierbarkeit, um den Aufwand des Nachtrainierens zu reduzieren (A2)

Das Transfermodul reduziert durch die Möglichkeit, passende Daten oder passende Parameter vergangener (Teil-)Probleme zu übernehmen, den Datenerhebungsaufwand, da so potenziell weniger Daten des Zielproblems benötigt werden. Der in der vorgestellten Architektur manifestierte, modulare Aufbau des Gesamtalgorithmus reduziert den Anpassungsaufwand an neue (Teil-)Probleme, da (Teil-)Komponenten einfach übernommen werden können. Schließlich reduziert die Merkmalsextraktion den Rechenaufwand beim Training, da die resultierenden Merkmalsvektoren deutlich kürzer als die initialen Eingangsdatenvektoren sind und somit von deutlich kompakteren neuronalen Netzwerken verarbeitet werden können. Darüber hinaus ermöglicht das Transfermodul mit seinem Vergleich von auch ungelabelten Trainingsdatensätzen mit in der Repräsentationsdatenbank hinterlegten, bereits bekannten und gelabelten Datensätzen einen Transfer dieser Informationen vom bekannten auf das neue, unbekannte (Teil-)Problem. Eine derartige Übertragung kann bspw. mittels unüberwachter Domänen-Adaption erfolgen.

Wiederverwendbarkeit von Daten zur Anpassung an neue Anwendungskontexte (A3)

Das Eingangsmodul ermöglicht mittels der mehrköpfigen Merkmalsextraktion die einfache Integration heterogener Datenquellen, da jeder Datenquelle ein anderes Merkmalsextraktions-Teilmodul zugeordnet werden kann und die nachfolgenden Module lediglich die resultierenden Merkmale verarbeiten. Somit ist eine gleichzeitige Behandlung von bspw. Bild- und Zeitreihendaten möglich. Weiterhin erlaubt die Repräsentationsdatenbank das Datensammeln unabhängig vom konkreten Anwendungsfall und einen Austausch dieser Daten über Anwendungsfall-Grenzen hinweg.

Wiederverwendbarkeit von Lernalgorithmen zur Anpassung an neue Anwendungskontexte (A4)

Durch den modularen Aufbau wird eine hohe Wiederverwendbarkeit von Code erzielt, bspw. in Form der Übernahme kompletter Module von einem Anwendungsfall zum nächsten. Zusätzlich ermöglicht die innere Struktur der Eingangsmoduls, namentlich die Nutzung von unterschiedlichen Merkmalsextrakteuren für verschiedene Sensortypen, Sensorparametersätze oder Messkontexte, deren Wiederverwendung auch in Teilen in anderen Anwendungsfällen.

Fazit

Zusammenfassend lässt sich feststellen, dass die Lösungsansatzvalidierung für das Lösungsartefakt „Architektur für industrielles Transfer-Lernen“ einen hohen Erfüllungsgrad der Anforderungen ergibt. Die Architektur ist damit grundsätzlich zur Beantwortung der Forschungsfrage geeignet und soll daher im folgenden Kapitel im Rahmen einer zweiten Iteration des DSR-Entwurfszyklus für verschiedene Anwendungsfälle implementiert und evaluiert werden.

5 Machbarkeitsuntersuchung: Prototypische Umsetzung und Evaluierung

In diesem Kapitel wird die in Kapitel 4 vorgestellten Architektur für industrielles Transfer-Lernen im Sinne einer zweiten Iteration des DSR-Entwurfszyklus näher untersucht. Dazu werden nun in drei Evaluierungsfällen ihre prototypischen Realisierungen als konkrete Lösungsartefakte entworfen und einer angewandten Validierung unterzogen. Auf diese Weise wird die Eignung der Architektur zur Beantwortung der in Kapitel 3.4 formulierten Forschungsfrage mittels eines Abgleichs mit den in Kapitel 1.4 formulierten Anforderungen geprüft.

5.1 Aufbau der Machbarkeitsuntersuchung

Grundsätzlich zielen die beschriebenen Maßnahmen auf die Gewährleistung effizienten Lernens angesichts veränderlicher (Teil-)Probleme ab. Daraus ergibt sich, dass diese Veränderlichkeit auch von den zur Evaluierung genutzten Szenarien an den Tag gelegt werden muss. Darüber hinaus soll die Architektur Anwendungsfall-übergreifend in der Automatisierungstechnik einsetzbar sein. Im Folgenden werden daher drei dynamische Evaluierungsfälle mit unterschiedlichen Problemstellungen aus der Industrieautomatisierung verwendet.

Diese beiden Problemstellungen „Anomaliedetektion“ und „Ausfallvorhersage“ werden in der wissenschaftlichen Literatur vielfältig behandelt (siehe Kapitel 3.2.1 und 3.2.2) und auch von einschlägigen Verbänden wie bspw. der *Plattform Industrie 4.0* als relevant eingestuft [46, 246]. Darüber hinaus konnte am Institut für Automatisierungstechnik und Softwareentwicklung (IAS) der Universität Stuttgart auf entsprechende Datensätze zugegriffen werden, die eine Evaluation anhand dieser Problemstellungen überhaupt erst ermöglichen. Mit der Auswahl dieser Problemstellungen ist keine Aussage über deren universelle Repräsentativität für „industrietypische Probleme“ gemäß der Forschungsfrage verbunden, sie stellen stattdessen den Rahmen für die Artefaktbasierte Ableitung allgemeinen, übertragbaren und fundierten Gestaltungswissens im Sinne der DSR.

Entsprechend Abbildung 5.1 erfolgt die Machbarkeitsuntersuchung auf zwei Ebenen: Innerhalb eines Evaluierungsfalls wird das Hauptaugenmerk auf Effizienz- und Genauigkeitsgewinne angesichts dynamischer (Teil-)Problemstellungen gelegt. Es handelt sich dabei um klassisches Transfer-Lernen gemäß der Definition aus Kapitel 2.2.1. Über die verschiedenen Evaluierungsfälle hinweg liegt der Fokus auf der vielseitigen und einfachen Anwendbarkeit der vorgestellten Architektur auf verschiedene Problemstellungen, wobei insbesondere die Möglichkeit der Wiederverwendung von Architekturelementen und die Flexibilität der Architektur betrachtet wird.

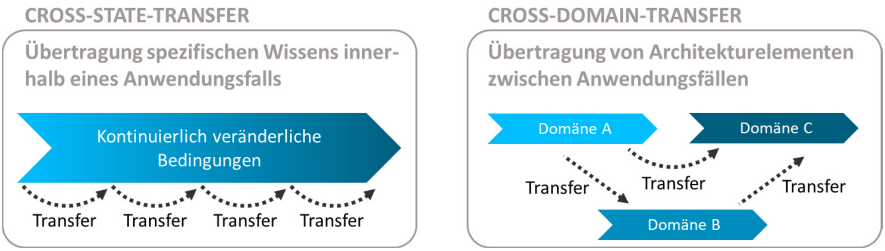

Abbildung 5.1: Unterschiedliche Ebenen des Transfer-Lernens nach [116]

In den Evaluierungsfällen 1 und 2 wird dabei eine Referenz-Implementierung vorgestellt und detailliert evaluiert, die aufgrund ihrer hohen Modularität ein breites Einsatzspektrum verspricht. Trotz des deutlich anderen Transfer-Szenarios und veränderter Eingangsdaten können in Evaluierungsfall 2 signifikante Anteile Code aus Evaluierungsfall 1 wiederverwendet werden. In Evaluierungsfall 3 wird demgegenüber ein stärker integrierter Ansatz auf Basis einer Domänen-Adaption realisiert und getestet, um auch die methodische Flexibilität der Architektur zu demonstrieren.

Grundsätzlich unterteilt sich jedes Evaluierungsfall-Unterkapitel in die folgenden Abschnitte: Zuerst werden der Anwendungsfall und die dazu vorliegenden Datensätze eingeführt. Anschließend wird die Evaluierungsfall-spezifische, prototypische Realisierung der Architektur beschrieben, bevor diese dann evaluiert wird.

5.2 Evaluierungsfall 1: Anomaliedetektion auf Pumpendruckverläufen

Evaluierungsfall 1 dient als Referenz-Implementierung der in Kapitel 4 vorgestellten Architektur für industrielles Transfer-Lernen. Auf einem umfangreichen Datensatz eines schnell veränderlichen Industrieprozesses soll basierend auf einer Ähnlichkeitsanalyse Wissen von bekannten, vergangenen Prozesszuständen hin zu neuen, bisher unbekannten Prozesszuständen übertragen werden, um eine effizientere Anomaliedetektion mittels neuronaler Netze zu ermöglichen. Zu diesem Zweck werden alle in Kapitel 4 genannten Komponenten prototypisch realisiert.

Dabei liegt Evaluierungsfall 1 die Annahme zugrunde, dass der Zeitpunkt des Übergangs zwischen zwei Prozesszuständen bekannt ist. Darüber hinaus sind die Datensätze der bereits bekannten Problemvarianten gelabelt. Die Aufgabe ist nun, für neue Prozesszustände, d.h. andere bisher unbekannte Problemvarianten, eine Lösung auf Basis ungelabelter Daten zu finden. Es handelt sich also um unüberwachtes Lernen.

5.2.1 Anwendungsfall

In Evaluierungsfall 1 wird die Anomaliedetektion für Hydraulikpumpen auf Basis von Druckverlaufs-Daten betrachtet. Die insgesamt acht Axialkolbenpumpen beaufschlagen ein gemeinsames Ölreservoir mit Druck, dass im Rahmen eines Massivumformungsprozesses zwei Schmiedeformen gegeneinanderdrückt und damit aus einem metallenen Halbzeug ein fertiges Exemplar eines Produktes erzeugt.

Pro Produkt werden nur wenige 100 Stück gefertigt und auch während der Fertigung eines Produkts kann es zu Prozessverbesserungen kommen, die eine Veränderung der Druckverläufe bewirken. Die Druckverläufe können sich dabei unter anderem hinsichtlich ihrer Dauer und der Anzahl der Pressvorgänge pro gefertigtem Exemplar unterscheiden.

Weil die Pumpen die Schmiedepresse gemeinsam mit Druck beaufschlagen wird anormales Verhalten einer Pumpe weitgehend von den anderen kompensiert. Dies führt jedoch zu erhöhtem Verschleiß der anderen Pumpen und birgt die Gefahr eines ungeplanten Komplettausfalls der Anlage. Aufgrund der hohen Belastung der Pumpen ist anormales Verhalten bei mindestens einer der acht Pumpen zudem eher die Regel als die Ausnahme. Daher ist die frühzeitige Detektion anormalen Pumpenverhaltens wünschenswert, um rechtzeitig einen Austausch vornehmen und die anderen Pumpen somit schonen zu können.

Die datengetriebene Anomaliedetektion für Hydraulikpumpen steht somit vor zwei Herausforderungen: Einerseits müssen schon kleine Abweichungen der Druckverläufe als Anomalie erkannt werden, andererseits weisen die Druckverläufe zwischen den verschiedenen Fertigungsvarianten erheblich größere Unterschiede des Sollverhaltens auf. Zusätzlich ist ein Labeling der Trainingsdaten aufgrund der hohen Frequenz neuer Fertigungsvarianten nicht praktikabel.

5.2.1.1 Datensatz

Im Rahmen von Evaluierungsfall 1 wird auf einen Datensatz aus einem vorangegangenen Drittmittelprojekt zurückgegriffen.

Während des normalen Betriebs der Fertigungsanlage wurden Pumpendruckverläufe aufgezeichnet. Je nach Zustand der Pumpe sind diese entweder „normal“, „anormal“ oder „defekt“ (siehe Abbildung 5.2, links). Der Druck bewegt sich dabei im Bereich von 0 bis ca. 30 Megapascal. Abhängig vom zu fertigenden Produkt und gewählten Fertigungsprozess wird der Druck unterschiedlich oft (siehe „Anzahl Pressvorgänge“ in Tabelle 5.1), unterschiedlich lang und unterschiedlich schnell aufgebracht (siehe Abbildung 5.2, rechts). Ein Fertigungsvorgang kann damit von ca. 12 Sekunden bis ca. 30 Sekunden dauern. Die Abtastrate beträgt 10 Hz. Jeweils acht Druckverläufe bilden einen Fertigungsvorgang ab.

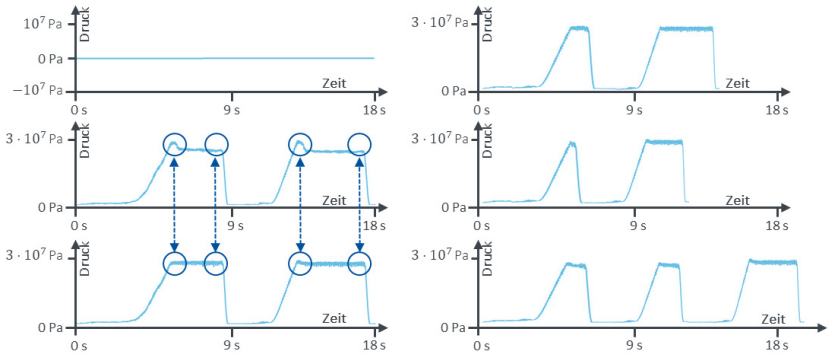

Abbildung 5.2: Druckverlauf einer defekten, einer anormalen und einer normalen Pumpe (links, von oben nach unten) mit Hervorhebung der charakteristischen Unterschiede; unterschiedliche Varianten der Fertigung desselben Produktes Nr. 37154 (rechts)

5.2.1.2 Datenvorverarbeitung

Im Rahmen von [247, 248] wurden die in Tabelle 5.1 aufgelisteten Pumpendruck-Teildatensätze aus dem Gesamtdatensatz herausextrahiert und manuell gelabelt. Die unterschiedliche Verteilung der drei Pumpenzustände „normal“, „anormal“ und „defekt“ folgt keiner besonderen Auswahllogik, sondern ist dem zufälligen Auftreten von Pumpenfehlern im Produktivbetrieb und der notwendigerweise zeitversetzten Reaktion darauf geschuldet. 100 gefertigte Exemplare entsprechen je nach Dauer der Fertigung eines Exemplars ungefähr zwischen 20 und 50 Minuten Produktionszeit, jedoch wurden nicht in jeder Variante auch mindestens 100 Exemplare gefertigt, sodass die Anzahl der Druckverläufe in beide Richtungen leicht vom Mittelwert 800 abweicht.

Tabelle 5.1: Überblick über die Pumpendruck-Datensätze in Evaluierungsfall 1

Fertigungsvariante	Anzahl Exemplare	Anzahl Druckverläufe	Anzahl Pressvorgänge	Anzahl Proben „Normal“	Anzahl Proben „Anormal“	Anzahl Proben „Defekt“
37154-1	100	800	2	500	100	200
37154-2	89	712	2	445	89	178
37154-3	118	944	3	826	0	118
38423-1	103	824	2	675	46	103
38423-2	102	816	2	714	0	102
38423-3	77	616	3	446	93	77

Ferti-gungs-variante	Anzahl Exemp-lare	Anzahl Druckver-läufe	Anzahl Pressvor-gänge	Anzahl Proben „Normal“	Anzahl Proben „Anormal“	Anzahl Proben „Defekt“
38423-4	109	872	3	654	109	109
38520-1	105	840	4	619	11	210
38674-1	97	776	3	642	37	97
38675-1	104	832	3	624	104	104
38675-2	95	760	3	570	0	190
38689-1	98	784	3	491	195	98
38689-2	100	800	4	700	0	100

Die einzelnen Proben werden auf einen Wertebereich von 0 bis 1 normalisiert und auf eine Länge von 3000 mit Einträgen vom Wert 0 erweitert (engl.: Padding).

5.2.2 Realisierung

Im Rahmen von [112, 247–249] wurden verschiedene Varianten eines von der in Kapitel 4 vorgestellten Architektur abgeleiteten, industriellen Transfer-Lernalgorithmus zur Anomaliedetektion (vgl. Kapitel 3.2.1.1) entworfen. Dabei kamen die Frameworks PyTorch 1.7.0 [250] und scikit-learn 0.23.2 [239] unter Python 3.6 [251] zum Einsatz.

Abbildung 5.3 gibt einen schematischen Überblick über die dabei genutzte Realisierung der Lern-Architektur:

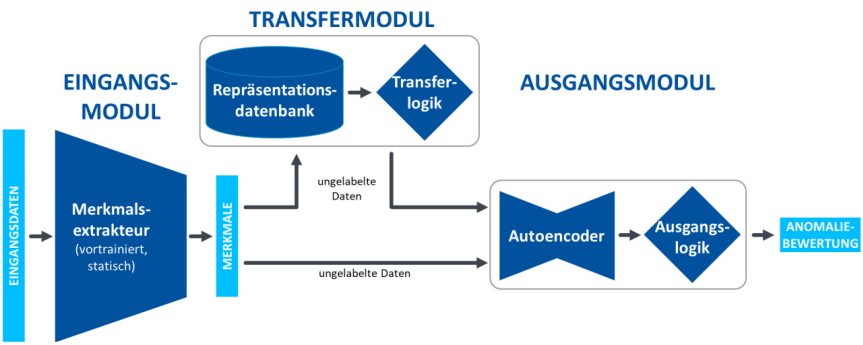

Abbildung 5.3: Schematische Darstellung der Realisierung der Lern-Architektur in Evaluierungsfall 1

Die Eingangsdaten werden mittels des Merkmalsextrakteurs im Eingangsmodul auf erheblich kürzere Merkmalsvektoren reduziert. Diese Merkmalsvektoren werden inklusive dazugehöriger Metadaten einerseits der Repräsentationsdatenbank zur dauerhaften Speicherung übermittelt. Andererseits wird bei neuen, noch unbekannten Prozessen mithilfe dieser neuen Merkmalsvektoren durch die Transferlogik aus der Repräsentationsdatenbank ein Transfer-Daten- bzw. -Parametersatz gebildet, der anschließend zum Training des Ausgangsmodul genutzt wird. Das Ausgangsmodul besteht aus einem Autoencoder und einer Ausgangs-Logik, die anhand des Rekonstruktionsfehlers des Autoencoders ermittelt, ob es sich um normale oder anormale Proben handelt.

5.2.2.1 Eingangsmodul: Merkmalsextraktion

Entsprechend den Erkenntnissen aus Kapitel 3.1.2 werden für das Eingangsmodul Merkmalsextrakteure basierend auf dem Konzept des Autoencoders (vgl. Kapitel 2.1.3) verwendet.

Es werden drei gängige Merkmalsextraktions-Architekturen, LSTM, CNN und FCNN, implementiert und miteinander vergleichen. Eine systematische Hyperparameter-Optimierung findet aufgrund des damit verbundenen Rechenaufwands nicht statt, stattdessen werden die Hyperparameter basierend auf Erfahrungswerten ausgewählt. Dies trägt auch dem Umstand Rechnung, dass in dieser Arbeit der Fokus auf dem Beleg der grundsätzlichen Anwendbarkeit des Konzepts und nicht auf der Findung einer optimalen Lösung für den spezifischen Anwendungsfall, hier die Anomaliedetektion für Hydraulikpumpen, liegt.

Tabelle 5.2 gibt einen Überblick über die Struktur der verschiedenen evaluierten Merkmalsextrakteurs-Varianten. Aufgrund des Paddings der Eingangsdaten beträgt die Eingangsvektorlänge in Evaluierungsfall 1 immer 3000. Als Aktivierungsfunktion kommen innerhalb der Kodierer ReLU und an ihrem Ausgang SELU zum Einsatz. Die letzte Schicht des FCNN hat die Länge der Merkmalsvektoren $L_{Merkmal}$, Padding und Dilation kommen im CNN nicht zum Einsatz. Als Bewertungskriterium kommt der gemittelte, quadratische Fehler (engl.: Mean Squared Error, kurz: MSE) zum Einsatz.

Tabelle 5.2: Überblick über die Merkmalsextrakteurs-Varianten in Evaluierungsfall 1

Nr.	Kodierer-Struktur	Nr.	Kodierer -Struktur
1	3-Schichtiges CNN Schicht 1: $3000 \times \frac{L_{Merkmal}}{3} \times 10, Stride = 5$ Schicht 2: $\frac{L_{Merkmal}}{3} \times \frac{L_{Merkmal}}{2} \times 5, Stride = 3$ Schicht 3: $\frac{L_{Merkmal}}{2} \times L_{Merkmal} \times 3, Stride = 2$ Legende: Eingang x Ausgang x Kernel	2	2-Schichtiges FCNN Schicht 1: 500 Knoten
		3	2-Schichtiges LSTM-Netzwerk Schicht 1: $10 \cdot L_{Merkmal}$ Knoten Schicht 2: $L_{Merkmal}$ Knoten

5.2.2.2 Transfermodul: Repräsentationsdatenbank und Transferlogik

Das Transfermodul wird entsprechend den Ausführungen in Kapitel 4.4 in verschiedenen Varianten des Instanz- und Parameter-Transfers realisiert. Es liefert somit ein um Merkmalsvektoren und bzw. oder Netzwerkparameter aus bereits bekannten (Teil-)Problemen erweitertes Transfer-Wissen für je nach Transfer-Strategie Initialisierung und bzw. oder Training des Ausgangsmoduls.

Abbildung 5.4 gibt einen schematischen Überblick über den Aufbau des Transfermoduls in der Transfer-Strategie „Datentransfer“:

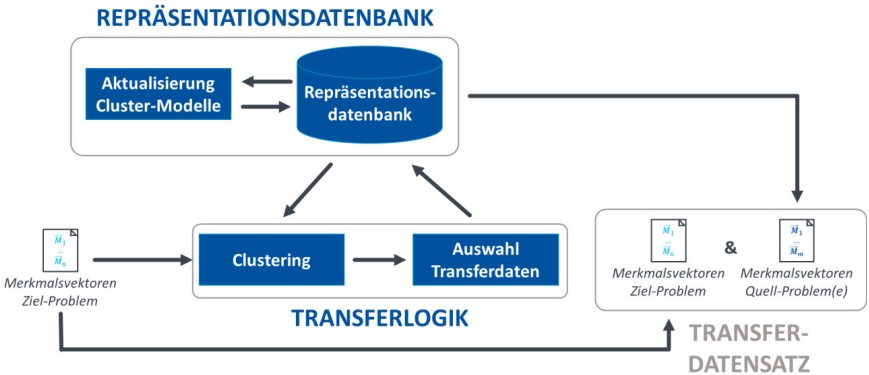

Abbildung 5.4: Schematische Darstellung des Transfermoduls in Evaluierungsfall 1 unter Anwendung der Transfer-Strategie „Datentransfer“

Die eigentliche Repräsentationsdatenbank ist als strukturierte Sammlung von .txt-, .joblib- und .pt-Dateien ausgeführt. Die .txt-Dateien enthalten die Merkmalsvektoren bereits erlernter (Teil-) Probleme sowie die damit zusammenhängenden Meta-Daten, wie bspw. einen Zeitstempel oder die Fertigungsvariante. Die .joblib-Dateien enthalten die Cluster-Beschreibungen bereits geclusterter Proben, also neben den Relationen der Verzweigungsknoten zueinander für jeden dieser Verzweigungsknoten die Anzahl der enthaltenen Proben, deren Vektorsumme und quadratische Vektorsumme pro Dimension. Die .pt-Dateien enthalten pro Cluster die Netzwerkparameter eines auf den Daten dieses Clusters trainierten Ausgangsmoduls. Sind hinreichend viele neue Proben in die Repräsentationsdatenbank aufgenommen worden, so findet eine Aktualisierung der Cluster durch eine komplette Wiederholung des Clusterings statt.

Aufgrund der kurzen Prozessdauern und der klaren Abgrenzbarkeit von unterschiedlichen Prozessen muss der Transfer-Bedarf nicht aufwendig ermittelt werden, sondern ist dem Bedienpersonal üblicherweise bekannt. Nach einem Prozess-Wechsel werden die ersten Merkmalsvektoren des nun neuen (Ziel-)Problems dem Clustering zugeführt. Die Nutzung des BIRCH-Algorithmus

verspricht dabei das effiziente Hinzufügen dieser wenigen, neuen Proben zu den bereits geclusterten Proben in der Repräsentationsdatenbank (vgl. Kapitel 3.3). Dies reduziert den Clustering-Aufwand signifikant. Basierend auf dem Clustering-Ergebnis wird anschließend das Transfer-Wissen, also der Transfer-Daten- oder Parametersatz ausgewählt. Dazu werden die bereits bekannten (Quell-)Proben bzw. die damit zusammenhängenden (Quell-)Netzwerkparameter aus dem Cluster, dem die meisten Zielproben zugeordnet werden, als Transferwissen definiert. Nach Bedarf kann zusätzlich eine Ähnlichkeitsbetrachtung basierend auf der Cosinus-Ähnlichkeit (vgl. 2.3.1) durchgeführt werden, um innerhalb des Clusters eine noch passgenauere Auswahl dieses Transferwissens zu ermöglichen.

5.2.2.3 Ausgangsmodul: Anomaliedetektion

Entsprechend den Ausführungen in Kapitel 3.2.1 können verschiedene Deep-Learning-Architekturen zur Anomaliedetektion eingesetzt werden. In Evaluierungsfall 1 wird auf einen Autoencoder realisiert mittels eines simplen, einschichtigen FCNN in Verbindung mit einer nachgeschalteten Ausgangslogik zurückgegriffen.

Tabelle 5.3 gibt einen Überblick über die Struktur des verwendeten Autoencoders. Aufgrund der potentiell unterschiedlichen Längen der Merkmalsvektoren ist eine Definition der Struktur in Abhängigkeit von dieser Merkmalsvektorlänge $L_{Merkmal}$ sinnvoll – in Kapitel 5.2.3 kommen jedoch ausschließlich Autoencoder für $L_{Merkmal} = 50$ zum Einsatz. Als Aktivierungsfunktion kommen im Kodierer ReLU und im Dekodierer SELU zum Einsatz. Als Bewertungskriterium kommt der MSE zum Einsatz.

Tabelle 5.3: Aufbau des Autoencoders zur Anomaliedetektion in Evaluierungsfall 1

Kodierer-Struktur	**Dekodierer-Struktur**
1-Schichtiges FCNN Schicht 1: $\frac{1}{2} \cdot L_{Merkmal}$ Knoten	1-Schichtiges FCNN Schicht 1: $L_{Merkmal}$ Knoten

Relevant für die Anomaliedetektion sind nicht der zwischen Kodier- und Dekodier-Schicht vorliegende, komprimierte Merkmalsvektor oder der am Ausgang reduzierte (Eingangs-)Merkmalsvektor, sondern der Rekonstruktionsverlust, d.h. der Unterschied zwischen Ein- und Ausgangs-Merkmalsvektor. Dieser Rekonstruktionsverlust wird in der Ausgangslogik interpretiert:

Der Rekonstruktionsverlust-Schwellenwert wird ähnlich des dynamischen Schwellenwerts (engl.: Dynamic Threshold) aus [207, 252] zur Unterscheidung zwischen normalen und anormalen Proben als

$$\text{Schwellenwert} = \tilde{L} + k \cdot \underset{i=0}{\overset{n}{\text{Median}}} \left(\left|\tilde{L} - L_i\right|\right)$$

mit $\tilde{L}$ als dem Median des Rekonstruktionsverlusts über alle zur Bestimmung des Anomalie-Schwellenwerts genutzten Proben, k als empirisch zu bestimmender Vorfaktor, n als Anzahl der zur Bestimmung des Anomalie-Schwellenwerts genutzten Proben und L_i als Rekonstruktionsverlust der i-ten zur Bestimmung des Anomalie-Schwellenwerts genutzten Probe definiert. In Evaluierungsfall 1 wird $k = 7$ gesetzt.

