

# **Transfer Meta Learning**

## **Herausforderungen der Mustererkennung**

### **Dissertation**

Dissertation zur Erlangung des Grades eines  
Doktor-Ingenieurs (bzw. Ph.D.)  
der Fakultät für Elektrotechnik und Informationstechnik  
an der Ruhr-Universität Bochum

**Nico Zengeler**  
aus Luckenwalde

Erscheinungsjahr: 2022

Namen der Berichter/innen:  
Prof. Dr. Tobias Glasmachers  
Prof. Dr. Asja Fischer  
Prof. Dr. Uwe Handmann

Tag der mündlichen Prüfung: 30.08.2022

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Forschungsfrage . . . . .	4
1.2 Aufbau . . . . .	5
<b>2 Grundlagen</b>	<b>6</b>
2.1 Evolutionäre Algorithmen . . . . .	7
2.1.1 Genetische Algorithmen . . . . .	8
2.1.2 Genetische Programmierung . . . . .	9
2.1.3 Evolutionsstrategien . . . . .	9
2.2 Tiefe künstliche neuronale Netzwerke . . . . .	11
2.2.1 Aktivierungsfunktion . . . . .	12
2.2.2 Fehlerrückführung . . . . .	14
2.2.3 Mehrschichtiges Perzepton . . . . .	15
2.2.4 Faltendes neuronales Netzwerk . . . . .	17
2.2.5 Long Short-Term Memory . . . . .	19
2.3 Transferlernen . . . . .	21
2.3.1 Verwandte Verfahren . . . . .	22
2.3.2 Anwendungen . . . . .	22
2.3.3 Feinabstimmung . . . . .	23
2.3.4 Elastische Gewichtskonsolidierung . . . . .	24
2.3.5 Progressive neuronale Netzwerke . . . . .	26
2.3.6 Gegnerische Umprogrammierung . . . . .	28
2.3.7 Weltmodelltransfer . . . . .	30
2.4 Verstärkungslernen . . . . .	32
2.4.1 Markov'scher Entscheidungsprozess . . . . .	32
2.4.2 Q-Learning . . . . .	34
2.4.3 Asynchrone Verfahren . . . . .	37
<b>3 Herausforderungen</b>	<b>38</b>
3.1 Automatische Handelssysteme . . . . .	39
3.1.1 Differenzkontrakte . . . . .	40
3.1.2 Aufbau der Simulation . . . . .	40
3.1.3 Künstliche neuronale Architekturen . . . . .	42
3.1.4 Methode . . . . .	45
3.1.5 Auswertung . . . . .	49
3.1.6 Diskussion . . . . .	55

3.2	Handgestenerkennung . . . . .	57
3.2.1	Datensätze . . . . .	60
3.2.2	Modelle . . . . .	61
3.2.3	Diskussion . . . . .	62
3.3	Personenerkennung . . . . .	63
3.3.1	Datensatz . . . . .	66
3.3.2	Modelle . . . . .	71
3.3.3	Auswertung . . . . .	73
3.3.4	Diskussion . . . . .	76
3.4	Personenverfolgung . . . . .	76
3.4.1	Versuchsaufbau . . . . .	80
3.4.2	Fußpunktterkennung in Weltkoordinaten . . . . .	80
3.4.3	Sensorfusion . . . . .	82
3.4.4	Spur trennung . . . . .	83
3.4.5	Auswertung . . . . .	83
3.4.6	Diskussion . . . . .	87
3.5	Bildklassifikationstransfer . . . . .	87
3.5.1	Datensätze und Modelle . . . . .	88
3.5.2	Auswertung . . . . .	93
3.5.3	Diskussion . . . . .	100
<b>4</b>	<b>Transfer Meta Learning</b>	<b>101</b>
4.1	Motivation . . . . .	103
4.2	Konzept . . . . .	104
4.2.1	Modellierung nach EVA-Prinzip . . . . .	105
4.2.2	Objektorientierte Umsetzung . . . . .	106
4.3	Methode . . . . .	112
4.3.1	Aufgaben und Modelle . . . . .	113
4.3.2	Rastersuche . . . . .	114
4.3.3	Metadatensatz . . . . .	114
4.3.4	Metaleernen . . . . .	114
4.4	Auswertung . . . . .	116
4.4.1	Versuchsaufbau . . . . .	116
4.4.2	Ergebnisse . . . . .	117
4.4.3	Vergleich . . . . .	119
<b>5</b>	<b>Diskussion</b>	<b>125</b>
5.1	Limitationen . . . . .	126
5.2	Ausblick . . . . .	127
5.3	Ethische Betrachtungen . . . . .	128
<b>6</b>	<b>Zusammenfassung</b>	<b>131</b>

# 1 Einleitung

Die Informatik gilt als Wissenschaft, deren Untersuchungsgegenstand in der Struktur von Information und deren automatisierter Verarbeitung besteht. Die frühe Informatik entstammt mathematischen Axiomen, insbesondere solchen der Zahlentheorie und Mengenlehre, welche sich im Grunde mit der Beweisbarkeit von Aussagen befassen [1]. Auf diesen Axiomen aufbauend lässt sich die Erkenntnis gewinnen, dass nicht alle Fragen, die sich in mathematischen Systemen ausdrücken lassen, formell entscheidbar sind [2]; nach dem Gödel'schen Unvollständigkeitssatz gibt es in solchen Systemen immer Aussagen, die sich weder beweisen noch widerlegen lassen. Die Frage nach der Entscheidbarkeit einer derart axiomatisierten Fragestellung lässt sich auf die Frage nach der Berechenbarkeit der Ergebniszahlen in Polynomialzeit mit einer deterministischen Turingmaschine zurückführen, wie etwa einem Computer, zurückführen [3].

Die heutige Informatik unterteilt sich in verschiedene Disziplinen; namentlich die theoretische, die praktische, die technische und die angewandte Informatik. Während sich die theoretische Informatik grundlegenden Fragen der Berechenbarkeit, formalen Sprachen und Automatentheorie stellt, beschäftigt sich die angewandte Informatik mit der Anwendbarkeit der informatischen Methoden zur Lösung von Fragen aus anderen Fachbereichen, wie etwa Fragestellungen bezüglich großer Datenmengen aus der Medizin, aus den Ingenieursdisziplinen, aus den Wirtschaftswissenschaften und aus der Psychologie. Hierbei ist Information, je nach Quelle, von unterschiedlicher Gestalt; Sensordaten, wie etwa Kamerabilder, Audiosignale oder andere physikalische Messgrößen, beschreiben ebenso einen kommunikationstheoretischen Wissensgewinn wie beispielsweise Baupläne, Handelsdaten oder Nachrichtentexte. Auf Grundlage von Information können Automatismen ausgeführt werden, welche zum Beispiel in der Regelungstechnik, der Nachrichtentechnik, der Mensch-Maschine Interaktion oder der Navigation Anwendung finden. Die Verbindung von Eingangsdaten, wie etwa verrauschter Sensorinformation, mit konkreter Handlung, beispielsweise der Regelung oder Steuerung eines physikalischen Systems, eröffnet die Suche nach einer intelligenten automatischen Verarbeitung. Dem Menschen gelingt es von Natur aus, Unschärfe in sich präsentierenden Qualia zu erfassen, in diesen Zusammenhänge zu erkennen und Unregelmäßigkeiten zu ergründen und einzuordnen. Die Suche nach einem reproduzierbaren Ablauf, welcher diese Fähigkeit emuliert, eröffnet das Forschungsfeld der künstlichen Intelligenz (Englisch: *Artificial Intelligence, AI*).

Künstliche Intelligenz ist eine Abbildung natürlicher Intelligenz. Biologische Vorgänge natürlicher Intelligenz, vornehmlich die Verhaltensweisen von Nervenzellen, werden mathematisch abstrahiert und somit für maschinelle Lernverfahren programmatisch anwendbar. Die Möglichkeit, intelligentes Verhalten auf diese Weise maschinell berechenbar zu gestalten, eröffnet die Suche nach algorithmischen Lernverfahren und die Frage nach der Unterscheidung derer Ergebnisse von natürlicher, etwa menschlicher, Intelligenz [4].

In seinem technischen Report zu intelligenten Maschinen beschreibt Turing diese Suche „*unter diesem Gesichtspunkt [...] als eine Aufgabe der menschlichen Gemeinschaft als Ganzes und nicht als eine Aufgabe des Einzelnen*“ [4].

Künstlich intelligente Programme, welche maschinelle Lernverfahren nutzen, dienen zum automatisierten Erkennen von Mustern in großen Datenmengen; dies gilt als die Kernkompetenz aller künstlich intelligenten Verfahren [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. Auch andere Anwendungsfälle sind denkbar. Im industriellen Kontext finden maschinelle Lernverfahren häufig Anwendung bei der Automatisierung arbeitsaufwändiger Tätigkeiten und werden als ein Werkzeug eingesetzt, solche Arbeiten sicherer, effizienter und effektiver zu gestalten [15, 16, 17]. Ebenso können diese Verfahren genutzt werden, um Muster zu vervollständigen oder fehlende Information zu ergänzen, was interessante Anwendungen in künstlerischen Bereichen ermöglicht. Künstliche Intelligenzen bieten dem Menschen eine Vielzahl von Werkzeugen zur Vereinfachung, Weiterentwicklung und Automatisierung der Arbeit und des sozialen Netzwerks. In den verschiedenen Disziplinen der Wissenschaft wird das maschinelle Lernen genutzt, um aus großen Datenmengen (E: *Big Data*) informative Statistiken und Schlussfolgerungen zu gewinnen sowie Vorhersagen über künftige Entwicklungen zu treffen. Die Industrie erkennt die Bedeutung derart intelligenter Programme, beispielsweise für die datengetriebene automatisierte Steuerung einer Maschine zu Zwecken der Effizienzsteigerung. Für den Handel besteht eine zunehmend an Bedeutung gewinnende Anwendung im automatischen Handeln von Energie- und Vermögenswerten, da intelligente Maschinen Entwicklungen schneller als der Mensch antizipieren und bei Bedarf in Sekundenbruchteilen automatisch entsprechende Handelsaufträge erteilen können. Die Ingenieursdisziplinen nutzen künstlich intelligente Methoden zur Lösung von steuerungstechnischen Fragestellungen, beispielsweise finden diese Ansätze Anwendung in der sogenannten Industrie 4.0, der Hausautomatisierung oder dem als „Internet der Dinge“ bezeichneten Netzwerk kommunizierender Geräte. Nach aktueller Vorgehensweise werden für einzelne Optimierungsprobleme jeweils eigene maschinelle Lernverfahren (E: *Machine Learning*, ML) eingesetzt, welche entsprechend der vorliegenden Datenlage optimale Parameter für ein Modell ermitteln, welches Eingaben und erwartete Ausgaben aufeinander abbilden kann. Auf diese Weise werden Einzellösungen für konkrete Anwendungen erzeugt, im derzeitigen Stand der Technik in der Regel mit dem Training tiefer künstlicher neuronaler Netzwerke (E: *Deep Learning*, DL). Mit dem Aufkommen neuer und immer größerer Datensätze entstehen beispielsweise Anwendungen in der Medizin, der Mensch-Maschineinteraktion, dem autonomen Fahren, der Arbeitssicherheit, der Kreislaufwirtschaft (E: *Circular Economy*, CE) sowie automatisierten Handels- und Vertragsoptimierungen. Diese technischen Fortschritte können Erfolge hinsichtlich der Verbesserung der Lebens- und Umweltbedingungen darstellen. Grundlage dieser Erfolge sind spezialisierte Expertensysteme, die mit Methoden des maschinellen Lernens eine Expertise in einzelnen Aufgabengebieten erlangen. Heutige Expertensysteme lernen in der Regel mit vorgefertigtem Wissen, so dass sie Vorhersagen und Entscheidungen auf der Grundlage von vorgegebenen Beispielmustern und implizierten Fakten und Regeln treffen. Solche Programme sind in der Regel schwer an eine sich ändernde Sachlage anzupassen, da ihre Lernfähigkeit durch ihre Wissensbasis, beziehungsweise durch die Trainingsdaten, eingeschränkt ist.

Diese Programme lernen also nicht wie Lebewesen, aber genau auf diese Art des Lernens konzentriert sich die aktuelle interdisziplinäre Forschung im Bereich der Neuroinformatik und des maschinellen Lernens. Einzelne Lösungen des maschinellen Lernens, beispielsweise die gelernten Parametervektoren künstlicher neuronaler Netzwerke, werden als schwache künstliche Intelligenzen bezeichnet. Eine starke künstliche Intelligenz hingegen zeichnet sich durch perpetuelle Verallgemeinerungs- und Konkretisierungsfähigkeit aus, also der Fähigkeit, beliebig Wissen zu akkumulieren und für neue Aufgaben zu nutzen.

Hierbei stellt sich die grundsätzliche Frage nach dem Wesen von Erkenntnis und dessen Übertragbarkeit in Form von Wissen. Im Rahmen des Deep Learning führt diese Suche zu den Forschungsfragen des Transferlernens und des Metalernens. Das Transferlernen umfasst eine besondere Klasse von Lernverfahren, welche bereits vorhandenes Wissen aus vorherigen Aufgaben wiederverwenden, um das Lernen neuer Aufgaben zu beschleunigen, stabiler zu gestalten und besser zu generalisieren. Im Metalernen werden keine Nutzdaten der Aufgaben verwendet, sondern lediglich deren Metadaten, um in diesen Merkmalen zu finden, welche der Übertragung von Wissen dienlich sind. Der in dieser Dissertation vorgestellte neue Ansatz des *Transfer Meta Learning* nutzt Metalernverfahren, um optimale Transferlernprozesse abzuschätzen [5, 6].

Mit der wachsenden Popularität des maschinellen Lernens, beispielsweise des Lernens tiefer künstlicher neuronaler Netzwerke, entstehen immer dann neue Anwendungen, wenn zu Fragen neue Daten verfügbar und optimale Lösungen gesucht werden. Die Optimierung tiefer künstlicher neuronaler Netzwerke kann, je nach Komplexität der Aufgabenstellung, jedoch sehr rechenintensiv sein, wenn von Grund auf gelernt wird; so benötigte das Training eines GPT-3 Sprachmodells im Jahr 2020 etwa 1 287 000 kWh Strom, was nach durchschnittlichem Emissionsfaktor einer Gesamtemission von etwa 552 100 kg Kohlenstoffdioxid entspricht [18] — umgerechnet also etwas über drei Millionen gefahrenen Kilometern mit einem modernen Personenkarfswagen. Die Forschung in den Feldern des Transfer- und Metalernens verspricht Reduktionen dieser Energiekosten, indem das von verschiedenen Modellen in unterschiedlichen Formen repräsentierte Wissen miteinander in Einklang gebracht wird, um mit weniger Rechenaufwand tiefgründigere Zusammenhänge abzubilden und größere, bisweilen ungelöste, maschinelle Lernprobleme zu lösen [19, 20, 21].

Die verschiedenen beteiligten wissenschaftlichen Disziplinen untersuchen die dem selbstständigen Lernen zugrundeliegenden Prinzipien aus unterschiedlichen Blickwinkeln, wodurch interdisziplinäre Antworten zunehmend an Bedeutung gewinnen. Die Erforschung der Grundlagen selbstlernender Systeme eröffnet eine Vielzahl von Fragen an verschiedene Disziplinen außerhalb der Informatik und Mathematik, unter anderem an die Psychologie [22]. Jede Lösung schafft Wissen, welches für folgende Lösungen wiederverwendet werden kann. Bei einer einfachen Addition von Wissen aus verschiedenen Aufgabendomänen zu einem einzigen künstlichen neuronalen Modell zeigt sich das Problem des katastrophalen Vergessens; ohne weitere Randbedingungen führt ein unregulierte, kontinuierliches Lernen aller anfallenden Daten in der Regel zu einer Verschlechterung bis hin zur Funktionsunfähigkeit des künstlichen neuronalen Netzwerks [23, 24, 25]. Methoden des Transferlernens bieten verschiedene Ansätze, das Problem des katastrophalen Vergessens im Rahmen schwacher künstlicher Intelligenz zu überwinden.

## 1.1 Forschungsfrage

Transferlernen ist eine Technik, welche vornehmlich für das Training tiefer künstlicher neuronaler Netzwerke eingesetzt wird, um bereits vorhandenes Wissen aus bekannten Aufgaben in unbekannte Aufgaben zu übertragen. Diese Technik erleichtert das Lernen einer neuen Aufgabe, indem beispielsweise weniger Trainingsdaten benötigt werden oder weniger Lernschritte durchgeführt werden müssen, um zum gleichen Ergebnis zu gelangen. Mit Methoden des Transferlernens wird in zahlreichen Anwendungen untersucht, inwiefern der Rechenbedarf für künstliche Intelligenzen mit hoher Verallgemeinerungsfähigkeit reduziert werden kann, indem bereits erworbenes Wissen wiederverwendet wird, um Lernprobleme zu erleichtern und somit die zugrundeliegenden Rechenprozesse zu beschleunigen [19, 20, 21, 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. Unter dem Paradigma des Transferlernens werden Ansätze verglichen, welche es ermöglichen, gelerntes Wissen aus einem Aufgabenbereich in einen anderen zu überführen [44, 45]. Im großen Bild wird versucht ein System zu entwickeln, welches in beliebigen Datenlagen, also beispielsweise auch in virtuellen Umgebungen, lernen und das Gelernte auf andere, etwa reale, Umgebungen übertragen kann [46].

Bei der Durchführung eines Transferlernprozesses stellt sich die Frage nach der Modellauswahl und der Auswahl optimaler Hyperparameter, wie etwa die Anzahl der Lernschritte oder die Lernrate; an dieser Stelle ist Expertenwissen erforderlich, um einen effizienten Transferlernprozess zu finden [6]. Dieses Problem der Modellauswahl und der Auswahl optimaler Hyperparameter wird systematisch mit Metalernen angegangen, welches Zusammenhänge in den Metadaten der Transferlernprozesses erkennt [5]. Die Kernmethodik des in dieser Dissertation erforschten Ansatzes besteht also darin, aus den Randbedingungen und Ergebnissen von Transferlernprozessen auf der Metaebene zu lernen. Der Ansatz zum *Transfer Meta Learning* gründet sich hierbei in der Frage: Wurde beim Lösen neuer Aufgaben etwas gefunden, was zur Lösung weiterer neuer Aufgaben hilfreich ist? Die Beantwortung dieser Frage bei jeder neuen Aufgabe führt zur Bildung einer breiten Wissensbasis für Transferlernmethoden, welche in Form von Metamodellen abgebildet wird. Dieser Ansatz ist motiviert durch die Nutzung von Weltmodellwissen für spezifische Aufgaben, verallgemeinert das zugrundeliegende Prinzip jedoch auf programmatischer Ebene für Wissenstransferanwendungen.

Zur Konzeptionalisierung dieses Ansatzes werden Begrifflichkeiten von Aufgaben und deren Lösungen benötigt, welche sich in Form der jeweiligen Datensätze und deren zugehörigen Optimierungsverfahren darstellen. Zur Schaffung einer Diskussionsgrundlage bezüglich der Forschungsfrage werden verschiedene Methoden des maschinellen Lernens hinsichtlich ihrer Eignung zur Muster- und Objekterkennung in verschiedenen Aufgabenbereichen untersucht [6, 7, 8, 9, 10, 11]. Basierend auf theoretischen Grundlagen und vorangegangenen praktischen Arbeiten werden verschiedene Aufgaben und Lösungsmöglichkeiten untersucht, systematisch ausgewertet und miteinander verglichen, um die Relevanz der Wiederverwendung von maschinell gewonnenem Wissen zu verdeutlichen.

## 1.2 Aufbau

Zunächst wird in Kapitel 2 der bereits erforschte Stand der Wissenschaft und Technik dargelegt, der im Rahmen der Anwendungen dieser Forschungsarbeit als Grundlage relevant ist. Hierzu werden die mathematischen Konzepte, welche den verschiedenen Lernverfahren tiefer künstlicher neuronaler Netzwerke zugrundeliegen, erläutert, aktuelle Ansätze des Transferlernens beleuchtet sowie die Methode des bestärkenden Lernens erklärt, welche relevant zum Verständnis einiger diskutierter Herausforderungen ist. Zunächst wird in diesem Kapitel jedoch kurz die Funktionsweise evolutionärer Algorithmen beschrieben, da diese zum Verständnis einiger weiterführender Transferlernmethoden relevant ist.

Auf den Grundlagen aufbauend, werden in Kapitel 3 die Herausforderungen diskutiert, bezüglich welcher, im Rahmen dieser Dissertation, entsprechende Anwendungen entwickelt wurden. Die Anwendungen werden diskutiert, um die Relevanz der Forschungsfrage zu verdeutlichen; in den Anwendungen wurden hochspezialisierte Lösungen entwickelt und das Ziel ist es, das hierbei entstandene Wissen weitergehend zu nutzen. Hierbei handelt es sich um Untersuchungen zu automatisierten Handelssystemen [7, 8], zur Handgestenerkennung [9], zur Personenerkennung auf Kamerabildern [10], zur Verfolgung von Personen auf Kamerabildern [11] und zum Transferlernen am Beispiel der Bildklassifikation [6]. In den Anwendungen zeigt sich der Nutzen des Wiederverwendens von gelernten Wissensrepräsentationen und Transferlernprozessen zur Reduktion der Rechenkosten sowie der Bedarf nach einer systematischen Analyse der hierbei anfallenden Metadaten. Jede beschriebene Herausforderung enthält in etwa das Muster der zugrundeliegenden Veröffentlichung, sodass der Kontext in der jeweiligen Sektion erhalten bleibt.

Kapitel 4 beschreibt das in dieser Dissertation eingeführte, neue Konzept des *Transfer Meta Learning* nach den Standardparadigmen der Informatik, etwa dem Eingabe-Verarbeitung-Ausgabe (**EVA**) Prinzip und der objektorientierten Modellierung. Die Auswertungen dieses Kapitels befassen sich mit dem Beispiel der visuellen Objektargumentation im Allgemeinen, beziehungsweise der Objekterkennung im Besonderen. Hierbei zeigt sich der praktische Nutzen der systematischen Verwendung der Metadaten von Transferlernprozessen zur Findung optimaler Modelle und Hyperparameter für eine maschinelle Wissensübertragung.

In der Diskussion in Kapitel 5 werden die informationstheoretischen Limitationen des Ansatzes, die praktischen Begrenzungen und zukünftige Arbeiten diskutiert und eine ethische Betrachtung vorgenommen. Diejenigen Themen und Ideen, welche in naher Zukunft interessante Anwendungsmöglichkeiten, Verbesserungen und Erweiterungen darstellen, werden hierbei als zukünftige Arbeiten elaboriert. An dieser Stelle werden andere akademische Abschlussarbeiten genannt, welche in Beziehung zu dieser Arbeit entstanden. Anschließend werden Chancen und Risiken der künstlichen Intelligenz im Allgemeinen und des *Transfer Meta Learning* im Besonderen diskutiert. Zum Abschluss fasst das Kapitel 6 die wichtigsten Aussagen dieser Dissertation kurz zusammen.

## 2 Grundlagen

Zur datengetriebenen Lösung von Optimierungsproblemen fokussiert sich der Stand der Wissenschaft und Technik auf maschinelle Lernverfahren mit dem Training tiefer künstlicher neuronaler Netzwerke [47]. Verschiedene Arten von Lernverfahren erfüllen unterschiedliche Anforderungen. In Methoden des überwachten Lernens (E: *Supervised Learning*, SL) werden die Eingabe- und Zielgrößen vorgegeben und eine Abbildung gesucht, welche die Vorhersagegenauigkeit optimiert [48]. Falls keine Zielwerte bekannt sind, können mit Hilfe von Ansätzen des unüberwachten Lernens (E: *Unsupervised Learning*, UL) strukturelle Abhängigkeiten in den Daten modelliert werden, beispielsweise über die Ähnlichkeiten von Nachbarschaften oder über die Minimierung eines Rekonstruktionsfehlers [49, 50]. Überwachte Lernmethoden eignen sich beispielsweise für das Training von Klassifikatoren und Detektoren [51, 52, 53, 54, 55, 56, 57, 58], unüberwachte Methoden hingegen eignen sich zur Dimensionsreduktion und zum Generieren neuer aber ähnlicher Daten, beispielsweise mit Autoencodern [59, 60, 61, 62]. Die Methoden des Verstärkungslernens (E: *Reinforcement Learning*, RL) erlauben eine Problemformulierung über die Interaktion eines Agenten mit einer Umgebung in Form eines markov'schen Entscheidungsprozesses [63]. In einigen Fällen kann sich die Anwendung eines evolutionären Algorithmus (E: *Evolutionary Algorithm*, EA) zur Optimierung eines Parametervektors als zielführend erweisen [19, 64, 65]; der evolutionäre Ansatz eignet sich besonders für vergleichsweise kleine Parametervektoren und für eine Methode des Transferlernens, welche Wissen aus einem großen Weltmodell in spezialisierte Modelle überträgt [19].

Mit dem Konzept des Deep Learning lassen sich alle diese verschiedene Lernverfahren maschinell durchführen, indem als Optimierungsmodell ein tiefes künstliches neuronales Netzwerk gewählt wird, dessen synaptische Gewichte entsprechend eines Optimierungskriteriums verändert werden [47]. Das neuronalen Lernverfahren des Deep Learning findet beispielsweise Anwendung in der Kontrolle des Plasmas von Kernfusionsreaktoren [66, 67], der Kreislaufwirtschaft [68, 69] oder der medizinischen Diagnostik [70, 71], beispielsweise bei der Früherkennung und Prognose der Alzheimerkrankheit [72, 73]. Künstlich intelligente Programme nutzen diese Methoden beispielsweise auch für die Gesichts- und Spracherkennung [74, 75], für die Navigation [16], für die Prognose von Marktentwicklungen [76], für eine automatisierte Recyclingunterstützung [26] und für die Verarbeitung natürlicher Sprache [77, 38]. Für die Modellierung des Klimas werden ebenfalls tiefe künstliche neuronale Netzwerke eingesetzt [78, 79, 80, 81]. Weitere höchst interessante Anwendungen finden sich in der neuronalen Kryptographie, da ein neuronaler Schlüsselaustausch als Kandidat für eine quantensichere Kommunikation gilt [82, 83, 84]. Werden neuronale Architekturen um ein Speicherwerk erweitert, sind diese neuronalen Turingmaschinen oder differenzierbaren neuronalen Rechner (E: *Differentiable Neural Computer*, DNC) in der Lage, einen Algorithmus zu lernen [85, 86].

## 2.1 Evolutionäre Algorithmen

Evolutionäre Algorithmen basieren auf dem Grundgedanken der biologischen Evolution und eignen sich insbesondere zur Lösung von Optimierungsproblemen, bei denen keine zielführende Ableitung des Optimierungskriteriums nach den lernbaren Parametern aufgestellt werden kann [87]. Mit evolutionären Algorithmen wird versucht, die in der biologischen Evolution beobachteten Abläufe maschinell nachzubilden und für Optimierungsprobleme in algorithmischer Form nutzbar zu machen [87]. Der Grundgedanke folgt der darwin'schen Evolution der Arten [88]. Ein evolutionärer Algorithmus findet Lösungen in Form einer Population von Individuen, welche jeweils verschiedene mögliche Lösungen eines Optimierungsproblems darstellen. Die Maximierung der Lösungsqualität, hier als Fitnesswert bezeichnet, geschieht über die iterative Anwendung von Prozeduren der Selektion, Rekombination und Mutation bis zum Erreichen eines Abbruchkriteriums. Nach einer hinreichenden Anzahl von Iterationen entwickeln sich Individuen, welche über einen hohen Fitnesswert verfügen [89].

Der Fitnesswert entspricht dem Maß für die Qualität der gefundenen Lösung bezüglich des Optimierungsproblems. Die Population der Lösungen liegt zunächst in einer Menge von Genotypen  $X$  vor und wird über eine Zuweisungsfunktion  $\phi$  in entsprechend auswertbare Phänotypenmenge  $Y$  überführt:

$$\phi : X \rightarrow Y \quad (2.1)$$

Diesen Phänotypen wird nun anhand einer Fitnessfunktion  $f$  ein reeller Fitnesswert zugewiesen, welchen der evolutionäre Algorithmus maximieren soll:

$$f : Y \rightarrow \mathbb{R} \quad (2.2)$$

Diese grundlegende Idee kann auf unterschiedliche Weisen auf konkrete Herausforderungen angewandt werden. Die verschiedenen evolutionären Algorithmen sind in der Lage, verschiedenartige Blackboxprobleme zu lösen. Genetische Algorithmen sind eine einfache Form der evolutionären Optimierung und treffen keine weiteren Annahmen über die Natur des Optimierungsproblems. In der genetischen Programmierung hingegen wird die Annahme getroffen, dass sich die Lösungen des Optimierungsproblems als Wort oder Satz einer Sprache ausdrücken lassen, beispielsweise als Information in Form einer Zeichenkette, einer Formel oder eines Quelltextes. Evolutionsstrategien erweitern den Grundgedanken dahingehend, dass die Genotypen als Vektoren reeller Zahlen repräsentiert werden, die nach einer Wahrscheinlichkeitsverteilung mutieren, welche während des Evolutionsprozesses strategisch vorteilhaft angepasst wird.

Im Kontext des Trainings tiefer künstlicher neuronaler Netzwerke im Allgemeinen und des Transferlernens im Besonderen finden evolutionäre Algorithmen Anwendung in der Optimierung von Hyperparametern [90], in der gegnerischen Umprogrammierung [91, 92] oder bei der Inferenz eines Weltmodells durch ein aufgabenspezifisches künstliches neuronales Netzwerk mit wenigen synaptischen Gewichten [19].

### 2.1.1 Genetische Algorithmen

Genetische Algorithmen sind eine Klasse von Algorithmen, welche unter Einsatz von Rekombinations- sowie Mutations- und Selektionsoperatoren mögliche Lösungen eines Optimierungsproblems suchen und in Form genotypischer Datenstrukturen speichern [93, 94]. Nach der Initialisierung der Population beginnt eine Schleife aus Selektion, Rekombination und Mutation. Bei Anwendung einer Evolutionsstrategie können interne Parameter des Algorithmus, wie beispielsweise die Mutationswahrscheinlichkeitsverteilung, angepasst werden. In jeder Iteration des genetischen Algorithmus ergibt sich eine verbesserte Kindpopulation aus einer Elternpopulation. Sobald das Abbruchkriterium erreicht ist, endet der Algorithmus. Die Population stellt dann eine Menge von optimalen Lösungen dar, das Individuum mit dem höchsten Fitnesswert die beste gefundene Lösung. Die Skizze in Abbildung 2.1 illustriert die iterative Abfolge der einzelnen Schleifenschritte. Die Mutations- und Rekombinationsfunktionen eines genetischen Algorithmus hängen von der konkreten Genrepräsentation ab. Beispielsweise kann ein binärcodiertes Gen durch zufälliges Invertieren oder Vertauschen einzelner Bits mutiert werden. Die Gene von Individuen, welche als Vektoren dargestellt werden, können beispielsweise durch die Bildung eines Mittelwertvektors rekombiniert werden. Die angewandten Selektionsmechanismen sollen gute Lösungen mit höherer Wahrscheinlichkeit erhalten. Es existieren verschiedene Selektionsmechanismen, deren konkrete Ausgestaltung von der Genrepräsentation und der Fitnessfunktion abhängt. Ein elitärer Selektionsansatz sieht vor, die besten Lösungen aus der Elterngeneration beizubehalten. Eine besondere Form der elitären Selektion ist die Plusselektion. In einer  $(\mu + \lambda)$ -Plusselektion werden die besten  $\mu$  Individuen aus einer Population der Größe  $(\mu + \lambda)$  ausgewählt. Wird eine Kindgeneration  $\lambda$  der Größe  $\lambda > \mu$  erzeugt, kann eine  $(\mu, \lambda)$ -Kommaselektion angewandt werden. In dieser Selektionsform bleibt kein Individuum der Elterngeneration erhalten. Ein anderer Ansatz, die Auswahl durch ein Turnier (E: *Tournament Selection*), vergleicht eine zufällige Auswahl von Individuen der Elterngeneration und erzeugt eine Kindgeneration aus denjenigen Individuen, welche in den direkten Vergleichen die höheren Fitnesswerte aufweisen.

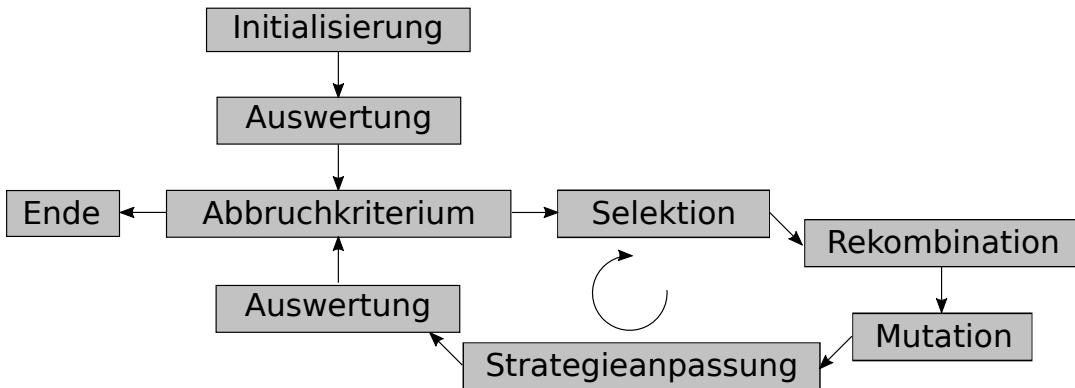


Abbildung 2.1: Eine Programmablaufskizze genetischer Algorithmen.

## 2.1.2 Genetische Programmierung

Die genetische Programmierung ist eine evolutionäre Methode zur automatisierten Entwicklung eines Programms (E: *Automatic Programming*) [95, 96, 97]. Die Lösungen stellen Elemente einer Sprache dar, welche auch turingvollständig sein und somit einen Algorithmus beschreiben können [97]. Die sprachlich abzubildenden Programmanweisungen können in Form einer Baumstruktur beschrieben werden. Ausgehend von einem Wurzelknoten werden Worte der Sprache miteinander verbunden. Konkrete Werte und Variablen befinden sich in den Endknoten des zirkelfreien Graphen und sind durch operative Knoten miteinander verbunden. Ausgehend vom Wurzelknoten führt eine Rekursion über die Baumstruktur zu einer auswertbaren Ergebniszeichenkette. Der Fitnesswert des Individuums ergibt sich durch die Auswertung der beschriebenen Lösung in verschiedenen Testfällen mit jeweils eigenen Eingabe- und Zielgrößen.

Ähnlich wie bei einem genetischen Algorithmus wird das Programm in Form eines Genes repräsentiert. Aufgrund der Baumstruktur eröffnen sich in der genetischen Programmierung jedoch weitere Rekombinations- und Mutationsmöglichkeiten. Eine Baumstruktur kann beispielsweise mutiert werden, indem zufällig ausgewählte Knoten durch neue, zufällig generierte Unterbaumstrukturen ersetzt werden oder indem zufällige Operatoren, Werte und Variablen geändert werden. Die Rekombination zweier Baumstrukturen kann erfolgen, indem zufällig gewählte Unterbaumstrukturen der gewählten Eltern miteinander vertauscht werden [96].

## 2.1.3 Evolutionsstrategien

Evolutionsstrategien sind eine, auf kontinuierliche Variablen spezialisierte, evolutionäre Optimierungsmethode [98, 99]. Evolutionäre Strategien nehmen genotypische Lösungen als Vektoren  $x \in \mathbb{R}^n$  angenommen, deren Mutationen einer Wahrscheinlichkeitsverteilung folgen. Die Anpassung der Mutationswahrscheinlichkeitsverteilung entspricht der evolutionären Strategie. Wird die Mutationswahrscheinlichkeitsverteilung beispielsweise als eine Gaußverteilung modelliert, deren Mittelwert das Individuum  $x$  bildet, welches mit der Standardabweichung  $\sigma$  entlang beliebiger Optimierungsdimensionen mutieren kann, so kann über die iterative Anpassung dieser Parameter  $x, \sigma$  eine zielführende Veränderung bewirkt werden. In einer  $(1 + 1)$ -Evolutionsstrategie kann beispielsweise Rechenbergs  $\frac{1}{5}$ -Regel zur Anpassung der Mutationswahrscheinlichkeitsverteilungsparameter eingesetzt werden [99, 100]. Pseudocode 1 demonstriert die Parameteraktualisierung der Mutationswahrscheinlichkeitsverteilung für eine  $(1 + 1)$ -Plusselektion und einer Evolutionsstrategieadaption über den Parameter  $\sigma$ .

Fortgeschrittene Evolutionsstrategien modellieren die Mutationswahrscheinlichkeitsverteilung in der Regel mit einer Kovarianzmatrix [101, 102, 103]. Durch die Anpassung der Kovarianzmatrix (E: *Covariance Matrix Adaptation, CMA*) nutzt die Evolutionsstrategie die Fehlerlandschaft der Fitnessfunktion für die Populationsentwicklung. Pseudocode 2 zeigt eine vereinfachte Evolutionsstrategie auf Basis der Kovarianzmatrixadaption bei einer  $(\mu, \lambda)$ -Kommaselektion. Der Vektor  $m$  entspricht hierbei den Mittelwerten der Mutationswahrscheinlichkeitsverteilungskomponenten.

Die Gewichte  $w_{1:\mu}, \dots, w_{\mu:\mu}$  entsprechen den Rängen der Individuen bezüglich der Fitnessfunktion, sodass besser angepasste Individuen höher gewichtet werden. Der Parameter  $\eta$  erfüllt im Wesentlichen die Funktion einer Lernrate, die dabei hilft, dass das Optimierungsverfahren nicht in lokalen Optima endet.

---

**Algorithmus 1**  $(1+1)$ -Evolutionsstrategie nach Rechenbergs  $\frac{1}{5}$ -Regel

---

```

1: Parameter  $c > 1$ 
2: Initialisiere  $x \in \mathbb{R}^n$ ,  $\sigma > 0$ 
3: while  $\neg$  Abbruchkriterium do
4:    $x' \leftarrow \mathcal{N}(x, \sigma^2)$ 
5:   if  $f(x') \leq f(x)$  then
6:      $x \leftarrow x'$ 
7:      $\sigma \leftarrow \sigma \cdot c^4$ 
8:   else
9:      $\sigma \leftarrow \frac{\sigma}{c}$ 
10:  end if
11: end while
```

---

**Algorithmus 2**  $(\mu, \lambda)$ -Evolutionsstrategie mit Kovarianzmatrixadaption

---

```

1: Parameter  $\mu, \lambda, \eta, w_{1:\mu}, \dots, w_{\mu:\mu}, m, C$ 
2: while  $\neg$  Abbruchkriterium do
3:   for  $i \in \{1, \dots, \lambda\}$  do
4:      $z_i \leftarrow \mathcal{N}(\boldsymbol{\mu}, \mathcal{I})$ 
5:   end for
6:   for  $i \in \{1, \dots, \lambda\}$  do
7:      $\delta_i = \sqrt{C} \cdot z_i \mathcal{N}(\boldsymbol{\mu}, \mathcal{C})$ 
8:   end for
9:   for  $i \in \{1, \dots, \lambda\}$  do
10:     $x_i = m + \delta_i \mathcal{N}(\boldsymbol{\mu}, \mathcal{C})$ 
11:  end for
12:  Bestimme  $x_{1:\mu}, \dots, x_{\mu:\mu}$  nach  $f(x_1), \dots, f(x_\lambda)$ 
13:   $\hat{m} = \sum_{r=1}^{\mu} w_{r:\mu} \cdot x_{r:\mu}$ 
14:   $\hat{C} = \sum_{r=1}^{\mu} w_{r:\mu} \cdot \text{delta}_{r:\mu} \text{delta}_{r:\mu}^T$ 
15:   $m \leftarrow \hat{m}$ 
16:   $C \leftarrow (1 - \eta)C + \eta \hat{C}$ 
17: end while
```

---

## 2.2 Tiefe künstliche neuronale Netzwerke

Inspiriert durch das biologische Vorbild natürlich gewachsener neuronaler Netzwerke, wie sie beispielsweise im Gehirn des Säugetiers vorkommen, werden Modelle und Algorithmen entwickelt, welche neurowissenschaftliche Erkenntnisse mathematisch abstrahieren und maschinell anwendbar machen. Einzelne neuronale Netzwerke, beispielsweise das mehrschichtige Perzeptron, das faltende neuronale Netzwerk oder das Long Short-Term Memory Netzwerk, können wie Bausteine miteinander zu einer neuronalen Architektur verbunden werden. Die zu findenden Entscheidungsparameter eines künstlichen neuronalen Netzwerks werden als Parametervektor  $\Theta$  bezeichnet, dessen Komponenten die lernenden synaptischen Gewichte  $w$  und zugehörigen Biaswerte  $b$  darstellen. Zur Modellierung einer gemessenen Werteverteilung entlang einer Eingabedimension  $x$  nennt eine Fehlerfunktion  $L(y, t)$ , auch Verlustfunktion (E: *Loss function*) genannt, die Abweichung der hypothetischen Vorhersage  $y(x)$  zu den tatsächlichen Zielwerten  $t$ .

In einem Verfahren, welches als Fehlerrückführung (E: *Backpropagation of errors*) bezeichnet wird, werden die lernbaren Parameter  $w$  mittels eines stochastischen Gradientenabstiegs schrittweise derart verändert, dass die Abweichung zwischen Eingabedaten  $x$  und Zielgrößen  $t$  minimiert wird. Im Fall des überwachten Lernens sind die Werte  $t$  bekannt, im Fall des unüberwachten Lernens wird an dieser Stelle eine inhärente oder emergente Eigenschaft des jeweiligen Optimierungsproblems gewählt, beispielsweise der Abstand zwischen Datenpunkten, beziehungsweise deren Ähnlichkeit. Geringdimensionale Zusammenhänge können in der Regel mit einem rein vorwärtsverbundenen mehrschichtigen Perzeptron (E: *Multilayer Perceptron*, MLP) abgebildet werden. MLPs sind universelle Funktionsapproximatoren. Dies bedeutet, dass ein MLP eine Funktion  $f(x)$  bis auf eine minimale Abweichung  $\epsilon$  genau abbilden kann, wenn genügend lernbare Entscheidungsparameter vorhanden sind [104]. Demnach existiert für jede  $n$ -dimensionale Funktion  $f : [0, 1]^n \rightarrow \mathbb{R}$  ein MLP mit endlicher Neuronenzahl, sodass:

$$|y(x) - f(x)| \leq \epsilon : \forall x_i \in [0, 1]^n \quad (2.3)$$

Einige Lernprobleme, insbesondere solche aus der digitalen Bildverarbeitung, zeichnen sich durch einen Datenraum hoher Dimensionalität aus, welche für eine Abbildung durch ein MLP entsprechend viele Parameter benötigen. Das Training dieser Parameter ist so ressourcenintensiv, dass in der Praxis andere neuronale Architekturen eingesetzt werden, welche durch ihre Struktur einen Zeitvorteil gewähren. Zum Beispiel können Muster in den Nachbarschaften der Eingabebildpunkte mit einem, vom visuellen System inspirierten, faltenden neuronalen Netzwerk (E: *Convolutional Neural Network*, CNN) gelernt werden, was die Anzahl der benötigten Parameter mindestens um den Faktor der Filtergröße reduziert. Die Anzahl der Filterparameter eines CNN ist also vergleichsweise gering und somit schnell zu lernen, die strukturellen Hierarchien dieser Netzwerke vereinfachen die Zusammensetzung komplexerer Muster aus einfachen Mustern.

Von essentieller Bedeutung in allen Ansätzen des Deep Learning ist das Konzept der neuronalen Aktivierung  $a$ , welche im Prinzip die Summe der Aktivierungen der eingehenden Signale abbildet. Je nach Aufbau werden verschiedene Nichtlinearitäten in Form

von Aktivierungsfunktionen  $\sigma(a)$  mit in die Berechnungen aufgenommen. Die Aktivierungsfunktion führt zur Begrenzung der neuronalen Aktivierung, was der Begrenzung der Feuerraten eines biologischen Neurons entspricht. Nach Kenntnis der Datenlage und der neuronalen Architektur kann ein passendes Optimierungsverfahren gewählt werden, um den Vorhersagefehler auf die einzelnen neuronalen Gewichte zurückzuführen und diese, entsprechend dem Optimierungskriterium, zu verändern. Hierbei wird zunächst schichtweise die Aktivierung bestimmt, zuletzt die Aktivierung der Ausgabeschicht betrachtet und einer erwarteten Zielausgabe gegenübergestellt.

Neben den Anwendungen und vielzähligen Erfolgen im Bereich künstlicher Intelligenz werden auch die Grenzen und Nachteile dieses Ansatzes untersucht. In einem Artikel über die Grenzen des Deep Learning identifiziert [105] einige problematische Eigenschaften und Schwachstellen. Zum einen benötigt das Training tiefer Netzwerke verhältnismäßig viele Daten und die gelernten Zusammenhänge bleiben unerklärt [106, 107]. Zum anderen sind künstliche neuronale Netzwerke nicht vor Angriffen über manipulierte Eingabedaten geschützt [108]. Der Parametervektor eines tiefen künstlichen neuronalen Netzwerks wird in der Regel schrittweise in Bezugnahme auf den Gradienten einer Verlustfunktion optimiert; dabei kann der Parametervektor jedoch in ein lokales Optimum verfallen oder in Form einer Überanpassung die Generalisierungsfähigkeit verlieren. Ein solches, durch ein suboptimal parametrisiertes Lernverfahren mangelhaft konditioniertes, künstliches neuronales Netzwerk erlaubt Angriffe gegen seine Funktionsweise. Beispielsweise kann ein minimal verändertes Bild von einem künstlichen neuronalen Netzwerk falsch klassifiziert werden. Auf diese Weise kann eine Abweichung berechnet werden, die zu einer systematischen Fehlklassifikation führt [108, 109, 110].

### 2.2.1 Aktivierungsfunktion

Durch Aktivierungsfunktionen wird eine essentielle Nichtlinearität eingeführt, welche es einem neuronalen Optimierungsverfahren erlaubt, verschiedene optimale Parameterkonfigurationen anzunehmen. Die Aktivierung eines Neurons ist das Signal, welches an nachfolgende Neuronen weitergeleitet wird und bildet sich im Prinzip aus der Summe der Aktivierungen vorhergehender Neuronen, multipliziert mit dem jeweiligen synaptischen Gewicht. Nach biologischem Vorbild natürlicher Nervenzellen, deren Physiologie die maximale Anregung begrenzt, wird die Aktivierung  $a$  für ein künstliches Neuron durch eine Aktivierungsfunktion  $\sigma(a)$  begrenzt. Die Wahl der Funktionsvorschriften der Aktivierungsfunktionen einzelner neuronaler Schichten unterliegt den durch die Aufgabenstellung gegebenen Bedingungen. In Abbildung 2.2 werden die wichtigsten Aktivierungsfunktionen gezeigt, welche die Ausgabeaktivierung  $\sigma(x)$  eines künstlichen Neurons entlang einer Eingabedimension  $x$  auf unterschiedliche Weise begrenzen.

Welche Aktivierungsfunktion in welcher neuronalen Schicht genutzt wird hängt vom zu lösenden Optimierungsproblem ab. Um während der Fehlerrückführung verschwindend geringe oder über die Maßen hohe Gradienten zu vermeiden sind Aktivierungsfunktionen in der Regel stetig differenzierbar. Liegt die Zielgröße per Definition im Bereich  $[-1, 1]$ , kann  $\tanh(x)$  als Nichtlinearität verwendet werden. Für Zielwerte im Bereich  $[0, 1]$ , wie sie bei der Regression von Prozentzahlen oder bei Zuordnungswahrscheinlichkeiten in

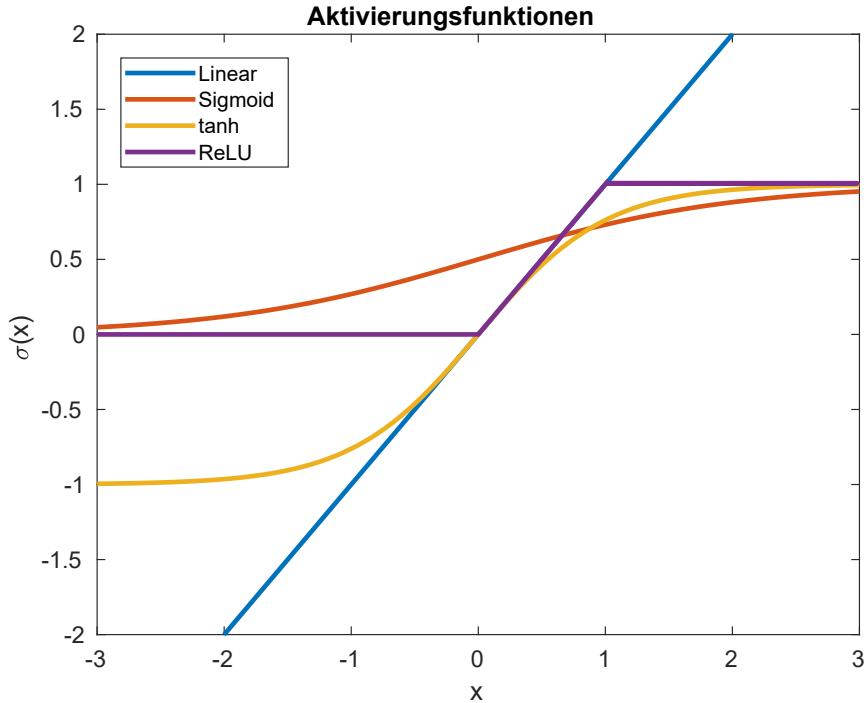


Abbildung 2.2: Eine Zusammenstellung der, im Kontext tiefer künstlicher neuronaler Netzwerke, häufig verwendeten Aktivierungsfunktionen. Die verschiedenen neuronalen Aktivierungsfunktionen  $\sigma(x)$  weisen einer Anregung entlang einer Dimension  $x$  eine beschränkte Ausgabeaktivierung zu. Mit Ausnahme der *ReLU* Funktion sind alle dargestellten Aktivierungsfunktionen stetig differenzierbar.

Klassifikationsproblemen auftreten, bietet sich entweder die Sigmoidfunktion oder eine Aktivierung über gleichrichtende Lineareinheiten (*E: Rectifying Linear Unit, ReLU*) an. Bei der Sigmoidfunktion, auch bezeichnet als logistische Aktivierung, kann folgende Eigenschaft der Ableitung genutzt werden, um Berechnungen zu beschleunigen:

$$\frac{\sigma'(a)}{\sigma(a)} = 1 - \sigma(a) \quad (2.4)$$

Die ReLU-Aktivierungsfunktion schneidet Werte außerhalb des Bereichs  $[0, 1]$  einfach ab, sodass der Gradient außerhalb des Wertebereichs gleich null ist. Weiterhin existieren Aktivierungsfunktionen, welche die Aktivierung in der gesamten neuronalen Schicht betrachten. So normiert beispielsweise die *softmax*-Aktivierungsfunktion die Aktivierung einer gesamten neuronalen Schicht auf einen festgelegten Wertebereich und eignet sich somit, um die Aktivierungsverteilung der Schicht als quantisierte Wahrscheinlichkeitsverteilung zu formulieren. Zur Annäherung an lineare Werte, beispielsweise Belohnungssignale einer Umgebung, können lineare Aktivierungen angenommen werden.

## 2.2.2 Fehlerrückführung

Die numerischen Methoden des Gradientenabstiegs dienen zur Lösung eines Optimierungsproblems über die schrittweise Annäherung eines Parametervektors  $\Theta$  an einen optimalen Parametervektor  $\Theta^*$ . Durch die Ableitung der synaptischen Gewichte nach dem Abstand zwischen Vorhersage- und Zielwert können die Richtungen und Längen der Änderungen am Parametervektor berechnet werden. Die schichtweise Berechnung der Ableitungen der Aktivierung mit der Kettenregel wird als Fehlerrückführung bezeichnet. Zunächst werden in einem Vorwärtspass schichtweise die Aktivierungen im künstlichen neuronalen Netzwerk, von der Eingabeschicht bis hin zur Ausgabeschicht, berechnet. Eine Fehler- oder Verlustfunktion  $L$  gibt die Differenz zwischen Ausgabe- und Zielwerten an. Auf Grundlage von Verlust  $L$  und Aktivierung  $a$  wird ein Gradient bezüglich der Gewichte  $w_i \in \Theta$  aufgestellt:

$$\Delta\Theta = \frac{dL}{dw_i} \quad (2.5)$$

In einem Rückwärtspass wird der Gradient für jedes Gewicht berechnet und auf dieses addiert, sodass die Fehlerfunktion minimiert wird. Eine Lernrate  $\eta$  tritt hierbei als Faktor auf, um das Ausmaß der Veränderungen gering genug zu halten, damit das Optimum nicht übersprungen wird. Pseudocode 3 illustriert den Ablauf dieses Optimierungsverfahrens:

---

### Algorithmus 3 Fehlerrückführung

---

```

Eingabewerte x, Zielwerte t
Berechne schichtweise die Aktivierungen ai und Ausgabe y
Berechne die Fehlerfunktion L(y, t)
Berechne die Ableitung Δw0 ← dL/da0
for wi ∈ W \ w0 do
    Berechne die Ableitung Δwi ← dL/dwi = dL/dai · dai/dwi
end for
Ändere die synaptischen Gewichte wi ← wiη · Δwi

```

---

In Anwendungsfällen mit großem Datenaufkommen benötigt ein einfacher stochastischer Gradientenabstieg in der Regel viele Lernschritte. Zur Optimierung der benötigten Rechenzeit kann ein solcher Algorithmus auch als Stapellernverfahren formuliert werden. Für einen Datenstapel der Größe  $b$  werden die berechneten Abweichungen über den Stapel akkumuliert. Es existieren mathematisch erweiterte Ansätze, um die Fehlerrückführung zu stabilisieren oder zu beschleunigen [111]. Diese Methoden verändern nicht nur den Parametervektor selber, sondern ändern auch die Geschwindigkeit der Änderungen, nach dem Vorbild des physikalischen Konzeptes der Beschleunigung. Dieser Ansatz erlaubt eine Beschleunigung der Gewichtsänderungen auf den relevanten Teilen des Parametervektors. Die Methode *RMSProp* nutzt den Mittelwert der bisherigen Gradienten als Skalierungsfaktor für die derzeitige Lernrate  $\eta_t$ ; dieser Ansatz führt bei tiefen neuronalen Netzwerken zu schnellerer Konvergenz [112].