5.2.3 Evaluierung

Aufgrund des referentiellen Charakters der am Beispiel von Evaluierungsfall 1 erstellten Implementierung der in Kapitel 4 vorgestellten Architektur für industrielles Transfer-Lernen wird im Folgenden eine ausführliche Evaluierung sowohl der einzelnen Komponenten als auch des Gesamtalgorithmus durchgeführt.

5.2.3.1 Versuchsdurchführung

Begonnen wird in Kapitel 5.2.3.2 mit einer Evaluierung des Eingangsmoduls. Neben der Untersuchung der grundsätzlichen Funktionalität liegt der Fokus auf der Auswahl einer geeigneten Merkmalsextrakteurs-Struktur sowie Merkmalsvektorlänge für die nachfolgenden Experimente.

In Kapitel 5.2.3.3 wird anschließend das Transfermodul evaluiert. Der Fokus liegt hier auf der Robustheit des verwendeten BIRCH-Algorithmus zur Plausibilisierung der praktischen Einsatzfähigkeit im vorliegenden Anwendungsfall.

Schließlich wird in Kapitel 5.2.3.4 der komplette Lernalgorithmus evaluiert. Hier werden zum Zweck einer vergleichenden Betrachtung drei unterschiedliche Transfer-Strategien implementiert und hinsichtlich ihrer Leistungsfähigkeit auf unterschiedlichen Mengen vorliegender Zielproben mit einem Lernalgorithmus ohne Transfer-Funktionalität verglichen:

Beim **Datentransfer** wird ein zufällig initialisierter Autoencoder des (Ziel-)Ausgangsmoduls mit den (Quell-)Proben aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, und den vorliegenden Zielproben trainiert. Anschließend wird der Rekonstruktionsverlust-Schwellenwert der (Ziel-)Ausgangslogik auf den vorliegenden Zielproben kalibriert.

Beim **nachtrainierten Modelltransfer** wird der Autoencoder des (Ziel-)Ausgangsmoduls mit den (Quell-)Netzwerkparameter aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, initialisiert. Anschließend wird der Autoencoder des (Ziel-)

Ausgangsmoduls mit Zielproben nachtrainiert (Feinabstimmung). Schließlich wird der Rekonstruktionsverlust-Schwellenwert der (Ziel-)Ausgangslogik auf den vorliegenden Zielproben kalibriert.

Beim **nicht nachtrainierten Modelltransfer** wird der Autoencoder des (Ziel-)Ausgangsmoduls mit den (Quell-)Netzwerkparameter aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, initialisiert. Ein (Nach-)Training des Autoencoders des (Ziel-)Ausgangsmoduls findet nicht statt. Anschließend wird der Rekonstruktionsverlust-Schwellenwert der (Ziel-)Ausgangslogik auf den vorliegenden Zielproben kalibriert.

Dem **Lernalgorithmus ohne Transfer-Funktionalität** stehen zum Training lediglich die Proben des Zielproblems zur Verfügung. Mit diesen wird ein zufällig initialisierter Autoencoder des (Ziel-)Ausgangsmoduls trainiert.

Darüber hinaus wird die Trainingszeit der verschiedenen Lernalgorithmen miteinander verglichen.

Alle Trainings werden über 10 Epochen und einer Batch-Größe von 32 unter Nutzung des Adam-Optimizers durchgeführt. Alle Rechnungen werden auf einem Computer mit einer Intel i5-6200U CPU und einer NVIDIA GeForce 940MX 2GB GPU unter Ubuntu 20.04 ausgeführt.

5.2.3.2 Merkmalsextraktion: Ergebnisse und Diskussion

Für eine Bewertung des Eingangsmoduls zur Merkmalsextraktion werden verschiedene Versionen davon trainiert und getestet. Neben den verschiedenen Deep-Learning-Methoden LSTM, CNN und FC wird auch die Auswirkungen unterschiedlicher Merkmalsvektorlängen auf die Rekonstruktionsleistung analysiert: Der Algorithmus komprimierte den anfänglichen Eingabevektor mit 3.000 Werten in einen Vektor der Länge 50, 100 oder 150.

Für das Training dieses Experiments werden ungelabelte Druckverläufe aus der Produktion von neun verschiedenen Produkten verwendet, die jeweils tausende Male produziert wurden. Dieser Trainingsdatensatz ist somit deutlich größer als der zuvor beschriebene, gelabelte Datensatz. Nach Ausschluss der Daten von defekten Pumpen wird ein Durchschnitt aller verbleibenden Pumpen für jedes Produktionsereignis berechnet und als Eingabe verwendet.

Tabelle 5.4 zeigt die Validierungsverlust der Merkmalsextraktion in Abhängigkeit von der genutzten Merkmalsextrakteurs-Architektur und der Merkmalsvektorlänge. Die Validierungen werden mit 3.895 Proben aus der Produktion von drei weiteren, dem Algorithmus zuvor unbekannten Produkten durchgeführt. Der CNN-basierte Ansatz übertrifft die anderen um zwei Größenordnungen bei einer Codegröße von 150 und um eine Größenordnung bei einer Codegröße von 50. Der absolute Testverlust ist sehr gering, was auf eine hohe Merkmalsqualität trotz der erheblichen Reduzierung der Vektorlänge (zwischen 1:60 und 1:20) hinweist.

Tabelle 5.4: Validierungsverlust der Merkmalsextraktion nach Merkmalsextrakteurs-Architektur und Merkmalsvektorlänge

Architektur	Merkmalsvektorlänge		
	50	**100**	**150**
LSTM	4.71×10^{-3}	4.05×10^{-3}	$\mathbf{3.91 \times 10^{-3}}$
CNN	1.17×10^{-4}	1.09×10^{-4}	$\mathbf{2.73 \times 10^{-5}}$
FCNN	$\mathbf{1.96 \times 10^{-3}}$	2.35×10^{-3}	2.66×10^{-3}

Abbildung 5.5 zeigt beispielhaft derartig erstellte Merkmalsvektoren. Für jede der drei Architekturen wurden die gleichen 100 normalen und anormalen Proben der Merkmalsextraktion unterzogen. Die resultierenden Merkmalsvektoren, die aus jeweils 50 normalisierten Werten bestehen, werden in den Diagrammen aufgetragen. Auch hier wird die gute Leistung der CNN-Architektur im Vergleich zu den LSTM- oder FC-Architekturen deutlich sichtbar: Während sich die CNN-extrahierten Merkmalsvektoren deutlich ähneln und sich nur hinsichtlich einer begrenzten Anzahl von Einzelmerkmalen unterscheiden (siehe grüne Markierungen), ergibt sich bei den anderen kein klares Muster.

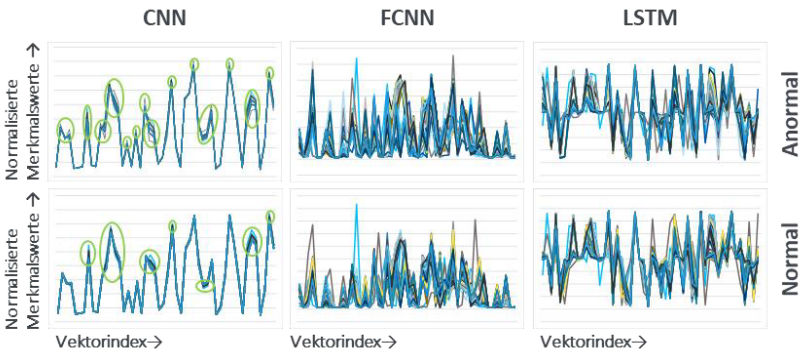

Abbildung 5.5: Jeweils 100 normalisierte normale und anormale Merkmalsvektoren der Länge 50 erstellt mit CNN-, FCNN- und LSTM-Merkmalsextrakteuren

Für die folgenden Experimente wird aufgrund ihrer guten Ergebnisse die CNN-Merkmalsextrakteursvariante genutzt.

5.2.3.3 Transferlogik: Ergebnisse und Diskussion

Für eine Bewertung der Robustheit des Transfermoduls wird dieses entsprechend den in Kapitel 5.2.3.1 definierten Fragestellungen getestet. Soweit nicht anders angegeben werden die Merk-

malsvektorlänge $L_{Merkmal} = 50$, der BIRCH-Schwellenwert $T = 0{,}6$, der BIRCH-Verzweigungsfaktor $B = 50$ (Werte experimentell bestimmt, für eine Erläuterung der Parameter siehe [146]) und pro Fertigungsvariante 100 Proben zum Training genutzt.

Tabelle 5.5 zeigt den final resultierenden Silhouetten-Index (SI) und die final resultierende Clusteranzahl nach Abschluss des Trainings des BIRCH-Clustering-Algorithmus in Abhängigkeit von der Trainingsstrategie und der Fertigungsvarianten-Sequenz (FS). Für die verschiedenen FS werden immer dieselben 10 Fertigungsvarianten in zufälliger Reihenfolge zu 10 9-schrittigen Sequenz zusammengefügt, wobei der erste Schritt aus zwei Fertigungsvarianten besteht. Die Trainingsstrategien unterscheiden sich dahingehend, dass entweder die komplette FS auf einmal („einmaliges Training“) oder dass die einzelnen Schritte separat und nacheinander („sequenzielles Training“) dem Algorithmus zugeführt werden [248].

Tabelle 5.5: Resultierender Silhouetten-Index (SI) und Clusteranzahl in Abhängigkeit von BIRCH-Trainingsstrategie und Fertigungsvarianten-Sequenz (FS)

FS	SI		Cluster-Anzahl
	Einmaliges Training	Sequenzielles Training	
1	0,728	0,728	4
2	0,659	0,659	5
3	0,659	0,659	5
4	0,718	0,718	4
5	0,619	0,619	5
6	0,575	0,575	6
7	0,623	0,623	5
8	0,728	0,728	4
9	0,662	0,662	6
10	0,728	0,728	4

Es zeigt sich, dass die Trainingsstrategie bei gleichbleibender FS keinen Einfluss auf den final resultierenden SI hat (zeilenweise Betrachtung). Die FS beeinflusst jedoch die final resultierende Cluster-Anzahl und damit den final resultierenden SI (spaltenweise Betrachtung). Weiterführende Untersuchungen ergeben, dass eine Mischung des Trainingsdatensatzes vor dem „einmaligen Training“ unabhängig von der FS in einem SI von 0,728 und einer Cluster-Anzahl von 4 resultiert [248].

Im Kontext der angestrebten Nutzung, d.h. einer sich nur vergleichsweise langsam ändernden Repräsentationsdatenbank gegenüber der neu hinzukommende Merkmalsvektoren gelabelt werden müssen, ist dieses Verhalten positiv: Bereits geclusterten Proben können weitere sequenziell hinzugefügt werden, ohne ein komplettes Neu-Training zu erfordern. Die FS ist dabei durch den

Anwendungsfall vorgegeben und ein Vergleich mit anderen FS praktisch irrelevant. Gelegentlich kann die komplette Repräsentationsdatenbank gemischt neu geclustert werden, um ggf. entstandene Suboptimalitäten aufgrund des sequenziellen Lernens zu korrigieren.

Für die FS 1 und 2 werden zur Überprüfung der Reproduzierbarkeit der Ergebnisse und damit der Determiniertheit des Clustering-Verfahrens bei gleichen Eingangsdaten und Trainings-Parametern 10 Versuchsläufe mit der Trainingsstrategie „einmaliges Training“ mit der Probenreihenfolge entsprechend der FS durchgeführt. Es ergeben sich keine Abweichungen zwischen den Versuchsläufen [248].

Weiterführende Untersuchungen ergeben, dass weder die Datendimension, d.h. die Merkmalsvektorlänge, noch die Datenmenge, d.h. die pro Fertigungsvariante genutzte Probenzahl, bei ansonsten gleichbleibenden Trainings-Parametern einen signifikanten, d.h. eine Standardabweichung $\sigma > 0{,}05$ bewirkenden, Einfluss auf den SI haben [248]. Es muss somit für ein erfolgreiches Clustering weder eine Dimensionsreduzierung durchgeführt werden noch eine besonders große Datenmenge vorhanden sein. Letzteres wird bestätigt durch die dem Mini-Batch K-Means Clustering-Algorithmus zugrunde liegenden Annahme, dass für ein aussagekräftiges Clustering-Ergebnis vielfach das Clustering einer Untermenge des Gesamtdatensatzes ausreichend ist [149].

5.2.3.4 Gesamtsystem: Ergebnisse und Diskussion

Für eine Bewertung der Leistungsfähigkeit bzw. Robustheit, der Effizienz sowie der Nützlichkeit des Clusterings im Kontext des Gesamtalgorithmus wird dieser entsprechend den in Kapitel 5.2.3.1 definierten Fragestellungen getestet. Soweit nicht anders angegeben werden die Merkmalsvektorlänge $L_{Merkmal} = 50$, der BIRCH-Schwellenwert $T = 0{,}61$ (siehe Anhang A), der BIRCH-Verzweigungsfaktor $B = 50$ (Werte experimentell bestimmt, für eine Erläuterung der Parameter siehe [146]) und pro Fertigungsvariante 300 Proben (entsprechen einer Nutzung von 100 % der zur Verfügung stehenden Zieldaten) zum Training genutzt. Es werden jeweils 10 Versuchsläufe durchgeführt und die dabei erzielten Resultate gemittelt.

Leistungsfähigkeit

Als Metrik der Leistungsfähigkeit wird auf eine Pseudo-Kappa κ_{pseudo} genannte Variante von Cohens Kappa [253] zurückgegriffen, die die Genauigkeit eines Klassifikators mit der Genauigkeit eines hypothetischen Zufallsalgorithmus, der alle Proben der Hauptklasse zuordnet, vergleicht. Pseudo-Kappa sei definiert als

$$\kappa_{pseudo} = 1 - \frac{1 - \mathrm{G}}{1 - \frac{n_{HK}}{n_{Gesamt}}}$$

mit G als der Genauigkeit des Klassifikators, n_{HK} als der Anzahl von Proben der Hauptklasse und n_{Gesamt} als der Gesamtprobenzahl. Werden alle Proben korrekt klassifiziert, ist sowohl die Genauigkeit $G = 1$ als auch $\kappa_{pseudo} = 1$ [248].

Abbildung 5.6 zeigt den Verlauf des absoluten κ_{pseudo} bei unterschiedlichen Transfer-Strategien (siehe Kapitel 5.2.3.1) in Abhängigkeit vom Anteil genutzter Zieldaten über alle Fertigungsvarianten gemittelt. In diesem Szenario werden also unterschiedliche Mengen Zielproben, hier dargestellt als Anteil tatsächlich genutzter Zielproben von den 300 hier insgesamt zur Verfügung stehenden Zielproben, je nach Transfer-Strategie entweder um Quellproben ergänzt oder in Verbindung mit auf Quellproben trainierten Quellmodellen genutzt.

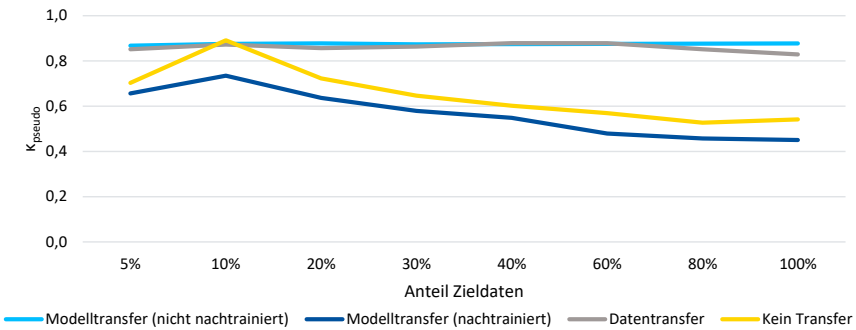

Abbildung 5.6: κ_{pseudo} bei unterschiedlichen Transfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten

Es zeigt sich, dass der Lernalgorithmus ohne Transfer-Funktionalitäten zwar den höchsten Wert für κ_{pseudo} erreicht, dabei allerdings eine starke Abhängigkeit vom Anteil Zieldaten besteht. Auch der nachtrainierte Modelltransfer verhält sich ähnlich, wenn auch jeweils auf einem reduzierten κ_{pseudo}-Niveau. Unabhängig vom Zieldaten-Anteil scheinen der nicht nachtrainierte Modelltransfer sowie der Datentransfer. Sie liegen in einem engen Band um $\kappa_{pseudo} = 0{,}87$.

Abbildung 5.7 zeigt den Verlauf der Veränderung von κ_{pseudo} bei unterschiedlichen Transfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten und setzt die Verläufe aus Abbildung 5.6 somit ins Verhältnis.

Es zeigt sich, dass der Modelltransfer (nicht nachtrainiert) durchgehend eine negative Transferleistung (siehe Kapitel 2.2.1.3) von durchschnittlich -12,6 % erbringt, während die Transfer-Funktionalität des nicht nachtrainierten Modelltransfers sowie des Datentransfers überwiegend positive Beiträge von durchschnittlich 38,2 % bzw. 35,8 % leisten.

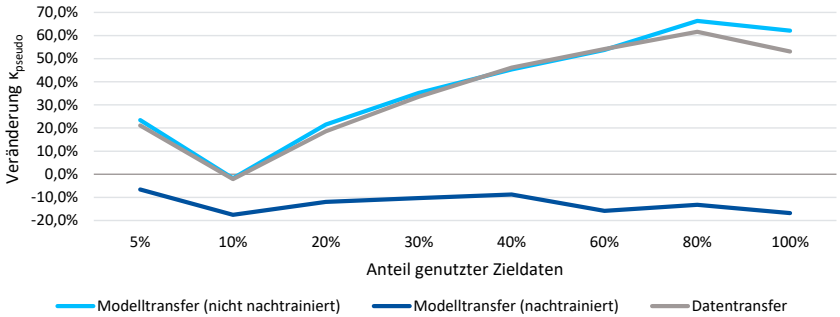

Abbildung 5.7: Veränderung von κ_{pseudo} bei unterschiedlichen Transfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten

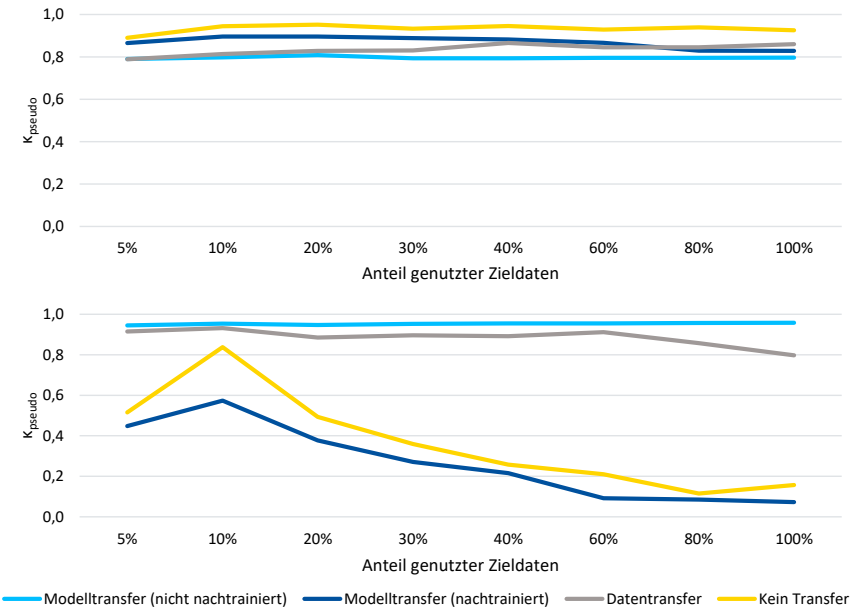

Abbildung 5.8: κ_{pseudo} bei unterschiedlichen Transfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über Fertigungsvarianten ohne (oben) bzw. mit (unten) Anomalien im Trainingsdatensatz

Abbildung 5.8 dient der weitergehenden Untersuchung dieser Ergebnisse. Es wird das κ_{pseudo} bei unterschiedlichen Transfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über Fertigungsvarianten ohne bzw. mit Anomalien im Trainingsdatensatz dargestellt.

Es zeigt sich, dass der Lernalgorithmus ohne Transfer-Funktionalitäten sowie der nachtrainierte Modelltransfer deutlich unter der Anfälligkeit von Autoencoder-basierter Anomalie-Erkennung für zu hohe Mengen anormaler Proben im Trainingsdatensatz [191–194, 254] leiden: Während der Lernalgorithmus ohne Transfer-Funktionalität für Fertigungsvarianten ohne anormale Proben im Trainingsdatensatz die höchsten κ_{pseudo}-Werte liefert, ist das Ergebnis für Fertigungsvarianten mit anormalen Proben im Trainingsdatensatz durchweg schlechter als das des nicht nachtrainierten Modelltransfers oder des Datentransfers.

Effizienz

Als Metrik der Effizienz wird die Trainingszeit der Autoencoder des Ausgangsmoduls mit aktivierter Möglichkeit zum vorzeitigen Abbruch (engl.: Early Stopping) verwendet. Verglichen werden im Folgenden nur die Trainingsstrategien nicht nachtrainierter Modelltransfer und Datentransfer mit dem Lernalgorithmus ohne Transfer-Funktionalität. Auf eine weitergehende Evaluierung des nachtrainierten Modelltransfers wird aufgrund dessen schlechter Leistungsfähigkeit im vorangehenden Abschnitt verzichtet. Da der nicht nachtrainierte Modelltransfer kein Training eines eigenständigen Ziel-Ausgangsmoduls vorsieht, wurde hier stattdessen die Trainingszeit des Cluster-spezifischen Ausgangsmoduls verwendet [248].

Abbildung 5.9 zeigt die absolute Trainingszeit bei unterschiedlichen Transfer-Strategien (siehe Kapitel 5.2.3.1) in Abhängigkeit vom Anteil genutzter Zieldaten über alle Fertigungsvarianten gemittelt. Auch hier werden also unterschiedliche Mengen Zielproben, hier dargestellt als Anteil tatsächlich genutzter Zielproben von den 300 hier insgesamt zur Verfügung stehenden Zielproben, je nach Transfer-Strategie entweder um Quellproben ergänzt oder in Verbindung mit auf Quellproben trainierten Quellmodellen genutzt.

Es zeigt sich, dass die Trainingszeit des Lernalgorithmus ohne Transfer-Funktionalität im Schnitt erheblich größer ist als die der beiden Transfer-Strategien. Während dessen Trainingszeit im Bereich zwischen 20 % und 100 % Anteil genutzter Zieldaten einigermaßen stabil bei ca. 90 s ist, fallen die Trainingszeiten für 5 % und 10 % deutlich aus der Reihe: Bei 5 % Anteil genutzter Zieldaten benötigt der Lernalgorithmus ohne Transfer-Funktionalität etwa dreimal so lange wie im Bereich zwischen 20 % und 100 %. Und bei 10 % Anteil genutzter Zieldaten benötigt er lediglich ca. ein Drittel der zwischen 20 % und 100 % üblichen Trainingszeit. Der Datentransfer benötigt über alle Zieldaten-Anteile hinweg bei geringen Abweichungen im Mittel ca. 48 s. Das Training des Clustermodells ist definitionsgemäß unabhängig vom Anteil genutzter Zieldaten und benötigt daher immer 45 s.

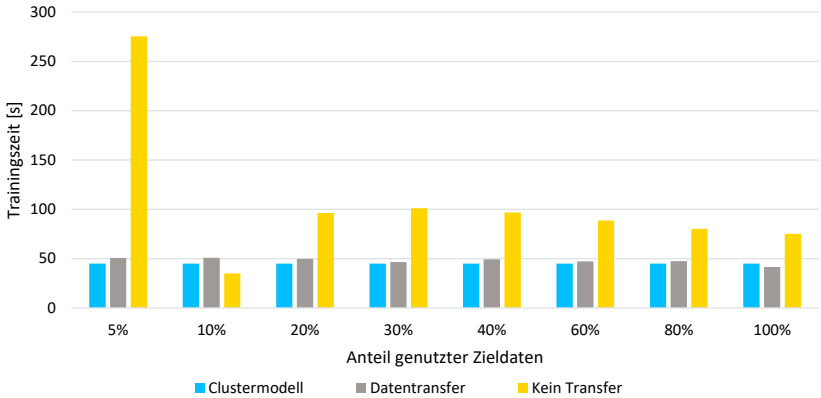

Abbildung 5.9: Absolute Trainingszeit bei unterschiedlichen Transfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten

Abbildung 5.10 zeigt die absolute Trainingszeit der Bestimmung des Rekonstruktionsverlust-Schwellenwerts in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten. Diese Bestimmung ist unabhängig von der gewählten Transfer-Strategie immer notwendig.

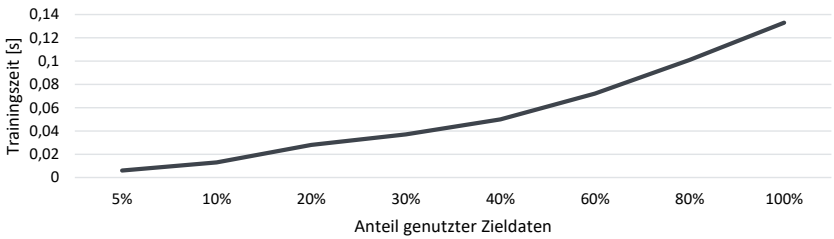

Abbildung 5.10: Absolute Trainingszeit der Bestimmung des Rekonstruktionsverlust-Schwellenwerts in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten

Es zeigt sich, dass ihr Training erwartbarer Weise mit steigender Zieldaten-Menge ebenfalls zunimmt. Absolut spielt sie jedoch verglichen mit der Trainingszeit des Autoencoders des Ausgangsmoduls keine Rolle.

Nützlichkeit Clustering

Zur Bewertung der Nützlichkeit des Clusterings wird erneut auf die Genauigkeitsmetrik Pseudo-Kappa κ_{pseudo} zurückgegriffen. Betrachtet werden im Folgenden nur verschiedene Varianten der Trainingsstrategie nicht nachtrainierter Modelltransfer.

Abbildung 5.11 zeigt den Verlauf des absoluten κ_{pseudo} bei unterschiedlichen Graden der Ähnlichkeits-basierten Auswahl von Transfer-Fällen in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten. Dabei werden die trainierten Clustermodelle einmal nur auf Fertigungsvarianten desselben Clusters angewandt („Clustering“) und einmal auf alle Fertigungsvarianten.

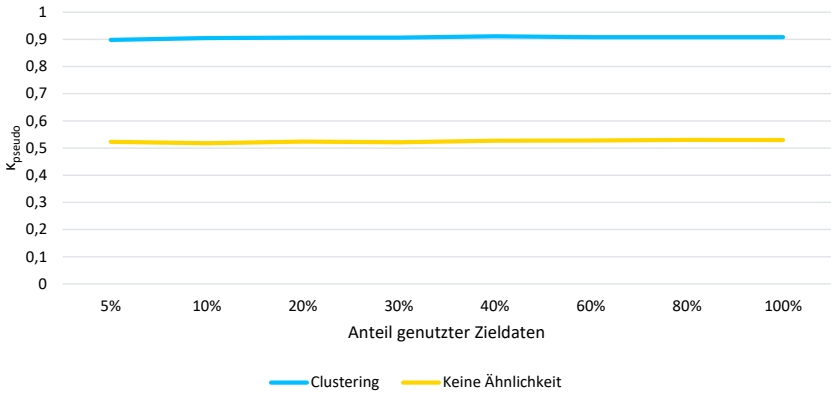

Abbildung 5.11: κ_{pseudo} bei unterschiedlichen Graden der Ähnlichkeits-basierten Auswahl von Transfer-Fällen in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Fertigungsvarianten

Es zeigt sich, dass die im Clustering stattfindende Ähnlichkeits-basierte Auswahl von Transfer-Fällen im Schnitt eine deutliche Verbesserung gegenüber einer zufälligen Auswahl von Transfer-Fällen erzielt. Zwar ergibt eine Detail-Analyse vereinzelt Fälle einer verbesserten Leistungsfähigkeit, jedoch sind diese rar und insbesondere im Kontext unüberwachten Lernens nicht robust zu detektieren.

5.2.4 Bewertung

Im Kontext von Evaluierungsfall 1 ergeben sich folgende Ergebnisse:

Die Merkmalsextraktion mittels CNN liefert reproduzierbar auch bei starker Kompression der Eingangsdatenvektoren Merkmalsvektoren, die zu einem geringen Rekonstruktionsverlust führen.

Die Transferlogik kann aufgrund des Clusterings mittels des BIRCH-Algorithmus einfach und reproduzierbar Ähnlichkeitsbestimmungen durchführen und auf dieser Basis geeignete Transfer-Datensätze zur Initialisierung bzw. zum Training von (Ziel-)Ausgangsmodulen auswählen.

Das resultierende Gesamtsystem erweist sich als robust gegenüber in Bezug auf die Menge enthaltener anormaler Proben unausgewogenen Trainingsdatensätzen – einem gängigen [4, 31, 49] und auch in diesem Fall vorliegenden Problem der Anomaliedetektion mittels Autoencodern. Dazu werden die Transfer-Strategien Datentransfer und nicht nachtrainierter Modelltransfer genutzt, die zudem auch eine Reduzierung der nötigen Trainingszeit bewirken. Dabei hat das Clustering einen zentralen Anteil an der Leistungsfähigkeit des Gesamtsystems.

Die besten Resultate erbringt der nicht nachtrainierte Modelltransfer, bei dem ein auf Quellproben aus dem Cluster, dem die meisten Zielproben zugeordnet werden konnten, trainiertes Ausgangsmodul unverändert auf das Zielproblem angewendet wird. Abhängig von der Menge zur Verfügung stehender Zielproben können so Verringerungen des MSE von bis zu 66% erzielt werden. Die Vorhersagequalität dieses Gesamtsystems ist dabei über den gesamten Versuchsbereich hinweg sehr stabil.

Die im Rahmen von Evaluierungsfall 1 durchgeführten Untersuchungen ergeben somit, dass der dabei erstellte Prototyp als Lösungsartefakt alle vier Anforderungen zur Beantwortung der Forschungsfrage erfüllt, wobei der Fokus auf Anforderung 1 liegt: Das resultierende Gesamtsystem steigert die Robustheit des Lernalgorithmus und macht je nach gewählter Transfer-Strategie ein Nachtrainieren auf neuen (Teil-)Problemen überflüssig, wenn die Zielproben sich einem bereits bestehenden Probencluster in der Repräsentationsdatenbank zuordnen lassen. Darüber hinaus ist durch die Nutzung der Architektur der Aufwand des Nachtrainierens, wenn doch nötig, bei neuen (Teil-)Probleme deutlich reduziert (A2), es werden dazu in der Repräsentationsdatenbank hinterlegte Quellmerkmalsvektoren weitergenutzt (A3) und weite Teile der Codebasis über die verschiedenen Teilprobleme beibehalten (A4). Für eine tiefergehende Analyse hinsichtlich der Beantwortung der Forschungsfrage sei auf Kapitel 5.5 verwiesen.

5.3 Evaluierungsfall 2: Ausfallvorhersage für elektromechanische Relais

Evaluierungsfall 2 dient als zweiter Anwendungsfall der Referenz-Implementierung der in Kapitel 4 vorgestellten Architektur für industrielles Transfer-Lernen und soll darüber hinaus die umfangreichen Möglichkeiten zur Wiederverwendung von Architekturelementen demonstrieren. Auf einem Datensatz zum Verschleißverhalten von industriellen Standardkomponenten soll basierend auf einer Ähnlichkeitsanalyse Wissen von bekannten Betriebszuständen bzw. Herstellern hin zu neuen, bisher unbekannten Betriebszuständen bzw. Herstellerübertragen werden, um eine effizientere Ausfallvorhersage mittels neuronaler Netze zu ermöglichen. Zu diesem Zweck werden alle in Kapitel 4 genannten Komponenten prototypisch realisiert.

Dabei liegt Evaluierungsfall 2 die Annahme zugrunde, dass sowohl Betriebszustand als auch Hersteller der Komponente bekannt sind. Darüber hinaus sind sowohl die Datensätze der bereits bekannten als auch die der neuen Problemvarianten gelabelt. Es handelt sich also um überwachtes Lernen.

5.3.1 Anwendungsfall

In Evaluierungsfall 2 wird die Ausfallvorhersage für elektromechanische Reed-Relais auf Basis von Spannungsverläufen, dem Kontaktwiderstand sowie verschiedenen Betriebsparametern betrachtet. Die Relais werden von einem kleinen Steuerstrom geschalten, um einen größeren Laststrom entweder zu blockieren oder freizugeben. Der Schaltvorgang selbst basiert dabei auf der vom Steuerstrom induzierten magnetischen Beeinflussung einer ferromagnetischen Kontaktzunge [255, 256].

Bei jedem Schaltvorgang gibt es einen kurzen Zeitbereich, indem der Luftspalt zwischen Kontaktzunge und Auflagepunkt der Kontaktzunge nicht mehr groß genug ist, um ein Überschlagen der Spannung zu verhindern. Es kommt dabei kurzzeitig zu einem Lichtbogen, der (auf Dauer) die Kontaktflächen beschädigt, was sich in einer veränderten Kontaktflächenbeschaffenheit mit Auswirkung den Kontaktwiderstand äußert und im schlimmsten Fall zum Zusammenschweißen der Kontaktflächen und damit zum Ausfall des Relais führen kann. Das Ausmaß der Beschädigung ist unter anderem von Lastspannung und Laststrom abhängig [255–257].

Zwar sind Relais selbst in der Regel günstige Standardkomponenten, jedoch führt ihr Ausfall aufgrund ihrer Rolle als zentrales Leistungsschaltelement einer Maschine vielfach auch zum Ausfall der Maschine insgesamt. Der Hauptkostenfaktor bei einem Relais-Defekt ist damit nicht das Relais selbst, sondern die reduzierte Maschinenverfügbarkeit. Eine möglichst genaue, Anwendungsfall-spezifische Vorhersage des Relais-Verschleißes ist damit wünschenswert, um rechtzeitig einen Austausch vornehmen und ungeplante Ausfälle der Maschine vermeiden zu können [258].

Trotzdem steht die datengetriebene Ausfallvorhersage für elektromechanische Relais vor der Herausforderung, dass Relaisverschleiß in hohem Maße von den Betriebsbedingungen abhängig ist [259], nur in seltenen Fällen hochaufgelöste Messdaten über das Relaisverschleißverhalten erhoben werden und diese dann nur einen Bruchteil der möglichen Betriebsbedingungen abdecken können. Damit stehen Relais stellvertretend für viele andere, zentrale Standardkomponenten in der Automatisierungstechnik.

5.3.1.1 Datensatz

Im Rahmen von Evaluierungsfall 2 wird auf einen selbsterhobenen Datensatz zurückgegriffen. Dieser Datensatz ist inklusive einer ausführlichen Beschreibung unter [258] frei verfügbar.

Zum diesem Zweck wurde am IAS ein Messstand (siehe Abbildung 5.12) entwickelt, der den Verschleiß elektromechanischer Relais mittels der Messung verschiedener Betriebsparameter dokumentieren kann [260–262]. Dazu sollen verschiedene Relais parallel bis zu ihrem Ausfall verschiedenen Lasten schalten. Jedes Relais wird eingangsseitig von einem Mikrocontroller gesteuert und ausgangsseitig mit einer Last beschaltet. Der Lastkreis wird von einer separaten, variablen Spannungsversorgung betrieben [258].

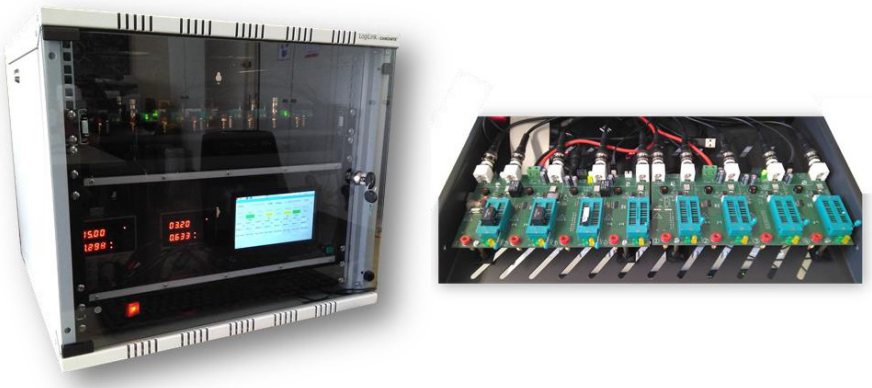

Abbildung 5.12: Relais-Verschleiß-Messstand – Gesamtansicht (li.) und Messschublade (re.)

Die am Relais anliegende Spannung wird mit einem USB-Oszilloskop automatisch als Spannung über der Last gemessen. Der Kontaktwiderstand des Relais wird in der Messschaltung indirekt über den Spannungsabfall am Schaltkontakt im geschlossenen Zustand, sowie den Stromfluss durch den Schaltkontakt bestimmt [258].

Das Kernstück der Anlage bildet ein PC-System in Form eines Single-Board Computers, welcher alle Abläufe steuert und die Speicherung der erhobenen Daten übernimmt. Über einen Touchscreen stellt er zudem eine Bedienoberfläche für die Interaktion mit dem Nutzer zur Verfügung. Ein Mikrocontroller übernimmt die Hardwareansteuerung der Anlage. Für die Spannungsversorgung der Lasten kommen zwei fernsteuerbare Labornetzteile zum Einsatz. Labornetzteile liefern für den Einsatz in der Anlage die nötige Genauigkeit und Stabilität der Ausgangsspannung. Alle Komponenten sind in einem Gehäuse untergebracht [258].

Mit diesem Messstand wurden 100 Relais vergleichbarer Leistungsklasse von fünf verschiedenen Herstellern an 22 verschiedenen Betriebspunkte definiert über die Versorgungsspannung und den Lastwiderstand bis zum Ausfall abwechselnd geöffnet und geschlossen. Dabei wurde für jeden 50. Zyklus der Spannungsverlauf bei Öffnung und Schließung des Lastkreises des jeweiligen Relais sowie der Kontaktwiderstand gemessen. Insgesamt ergab dies 32449 Proben. Für jedes Relais

wurden auch die entsprechenden Meta-Daten, d.h. insbesondere die Zyklenzahl bis zum Ausfall, die Versorgungsspannung und der Lastwiderstand, gespeichert [258, 263].

5.3.1.2 Datenvorverarbeitung

Im Rahmen von [263] wurden zuerst aus den Spannungsverläufen bei Öffnung und Schließen des Lastkreises die Öffnungs- und Schließverzögerung abgeleitet. Diese wurden definiert als Zeit zwischen Schaltsignal und über- bzw. unterschreiten der 50%-Schwelle der Lastspannung. Anschließend wurden die Datensätze anhand der bekannten Position einzelner Proben relativ zur Gesamtlebensdauer bis zum Ausfall gelabelt und auf die jeweils theoretischen Maximalwerte normalisiert.

5.3.2 Realisierung

Im Rahmen von [263] wurden verschiedene Varianten eines von der in Kapitel 4 vorgestellten Architektur abgeleiteten, industriellen Transfer-Lernalgorithmus zur Ausfallvorhersage (vgl. Kapitel 3.2.2.1) entworfen. Dabei kamen die Frameworks PyTorch 1.7.0 [250] und scikit-learn 0.23.2 [239] unter Python 3.6 [251] zum Einsatz.

Aufgrund der Nutzung von Evaluierungsfall 2 als zweiten Anwendungsfall der in Kapitel 5.2 vorgestellten Referenz-Implementierung konnten wesentliche Komponenten übernommen werden. Dazu gehört insbesondere die Realisierung des Transfermoduls.

Abbildung 5.13 gibt einen schematischen Überblick über die dabei genutzte Realisierung der Lern-Architektur. Wesentliche Abweichungen zur in Evaluierungsfall 1 genutzten Implementierung finden sich in Eingangsvektor und -Modul sowie im Ausgangsmodul:

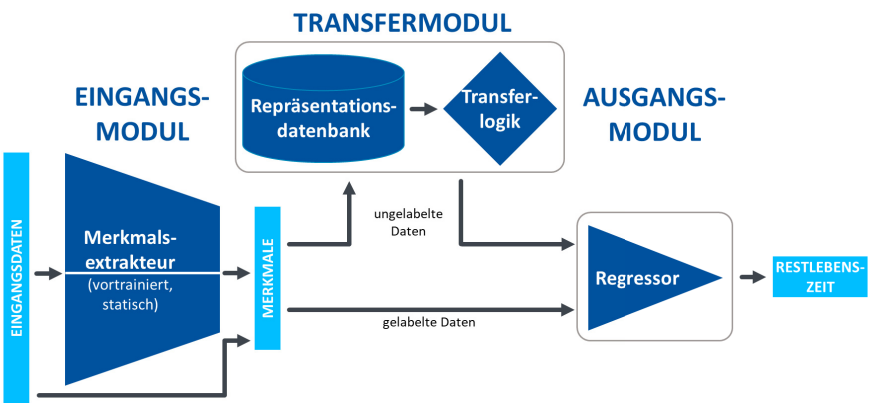

Abbildung 5.13: Schematische Darstellung der Realisierung der Lern-Architektur in Evaluierungsfall 2

Der Eingangsvektor besteht in Evaluierungsfall 2 aus zwei Teil-Vektoren, dem Spannungsverlauf bei Öffnung und Schließung des Vektors, sowie aus fünf Skalaren: der Zykluszähler, dem Kontaktwiderstand, dem Laststrom sowie der Schließ- und der Öffnungsverzögerung.

Das Eingangsmodul muss diese Veränderungen widerspiegeln. Es besteht daher aus zwei identischen Merkmalsextrakteuren, die die beiden Teil-Vektoren im Eingangsdatenvektor auf deutlich kürzere Teil-Merkmalsvektoren reduzieren. Die Skalare aus dem Eingangsdatenvektor werden unverändert an den Gesamt-Merkmalsvektor angehängt.

Das Ausgangsmodul muss der veränderten Problemstellung in Evaluierungsfall 2 Rechnung tragen. Es ist daher zur Vorhersage eines kontinuierlichen Ausgangswerts, der Restlebenszeit eines Relais, als Regressor ausgeführt. Eine darüberhinausgehende Ausgangslogik ist nicht erforderlich.

5.3.2.1 Eingangsmodul: Merkmalsextraktion

Das Eingangsmodul wurde analog zu dem Eingangsmodul von Evaluierungsfall 1 (siehe Kapitel 5.2.2.1) realisiert. Änderungen ergeben sich jedoch durch den veränderten Eingangsdatenvektor.

Tabelle 5.6: Kodierer-Struktur des Merkmalsextrakteurs in Evaluierungsfall 2

Kodierer-Struktur
3-Schichtiges CNN Schicht 1: $1 \times \frac{L_{Merkmal}}{3} \times 10, Stride = 5$ Schicht 2: $\frac{L_{Merkmal}}{3} \times \frac{L_{Merkmal}}{2} \times 5, Stride = 2$ Schicht 3: $\frac{L_{Merkmal}}{2} \times L_{Merkmal} \times 2, Stride = 2$ Flatten + 1-Schichtiges FCNN Legende: Eingang x Ausgang x Kernel

Tabelle 5.6 gibt einen Überblick über die Struktur des verwendeten Merkmalsextrakteurs. Eine systematische Hyperparameter-Optimierung findet aufgrund des damit verbundenen Rechenaufwands nicht statt, stattdessen werden die Hyperparameter basierend auf Erfahrungswerten ausgewählt. Dies trägt auch dem Umstand Rechnung, dass in dieser Arbeit der Fokus auf dem Beleg der grundsätzlichen Anwendbarkeit des Konzepts und nicht auf der Findung einer optimalen Lösung für den spezifischen Anwendungsfall, hier die Ausfallvorhersage von elektromechanischen Relais, liegt. Aufgrund der einheitlichen Struktur der Eingangsdaten beträgt die Eingangsvektorlänge in Evaluierungsfall 2 immer 305. Als Aktivierungsfunktion kommen innerhalb des Kodierers ReLU und an seinem Ausgang SELU zum Einsatz. Das einschichtige FCNN hat die Länge

der Merkmalsvektoren $L_{Merkmal}$, Padding und Dilation kommen im CNN nicht zum Einsatz. Als Bewertungskriterium kommt der MSE zum Einsatz.

5.3.2.2 Ausgangsmodul: Regressor

Entsprechend den Ausführungen in Kapitel 3.2.1.4 können verschiedene Deep-Learning-Architekturen zur Ausfallvorhersage eingesetzt werden. In Evaluierungsfall 2 wird auf einen Regressor realisiert mittels eines simplen, vierschichtigen FCNN zurückgegriffen.

Tabelle 5.7 gibt einen Überblick über die Struktur des verwendeten Regressors. Aufgrund der potentiell unterschiedlichen Längen der Merkmalsvektoren ist eine Definition der Struktur in Abhängigkeit von dieser Merkmalsvektorlänge $L_{Merkmal}$ sinnvoll – in Kapitel 5.3.3 kommen jedoch ausschließlich Regressoren für $L_{Merkmal} = 305$ zum Einsatz. Die letzte Schicht des FCNN hat die Knotenanzahl 1. Als Aktivierungsfunktion werden ReLU genutzt. Als Bewertungskriterium kommt der MSE zum Einsatz.

Tabelle 5.7: Aufbau des Regressors zur Ausfallvorhersage in Evaluierungsfall 2

Regressor-Struktur
4-Schichtiges FCNN
Schicht 1: $\frac{5}{4} \cdot L_{Merkmal}$ Knoten
Schicht 2: $\frac{5}{4} \cdot L_{Merkmal}$ Knoten
Schicht 3: $\frac{13}{12} \cdot L_{Merkmal}$ Knoten

5.3.3 Evaluierung

Aufgrund der bereits an Evaluierungsfall 1 erfolgten, grundsätzlichen Evaluierung der vorgestellten Referenz-Implementierung der in Kapitel 4 vorgestellten Architektur für industrielles Transfer-Lernen wird im Folgenden lediglich der Gesamtalgorithmus in Bezug auf die Erfordernisse des Evaluierungsfalls 2 getestet. Ein ausführlicher Test der einzelnen Komponenten des Gesamtalgorithmus findet nicht erneut statt.

5.3.3.1 Versuchsdurchführung

In Kapitel 5.3.3.2 wird zuerst der komplette Lernalgorithmus basierend auf Datentransfer-Ansätzen evaluiert. Hier werden zum Zweck einer vergleichenden Betrachtung zwei unterschiedliche Transfer-Strategien implementiert und hinsichtlich ihrer Vorhersagequalität auf unterschiedlichen Mengen vorliegender Zielproben (für einen Überblick über den Test-Datensatz siehe Tabelle 5.8) mit einem Lernalgorithmus ohne Transfer-Funktionalität verglichen:

Tabelle 5.8: Überblick über die Relais-Test-Datensätze für Datentransfer-basierte Ansätze in Evaluierungsfall 2

Hersteller	Relais	Betriebsbedingung	Anzahl Proben
A	A98	Versorgungsspannung: 20 V Lastwiderstand: 32 Ω	603
	A97		51
	A96		210
B	B73	Versorgungsspannung: 20 V Lastwiderstand: 23,5 Ω	169
	B72		340
	B71		591
C	C44	Versorgungsspannung: 20 V Lastwiderstand: 25 Ω	177
	C43		812
	C42		578
D	D22	Versorgungsspannung: 20 V Lastwiderstand: 21 Ω	507
	D21		526
	D20		90
E	E9	Versorgungsspannung: 10 V Lastwiderstand: 8,5 Ω	574
	E10		19
	E1		332

Beim **Datentransfer mit ähnlichen Daten** wird ein zufällig initialisierter Regressor des (Ziel-) Ausgangsmoduls mit den (Quell-)Proben aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, und den vorliegenden Zielproben trainiert.

Beim **Datentransfer mit allen Daten** wird ein zufällig initialisierter Regressor des (Ziel-) Ausgangsmoduls mit allen (Quell-)Proben aus der Repräsentationsdatenbank und den vorliegenden Zielproben trainiert.

Dem **Lernalgorithmus ohne Transfer-Funktionalität** stehen zum Training lediglich die Proben des Zielproblems zur Verfügung. Mit diesen wird ein zufällig initialisierter Regressor des (Ziel-) Ausgangsmoduls trainiert.

Anschließend wird der komplette Lernalgorithmus basierend auf Modelltransfer-Ansätzen evaluiert. Erneut werden zum Zweck einer vergleichenden Betrachtung zwei unterschiedliche Transfer-Strategien implementiert und hinsichtlich ihrer Vorhersagequalität auf unterschiedlichen Mengen vorliegender Zielproben (für einen Überblick über den Test-Datensatz siehe Tabelle 5.9) mit einem Lernalgorithmus ohne Transfer-Funktionalität verglichen:

Tabelle 5.9: Überblick über die Relais-Test-Datensätze für Modelltransfer-basierte Ansätze in Evaluierungsfall 2

Hersteller	Relais	Betriebsbedingung	Anzahl Proben
A	A98	Versorgungsspannung: 20 V Lastwiderstand: 32 Ω	603
	A97		51
	A96		210
	A95		805
B	B74	Versorgungsspannung: 20 V Lastwiderstand: 23,5 Ω	228
	B73		169
	B72		340
	B71		591
C	C44	Versorgungsspannung: 20 V Lastwiderstand: 25 Ω	177
	C43		812
	C42		578
	C60		258
D	D23	Versorgungsspannung: 20 V Lastwiderstand: 21 Ω	590
	D22		507
	D21		526
	D20		90
E	E9	Versorgungsspannung: 10 V Lastwiderstand: 8,5 Ω	574
	E10		19

Beim **Modelltransfer** wird der Regressor des (Ziel-)Ausgangsmoduls mit den (Quell-)Netzwerkparameter aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, initialisiert. Anschließend wird er mit Zielproben nachtrainiert (Feinabstimmung).

Beim **Modell-/Datentransfer** stellt eine Mischung aus Modell- und Datentransfer dar. Hier wird der Regressor des (Ziel-)Ausgangsmoduls mit den (Quell-) Netzwerkparameter aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, initialisiert. Anschließend wird er mit den (Quell-)Proben aus dem Cluster in der Repräsentationsdatenbank, dem die meisten Zielproben zugeordnet werden, und den vorliegenden Zielproben nachtrainiert (Finetuning).

Dem **Lernalgorithmus ohne Transfer-Funktionalität** stehen zum Training erneut lediglich die Proben des Zielproblems zur Verfügung. Mit diesen wird ein zufällig initialisierter Regressor des (Ziel-)Ausgangsmoduls trainiert.

Alle Trainings werden über 5000 Epochen, mit der aktivierten Möglichkeit zum vorzeitigen Abbruch des Trainings und damit gewöhnlich über eine zweistellige Epochenzahl, mit einer Lernrate von 0,001 und einer Batch-Größe von 5 durchgeführt. Alle Rechnungen werden auf einem Computer mit einer AMD Ryzen Threadripper 2920X CPU und einer NVIDIA GeForce RTX 2080 8GB GPU unter Ubuntu 20.04 ausgeführt.

5.3.3.2 Gesamtsystem: Ergebnisse und Diskussion

Für eine Bewertung der Vorhersagequalität des Gesamtalgorithmus wird dieser entsprechend den in Kapitel 5.3.3.1 definierten Fragestellungen getestet. Soweit nicht anders angegeben, werden die Merkmalsvektorlänge $L_{Merkmal} = 305$ (die Teil-Vektorlänge für den Spannungsverlauf bei Öffnung und Schließung eines Relais beträgt somit 150), der BIRCH-Schwellenwert $T = 2{,}33$ (siehe Anhang A), der BIRCH-Verzweigungsfaktor $B = 50$ (Werte experimentell bestimmt, für eine Erläuterung der Parameter siehe [146]) und pro Relais alle zur Verfügung stehenden Proben (entsprechen einer Nutzung von 100 % der verfügbaren Zieldaten) zum Training genutzt. Als Bewertungskriterium kommt der MSE zum Einsatz.

Datentransfer-basierte Ansätze

Für eine Bewertung der Vorhersagequalität des Gesamtalgorithmus bei datenbasiertem Transfer-Lernen werden die drei in Kapitel 5.3.3.1 beschriebenen Transfer-Strategien verglichen. Es werden jeweils 3 Versuchsläufe durchgeführt und die dabei erzielten Resultate gemittelt.

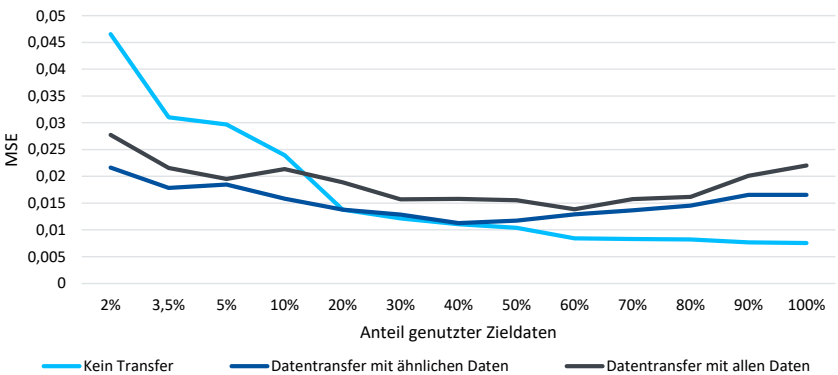

Abbildung 5.14: MSE bei unterschiedlichen Datentransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais

Abbildung 5.14 zeigt den Verlauf des MSE bei den genannten Datentransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais. In diesem Szenario werden also unterschiedliche Mengen Zielproben, hier dargestellt als Anteil tatsächlich genutzter

Zielproben von den maximal zur Verfügung stehenden Zielproben, je nach Transfer-Strategie um Quellproben ergänzt genutzt.

Es zeigt sich, dass der Lernalgorithmus ohne Transfer-Funktionalitäten zwar den insgesamt niedrigsten MSE erreicht, dabei allerdings eine starke Abhängigkeit vom Anteil Zieldaten besteht. Für geringe Anteile Zieldaten lässt die Vorhersagequalität stark nach. Demgegenüber bieten beide Datentransfer-Ansätze einen konstanteren MSE, der für Zieldaten-Anteile größer 10 % bzw. 20 % höher als beim Lernalgorithmus ohne Transfer-Funktionalität ist, bei geringeren Zieldaten-Anteilen aber niedriger ausfällt. Der Datentransfer mit ähnlichen Daten ist dabei durchgehend besser als der Datentransfer mit allen Daten.

Abbildung 5.15 zeigt den Verlauf der Veränderung des MSE bei den genannten Datentransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais und setzt die Verläufe aus Abbildung 5.14 somit ins Verhältnis.