In konkreten Implementierungen können die Berechnungen der Lernraten  $\eta_t$ , gegeben ein Gradientenbetrag  $g$ , wie folgt lauten:

$$r_t = \rho r_{t-1} + (1 - \rho) * g^2 \quad (2.6)$$

$$\eta_t = \frac{\eta}{\sqrt{r_t + \epsilon}} \quad (2.7)$$

Typische Werte für die Hyperparameter  $\rho$  und  $\epsilon$  lieben bei etwa  $\rho = 0,95$  und  $\epsilon = e^{-6}$ . Andere fortgeschrittene Ansätze, wie beispielsweise *ADAM*, *AdaDelta* oder *AdaGrad*, skalieren die Lernrate  $\eta_t$  mittels des Verhältnisses der vorangegangenen zu den derzeitigen Gradientenbeträgen [113, 114, 115]. Hierbei ändert sich ein Parameters  $w$  in Abhängigkeit des Verhältnisses der Gradienten sowie zweier Hyperparameter  $\beta_1$  und  $\beta_2$ :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) * g_t \quad (2.8)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) * g_t^2 \quad (2.9)$$

$$a_t = \eta \frac{(1 - \beta_2)^t}{(1 - \beta_1)^t} \quad (2.10)$$

$$\Delta w = \frac{a_t \cdot m_t}{\sqrt{v_t + \epsilon}} \quad (2.11)$$

In einigen Fällen, beispielsweise bei verteilten Architekturen oder Lernproblemen paralleler Natur, bietet es sich an, den Gradientenabstieg auf asynchrone Weise zu parallelisieren. Unter der Randbedingung, dass eine einzelne Änderung nur kleine Teile des Parametervektors überschreibt, können die Änderungen einfach nacheinander auf den synaptischen Gewichtsvektor addiert werden. Hierbei bietet das *HOGWILD!*-Schema optimale Konvergenzraten für das Training verteilter Modelle mit einem geteilten Parametervektor, ohne zeitintensive Sperrungen der betroffenen Speicherbereiche [116].

### 2.2.3 Mehrschichtiges Perzeptron

Das Perzeptron bezeichnet ein künstliches Neuron, dessen Aktivierung sich als gewichtete Summe der Eingabesignale ergibt [117]. Mehrere Perzeptrone, welche Eingangssignale derselben Vorgänger erhalten, können in einer neuronalen Schicht zusammengefasst werden. Ein mehrschichtiges Perzeptron (E: *Multilayer Perceptron*, MLP) besteht aus mehreren solcher Schichten, welche die Ausgabeaktivierung der vorhergehenden Schicht als Eingabesignal nutzen. Das zur Erklärung in Abbildung 2.3 dargestellte einfache mehrschichtige Perzeptron besteht aus einer Schicht von Eingabeneuronen  $\underline{x}$ , einer versteckten Schicht  $\underline{z}$  und einer Ausgabeschicht, in diesem einfachen Beispiel mit nur einem einzigen Neuron  $y$ . Wird eine lineare Aktivierung des Ausgabeneurons  $y$  angenommen, so ergibt sich dessen Funktionsvorschrift als inneres Produkt des Aktivierungsvektors der versteckten Schicht  $\underline{z}$  mit der zugehörigen synaptischen Gewichtsmatrix  $\underline{\underline{W}}^{(2)}$ :

$$y = w_0^{(2)} + \sum_i w_i^{(2)} \cdot z_i \quad (2.12)$$

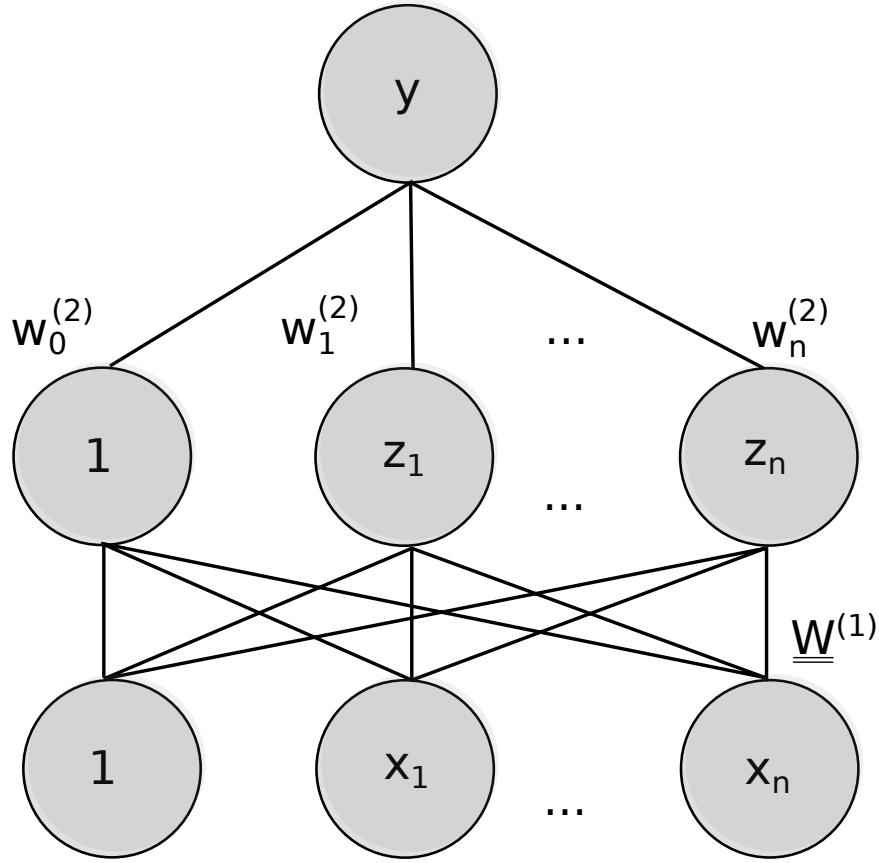


Abbildung 2.3: Ein einfaches MLP mit einer Eingabeschicht  $\underline{x}$ , einer versteckten Schicht  $\underline{z}$  und einer Ausgabeschicht, welche in diesem Beispiel aus nur einem einzigen Neuron  $y$  besteht. Die konstanten Biaswerte betragen 1 und dienen als additiver Term. Die synaptischen Gewichte zwischen den Schichten präsentieren sich als Matrizen  $\underline{\underline{W}}^{(1),(2)}$ .

Hierbei entspricht die skalare Aktivierung der einzelnen neuronalen Komponenten  $z_j$  in der versteckten Schicht einer Multiplikation des Eingabeneuronenvektors  $\underline{x}$  mit den jeweiligen synaptischen Gewichten  $w_{ij}^{(1)}$  der ersten Schicht, gefolgt von einer, in der Regel stetig differenzierbaren, Aktivierungsfunktion  $\sigma$ :

$$a_j = \sum_i w_i^{(1)} x_i \quad (2.13)$$

$$z_j = \sigma(a_j) \quad (2.14)$$

Die konstanten Biaswerte 1 werden hierbei als Aktivierung des Neurons  $x_0$  in die Gleichung aufgenommen. Die Verlustfunktion  $L$  zwischen der Ausgabe  $y$  und den Zielwerten  $t$  kann, für die Architektur in Abbildung 2.3, vektoriell formuliert werden als:

$$L = |y(\underline{x}) - \underline{t}| \quad (2.15)$$

Der Fehler  $L_{z_j}$  bezüglich der synaptischen Gewichte eines Neurons der Zwischenschicht  $z_j$  kann mit Kenntnis einer, beispielsweise logistischen, Aktivierungsfunktion auf Grundlage des vorhergehenden Fehlers  $L$  berechnet werden:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \quad (2.16)$$

$$L_{z_j} = z_j \cdot (1 - z_j) \cdot L \cdot w_i^{(2)} \quad (2.17)$$

$$L_{z_j} = \sigma(a_j) \cdot (\sigma'(a_j)) \cdot L \cdot w_i^{(2)} \quad (2.18)$$

$$L_{z_j} = \sigma(a_j) \cdot (1 - \sigma(a_j)) \cdot L \cdot w_i^{(2)} \quad (2.19)$$

Für die in diesem Beispiel gezeigte Architektur können die Änderungen an den einzelnen synaptischen Gewichte, mit einer Lernrate  $\eta$ , berechnet werden:

$$\Delta w_0^{(2)} = -\eta \cdot L \quad (2.20)$$

$$\Delta w_i^{(2)} = -\eta \cdot L \cdot z_i \quad (2.21)$$

$$\Delta w_{0i}^{(1)} = -\eta \cdot L_{z_i} \quad (2.22)$$

$$\Delta w_{ji}^{(1)} = -\eta \cdot L_{z_i} \cdot x_j \quad (2.23)$$

Wobei die Veränderung des synaptischen Gewichtsvektors auch durch Anwendung der Fehlerrückführung mittels Kettenregel berechnet werden kann:

$$\frac{dL}{dw_{ji}} = \frac{dL}{dz_j} \cdot \frac{dz_j}{dw_{ji}} \quad (2.24)$$

## 2.2.4 Faltendes neuronales Netzwerk

Einfache vorwärtsgerichtete künstliche neuronale Netze, wie das MLP, haben in der Regel vollständig verbundene Neuronen. Folglich steigt die Anzahl der Gewichte mit jedem zusätzlichen Neuron um die Anzahl der Neuronen in der vorherigen Schicht. Dies führt zu einem hohen Ressourcenbedarf, da die Modelldaten mehr Speicherplatz beanspruchen und die Trainingsverfahren mehr Rechenzeit für die Parameteroptimierung benötigen. Bei besonders hochdimensionalen Eingabedaten, wie etwa Farbbildern, steht der Rechenaufwand in schlechtem Verhältnis zur Parametereffizienz. Gibt es in den Eingabedaten strukturelle Merkmale, wie etwa Nachbarschaften, kann die Komplexität des Parametervektors durch Ausnutzung dieser Strukturen reduziert werden. Typischerweise enthalten digitale Farbbilder viele Daten solcher Merkmale; Pixel springen in der Regel nicht beliebig in ihrer Farbe sondern folgen Verläufen von Kanten und Flächen.

Faltende neuronale Netzwerke reduzieren den benötigten Rechenbedarf durch die Nutzung einer neuronalen Faltungsfunktion, was einen praktischen Nutzen für maschinelle Lernverfahren auf Basis digitaler Bilddaten bringt. Um bestimmte Merkmale in einem Eingangsbild zu erkennen, können Filter definiert werden, welche der Struktur der gesuchten spezifischen Merkmale entsprechen. Diese Filter, auch Kernel, Maske oder Faltungsmatrix genannt, werden, bildlich beschrieben, über das Eingangssignal geschoben und erzeugen

an jedem Punkt eine Filterantwort; das Eingabesignal wird also mit dem Filter gefaltet. Die Antwort des Filters an den Ansatzpunkten entspricht dem Vorhandensein des beschriebenen Merkmals. Die Faltungsoperation  $O$  (E: *convolution*) eines Eingabesignals  $I$  mit einem Filter  $F$  kann, beispielsweise für ein zweidimensionales Eingabesignal  $I$ , wie etwa ein Grauwertbild, beschrieben werden als:

$$O(x_1, x_2) = I \circ F(x_1, x_2) \sum_{x'_1} \sum_{x'_2} I(x_1 - x'_1, x_2 - x'_2) \cdot F(x'_1, x'_2) \quad (2.25)$$

Hierbei entsprechen  $(x_1, x_2)$  den Koordinaten im Eingabesignal und  $(x'_1, x'_2)$  den Filterkoordinaten. Das Konzept der Faltungsoperation kann als neuronale Schicht in ein künstliches neuronales Netzwerk integriert werden, indem ein Filter als Gewichtsmatrix formuliert wird. Auf ein beispielsweise zweidimensionales Eingabesignal  $I$  der Dimensionalität  $(N, M)$  kann eine Faltungsschicht mit einer Anzahl von  $k$  Filtern der Weite  $n$  und der Höhe  $m$  folgen. Hier berechnet sich die Anzahl der Neuronen in der entstandenen Faltungsschicht als  $k \cdot (N - n + 1) \cdot (M - m + 1)$ . Diese Faltungsschichten können aufeinander folgen, sodass in jeder Schicht neue Zusammenhänge auf Grundlage der von der Vorgängerschicht erkannten Merkmale gelernt werden können.

Die rezeptiven Felder der ersten Schicht erfassen grundlegende Merkmale, welche die folgenden Schichten zu komplexeren Repräsentationen zusammensetzen. Das rezeptive Feld eines jeden Neurons einer Faltungsschicht erfasst jeweils eine Nachbarschaft aus der Vorgängerschicht. Die so erfassten Nachbarschaften werden mit jedem Faltungskern  $W_i$  gewichtet und ergeben neuronale Aktivierungen in der Faltungsschicht. Die Gesamtaktivierung der Faltungsschicht entspricht den Antworten aller Filter auf das Eingabesignal. Hierbei ergibt sich die Aktivierungsfunktionsvorschrift eines einzelnen Neurons  $y_{ij}$  der Faltungsschicht als:

$$y_{ij}(I) = \sum_{w'} \sum_{h'} I(i - w', j - h') \cdot \underline{W}_{w', h'} \quad (2.26)$$

Um die Komplexität dieser Berechnung zu reduzieren, kann der Filter mit einem Versatz aufgesetzt werden oder nur die Aktivierung des jeweils aktivsten Neurons einer Nachbarschaft weitergeführt werden.

Der aktuelle Stand der Forschung kennt verschiedene Arten von Architekturen faltender neuronaler Netzwerke. Tiefenbasierte faltende neuronale Netzwerke werden durch die Anzahl der aufeinanderfolgenden Schichten definiert [118]. In der Theorie bieten tiefe Netzwerke genauere Abbildungen als flache Architekturen. Das Kaskadieren von Schichten ist jedoch nicht trivial und geht mit einem exponentiellen Anstieg der Rechenkosten einher. Die Erhöhung der Tiefe des faltenden neuronalen Netzwerks durch Hinzufügen von Schichten hat Auswirkungen auf den Erfolg des Lernverfahrens, insbesondere bei überwachten Klassifizierungsaufgaben müssen Hyperparameter wie etwa die Lernrate sorgfältig gewählt werden, da der synaptische Parametervektor des künstlichen neuronalen Netzwerks leicht ein lokales Optimum annehmen kann. Durch das Kaskadieren von Schichten können zwar verschiedene Merkmalsrepräsentationen erlernt werden, aber die Qualität der Merkmale bleibt ungeklärt. Breitenbasierte faltende neuronale Netzwerke

versuchen, den Verlust an den lokalen Minima mit einer größeren Breite der Schichten zu verringern [119]. Viele Parameter und Hyperparameter spielen eine Rolle beim Lernprozess eines faltenden neuronalen Netzwerks, darunter die synaptischen Gewichte, die Biaswerte, die Anzahl der Schichten, die Aktivierungsfunktion und die Lernrate – aber vor allem die Filtergröße. Die Filtergröße bestimmt den Grad der Granularität, die Wahl dieser Größe wirkt sich auf die Korrelation der benachbarten Pixel aus. Filterbasierte faltende neuronale Netzwerke nutzen kleinere Größen zum Extrahieren lokaler Merkmale und größerer Filter zur Extraktion grober Merkmale [120]. Obwohl von tieferen Netzwerken auch höhere Genauigkeiten erwartet werden, können sie mit Problemen wie verschwindenden oder explodieren Gradienten, Leistungsabfall, Überanpassung oder hohen Trainingsfehlern konfrontiert sein [121]. Um diesen Problemen entgegenzuwirken, können Zwischenschichten übersprungen werden, um einen speziellen Informationsfluss zu ermöglichen [122]. Tiefe faltende neuronale Netzwerke sind zwar sehr leistungsfähig beim automatischen Lernen von Nachbarschaftsmerkmalen, allerdings spielen einige Merkmale bei der Objektunterscheidung nur eine geringe oder gar keine Rolle, was eine Überanpassung an schwache Merkmale verursachen und durch gezieltes Annullieren betroffener Filterkanäle verhindert werden kann [123].

### 2.2.5 Long Short-Term Memory

Vorwärtsgerichtete künstliche neuronale Netzwerke, wie das mehrschichtige Perzepron oder das faltende neuronale Netzwerk, halten die neuronale Aktivierung nur für den Moment der Inferenz. Das Signal fließt von der Eingabeschicht zur Ausgabeschicht und verändert sich dabei entsprechend der Gewichtsmatrizen, ist für die nächste Inferenz aber nicht mehr verfügbar. Aus diesem Grund stellen Sequenzen von Eingabesignalen, beispielsweise Video- oder Audioaufnahmen, die vorwärtsgerichteten Ansätze vor konzeptionelle Herausforderungen. Eine vorwärtsgerichtete neuronale Architektur kann zur Verarbeitung dieser Art von Information in der Lage sein, indem mehr lernbare Parameter hinzugefügt werden. Ein rekurrentes künstliches neuronales Netzwerk bietet jedoch den strukturellen Vorteil der zeitübergreifenden Aufbewahrung früherer Information in der Aktivierung der künstlichen neuronalen Zellen. Anstatt ausschließlich vorwärts gerichteter Verbindungen verfügt ein mit sich selbst verbundenes künstliches Neuron auch über eine Gewichtung zur eigenen Aktivierung im letzten Zeitschritt sowie zu anderen Neuronen derselben Schicht. Dies erlaubt den Zellen der rekurrenten Schicht eine wechselseitige Selbstanregung über längere Zeiträume. Die beim Lernvorgang auftretenden Gradienten können divergieren, also verschwindend gering oder sinnwidrig groß werden und somit den Lernvorgang zunichtemachen; hier muss der Fehlerflusses reguliert werden.

Ein Ansatz, der dieser Herausforderung mit dem Konzept sogenannter künstlicher Torneuronen begegnet, ist die Long Short-Term Memory Zelle (LSTM) [124]. Eine einzelne LSTM Zelle besteht im Kern aus einer Erinnerungszelle, welche ihre eigene Aktivierung über die Zeit aufrecht erhalten kann, sowie einer Anzahl von Torneuronen für die Eingabe, Ausgabe und das Vergessen. Die Torneuronen verfügen über eigene Gewichtsmatrizen, welche ähnlich den vorwärtsgerichteten künstlichen neuronalen Netzwerken durch einen Gradientenabstieg verändert werden können.

Die Haltedauer der neuronalen Aktivierung in der Erinnerungszelle wird über das Öffnen und Schließen der neuronalen Toreinheiten bestimmt. Eine Erinnerungszelle  $j$  behält ihre Aktivierung  $y_j$  so lange bei, wie sich die Aktivierung der Torneuronen nicht ändert, also:

$$y_j(t+1) = \sigma(y_j(t+1)) = \sigma(w_{jj}y_j(t)) = y_j(t) \quad (2.27)$$

Im Beitrag von [124] wird  $\sigma$  als lineare Aktivierung gewählt,  $w_{jj} = 1$  gesetzt und das Konzept des *constant error carousel* (CEC) eingeführt. Während der BP kontrolliert das Eingangstorneuron einer LSTM Zelle den Fehlerfluss in das jeweilige CEC, das Ausgangstorneuron die Ausleitung des Fehlerflusses zu den anderen Zellen. Zur Laufzeit vermeidet das Öffnen und Schließen der Torneuronen Störungen durch für die jeweilige Erinnerungszelle unbedeutende Information. Wird das Eingangstorneuron geöffnet, so wird der Aktivierung der Erinnerungszelle  $i$  mit dem Eingabesignal ganz oder teilweise überschrieben. Öffnet das Ausgabetorneuron  $o_i$ , wird der Aktivierung der Erinnerungszelle den verbundenen Neuronen, entsprechend der Gewichtung des Ausgabetorneurons, zugänglich gemacht:

$$o_t = \sigma(a_o(t)) = \sigma(\sum_u w_{ou}y_u(t-1)) \quad (2.28)$$

$$i_t = \sigma(a_i(t)) = \sigma(\sum_u w_{iu}y_u(t-1)) \quad (2.29)$$

Zur Bestimmung der internen Aktivierung  $s_c$  der Erinnerungszelle  $c$  nutzt [124] zwei differenzierbare Funktionen  $g$  und  $h$ :

$$c_t = o_t h(s_c(t)) \quad (2.30)$$

$$s_c(t) = s_c(t-1) + i_t g(\sum_u w_{cu}y_u(t-1)) \quad (2.31)$$

In einem späteren Beitrag von [125] wird die LSTM Architektur um ein Vergessenstorneuron  $f$  zum Erlernen eines systematischen Vergessens gehaltener Aktivierungen erweitert. Dies ermöglicht einer LSTM Zelle, ihre interne Aktivierung zu einem günstigen Zeitpunkt zurückzusetzen. Abbildung 2.4 veranschaulicht diese LSTM Architektur. Hier berechnen sich die Aktivierungen in der gezeigten Architektur wie folgt, wobei  $h$  hier der über die Zeit gehaltenen Aktivierung entspricht:

$$i_t = \sigma_i(x_t W_{xi} + h_{t-1} W_{hi} + w_{ci} \odot c_{t-1} + b_i) \quad (2.32)$$

$$f_t = \sigma_f(x_t W_{xf} + h_{t-1} W_{hf} + w_{cf} \odot c_{t-1} + b_f) \quad (2.33)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(x_t W_{xc} + h_{t-1} W_{hc} + b_c) \quad (2.34)$$

$$o_t = \sigma_o(x_t W_{xo} + h_{t-1} W_{ho} + w_{co} \odot c_t + b_o) \quad (2.35)$$

$$h_t = o_t \odot \sigma_h(c_t) \quad (2.36)$$

Das gängige Gradientenverfahren zum Training eines LSTM Netzwerks ist die Fehlerrückführung durch die Zeit (E: *Backpropagation through time*). Hierbei werden die Parameter des rekurrenten Netzwerks entlang der zu lernenden Sequenz entfaltet.

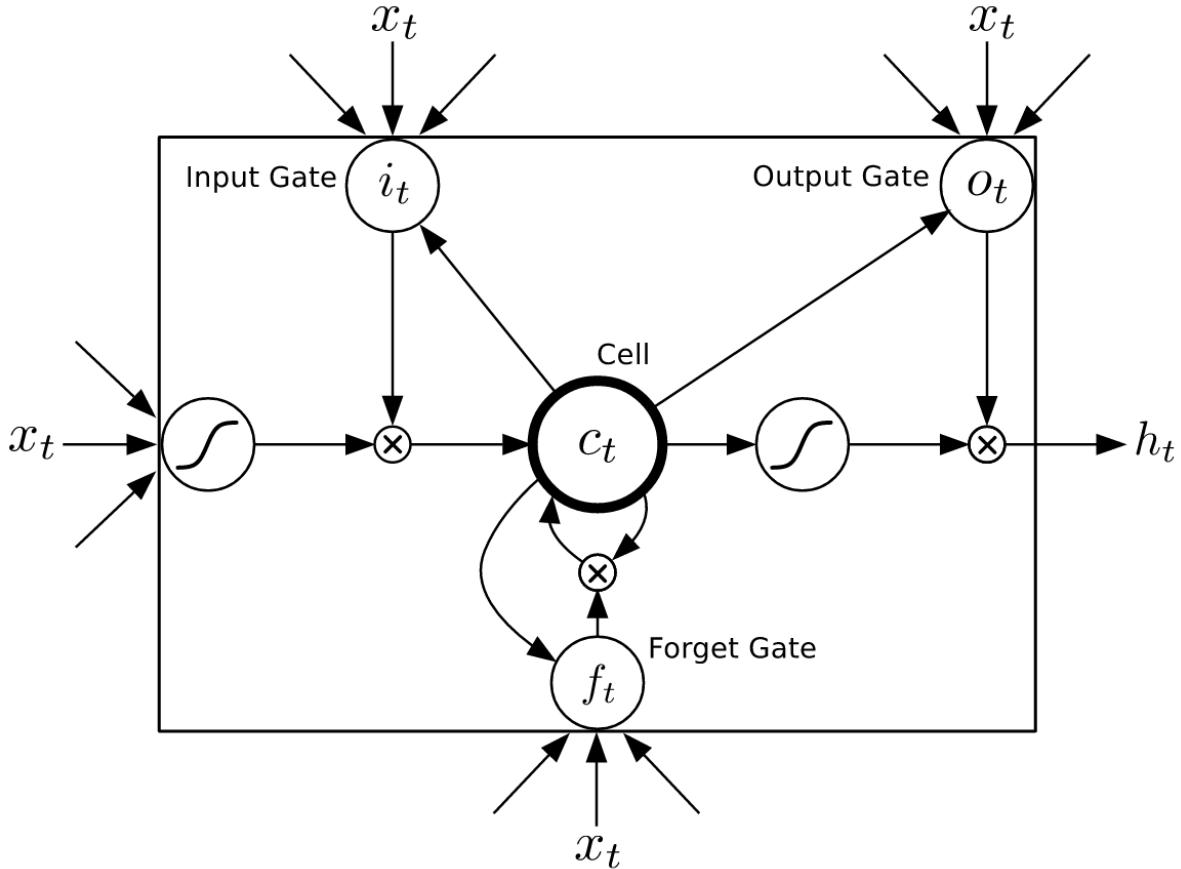


Abbildung 2.4: Eine Zelle der rekurrenten neuronalen Long Short-Term Memory Architektur mit Vergessenstoren, veröffentlicht in [125].

Über diese, in Raum und Zeit des Eingabesignals lokale, Struktur kann nun eine reguläre Fehlerrückausbreitung durchgeführt werden, indem die Gradienten beispielsweise einfach aufsummiert und anschließend auf den Parametervektor addiert werden.

## 2.3 Transferlernen

Die Fähigkeit, verschiedene Aufgaben zu erlernen und das Gelernte als Grundlage für zukünftige Aufgaben weiter zu verwenden, ist für die Entwicklung einer generischen künstlichen Intelligenz von essentieller Bedeutung. Ein tiefes künstliches neuronales Netzwerk ist nicht ohne Weiteres dazu in der Lage, einen solchen Lerntransfer von Hause aus zu leisten; hieraus erwächst das Konzept des Transferlernens. Als Methoden des Transferlernens werden Verfahren bezeichnet, welche Wissen über bereits gelernte Aufgaben nutzen und in ein Optimierungsverfahren mit einbeziehen [126]. Wissen aus einer Aufgabendomäne  $A$  soll genutzt werden, um den Lernvorgang in einer Domäne  $B$  zu verbessern, zu beschleunigen oder zu stabilisieren [127, 128, 23, 25, 6, 129, 44, 45, 130, 131].

### 2.3.1 Verwandte Verfahren

Die Idee des Transferlernens steht in enger Verbindung mit den Konzepten des fortwährenden Lernens (E: *Continual Learning*) [132], des gleichzeitigen Lernens mehrerer Aufgaben (E: *Multitask Learning*) [133], des Lernens von Metarepräsentationen (E: *Meta Learning*) [134] und der Idee der Wissensdistillation (E: *Knowledge Distillation*) [135, 27]. Das Verfahren des Zero-Shot-Learning ermöglicht es, während des Trainings ohne Zugriff auf Labeldaten ein tiefes künstliches neuronales Netzwerk zu trainieren, indem Label aus der Quelldomäne mitsamt Hilfsinformationen verwendet werden [136], die Test- und Trainingsdaten sind disjunkt [137]. Die Annahme hierbei ist, dass die Aufgaben von so ähnlicher Struktur sind, dass das Modell Instanzen von ungewohnten Beispielen allein über Hilfsinformation, wie beispielsweise Zwischenattributklassifikatoren [138], Mischungen gesehener Klassenanteile [139] oder Labelkompatibilität [140], klassifizieren kann. Diese Ansätze stellen sich als besonders nützlich für solche Klassifizierungsaufgaben heraus, in denen es nur wenige Daten gibt [136].

Beim One-Shot-Lernen wird das Netzwerk auf der Grundlage eines oder weniger Lernbeispiele konditioniert, beispielsweise durch die Darstellung von Kernmerkmalen [141] oder durch die Nutzung kernelbasierter Metriken [142]. One-Shot-Lernen ist besonders effizient, wenn die Anzahl der bekannten Labels zunimmt, da das Modell wahrscheinlich bereits Beispiele gesehen hat, welche dem zu lernenden Label ähneln [143]. Manche Methoden des Transferlernens kennen auch das sogenannte Few-Shot-Lernen, welche generell wenig Beispiele benutzen [144, 145, 146, 144].

### 2.3.2 Anwendungen

Maschinelles Lernen, insbesondere das Training tiefer künstlicher neuronaler Netzwerke, ist datenintensiv. Bei genügenden Daten kann es hervorragende Genauigkeiten bringen, aber bei kleinen Datenmengen ist der Erfolg in der Regel begrenzt. Je nach Anwendungsfall und Lernmethode, beispielsweise im Verstärkungslernen (E: *Reinforcement Learning*) [21] oder teilüberwachten Lernen (E: *Semi-Supervised Learning*) [147, 148], bieten sich verschiedene Techniken an [44, 129, 130, 58, 131]. Anwendungen des Transferlernens finden sich in visueller Objektargumentation [149], natürlicher Sprachverarbeitung [150, 151, 152, 153], der Robotik [46] sowie der Objekt- und Personenerkennung [154, 155]. Auch kann Wissen aus einer Simulation auf die Realität übertragen werden, um beispielsweise das Training eines physischen Roboters zu beschleunigen [156]. Transferlernen hilft, zum Beispiel, den Raubbau an natürlichen Ressourcen zu verringern, indem es die Kreislaufwirtschaft unterstützt [68, 26]; es hilft bei der medizinischen Bildgebung und Diagnostik [33] und auch bei der Arbeitssicherheit [32, 31]. Ferner existieren sicherheitskritische Anwendungen, welche über manipulierte Eingabedaten eine feindliche Umprogrammierung vornehmen (E: *Adversarial Reprogramming*) [108]. Methoden des Transferlernens sind insbesondere nützlich für Anwendungen, bei welchen aufgrund von Bedenken bezüglich des Datenschutzes, der Sicherheit oder ethischer Abwägungen wenige Lernbeispiele verfügbar sind [157]. Ein gemeinsamer Vorteil aller dieser Methoden besteht darin, dass sie die Rechenkosten beträchtlich verringern.

### 2.3.3 Feinabstimmung

Eine sehr einfache und, je nach Wahl der Parameter, effiziente Form des Transferlernens besteht in der Feinabstimmung einzelner Bestandteile eines vortrainierten Netzwerks (E: *Finetuning*) [26, 158, 38, 159]. Die Feinabstimmung gilt als Standardverfahren des Transferlernens; wenn beispielsweise Wissen aus einem Faltungsnetzwerk für eine neue Bildklassifikationsaufgabe wiederverwendet werden soll, so geschieht dies in der Regel mit Feinabstimmung der Klassifikationsteile des jeweiligen Netzwerks [26]. Hierzu müssen zum Erlernen einer neuen Aufgabe also lediglich die Gewichte in den relevanten Ausgabeschichten neu optimiert werden, die anderen Parameter können jedoch unverändert erhalten bleiben, wie in Abbildung 2.5 skizziert.

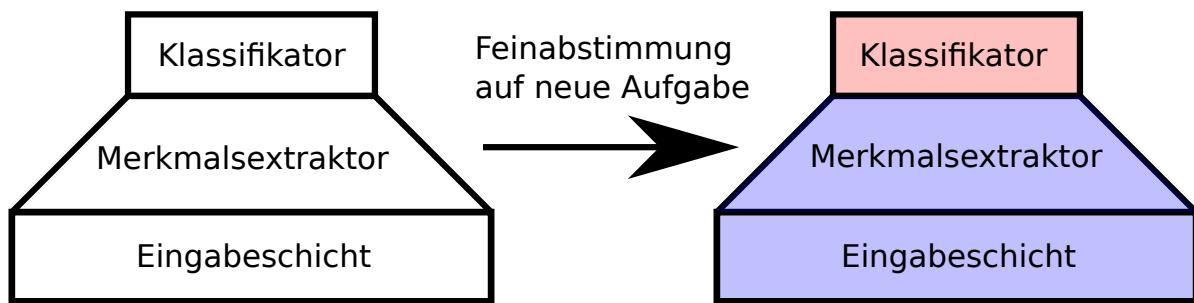


Abbildung 2.5: Eine konzeptionelle Skizze der Feinabstimmung des Klassifikatornteils eines beispielhaften faltenden neuronalen Netzwerks zur Bildklassifikation; das Lernen relevanter Merkmale geschieht über Faltungsschichten. Links: das vortrainierte Netzwerk. Rechts: die Feinabstimmung. Die blauen Schichten (Eingabeschicht und Merkmalsextraktion) bleiben unverändert, die rote Schicht (Klassifikation) wird feinabgestimmt.

In der Ausarbeitung von [38] wird die Feinabstimmung großer vortrainierter Modelle der natürlichen Sprachverarbeitung (E: *Natural Language Processing*, NLP) untersucht und festgestellt, dass in vielen Zielaufgaben die Feinabstimmung in Bezug auf die Nutzung der zur Verfügung stehenden lernbaren Parameter nicht effizient ist; für jede neue Zielaufgabe wäre ein völlig neues Modell erforderlich. Als Lösungsansatz wird die Übertragung mit Adaptermodulen vorgeschlagen. Adaptermodule ergeben ein kompaktes und erweiterbares Modell, in welchem nur wenige lernbare Parameter pro Aufgabe hinzugefügt werden. Die Untersuchungen von [158] nutzen Feinabstimmungsmethoden zur Erkennung von Hirntumoren auf Magnetresonanzbildern mit einer generischen Methode, welche kein menschliches Vorwissen verwendet und nur eine minimale Vorverarbeitung erfordert. Im Beitrag von [159] wird ein adaptiver Feinabstimmungsansatz vorgeschlagen, welcher die optimale Feinabstimmungsstrategie für die Daten einer Zielaufgabeninstanz ermittelt. Hierbei wird ein Bild aus der Zielaufgabe und ein Strategienetzwerk verwendet, welches Entscheidungen darüber trifft, ob das Bild durch die fein abgestimmten Schichten oder durch die vortrainierten Schichten geleitet wird.

Bei unregulierter Feinabstimmung eines künstlichen neuronalen Netzwerks auf mehrere Aufgaben tritt in der Regel das Problem des katastrophalen Vergessens auf (E: *catastrophic forgetting*) [23]. Durch das Überschreiben aufgabenspezifischer synaptischer Strukturen besteht die Gefahr, dass wichtige Wissenszusammenhänge mit dem Aufkommen neuer Aufgaben verlernt werden. Dies führt dazu, dass das betroffene künstliche neuronale Netzwerk keine der zugesagten Aufgaben mehr erfüllen kann. Daher werden beim Finetuning die alten Modelldateien in der Regel beibehalten, falls optimale Lösungen für vorherige Aufgaben benötigt werden.

### 2.3.4 Elastische Gewichtskonsolidierung

Ein weiterer gut untersuchter Ansatz des Transferlernens, welcher das Problem des katastrophalen Vergessens durch einen Regularisierungsterm im Optimierungskriterium überwindet, ist die Konsolidierung synaptischer Gewichte nach Aufgabenrelevanz [23, 25]. Hierbei besteht der Grundgedanke in der Identifikation derjenigen Teile des Parametervektors, welche für eine Aufgabe relevant sind, um diese beim Lernen neuer Aufgaben möglichst nicht zu verändern. Die Bedeutsamkeit der einzelnen synaptischen Gewichte zur Erfüllung der Aufgabe können einer Fisherinformationsmatrix abgelesen werden, welche während des Lernvorgangs mitberechnet wird [160].

Ohne die Parameteranzahl des künstlichen neuronalen Netzwerks zu verändern, wird Information über zusätzliche Aufgaben in den nicht relevanten Teilen des Parametervektors untergebracht, indem die Lernrate in diesen Teilen erhöht und in den anderen gesenkt wird. In den für eine Aufgabe  $A$  als relevant identifizierten synaptischen Gewichten wird für das Lernen einer neuen Aufgabe  $B$  die Lernrate reduziert, sodass das Wissen über Aufgabe  $A$  mit höherer Wahrscheinlichkeit erhalten bleibt.

Kirkpatrick et al. zeigen die Umsetzung eines solchen Ansatzes für Klassifikations- und Verstärkungslernaufgaben, in welchem das Erlernen der für eine Aufgabe relevanten neuronalen Gewichte selektiv verlangsamt wird [23]. Die vorgestellte Optimierungsmethode beruht auf dem Konzept einer elastischen Konsolidierung der synaptischen Gewichte (E: *Elastic Weight Consolidation, EWC*), skizziert in Abbildung 2.6.

Von essentieller Bedeutung ist die Fisherinformationsmatrix  $F$ . Die Matrix  $F$  ist positiv semidefinit und gleicht nahe dem Minimum der Verlustfunktion der zweiten Ableitung, kann aber auf der Basis von Ableitungen erster Ordnung berechnet werden und ist somit auch für große Modelle geeignet [161]. Mit einer Abschätzung  $\lambda$  der Wichtigkeit der Aufgabe  $A$  bezüglich einer Aufgabe  $i$  wird in der elastischen Gewichtskonsolidierung die folgende Verlustfunktion zur Annäherung des, für beide Aufgaben optimalen, synaptischen Parametervektors  $\Theta_{A,i}^*$  minimiert [23]:

$$L = L_B + \sum_i \frac{\lambda}{2} F_i (\Theta_i - \Theta_{A,i}^*)^2 \quad (2.37)$$

Zum Erlernen weiterer Aufgaben werden die Parameter des künstlichen neuronalen Netzwerks bei einer elastischen Gewichtskonsolidierung nahe am bisher optimalen Parametervektor  $\Theta_{A,i}^*$  gehalten.

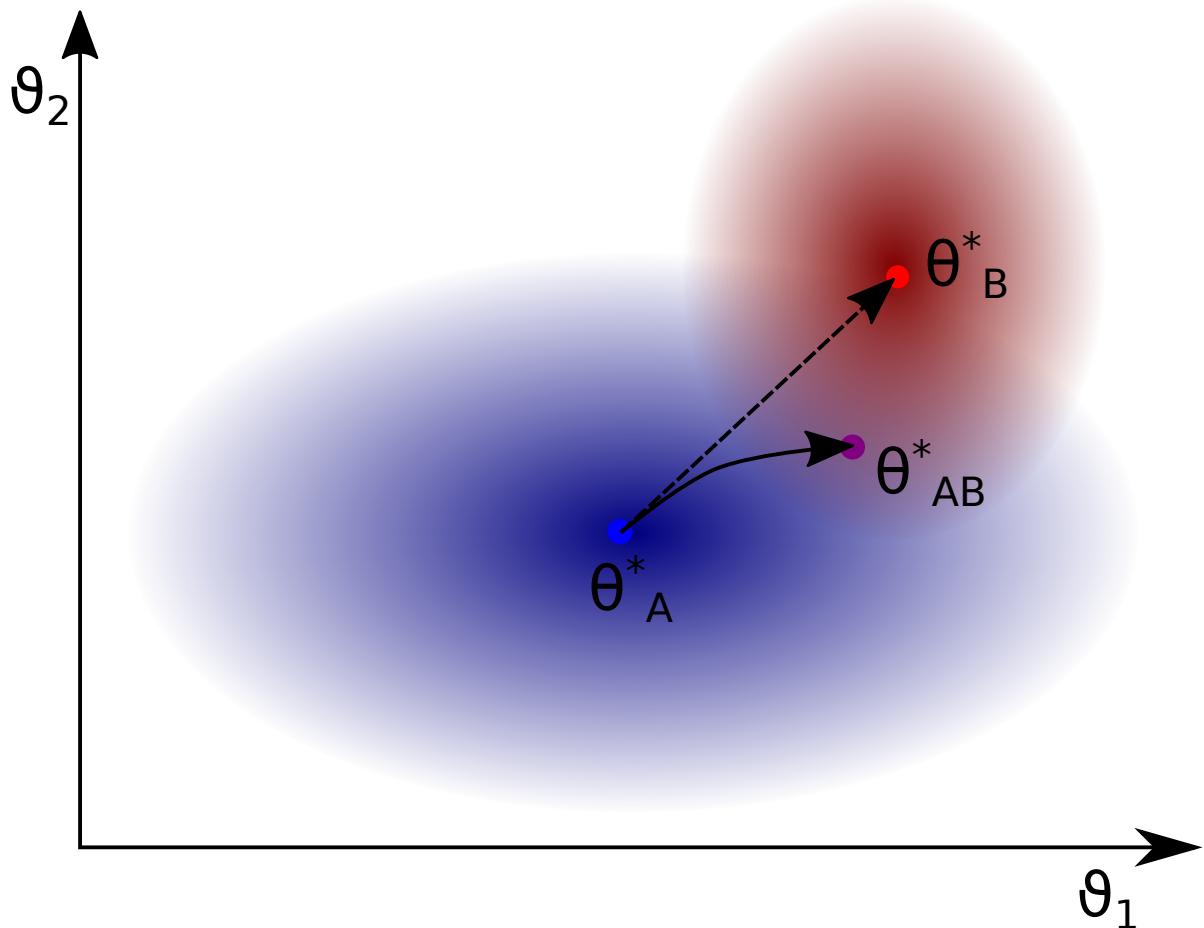


Abbildung 2.6: Konzeptionelle Skizze der elastischen Gewichtskonsolidierung (EWC) für einen beispielhaften zweidimensionalen Parametervektor  $\Theta = (\theta_1, \theta_2)^T$ . Für zwei Aufgaben  $A, B$  liegen die jeweils optimalen Parametervektoren  $\Theta_A^*, \Theta_B^*$  im Minimum der jeweiligen Fehlerregion der Aufgabe ( $A$  blau,  $B$  rot). Ein einfaches Neutrainieren eines für Aufgabe  $A$  konditionierten Netzwerks führt zum Optimum für Aufgabe  $B$  (gestrichelte Linie). Die Methode der EWC führt hingegen zu einem Optimum  $\Theta_{AB}^*$ , welches bei gleicher Parameteranzahl beide Aufgaben gut lösen kann.

Die initialen Gewichte des künstlichen neuronalen Netzwerks zu Beginn des Lernens einer neuen Aufgabe sind also die Gewichte nach Abschluss des vorherigen Lernvorgangs. Nach bayes'scher Formulierung ist die a-posteriori-Verteilung nach einem Training also die a-priori-Verteilung vor dem nächsten Training. Die elastische Gewichtskonsolidierung nimmt eine diagonale Kovarianzmatrix in der a-posteriori-Verteilung an, trifft also zunächst die Annahme, dass die synaptischen Gewichte nicht korrelieren.

Diese Annahme vernachlässigt die Bedeutung der Aufgabenbezogenheit zusammenhängender Komponenten des Parametervektors. In einer weiterführenden Methode der neuronalen Gewichtskonsolidierung, namentlich der inkrementellen Momentabgleichung (E: *Incremental Moment Matching*, IMM), wird die Bedeutsamkeit der Korrelationen in der a-posteriori-Verteilung modelliert [25]. Zur Modellierung der a-posteriori Korrelationswahrscheinlichkeiten kann eine Gaußverteilung  $q$  angenommen werden. Für eine Sequenz von  $K$  Aufgaben mit den Datensätzen  $(x_{1:K}, y_{1:k})$  kann die wahre A-posteriori-Verteilung  $p_k$  inkrementell über die Aufgaben  $k \in K$  approximiert werden:

$$p_k = p(\Theta|x_k, y_k) \approx q_k = q(\Theta|\mu_k, \Sigma_k) \quad (2.38)$$

Zu ermitteln ist der optimale Mittelwertvektor  $\mu_1^*: K$  und die optimale Kovarianzmatrix  $\Sigma_{1:K}^*$  der Verteilung  $q$ . Die Dimensionalität des Vektors  $\mu$  und der Matrix  $\Sigma$  ergibt sich aus der Parameteranzahl des künstlichen neuronalen Netzwerks. Die Ermittlung von  $\mu_k$  und  $\Sigma_k$  kann auf verschiedene Weisen realisiert werden, was zu unterschiedlichen Transfertechniken führt [25]. So kann die inkrementelle Momentabgleichung der Gaußverteilungen beispielsweise auf der Minimierung einer Kullback-Leibler-Divergenz basieren.

### 2.3.5 Progressive neuronale Netzwerke

Ein anderer Ansatz des Transferlernens, welcher auf dem Hinzufügen neuer Parameter basiert, sind progressive neuronale Netzwerke [156, 46]. Für jede neue Aufgabe wird ein neues künstliches neuronales Netzwerk angelegt, welches durch einseitige Verbindungen eine Art Lesezugriff auf den Zustand des künstlichen neuronalen Netzwerks der vorherigen Aufgabe hat. Während des Lernvorgangs für die neue Aufgabe werden die Parameter des vorherigen Netzwerks eingefroren und nicht weiter verändert.

Sind die Aufgaben einander ähnlich, kann angenommen werden, dass die neuronale Aktivität im früheren Netzwerk bezüglich des fremden Eingabedatums dennoch Information beinhaltet, die für einen Transfer in die neue Aufgabe geeignet sein kann. Zunächst wird angenommen, dass die Eingabedaten der verschiedenen Aufgaben in selber Dimensionalität präsentiert werden können. So kann ein Eingabedatum einer neuen Aufgabe an ein früheres, nicht für diese neue Aufgabe trainiertes, Netzwerk weitergegeben werden. Unter Annahme ähnlicher Aufgaben kann nun die Aktivität im vorhergehenden Netzwerk für einen Wissenstransfer genutzt werden. Über ein Adapternetzwerk verfügt das nachfolgende Netzwerk über laterale Verbindungen zu den Neuronen des vorhergehenden Netzwerks. Nach [156, 46] lässt sich dieser Zusammenhang generalisieren als:

$$h_k, i = \sigma(W_{k,i} h_{k,i-1} + \sum_{j < k} U_{i,k:j} h_{j,i-1}) \quad (2.39)$$

Die Aktivität  $h_k, i$  in den neuronalen Schichten  $i$  des Netzwerks der Aufgabe  $k$  errechnen sich also aus den über die jeweilige Gewichtsmatrix  $W_{k,i}$  gewichtete Aktivität  $h_{k,i-1}$  der vorhergehenden neuronalen Schicht derselben Aufgabe  $k$  sowie der über eine Adoptionsmatrix  $U$  transformierte Aktivität  $h_{j,i-1}$  der neuronalen Schicht des Netzwerks

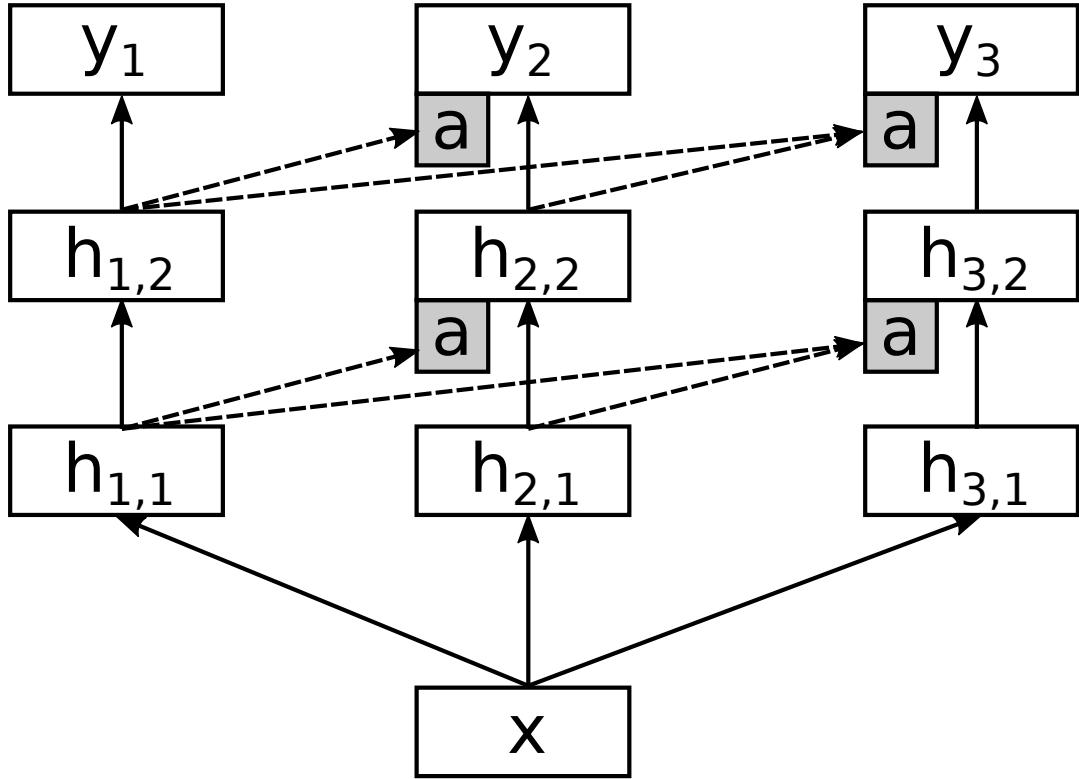


Abbildung 2.7: Skizze eines progressiven neuronalen Netzwerks nach dem Konzept aus [156]. Die einzelnen neuronalen Netzwerke optimieren jeweils eine Aufgabe  $A$ ,  $B$  oder  $C$ .

der vorangegangenen Aufgaben  $j$ . Die Parameter der Adoptionsmatrix  $U_{i,k;j}$ , welche die in diesem Modell essentielle Transferleistung erbringt, werden beim Training für die Aufgabe  $k$  bezüglich der Parameter der vorhergehenden Aufgaben optimiert. Abbildung 2.7 skizziert den Aufbau eines progressiven neuronalen Netzwerks für drei Aufgaben mit Eingabedaten  $X$  gleicher Dimensionalität.

Dieses Konzept kann erweitert werden, sodass auch Transferleistungen zwischen Aufgaben unterschiedlicher Eingabedatendimensionalität erlernt werden können [46]. In diesem Beitrag wird ein Reinforcement Learning System vorgestellt, welches Transferlernen anwendet, um die Unterschiede zwischen einem simulierten und einem realen Roboter mit Hilfe von progressiven neuronalen Netzwerken zu überwinden. Auf diese Weise kann Wissen aus einer Simulation auf einen physikalischen Roboterarm übertragen werden, was die notwendigen Lernzeiten derart reduziert, dass ein robotischer Manipulator bereits nach wenigen Stunden eine neue Aufgabe übernehmen kann [46]. Hierbei wird angenommen, dass nachfolgende Aufgaben konzeptionelle Erweiterungen vorhergehender Aufgaben darstellen. Eine nachfolgende Aufgabe verfügt demnach über Eingabedaten höherer Dimensionalität, wobei die Eingabedaten der nachfolgenden Aufgabe prinzipiell in die Dimensionalität der Eingabedaten der vorhergehenden Aufgabe überführt werden können, beispielsweise über die Skalierung eines hochauflösendes Farbbildes.

### 2.3.6 Gegnerische Umprogrammierung

Die gegnerische Umprogrammierung (E: *Adversarial Reprogramming*) ist ein Konzept des Transferlernens, in welchem das in einem maschinellen Lernmodell vorhandene Wissen zur Erfüllung der Aufgabe eines Angreifers eingesetzt werden soll. Hierbei werden die Eingabedaten und die Interpretation der Ausgabe eines neuronalen Netzwerks derart verändert, dass das künstliche neuronale Netzwerk eine andere als die ursprünglich vorgesehene Aufgabe erfüllt [108, 110, 92]. Zunächst wird eine Neuzuordnung eines Teils der Ausgabe  $y_{alt}$  des künstlichen neuronalen Netzwerks zum Ausgaberaum  $y_{neu}$  der neuen Aufgabe vorgenommen; beispielsweise von den gelernten Klassen *Hund*, *Katze*, *Maus* eines Klassifikators hin zu neuen, nicht gelernten Klassen *Apfel*, *Birne*, *Banane*. Das umzuprogrammierende Netzwerk soll nun sein Wissen über die Hunde, Katzen und Mäuse nutzen, um Bilder von Äpfeln, Birnen und Bananen zu klassifizieren. Hierzu wird ein gegnerisches Programm  $P$  optimiert, bestehend aus den eigentlichen Manipulationsparametern  $W$  sowie einer Maske  $M$  [108]:

$$P = \tanh(WM) \quad (2.40)$$

Die Matrix  $M$  trägt den Wert 1 dort, wo die Nutzdaten der gegnerischen Information liegen, ansonsten 0. Die Eingabebilder  $\tilde{x}$  der neuen Klassen werden nun mit  $P$  maskiert:

$$x_{neu} = \tilde{x} + P \quad (2.41)$$

Das über die Eingabedaten umprogrammierte aber ansonsten unveränderte künstliche neuronale Netzwerk soll nun, entsprechend der angestrebten Neuzuordnung  $x_{neu} \rightarrow y_{neu}$ , die für eine andere Aufgabe gelernten Parameter nutzen, um eine neue Aufgabe erfüllen. Abbildung 2.8 illustriert dies am Beispiel aus [108].