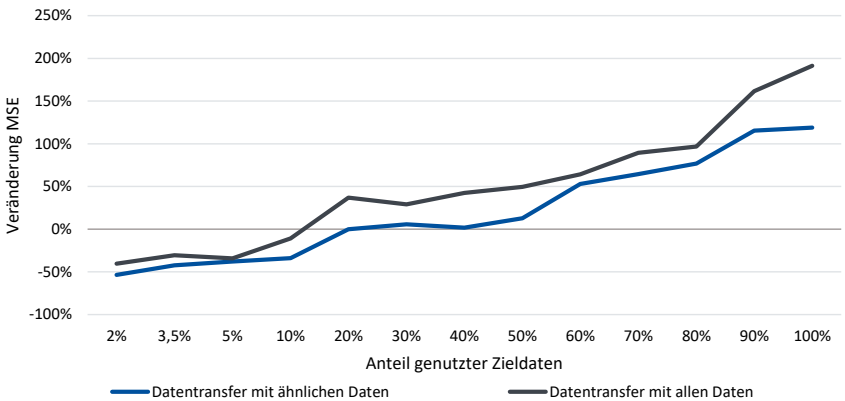

Abbildung 5.15: Veränderung des MSE bei unterschiedlichen Datentransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais

Es zeigt sich, dass der Datentransfer mit ähnlichen Daten auf sehr kleinen Zieldaten-Anteilen bis ca. 50 % Verbesserung erbringt und sich zumindest bis ca. 40 % Zieldatenanteil nicht stark negativ auswirkt. Demgegenüber erbringt der Datentransfer mit allen Daten maximal 40 % Verbesserung und wirkt sich bereits bei einem Zieldaten-Anteil von 20 % ähnlich stark negativ aus.

Modelltransfer-basierte Ansätze

Für eine Bewertung der Vorhersagequalität des Gesamtalgorithmus bei modellbasiertem Transfer-Lernen werden die drei in Kapitel 5.3.3.1 beschriebenen Transfer-Strategien verglichen. Es werden jeweils 10 Versuchsläufe durchgeführt und die dabei erzielten Resultate gemittelt.

Abbildung 5.16 zeigt den Verlauf des MSE bei den genannten Modelltransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais. Auch hier werden also unterschiedliche Mengen Zielproben, hier dargestellt als Anteil tatsächlich genutzter Zielproben von den maximal zur Verfügung stehenden Zielproben, je nach Transfer-Strategie in Verbindung mit auf Quellproben trainierten Quellmodellen genutzt oder um Quellproben ergänzt und in Verbindung mit auf Quellproben trainierten Quellmodellen genutzt. Das zu transferierende Quellmodell erzielt auf dem Zielproblem ohne Nachtraining einen durchschnittlichen MSE von 0,076925.

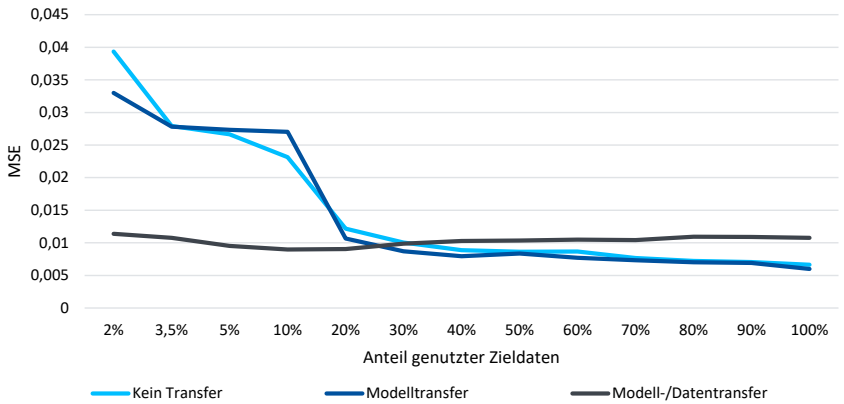

Abbildung 5.16: MSE bei unterschiedlichen Modelltransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais

Es zeigt sich, dass in diesem Szenario der Modelltransfer-Ansatz minimal besser ist als der Lernalgorithmus ohne Transfer-Funktionalitäten. Beide Verläufe sind sehr ähnlich und weisen eine starke Abhängigkeit vom Anteil Zieldaten auf. Für geringe Anteile Zieldaten lässt die Vorhersagequalität stark nach. Demgegenüber bietet die Kombination aus Modell- und Datentransfer einen sehr konstanten MSE, der für Zieldaten-Anteile größer 30 % höher als beim Lernalgorithmus ohne Transfer-Funktionalität ist, bei geringeren Zieldaten-Anteilen aber erheblich niedriger ausfällt. Auch verglichen mit einer reinen Datentransfer-Strategie ist die Kombination aus Modell- und Datentransfer aufgrund ihrer besseren Vorhersagequalität vorteilhaft.

Abbildung 5.17 zeigt den Verlauf der Veränderung des MSE bei den genannten Modelltransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais und setzt die Verläufe aus Abbildung 5.16 somit ins Verhältnis.

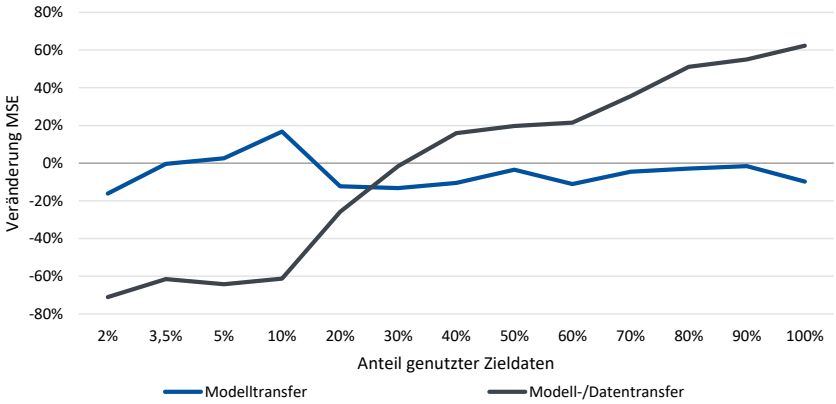

Abbildung 5.17: Veränderung des MSE bei unterschiedlichen Modelltransfer-Strategien in Abhängigkeit vom Anteil genutzter Zieldaten gemittelt über alle Ziel-Relais

Es zeigt sich, dass die Kombination aus Modell- und Datentransfer auf sehr kleinen Zieldaten-Anteilen bis ca. 10 % erhebliche Verbesserungen von > 60 % MSE-Reduktion erbringt und sich zumindest bis ca. 30 % Zieldatenanteil nicht negativ auswirkt. Der reine Modelltransfer-Ansatz hingegen zeigt kein eindeutig besseres Verhalten, sondern schwankt im Bereich zwischen -20 % und 20% MSE-Veränderung.

5.3.4 Bewertung

Im Rahmen der Versuche zu Evaluierungsfall 2 ergeben sich folgende Ergebnisse:

Das Gesamtsystem erzielt abhängig von der gewählten Transferstrategie auch auf dem Anwendungsfall Ausfallvorhersage für elektromechanische Relais eine erhebliche Verbesserung der Vorhersagequalität. Die dazu nötigen Anpassungen des Algorithmus beziehen sich lediglich auf die Behandlung der veränderten Eingangsdatenvektoren sowie ein verändertes Ausgangsmodul.

Die besten Resultate erbringt eine Mischung aus Modell- und Datentransfer, bei der ein auf Quellproben vortrainiertes Quellmodell mit einer Mischung aus Quell- und Zielproben nachtrainiert wird. Für geringe Mengen zur Verfügung stehender Zielproben können so Verringerungen des MSE von bis zu 71% erzielt werden. Die Vorhersagequalität dieses Gesamtsystems ist dabei über den gesamten Versuchsbereich hinweg sehr stabil.

Die im Rahmen von Evaluierungsfall 2 durchgeführten Untersuchungen ergeben somit, dass der dabei erstellte Prototyp als Lösungsartefakt drei Anforderungen zur Beantwortung der Forschungsfrage erfüllt: Das resultierende Gesamtsystem ist hinsichtlich seiner Vorhersagequalität

unabhängig vom Anteil genutzter Zielproben (A1), benötigt weniger Zielproben, um erfolgreich nachtrainiert zu werden (A2), nutzt dazu in der Repräsentationsdatenbank hinterlegte Quellmerkmalsvektoren weiter (A3), verwendet grundsätzlich aus verschiedenen Datenarten bestehende Eingangsdaten (A3), behält weite Teile der Codebasis über die verschiedenen Teilprobleme bei und teilt sich große Abschnitte der Codebasis mit dem Gesamtsystem aus Evaluierungsfall 1 (A4). Für eine tiefergehende Analyse hinsichtlich der Beantwortung der Forschungsfrage sei auf Kapitel 5.5 verwiesen.

5.4 Evaluierungsfall 3: Ausfallvorhersage auf Vibrationsdaten von Kugellagern

Evaluierungsfall 3 liegt die Annahme zugrunde, dass für einzelne Varianten eines Problems gelabelte Trainingsdaten vorliegen. Die Aufgabe ist nun, für andere Varianten desselben Problems eine Lösung auf Basis ungelabelter Daten zu finden. Es handelt sich dabei also um unüberwachtes Lernen.

Hierbei soll industrielles Transfer-Lernen auf Basis einer Domänen-Adaption (vgl. Kapitel 3.1.2.1) zum Einsatz kommen. Aufgrund der geringen Bandbreite verfügbarer Regressions-Datensätze konnte auf eine Clustering-basierte Ähnlichkeitsanalyse der zu transferierenden Daten verzichtet werden. Das Transfermodul wird daher lediglich als einfache Repräsentationsdatenbank realisiert. Die Auswahl der zu transferierenden Daten geschieht im Rahmen der Versuchsplanung manuell.

5.4.1 Anwendungsfall

In Evaluierungsfall 3 wird die Ausfallvorhersage für Kugellager auf Basis von Vibrationsdaten betrachtet.

Kugellager sind eine Sonderform der Wälzlager, bei denen zwischen Innenring und Außenring Kugeln zur Reduzierung des Reibungswiderstands zum Einsatz kommen [264]. Sie dienen der Fixierung von Achsen und Wellen und sind eine massenproduzierte Standardkomponente im Maschinenbau [265, 266]. Aufgrund ihrer weiten Verbreitung in unterschiedlichsten Belastungskontexten und der teils großen Auswirkungen von Ausfällen (bspw. Maschinenstillstand oder sogar Schäden an anderen Anlagen oder Menschen) ist eine Vermeidung von Ausfällen wünschenswert [265–267]. Ein Indikator für den Zustand eines Kugellagers sind dabei die im Betrieb auftretenden Vibrationen, die mit fortschreitendem Verschleiß ebenfalls zunehmen [265–267]. Vibrationen sind als periodische Beschleunigungen unterschiedlicher Frequenz in verschiedene Raumrichtungen einfach messbar.

Die datengetriebene Ausfallvorhersage für Kugellager steht jedoch vor zwei Herausforderungen: Einerseits ist die Lebenszeit eines Kugellagers stark vom jeweilig individuellen Belastungskontext abhängig. Und andererseits sollen Ausfälle üblicherweise vermieden werden, wodurch für viele Belastungskontexte kaum oder keine gelabelten Trainingsdaten zur Verfügung stehen – denn die Restlebensdauer eines Kugellagers, dass vor dem Auftreten eines Defekts ausgetauscht wird, ist prinzipiell nicht bestimmbar [236].

5.4.1.1 Datensätze

Im Rahmen von Evaluierungsfall 3 wird auf vier öffentlich verfügbare Datensätze zurückgegriffen:

- Der Datensatz der Case Western Reserve University (CWRU) [268] umfasst Messungen von Kugellagern in verschiedenen Verschleißzuständen. Dazu wurden intakte und auf verschiedene Arten künstlich mittels Funkenerosion verschlissene Lager verschiedener Hersteller an der einen oder anderen Seite eines Elektromotors installiert und die dabei auftretenden Vibrationen gemessen. Auf der jeweils anderen Seite wurde ein intaktes, baugleiches Lager installiert, dessen Vibrationen ebenfalls gemessen wurden. Es wurden 319 Messreihen mit einer durchschnittlichen Länge von ca. 9,6 s und einer Abtastrate von 12 kHz oder 18 kHz aufgezeichnet. Aufgrund des Versuchsdesigns, das innerhalb einer Messreihe von einem statischen Verschleißzustand ausgeht, liegen keine Informationen zu Restlebensdauern oder bereits verstrichener Betriebsdauer vor.

- Der Datensatz der Society for Machinery Failure Prevention Technology (MFPT) [269] umfasst Messungen von Kugellagern in verschiedenen Verschleißzuständen. Dazu wurden die an einem intakten und an auf verschiedene Arten verschlissenen Lagern auftretenden Vibrationen unter verschiedenen radialen Belastungen gemessen. Es wurden 23 Messreihen mit einer durchschnittlichen Länge von ca. 6,4 s und einer Abtastrate von ca. 6 kHz, ca. 24 kHz, ca. 49 kHz oder ca. 98 kHz aufgezeichnet. Aufgrund des Versuchsdesigns, das innerhalb einer Messreihe von einem statischen Verschleißzustand ausgeht, liegen keine Informationen zu Restlebensdauern oder bereits verstrichener Betriebsdauer vor.

- Der Datensatz der Franche-Comté Électronique Mécanique Thermique et Optique – Sciences et Technologies (FEMTO-ST) [270] umfasst Messungen aus verschiedenen Degradationstests von Kugellagern. Dazu wurden im Rahmen von siebzehn Langzeitversuchen jeweils ein Kugellager an einer Welle bis zum Defekt (definiert als Überschreitung der Beschleunigung des Lagers von 20 g) des Lagers unter insgesamt drei verschiedenen radialen Belastungen betrieben. Dabei wurden die an den Lagern auftretenden Vibrationen mittels zweier, um 90° um die Rotationsachse der Lager versetzten Sensoren gemessen. Messdaten wurden nicht kontinuierlich, sondern in 10-Sekunden-Intervallen für jeweils

0,1 s mit einer Abtastrate von 20 kHz aufgezeichnet. Für jedes der siebzehn Lager liegen damit Informationen über die Restlebensdauern bzw. die jeweils bereits verstrichene Betriebsdauer vor.

- Der Datensatz der Intelligent Maintenance Systems (IMS) der University of Cincinnati [271] umfasst Messungen aus verschiedenen Degradationstests von Kugellagern. Dazu wurden im Rahmen von drei Langzeitversuchen jeweils vier Kugellager an derselben Welle bis zum nicht näher spezifizierten Defekt eines der Lager unter derselben radialen Belastung betrieben. Dabei wurden die an den Lagern auftretenden Vibrationen mittels jeweils zweier, um 90° um die Rotationsachse der Lager versetzten Sensoren gemessen. Messdaten wurden nicht kontinuierlich, sondern in 10-Minuten-Intervallen für jeweils 1 s mit einer Abtastrate von 20 kHz aufgezeichnet. Da die Messreihen beim Defekt des ersten Lagers beendet wurden, liegt jeweils nur für eines von vier Lagern eine Information über die Restlebensdauern bzw. die jeweils bereits verstrichene Betriebsdauer vor.

5.4.1.2 Datenvorverarbeitung

Zur gemeinsamen Nutzung der Datensätze im Rahmen von Evaluierungsfall 3 ist eine Vorverarbeitung der Daten notwendig [171]:

Initial repräsentieren die einzelnen Proben der verschiedenen Messreihen unterschiedlich lange Zeitbereiche – teilweise auch innerhalb eines Datensatzes. Dies würde eine Verarbeitung durch dasselbe neuronale Netz, bspw. einen Merkmalsextrakteur, verhindern. Zuerst werden daher die verschiedenen Messreihen in kontinuierliche Proben mit einer Dauer von 0,1 s unterteilt. Dies entspricht der Dauer der kürzesten, kontinuierlichen Messreihensegmente (der des FEMTO-ST-Datensatzes). Etwaig übrigbleibende, kürzere Messreihensegmente werden dabei verworfen.

Anschließend ist für das Labeling eine einheitliche Defekt-Definition notwendig. Die im FEMTO-ST-Datensatz genutzte Definition einer Überschreitung der Beschleunigung von 20 g wird selbst innerhalb des Datensatzes nicht vollumfänglich angewandt (siehe bspw. Bearing1_5 oder Bearing2_4). Auf den IMS-Datensatz ist die Definition aufgrund der stark abweichenden Betriebsbedingungen ebenfalls nicht übertragbar und schließlich ist die Definition anfällig für punktuelle Anomalien, die nicht notwendigerweise im Kugellagerzustand begründet liegen müssen. Alternativ wird daher ein Defekt angenommen, wenn

$$\bar{x}_{Sample} \geq \bar{x}_{Messreihe} + 3\bar{\sigma}_{Messreihe}$$

mit $\bar{x}$ als dem empirischen Mittelwert und $\bar{\sigma}$ als der empirischen Varianz der mittleren absoluten Amplitude einer Probe bzw. aller Proben einer Messreihe. Dies hat den Vorteil, dass ein Defekt relativ zum üblichen Betriebsbereich eines Kugellagers definiert wird und nicht anhand eines Absolutwertes, dass durch Nutzung des Mittelwertes kurzzeitige Signalspitzen nicht unmittelbar zur

Wertung als Defekt führen und dass eine auf der dreifachen Standardabweichung beruhende Defekt-Definition an den vorliegenden Messreihen plausiblere Bewertungen ergab als die zuvor genutzten. Trotzdem können nicht für alle von den Erstellenden als defekt definierte Messreihen ein Defekt entsprechend oben genannter Definition festgestellt werden.

Die Proben des FEMTO-ST- und des IMS-Datensatzes werden anschließend soweit zutreffend mit der bis zum Defekt verbleibenden, auf den Wertebereich von 1 bis 0 normierten Restlebensdauer gelabelt. Eine Restlebensdauer von 1 liegt demnach zu Beginn einer Messreihe vor.

Anschließend werden alle Proben mittels einer Fast Fourier Transformation (FFT) in den Frequenzbereich transformiert. Dabei gehen prinzipbedingt keine Informationen verloren. Durch den reellen Charakter der hier betrachteten Zeitsequenzen ist zudem eine knappe Hälfte des Spektrums redundant, wodurch die Eingangssequenz und damit die Parameterzahl eines sie verarbeitenden neuronalen Netzes, bspw. eines Merkmalsextrakteurs, auf gut die Hälfte erheblich gekürzt werden kann.

Aufgrund der unterschiedlichen Abtastraten bestehen die zwar zeitlich gleich langen Proben aus unterschiedlich vielen Datenpunkten. Dies würde eine Verarbeitung durch dasselbe neuronale Netz, bspw. einen Merkmalsextrakteur, verhindern. Die Auflösung des Frequenzvektors Δf ist dabei konstant, denn es gilt

$$\Delta f = \frac{f_s}{L} = \frac{f_s}{0{,}1\ s * f_s} = \frac{1}{0{,}1\ s} = 10\ Hz$$

mit f_s als der Abtastfrequenz und L als der Anzahl der Elemente des Zeitvektors. Durch die konstante Auflösung kann der Frequenzvektor an seinem Ende durch das Anhängen von Nullen, die Frequenzanteile repräsentieren, die aufgrund der zu geringen Abtastfrequenz nicht mehr erfasst werden konnten, oder durch das Entfernen von Vektorelementen, die Frequenzanteile geringer Relevanz repräsentieren, in seiner Länge verändert werden. Aufgrund der geringen Relevanz der dabei hinzugefügten oder entfernten Frequenzanteile ändert sich das Signal nur geringfügig. Es wird jedoch auf diese Weise eine einheitliche Anzahl Datenpunkte pro Probe (im Frequenzbereich) erzielt. Mittels einer inversen FFT wird auch im Zeitbereich eine einheitliche Anzahl Datenpunkte pro Probe realisiert. Es liegen somit für jede Probe zwei Varianten, eine im Zeit- und eine im Frequenzbereich, vor.

Abschließend wird optional der Wertebereich der Proben auf das Intervall von -1 bis 1 normalisiert, indem für den normalisierten Vektor $\underline{x}_{norm}$ der Zusammenhang

$$\underline{x}_{norm} = \frac{\underline{x}}{\sqrt[2]{\sum_i x_i^2}}$$

mit x_i als dem i-ten Vektorelement gelten soll. Es liegen somit für jede Probe vier Varianten vor, jeweils eine normalisierte und eine nicht-normalisierte im Zeit- und Frequenzbereich.

Tabelle 5.10 gibt einen Überblick über die genutzten Kugellager-Datensätze und schlüsselt dabei insbesondere die einfache (also ohne die oben genannten Varianten zu berücksichtigende) Anzahl Proben sowie die wiederum einfache Anzahl Proben mit bekannter, nicht negativer Restlebensdauer auf. Die Anzahl unterschiedlicher Betriebszustände berücksichtigt auch die Anzahl unterschiedlich ausgerichteter Sensoren als Betriebszustandsdimension.

Tabelle 5.10: Überblick über die Kugellager-Datensätze in Evaluierungsfall 3

Datensatz	Anzahl Proben	Anzahl Proben mit bekannter, nicht negativer Restlebenszeit	Anzahl unterschiedlicher Betriebszustände
CWRU	29.962	0	161
MFTP	1.480	0	19
FEMTO-ST	49.778	44.494	6
IMS	464.800	114.014	2

5.4.2 Realisierung

Im Rahmen von [171] wurden verschiedene Varianten eines von der in Kapitel 4 vorgestellten Architektur abgeleiteten, industriellen Transfer-Lernalgorithmus zur Ausfallvorhersage (vgl. Kapitel 3.2.2.1) unter Verwendung des Konzepts der Domänen-Adaption (vgl. Kapitel 3.1.2.1) entworfen. Dabei kam das Framework PyTorch 1.7.1 [250] unter Python 3.7 [251] zum Einsatz.

Abbildung 5.18 gibt einen schematischen Überblick über die dabei genutzte Realisierung der Lern-Architektur:

Die Eingangsdaten werden mittels des Merkmalsextrakteurs im Eingangsmodul auf einen erheblich kürzeren Merkmalsvektor reduziert. Gelabelte, d.h. mit Restlebenszeit-Informationen versehene Daten werden in der Repräsentationsdatenbank hinterlegt. Die Lebenszeitvorhersage auf Basis dieser gelabelten Daten ist nicht Gegenstand der hier angestellten Betrachtungen. Werden ungelabelte Daten eingelesen, so werden diese zusammen mit ähnlichen gelabelten Daten dem Domänen-Adapter im Ausgangsmodul übergeben. In diesem findet eine symmetrische Domänen-Anpassung statt, die eine Übertragung der Label der gelabelten auf die ungelabelten Daten und damit eine Restlebenszeitvorhersage für beide Datensätze ermöglicht.

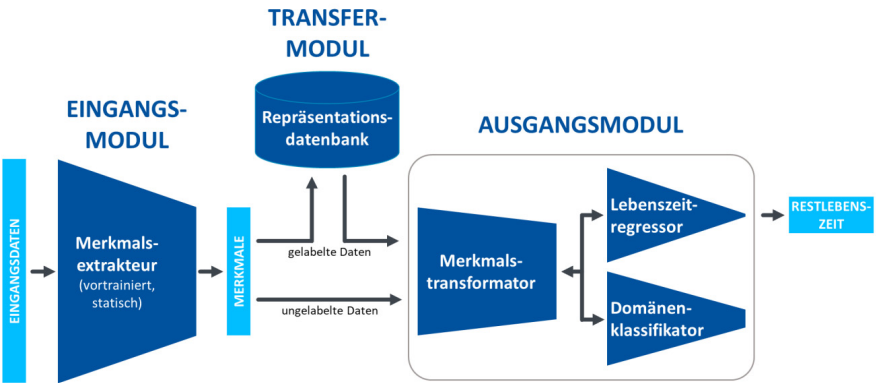

Abbildung 5.18: Schematische Darstellung der Realisierung der Lern-Architektur in Evaluierungsfall 3

5.4.2.1 Eingangsmodul: Merkmalsextrakteur

Entsprechend den Erkenntnissen aus Kapitel 3.1.2 werden für das Eingangsmodul Merkmalsextrakteure basierend auf dem Konzept des Autoencoders (vgl. Kapitel 2.1.3) verwendet.

Anstelle einer systematischen Hyperparameter-Optimierung werden im Folgenden lediglich neun auf Erfahrungswerten basierende Merkmalsextrakteurs-Strukturen miteinander verglichen. Dies liegt einerseits in dem erheblichen Rechenaufwand einer solchen systematischen Optimierung begründet. Andererseits liegt auch in Evaluierungsfall 3 der Fokus auf dem Beleg der grundsätzlichen Anwendbarkeit des Konzepts und nicht auf der Findung einer optimalen Lösung für den spezifischen Anwendungsfall, hier die Ausfallvorhersage von Kugellagern.

Tabelle 5.11 gibt einen Überblick über die Struktur der verschiedenen evaluierten Merkmalsextrakteurs-Varianten. Aufgrund der unterschiedlichen Längen der Proben ist eine Definition der Struktur in Abhängigkeit von dieser Eingangsvektorlänge L_{Probe} notwendig. L_{Probe} kann dabei Werte zwischen 1281 (kürzester untersuchter Frequenzbereichsvektor) und 3000 (längster untersuchter Zeitbereichsvektor) annehmen. Als Aktivierungsfunktion kommen innerhalb der Kodierer ReLU und an ihrem Ausgang SELU zum Einsatz. Die letzte Schicht von FCNN hat immer die Länge der Merkmalsvektoren, Padding und Dilation kommen nicht zum Einsatz. Als Bewertungskriterium kommt der MSE zum Einsatz.

Tabelle 5.11: Überblick über die Merkmalsextrakteurs-Varianten in Evaluierungsfall 3

Nr.	Kodierer-Struktur	Nr.	Kodierer-Struktur
1	3-Schichtiges CNN Schicht 1: $1 \times c_3 \times 10, Stride = 5$ Schicht 2: $c_3 \times c_2 \times 5, Stride = 3$ Schicht 3: $c_2 \times c \times s_2, Stride = 1$	6	2-Schichtiges FCNN Schicht 1: $0{,}55 \cdot L_{Probe}$ Knoten
2	5-Schichtiges CNN Schicht 1: $1 \times c_1 \times 10, Stride = 5$ Schicht 2: $c_1 \times c_2 \times 7, Stride = 4$ Schicht 3: $c_2 \times c_5 \times 5, Stride = 3$ Schicht 4: $c_5 \times c_6 \times 3, Stride = 2$ Schicht 5: $c_6 \times c \times s_5, Stride = 1$	7	4-Schichtiges FCNN Schicht 1: $0{,}8 \cdot L_{Probe}$ Knoten Schicht 2: $0{,}5 \cdot L_{Probe}$ Knoten Schicht 3: $0{,}3 \cdot L_{Probe}$ Knoten
3	2-Schichtiges CNN Schicht 1: $1 \times c_2 \times 30, Stride = 30$ Schicht 2: $c_2 \times c \times s_6, Stride = 1$	8	3-Schichtiges CNN Schicht 1: $1 \times c_1 \times 10, Stride = 5$ Schicht 2: $c_1 \times c_2 \times 5, Stride = 3$ Schicht 3: $c_2 \times c \times 3, Stride = 2$ Flatten + 1-Schichtiges FCNN
4	3-Schichtiges CNN Schicht 1: $1 \times c_7 \times 30, Stride = 1$ Schicht 2: $c_7 \times c_2 \times 31, Stride = 15$ Schicht 3: $c_2 \times c \times s_7, Stride = 1$	9	3-Schichtiges CNN Schicht 1: $1 \times 20 \times 10, Stride = 5$ Schicht 2: $20 \times 40 \times 5, Stride = 3$ Schicht 3: $40 \times 60 \times 3, Stride = 2$ Flatten + 2-Schichtiges FCNN Schicht 1: 500 Knoten
5	2-Schichtiges FCNN Schicht 1: $0{,}17 \cdot L_{Probe}$ Knoten		

Legende: Eingang x Ausgang x Kernel, $c = L_{Merkmal}, c_1 = \frac{c}{3}, c_2 = \frac{c}{2}, c_3 = \frac{c}{4}, c_4 = \frac{c}{3}, c_5 = \frac{2c}{3}, c_6 = \frac{5c}{6}, c_7 = \frac{c}{10}, c_8 = \frac{c}{6}$,
$s_1 = \frac{L_{Probe}-10}{5} + 1,\ s_2 = \frac{s_1-5}{3} + 1, s_3 = \frac{s_1-7}{4} + 1, s_4 = \frac{s_3-5}{3} + 1, s_5 = \frac{s_4-3}{2} + 1, s_6 = \frac{L_{Probe}-30}{30} + 1,\ s_7 = \frac{L_{Probe}-60}{15} + 1$

5.4.2.2 Transfermodul: Repräsentationsdatenbank

Aufgrund der lediglich acht unterschiedlichen Betriebszustände aus zwei Datensätzen ist eine automatische Ähnlichkeitsanalyse in Evaluierungsfall 3 nicht notwendig. Entsprechend kann auch auf das dafür nötige Clustering verzichtet werden. Das Transfermodul besteht somit lediglich aus einer Repräsentationsdatenbank, in der Merkmalsvektoren und Label der gelabelten Quelldatensätze abgelegt werden. Diese Repräsentationsdatenbank ist als .txt-Datei ausgeführt.

5.4.2.3 Ausgangsmodul: Regressor mit Domänen-Adaption

Entsprechend der Bewertung in Kapitel 3.1.3 wird für die Domänen-Adaption auf UDAP nach [175] zurückgegriffen.

Aufgrund der bereits im Vorhinein erfolgten Merkmalsextraktion, die den Speicherbedarf und, im Falle breiterer Verfügbarkeit von Datensätzen, den für das Clustering nötigen Rechenbedarf reduziert, kann die Merkmalsextraktion im Rahmen der Merkmalstransformation auf ein Minimum reduziert werden. In diesem Fall wird die Merkmalsvektorlänge halbiert.

In [175] wird ein Klassifikations-Problem gelöst. Es ist daher nötig, den dort genutzten Label-Prädiktor durch einen Lebenszeit-Regressor zu ersetzen. Als Bewertungskriterium kommt der MSE zum Einsatz.

Tabelle 5.12: Überblick über die Domänen-Adapter-Varianten in Evaluierungsfall 3

Teil-Netz	Struktur Domänen-Adapter Nr. 1	Struktur Domänen-Adapter Nr. 2
Merkmals-Transformator	1-Schichtiges FCNN: $0{,}5 \cdot L_{Merkmal}$ Knoten	3-Schichtiges CNN Schicht 1: $1 \times 64 \times 8, Stride = 1$ Max Pooling: $3, Stride = 2$ Schicht 2: $64 \times 128 \times 6, Stride = 1$ Max Pooling: $3, Stride = 2$ Schicht 3: $128 \times 256 \times 3, Stride = 1$ Flatten
Lebenszeit-Regressor	3-Schichtiges FCNN Schicht 1: $0{,}5 \cdot L_{Merkmal}$ Knoten Schicht 2: $0{,}5 \cdot L_{Merkmal}$ Knoten Dropout (Wahrscheinlichkeit 0,5) Schicht 3: 1 Knoten	2-Schichtiges FCNN Schicht 1: $\frac{L_{Merkmal}-10}{168} - \frac{13}{84}$ Knoten Dropout (Wahrscheinlichkeit 0,5) Schicht 2: 1 Knoten
Domänen-Klassifikator	3-Schichtiges FCNN Schicht 1: $0{,}5 \cdot L_{Merkmal}$ Knoten Schicht 2: $0{,}5 \cdot L_{Merkmal}$ Knoten Dropout (Wahrscheinlichkeit 0,5) Schicht 3: 1 Knoten	2-Schichtiges FCNN Schicht 1: $\frac{L_{Merkmal}-10}{168} - \frac{13}{84}$ Knoten Dropout (Wahrscheinlichkeit 0,5) Schicht 2: 1 Knoten

Legende: Eingang x Ausgang x Kernel

In Bezug auf den Domänen-Klassifikator werden vier Parameter-Varianten untersucht: Einerseits kommt als Gradienten-Umkehrschicht-Faktor λ entweder ein konstanter Wert von $\lambda = 1$ (zur dauerhaften gleich-Gewichtung von Regressions- und Klassifikationsverlust) oder nach [175]

$$\lambda = \frac{2}{1 + e^{-10p}} - 1$$

mit p als Trainingsfortschritt, hier ein Wert zwischen 0 und 1, um den anfangs verrauschten Klassifikator-Gradienten zu unterdrücken. Andererseits kommen als Bewertungskriterium entweder die binäre Kreuzentropie (engl.: Binary Cross Entropy, kurz: BCE) oder der MSE zum Einsatz.

Tabelle 5.12 gibt einen Überblick über die Struktur der beiden evaluierten Domänen-Adapter-Varianten. Aufgrund der unterschiedlichen Längen der zuvor extrahierten Merkmalsvektoren ist eine Definition der Struktur in Abhängigkeit von dieser Merkmalsvektorlänge $L_{Merkmal}$ notwendig. $L_{Merkmal}$ kann dabei entweder 100 oder 300 sein. Als Aktivierungsfunktion kommen ReLU zum Einsatz.

5.4.2.4 Vergleichs-Algorithmus: Regressor ohne Domänen-Adaption

Zur Evaluation der Domänen-Adaption werden zwei Vergleichsalgorithmen entsprechend den beiden Domänen-Adaptern aus Kapitel 5.4.2.3 erstellt. Diese verbinden jeweils die Schichten eines Merkmals-Transformators mit denen eines Lebenszeit-Regressors und sind damit hinsichtlich der Restlebenszeitvorhersage identisch zu den jeweiligen Domänen-Adaptern strukturiert.

Aufgrund der unterschiedlichen Längen der genutzten Eingangsvektoren (teilweise Merkmalsvektoren, teilweise Zeit- oder Frequenzvektoren) sind auch die Strukturen der Regressoren ohne Domänen-Adaption in Abhängigkeit von dieser Eingangsvektorlänge $L_{Eingang}$ definiert. $L_{Eingang}$ kann hier Werte zwischen 100 (kürzester untersuchter Merkmalsvektor) und 3000 (längster untersuchter Zeitbereichsvektor) annehmen. Als Aktivierungsfunktion kommen ReLU zum Einsatz, als Bewertungskriterium wird der MSE genutzt.

5.4.3 Evaluierung

Evaluierungsfall 3 soll gegenüber der Referenz-Implementierung von Evaluierungsfall 1 und 2 die methodische Flexibilität der in Kapitel 4 vorgestellten Architektur für industrielles Transfer-Lernen demonstrieren. Der hier realisierte, stark integrierte Domänen-Adaptions-basierte Gesamtalgorithmus wird dazu mit einem konventionellen Vergleichs-Algorithmus verglichen.

Darüber hinaus zeichnet sich Evaluierungsfall 3 durch die stark periodisch geprägten Eingangsdaten aus. Derartige (bspw. Vibrations-)Signale sind in der Automatisierungstechnik stark verbreitet und unterscheiden sich hinsichtlich ihrer charakteristischen Merkmale, sodass im Folgenden auch die Merkmalsextraktion erneut getestet wird.

5.4.3.1 Versuchsdurchführung

Begonnen wird in Kapitel 5.4.3.2 mit einer Evaluierung des Eingangsmoduls. Neben der Untersuchung der grundsätzlichen Funktionalität liegt der Fokus auf der Auswahl einer geeigneten Eingangsmodul-Parametrierung sowie Merkmalsextrakteurs-Struktur für die nachfolgenden Experimente. Die entsprechenden Freiheitsgrade können Tabelle 5.13 entnommen werden.

Tabelle 5.13: Überblick über die untersuchten Freiheitsgrade des Eingangsmoduls in Evaluierungsfall 3

Parameter	Wertebereich
Freiheitsgrade der Eingangsdaten	
Zeit- oder Frequenzbereich Eingangsdaten	{Zeitbereich; Frequenzbereich}
Normalisierung Eingangsdaten	{ja; nein}
Freiheitsgrade der Merkmalsextraktion	
Merkmalsextrakteurs-Struktur	{ME1; ME2; ME3; ME4; ME5; ME6; ME7; ME8; ME9}
Länge Merkmalsvektor	{100; 300}

Die neun Merkmalsextrakteurs-Varianten werden dazu mit dem CWRU-, dem MFPT- und dem FEMTO-ST-Datensatz trainiert. Die fehlenden Label von CWRU- und MFPT-Datensatz spielen hier keine Rolle und die große Bandbreite unterschiedlicher (Verschleiß-) und Betriebszustände sorgt für eine umso robustere Merkmalsextraktion. Aus diesem Grund wird auch der FEMTO-ST-Datensatz dem IMS-Datensatz zum Training des Merkmalsextrakteurs vorgezogen, während Letzterer nicht genutzt wird, um als bisher unbekannter Validierungsdatensatz genutzt werden zu können.

Aufgrund der Nutzung des FEMTO-ST-Datensatzes zum Training des Merkmalsextrakteurs und seiner damit einhergehenden Bekanntheit zumindest gegenüber einem Teil des Gesamtalgorithmus muss dieser zwingend als Quelldomäne eingesetzt werden (vgl. Kapitel 2.2). Die Zieldomäne wird damit durch den IMS-Datensatz repräsentiert. Aufgrund der Anordnung der Beschleunigungssensoren zur radialen Last wird entschieden, dass idealerweise IMS-Sensor 1 mit FEMTO-ST-Sensor 2 sowie IMS-Sensor 2 mit FEMTO-ST-Sensor 1 gepaart werden sollten.

Anschließend wird in Kapitel 5.4.3.3 der komplette Lernalgorithmus basierend auf Domänen-Adaption evaluiert. Hierzu wird dessen Vorhersagequalität auf den drei Transfer-Szenarien von Betriebsbedingung 1, Sensor 2, von Betriebsbedingung 2, Sensor 2 und von Betriebsbedingung 3, Sensor 2 des FEMTO-ST-Datensatzes zu Betriebsbedingung 1, Sensor 1 des IMS-Datensatzes der des Vergleichs-Algorithmus ohne Domänen-Adaption aus Kapitel 0 gegenübergestellt.

Alle Trainings wurden über 100 Epochen, mit einer Lernrate von 0,001, einer Batch-Größe von 512, und einer zufälligen Aufteilung in Trainings- und Validierungsdaten mit einem Verhältnis von 90:10 durchgeführt. Alle Rechnungen wurden auf einem Computer mit einer AMD Ryzen Threadripper 2920X CPU und einer NVIDIA GeForce RTX 2080 8GB GPU unter Ubuntu 20.04 ausgeführt.

5.4.3.2 Merkmalsextraktion: Ergebnisse und Diskussion

Für eine Parametrierung des Eingangsmoduls zur Merkmalsextraktion der vorliegenden Vibrationsdaten werden verschiedene Versionen davon trainiert und getestet. Der Algorithmus testet die verschiedenen Merkmalsextrakteurs-Strukturen an normalisierten und nicht normalisierten Eingangsdatenvektoren in Zeit- und Frequenzbereich und komprimiert diese damit in einen Vektor der Länge 100 oder 300.

Tabelle 5.14 zeigt den Einfluss der Normalisierung bzw. der Nutzung von Eingangsdaten im Zeit- oder Frequenzbereich auf den mittleren Rekonstruktionsverlust. Hierbei wird ersichtlich, dass aufgrund der unterschiedlichen Wertebereiche der Eingangsdaten in den verschiedenen Fällen auch die MSE-Rekonstruktionsverluste einen völlig anderen Wertebereich aufweisen. Es ist daher die Einführung eines Hilfskonstrukts, des Bewertungsfaktors, nötig, um einen Vergleich der verschiedenen Fälle zu ermöglichen. Eine detaillierte Erklärung seiner Berechnung ist Anhang B zu entnehmen.

Tabelle 5.14: Einfluss der Normalisierung der Eingangsdaten im Zeit- oder Frequenzbereich auf Rekonstruktionsverlust und Bewertungsfaktor der Merkmalsextraktion

Mittlerer Rekonstruktionsverlust	Keine Normalisierung	Normalisierung	**Mittlerer Bewertungsfaktor**
Zeitbereich	0,877221	0,000251	86,18
Frequenzbereich	453,737645	0,000098	15,86
Mittlerer Bewertungsfaktor	57,92	44,12	-

Es wird ersichtlich, dass die Merkmalsextraktion auf Eingangsdaten im Frequenzbereich besser funktioniert als auf Eingangsdaten im Zeitbereich. Der Bewertungsfaktor unterscheidet sich um ein Vielfaches. Auch die Betrachtung der Einzelergebnisse offenbart, dass nicht einmal die beste Merkmalsextraktion auf Eingangsdaten im Zeitbereich besser als die schlechteste Merkmalsext-

raktion auf Eingangsdaten im Frequenzbereich ist. In den folgenden Schritten der Evaluation werden daher nur Merkmalsextraktionen auf Basis von Eingangsdaten im Frequenzbereich verwendet.

Weiterhin wird ersichtlich, dass die Merkmalsextraktion auf normalisierten Eingangsdaten besser funktioniert als auf nicht normalisierten Eingangsdaten. Der mittlere Bewertungsfaktor der Merkmalsextraktion auf nicht normalisierten Eingangsdaten ist ca. 31% größer als der der Merkmalsextraktion auf normalisierten Eingangsdaten. Auch die Betrachtung der Einzelergebnisse ergibt, dass nur einzelne Merkmalsextraktionen auf nicht-normalisierten Eingangsdaten besser als die schlechtesten Merkmalsextraktionen auf normalisierten Eingangsdaten sind. Dies betrifft sowohl Merkmalsextraktionen auf Eingangsdaten im Frequenz- als auch im Zeitbereich. In den folgenden Schritten der Evaluation werden daher nur Merkmalsextraktionen auf Basis normalisierter Eingangsdaten verwendet.

Tabelle 5.15 zeigt den Einfluss der Merkmalsvektorlänge auf den mittleren Bewertungsfaktor. Hier ist das Ergebnis weniger eindeutig, der Unterschied zugunsten einer Merkmalsvektorlänge von 300 beträgt lediglich ca. 5%. Auch die Betrachtung der Einzelergebnisse ergibt keinen eindeutigen Trend, da sich auch unter den besten Merkmalsextraktionen solche mit einer Merkmalsvektorlänge von 100 finden. Darüber hinaus versprechen diese kürzeren Merkmalsvektorlängen in der weiteren Betrachtung Rechenzeit- und Speicherplatzersparnisse. In den folgenden Schritten der Evaluation werden daher Merkmalsextraktionen auf Basis von sowohl der Merkmalsvektorlänge 100 als auch 300 verwendet.

Tabelle 5.15: Einfluss der Merkmalsvektorlänge auf Validierungsverlust und Bewertungsfaktor der Merkmalsextraktion

Merkmalsvektorlänge	Mittlerer Bewertungsfaktor
100	52,28
300	49,76

Insgesamt liefern Merkmalsextrakteursvarianten 4 und 8 die besten Ergebnisse. Für die folgenden Experimente werden daher diese beiden Merkmalsextrakteure in jeweils einer Version für Merkmalsvektorlänge 100 und 300 genutzt.

5.4.3.3 Gesamtsystem: Ergebnisse und Diskussion

Für eine Bewertung der Vorhersagequalität des Gesamtalgorithmus wird dieser entsprechend der in Kapitel 5.4.3.1 definierten Fragestellung getestet. Dazu werden die beiden Domänen-Adapter-Varianten mit den Vergleichs-Algorithmen unter denselben Bedingungen bzw. mit denselben Parametern verglichen. Es werden Versuchsläufe mit den beiden Merkmalsextrakteursvarianten, den

beiden Merkmalsvektorlängen und konstantem Gradienten-Umkehrschicht-Faktor auf normalisierten Merkmalsvektoren durchgeführt und die dabei erzielten Resultate gemittelt. Domänen-Adapter- bzw. Vergleichsalgorithmus-Variante 1 erreichte bei Nutzung des BCE als Bewertungskriterium eine größere Verbesserung der Vorhersagequalität, während Variante 2 mit dem MSE bessere Ergebnisse erzielte. Es wird im Folgenden das jeweils optimale Bewertungskriterium genutzt.

Abbildung 5.19 zeigt den absoluten Fehler der beiden Algorithmus-Varianten mit und ohne Domänen-Adaption in Abhängigkeit vom Zielproblem.

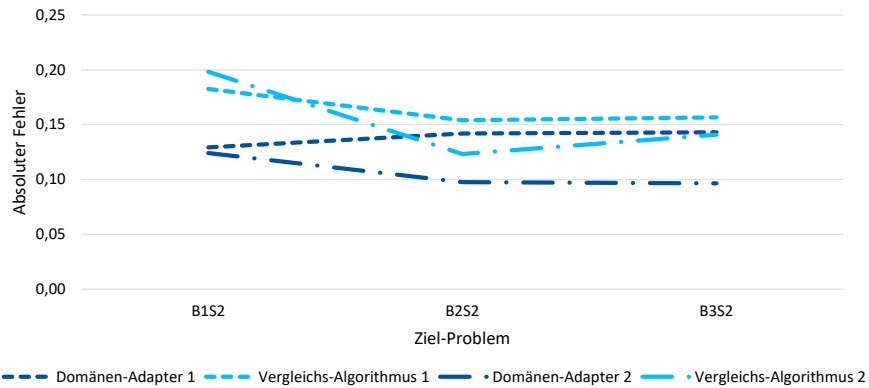

Abbildung 5.19: Absoluter Fehler der beiden Algorithmus-Varianten mit und ohne Domänen-Adaption in Abhängigkeit vom Zielproblem

Es zeigt sich, dass die Domänen-Adaption die Vorhersagequalität für alle drei Zielprobleme verbessert. Durchweg die besten Ergebnisse erzielte dabei Domänen-Adapter 2, während innerhalb der Vergleichs-Algorithmen abhängig vom Zielproblem sowohl Variante 1 als auch Variante 2 die höhere Vorhersagequalität beweisen. Für den Transfer hin zu Betriebspunkt 2, Sensor 2 erzielte Vergleichs-Algorithmus 2 sogar eine höhere Vorhersagequalität als Domänen-Adapter 1.

Abbildung 5.20 zeigt die Veränderungen des Fehlers der beiden Algorithmus-Varianten durch Domänen-Adaption in Abhängigkeit vom Zielproblem und setzt die Verläufe aus Abbildung 5.19 somit ins Verhältnis.

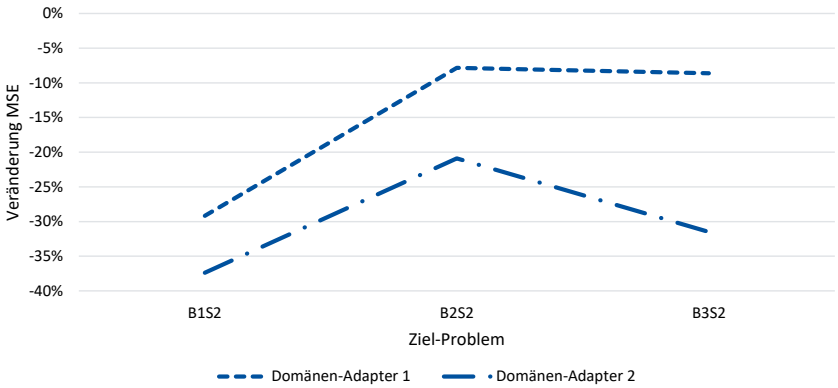

Abbildung 5.20: Veränderung des Fehlers der beiden Algorithmus-Varianten durch Domänen-Adaption in Abhängigkeit vom Zielproblem

Es zeigt sich, dass die prozentuale Verbesserung, die durch den Einsatz von Domänen-Adaption erzielt werden kann, stark vom Zielproblem abhängt. Domänen-Adapter 2 erzielt dabei auch relativ die besseren Ergebnisse – trotzdem schwankt die Verbesserung zwischen ca. 21 % und ca. 37 %.

5.4.4 Bewertung

Im Rahmen der Versuche zu Evaluierungsfall 3 ergeben sich folgende Ergebnisse:

Die Merkmalsextraktion von stark periodisch geprägten Eingangsdaten liefert reproduzierbar auch bei starker Kompression der Eingangsdatenvektoren Merkmalsvektoren, die zu einem geringen Rekonstruktionsverlust führen. Dabei ist eine Transformation der Daten in den Frequenzbereich in Verbindung mit einer Normalisierung empfehlenswert.

Das Gesamtsystem erzielt durch den Einsatz von Domänen-Adaption auch auf dem Anwendungsfall Ausfallvorhersage für Kugellager eine erhebliche Verbesserung der Vorhersagequalität. Es lassen sich somit bei vorhandenen, gelabelten Quelldaten auch mit ungelabelten Zieldaten halbüberwacht entsprechende Lern-Probleme lösen.

Die besten Resultate erbringt ein Domänen-Adapter auf CNN-Basis mit konstantem Gradienten-Umkehrschicht-Faktor und normalisiertem Merkmalsvektor. In den Transfer-Szenarien kann so eine Verbesserung gegenüber dem lediglich um die Domänen-Adaption verkürzten Vergleichsalgorithmus von bis zu 37 % erzielt werden.

Die im Rahmen von Evaluierungsfall 3 durchgeführten Untersuchungen ergeben somit, dass der dabei erstellte Prototyp als Lösungsartefakt drei Anforderungen zur Beantwortung der Forschungsfrage erfüllt: Das resultierende Gesamtsystem benötigt keine gelabelten Zielproben, um erfolgreich nachtrainiert zu werden (A2), es nutzt dazu in der Repräsentationsdatenbank hinterlegte Quellmerkmalsvektoren weiter (A3) und behält weite Teile der Codebasis über die verschiedenen Teilprobleme bei (A4). Für eine tiefergehende Analyse hinsichtlich der Beantwortung der Forschungsfrage sei auf das nachfolgende Kapitel 5.5 verwiesen.

5.5 Erfüllung der Zielsetzung

Die zuvor beschriebenen, im Rahmen der drei Evaluierungsfälle entwickelten Prototypen stellen Instanzen der Architektur für industrielles Transfer-Lernen als allgemeine Lösungsartefakte im Sinne des DSR dar. In der folgenden zweiten Lösungsansatzvalidierung wird an ihnen die Eignung der Architektur zur Beantwortung der in Kapitel 3.4 formulierten Forschungsfrage mittels eines Abgleichs mit den in Kapitel 1.4 formulierten Anforderungen geprüft.

Robustheit, um die Notwendigkeit des Nachtrainierens zu reduzieren (A1)

In Evaluierungsfall 1 ermöglicht die Nutzung industriellen Transfer-Lernens eine robuste, unüberwachte Anomaliedetektion unter Vermeidung der sonst auftretenden Probleme bei einem zu hohen Anteil anormaler Trainingsdaten.

In allen Evaluierungsfällen generalisiert die Nutzung der Architektur für industrielles Transfer-Lernen die Lernergebnisse des jeweiligen Algorithmus durch eine breitere Trainingsbasis und erhöht somit grundsätzlich dessen Robustheit. Inwiefern eine derartige Erhöhung der Robustheit eine Übertragung zwischen (Teil-)Problemen ohne Nachtraining erlaubt, ist jedoch vom jeweiligen Anwendungsfall abhängig.

Adaptierbarkeit, um den Aufwand des Nachtrainierens zu reduzieren (A2)

In den Evaluierungsfällen 1 und 2 kann durch die Nutzung industriellen Transfer-Lernens die Anforderungen an den Umfang der Ziel-Trainingsdatensätze reduziert werden. Dies reduziert den Aufwand der Daten-Erhebung bei einer Veränderung des betrachteten Problems.

Innerhalb der Evaluierungsfälle 1 bis 3 sowie von Evaluierungsfall 1 hin zu Evaluierungsfall 2 kann durch die Nutzung der Architektur für industrielles Transfer-Lernen einfach auf veränderte Problemstellungen reagiert werden. Dies reduziert den Aufwand der Algorithmus-Anpassung bei einer Veränderung des betrachteten Problems.

In allen Evaluierungsfällen reduziert die Merkmalsextraktion die Anzahl nötiger Netzwerkknoten in dem für die Lösung des konkreten Problems zuständigen Ausgangsmodul. Dies reduziert den Trainings-Aufwand bei einer Veränderung des betrachteten Problems.

Wiederverwendbarkeit von Daten zur Anpassung an neue Anwendungskontexte (A3)

In Evaluierungsfall 2 ermöglicht die Nutzung industriellen Transfer-Lernens die gleichzeitige Integration sowohl mehrerer Zeitreihen als auch von Metadaten in das Eingangsmodul. Der Prozess unterscheidet sich dabei nicht wesentlich von der Behandlung von bspw. nur Zeitreihendaten in Evaluierungsfall 1 oder 3.

In Evaluierungsfall 3 ermöglicht die Nutzung industriellen Transfer-Lernens halbüberwachtes Lernen auf Basis gelabelter Quelldaten in Verbindung mit ungelabelten Zieldaten.

In allen drei Evaluierungsfällen ermöglicht die Nutzung industriellen Transfer-Lernens mit lediglich geringfügigen Abwandlungen die Nutzung derselben Eingangsmodule trotz unterschiedlich strukturierter Eingangsdaten. Darüber hinaus ermöglicht die Nutzung industriellen Transfer-Lernens die Wiederverwendbarkeit von Daten bzw. Modellen zwischen verschiedenen (Teil-)Problemen eines Anwendungsfalls. Auf diesem Weg kann ein Transfer von Wissen und Informationen zwischen (Teil-)Problemen realisiert werden.

Wiederverwendbarkeit von Code zur Anpassung an neue Anwendungskontexte (A4)

Von Evaluierungsfall 1 zu Evaluierungsfall 2 kann eine hohe Wiederverwendbarkeit des Codes durch die Nutzung der Architektur für industrielles Transfer-Lernen erreicht werden. Konkret betrifft das in diesen Fällen neben der allgemeinen Architektur das Transfermodul sowie den grundsätzlichen Aufbau des Eingangsmoduls.

Fazit

Zusammenfassend lässt sich feststellen, dass die Lösungsansatzvalidierung für das Lösungsartefakt „Architektur für industrielles Transfer-Lernen“ auch anhand seiner Implementierungsinstanzen im Kontext konkreter Anwendungsfälle einen hohen Erfüllungsgrad der Anforderungen ergibt. Die Architektur eignet sich somit zur Beantwortung der Forschungsfrage in Form allgemeinen, übertragbaren und fundierten Gestaltungswissens:

Konkret konnte im Rahmen dieser Arbeit eine Architektur für übertragbare Lernalgorithmen vorgestellt werden, die auch auf kleinen Datensätzen und trotz der hohen Dynamik von Automatisierungssystemen robust Industrie-typische Probleme lösen können. Die Evaluation anhand von drei Evaluierungsfällen zeigt einen hohen Grad der Übertragbarkeit von spezifischem Wissen innerhalb eines Anwendungsfalls und von Architekturkomponenten, bspw. Eingangs-(Teil-)Modulen oder Transferlogik, zwischen verschiedenen Anwendungsfällen.

6 Schlussbetrachtung

In diesem Kapitel wird neben einer Zusammenfassung der Ergebnisse dieser Arbeit ein Ausblick auf potenziell weiterführende Forschungsaktivitäten gegeben.

6.1 Zusammenfassung der Ergebnisse

Trotz des in vielfältigen Kontexten demonstrierten Potentials maschinellen Lernens mittels neuronaler Netze für Anwendungsfälle in der Automatisierungstechnik können sich diese bisher nicht flächendeckend durchsetzen. Die entgegenstehenden Herausforderungen sind ihre mangelnde Übertragbarkeit zwischen Anlagen, über Prozessgrenzen hinweg sowie zwischen Datensilos, die zusammen zu wenig generalisierten Lernergebnissen und hohen Anpassungsaufwänden schon bei kleinen Veränderungen der betrachteten (Teil-)Probleme führen.