Je nach Kenntnis über die internen Parameter den umzuprogrammierenden Netzwerks bietet sich zur Optimierung des gegnerischen Programms ein White-box oder ein Black-box Ansatz an [110]. Während im White-box Ansatz ein gradientenbasiertes Verfahren angewandt werden kann, erlaubt eine Black-box Optimierung lediglich Ansätze des Verstärkungslernens oder evolutionärer Algorithmen [92].

Neuste Arbeiten nutzen die Methode der gegnerischen Umprogrammierung als Ansatz für Transferlernen von Sprachmodellen [162, 110]. Die Autoren der Ausarbeitung [110] schlagen verschiedene gegnerische Umprogrammierungsverfahren für Textklassifikationsmodelle vor, welches über die Neuzuordnung eines Vokabulars Wissen auf Grundlage von Symbolsequenzen in neue Aufgaben überträgt. Die Anwendbarkeit dieser Methodik wird für faltende neuronale Netzwerke und Long Short-Term Memory Netzwerke gezeigt [110]. In der Forschungsarbeit von [162] wird ein Ansatz vorgestellt, welcher auf der gegnerischen Umprogrammierung basiert und um aufgabenspezifische Worteinbettungen zu lernen, welche nach der Verknüpfung mit einem Eingabetext das Sprachmodell anweisen, eine neue Aufgabe zu lösen. Mit dieser Vorgehensweise übertreffen die Autoren von [162] die bisher existierenden Lösungen auf dem GLUE Benchmark [163].

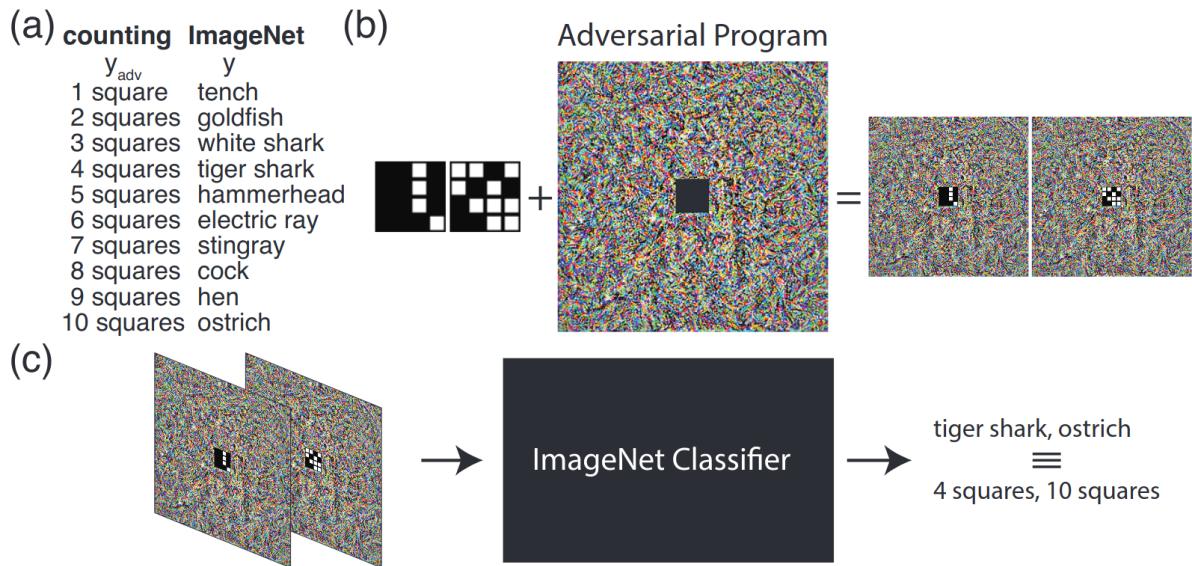


Abbildung 2.8: Eine Beispielanwendung der gegnerischen Umprogrammierung, entnommen aus [108]. Ein ImageNet Klassifikator soll umprogrammiert werden, sodass er anstelle der gelernten Klassifikation vielmehr die Anzahl weißer Vierecke auf einer schwarzen Fläche zählt. Zunächst wird eine Neuzuordnung zwischen gelernten Klassen und der Anzahl von weißen Vierecken vorgenommen (a). Das zu lernende gegnerische Programmbettet die Nutzdaten der neuen Aufgabe in eine, bezüglich der Neuzuordnung optimierte, Pixelmenge ein; den Angriffsvektor (b). Werden nun die Eingabedaten der neuen Aufgabe mitsamt des gegnerischen Programms präsentiert, so nennt das derart umprogrammierte Netzwerk die Klasse, welcher nach der Neuzuordnung der Anzahl der weißen Vierecke entspricht (c).

### 2.3.7 Weltmodelltransfer

Eine weiterer Ansatz des Transferlernens ist das Übertragen von bereits gelerntem Welt- oder Sprachmodellwissen auf eine unbekannte, neue Aufgabe; ein Weltmodell umfasst hierbei nicht nur Wissen über eine spezielle Aufgabe, sondern versucht, einen möglichst breiten Wissensschatz abzubilden [20, 164, 19, 153]. Dementsprechend eignen sich große Sprachmodelle, um Zusammenhänge auf konzeptioneller Ebene zu modellieren; von besonderem Nutzen sind Transformer-Netzwerke für allgemeine Zwecke (E: *General Purpose Transformer*, GPT) [165, 77]. Die Veröffentlichung von [77, 153] trägt das Sprachmodell GPT-2 bei, welches verschiedene natürliche Sprachverarbeitungsaufgaben mit löst und den bisherigen Stand der Technik übertrifft. Die Weiterentwicklung dieses Netzwerks, GPT-3, ist in der Lage, noch wesentlich kohärentere Sprache zu modellieren, welche sich von menschlicher Leistung nur noch schwer auseinanderhalten lässt [165]. Auch die Komposition von Musik kann als natürliche Sprachverarbeitungsaufgabe formuliert und mit einem entsprechenden Sprachmodell ermöglicht werden [166].

Unter der Annahme, dass genügend große Sprachmodelle in der Lage sind, konzeptionelle Zusammenhänge einer komplexen Welt abzubilden, kann ein Lerntransfer in eine neue Aufgabe als Frage der Inferenz eines Weltmodells formuliert werden. Für Aufgaben aus dem Bereich der natürlichen Sprachverarbeitung, wie etwa die Beantwortung von Fragen zu einem Text, die Zusammenfassung eines Textes oder die Kommunikation in einem Dialog, existieren Sprachmodelle, die inhärent zu einem sofortigen Lerntransfer (E: *Zero-Shot Transfer*) fähig sind [151, 152, 77]. Das Lernen einer Aufgabe  $A$  wird als Modellierung einer Wahrscheinlichkeitsverteilung  $P(\text{output}|\text{input}, A)$  formuliert, wobei  $\text{input}$  und  $\text{output}$  Symbolsequenzen sind, welche die Grundlage von natürlichen Sprachverarbeitungsaufgaben bilden. Nach dieser Formulierung entspricht das Finden der wahrscheinlichsten Antwort  $\text{output}$  der Abschätzung einer Wahrscheinlichkeitsverteilung  $p(x)$  auf Grundlage einer Symbolreihe  $\text{input} = \{s_1, \dots, s_n\}$  [167, 168, 77]:

$$p(x) = \prod_{i=1}^n p(s_i|s_1, \dots, s_n) \quad (2.42)$$

Zur Abschätzung dieser Wahrscheinlichkeitsverteilung nutzen GPT-Netzwerke ein Transformer-Netzwerk mit einem Aufmerksamkeitsmechanismus [169].

Anstelle von Transformer-Netzwerken zur natürlichen Sprachverarbeitung können Weltmodelle für Problemstellungen des visuellen Verstärkungslernens mit rekurrenten neuronalen Netzwerken umgesetzt werden, welche den Stand der Technik in schwierigen Umgebungen übertreffen [19]. Die Beobachtungen des markov'schen Entscheidungsprozesses, in Form von Farbbildern, werden zunächst mit einem Variational Autoencoder (VAE) in eine latente Variable  $z$  überführt [170, 59, 62]. Das Weltmodell  $M$  besteht aus einem rekurrenten neuronalen Netzwerk mit versteckten Aktivierungen  $h_t$ , welches ausgehend von einer Aktion  $a$  und einem latenten Zustand  $z_t$  versucht, den Folgezustand vorherzusagen. Anstelle eines einzigen Folgezustandes wird eine Wahrscheinlichkeitsverteilung  $P(z_{t+1}|z_t, a_t, h_t)$  als Mischung von Gaußverteilungen mit Hilfe von Mixture Density Networks (MDN) modelliert [171]. Das Transferlernen geschieht, indem ein kleines Netzwerk auf dem Weltmodell (VAE+MDN) aufbaut, um das dort enthaltene Wissen für das Lernen einer

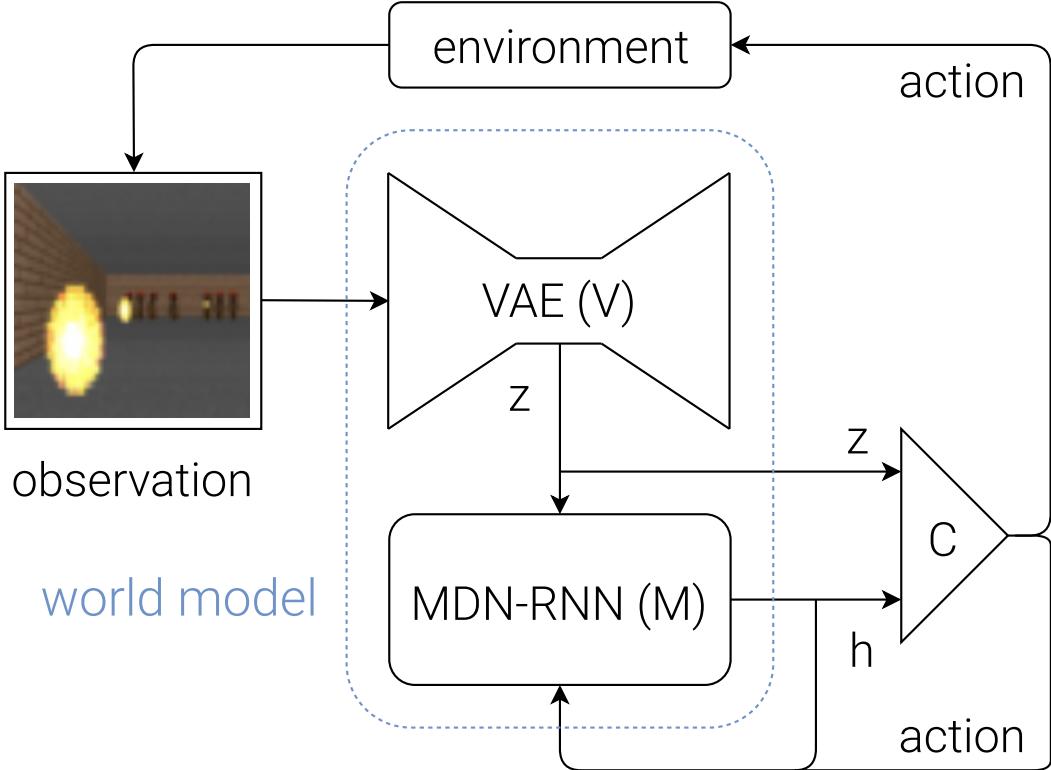


Abbildung 2.9: Darstellung des Weltmodellkonzepts für visuelles Verstärkungslernen, entnommen aus [19]. Zunächst werden die Beobachtungen in eine latente Variable  $z_t$  encodiert. Das Weltmodell  $M$  modelliert eine Wahrscheinlichkeitsverteilung für die Folgezustände  $z_{t+1}$ , ausgehend vom aktuellen Zustand  $z_t$  und einer Aktion  $a$ . Die Transferleistung erbringt der Controller  $C$ , welcher auf Grundlage der neuronalen Aktivierung im Weltmodell und der Beobachtung des aktuellen Zustands der Umgebung die optimale Strategie bezüglich für den jeweiligen Aktionsraum approximiert.

neue Aufgabe wiederzuverwenden. Auf Grundlage der versteckten Aktivierungen  $h_t$  und der latenten Zustandsvariable  $z_t$  kann ein Controllernetzwerk  $C$  genutzt werden, welches das Weltmodellwissen nutzt, um mit geringer Parameteranzahl eine optimale Strategie zu finden. Das Controllernetzwerk  $C$  kann aufgrund seiner geringen Parameterzahl auch über einen evolutionären Algorithmus entwickelt werden, [19] verwendet hier eine Evolutionsstrategie mit Kovarianzmatrixadaption auf 867 Parametern. Das Belohnungssignal des markovsch'en Entscheidungsprozesses wird bei diesem Ansatz nur dem Controllernetzwerk  $C$  zu Teil, das Encodernetzwerk  $V$  und das Weltmodell  $M$  arbeiten lediglich mit den Beobachtungen.

## 2.4 Verstärkungslernen

Entscheidungsprozesse stellen eine besonders interessante Klasse von Herausforderungen für die Methoden des maschinellen Lernens dar, da das zu lösende Lernproblem sich nicht nur auf einen statischen Datensatz begrenzt, sondern sich auf eine interaktive Umgebung erweitert, was ein realistischeres Modell natürlicher Lernprozesse darstellt [63]. Hierbei sollen optimale Entscheidungen auf Grundlage von Beobachtungen getroffen werden, was im Allgemeinen unter dem Konzept der markow'schen Entscheidungsprozesse zusammengefasst wird [63, 172]. Das Verstärkungslernen, auch als bestärkendes Lernen bezeichnet, erlaubt das operante Konditionieren eines Agenten zur Handlung nach einer optimalen Strategie für eine gegebene Umgebung mit einem Belohnungsschema [63, 173]. In Problemstellungen dieser Art soll ein handelnder Agent eine optimale Strategie mit Hilfe des Verstärkens gewünschter Verhaltensmuster (E: *Reinforcement Learning*, RL) erlernen. Das Konzept des Verstärkungslernens nimmt einen Agenten an, der mit einer Umgebung interagiert und für seine Handlungen Belohnungen oder Bestrafungen vermittelt bekommt, bezeichnet als positiver oder negativer *reward*. Über die erhaltene Belohnung oder Bestrafung kann der Agent einem jeden Zustand einen Zustandswert zuordnen. Dieser Zustandswert soll maximiert werden; die Umgebung soll also immer in den bestmöglichen Zustand überführt werden. Zur Findung einer optimalen Strategie ordnet ein Handelnder seinen Beobachtungen einen Zustandswert zu und wählt seine Aktionen so, dass die Umgebung mit hoher Wahrscheinlichkeit in den Folgezustand mit dem höchstmöglichen Zustandswert übergeht. Der Stand der Wissenschaft und Technik kennt eine Vielzahl unterschiedlicher Optimierungsverfahren für Strategien, welche maximale Zustandswerte in markov'schen Entscheidungsprozessen erreichen können, die je nach Beschaffenheit des Optimierungsproblems verschiedene Vorteile bringen [63, 174, 175, 176, 19, 65, 177]. Der gemeinsame Grundgedanke aller Verstärkungslernansätze besteht in der Anpassung der Bewertung von Zuständen als Abbildung erwarteter zukünftiger Belohnung [63].

Zur wissenschaftlichen Untersuchung der verschiedenen Ansätze des Verstärkungslernens eignen sich spieltheoretische Probleme, da sie aufgrund ihrer klar definierten Regelwerke eine Grundlage für reproduzierbare Experimente bieten [178, 179]. Die ersten Anfänge in der Erforschung selbstständig lernender Agenten untersuchten das Spiel Schach, mit vollständiger Information über den Spielzustand [180]. Frühe Forschungen auf dem Gebiet des visuellen Verstärkungslernens untersuchten bereits *Atari 2600* Spiele [176], *Doom* [181, 182, 183], auch *Unreal Tournament* [184] sowie *Minecraft* [185]; die Zahl der einschlägigen Publikationen dieser Domäne wächst schnell. Neuere Untersuchungen beschäftigen sich mit dem hochkomplexen Echtzeitstrategiespiel *Starcraft 2* [186, 187, 188, 189, 190].

### 2.4.1 Markov'scher Entscheidungsprozess

Zur Beschreibung der dem bestärkenden Lernen zugrundeliegenden mathematischen Zusammenhänge zwischen einem Handelnden und dessen Umgebung wird der markow'sche Entscheidungsprozess (E: *Markov decision process*, MDP) herangezogen [172]. Dem markov'schen Entscheidungsprozess liegt das stochastische Modell der Markowkette zu Grunde, nach welchem ein Zustand  $s$  aus einem Zustandsraum  $S$ , nach einer gewissen

Wahrscheinlichkeitsverteilung, in einen Folgezustand  $s \in S$  übergeht. Der markow'sche Entscheidungsprozess stellt eine Erweiterung dieser Idee dar, indem die Übergangswahrscheinlichkeiten in Abhängigkeit einer Aktion formuliert werden.

Da sich der Zustand der Umgebung nach der Handlung von deren Zustand vor der Handlung unterscheidet, ergibt sich ein Begriff schrittweiser zeitlicher Ordnung. Für jeden Zeitschritt  $t$  existiert ein Zustandsübergang  $s_t \rightarrow s_{t+1}$ , bis die Umgebung in einen Endzustand übergeht und sich nicht weiter verändert. In Zusammenhang mit jedem Zustandsübergang steht eine Aktion  $a_t$  aus dem Aktionsraum  $A$ , sowie eine, durch die Umgebung vermittelte, positive oder negative Belohnung  $r \in \mathbb{R}$ . Eine Relation  $P : S \times A$  nennt die Wahrscheinlichkeiten, mit denen ein Zustand in einen der Folgezustände  $s_{t+1} = P(s_t, a_t)$  übergeht, die Zuordnung  $r_t = R(s_t, a_t)$  die entsprechenden Belohnungen. Wären die Übergangswahrscheinlichkeiten  $P(s_t, a_t)$  bereits bekannt, so könnten die tatsächlichen Zustandswerte berechnet werden; diese Übergangswahrscheinlichkeiten sind dem Agenten jedoch unbekannt. Das Auswahlverfahren der Aktion  $a_t$  im Zustand  $s_t$  wird als Strategie  $a = \pi(s_t)$  (E: *policy*) bezeichnet.

An dieser Stelle muss in der Praxis eine Abwägung zwischen der Erkundung der Umgebung und dem Verfolgen der optimalen Strategie vorgenommen werden (E: *Exploration Exploitation Dilemma*) [191]. Zur Lösung dieses Dilemmas können, einem linear abfallenden Prozentwert  $\epsilon$  folgend, zufällige Aktionen durchgeführt werden oder die Entropie der Strategie in das Optimierungskriterium integriert werden [175, 192, 193].

Eine Strategie, welche auf Grundlage der Beobachtungen immer diejenigen Handlungen entscheidet, welche die Umgebung in Zustände mit höherem Zustandswerten versetzen, wird als optimale Strategie  $\pi^*$  bezeichnet. Um die notwendige Zuordnung eines Zustandswertes zu einer Beobachtung vorzunehmen, soll der Agent eine Wertefunktion  $V(s)$  lernen. Hierbei wird, von einem Anfangszustand  $s_0$  und der Befolgung der optimalen Strategie  $\pi^*$  ausgehend, die Zustandswertefunktion  $V$  als Erwartungswert über die Belohnungen  $r$  aller nachfolgenden Zustandsübergänge definiert:

$$V(s_0) = E\left\{\sum_{t=t_0}^{\infty} r(s_t) | s(t_0) = s_0\right\} \quad (2.43)$$

Nach dieser Formulierung wäre eine direkte Berechnung aller korrekten Zustandswerte denkbar, indem in jedem Zustand jede Aktion ausgeführt würde. In komplexen Umgebungen ist dieses Vorgehen so nicht durchführbar, da dieselbe Aktion im selben Zustand nicht zum erwarteten Folgezustand führen kann. Beispielsweise gibt es Hintergrundprozesse in Umgebungen, auf welche der Agent keinen Einfluss hat, oder aber auch andere Agenten, welche ihrerseits ebenfalls Einfluss nehmen. Da die Ausführung jeder möglichen Aktion in jedem Zustand unter diesen Annahmen nicht zielführend sein kann, wird zur Reduktion der Komplexität zu jedem Zeitschritt  $t$  nur die vielversprechendste Aktion ausgeführt:

$$a_t = \pi(s_t) = \arg \max_{a'} (R(s_t, a') + V(S(s_t, a')) \quad (2.44)$$

Der Wert des verlassenen Zustandes  $s_t$  wird anschließend anhand der tatsächlich erhaltenen Belohnung  $r_t$  neu abgeschätzt.

$$s_{t+1} = S(s_t, a_t) \quad (2.45)$$

$$r_t = R(s_t, a_t) \quad (2.46)$$

$$V(s_t) = r_t + V(s_{t+1}) \quad (2.47)$$

Dieser Ansatz entspricht dem Bellman'schen Optimierungsprinzip, welches besagt, dass sich optimale Teillösungen iterativ zu einer optimalen Gesamtlösung zusammenfügen lassen [194]. Das zeitschrittweise Approximieren wird auch als Zeitdifferenzlernen (E: *temporal difference learning, TD Learning*) bezeichnet [195].

Gibt es einen sehr großen Zustandsraum, wie beispielsweise einen Farbbildraum, oder gibt es verschiedene beziehungsweise keine Endzustände in der Umgebung, so kann die Wertefunktion leicht divergieren und den Lernprozess in Zusammenhanglosigkeit führen. Die Erwartung übermäßig hoher, beziehungsweise nicht existenter, Belohnungen bringt die Zuordnung der Zustandswerte besonders dann durcheinander, wenn aufgrund einer hohen Dimensionalität des Zustands, wie etwa bei einem Farbbild, keine zuverlässige Aussage mehr über die Relevanz der Bildelemente getroffen werden kann. Dies verhindert die Findung einer optimalen Strategie. Um einer Divergenz der Wertefunktion entgegen zu wirken, wird ein Diskontfaktor  $\gamma$  eingeführt, welcher die Wertefunktion, also die Summe der erwarteten Belohnungen, über die Zeit abwertet:

$$V_{t+1}(s_t) = R(s_t, a_t) + \gamma V_t(S(s_t, a_t)) \quad (2.48)$$

## 2.4.2 Q-Learning

Der einfache markow'sche Entscheidungsprozess trifft die Annahme, dass der Agent den Umgebungszustand vollumfänglich beobachten kann. Bei vielen Prozessoptimierungen besteht aber die Herausforderung eines nur teilweise beobachtbaren Umgebungszustandes. Beispielsweise zeigt die Bildausgabe eines Computerspiels in einer dreidimensionalen Welt nur eine zweidimensionale Abbildung des wahren Umgebungszustandes. Diese konzeptionelle Erweiterung des markov'schen Entscheidungsprozesses wird als teilweise beobachtbarer markow'scher Entscheidungsprozess (E: *partially observable Markov decision process, POMDP*) bezeichnet [196]. Die Lernverfahren für teilweise beobachtbare Umgebungen finden eine optimale Strategie unter der Annahme, dass nur ein gewisser Ausschnitt erfasst werden kann, wie etwa bei visuellen Computerspielen. Anstatt der gesamten Umgebungsinformation, welche beispielsweise die Positionen und Eigenschaften aller anderen Akteure und Gegenstände enthalten würde, besteht die Beobachtung hier nur aus einem Farbbild, welches den Ausschnitt der Umgebung im Sichtfeld zeigt. Um die tatsächlichen Werte der Zustände eines teilweise beobachtbaren markov'schen Entscheidungsprozesses abzuschätzen, kann die Wertefunktion  $V(s)$  zu einer Qualitätsfunktion  $Q(s, a)$  umformuliert werden. Die Qualitätsfunktion  $Q(s, a)$  nennt nicht den Wert eines Folgezustandes, sondern trifft stattdessen eine Aussage über den erwarteten Folgezustandswert für ein Tupel aus einem Zustand  $s$  und einer Aktion  $a$  [174]:

$$Q(s, a) = Q(s, a) + R(s, a) + \gamma \max_{a'} Q(s_{t+1}, a') \quad (2.49)$$

Die Abweichung der vorhergesagten zu den tatsächlichen  $Q$ -Werten dient einem künstlichen neuronalen Netzwerk als Optimierungskriterium für einen stochastischen Gradientenabstieg, beispielsweise über die Minimierung eines mittleren quadratischen Fehlers. Der Wert eines Zustandes  $s$  ergibt sich als höchster Wert der  $Q$ -Funktion, unter Betrachtung aller möglichen Aktionen  $a'$ :

$$V(s) = \max_{a'} Q(s, a') \quad (2.50)$$

Ebenso ergibt sich die optimale Aktion  $a^*$  als die Aktion mit dem höchsten  $Q$ -Wert im aktuellen Zustand:

$$a^*(s) = \arg \max_{a'} Q(s, a') \quad (2.51)$$

Eine binäre Variable  $e$  kann angeben, ob es sich beim Folgezustand um einen Endzustand handelt; in diesem Fall wird nur die Belohnung  $r$  und nicht der geschätzte Folgewert  $q_2$  zur Zielvorgabe  $q_{Ziel}$  addiert. Pseudocode 4 skizziert einen solchen Ablauf des Q-Lernschrittes:

---

#### **Algorithmus 4** Q-Lernschritt

---

```

1: procedure Q-LERNEN( $s_t, a, s_{t+1}, r, e$ )
2:   Parametervektor  $\Theta$ 
3:   Lernrate  $\eta$ 
4:   Diskontfaktor  $\gamma$ 
5:    $q_1 \leftarrow \text{Ausgabe}(\Theta|s_t)$ 
6:    $q_2 \leftarrow \text{Maximum}(\text{Ausgabe}(\Theta|s_{t+1}))$ 
7:    $q_{Ziel}(s_t, a) \leftarrow r + \gamma(1 - e)q_2$ 
8:    $L = \text{Optimierungskriterium}(q_1, q_{Ziel})$ 
9:    $\Delta\Theta = \frac{dL}{d\Theta}$ 
10:   $\Theta = \Theta + \eta\Delta\Theta$ 
11: end procedure

```

---

Mit der Einführung eines Erinnerungswiederholungsprozesses (E: *Action Replay Process*, ARP) zeigt sich die Stabilisierung der Strategiefindung [174]. Während des Lernprozesses werden die Erfahrungen, also die Zustandsübergänge mitsamt durchgeföhrter Handlungen und erhaltener Belohnungen, in einem Erinnerungsspeicher gesammelt und dem Optimierungsprozess, gewichtet nach ihrer Wichtigkeit, präsentiert. Die Wichtigkeiten der einzelnen Erinnerungen folgen aus den Ausmaßen der jeweiligen Abschätzungsfehler.