Eine Untersuchung des Stands von Wissenschaft und Technik ergab ein steigendes Interesse am Thema Transfer-Lernen in den betrachteten industriellen Anwendungsfällen Anomaliedetektion und Ausfallvorhersage. Sie offenbarte jedoch auch, dass es bisher keine Beschreibungen von Ansätzen gibt, die alle der zuvor beschriebenen Herausforderungen adressieren. Es ließen sich jedoch Teilaspekte identifizieren, die einen derartigen Ansatz ermöglichen oder verbessern würden.

Basierend auf diesen Teilaspekten wurde in der vorliegenden Arbeit eine modulare Architektur für industrielles Transfer-Lernen entwickelt, die die zuvor beschriebenen Herausforderungen adressiert:

In Anlehnung an entsprechende, bspw. in der Bilderkennung oder Sprachverarbeitung erfolgreich etablierte Ansätze wird eine Trennung von Merkmalsextraktion und eigentlicher Problemlösung vorgenommen. Ein auf dem Prinzip der Mehrköpfigkeit beruhendes, statisches Eingangsmodul extrahiert Merkmale aus den Eingangsdaten. Ein Transfermodul ermöglicht darauf aufbauend mittels Clustering die Ähnlichkeits-basierte Bereitstellung von merkmalsextrahierten Daten und fertig trainierten Modellen für die Erleichterung des Trainings neuer Probleme im Ausgangsmodul. Dieses Ausgangsmodul stellt den eigentlichen Problemlöser dar und wird anhand neuer Eingangsdaten und vom Transfermodul bereitgestellter Daten oder Parametersätze bei Bedarf nachtrainiert. Es verarbeitet dabei in der Regel keine Rohdaten, sondern Merkmalsvektoren.

Anhand von drei unterschiedlichen, Industrie-typischen Anwendungsfällen wurde die Architektur anschließend evaluiert:

Im ersten Evaluierungsfall bestand die Aufgabe darin, eine Anomaliedetektion für die Pumpen einer Hydraulikpresse auf Basis von Druckverläufen zu realisieren, obwohl die gefertigten Pro-

dukte häufig wechselten und mit jedem neuen Produkt potenziell andere charakteristische Druckverläufe einhergingen. Mittels Modell- oder Datentransfer konnte jedoch die Menge der benötigten Trainingsdaten sowie gleichzeitig die Abhängigkeit von der richtigen Trainingsdatenmenge erheblich reduziert werden. Damit einher ging eine deutliche Reduktion der Trainingszeit.

Im zweiten Evaluierungsfall bestand die Aufgabe darin, eine Ausfallvorhersage für elektromechanische Relais auf Basis von verschiedenen Messgrößen zu realisieren, obwohl die Relais unter unterschiedlichen Einsatzbedingungen verwendet wurden und mit jeder anderen Einsatzbedingung potenziell andere Ausfallcharakteristiken einhergingen. Mittels einer Mischung von Modell- und Datentransfer konnte jedoch die Menge der benötigten Trainingsdaten erheblich reduziert werden. Dabei konnten von Evaluierungsfall 1 das Eingangsmodul in angepasster Form sowie das Transfermodul unverändert wiederverwendet werden.

Im dritten Evaluierungsfall bestand die Aufgabe darin, eine Ausfallvorhersage für Kugellager auf Basis von Vibrationsdaten zu realisieren, obwohl die Kugellager unter unterschiedlichen Einsatzbedingungen verwendet wurden und mit jeder anderen Einsatzbedingung potenziell andere Ausfallcharakteristiken einhergingen. Mittels einer Domänen-Adaption konnten jedoch erfolgreich Zusammenhänge aus bekannten, gelabelten Einsatzbedingungen auf neue, ungelabelte Einsatzbedingungen übertragen werden.

Mittels der Methodik des Design Science Research konnte für die untersuchte Problemstellung ein Lösungsartefakt entworfen und validiert werden. Es eignet sich somit zur Beantwortung der Forschungsfrage in Form allgemeinen, übertragbaren und fundierten Gestaltungswissens. So ließ sich nachweisen, dass durch industrielles Transfer-Lernen in allen drei Evaluierungsfällen die darin manifestierten Herausforderungen überwunden werden konnten. Konkret ließ sich je nach Anwendungsfall die Vektorlänge um über 98 % reduzieren, was erhebliche Einsparungen bezüglich des Speicher- und Rechenleistungsbedarfs bewirkte. Die Problemlösefähigkeit des Gesamtalgorithmus nahm je nach Anwendungsfall um bis zu 66 % zu bzw. der Fehler um bis zu 71 % ab, während die dazu nötige Rechenzeit um ca. 50 % reduziert werden konnte. Die vorgestellte Architektur ermöglichte dabei in den untersuchten Szenarien eine aufwandsarme Anpassung an Veränderungen unter Wiederverwendung von Architekturelementen sowohl innerhalb eines Anwendungsfalls als auch über verschiedene Anwendungsfälle hinweg.

Ein Anspruch auf Optimalität der Partikularlösung eines Evaluierungsfalls besteht nicht, da es innerhalb dieser Arbeit lediglich um den Beleg der grundsätzlich lösungsfördernden Machbarkeit des vorgestellten Ansatzes ging. Im folgenden Kapitel werden daher sowohl weiterführende Forschungsaktivitäten hinsichtlich einer Untersuchung der Optimalität der vorgestellten Architektur bzw. darauf basierender Partikularlösungen als auch hinsichtlich einer Erweiterung seines Funktionsumfangs vorgestellt.

6.2 Ausblick auf weiterführende Forschungsaktivitäten

Auf Basis der vorgestellten Architektur für industrielles Transfer-Lernen ist nun die Transfer-Szenario-übergreifende Erforschung (Teil-)Problem-übergreifender Deep-Learning-Methoden möglich. Ein Fokus sollte dabei insbesondere auf der Findung möglichst allgemeingültiger Ansätze liegen, um die beschriebenen Anwendungshürden tatsächlich zu überwinden (siehe Kapitel 1.2).

Im Rahmen dieser Arbeit konnten darüber hinaus einige Aspekte identifiziert werden, die als Ausgangspunkte für weitere Forschungsvorhaben zur unmittelbaren Erweiterung oder Verbesserung der vorgestellten Architektur für industrielles Transfer-Lernen dienen können:

Automatische Ermittlung des Transfer-Bedarfs: Insbesondere bei lang andauernden Prozessen oder bei Prozessfolgen ohne klare Abgrenzung zwischen einzelnen Prozessen ist eine manuelle Auslösung des Wissens-Transfers (siehe Kapitel 5.2.2.2) nicht sinnvoll, da sie Expertenwissen sowohl über den Prozess und die dabei genutzten Anlagen als auch über den Lernalgorithmus voraussetzt. Sinnvoll wäre daher eine automatische Ermittlung des Transfer-Bedarfs [5]. Dies wäre beispielsweise mittels regelmäßiger, versuchsweiser und vorerst vorläufiger Auslösung eines Wissens-Transfers oder über die Beobachtung noch zu identifizierender Transfer-Indikatoren denkbar.

Erweiterung der Domänen-Adaption: In Evaluierungsfall 3 ist aufgrund der verwendeten Datensätze keine Ähnlichkeits-basierte Auswahl der Transfer-Datensätze für die Nutzung in der Domänen-Adaption sinnvoll (siehe Kapitel 5.4.2.2). Es ist jedoch zu erwarten, dass die Verwendung einer derartig erweiterten Transferlogik analog zu den Erfolgen in Evaluierungsfall 1 und 2 eine erhöhte Leistungsfähigkeit der Domänen-Adaption bewirken würde [48, 272]. Es wäre daher denkbar, auf einem anderen Datensatz auch für die Domänen-Adaption eine Clustering-basierte Ähnlichkeitsbetrachtung zu implementieren und deren Auswirkungen zu analysieren. Auch eine Domänen-Adaption zwischen mehr als zwei Domänen könnten in diesem Kontext eine sinnvolle Ergänzung sein, um vielfältigere Transfer-Datensätze auch aus mehreren Domänen zu ermöglichen.

Berücksichtigung kontinuierlicher Zeitreihendaten: In den Evaluierungsfällen dieser Arbeit werden Zeitreihendaten nur segmentiert betrachtet. In vielen Anwendungsfällen entsteht jedoch ein informativer Mehrwert, wenn man kontinuierlich erhobene Zeitreihendaten auch entsprechend kontinuierlich weiterverarbeitet und ggf. um Zustandsunterscheidungen erweitert [77]. Dies wäre bspw. mittels einer Einbeziehung rekurrenter neuronaler Netzwerke denkbar [217].

Die folgenden Aspekte gehen über die unmittelbare Auseinandersetzung mit einzelnen Komponenten der vorgestellten Architektur für industrielles Transfer-Lernen hinaus:

Universeller(er) Merkmalsextrakteur für industrielle Zeitreihendaten: Unter Inkaufnahme geringfügiger Leistungseinbußen ist ein universeller Merkmalsextrakteur für industrielle Zeitreihen denkbar. Dieser hätte den Vorteil, dass für das Eingangsmodul des Lernalgorithmus keine Hyperparameteroptimierung bzw. manuelle Anpassung der Netzwerktopologie mehr erforderlich wäre. Als erster Schritt in diese Richtung wäre auch ein lediglich zwischen Sensorarten (bspw. Vibrationssensor, Temperatursensor) unterscheidender Merkmalsextrakteur denkbar.

Lebenszyklus-übergreifendes Transfer-Lernen: Transfer-Lernen ist nicht nur während der Betriebsphase eines Prozesses oder einer Anlage sinnvoll, sondern auch darüber hinaus. Denkbar wäre ein Lebenszyklus-übergreifender Wissens-Transfer bspw. in Kombination mit dem Digitalen Zwilling – einerseits als Datenlieferant auch aus der Simulation heraus und andererseits als Testumgebung vor der automatischen Anwendung erlernter, aber potenziell unerwünschter Verhaltensweisen auf der realen Anlage. Erste Vorüberlegungen dazu wurden bereits in [57, 273] angestellt.

Schaffung eines Frameworks für industrielles Transfer-Lernen: Ausgehend von der vorgestellten Architektur für industrielles Transfer-Lernen wäre ein entsprechendes Framework denkbar. Dazu wäre mindestens ein Assistenzsystem zur Unterstützung beim Aufbau bzw. der Parametrierung der einzelnen Module notwendig – auch eine weitgehende Automatisierung könnte in Folge einer Erprobung auf weiteren Anwendungsfällen möglich sein. Dies würde dem Zweck dienen, tatsächlich Expertenwissen einzusparen und nicht nur die Notwendigkeit von Fach-Expertenwissen durch einen erhöhten Bedarf an Deep-Learning-Expertenwissen zu ersetzen.

7 Literaturverzeichnis

[1] S. Khan und T. Yairi, “A review on the application of deep learning in system health management,” *Mechanical Systems and Signal Processing*, Bd. 107, S. 241–265, 2018, DOI: 10.1016/j.ymssp.2017.11.024.

[2] G. Reinhart, “Assembly Automation,” in *CIRP Encyclopedia of Production Engineering*, S. Chatti, L. Laperrière, G. Reinhart, und T. Tolio, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2019, S. 78–80.

[3] J. Bell, *Machine learning: Hands-on for developers and technical professionals*. Indianapolis, USA: John Wiley & Sons, 2015.

[4] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham, Schweiz: Springer International Publishing, 2018.

[5] B. Maschler, H. Tercan, C. Bitter, H. Vietz, T. Meisen, und M. Weyrich, “Industrielles Transfer-Lernen: Von der Wissenschaft in die Praxis,” *atp Magazin*, Bd. 63, Nr. 9, S. 86–93, 2022.

[6] R. Kohavi und F. Provost, “Glossary of Terms,” *Machine Learning*, Bd. 30, 2-3, S. 271–274, 1998.

[7] Plattform Industrie 4.0, “Glossar,” 2021.

[8] K. Alexopoulos und G. Chryssolouris, “Process,” in *CIRP Encyclopedia of Production Engineering*, S. Chatti, L. Laperrière, G. Reinhart, und T. Tolio, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2019, S. 1349–1352.

[9] ISO/IEC/IEEE 24765:2017-09, *Systems and software engineering - Vocabulary*.

[10] U. Lämmel und J. Cleve, *Künstliche Intelligenz: Lehr- und Übungsbuch,* 3. Aufl. München, Deutschland: Carl Hanser Verlag GmbH & Co. KG, 2008.

[11] S. Vaidya, P. Ambad, und S. Bhosle, “Industry 4.0 – A Glimpse,” *Procedia Manufacturing*, Bd. 20, S. 233–238, 2018, DOI: 10.1016/j.promfg.2018.02.034.

[12] E. Capawa Fotsoh, N. Mebarki, P. Castagna, und P. Berruet, “A Classification for Reconfigurable Manufacturing Systems,” in *Springer Series in Advanced Manufacturing, Reconfigurable Manufacturing Systems: From Design to Implementation*, L. Benyoucef, Hrsg., Cham, Schweiz: Springer International Publishing, 2020, S. 11–28.

[13] B. Abdelilah, A. El Korchi, und M. A. Balambo, “Flexibility and agility: evolution and relationship,” *Journal of Manufacturing Technology Management*, Bd. 29, Nr. 7, S. 1138–1162, 2018, DOI: 10.1108/JMTM-03-2018-0090.

[14] M. Tabaa, F. Monteiro, H. Bensag, und A. Dandache, “Green Industrial Internet of Things from a smart industry perspectives,” *Energy Reports*, Bd. 6, S. 430–446, 2020, DOI: 10.1016/j.egyr.2020.09.022.

[15] M. Mabkhot, A. Al-Ahmari, B. Salah, und H. Alkhalefah, “Requirements of the Smart Factory System: A Survey and Perspective,” *Machines*, Bd. 6, Nr. 2, S. 23, 2018, DOI: 10.3390/machines6020023.

[16] O. Salunkhe und A. Fast-Berglund, “Increasing operational flexibility using Industry 4.0 enabling technologies in final assembly,” in *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, Cardiff, Großbritannien, 2020, S. 1–5, DOI: 10.1109/ICE/ITMC49519.2020.9198630.

[17] Plattform Industrie 4.0, “Leitbild 2030 für Industrie 4.0: Digitale Ökosysteme global gestalten,” 2019.

[18] A. Faul, N. Jazdi, und M. Weyrich, “Approach to interconnect existing industrial automation systems with the Industrial Internet,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Deutschland, 2016, S. 1–4, DOI: 10.1109/ETFA.2016.7733750.

[19] M. Weyrich und C. Ebert, “Reference Architectures for the Internet of Things,” *IEEE Software*, Bd. 33, Nr. 1, S. 112–116, 2016, DOI: 10.1109/MS.2016.20.

[20] N. Jazdi, “Cyber physical systems in the context of Industry 4.0,” in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, Cluj-Napoca, Rumänien, 2014, S. 1–4, DOI: 10.1109/AQTR.2014.6857843.

[21] G. Schuh, R. Anderl, R. Dumitrescu, A. Krüger, und M. ten Hompel, Hrsg., “Industrie 4.0 Maturity Index: Managing the Digital Transformation of Companies,” Update 2020, Acatech, 2020.

[22] K.-D. Thoben, S. Wiesner, und T. Wuest, ““Industrie 4.0” and Smart Manufacturing – A Review of Research Issues and Application Examples,” *International Journal of Automation Technology*, Bd. 11, Nr. 1, S. 4–16, 2017, DOI: 10.20965/ijat.2017.p0004.

[23] J. Qin, Y. Liu, und R. Grosvenor, “A Categorical Framework of Manufacturing for Industry 4.0 and Beyond,” *Procedia CIRP*, Bd. 52, S. 173–178, 2016, DOI: 10.1016/j.procir.2016.08.005.

[24] J. Otto, B. Vogel-Heuser, und O. Niggemann, “Automatic Parameter Estimation for Reusable Software Components of Modular and Reconfigurable Cyber-Physical Production Systems in the Domain of Discrete Manufacturing,” *IEEE Transactions on Industrial Informatics*, Bd. 14, Nr. 1, S. 275–282, 2018, DOI: 10.1109/TII.2017.2718729.

[25] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, und P. Dhariwal *et al.,* “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Vancouver, Kanada, 2020, S. 1877–1901.

[26] K. Simonyan und A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2014, arXiv: 1409.1556v6.

[27] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, und A. Guez *et al.,* “Mastering the game of Go without human knowledge,” *Nature*, Bd. 550, Nr. 7676, S. 354–359, 2017, DOI: 10.1038/nature24270.

[28] K. R. Chowdhary, *Fundamentals of Artificial Intelligence*. New Delhi, Indien: Springer India, 2020.

[29] O. Calin, *Deep Learning Architectures*. Cham, Schweiz: Springer International Publishing, 2020.

[30] B. Maschler, D. White, und M. Weyrich, “Anwendungsfälle und Methoden der künstlichen Intelligenz in der anwendungsorientierten Forschung im Kontext von Industrie 4.0,” in *Springer Reference Technik, Handbuch Industrie 4.0*, M. ten Hompel, B. Vogel-Heuser, und T. Bauernhansl, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2020, S. 1–15.

[31] G. Xu, M. Liu, J. Wang, Y. Ma, J. Wang, und F. Li *et al.,* “Data-Driven Fault Diagnostics and Prognostics for Predictive Maintenance: A Brief Overview *,” in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, Vancouver, Kanada, 2019, S. 103–108, DOI: 10.1109/COASE.2019.8843068.

[32] B. Lindemann, B. Maschler, N. Sahlab, und M. Weyrich, “A Survey on Anomaly Detection for Technical Systems using LSTM Networks,” *Computers in Industry*, Nr. 131, 103498, 2021, DOI: 10.1016/j.compind.2021.103498.

[33] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, und A. Troncoso, “Deep Learning for Time Series Forecasting: A Survey,” *Big Data*, Bd. 9, Nr. 1, S. 3–21, 2021, DOI: 10.1089/big.2020.0159.

[34] J. Wang, Y. Ma, L. Zhang, R. X. Gao, und D. Wu, “Deep learning for smart manufacturing: Methods and applications,” *Journal of Manufacturing Systems*, Bd. 48, S. 144–156, 2018, DOI: 10.1016/j.jmsy.2018.01.003.

[35] V. Vercruyssen, W. Meert, und J. Davis, “Transfer learning for time series anomaly detection,” in *Proceedings of the Workshop and Tutorial on Interactive Adaptive Learning ECML/PKDD 2017*, 2017, Bd. 1924, S. 27–37.

[36] R. Anderl, K. Bauer, T. Bauernhansl, J.-H. Fabian, M. Gebauer, und D. Goericke *et al.,* “Industrie 4.0 – Forschung für die Gestaltung der Zukunft: Impulsbericht des Forschungsbeirats der Plattform Industrie 4.0,” 2021.

[37] J. Metternich, T. Biegel, B. Cassoli, F. Hoffmann, N. Jourdan, und J. Rosemeyer *et al.,* “Künstliche Intelligenz zur Umsetzung von Industrie 4.0 im Mittelstand: Expertise des Forschungsbeirats der Plattform Industrie 4.0,” 2021.

[38] D. Vranjes, P. Topalis, und O. Niggemann, “Chancen und Herausforderungen für Künstliche Intelligenz in kleinen und mittelständischen Unternehmen: Best Practices zur erfolgreichen Integration von Künstlicher Intelligenz für den produzierenden Mittelstand,” (in de), *VDI Reports*, Nr. 2399, S. 399–412, 2022.

[39] J. Krauß, M. Frye, G. T. D. Beck, und R. H. Schmitt, “Selection and Application of Machine Learning- Algorithms in Production Quality,” in *Technologien für die intelligente Automation, Machine Learning for Cyber Physical Systems*, J. Beyerer, C. Kühnert, und O. Niggemann, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2019, S. 46–57.

[40] T. Bernard, C. Kühnert, und E. Campbell, “Web-based Machine Learning Platform for Condition-Monitoring,” in *Technologien für die intelligente Automation, Machine Learning for Cyber Physical Systems*, J. Beyerer, C. Kühnert, und O. Niggemann, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2019, S. 36–45.

[41] S. Gellrich, M.-A. Filz, A.-S. Wilde, T. Beganovic, A. Mattheus, und T. Abraham *et al.,* “Deep Transfer Learning for Improved Product Quality Prediction: A Case Study of Aluminum Gravity Die Casting,” *Procedia CIRP*, Bd. 104, S. 912–917, 2021, DOI: 10.1016/j.procir.2021.11.153.

[42] B. Maschler, N. Jazdi, und M. Weyrich, “Maschinelles Lernen für intelligente Automatisierungssysteme mit dezentraler Datenhaltung am Anwendungsfall Predictive Maintenance,” (in de), *VDI Reports*, Nr. 2351, S. 739–751, 2019, DOI: 10.18419/OPUS-10503.

[43] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, und J. Lin, “Machinery health prognostics: A systematic review from data acquisition to RUL prediction,” *Mechanical Systems and Signal Processing*, Bd. 104, S. 799–834, 2018, DOI: 10.1016/j.ymssp.2017.11.016.

[44] M. Russell und P. Wang, “Domain Adversarial Transfer Learning for Generalized Tool Wear Prediction,” in *Proceedings of the 12th Annual Conference of the PHM Society*, 2020, Bd. 12, S. 8, DOI: 10.36001/phmconf.2020.v12i1.1137.

[45] G. Michau und O. Fink, “Unsupervised transfer learning for anomaly detection: Application to complementary operating condition transfer,” *Knowledge-Based Systems*, Bd. 216, S. 106816, 2021, DOI: 10.1016/j.knosys.2021.106816.

[46] K. Ahlborn, G. Bachmann, F. Biegel, J. Bienert, S. Falk, und A. Fay *et al.,* “Technologieszenario: Künstliche Intelligenz in der Industrie 4.0,” Plattform Industrie 4.0, Bundesministerium für Wirtschaft und Energie (BMWi), 2019.

[47] A. Diez-Olivan, J. Del Ser, D. Galar, und B. Sierra, “Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0,” *Information Fusion*, Bd. 50, S. 92–111, 2019, DOI: 10.1016/j.inffus.2018.10.005.

[48] C. Xu, J. Wang, J. Zhang, und X. Li, “Anomaly detection of power consumption in yarn spinning using transfer learning,” *Computers & Industrial Engineering*, Bd. 152, S. 107015, 2021, DOI: 10.1016/j.cie.2020.107015.

[49] M. Canizo, I. Triguero, A. Conde, und E. Onieva, “Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study,” *Neurocomputing*, Bd. 363, S. 246–260, 2019, DOI: 10.1016/j.neucom.2019.07.034.

[50] R. Moradi und K. Groth, “On the application of transfer learning in prognostics and health management,” *Annual Conference of the PHM Society*, Bd. 12, Nr. 1, S. 8, 2020, DOI: 10.36001/phmconf.2020.v12i1.1300.

[51] Y. Fan, S. Nowaczyk, und T. Rögnvaldsson, “Transfer learning for remaining useful life prediction based on consensus self-organizing models,” *Reliability Engineering & System Safety*, Bd. 203, S. 107098, 2020, DOI: 10.1016/j.ress.2020.107098.

[52] B. Maschler, S. Ganssloser, A. Hablizel, und M. Weyrich, “Deep learning based soft sensors for industrial machinery,” *Procedia CIRP*, Bd. 99, S. 662–667, 2021, DOI: 10.1016/j.procir.2021.03.115.

[53] I. Torn und T. Vaneker, “Mass Personalization with Industry 4.0 by SMEs: a concept for collaborative networks,” *Procedia Manufacturing*, Bd. 28, S. 135–141, 2019, DOI: 10.1016/j.promfg.2018.12.022.

[54] X. Shang, Z. Shen, G. Xiong, F.-Y. Wang, S. Liu, und T. R. Nyberg *et al.,* “Moving from mass customization to social manufacturing: a footwear industry case study,” *International Journal of Computer Integrated Manufacturing*, Bd. 32, Nr. 2, S. 194–205, 2019, DOI: 10.1080/0951192X.2018.1550675.

[55] L. Block, M. Werner, M. Mikoschek, und S. Stegmüller, “Developing Technology Strategies for Flexible Automotive Products and Processes,” in *ARENA2036, Advances in Automotive Production Technology – Theory and Application*, P. Weißgraeber, F. Heieck, und C. Ackermann, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2021, S. 97–107.

[56] D. Sabadka, V. Molnár, und G. Fedorko, “Shortening of Life Cycle and Complexity Impact on the Automotive Industry,” *TEM Journal - Technology, Education, Management, Informatics*, Nr. 4, S. 1295–1301, 2019, DOI: 10.18421/TEM84-27.

[57] B. Maschler, T. Müller, A. Löcklin, und M. Weyrich, “Transfer Learning as an Enhancement for Reconfiguration Management of Cyber-Physical Production Systems,” *Procedia CIRP*, (accepted), 2022.

[58] T. Müller, S. Walth, N. Jazdi, und M. Weyrich, “Identification of Reconfiguration Demand and Generation of Alternative Configurations for Cyber-Physical Production Systems,” in *ARENA2036, Advances in Automotive Production Technology – Theory and Application*, P. Weißgraeber, F. Heieck, und C. Ackermann, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2021, S. 63–70.

[59] T. Müller, N. Jazdi, J.-P. Schmidt, und M. Weyrich, “Cyber-physical production systems: enhancement with a self-organized reconfiguration management,” *Procedia CIRP*, Bd. 99, S. 549–554, 2021, DOI: 10.1016/j.procir.2021.03.075.

[60] E. Jarvenpaa, N. Siltala, und M. Lanz, “Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems,” in *2016 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, Fort Worth, USA, 2016, S. 120–125, DOI: 10.1109/ISAM.2016.7750724.

[61] R. Heesch, N. Widulle, A. Köcher, A. Nordhausen, Vieira da Silva, L. M., und J. Putzke *et al.,* “Methoden der künstlichen Intelligenz für die automatisierte Planung von modularen Produktionsprozessen: Vergleich bestehender Ansätze und Vorstellung einer Benchmark,” (in de), *VDI Reports*, Nr. 2399, S. 439–451, 2022.

[62] H. Tercan, A. Guajardo, J. Heinisch, T. Thiele, C. Hopmann, und T. Meisen, “Transfer-Learning: Bridging the Gap between Real and Simulation Data for Machine Learning in Injection Molding,” *Procedia CIRP*, Bd. 72, S. 185–190, 2018, DOI: 10.1016/j.procir.2018.03.087.

[63] B. Maschler, T. T. H. Pham, und M. Weyrich, “Regularization-based Continual Learning for Anomaly Detection in Discrete Manufacturing,” *Procedia CIRP*, Bd. 104, 452-457, 2021, DOI: 10.1016/j.procir.2021.11.076.

[64] B. Maschler, S. Tatiyosyan, und M. Weyrich, “Regularization-based Continual Learning for Fault Prediction in Lithium-Ion Batteries,” in *2021 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering (ICME)*, Golf von Neapel, Italien, 2021, 1-6.

[65] C. A. Escobar, M. E. McGovern, und R. Morales-Menendez, “Quality 4.0: a review of big data challenges in manufacturing,” *Journal of Intelligent Manufacturing*, 2021, DOI: 10.1007/s10845-021-01765-4.

[66] F. von Bulow, F. Heinrich, und T. Meisen, “Fleet Management Approach for Manufacturers displayed at the Use Case of Battery Electric Vehicles,” in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Melbourne, Australien, 2021, S. 3218–3225, DOI: 10.1109/SMC52423.2021.9658680.

[67] A. Khan und K. Turowski, “A Preliminary Study on Industry 4.0,” *Journal of Industrial and Intelligent Information*, Bd. 4, Nr. 3, S. 230–234, 2016, DOI: 10.18178/jiii.4.3.230-234.

[68] Y. Cui, S. Kara, und K. C. Chan, “Monitoring and Control of Unstructured Manufacturing Big Data,” in *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapur, 2020, S. 928–932, DOI: 10.1109/IEEM45057.2020.9309975.

[69] R. Lorenz, “Data-Enabled Productivity Improvement in Manufacturing,” Dissertation, ETH Zurich, 2021.

[70] N. Widulle, J. Ehrhardt, M. Krantz, A. Liebert, A. Nordhausen, und O. Niggemann, “Eine Simulationsumgebung für flexible Cyber-Physische Produktionssysteme zur Generierung realistischer Datensätze für maschinelle Lernverfahren,” (in de), *VDI Reports*, Nr. 2399, S. 399–412, 2022.

[71] R. Rai, M. K. Tiwari, D. Ivanov, und A. Dolgui, “Machine learning in manufacturing and industry 4.0 applications,” *International Journal of Production Research*, Bd. 59, Nr. 16, S. 4773–4778, 2021, DOI: 10.1080/00207543.2021.1956675.

[72] R. S. Peres, X. Jia, J. Lee, K. Sun, A. W. Colombo, und J. Barata, “Industrial Artificial Intelligence in Industry 4.0 - Systematic Review, Challenges and Outlook,” *IEEE Access*, Bd. 8, S. 220121–220139, 2020, DOI: 10.1109/ACCESS.2020.3042874.

[73] M. Woschank, E. Rauch, und H. Zsifkovits, “A Review of Further Directions for Artificial Intelligence, Machine Learning, and Deep Learning in Smart Logistics,” *Sustainability*, Bd. 12, Nr. 9, S. 3760, 2020, DOI: 10.3390/su12093760.

[74] V. Nasir und F. Sassani, “A review on deep learning in machining and tool monitoring: methods, opportunities, and challenges,” *The International Journal of Advanced Manufacturing Technology*, Bd. 115, 9-10, S. 2683–2709, 2021, DOI: 10.1007/s00170-021-07325-7.

[75] S. Kamm, N. Jazdi, und M. Weyrich, “Knowledge Discovery in Heterogeneous and Unstructured Data of Industry 4.0 Systems: Challenges and Approaches,” *Procedia CIRP*, Bd. 104, S. 975–980, 2021, DOI: 10.1016/j.procir.2021.11.164.

[76] Y. Zhang, S. Ren, Y. Liu, und S. Si, “A big data analytics architecture for cleaner manufacturing and maintenance processes of complex products,” *Journal of Cleaner Production*, Bd. 142, S. 626–641, 2017, DOI: 10.1016/j.jclepro.2016.07.123.

[77] O. Niggemann, G. Biswas, J. S. Kinnebrew, N. Hranisavljevic, und A. Bunte, “Konzeptualisierung als Kernfrage des Maschinellen Lernens in der Produktion,” in *Springer Reference Technik, Handbuch Industrie 4.0*, M. ten Hompel, B. Vogel-Heuser, und T. Bauernhansl, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2020, S. 1–19.

[78] B. Taskazan, J. Navratil, M. Arnold, A. Murthi, G. Venkataraman, und B. Elder, “Not Your Grandfathers Test Set: Reducing Labeling Effort for Testing,” 2020, arXiv: 2007.05499v1.

[79] A. Culotta und A. McCallum, “Reducing labeling effort for structured prediction tasks,” in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*, Pittsburgh, USA, 2005, Bd. 2, S. 746–751.

[80] M. Nashaat, A. Ghosh, J. Miller, S. Quader, C. Marston, und J.-F. Puget, “Hybridization of Active Learning and Data Programming for Labeling Large Industrial Datasets,” in *2018 IEEE International Conference on Big Data (Big Data)*, Seattle, USA, 2018, S. 46–55, DOI: 10.1109/BigData.2018.8622459.

[81] M. Gao, Z. Zhang, G. Yu, S. Ö. Arık, L. S. Davis, und T. Pfister, “Consistency-Based Semi-supervised Active Learning: Towards Minimizing Labeling Cost,” in *Lecture Notes in Computer Science*, Bd. 12355, *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, und J.-M. Frahm, Hrsg., Cham, Schweiz: Springer International Publishing, 2020, S. 510–526.

[82] K.-L. Du und M. N. S. Swamy, *Neural Networks and Statistical Learning*. London, Großbritannien, Heidelberg, Deutschland, New York, USA, Dordrecht, Niederlande: Springer, 2014.

[83] J. Frochte, *Maschinelles Lernen: Grundlagen und Algorithmen in Python,* 2. Aufl. München, Deutschland: Hanser, 2019.

[84] V. Lomonaco, “Continual Learning with Deep Architectures,” Dissertation, Universität Bologna, Bologna, Italien, 2019.

[85] M. Heizmann, A. Braun, A. Frommknecht, M. Glitzner, M. Günther, und G. Hasna *et al.,* “Maschinelles Lernen in KMU: Künstliche Intelligenz im eigenen Unternehmen nutzen,” VDI-Statusreport, VDI, Düsseldorf, Deutschland, 2020.

[86] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, und P. Loos, “Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry,” *Procedia Computer Science*, Bd. 114, S. 242–249, 2017, DOI: 10.1016/j.procs.2017.09.066.

[87] H. Tercan, P. Deibert, und T. Meisen, “Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer,” *Journal of Intelligent Manufacturing*, 2021, DOI: 10.1007/s10845-021-01793-0.

[88] H. Tercan, A. Guajardo, und T. Meisen, “Industrial Transfer Learning: Boosting Machine Learning in Production,” in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finnland, 2019, S. 274–279, DOI: 10.1109/INDIN41052.2019.8972099.

[89] Y. Wang, B. Li, R. Luo, Y. Chen, N. Xu, und H. Yang, “Energy efficient neural networks for big data analytics,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Deutschland, 2014, S. 1–2, DOI: 10.7873/DATE.2014.358.

[90] T.-J. Yang, Y.-H. Chen, und V. Sze, “Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning,” in *2017 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017, S. 6071–6079, DOI: 10.1109/CVPR.2017.643.

[91] Y. Cao, Y. Chen, und D. Khosla, “Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition,” *International Journal of Computer Vision*, Bd. 113, Nr. 1, S. 54–66, 2015, DOI: 10.1007/s11263-014-0788-3.

[92] R. Birkhofer, G. Feldmeier, J. Kalhoff, C. Kleedörfer, M. Leidner, und R. Mildenberger *et al.,* “Life-Cycle-Management für Produkte und Systeme der Automation: Ein Leitfaden des Arbeitskreises Systemaspekte im ZVEI Fachverband Automation,” ZVEI Automation Division, Frankfurt, 2012.

[93] A. R. Hevner, S. T. March, J. Park, und S. Ram, “Design Science in Information Systems Research,” *MIS Quarterly*, Bd. 28, Nr. 1, S. 75–105, 2004, DOI: 10.2307/25148625.

[94] S. Gregor und A. R. Hevner, “Positioning and Presenting Design Science Research for Maximum Impact,” *MIS Quarterly*, Bd. 37, Nr. 2, S. 337–355, 2013, DOI: 10.2307/25148625.

[95] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[96] A. Hevner und S. Chatterjee, "Design Science Research in Information Systems," in *Integrated Series in Information Systems, Design Research in Information Systems*, A. Hevner und S. Chatterjee, Hrsg., Boston, USA: Springer US, 2010, S. 9–22.

[97] R. Baskerville, A. Baiyere, S. Gergor, A. Hevner, und M. Rossi, "Design Science Research Contributions: Finding a Balance between Artifact and Theory," *Journal of the Association for Information Systems*, Bd. 19, Nr. 5, S. 358–376, 2018, DOI: 10.17705/1jais.00495.

[98] K. Peffers, T. Tuunanen, M. A. Rothenberger, und S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, Bd. 24, Nr. 3, S. 45–77, 2007, DOI: 10.2753/MIS0742-1222240302.

[99] T. Trepper, "Forschungsmethodik," in *Fundierung der Konstruktion agiler Methoden*, T. Trepper, Hrsg., Wiesbaden, Deutschland: Springer Fachmedien Wiesbaden, 2015, S. 11–28.

[100] I. Goodfellow, Y. Bengio, und A. Courville, *Deep learning*. Cambridge, USA, London, Großbritannien: MIT Press, 2016.

[101] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, und H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, Bd. 405, Nr. 6789, S. 947–951, 2000, DOI: 10.1038/35016072.

[102] G. Klambauer, T. Unterthiner, A. Mayr, und S. Hochreiter, "Self-Normalizing Neural Networks," in *Advances in Neural Information Processing Systems*, Long Beach, USA, 2017, S. 972–981.

[103] Y. LeCun, "Generalization and network design strategies: Technical Report CRG-TR-89-4," University of Toronto Connectionist Research Group (CRG), 1989.

[104] S. Hochreiter und J. Schmidhuber, "Long short-term memory," *Neural Computation*, Bd. 9, Nr. 8, S. 1735–1780, 1997, DOI: 10.1162/neco.1997.9.8.1735.

[105] S. Skansi, "Recurrent Neural Networks," in *Undergraduate Topics in Computer Science, Introduction to Deep Learning*, S. Skansi, Hrsg., Cham, Schweiz: Springer International Publishing, 2018, S. 135–152.

[106] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J (American Institut of Chemical Engineers Journal)*, Bd. 37, Nr. 2, S. 233–243, 1991, DOI: 10.1002/aic.690370209.

[107] F. J. Kurfess, "Neural Networks and Structured Knowledge: Knowledge Representation and Reasoning," *Applied Intelligence*, Bd. 11, Nr. 1, S. 5–13, 1999, DOI: 10.1023/A:1008327412259.

[108] J. C. Hoskins und D. M. Himmelblau, "Artificial neural network models of knowledge representation in chemical engineering," *Computers & Chemical Engineering*, Bd. 12, 9-10, S. 881–890, 1988, DOI: 10.1016/0098-1354(88)87015-7.

[109] F. Chollet, *Deep learning with Python*. Shelter Island, USA: Manning, 2018.

[110] C. Z. Cremer, "Deep limitations? Examining expert disagreement over deep learning," *Progress in Artificial Intelligence*, 2021, DOI: 10.1007/s13748-021-00239-1.

[111] B. Maschler, H. Vietz, H. Tercan, C. Bitter, T. Meisen, und M. Weyrich, "Insights and Example Use Cases on Industrial Transfer Learning," *Procedia CIRP*, Nr. 107, 511-516, 2022, DOI: 10.1016/j.procir.2022.05.017.

[112] B. Maschler, T. Knodel, und M. Weyrich, "Towards Deep Industrial Transfer Learning: Clustering for Transfer Case Selection," in *2022 27th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Stuttgart, Deutschland, 2022, S. 1–8.

[113] S. J. Pan und Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, Bd. 22, Nr. 10, S. 1345–1359, 2010, DOI: 10.1109/TKDE.2009.191.

[114] K. Weiss, T. M. Khoshgoftaar, und D. Wang, “A survey of transfer learning,” *Journal of Big Data*, Bd. 3, Nr. 1, 2016, DOI: 10.1186/s40537-016-0043-6.

[115] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, und H. Zhu *et al.,* “A Comprehensive Survey on Transfer Learning,” *Proceedings of the IEEE*, Bd. 109, Nr. 1, S. 43–76, 2021, DOI: 10.1109/JPROC.2020.3004555.

[116] B. Maschler und M. Weyrich, “Deep Transfer Learning for Industrial Automation,” *Industrial Electronics Magazine*, Bd. 15, Nr. 2, 65-75, 2021, DOI: 10.1109/MIE.2020.3034884.

[117] Y. Zhang und Q. Yang, “An overview of multi-task learning,” *National Science Review*, Bd. 5, Nr. 1, S. 30–43, 2018, DOI: 10.1093/nsr/nwx105.

[118] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, und N. Díaz-Rodríguez, “Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges,” *Information Fusion*, Bd. 58, S. 52–68, 2020, DOI: 10.1016/j.inffus.2019.12.004.

[119] R. Raina, A. Battle, H. Lee, B. Packer, und A. Y. Ng, “Self-taught learning,” in *2007 24th International Conference on Machine Learning (ICML): Proceedings*, Corvalis, USA, 2007, S. 759–766, DOI: 10.1145/1273496.1273592.

[120] J. Zhang, W. Li, P. Ogunbona, und D. Xu, “Recent Advances in Transfer Learning for Cross-Dataset Visual Recognition,” *ACM Computing Surveys*, Bd. 52, Nr. 1, S. 1–38, 2019, DOI: 10.1145/3291124.

[121] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, und C. Liu, “A Survey on Deep Transfer Learning,” *2018 27th International Conference on Artificial Neural Networks (ICANN); Lecture Notes in Computer Science*, Bd. 11141, S. 270–279, 2018, DOI: 10.1007/978-3-030-01424-7_27.

[122] Z. Wang, Z. Dai, B. Poczos, und J. Carbonell, “Characterizing and Avoiding Negative Transfer,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, USA, 2019, S. 11285–11294, DOI: 10.1109/CVPR.2019.01155.

[123] X. Yu, M. Wu, Y. Jian, K. E. Bennin, M. Fu, und C. Ma, “Cross-company defect prediction via semi-supervised clustering-based data filtering and MSTrA-based transfer learning,” *Soft Computing*, Bd. 22, Nr. 10, S. 3461–3472, 2018, DOI: 10.1007/s00500-018-3093-1.

[124] L. Ge, J. Gao, H. Ngo, K. Li, und A. Zhang, “On handling negative transfer and imbalanced distributions in multiple source transfer learning,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, Bd. 7, Nr. 4, S. 254–271, 2014, DOI: 10.1002/sam.11217.

[125] H. Tang, K. Chen, und K. Jia, “Unsupervised Domain Adaptation via Structurally Regularized Deep Clustering,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, USA, 2020, S. 8722–8732, DOI: 10.1109/CVPR42600.2020.00875.

[126] S. Ben-David und R. S. Borbely, “A notion of task relatedness yielding provable multiple-task learning guarantees,” *Machine Learning*, Bd. 73, Nr. 3, S. 273–287, 2008, DOI: 10.1007/s10994-007-5043-5.

[127] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in Cognitive Sciences*, Bd. 3, Nr. 4, S. 128–135, 1999, DOI: 10.1016/S1364-6613(99)01294-2.

[128] D. Maltoni und V. Lomonaco, “Continuous learning in single-incremental-task scenarios,” *Neural Networks*, Bd. 116, Nr. 116, S. 56–73, 2019, DOI: 10.1016/j.neunet.2019.03.010.

[129] M. Mermillod, A. Bugaiska, und P. Bonin, “The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects,” *Frontiers in psychology*, Bd. 4, S. 504, 2013, DOI: 10.3389/fpsyg.2013.00504.

[130] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, und S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, Bd. 113, S. 54–71, 2019, DOI: 10.1016/j.neunet.2019.01.012.

[131] R. Ribani und M. Marengoni, “A Survey of Transfer Learning for Convolutional Neural Networks,” in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, Rio de Janeiro, Brasilien, 2019, S. 47–57, DOI: 10.1109/SIBGRAPI-T.2019.00010.

[132] W. Wang, V. W. Zheng, H. Yu, und C. Miao, “A Survey of Zero-Shot Learning,” *ACM Transactions on Intelligent Systems and Technology*, Bd. 10, Nr. 2, S. 1–37, 2019, DOI: 10.1145/3293318.

[133] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, und Y. Gao, “A survey on federated learning,” *Knowledge-Based Systems*, Bd. 216, S. 106775, 2021, DOI: 10.1016/j.knosys.2021.106775.

[134] L. Li, Y. Fan, und K.-Y. Lin, “A Survey on federated learning *,” in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, Singapur, 2020, S. 791–796, DOI: 10.1109/ICCA51439.2020.9264412.

[135] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, und B. Aguera y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, USA, 2017, Bd. 54, S. 1273–1282.

[136] L. Rokach, “A survey of Clustering Algorithms,” in *Data Mining and Knowledge Discovery Handbook*, O. Maimon und L. Rokach, Hrsg., Boston, USA: Springer US, 2010, S. 269–298.

[137] L. Vendramin, R. J. G. B. Campello, und E. R. Hruschka, “Relative clustering validity criteria: A comparative overview,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, Bd. 3, Nr. 4, S. 209–235, 2010, DOI: 10.1002/sam.10080.

[138] Y. Liu, Z. Li, H. Xiong, X. Gao, und J. Wu, “Understanding of Internal Clustering Validation Measures,” in *2010 IEEE International Conference on Data Mining*, Sydney, Australien, 2010, S. 911–916, DOI: 10.1109/ICDM.2010.35.

[139] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, Bd. 20, S. 53–65, 1987, DOI: 10.1016/0377-0427(87)90125-7.

[140] T. Calinski und J. Harabasz, “A dendrite method for cluster analysis,” *Communications in Statistics - Theory and Methods*, Bd. 3, Nr. 1, S. 1–27, 1974, DOI: 10.1080/03610927408827101.

[141] B. S. Everitt, S. Landau, M. Leese, und D. Stahl, “Measurement of Proximity,” in *Wiley Series in Probability and Statistics, Cluster Analysis*, B. S. Everitt, S. Landau, M. Leese, und D. Stahl, Hrsg., Chichester, Großbritannien: John Wiley & Sons, Ltd, 2011, S. 43–69.

[142] L. Kaufman und P. J. Rousseeuw, *Finding Groups in Data*. Hoboken, USA: John Wiley & Sons, Inc, 1990.

[143] J. Han, M. Kamber, und J. Pei, “Getting to Know Your Data,” in *Data Mining*: Elsevier, 2012, S. 39–82.

[144] B. S. Everitt, S. Landau, M. Leese, und D. Stahl, “Hierarchical Clustering,” in *Wiley Series in Probability and Statistics, Cluster Analysis*, B. S. Everitt, S. Landau, M. Leese, und D. Stahl, Hrsg., Chichester, Großbritannien: John Wiley & Sons, Ltd, 2011, S. 71–110.

[145] T. Cover und P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, Bd. 13, Nr. 1, S. 21–27, 1967, DOI: 10.1109/TIT.1967.1053964.

[146] T. Zhang, R. Ramakrishnan, und M. Livny, “BIRCH: A New Data Clustering Algorithm and Its Applications,” *Data Mining and Knowledge Discovery*, Bd. 1, Nr. 2, S. 141–182, 1997, DOI: 10.1023/A:1009783824328.

[147] S. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory*, Bd. 28, Nr. 2, S. 129–137, 1982, DOI: 10.1109/TIT.1982.1056489.

[148] R. Scitovski, K. Sabo, F. Martínez-Álvarez, und Š. Ungar, *Cluster Analysis and Applications*. Cham, Schweiz: Springer International Publishing, 2021.

[149] D. Sculley, “Web-scale k-means clustering,” in *Proceedings of the 19th international conference on World wide web - WWW '10*, Raleigh, USA, 2010, S. 1177, DOI: 10.1145/1772690.1772862.

[150] M. Ester, H.-P. Kriegel, J. Sander, und X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *The 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, Oregon, USA, 1996, Bd. 96, 226-231.

[151] Statistisches Bundesamt, Hrsg., “Systematik der Wirtschaftszweige mit Erläuterungen: Unternehmens- und Betriebssystematik,” Stuttgart, Deutschland, 1979.

[152] F. Tao, Q. Qi, A. Liu, und A. Kusiak, “Data-driven smart manufacturing,” *Journal of Manufacturing Systems*, Bd. 48, S. 157–169, 2018, DOI: 10.1016/j.jmsy.2018.01.006.

[153] Y. Wang, M. Perry, D. Whitlock, und J. W. Sutherland, “Detecting anomalies in time series data from a manufacturing system using recurrent neural networks,” *Journal of Manufacturing Systems*, 2020, DOI: 10.1016/j.jmsy.2020.12.007.

[154] A. Essien und C. Giannetti, “A Deep Learning Model for Smart Manufacturing Using Convolutional LSTM Neural Network Autoencoders,” *IEEE Transactions on Industrial Informatics*, Bd. 16, Nr. 9, S. 6069–6078, 2020, DOI: 10.1109/TII.2020.2967556.

[155] N. Sahlab, S. Kamm, T. Muller, N. Jazdi, und M. Weyrich, “Knowledge Graphs as Enhancers of Intelligent Digital Twins,” in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, Victoria, Kanada, 2021, S. 19–24, DOI: 10.1109/ICPS49255.2021.9468219.

[156] P. R. d. O. Da Costa, A. Akçay, Y. Zhang, und U. Kaymak, “Remaining useful lifetime prediction via deep domain adaptation,” *Reliability Engineering & System Safety*, Bd. 195, S. 106682, 2020, DOI: 10.1016/j.ress.2019.106682.

[157] B. Maschler, S. Kamm, und M. Weyrich, “Deep Industrial Transfer Learning at Runtime for Image Recognition,” *at - Automatisierungstechnik*, Bd. 69, Nr. 3, 211-220, 2021, DOI: 10.1515/auto-2020-0119.

[158] B. Maschler, H. Vietz, N. Jazdi, und M. Weyrich, “Continual Learning of Fault Prediction for Turbofan Engines using Deep Learning with Elastic Weight Consolidation,” in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Wien, Österreich, 2020, S. 959–966, DOI: 10.1109/ETFA46521.2020.9211903.

[159] J. L. McClelland, B. L. McNaughton, und R. C. O'Reilly, “Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory,” *Psychological Review*, Bd. 102, Nr. 3, S. 419–457, 1995, DOI: 10.1037/0033-295X.102.3.419.

[160] B. Maschler, S. Kamm, N. Jazdi, und M. Weyrich, “Distributed Cooperative Deep Transfer Learning for Industrial Image Recognition,” *Procedia CIRP*, Bd. 93, S. 437–442, 2020, DOI: 10.1016/j.procir.2020.03.056.

[161] R. Kemker, M. McClure, A. Abitino, T. Hayes, und C. Kanan, “Measuring Catastrophic Forgetting in Neural Networks,” *Proceedings of the AAAI Conference on Artificial Intelligence*, Bd. 32, Nr. 1, 2018.

[162] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, und A. A. Rusu *et al.,* “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences of the United States of America*, Bd. 114, Nr. 13, S. 3521–3526, 2017, DOI: 10.1073/pnas.1611835114.

[163] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, und R. Pascanu *et al.,* “Progress & Compress: A scalable framework for continual learning,” *Proceedings of Machine Learning Research*, Nr. 80, S. 4528–4537, 2018. [Online]. Verfügbar unter: http://proceedings.mlr.press/v80/schwarz18a/schwarz18a.pdf

[164] F. Zenke, B. Poole, und S. Ganguli, “Continual Learning Through Synaptic Intelligence,” *Proceedings of Machine Learning Research*, Nr. 70, S. 3987–3995, 2017. [Online]. Verfügbar unter: http://proceedings.mlr.press/v70/zenke17a/zenke17a.pdf

[165] Z. Li und D. Hoiem, “Learning without Forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 40, Nr. 12, S. 2935–2947, 2018, DOI: 10.1109/TPAMI.2017.2773081.

[166] R. Pascanu und Y. Bengio, “Revisiting Natural Gradient for Deep Networks,” 2013.

[167] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, und T. Tuytelaars, “Memory Aware Synapses: Learning What (not) to Forget,” in *Lecture Notes in Computer Science*, Bd. 11207, *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, und Y. Weiss, Hrsg., Cham, Schweiz: Springer International Publishing, 2018, S. 144–161.

[168] G. M. van de Ven und A. S. Tolias, “Three scenarios for continual learning,” *2018 32nd Conference on Neural Information Processing Systems (NeurIPS) Continual Learning Workshop.* [Online]. Verfügbar unter: http://arxiv.org/pdf/1904.07734v1

[169] Y.-C. Hsu, Y.-C. Liu, A. Ramasamy, und Z. Kira, “Re-evaluating Continual Learning Scenarios: A Categorization and Case for Strong Baselines,” *2018 32nd Conference on Neural Information Processing Systems (NeurIPS) Continual Learning Workshop.* [Online]. Verfügbar unter: https://arxiv.org/abs/1810.12488

[170] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, und A. Leonardis *et al.,* “A continual learning survey: Defying forgetting in classification tasks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, DOI: 10.1109/TPAMI.2021.3057446.

[171] S. Katzschner, “Entwicklung eines Deep Industrial Transfer Learning Algorithmus zur Verschleißvorhersage mittels Domänen-Adaption,” Masterarbeit Nr. 3199, IAS, Universität Stuttgart, 2021.

[172] H. Daume III und D. Marcu, “Domain Adaptation for Statistical Classifiers,” *Journal of Artificial Intelligence Research*, Bd. 26, S. 101–126, 2006, DOI: 10.1613/jair.1872.

[173] R. Gopalan, R. Li, und R. Chellappa, “Domain adaptation for object recognition: An unsupervised approach,” in *2011 International Conference on Computer Vision*, Barcelona, Spanien, 2011, S. 999–1006, DOI: 10.1109/ICCV.2011.6126344.

[174] S. Kullback und R. A. Leibler, “On Information and Sufficiency,” *The Annals of Mathematical Statistics*, Bd. 22, Nr. 1, S. 79–86, 1951.

[175] Y. Ganin und V. Lempitsky, “Unsupervised Domain Adaptation by Backpropagation,” in *2015 32nd International Conference on Machine Learning (ICML): Proceedings*, Lille, Frankreich, 2015, Bd. 37, S. 1180–1189.

[176] X. Li, W. Zhang, Q. Ding, und J.-Q. Sun, “Multi-Layer domain adaptation method for rolling bearing fault diagnosis,” *Signal Processing*, Bd. 157, S. 180–197, 2019, DOI: 10.1016/j.sigpro.2018.12.005.

[177] S. J. Pan, I. W. Tsang, J. T. Kwok, und Q. Yang, “Domain adaptation via transfer component analysis,” *IEEE transactions on neural networks*, Bd. 22, Nr. 2, S. 199–210, 2011, DOI: 10.1109/TNN.2010.2091281.

[178] X. Glorot, A. Bordes, und Y. Bengio, “Domain adaptation for large-scale sentiment classification: a deep learning approach,” in *2011 28th International Conference on Machine Learning (ICML): Proceedings*, Bellevue, USA, 2011, S. 513–520.

[179] M. A. Morid, A. Borjali, und G. Del Fiol, “A scoping review of transfer learning research on medical image analysis using ImageNet,” *Computers in biology and medicine*, Bd. 128, S. 104115, 2021, DOI: 10.1016/j.compbiomed.2020.104115.

[180] S. Kaushik, A. Choudhury, N. Dasgupta, S. Natarajan, L. A. Pickett, und V. Dutt, “Ensemble of Multi-headed Machine Learning Architectures for Time-Series Forecasting of Healthcare Expenditures,” in *Algorithms for*

Intelligent Systems, Applications of Machine Learning, P. Johri, J. K. Verma, und S. Paul, Hrsg., Singapur: Springer Singapore, 2020, S. 199–216.

[181] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, und S. L. Baxter *et al.,* "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," *Cell*, Bd. 172, Nr. 5, 1122-1131.e9, 2018, DOI: 10.1016/j.cell.2018.02.010.

[182] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, und S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, Bd. 469, S. 28–51, 2022, DOI: 10.1016/j.neucom.2021.10.021.

[183] M. Biesialska, K. Biesialska, und M. R. Costa-jussà, "Continual Lifelong Learning in Natural Language Processing: A Survey," in *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spanien (Online), S. 6523–6541, DOI: 10.18653/v1/2020.coling-main.574.

[184] B. Fresz, "Development and Evaluation of Regularization-Based Continual Learning Methods for Industrial Applications," Masterarbeit Nr. 3210, IAS, Universität Stuttgart, 2021.

[185] M. Long, J. Wang, G. Ding, S. J. Pan, und P. S. Yu, "Adaptation Regularization: A General Framework for Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, Bd. 26, Nr. 5, S. 1076–1089, 2014, DOI: 10.1109/TKDE.2013.111.