Zeigt sich bei der Neubewertung der Erinnerung ein hohes Fehlermaß, so birgt diese Erinnerung ein hohes Lernpotential [197]. Zur Optimierung der Entscheidungsparameter wird wiederholt eine Auswahl gesammelter Erfahrungen präsentiert und die Abweichung der  $Q$ -Werte minimiert. Ein genügend großer Erinnerungsspeicher soll dem Agenten ermöglichen, über genügend Lernzeit alle wichtigen Zustandsübergänge zu erfassen. Auf diese Weise werden diejenigen Erinnerungen mit höherer Wahrscheinlichkeit präsentiert, welche durch ihren höheren Zeitdifferenzfehler ein höheres Lernpotential enthalten [197]. Pseudocode 5 beschreibt das Hinzufügen einer Aufzeichnung in den priorisierten Erinnerungsspeicher und Pseudocode 6 das Aktualisieren des Prioritätsvektors nach [197]. Der kleinstmögliche Wert  $\epsilon > 0$  dient zur Vermeidung einer Nulldivision.

---

**Algorithmus 5** Hinzufügen zum priorisierten Erinnerungsspeicher

---

```

procedure HINZUFÜGEN(Aufzeichnung  $[s_t, s_{t+1}, a, r, e]$ )
    Erinnerungsliste  $M$ 
    Prioritätsvektor  $P$ 
    Wahrscheinlichkeitsverteilung  $p$ 
    Vertrauensfaktor  $\alpha$ 
     $i \leftarrow i + 1 \bmod |M|$ 
     $M[i] \leftarrow [s_t, s_{t+1}, a, r, e]$ 
     $P[i] \leftarrow \max(P)$ 
     $p_i \leftarrow \frac{P[i]^\alpha}{\sum_i P[i]^\alpha}$ 
end procedure

```

---

**Algorithmus 6** Aktualisieren des priorisierten Erinnerungsspeichers

---

```

procedure PRIORITÄTEN AKTUALISIEREN(Zeitdifferenzfehler  $\delta$ )
    Prioritätsvektor  $P$ 
    Kleinstmöglicher Wert  $\epsilon > 0$ 
    for  $d_i \in \delta$  do:
         $P[i] = d_i + \epsilon$ 
    end for
end procedure

```

---

### 2.4.3 Asynchrone Verfahren

Wenn Umgebungen verschiedene Startzustände aufweisen, bieten sich asynchrone Lernverfahren mit mehreren Agenten an, die in verschiedenen Startzuständen beginnen. Die Lernvorgänge der einzelnen Agenten werden in parallelen Prozessen berechnet, sodass die Umgebung in Form einer Breitensuche erkundet werden kann. Ein besonders gut untersuchtes Verfahren, welches unter Anderem das schwierige Problem der Navigation in einem dreidimensionalem Labyrinth löst, ist das asynchrone Aktor-Kritiker Vorteillernen (E: *Asynchronous Advantage Actor-Critic*, A3C). Der A3C Algorithmus erlaubt einen, in einer Umgebung verteilten, Lernprozess mit mehreren unabhängigen Agenten [175].

Der Lernprozess wird auf mehrere Instanzen einer Umgebung mit jeweils einem Agenten aufgeteilt. Die Lernprozesse approximieren additiv die optimale Strategie  $\pi^*$  nach dem REINFORCE Paradigma. REINFORCE ist hierbei das Akronym für „*Reward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility*“ und beschreibt eine Klasse von Algorithmen zum Verstärkungslernen, deren Gradientenabstiegsregeln sich wie folgt zusammenfassen lassen [177]:

$$\Delta w = \eta(r_t - b_t)e \quad (2.52)$$

Der Strategiegradient  $e$  entspricht der Ableitung der Strategie nach den Entscheidungsparametern und wird nach der Abweichung zwischen der tatsächlichen Belohnung  $r_t$  und der Belohnungsgrundlinie  $b_t$  gewichtet. Die Belohnungsgrundlinie  $b_t$  entspricht hierbei einer Zustandswertfunktion  $V(s_t)$  zum Lernschritt  $t$ .

Das Verfahren A3C nutzt akkumulierte Belohnungen  $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ , um Gradienten für den Entscheidungsparametervektor  $\Theta$  nach folgender Funktionsvorschrift aufzustellen [175]:

$$\nabla_{\Theta} \log \pi(a_t | s_t; \Theta) (R_t - b_t(s_t)) \quad (2.53)$$

Diese Formulierung besteht aus dem Kritikerterm  $(R_t - b_t(s_t))$ , welcher durch die Abweichung der erwarteten zur tatsächlichen Belohnung nach dem aktuellen Lernpotential gewichtet, so wie dem Aktorterm  $\pi(a_t | s_t; \Theta)$ , welcher eine Wahrscheinlichkeitsverteilung über den Aktionsraum darstellt. Ein Entropieterm  $H = -\sum \log(\pi(a_t))\pi(a_t)$  kann als Regularisierung in die Optimierungsfunktion mit aufgenommen werden, um den Agenten zur Erkundung anderer Aktionen zu motivieren.

Das A3C Verfahren verzichtet, wie andere *on policy* Lernverfahren auch, auf einen Erinnerungswiederholungsprozess. Stattdessen wird ein gemeinsamer Parametervektor  $\Theta$  verwendet. In regelmäßigen Abständen wird jedem Agenten eine aktualisierte Arbeitskopie  $\Theta'$  für einen individuellen Gradientenabstieg zur Verfügung gestellt. Wird der Endzustand der Umgebung oder ein bestimmtes Zeitlimit  $t_{max}$  erreicht, werden die lokalen Gradienten bezüglich der Parameter  $\Theta'$  berechnet und anschließend auf den gemeinsam geteilten Parametervektor  $\Theta$  addiert. Die resultierenden Gewichtsveränderungen werden nach dem sogenannten *HOGWILD!* Schema asynchron auf den gemeinsamen Parametervektor addiert. Dieses Verfahren konvergiert unter der Voraussetzung, dass die einzelnen Additionen nur kleine Teile des Parametervektors verändern [116].

# 3 Herausforderungen

Die Forschung in der angewandten Informatik besteht in der Anwendung der informatischen Methode auf Fragestellungen anderer wissenschaftlicher Bereiche, beispielsweise der Wirtschaftswissenschaft, der Mensch-Maschineinteraktion oder der Arbeitswissenschaft. Nach aktuellem Stand der Wissenschaft und Technik werden Herausforderungen aus anderen Disziplinen in der Regel als Optimierungsprobleme formuliert, welche meistens mit verschiedenen Architekturen künstlicher neuronaler Netzwerke angegangen werden. Die Lösungen dieser Probleme treten dann in Form von Anwendungen auf, also lauffähigen Programmen, welche die entsprechend gesuchten Verbindungen zwischen Ein- und Ausgabeinformation herstellen können, indem sie ein Modell der jeweiligen gesuchten Zusammenhänge aus einer Datenlage lernen und einem Anwender entsprechende Schnittstellen bereitstellen. Bei den Lernprozessen fallen Metadaten an, aus welchen systematisch Rückschlüsse auf die Güte eines Lernprozesses mit den jeweiligen Einstellungen und Datensätzen gewonnen werden können.

In dieser Arbeit werden verschiedene Anwendungen beschrieben, welche nach diesem Prinzip eine Unschärfe zwischen Sensordaten und formellen Konzepten auflösen, indem sie künstlich intelligente Abbildungen nutzen, welche Schlüsse aus einer Datenlage ziehen um den Anwender mit hilfreichen Erklärungen zu unterstützen. Bei den Anwendungen, welche im Rahmen dieser Arbeit entstanden, zeigt sich der Vorteil der Wiederverwendung von bereits erlerntem Wissen. Die Untersuchungen zur Bildklassifikation zeigen, dass einige Modelle sich besonders gut für einen Transferlernprozess eignen; jedoch bleibt hier die Frage offen, unter welchen Umständen ein Wissenstransfer von besonderem Nutzen ist [6]. Der Bedarf nach einer statistischen Auswertung der Metadaten dieser Transferlernprozesse führt zum Konzept des *Transfer Meta Learning* [5].

Die einzelnen Umsetzungen werden mit Konzepten der objektorientierten Programmierung erläutert. Durch die Anwendung der objektorientierten Paradigmen gestaltet es sich einfacher, einzelne Softwarekomponenten ohne Änderungen anderer Teile des Programms zu verändern oder für andere Programme wiederzuverwenden [198]. Ebenfalls eignet sich die Formulierung in Klassen, mitsamt derer Hierarchien und Beziehungen, beispielsweise in Darstellungen der vereinheitlichen Modellierungssprache (E: *Unified Modeling Language, UML*) wie etwa einem Sequenzdiagramm, zum einfachen Erfassen der relevanten Zusammenhänge [199]. Auch wenn nicht alle technischen Implementierungen bis ins Detail konsequent objektorientiert umgesetzt wurden, beispielsweise sind die Abläufe in den Lernprozessen der künstlichen neuronalen Netzwerke nach wie vor von prozeduraler Natur, so lassen sich die beteiligten Strukturen zur besseren Nachvollziehbarkeit und Wiederverwendbarkeit in Klassen abbilden.

### 3.1 Automatische Handelssysteme

Mit bestärkendem Lernen und tiefen künstlichen neuronalen Netzwerken können automatische Handelssysteme, im Rahmen dieser Herausforderung am Beispiel von Differenzkontrakten, umgesetzt werden [7, 8]. Unter einem automatischen Handelssystem wird allgemein ein Programm verstanden, welches allein auf Grundlage vorgegebener Parameter handeln kann, ohne dass ein Mensch eingreifen muss [200]. Wirtschaftliche Anwendungen der künstlichen Intelligenz im Allgemeinen und des maschinellen Lernens im Besonderen bieten neue Perspektiven, Möglichkeiten und Werkzeuge für die Modellierung von ökonomischen Vorgängen und Systemen [201].

Die Forschung im Bereich des automatisierten Handels mit Finanzanlagen deckt ein recht breites Spektrum mit verschiedenen Ansätzen ab, welche die Informatik und die Wirtschaftswissenschaften eng miteinander verbinden [202, 203, 204, 205]. In einer solchen Studie nutzen die Autoren allein langfristige Daten in Form von Trends über Phasen des Wachstums und des Rückgangs, ohne die ständigen Veränderungen am Markt in Betracht zu ziehen [202]. Andere Algorithmen, die bestimmte Beobachtungszeitpunkte berücksichtigen und auf dieser Grundlage unterschiedliche Vorhersagen und Schlussfolgerungen erstellen, nutzen Agenten, welche für einen bestimmten Zweck gebaut wurden und nach einem regelbasierten System handeln [204]. Moderne Börsenprognosen auf längeren Zeitskalen beziehen in der Regel externe Textinformationen aus Nachrichtenfeeds oder sozialen Medien ein [206, 207, 208]. Diese Ansätze erfordern jedoch ein gewisses Maß an zeitlicher Information über den aktuellen Markt. Unter Verwendung historischer Handelsdaten untersucht [209] ein LSTM-basiertes Kursvorhersagemodell für den chinesischen Aktienmarkt um steigende oder fallende Kurse auf täglichen Zeitskalen vorherzusagen und erreicht dabei Genauigkeiten zwischen 64,3% und 77,2%. Eine Deep-Learning Implementierung von [210] lernt mit LSTM-Netzwerken, Aktienkurse auf der Grundlage von kombinierten Nachrichtentexten und rohen Aktienkursinformationen vorherzusagen, was ebenfalls profitable Handelsanwendungen ermöglicht.

Die Nutzung solcher Systeme ermöglicht es Händlern, eine Strategie mit wesentlich höherer Frequenz abzuhandeln als es Menschen möglich sein kann; so ist der automatisierte Hochfrequenzhandel (*E: High Frequency Trading (HFT)*) für einen Großteil der Marktaktivität verantwortlich, fand in der Wissenschaft aber über einen langen Zeitraum verhältnismäßig wenig Beachtung [211, 212]. Bezüglich Algorithmen und Strategien für den Hochfrequenzhandel gibt es inzwischen eine Vielzahl von Anwendungen, darunter auch klassische Ansätze des maschinellen Lernens [212]. Was das Verstärkungslernen im Hochfrequenzhandel betrifft, so stellen [213] ein System zur Optimierung einer Strategie vor, welches Kauf-, Leerverkauf- und Halteentscheidungen auf der Grundlage von Finanz- und makroökonomischen Daten trifft. Es gibt auch Verstärkungslernansätze für den Hochfrequenzhandel an Devisenmärkten [214, 215, 216], jedoch keine Forschungsarbeiten zum maschinellen Hochfrequenzhandel mit Verstärkungslernen auf Differenzkontrakten.

### 3.1.1 Differenzkontrakte

Ein Differenzkontrakt (E: *Contract for Difference*, CfD) ist ein Total Return Swap Kreditderivat, welches es zwei Parteien ermöglicht, die Entwicklung eines Basiswerts gegen Zinszahlungen auszutauschen. In anderen Worten können Wirtschaftsakteure auf steigende oder fallende Kurse wetten und einen Gewinn erzielen, wenn die tatsächliche Kursentwicklung mit ihrer Wette übereinstimmt. Durch die Möglichkeit mit hoher Hebelwirkung zu wetten, können sowohl hohe Gewinne als auch hohe Verluste erzielt werden. Im Gegensatz zu anderen Derivaten, wie etwa Knock-out-Zertifikaten, Optionsscheinen oder Futures, können bei einem CfD der Stop-Loss- und der Take-Profit-Wert eigenhändig und unabhängig voneinander bestimmt werden. Durch die Festlegung eines Take-Profit- und eines Stop-Loss-Wertes wird die Transaktion automatisch geschlossen, wenn der Basiswert den entsprechenden Schwellenwert erreicht. Entspricht die Entwicklung nicht der Wette, sondern geht in die entgegengesetzte Richtung, so entsteht eine Schuld, welche zu zusätzlichem Finanzierungsbedarf führen kann. Eine Anleihe, auch Marge genannt, erfüllt den Zweck der Absicherung des Geschäfts. Wenn die zusätzlichen Finanzierungsverpflichtungen im Falle eines Ausfalls die Sicherheitshinterlegung übersteigen, können Händler innerhalb kürzester Zeit sehr hohe Verluste erleiden, wenn kein entsprechender Stop-Loss-Wert festgelegt wurde, welcher den Differenzkontrakt beendet.

Bezüglich rechtlicher Regulierungen besteht in den Vereinigten Staaten von Amerika derzeit ein Embargo für den Handel mit CfD. Nach einer Allgemeinverfügung der Bundesanstalt für Finanzdienstleistungsaufsicht darf ein Börsenmakler in Deutschland seinen Kunden solche spekulativen Optionen nur dann anbieten, wenn sie im Falle eines Ausfalls nicht zusätzlich haften, sondern nur ihre Sicherheitshinterlegung verlieren.

### 3.1.2 Aufbau der Simulation

Die Simulationsumgebung aus [7, 8] basiert auf historischen Handelsdaten, welche in entsprechenden Daten- und Kontrollstrukturen derart arrangiert werden, dass ein Agent eine Reihe von Marktwerten beobachten und eine Wette auf steigende oder fallende Kurse abschließen kann, welche mit fortschreitender Zeit in der simulierten Derivateumgebung zu einem Belohnungssignal ausgewertet wird. Implementiert wurde eine Marktlogik, die auf historischen Handelsdaten auf einer Tick-Zeitskala als Grundlage für einen teilweise beobachtbaren markov'schen Entscheidungsprozess (POMDP) arbeitet. Abbildung 3.1 skizziert die einem Lernschritt zugrundeliegende Sequenz und Pseudocode 7 beschreibt die Abhandlung der Marktsimulation auf Grundlage historischer Daten. Die Umgebung verarbeitet die Handelsaktionen der Agenten ohne Verzögerung, was zwar die analytische Untersuchung vereinfacht, aber den wichtigen Faktor der zeitlichen Verzögerung, wie er beispielsweise beim Hochfrequenzhandel eine essentielle Rolle spielt, außer Acht lässt. Als Zustand  $s$  stellt die simulierte Marktumgebung eine Folge von  $l$  Ticks bereit, ausgehend von einem beliebigen Punkt  $t$  in der Handelshistorie  $X$ . Die im Zustand enthaltenen Preise und Volumina werden um ihre Mittelwerte bereinigt:

$$s = X[t : t + l] - \bar{X}[t : t + l] \quad (3.1)$$

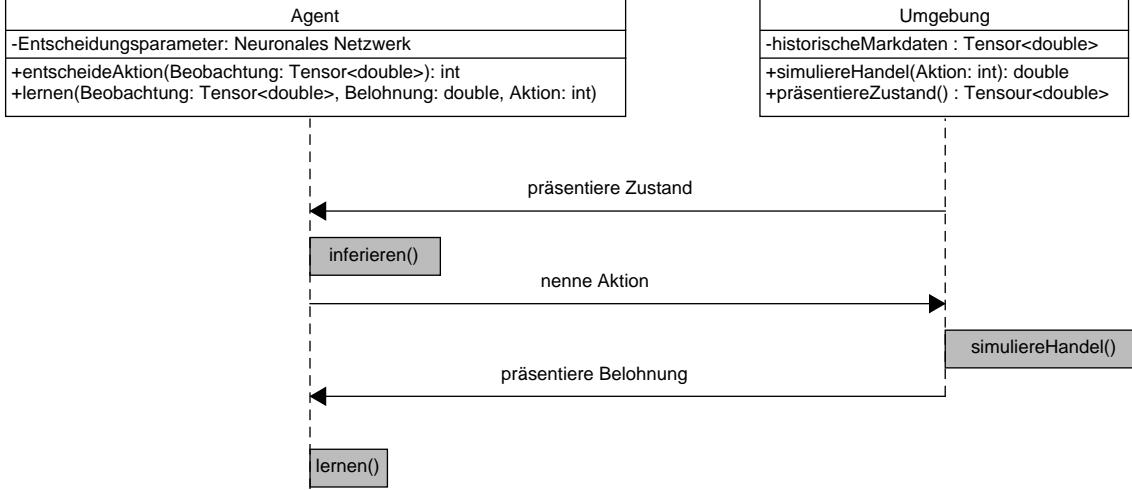


Abbildung 3.1: Ein Sequenzdiagramm zur Veranschaulichung des Ablaufs eines Lernschrittes in simulierten Marktumgebung. Die beteiligten Objekte, also der Agent und die Umgebung, kommunizieren über Zustand, Aktion und Belohnung.

Der Agent sieht also ein Eingabesignal fester Größe, welches aus einer Reihe von aufeinanderfolgenden Ticks besteht, die Preise für entsprechend ausgewählte Vermögenswerte darstellen. Jeder Tick umfasst den Zeitraum einer Sekunde und enthält sowohl den Brief- als auch den Geldkurs zum Sekundenschluss. Der Aktionsraum besteht also aus drei verschiedenen Aktionen: entweder kaufen, leerverkaufen oder abwarten:

$$A := \begin{cases} 0, & : \text{warten} \\ 1, & : \text{kaufen} \\ 2 & : \text{leerverkaufen} \end{cases} \quad (3.2)$$

Für jeden Zustand  $s$ , welchen die simulierte Marktumgebung präsentiert, wählt der Agent eine Aktion  $a \in A$ . Der Verstärkungslerner kann mit der Marktumgebung interagieren, indem er seine Aktionen  $a$  an die Umgebung bekannt gibt, welche nach Auswertung durch die Simulationslogik ein Belohnungssignal  $r$  in Form des finanziellen Gewinns oder Verlustes zurückgibt, welchen der Agent im Erwartungswert zu maximieren versucht. Wenn der Agent die Aktion  $a = 0$  wählt, um keinen Handel zu eröffnen, erhält er eine Belohnung von 0 und beobachtet den nächsten Zustand. Ein Auswertungsablauf in dieser Umgebung beginnt, wenn der Agent beschließt, einen Handel mit einer Aktion  $a \neq 0$  zu eröffnen. Je nach Handlung werden in der Simulationsumgebung Schleifen mit dem jeweiligen Abbruchkriterien aktiviert und solange ausgeführt, bis entweder ein Schwellwert überschritten oder das Ende der Aufzeichnungen erreicht ist; in letzterem Fall wird der Handel zum aktuellen Kurs ausgewertet und beendet. Nach Erreichen eines Take-Profit- oder den Stop-Loss-Wertes in Form von Gewinn- oder Verlustprozentsätzen von einem

festzulegenden Prozentwert, in den Ausarbeitungen [7, 8] willkürlich mit zehn Prozent beziffert, beendet die Umgebung die Auswertungsprozedur und gibt dem Agenten ein Belohnungssignal auf der Grundlage des erzielten Ergebnisses sowie den neuen Zustand der Marktumgebung als Beobachtung zurück.

Der Ansatz bietet die Möglichkeit, sowohl den Beobachtungszeitraum als auch die gehandelten Werte und Abbruchkriterien zu variieren. Die Beobachtung besteht aus einer Matrix, welche für jeden Sekudentick sowohl die entsprechenden Handelsvolumina als auch die Kauf- und Verkaufspreise der beobachteten Vermögenswerte zu diesem Zeitpunkt enthält. Auf der Grundlage dieser Beobachtungsprimitive kann ein Beobachtungszeitraum bestimmt werden; eine längere Beobachtung, beispielsweise über zehn Minuten, würde bedeuten, dass der Agent die letzten 600 Ticks der simulierten Derivatemarktumgebung als Eingabeinformation für seine Strategie zu sehen bekommt. Um zu untersuchen, ob und wenn ja wie sich der Beobachtungszeitraum und die Beobachtung der Korrelation zwischen verschiedenen Basiswerten auf den Lernerfolg des Handelsagenten auswirkt, kann die simulierte Derivatemarktumgebung eine prinzipiell beliebig lange Sequenz von Ticks über beliebig ausgewählte Basiswerte präsentieren. So kann durch Parameter der Simulation gesteuert werden, wie viel Information der Agenten bekommt, bevor er handeln kann. An diesem Punkt erscheint es als vernünftig anzunehmen, dass ein Akteur mit einem längeren Beobachtungszeitraum auch ein breiteres Verständnis des Marktes erlangen würde, was sich in höheren Gewinnen widerspiegeln würde; doch die in den folgenden Sektionen diskutierten Ergebnisse zeigen, dass kurzfristige Beobachtungszeiten sich bei bestimmten Werten als gewinnbringender erweisen.

### 3.1.3 Künstliche neuronale Architekturen

Ein bestärkend lernender Agent schätzt Zusammenhänge zwischen Beobachtung, Aktion und Belohnungssignal mit Hilfe eines künstlichen neuronalen Netzwerks. Da das Auftragsbuch des Handelsplatzes dem Agenten nicht zur Verfügung steht, präsentiert sich das Problem des Handelns mit Differenzkontrakten als ein teilweise beobachtbarer markov'scher Entscheidungsprozesses. In diesem Falle kann die Eingabeschicht eines künstlichen neuronalen Netzwerks nur für die beobachtbare Information modelliert werden; dies sind der Briefkurs  $p_{ask}$ , der Geldkurs  $p_{bid}$  und die entsprechenden Handelsvolumina  $v_{ask}, v_{bid}$ . Diese, über die Eingabeschicht präsentierten, Zustände  $s$ , welche in einem Zeitrahmen der Länge  $l$  vorliegen, ermöglichen es dem Agenten, eine Entscheidung über den Kauf oder den Leerverkauf eines Vermögenswerts zu treffen. In der Ausgabeschicht schätzt die neuronale Architektur die Q-Werte  $Q(s, a)$  für jede Aktion im Aktionsraum  $ainA$ . Um diese Werte zu approximieren, verwenden die umgesetzten Architekturen eine Ausgabeschicht mit  $|A|$  linear aktivierte Neuronen. Untersucht werden ein einfaches mehrschichtiges Perzeptron, eine rekurrente LSTM-Architektur zur Vorhersage auf Grundlage einzelner Kursverläufe sowie ein rekurrentes Faltungsnetzwerk, welches Korrelationen  $n$  verschiedener Basiswerte erfassen kann. Alle Architekturen haben prinzipiell den gleichen Aufbau von Eingabe- und Ausgabeschichten, aber unterschiedlich gestaltete Zwischenschichten; lediglich das rekurrente Faltungsnetzwerk erweitert die Eingabeschicht um eine weitere Dimension für die  $n$  zu korrelierenden Basiswerte.

---

**Algorithmus 7** Simulierte Marktlogik

---

```
1: procedure BELOHNUNGSFUNKTION  $R(\text{Aktion } a)$ 
2:   Take-Profit, Stop-Loss Faktoren  $d_{profit} \geq 0, d_{loss} \leq 0$ 
3:   Historische Handelsdaten  $X$ 
4:   Sequenzlänge  $l$ 
5:    $r \leftarrow 0$ 
6:   if  $a = 0$  then
7:      $t \leftarrow t+1$ 
8:   end if
9:   if  $a = 1$  then
10:     $\text{Eröffnungspreis} \leftarrow X[t + l][\text{Geldkurs}]$ 
11:    while  $t \leq \text{len}(X) - l \wedge \text{take-profit} \geq r \geq \text{stop-loss}$  do
12:       $t \leftarrow t+1$ 
13:       $\text{Preis} \leftarrow X[t + l][\text{Geldkurs}]$ 
14:       $r = \frac{\text{Preis} - \text{Eröffnungspreis}}{\text{Eröffnungspreis}}$ 
15:    end while
16:   end if
17:   if  $a = 2$  then
18:      $\text{Eröffnungspreis} \leftarrow X[t + l][\text{Briefkurs}]$ 
19:     while  $t \leq \text{len}(X) - l \wedge \text{take-profit} \geq r \geq \text{stop-loss}$  do
20:        $t \leftarrow t+1$ 
21:        $\text{Preis} \leftarrow X[t + l][\text{Briefkurs}]$ 
22:        $r = \frac{\text{Eröffnungspreis} - \text{Preis}}{\text{Eröffnungspreis}}$ 
23:     end while
24:   end if
25:
26:    $s \leftarrow X[t : t + l] - \text{mean}(X[t : t + l])$ 
27:   return  $s, r, t$ 
28: end procedure
```

---

## Mehrschichtiges Perzeptron

Das mehrschichtige Perzeptron besteht aus drei vollverbundenen vorwärtsgerichteten Schichten, wie in Abbildung 3.2 dargestellt. Die ersten beiden Schichten bestehen aus 500 gleichrichtenden Lineareinheiten mit einem Biaswert von 0, 1. Die Initialisierung der synaptischen Gewichte erfolgt nach der He-Gewichtsinitialisierung ([217]) mit einer gleichmäßigen Verteilung. Um zu Vergleichszwecken eine annähernd gleiche Parameterzahl wie in der rekurrenten LSTM-Architektur zu erhalten, wird eine dritte Schicht mit 180 gleichrichtenden linearen Einheiten hinzugefügt, ebenfalls mit einem Bias von 0, 1. Bei einer Eingabesequenzlänge von  $l = 500$  und einer Aktionsraumgröße von  $|A| = 3$  verfügt diese Architektur über insgesamt 840.540 lernbare Parameter.