[186] T. Han, C. Liu, W. Yang, und D. Jiang, "Deep transfer network with joint distribution adaptation: A new intelligent fault diagnosis framework for industry application," *ISA Transactions*, Bd. 97, S. 269–281, 2020, DOI: 10.1016/j.isatra.2019.08.012.

[187] K. He, X. Zhang, S. Ren, und J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, S. 770–778, DOI: 10.1109/CVPR.2016.90.

[188] J. Devlin, M.-W. Chang, K. Lee, und K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018, arXiv: 1810.04805v2.

[189] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, und A. Gesmundo *et al.,* "Parameter-Efficient Transfer Learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, USA, 2019, S. 2790–2799.

[190] J. Cao, J. Wang, W. Hamza, K. Vanee, und S.-W. Li, "Style Attuned Pre-Training and Parameter Efficient Fine-Tuning for Spoken Language Understanding," in *Interspeech 2020*, Shanghai, China, 2020, S. 1570–1574, DOI: 10.21437/Interspeech.2020-2907.

[191] V. Chandola, A. Banerjee, und V. Kumar, "Anomaly detection," *ACM Computing Surveys*, Bd. 41, Nr. 3, S. 1–58, 2009, DOI: 10.1145/1541880.1541882.

[192] R. Chalapathy und S. Chawla, "Deep Learning for Anomaly Detection: A Survey," 2019, arXiv: 1901.03407v2.

[193] S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, und D. Song, "Anomalous Example Detection in Deep Learning: A Survey," *IEEE Access*, Bd. 8, S. 132330–132347, 2020, DOI: 10.1109/ACCESS.2020.3010274.

[194] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, und K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, Bd. 22, S1, S. 949–961, 2019, DOI: 10.1007/s10586-017-1117-8.

[195] M. M. Moya, M. W. Koch, und L. D. Hostetler, "One-class classifier networks for target recognition applications," *NASA STI/Recon Technical Report N*, Bd. 93, S. 24043, 1993.

[196] D. Tax, "One-class classification: Concept learning in the absence of counter-examples," Dissertation, Delft University of Technology, Delft, Niederlande, 2001.

[197] D. P. Kingma und M. Welling, “An Introduction to Variational Autoencoders,” *Foundations and Trends® in Machine Learning*, Bd. 12, Nr. 4, S. 307–392, 2019, DOI: 10.1561/2200000056.

[198] A. A. Cook, G. Misirli, und Z. Fan, “Anomaly Detection for IoT Time-Series Data: A Survey,” *IEEE Internet of Things Journal*, Bd. 7, Nr. 7, S. 6481–6494, 2020, DOI: 10.1109/JIOT.2019.2958185.

[199] A. L. Alfeo, M. G. Cimino, G. Manco, E. Ritacco, und G. Vaglini, “Using an autoencoder in the design of an anomaly detector for smart manufacturing,” *Pattern Recognition Letters*, Bd. 136, S. 272–278, 2020, DOI: 10.1016/j.patrec.2020.06.008.

[200] M. Carletti, C. Masiero, A. Beghi, und G. A. Susto, “A deep learning approach for anomaly detection with industrial time series data: a refrigerators manufacturing case study,” *Procedia Manufacturing*, Bd. 38, S. 233–240, 2019, DOI: 10.1016/j.promfg.2020.01.031.

[201] G. Raman MR, N. Somu, und A. P. Mathur, “A multilayer perceptron model for anomaly detection in water treatment plants,” *International Journal of Critical Infrastructure Protection*, Bd. 31, S. 100393, 2020, DOI: 10.1016/j.ijcip.2020.100393.

[202] Z. Niu, K. Yu, und X. Wu, “LSTM-Based VAE-GAN for Time-Series Anomaly Detection,” *Sensors*, Bd. 20, Nr. 13, S. 3738, 2020, DOI: 10.3390/s20133738.

[203] B. Lindemann, N. Jazdi, und M. Weyrich, “Detektion von Anomalien zur Qualitätssicherung basierend auf Sequence-to-Sequence LSTM Netzen,” *at - Automatisierungstechnik*, Bd. 67, Nr. 12, S. 1058–1068, 2019, DOI: 10.1515/auto-2019-0076.

[204] R.-J. Hsieh, J. Chou, und C.-H. Ho, “Unsupervised Online Anomaly Detection on Multivariate Sensing Time Series Data for Smart Manufacturing,” in *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, Kaohsiung, Taiwan, 2019, S. 90–97, DOI: 10.1109/SOCA.2019.00021.

[205] P. Liang, H.-D. Yang, W.-S. Chen, S.-Y. Xiao, und Z.-Z. Lan, “Transfer learning for aluminium extrusion electricity consumption anomaly detection via deep neural networks,” *International Journal of Computer Integrated Manufacturing*, Bd. 31, 4-5, S. 396–405, 2018, DOI: 10.1080/0951192X.2017.1363410.

[206] R. Müller, F. Ritz, S. Illium, und C. Linnhoff-Popien, “Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning,” in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, Online Streaming, 2021, S. 49–56, DOI: 10.5220/0010185800490056.

[207] S. Baireddy, S. R. Desai, J. L. Mathieson, R. H. Foster, M. W. Chan, und M. L. Comer *et al.,* “Spacecraft Time-Series Anomaly Detection Using Transfer Learning,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Nashville, USA, 2021, S. 1951–1960, DOI: 10.1109/CVPRW53098.2021.00223.

[208] I. Ullah und Q. H. Mahmoud, “Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks,” *IEEE Access*, Bd. 9, S. 103906–103926, 2021, DOI: 10.1109/ACCESS.2021.3094024.

[209] H. Wang, W. Lu, S. Tang, und Y. Song, “Predict industrial equipment failure with time windows and transfer learning,” *Applied Intelligence*, 2021, DOI: 10.1007/s10489-021-02441-z.

[210] G.-B. Huang, Q.-Y. Zhu, und C.-K. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, Bd. 70, 1-3, S. 489–501, 2006, DOI: 10.1016/j.neucom.2005.12.126.

[211] C. Okoh, R. Roy, J. Mehnen, und L. Redding, “Overview of Remaining Useful Life Prediction Techniques in Through-life Engineering Services,” *Procedia CIRP*, Bd. 16, S. 158–163, 2014, DOI: 10.1016/j.procir.2014.02.006.

[212] H. M. Elattar, H. K. Elminir, und A. M. Riad, “Prognostics: a literature review,” *Complex & Intelligent Systems*, Bd. 2, Nr. 2, S. 125–154, 2016, DOI: 10.1007/s40747-016-0019-3.

[213] D. An, N. H. Kim, und J.-H. Choi, “Practical options for selecting data-driven or physics-based prognostics algorithms with reviews,” *Reliability Engineering & System Safety*, Bd. 133, S. 223–236, 2015, DOI: 10.1016/j.ress.2014.09.014.

[214] Y. Wang, Y. Zhao, und S. Addepalli, “Remaining Useful Life Prediction using Deep Learning Approaches: A Review,” *Procedia Manufacturing*, Bd. 49, S. 81–88, 2020, DOI: 10.1016/j.promfg.2020.06.015.

[215] M. G. Pecht und M. Kang, Hrsg., *Prognostics and Health Management of Electronics,* 2. Aufl. Chichester, Großbritannien: John Wiley and Sons Ltd, 2018.

[216] H. Tian, P. Qin, K. Li, und Z. Zhao, “A review of the state of health for lithium-ion batteries: Research status and suggestions,” *Journal of Cleaner Production*, Bd. 261, S. 120813, 2020, DOI: 10.1016/j.jclepro.2020.120813.

[217] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, und M. Weyrich, “A survey on long short-term memory networks for time series prediction,” *Procedia CIRP*, Bd. 99, S. 650–655, 2021, DOI: 10.1016/j.procir.2021.03.088.

[218] F. von Bülow, J. Mentz, und T. Meisen, “State of health forecasting of Lithium-ion batteries applicable in real-world operational conditions,” *Journal of Energy Storage*, Bd. 44, S. 103439, 2021, DOI: 10.1016/j.est.2021.103439.

[219] A. Bonfitto, E. Ezemobi, N. Amati, S. Feraco, A. Tonoli, und S. Hegde, “State of Health Estimation of Lithium Batteries for Automotive Applications with Artificial Neural Networks,” in *2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, Torino, Italien, 2019, S. 1–5, DOI: 10.23919/EETA.2019.8804567.

[220] A. Bonfitto, “A Method for the Combined Estimation of Battery State of Charge and State of Health Based on Artificial Neural Networks,” *Energies*, Bd. 13, Nr. 10, S. 2548, 2020, DOI: 10.3390/en13102548.

[221] L. Kirschbaum, D. Roman, V. Robu, und D. Flynn, “Deep Learning Pipeline for State-of-Health Classification of Electromagnetic Relays,” in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, Kyoto, Japan, 2021, S. 1–7, DOI: 10.1109/ISIE45552.2021.9576278.

[222] B. Lindemann, N. Jazdi, und M. Weyrich, “Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks,” in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Hong Kong, China, 2020, S. 1003–1010, DOI: 10.1109/CASE48305.2020.9216855.

[223] J. Li, J. Lu, C. Chen, J. Ma, und X. Liao, “Tool wear state prediction based on feature-based transfer learning,” *The International Journal of Advanced Manufacturing Technology*, Bd. 113, 11-12, S. 3283–3301, 2021, DOI: 10.1007/s00170-021-06780-6.

[224] N. Beganovic und D. Söffker, “Remaining lifetime modeling using State-of-Health estimation,” *Mechanical Systems and Signal Processing*, Bd. 92, S. 107–123, 2017, DOI: 10.1016/j.ymssp.2017.01.031.

[225] Y. Cao, M. Jia, P. Ding, und Y. Ding, “Transfer learning for remaining useful life prediction of multi-conditions bearings based on bidirectional-GRU network,” *Measurement*, Bd. 178, S. 109287, 2021, DOI: 10.1016/j.measurement.2021.109287.

[226] A. Zhang, H. Wang, S. Li, Y. Cui, Z. Liu, und G. Yang *et al.,* “Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation,” *Applied Sciences*, Bd. 8, Nr. 12, S. 2416, 2018, DOI: 10.3390/app8122416.

[227] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan, und X. Chen, “Deep Transfer Learning Based on Sparse Autoencoder for Remaining Useful Life Prediction of Tool in Manufacturing,” *IEEE Transactions on Industrial Informatics*, Bd. 15, Nr. 4, S. 2416–2425, 2019, DOI: 10.1109/TII.2018.2881543.

[228] M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, und X. Li, “Adversarial Transfer Learning for Machine Remaining Useful Life Prediction,” in *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Detroit, USA, 2020, S. 1–7, DOI: 10.1109/ICPHM49022.2020.9187053.

[229] Y. Tan und G. Zhao, “Transfer Learning with Long Short-Term Memory Network for State-of-Health Prediction of Lithium-Ion Batteries,” *IEEE Transactions on Industrial Electronics*, Bd. 67, Nr. 10, S. 8723–8731, 2020, DOI: 10.1109/TIE.2019.2946551.

[230] H. Zhang, Q. Zhang, S. Shao, T. Niu, X. Yang, und H. Ding, “Sequential Network with Residual Neural Network for Rotatory Machine Remaining Useful Life Prediction Using Deep Transfer Learning,” *Shock and Vibration*, Bd. 2020, S. 1–16, 2020, DOI: 10.1155/2020/8888627.

[231] H. Cheng, X. Kong, G. Chen, Q. Wang, und R. Wang, “Transferable convolutional neural network based remaining useful life prediction of bearing under multiple failure behaviors,” *Measurement*, Bd. 168, S. 108286, 2021, DOI: 10.1016/j.measurement.2020.108286.

[232] H. Cheng, X. Kong, Q. Wang, H. Ma, S. Yang, und G. Chen, “Deep transfer learning based on dynamic domain adaptation for remaining useful life prediction under different working conditions,” *Journal of Intelligent Manufacturing*, 2021, DOI: 10.1007/s10845-021-01814-y.

[233] Y. Ding, M. Jia, Q. Miao, und P. Huang, “Remaining useful life estimation using deep metric transfer learning for kernel regression,” *Reliability Engineering & System Safety*, Bd. 212, S. 107583, 2021, DOI: 10.1016/j.ress.2021.107583.

[234] M. Gribbestad, M. U. Hassan, und I. A. Hameed, “Transfer Learning for Prognostics and Health Management (PHM) of Marine Air Compressors,” *Journal of Marine Science and Engineering*, Bd. 9, Nr. 1, S. 47, 2021, DOI: 10.3390/jmse9010047.

[235] M. Marei, S. E. Zaatari, und W. Li, “Transfer learning enabled convolutional neural networks for estimating health state of cutting tools,” *Robotics and Computer-Integrated Manufacturing*, Bd. 71, S. 102145, 2021, DOI: 10.1016/j.rcim.2021.102145.

[236] F. Zeng, Y. Li, Y. Jiang, und G. Song, “An online transfer learning-based remaining useful life prediction method of ball bearings,” *Measurement*, Bd. 176, S. 109201, 2021, DOI: 10.1016/j.measurement.2021.109201.

[237] W. Zhang, X. Li, H. Ma, Z. Luo, und X. Li, “Transfer learning using deep representation regularization in remaining useful life prediction across operating conditions,” *Reliability Engineering & System Safety*, Bd. 211, S. 107556, 2021, DOI: 10.1016/j.ress.2021.107556.

[238] N. Tosun, “Determination of optimum parameters for multi-performance characteristics in drilling by using grey relational analysis,” *The International Journal of Advanced Manufacturing Technology*, Bd. 28, 5-6, S. 450–455, 2006, DOI: 10.1007/s00170-004-2386-y.

[239] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, und O. Grisel *et al.,* “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, Bd. 12, S. 2825–2830, 2011.

[240] S. Miyamoto und A. Terami, “Inductive vs. transductive clustering using kernel functions and pairwise constraints,” in *2011 11th International Conference on Intelligent Systems Design and Applications*, Cordoba, Spanien, 2011, S. 1258–1264, DOI: 10.1109/ISDA.2011.6121832.

[241] C. Homburg, “Datenanalyse und -interpretation,” in *Grundlagen des Marketingmanagements*, C. Homburg, Hrsg., Wiesbaden, Deutschland: Springer Fachmedien Wiesbaden, 2017, S. 87–118.

[242] M. Lane, A. Maiocco, S. K. Bhatia, und S. Climer, “Eyeing the patterns: Data visualization using doubly-seriated color heatmaps,” in *Advances in Computers*: Elsevier, 2020, S. 121–156.

[243] B. Kosch, B. A. Flubacher, D. Houdeau, M. Schmitt, O. Dressel, und P. Rost *et al.,* "Industrial Security und die Entwicklung von KI-Anwendungen in der Edge," Plattform Industrie 4.0, Bundesministerium für Wirtschaft und Energie (BMWi), 2021.

[244] ISO/IEC/IEEE 42010:2011-12, *Systems and software engineering - Architecture description*.

[245] R. M. French, "Pseudo-recurrent Connectionist Networks: An Approach to the 'Sensitivity-Stability' Dilemma," *Connection Science*, Bd. 9, Nr. 4, S. 353–380, 1997, DOI: 10.1080/095400997116595.

[246] J. Diemer, S. Elmer, M. Gaertler, T. Gamer, C. Görg, und J. Grotepass *et al.,* "KI in der Industrie 4.0:: Orientierung, Anwendungsbeispiele, Handlungsempfehlungen," Plattform Industrie 4.0, Bundesministerium für Wirtschaft und Energie (BMWi), 2020.

[247] T. Knodel, "Entwicklung eines Deep Industrial Transfer Learning Algorithmus zur Anomalieerkennung," Forschungsarbeit Nr. 3169, IAS, Universität Stuttgart, 2021.

[248] T. Knodel, "Weiterentwicklung eines Deep Industrial Transfer Learning Algorithmus zur Anomalieerkennung," Masterarbeit Nr. 3211, IAS, Universität Stuttgart, 2021.

[249] B. Maschler, T. Knodel, und M. Weyrich, "Towards Deep Industrial Transfer Learning for Anomaly Detection on Time Series Data," in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Västerås, Schweden, 2021, S. 1–8, DOI: 10.1109/ETFA45728.2021.9613542.

[250] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, und G. Chanan *et al.,* "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, 2019, Bd. 32.

[251] G. van Rossum und F. L. Drake, *Python 3 Reference Manual,* 3. Aufl. Scotts Valley, USA: CreateSpace, 2010.

[252] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, und T. Soderstrom, "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, Großbritannien, 2018, S. 387–395, DOI: 10.1145/3219819.3219845.

[253] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, Bd. 20, Nr. 1, S. 37–46, 1960.

[254] N. Merrill und A. Eskandarian, "Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning," *IEEE Access*, Bd. 8, S. 101824–101833, 2020, DOI: 10.1109/ACCESS.2020.2997327.

[255] V. Gurevich, *Electric Relays: Principles and applications*. Boca Raton, USA: CRC/Taylor & Francis, 2006.

[256] E. Vinaricky, Hrsg., *Elektrische Kontakte, Werkstoffe und Anwendungen*. Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2016.

[257] C. H. Leung und A. Lee, "Contact erosion in automotive DC relays," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, Bd. 14, Nr. 1, S. 101–108, 1991, DOI: 10.1109/33.76517.

[258] B. Maschler, A. Iliev, T. T. H. Pham, und M. Weyrich, "Stuttgart Open Relay Degradation Dataset (SOReDD)," Universität Stuttgart, Stuttgart, Deutschland, 2022.

[259] Y. Xuerong, M. Yue, M. Hang, und Z. Guofu, "Degradation failure model of electromagnetic relay," in *26th International Conference on Electrical Contacts (ICEC 2012)*, Peking, China, 2012, S. 116–123, DOI: 10.1049/cp.2012.0633.

[260] S. Davis, "Conception and Implementation of a Long-Term Experiment for Data Collection," Masterarbeit Nr. 3053, IAS, Universität Stuttgart, 2019.

[261] A. Michelberger, “Weiterentwicklung eines Langzeitversuchsaufbaus zur Datenerhebung,” Forschungsarbeit Nr. 3081, IAS, Universität Stuttgart, 2019.

[262] D. Kellner, “Entwicklung einer Web-basierten Steuerung und Visualisierung für einen Messstand,” Bachelorarbeit Nr. 3108, IAS, Universität Stuttgart, 2020.

[263] A. Illiev, “Entwicklung eines Deep Industrial Transfer Learning Algorithmus zur Vorhersage von Relais-Verschleiß,” Masterarbeit Nr. 3203, IAS, Universität Stuttgart, 2021.

[264] C. Brecher, “Bearing,” in *CIRP Encyclopedia of Production Engineering*, S. Chatti, L. Laperrière, G. Reinhart, und T. Tolio, Hrsg., Berlin, Heidelberg, Deutschland: Springer Berlin Heidelberg, 2019, S. 118–125.

[265] S. Schwendemann, Z. Amjad, und A. Sikora, “A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines,” *Computers in Industry*, Bd. 125, S. 103380, 2021, DOI: 10.1016/j.compind.2020.103380.

[266] S. Kumar, D. Goyal, R. K. Dang, S. S. Dhami, und B. S. Pabla, “Condition based maintenance of bearings and gears for fault detection – A review,” *Materials Today: Proceedings*, Bd. 5, Nr. 2, S. 6128–6137, 2018, DOI: 10.1016/j.matpr.2017.12.219.

[267] G.-Y. Lee, M. Kim, Y.-J. Quan, M.-S. Kim, T. J. Y. Kim, und H.-S. Yoon *et al.,* “Machine health management in smart factory: A review,” *Journal of Mechanical Science and Technology*, Bd. 32, Nr. 3, S. 987–1009, 2018, DOI: 10.1007/s12206-018-0201-1.

[268] Case Western Reserve University, *Bearing Data Center: Seeded Fault Test Data.* [Online]. Verfügbar unter: https://csegroups.case.edu/bearingdatacenter/home (zuletzt geprüft am 03.08.2021).

[269] E. Bechhoefer, *Condition Based Maintenance Fault Database for Testing of Diagnostic and Prognostics Algorithms.* [Online]. Verfügbar unter: https://www.mfpt.org/fault-data-sets/ (zuletzt geprüft am 03.08.2021).

[270] P. Nectoux, R. Gouriveau, K. Medjaher, E. Ramasso, B. Chebel-Morello, und N. Zerhouni *et al.,* “PRONOSTIA: An experimental platform for bearings accelerated degradation tests,” in *IEEE International Conference on Prognostics and Health Management (PHM '12)*, Denver, USA, 2012, S. 1–8.

[271] J. Lee, H. Qiu, G. Yu, J. Lin, and Rexnord Technical Services, *Bearing Data Set.* [Online]. Verfügbar unter: http://ti.arc.nasa.gov/project/prognostic-data-repository (zuletzt geprüft am 03.08.2021).

[272] L. Duan, D. Xu, und S.-F. Chang, “Exploiting web images for event recognition in consumer videos: A multiple source domain adaptation approach,” in *2012 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Providence, USA, 2012, S. 1338–1345, DOI: 10.1109/CVPR.2012.6247819.

[273] B. Maschler, D. Braun, N. Jazdi, und M. Weyrich, “Transfer learning as an enabler of the intelligent digital twin,” *Procedia CIRP*, Bd. 100, 127-132, 2021.

Anhang A Ansatz zur Bestimmung des Schwellenwerts für BIRCH-Clustering

Der in Kapitel 2.3.3 eingeführt SI reicht allein nicht aus, um eine Parametrierung des BIRCH-Clustering vorzunehmen. Für den Datensatz in Evaluierungsfall 1 ergibt sich der maximale SI bspw. für eine Cluster-Anzahl von zwei, obwohl ein händisches Clustering mindestens vier Cluster ergibt (siehe Abbildung A.1). Eine zu geringe Cluster-Anzahl kann trotz eines hohen SI den Transfer von Wissen erschweren, weil die Cluster nicht homogen genug sind.

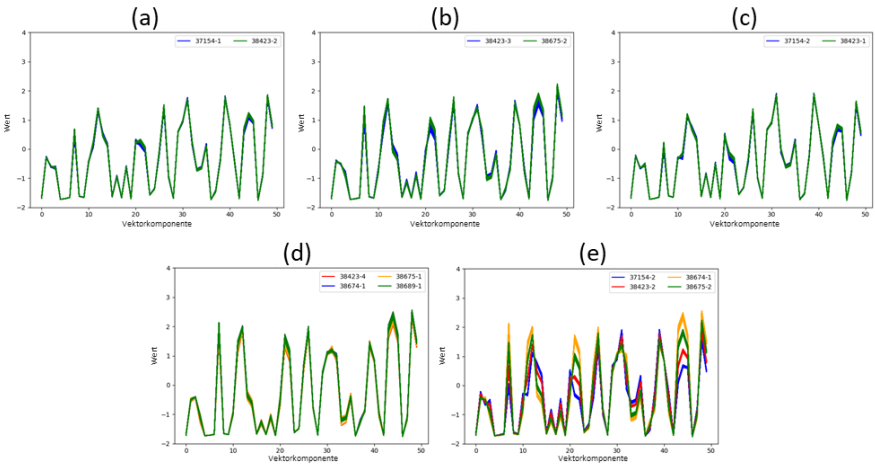

Abbildung A.1: Darstellung von 100 Merkmalsvektoren „Normal“ pro Fertigungsvariante aufgeteilt auf vier Cluster (a) bis (d) bzw. Gegenüberstellung der vier Cluster mit jeweils eine Fertigungsvariante pro Cluster (e)

In [112, 248] wird daher als zusätzliche Anforderung an das Clustering-Ergebnis eine manuell zu bestimmende Cluster-Mindestanzahl aufgenommen, um einen zweckdienlichen Parametersatz zu ermitteln. Diese wird anhand der Übereinstimmung der Clustering-Ergebnisse mit den vorliegenden (Teil-)Problemen bestimmt, wobei ein Teilproblem auf möglichst wenige Cluster aufgeteilt werden sollte, ohne dass die Cluster-Anzahl trivial klein wird.

Im Fall von Evaluierungsfall 1 in Kapitel 5.2.3.3 ist dies, dass der maximale SI für eine Cluster-Anzahl ≥ 4 einen geeigneten BIRCH-Schwellenwert T kennzeichnet. Wie in Abbildung A.2 dargestellt, muss der BIRCH-Schwellenwert T somit zwischen 0,48 und 0,67 liegen. Im Fall von Evaluierungsfall 2 in Kapitel 5.3.3.2 ist dies, dass der maximale SI für eine Cluster-Anzahl ≥ 7 einen geeigneten BIRCH-Schwellenwert T kennzeichnet, welcher zwischen 2,2 und 2,45 liegt.

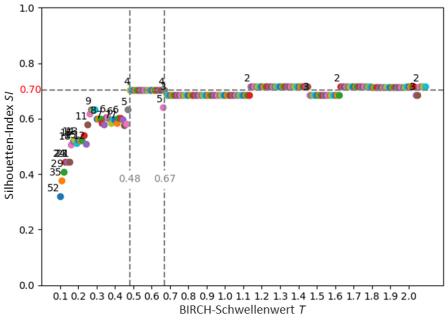

Abbildung A.2: Silhouetten-Index SI und Cluster-Anzahl in Abhängigkeit von unterschiedlichen BIRCH-Schwellenwerten T

Mit Hilfe der Anforderung einer Anwendungsfall abhängigen, minimalen Cluster-Anzahl zusätzlich zum SI ist nun eine zweckmäßige Bestimmung des BIRCH-Schwellenwerts T möglich.

Anhang B Ansatz zum Vergleich von Validierungsverlusten bei unterschiedlichen Eingangsdatenwertebereichen

Das Bewertungskriterium MSE, das in verschiedenen, im Rahmen dieser Arbeit entwickelten Modulen (siehe Kapitel 0, 5.2.2.3, 5.3.2.1, 0, 5.4.2.1, 5.4.2.3 und 0) für die Bewertung des Rekonstruktionsverlusts eingesetzt wird, ist in seinem Wertebereich abhängig vom Wertebereich der rekonstruierten Daten. Es ist definiert als

$$MSE = \frac{1}{N} \cdot \sum_{i=0}^{N} (x_i - y_i)^2$$

mit N als der Anzahl zu rekonstruierender Vektorelemente, x_i als dem i-ten Element des Eingangsvektors und y_i als dem i-ten Element des rekonstruierten Vektors, wobei $MSE \in [0;\ \infty]$ und ein niedrigerer Wert eine bessere Bewertung darstellt. Ein (sehr kleiner) konstanter Fehler von 0,1 % ergibt bei einem zu rekonstruierenden Wert von 1 einen MSE von 0,000001 – bei einem mittleren, zu rekonstruierenden Wert von 10.000 jedoch einen MSE von 100. Ein direkter Vergleich mittels MSE ist daher bei unterschiedlichen Größenordnungen der Eingangsdaten nur schwer möglich. Die in Kapitel 5.4.3.2 evaluierten Merkmalsextraktionen verwenden derartig unterschiedliche Eingangsvektoren.

In [171] wird zu diesem Zweck eine neue Größe, der sogenannte Bewertungsfaktor, eingeführt. Dieser basiert auf dem Vergleich des MSE mit einem fiktiven Bezugs-Verlust. Dieser Bezugsverlust wird als MSE bei einem relativen Fehler von 10 % definiert. Der Bewertungsfaktor ist somit definiert als

$$\text{Bewertungsfaktor} = \sum_{i=0}^{N} \frac{(x_i - y_i)^2}{(0{,}1\ x_i)^2}\ .$$

Mit Hilfe des Bewertungsfaktors ist nun ein direkter Vergleich der Algorithmus-Genauigkeit bei Eingangsdaten unterschiedlicher Größenordnungen möglich.