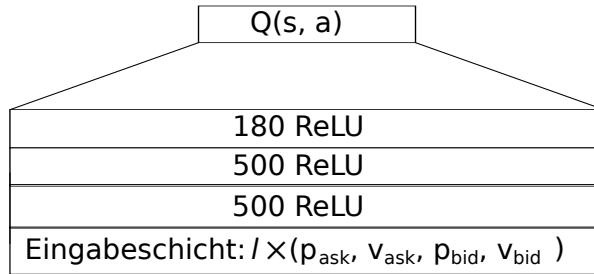


Abbildung 3.2: Das mehrschichtige Perzeptron, entnommen aus [7].

## Long Short-Term Memory

Als einfache rekurrente neuronale Architektur wird ein LSTM-Netzwerk mit Vergessensgattern in der Ausgestaltung untersucht, wie es von [125] vorgeschlagen wurde. Gleich dem mehrschichtigen Perzeptron besteht die Eingabeschicht aus einer Folge von Handelsdaten gleicher Sequenzlänge während des Trainings und des Tests. Die Ausgabeschicht approximiert lineare  $Q$ -Werte unter Verwendung des versteckten Zustands der LSTM-Schicht. Die verborgene LSTM-Schicht besteht aus einer einzigen rekurrenten Schicht mit 100 gleichrichtenden linearen Einheiten, wie in Abbildung 3.3 dargestellt. Die Initialisierung der Gewichte erfolgt nach der Normalverteilung. Bei einer festen Eingabelänge von  $l = 500$  und einer Aktionsraumgröße von  $|A| = 3$  verfügt dieses LSTM-Netzwerk insgesamt 840.300 lernbare Parameter.

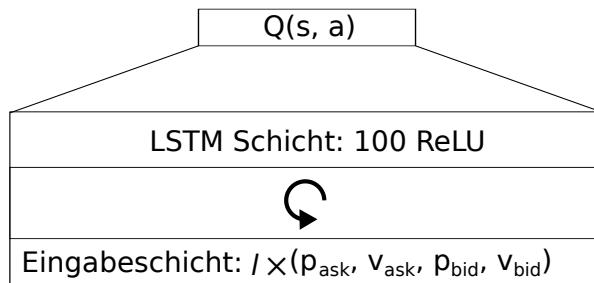


Abbildung 3.3: Das rekurrente neuronale Netzwerk, entnommen aus [7].

## Rekurrentes Faltungsnetzwerk

Abbildung 3.4 veranschaulicht die neuronale Architektur des rekurrenten Faltungsnetzwerks aus den Untersuchungen zu Beobachtungszeiteffekten von [8]. Hierbei besteht die Eingabeschicht für eine Anzahl von beobachteten Basiswerten  $n$  über eine Zeitspanne der Länge  $l$  aus  $(l \times n \times 4)$  Eingabeneuronen, wobei die Zahl vier die Anzahl der Preis- und Volumensmerkmale wiederspiegelt. Für die erste Faltungsschicht wird die Anzahl der Filter  $f$  so gewählt, dass die resultierende Aktivierungsform die Dimensionalitätsbedingungen der nachfolgenden Faltungsschicht erfüllt. Nach dem Faltungsteil, bestehend aus insgesamt vier faltenden Schichten, folgt eine LSTM-Schicht, welche aus 100 rekurrenten Neuronen mit gleichgerichteter linearer Aktivierung besteht. Dieses rekurrente faltende neuronale Netzwerk implementiert einen Aufmerksamkeitsmechanismus über eine Bandbreite von  $n$  beobachteten Basiswerten. Diese Form der Verwendung ermöglicht es, Korrelationen zwischen einzelnen Vermögenswerten zu erfassen. Hierbei wird nicht von Nachbarschaften zwischen den Vermögenswerten ausgegangen, sondern von zeitlichen Korrelationen. So kann beispielsweise ein Anstieg des Goldpreises erwartet werden, wenn ein Rückgang des Indexwertes beobachtet wird.

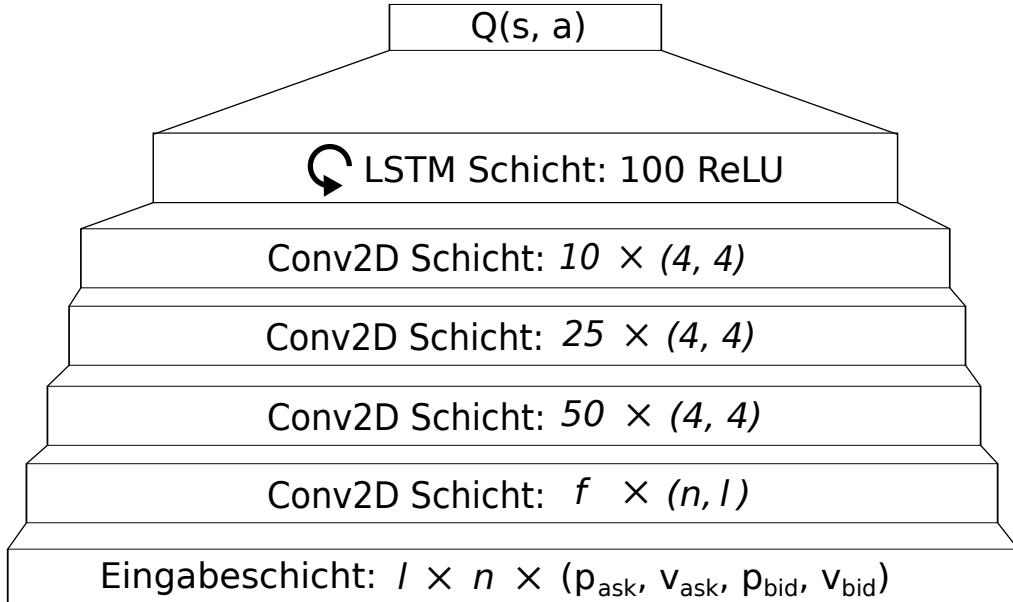


Abbildung 3.4: Das rekurrente Faltungsnetzwerk, entnommen aus [8].

### 3.1.4 Methode

Diese automatische Handelsmethode mit spekulativen Vermögenswerten beruht auf einer statistischen Risikominimierung, welche einen Derivatemarkt als einen teilweise beobachtbaren markov'schen Entscheidungsprozess für Verstärklernner simuliert, welche eine risikominimale Strategie abzuschätzen lernen. Zur Bestimmung des abzuschätzenden Belohnungssignals bewertet die Simulation die durchgeführten Handelsaktionen anhand

historischer Marktdaten und gibt den finanziellen Gewinn oder Verlust zurück. Ein bestärkend lernender Agent versucht, mittels Q-Learning, eine optimale Strategie zu finden, welche den erwarteten Gewinn maximiert und in gleichem Maße den Verlust minimiert. Zur Annäherung an eine optimale Strategie verfügen die untersuchten Agenten über unterschiedliche künstliche neuronale Netzwerke als Parametervektoren. Sowohl ein einfaches vorwärtsgerichtetes mehrschichtiges Perzepron, ein rekurrentes Long Short-Term Memory Netzwerk mit künstlichen Vergessenstoren wie auch ein rekurrentes Faltungsnetzwerk werden in der Marktumgebung trainiert und ausgewertet. Als Lernverfahren wird ein Q-Learning-Ansatz mit priorisiertem Erinnerungswiederholungsspeicher umgesetzt.

## Lernverfahren

Ein Q-Learning-Ansatz, inspiriert von [218], mit priorisiertem Erfahrungslernen, wie von [219] beschrieben, und den verschiedenen bereits beschriebenen Architekturen tiefer künstlicher neuronaler Netzwerke ermöglicht es, optimale Handelsstrategien maschinell aus einer simulierten CfD-Marktumgebung mit historischen Daten zu lernen. Zur Umsetzung der Modelle und des Lernverfahrens wurden die Programmiersprache Python mit den Bibliotheken Theano und Lasagne verwendet [220, 221].

In der Simulation führt jede Aktion  $a \in A$  zu einem normalisierten Belohnungssignal  $r \in \{-0.1...0.1\}$ , welche das Optimierungskriterium darstellt und als solches das Optimierungsverfahren anleitet. Der Agent schätzt mittels des neuronalen Netzwerks die Belohnung der Umgebung für seine Aktion und versucht auf diese Weise, die profitabelste Handelsaktion für eine bestimmte Marktbeobachtung zu finden. Als Optimierungsverfahren wird AdaGrad, wie von [115] beschrieben, genutzt, um als synaptische Gewichtsaktualisierungsregel zu einem optimalen Parametervektor zu finden.

Für jede aufgezeichnete Handlung im Erinnerungsspeicher ist der Anfangszustand  $s_1$ , die gewählte Aktion  $a$ , der Folgezustand  $s_2$ , die erreichte Belohnung  $r$  und eine Variable  $e$  bekannt, welche angibt, ob die Aktion in einem geschlossenen Handel endet. Dies wäre beispielsweise nicht der Fall, wenn die Episode der Simulation endet, bevor die entsprechenden Schwellwerte erreicht wurden. Aus diesem Grund werden zur Beschreibung des Lernvorgangs einzelne Lernschritte und nicht durchlaufene Episoden der Simulation betrachtet. In einem Lernvorgang führt der Agent insgesamt 250.000 Lernschritte durch. Für jeden Lernschritt werden  $b$  Q-Werte für eine Menge von  $b$  unabhängigen Erfahrungen  $(s_1, a, s_2, r, e)$  aus dem priorisierten Wiedergabespeicher abgeschätzt. Anschließend wird die AdaGrad-Gewichtsaktualisierung auf die Parameter des künstlichen neuronalen Netzwerks angewandt, welche auf der Differenz zwischen den vorhergesagten und den tatsächlichen Q-Werten basiert. Zur Lösung des Exploration-Exploitation Dilemmas wird eine  $\epsilon - greedy$  Strategie umgesetzt, nach welcher die ersten zehn Prozent der Lernzeit nur zufällige Aktionen ausgeführt werden und in den letzten 40 Prozent der Lernzeit nur die Strategie befolgt wird; in der Zwischenzeit fällt  $\epsilon$  linear. Der Pseudocode 8 beschreibt den Ablauf des Q-Lernverfahrens in der simulierten Derivatemarktumgebung aus [7, 8].

---

**Algorithmus 8** Training in der Marktsimulation

---

```
1: procedure TRAINING
2:   Historische Marktdaten  $X$ 
3:   Priorisierter Erinnerungsspeicher  $M$ 
4:   Eingabesequenzlänge  $l$ 
5:   Stapelgröße  $b$ 
6:   Strategie  $\pi(s|\Theta)$ 
7:   while Lernschritt < Anzahl Lernschritte do
8:      $t \leftarrow$  Zufälliger Zeitpunkt in der Markthistorie
9:     while  $t \leq \text{len}(X) - l$  do
10:       $s_1 \leftarrow X[t : t + l] - \bar{X}[t : t + l]$ 
11:       $a \leftarrow \pi(s_1)$ 
12:       $(s_2, r, t, e) \leftarrow R(a)$ 
13:       $M \leftarrow \text{Hinzufügen}(s_1, s_2, a, r, e)$ 
14:      if  $|M| \geq b$  then
15:        Stapel  $\leftarrow$  Aufzeichnungen aus  $M$ 
16:         $Q - \text{Lernen}(\text{Stapel})$ 
17:         $M \leftarrow \text{Prioritäten aktualisieren}(\text{Stapel})$ 
18:        Lernschritt  $\leftarrow$  Lernschritt + 1
19:      end if
20:    end while
21:  end while
22: end procedure
```

---

## Historische Marktdaten

Die Grundlage des simulierten Derivatemarktes bilden historische Handelsinformationen. Die Abbruchkriterien der Schleifen in der Simulation bedienen sich der aufgezeichneten Marktdaten, um die Belohnungsfunktion zu berechnen. Der historische Marktdatensatz gliedert sich in zwei Teile; ein Datensatz für eine Machbarkeitstudie auf einem einzelnen Indexwert [7] und ein Datensatz für eine tiefergehende Untersuchung der Beobachtungseffekte auf mehreren Basiswerten [8]. Hierzu wurden im Juli 2019 pro Sekunde fünf Werte des deutschen Aktienindexes für eine Machbarkeitsstudie aufgezeichnet. Von den fünf pro Sekunde aufgezeichneten Datenpunkten wurden unverändert aufeinanderfolgende Datenpunkte gelöscht. Nachdem auf diese Weise einen Monat lang aufgezeichnet wurde, wurden die Daten in einen Datensatz für die Trainingssimulation und einen Datensatz für eine Testsimulation aufgeteilt. Daraus ergibt sich eine Datenbasis von etwa drei Millionen Ticks für die Trainingsumgebung und etwa einer halben Million Ticks für das Testverfahren, anhand dessen die Qualität der Modelle bewertet wird.

Für weiterführende Untersuchungen wurden  $n = 43$  Basiswerte im Sekundentakt vom 25. Mai 2020 bis zum 27. Juli 2020 während derer gemeinsamen Handelszeiten zwischen 11:00 und 18:00 Uhr aufgezeichnet. In dieser Marktphase wurde eine allgemeine Erholung vom Börsencrash im März 2020 beobachtet, Abbildung 3.5 zeigt beispielsweise die Tagesschlusskurse des Indexwertes *US500* während des Aufzeichnungszeitraums. Die für die ausgewählten Vermögenswerte unterscheiden sich stark in der Art der Finanzanlagen, die sie darstellen; der Basiswert *US500* bezieht sich auf den Standard & Poor's Aktienindex, der die 500 größten börsennotierten Unternehmen in den Vereinigten Staaten von Amerika umfasst, während sich die Basiswerte *GOLD* und *OIL* auf die jeweiligen realen Rohstoffe beziehen. Der Basiswert *EURUSD* stammt von einem Währungsmarkt zwischen dem Euro und dem US-Dollar. Um eine Überanpassung zu vermeiden wurden die Daten in einen Trainings- und einen Testdatensatz aufgeteilt, so dass jeder dritte Tag dem Testdatensatz und nicht dem Trainingsdatensatz hinzugefügt wird.

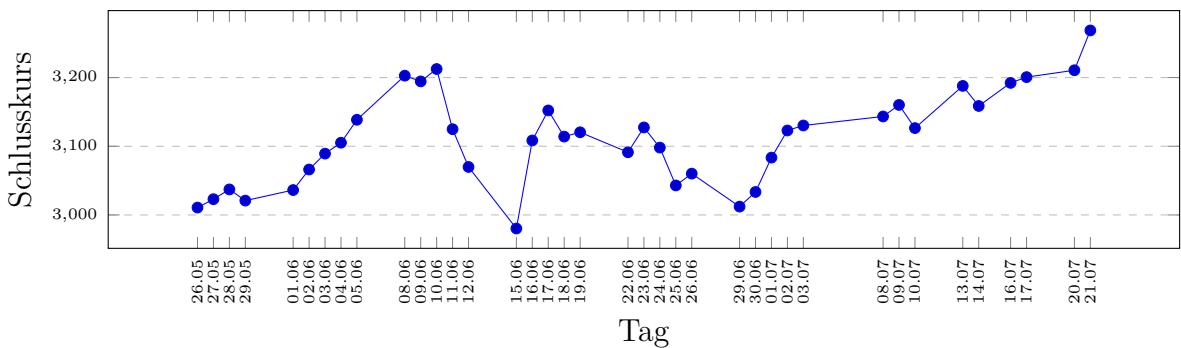


Abbildung 3.5: Verlauf des Indexwertes *US500* während Aufzeichnung der historischen Marktdaten im Jahr 2020 für die simulierte Derivatemarktumgebung, entnommen aus [8].

### 3.1.5 Auswertung

Zunächst wurde von [7] eine Machbarkeitsstudie mit den einfachen Modellen allein auf den Werten des deutschen Aktienindexes durchgeführt. In diesem Zuge wurde auch ein Echtzeittest unter realen Marktbedingungen durchgeführt, um die Anwendbarkeit des Ansatzes in der realen Welt zu bewerten. Weiterführende Auswertungen von [8] befassen sich mit den Auswirkungen der Korrelation verschiedener Basiswerte in verschiedenen Beobachtungszeiträumen.

#### Machbarkeitsstudie

Um zu evaluieren, ob dieser Ansatz funktioniert, werden die Aufzeichnungen des *DE30*-Index, welcher den deutschen Aktienindex abbildet, aus dem Juli 2019 genutzt. Zur Bestimmung guter Trainingsparameter wird eine Gittersuche in einem Raum aus Stapelgröße  $b \in \{10, 50, 100\}$ , Lernrate  $\eta \in \{10^{-4}, 10^{-5}, 10^{-6}\}$  und Eingabesequenzlänge  $l \in \{50, 100, 250\}$  durchgeführt. Durch den Vergleich der Kapitalentwicklung nach 1.000 Testtransaktionen kann eine optimale Parameterkonfiguration bestimmt werden. Für das mehrschichtige Perzeptron findet sich diese optimale Trainingsparameterkonfiguration in  $(b = 100, l = 50, \eta = 10^{-5})$ ; für das einschichtige LSTM-Netzwerk in  $(b = 10, l = 50, \eta = 10^{-4})$ . Zum Vergleich mit einer Baseline wurde, zusätzlich zu den Untersuchungen von [7], auch ein lineares Modell zur Annäherung der Q-Werte aus den Handelsdaten ausgewertet; hierfür wurde ebenfalls eine optimale Lernkonfiguration in den Parametern  $(b = 10, l = 50, \eta = 10^{-4})$  gefunden. Um nun die Modelle zu bewerten werden Tests in der Simulation mit den bisher ungesehenen Marktdaten durchgeführt. Hierbei führt der Agent keine Aktion aus wenn für eine optimale Aktion  $a$  die erwartete Belohnung  $Q(s, a) < 0$  ist, da der Agent einen Gewinn und nicht etwa einen minimalen Verlust umsetzen soll. Dadurch verlängert sich die Zeit zwischen den Handelsaktivitäten zugunsten einer höheren Erfolgswahrscheinlichkeit. Jedes mehrschichtige Perzeptron und LSTM-Netzwerk führt insgesamt 1.000 Testgeschäfte auf den ungesehenen Daten durch, jede Testreihe beginnt mit einem Eigenkapital von 1.000\$.

Aus der Verteilung der Aktionen in der Abbildung 3.6 ist ersichtlich, dass sowohl der MLP- als auch der LSTM-Agent häufiger Leerverkaufspositionen als Kaufpositionen eröffnen. Das lineare Modell trifft häufiger Kaufentscheidungen und führt zwar zu einer positiven Kapitalentwicklung bei geringeren Trainingszeiten, unterliegt jedoch den Erträgen der tiefen neuronalen Architekturen. Um einen Handel auszuführen, beobachtet das mehrschichtige Perzeptron im Durchschnitt 2429 Ticks, während das LSTM-Netzwerk auf 4.654 Beobachtungen wartet, bevor es einen Handel ausführt. Während das LSTM-Netzwerk zum Abwarten neigt und häufiger die Aktion  $a = 0$  wählt, trifft das MLP also schneller Entscheidungen. In der Eigenkapitalentwicklung aus Abbildung 3.6 kann ein Anstieg für alle drei Modelle gesehen werden; das LSTM-Netzwerk hat das Lernproblem am Besten gelöst, vermutlich aufgrund eines konzeptionellen Vorteils durch die immanente Behandlung von Sequenzen. Dies bestätigt sich bei der Betrachtung der in Abbildung 3.6 dargestellten Unterschiede in der Gewinnverteilung.

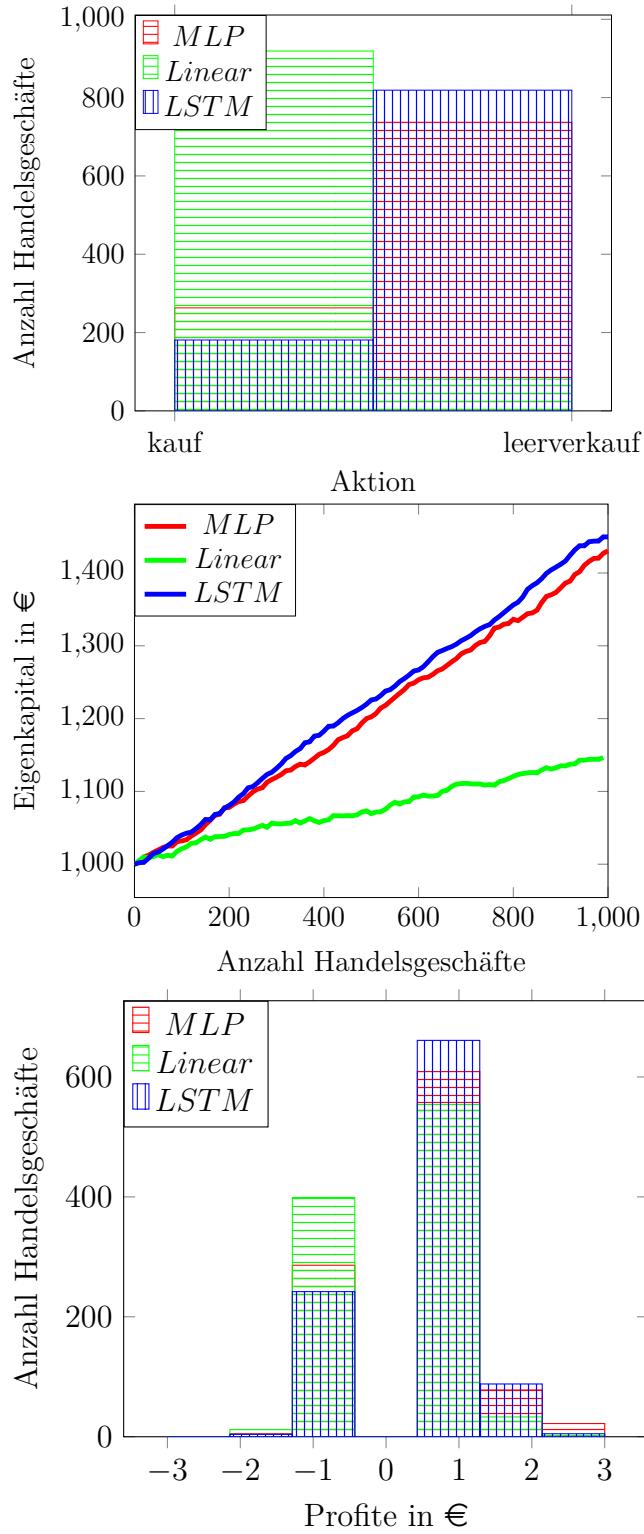


Abbildung 3.6: Auswertungsergebnisse aus der Machbarkeitsstudie, basierend auf den Untersuchungen aus [7]. Oben: Aktionsverteilung. Mitte: Kapitalentwicklung. Unten: Profitverteilung.

## Echtzeittest

Um die Ergebnisse der Machbarkeitsstudie unter realen Marktbedingungen zu überprüfen, wurde ein Demokonto bei einem CfD-Anbieter eingerichtet, welcher eine entsprechende API für maschinelles Handeln zur Verfügung stellt. Als Basiswert wurde der Differenzkontrakt *DE30* gewählt, welcher auf dem deutschen Aktienindex basiert. Die Handelsstrategie wird von einer LSTM-Architektur vorgeschlagen, da das LSTM-Netzwerk in der Machbarkeitsstudie bessere Ergebnisse erzielt hat als ein vergleichbares mehrschichtiges Perzepron. Zunächst wird versucht, das beste gefundene Modell ohne weitere Veränderungen anzuwenden; jedoch zeigten sich Latenzprobleme, da der Agent in der Simulation im Bereich von Sekundenbruchteilen gelernt hat, hier jedoch mit Verzögerungen in höheren Abständen handeln muss. Zum Einen veranlasst dies den Agenten, Entscheidungen auf der Grundlage einer längst vergangenen Beobachtung zu treffen, zum Anderen verzögert sich die Durchführung der Handelsorder, so dass sich die Marktsituation bereits geändert hat und die Festlegung der Schwellwerte nicht mehr passt. Als ersten Ansatz um diese Latenzprobleme zu lösen, wurde eine LSTM-Architektur mit einer zusätzlichen Schicht von 250 LSTM-Zellen entwickelt, wie in Abbildung 3.7 dargestellt. Auch wurde der Aktionsraum auf  $A = |10|$  vergrößert, indem eine Schwellwertfunktion  $d_{profit}(a)$  eingeführt wird, welche jeder Aktion  $a$  einen bestimmten Schwellwert  $\delta$  zuordnet. Anstelle eines multiplikativen Faktors kann der Agent im Echtzeittest über seine Aktion einen Schwellwert wählen, um höhere Spreads zu antizipieren indem mehrere mögliche Stop-Loss- und Take-Profit-Werte zur Auswahl stehen:

$$d_{profit}(a) := \begin{cases} \delta = 0, & : a = 0 \\ \delta = 2, & : a = 1, a = 6 \\ \delta = 5, & : a = 2, a = 7 \\ \delta = 10, & : a = 3, a = 8 \\ \delta = 25, & : a = 4, a = 9 \\ \delta = 50, & : a = 5, a = 10 \end{cases} \quad (3.3)$$

Hierbei entspricht die Aktion  $a = 0$  der Warteaktion, die Aktionen  $a \leq 5$  eröffnen entsprechende Kaufpositionen und die Aktionen  $a > 5$  entsprechende Leerverkaufspositionen. Durch diese Methode wurde ein gewisser Spielraum in der Strategie geschaffen, um die verschiedenen, durch die Latenzzeit verursachten, Probleme zu bewältigen. Diese anpassbaren Werte  $\delta$  ermöglichen es dem Agenten, unterschiedliche Preisänderungen zu antizipieren. Dies verringert zwar das Risiko eines sofortigen Ausfalls, führt in diesem Falle aber auch zu einem höheren Verlust. Um den Agenten auf aktuellen, realen Marktdaten zu trainieren, wurde dessen künstliches neuronales Netzwerk außerhalb der Handelszeiten mit den während dieses und vorheriger Handelstage aufgezeichneten Daten trainiert. Das LSTM-Netzwerk aus Abbildung 3.7 wurde mit einer Trainingsparameterkonfiguration von  $(b = 50, l = 250, \eta = 10^{-5})$  trainiert. In der entsprechenden Lerndynamik, dargestellt in Abbildung 3.8, kann gesehen werden, dass der Agent potenziell hohe Gewinne beibehält und gleichzeitig versucht, Verluste zu reduzieren, was während des Trainings insgesamt zu einem kleinen Gewinn führt. Der Test auf dem realen Handelsplatz wurde zehn Handelstage lang durchgeführt, an welchen der Agent komplett selbständig und ohne

manuelles Eingreifen 16 Handelsgeschäfte eröffnet und geschlossen hat. Hierzu wurde ein einfaches regelbasiertes Absicherungssystem umgesetzt, welches es dem Agenten erlaubt, eine Kaufposition, eine Leerverkaufsposition und eine dritte zufällige Position gleichzeitig zu eröffnen. Die Abbildung 3.9 zeigt die vom Agenten erzielten Gewinne und den entsprechenden Anstieg des Eigenkapitals.

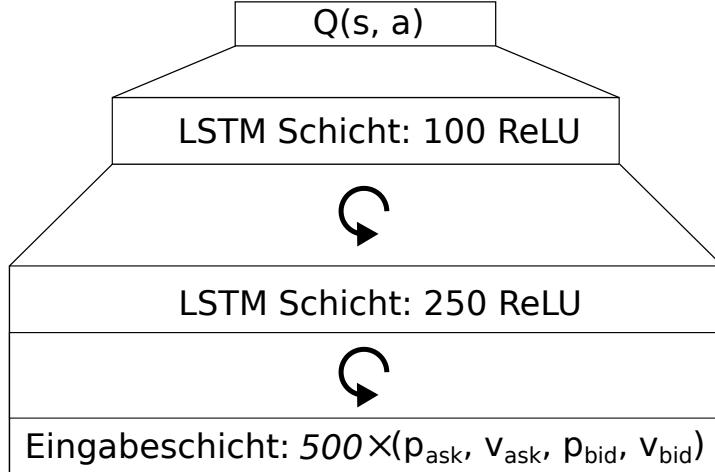


Abbildung 3.7: Die neuronale Architektur für die Anwendung im Echtzeittest, entnommen aus [7].

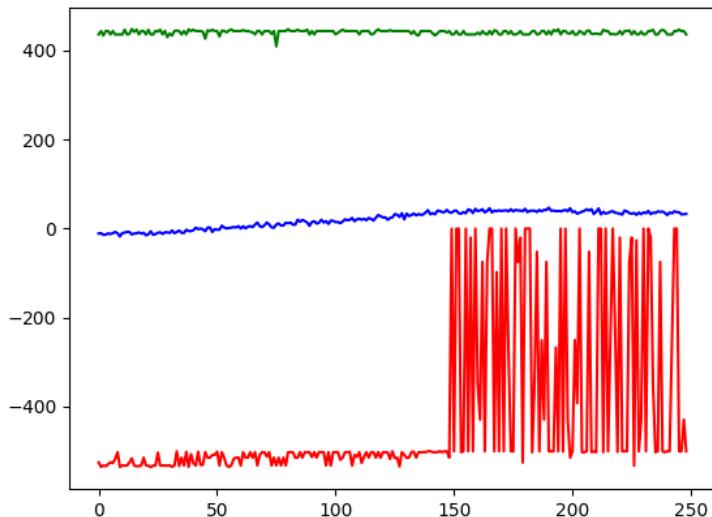


Abbildung 3.8: Eine Lerndynamik des LSTM-Modells im Echtzeittest, entnommen aus [7]. X-Achse: Anzahl Lernschritte in Tausend, Y-Achse: durchschnittliche Belohnung (blau), maximale Belohnung (grün), minimale Belohnung (rot).

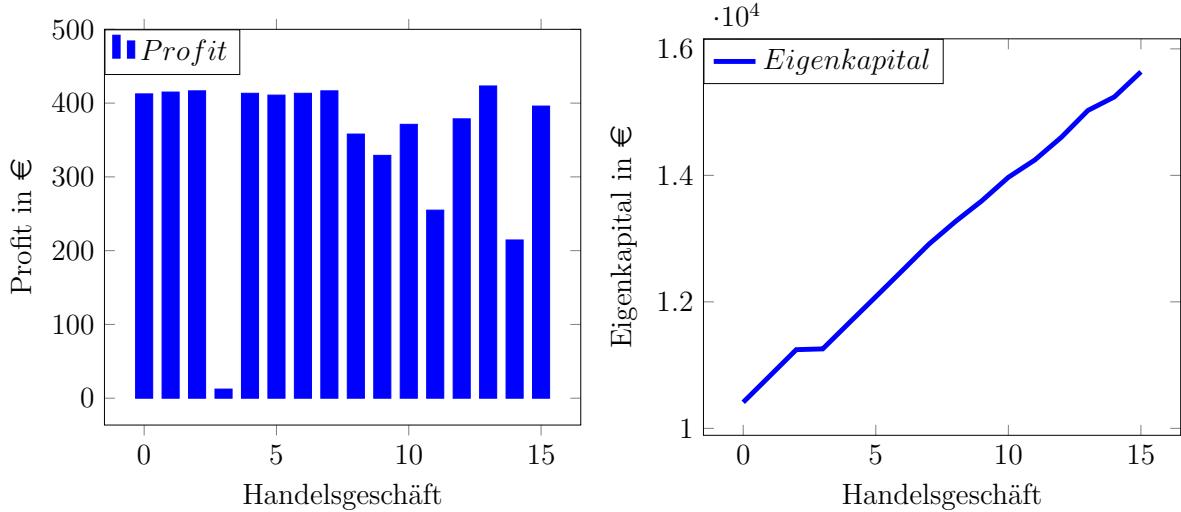


Abbildung 3.9: Die Auswertung der erweiterten LSTM-Architektur im Echtzeittest aus [7]. Links: Die erzielten Gewinne. Rechts: Die Eigenkapitalentwicklung.

### Beobachtungseffekte

Da bisher nur ein Basiswert mit einer festen Beobachtungslänge untersucht wurde, ergeben sich Fragen nach dem Effekt von unterschiedlichen Beobachtungszeiten und der Nutzung von Korrelationen zwischen verschiedenen Basiswerten. Um diese Auswirkungen zu verstehen, werden Kombinationen von Beobachtungszeiträumen und Vermögenswerten untersucht. Auf dieser Grundlage von Beobachtungen  $n = 43$  unterschiedlicher Basiswerte werden Beobachtungszeiten systematisch in unterschiedlichen Maßen angepasst. Hierzu werden verschiedene Zeiträume  $T = \{10s, 30s, 45s, 60s, 5m, 8m, 10m, 12m\}$  in jeweils fünf Testdurchläufen ausgewertet, welche dazu dienen statistische Effekte zu glätten, um zu einem aussagekräftigeren Ergebnis zu gelangen. Damit werden sowohl kleine Beobachtungszeiträume, wie zum Beispiel zehn Sekunden, aber auch recht lange Beobachtungszeiträume mit einer Eingabelänge von zwölf Minuten erfasst. An dieser Stelle könnte die Annahme getroffen werden, dass längere Beobachtungszeiträume es ermöglichen würden, eine breitere Strategiegrundlage zu entwickeln, welche zur weiteren Maximierung von Gewinne beitragen würde; jedoch stellt sich heraus, dass für jeden Basiswert verschiedene optimale Beobachtungszeiträume existieren.

Zur Auswertung eines jeden Beobachtungszeitraums wird derselbe Aufbau aus einem Haupttrainingszyklus und fünf verschiedenen Testzyklen verwendet. Ein Testzyklus besteht dabei aus jeweils einhundert Testtransaktionen, bei denen der Agent jeweils eine Entscheidung auf der Grundlage der gegebenen Beobachtung aus allen 43 Basiswerten trifft. Hierbei wird die rekurrente Faltungsarchitektur aus Abbildung 3.4 genutzt, wobei die Größe der Eingabeschicht je nach der gegebenen Beobachtungszeit verändert wird. Die optimalen Hyperparameter, wie etwa die Lernrate und die Stapelgröße, werden hierzu aus der Machbarkeitsstudie übernommen; an dieser Stelle würde sich, wie später diskutiert, ein Metalernansatz anbieten, um optimale Trainingsparameter abzuschätzen.

Als Indikator für die allgemeine Leistung eines auf diese Weise trainierten Handelsagenten wird die Anzahl der Tests mit finanziellem Gewinn oder Verlust gezählt. Wenn ein Agent einen Testlauf mit einem positiven Eigenkapitalzuwachs abschließt, zählen wir dies als Gewinn; es werden also nicht die Eigenkapitalentwicklungen sondern nur deren Vorzeichen gezählt. Dies ermöglicht es, ein breites Spektrum an möglichen Beobachtungszeiträumen zu vergleichen, da der Gesamterfolg nicht von konkreten Preisen abhängt. Für jeden der vier zu handelnden Basiswerte (*US500*, *OIL*, *GOLD*, *EURUSD*) werden fünf Testdurchläufe in den jeweils acht verschiedenen Beobachtungszeiträumen durchgeführt, also insgesamt 160 Versuche, um zu bestimmen, welche Auswirkungen die Beobachtungszeiträume auf den Handelserfolg in den bestimmten Märkten haben.

Abbildung 3.10 zeigt die Anzahl positiver Handelsabschlüsse für alle Beobachtungslängen und alle Vermögenswerte. Hier lässt sich kein eindeutiger Trend erkennen, jedoch kann interpretiert werden, dass eine optimale Laufzeit vom jeweiligen Basiswert abhängen muss. Die Darstellung der über alle Zeiträume summierten Ergebnisse in Abbildung 3.10 zeigt, dass der Ansatz des bestärkenden Lernens am zuverlässigsten auf dem Devisenmarkt funktioniert, während der Indexwert *US500* und der Goldmarkt an zweiter Stelle stehen und die geringste Anzahl von Gewinnen auf dem Ölmarkt zu finden ist. Um die Zusammenhänge im Detail darzulegen zeigt Tabelle 3.1 die Anzahl der Gewinne für alle einzelnen Vermögenswerte und zeitlichen Auflösungen; hierbei zeigt sich keine eindeutig überlegene Konfiguration, jedoch gute Ergebnisse für den Indexwert *US500* sowohl bei 45 Sekunden als auch bei einem Beobachtungszeitrahmen von zehn Minuten, gefolgt vom Goldmarkt bei zehn Minuten und dem *EURUSD* Devisenmarkt bei zehn Minuten und 45 Sekunden.

Gesamtzahl positiver Testergebnisse				
Zeit	US500	OIL	GOLD	EURUSD
10s	10	1	5	<b>19</b>
30s	7	5	7	8
45s	<b>13</b>	0	6	8
1m	<b>15</b>	3	11	4
5m	9	1	6	13
8m	6	1	7	5
10m	11	1	<b>12</b>	7
12m	9	3	7	<b>17</b>

Tabelle 3.1: Eine detaillierte Darstellung der Auswirkungen der Beobachtungszeiträume auf die Basiswerte in Tabellenform, entnommen aus [8]. Jede Zelle enthält die Gesamtzahl der positiven Auskommen (maximal 5) für jede Kombination von Vermögenswert und Beobachtungszeitrahmen.

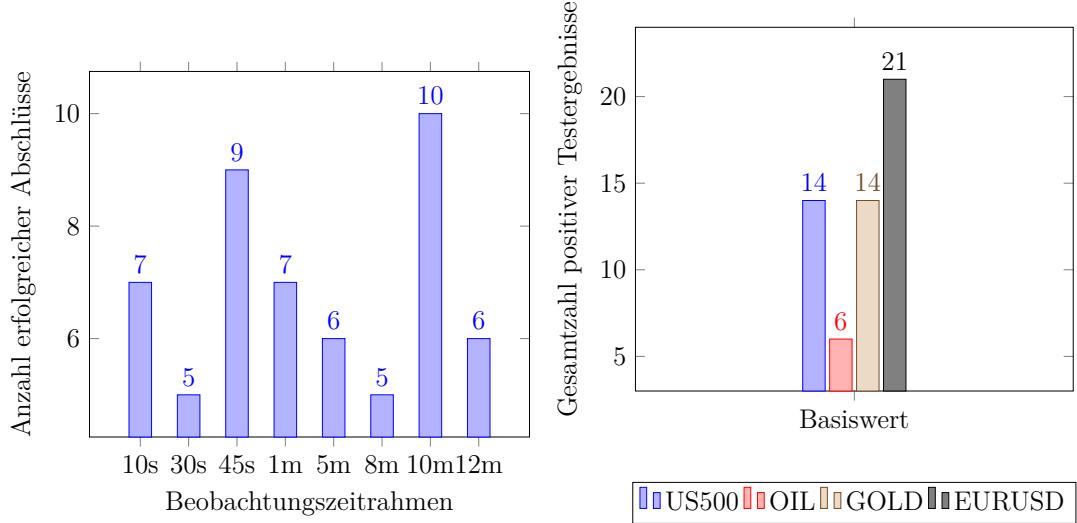


Abbildung 3.10: Links: Die Anzahl der positiven Ergebnisse in allen Versuchen einer bestimmten Beobachtungsdauer. Das Maximum von zehn positiven Versuchen findet sich bei einer Beobachtungsdauer von zehn Minuten, das zweitbeste Ergebnis bei 45 Sekunden. Rechts: Die Gesamtzahl der positiven Ergebnisse pro Basiswert, addiert über alle Zeitspannen. Beide Graphen sind entnommen aus [8].

### 3.1.6 Diskussion

Mit der im Rahmen dieser Herausforderung entwickelten Anwendung wurde parametrierbare Trainingsumgebung beigesteuert, welche das Training bestärkt lernender Agenten für einen Derivatemarkt ermöglicht. Die Implementierungen mit tiefen künstlichen neuronalen Netzen dienen als Konzeptnachweise für künstlich intelligente Handelsautomaten, welche auf hohen Frequenzen handeln können. Bezüglich der Marktsimulationslogik muss zunächst einmal festgestellt werden, dass die Untersuchungen den Einfluss der Handelsentscheidungen anderer Akteure auf die Preisentwicklung vernachlässigen. Eine Beobachtung der Lernerfolge in Derivatemarktsimulation zeigt auf, dass es wichtig ist, ein Trainingsdesign zu verwenden, welches den tatsächlichen Handelsbedingungen entspricht. Durch den Vergleich der Ergebnisse des rekurrenten LSTM-Netzes mit dem mehrschichtigen Perzeptron und einem linearen Modell kann die Annahme bestätigt werden, dass die Annahme von Sequenzen in Handelsdaten die Ergebnisse eines maschinellen Lernsystems verbessern. Die Untersuchungen zeigen, dass künstlich intelligente Handelsautomaten, je nach Umgebung, verschiedene optimale Strategien erlernen können. Jedoch wurde nicht untersucht, welche Auswirkungen eine Veränderung des Nutzen-Risiko-Verhältnisses während der Trainingszeit hat. Weiterhin wurde nicht untersucht, wie sich eine Vergrößerung des Wiederholungsspeichers, oder andere Lernverfahren als Q-Lernen, beispielsweise asynchrone Verfahren, auf den Lernerfolg auswirken. Anstelle einer Stapelnormalisierung mit einer Division durch die Standardabweichung wurde nur der Mittelwert abgezogen.

Für den Echtzeittest wurde ein Demokonto mit einer begrenzten Gültigkeit von nur 20 Handelstagen verwendet, was die Ergebnisse relativiert, da effektiv nur zehn Handelstage genutzt werden konnten, um den Ansatz unter realen Bedingungen zu evaluieren.

Für zukünftige Forschung in dieser Richtung bleibt die Frage nach einem Metalerner bezüglich der Variation der Hyperparameter des Lernvorgangs, welche das Lernverhalten des Agenten verändern können. Ebenso bleibt die Frage bestehen, inwiefern vortrainierte Modelle mit Hilfe von Transferlernen, beispielsweise über Feinabstimmung, an neue Marktsituationen angepasst werden können. Im Sinne des *Transfer Meta Learning* würde eine Vergleichsstudie benötigt, in welcher vortrainierte Modelle unter Zuhilfenahme von Metadaten auf neue Marktsituationen transferiert werden können. Um diese grundlegende Vergleichsstudie durchzuführen würde eine entsprechende Marktsimulation benötigt werden, welche die Daten in den verschiedenen benötigten Darstellungen bereitstellen und auch die verschiedenen Ordertypen interpretieren kann. Eine solche Umgebung könnte auch eine Handelslogik für verschiedene Basiswerte, Geschäftsarten und deren Derivate bieten. Durch die Einführung einer zufälligen Wartezeit würde die Simulation Versuche ermöglichen, welche den Anforderungen an Untersuchungen des Hochfrequenzhandels Rechnung tragen; so würde eine solche künstliche Latenz die Simulation verschiedener Agenten ermöglichen, welche in verschiedenen Signallaufzeiten miteinander konkurrieren. Die Erforschung der Auswirkungen einer Änderung der Beobachtungszeitspanne auf tägliche, oder gar wöchentliche, Zeitspannen bleibt ebenfalls Gegenstand für zukünftige Forschung. Darüber hinaus könnte in zukünftigen Arbeiten in dieser Richtung auch die Integration von Wirtschaftsnachrichten als Eingangswortvektoren in Betracht gezogen werden. Auf diese Weise können die Handelsvorschläge der bestärkend lernenden Agenten als hilfreicher Input für ausgefeilte Handelsalgorithmen dienen, welche beispielsweise auf vorheriges Marktwissen über gesellschaftspolitische Zusammenhänge in Form von Nachrichten zurückgreifen. Da diese Anwendung es ermöglicht, Handelsstrategien auf prinzipiell beliebigen Zeitskalen zu erlernen, könnten auch Minuten-, Stunden- oder sogar Tagesschlusskurse als Datenbasis für das Training bereitgestellt werden. So könnte beispielsweise ein regelbasiertes System, welches langfristige Marktkenntnisse integriert, die Vorschläge der kurzfristig agierenden Agenten nutzen, um ein vollautomatisches, zuverlässiges Handelsprogramm zu erstellen. Ein solches Programm könnte die Agenten auch daran hindern, Positionen oberhalb oder unterhalb eines bestimmten Schwellenwerts zu eröffnen, welchen beispielsweise ein menschlicher Bediener auf der Grundlage seiner vorherigen Marktkenntnisse festlegen könnte.

Weitere mögliche Entwicklungen in dieser Richtung bestehen darin, die verschiedenartigen Beobachtungen zunächst in einen latenten Raum zu überführen, beispielsweise mit einem, entsprechend der Herausforderung angepassten, *Variational Autoencoder* (VAE) [170, 59]. Mit einem solchen latenten Vektor, welcher den Zustand des Marktes zusammenfassend beschreibt, könnten, nach dem Vorbild der Weltmodelle [19], *Mixture Density Networks* (MDN) [171] Einsatz finden, um den nächsten Zustand anhand des aktuellen Zustands vorherzusagen. Auf diese Weise kann das in einem Weltmodell zusammen geführte Wissen mit nur wenigen lernbaren Parametern in beliebige neue Handelssituation übertragen werden. Ein Transferlernansatz über Weltmodellwissen hat die Vorteile der zeitlichen Invarianz, erklärbarer Strategien, und der schnellen Übertragung auf neue Aufgaben.

Zusammenfassend lässt sich sagen, dass diese Anwendung die Existenz eines Hochfrequenzhandelssystems belegt, welches den Markt zumindest in einer Simulation bei einer Latenzzeit gegen Null deutlich übertrifft. Der Echtzeittest zeigt außerdem, dass zusätzliche Modellparameter eine geringere Latenz bis zu einem gewissen Grad ausgleichen können. Abschließend wird bestätigt, dass sich keine allgemein optimale Beobachtungszeit finden lässt, sondern dass die optimale Beobachtungszeit stark von dem gewählten Handelsobjekt abhängt; bei der Verwendung unterschiedlicher Beobachtungszeiträume zeigt sich in keinem Fall eine generelle Verbesserung bei längerer oder kürzerer Dauer der Eingabesequenz.

## 3.2 Handgestenerkennung

Eine praktisch nützliche Anwendung maschinellen Lernens im Bereich der Mensch-Maschineinteraktion ist die Erkennung von Handgesten. Dies erlaubt einem Menschen über symbolische Handbewegungen mit einer Maschine, beispielsweise einem Automobilfahrzeug, intuitiv zu interagieren. Aktuelle Ansätze zur maschinellen Handgestenerkennung nutzen Kamerabilder, meistens in Kombination mit Raumtiefeninformation von Time-of-Flight-Sensoren, um mit faltenden neuronalen Netzwerken statische Gesten oder, mit Videoaufnahmen und LSTM-Netzwerken, dynamische Handgesten zu klassifizieren, wie in der Metastudie von [9] untersucht. Abbildung 3.11 veranschaulicht die Zusammenhänge dieser Problemstellung in Form eines UML-Diagramms.

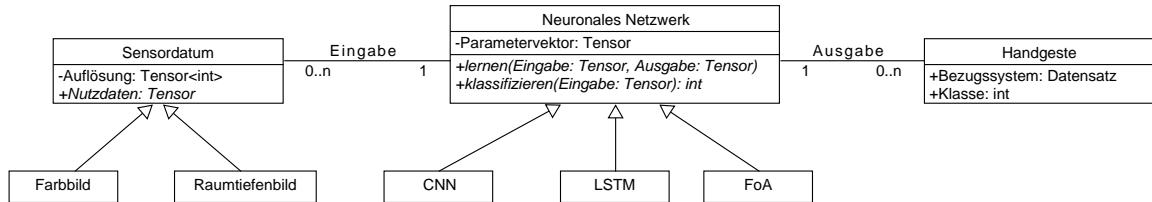


Abbildung 3.11: Ein UML-Diagramm zur Skizze der Zusammenhänge zwischen Sensordaten, neuronalen Netzwerken und Handgesten. Die Sensordaten entstammen entweder Farb- oder Raumtiefenkameras. Verschiedene neuronale Netzwerke versuchen aus diesen Daten eine Abbildung zu Handgesten zu lernen, welche sich über eine Ganzzahl als Klasse eines Datensatzes klassifizieren lassen.

In der Metastudie von [9] wurden verschiedene Handgestenerkennungssysteme auf Basis von Farb- und Raumtiefenbildern untersucht. Die untersuchten Forschungsarbeiten in diesem Bereich verwenden eine Vielzahl von unterschiedlichen Modellen und Datensätzen. Abbildung 3.12 zeigt die zeitliche Entwicklung datengetriebener Handgestenerkennungssysteme auf Basis von Raumtiefeninformation von den ersten Ansätzen bis hin zu aktuellen tiefen künstlichen neuronalen Architekturen.

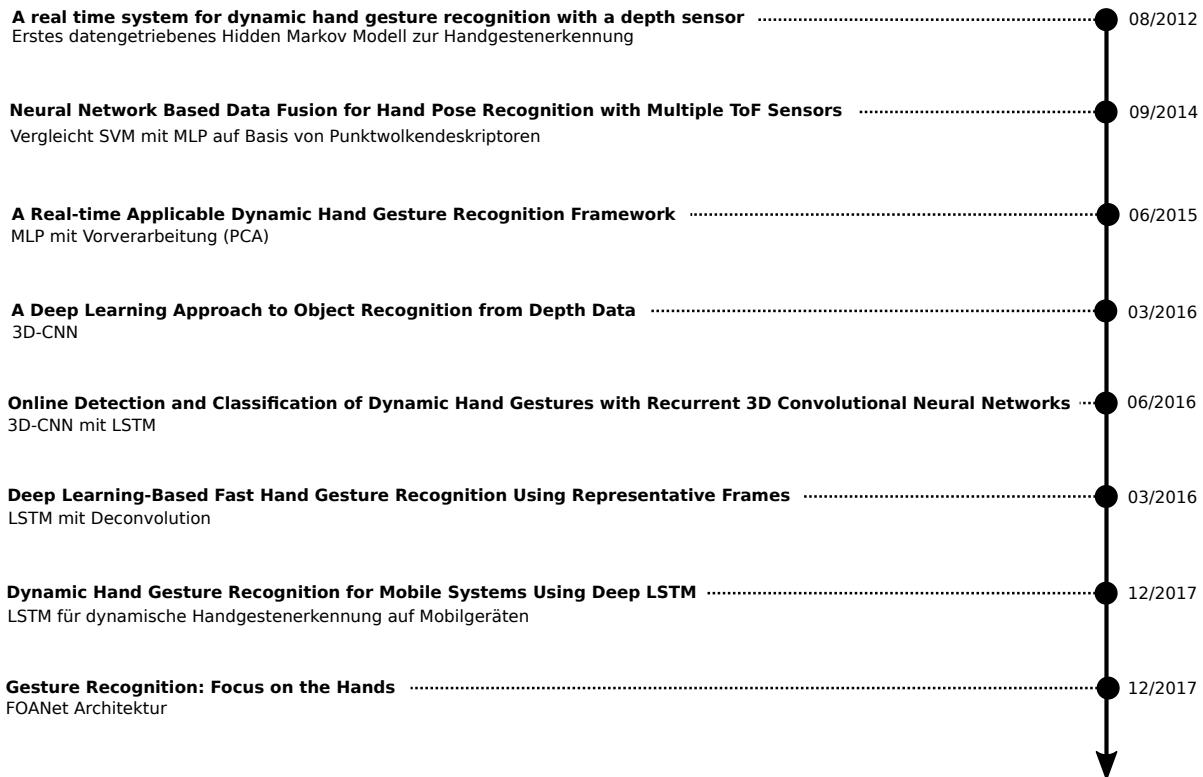


Abbildung 3.12: Die in der Metastudie von [9] zur Handgestenerkennung untersuchten Beiträge in chronologischer Reihenfolge.

In einer Literaturübersicht zur Handgestenerkennung mit Raumtiefeninformation hat [222] 37 Arbeiten mit insgesamt 24 Methoden untersucht. [223] präsentierte einen umfassenden Überblick über relevante Grundideen für die Handgestenerkennung, welche die Computer Vision im Allgemeinen und verschiedene maschinelle Lernverfahren im Besonderen betreffen. [224] gab einen Überblick über verschiedene Methoden, die zu guten Ergebnissen in konkreten Anwendungssituationen führen. [225] überprüfte die Literatur zu verschiedenen Gestenerkennungsmethoden, darunter neuronale Netzwerke, versteckte Markov-Modelle, Fuzzy-C-Means-Clustering und Orientierungshistogramme für die Merkmalsdarstellung. Weitere Literaturübersichten, wie [226], [227] und [228], vermitteln weitere Informationen über die Handgestenerkennung mit Tiefendaten. Eine prominente Anwendung solcher Systeme ist die Fahrassistenz, denn mit Hilfe von Handgestenerkennung kann die Interaktion mit einem Vehikel stattfinden, ohne den Blick von der Straße abzuwenden, um beispielsweise das Verhalten eines autonomen Fahrzeugs oder Unterhaltungsanwendungen zu steuern; und im Gegensatz zur Sprachsteuerung funktioniert dieser Ansatz auch bei einem hohen Geräuschpegel [229, 230, 231]. Um zu gewährleisten, dass sich die fahrende Person voll auf die Umwelt und die Verkehrssicherheit konzentrieren kann, soll ein Fahrassistenzsystem mit minimaler kognitiver Ablenkung bei der Steuerung der elektronischen Gerätschaften im Vehikel helfen [231]. Handgestenerkennungssysteme stellen eine solche zuverlässige Steuerschnittstelle dar,

welche sowohl die Anforderungen an den Komfort als auch an die Sicherheit erfüllt [230]. Andere interessante Anwendungen finden sich in der Mensch-Maschineinteraktion im operativmedizinischen Bereich; die Autoren von [232] stellen ein Gestenerkennungsnetzwerk für die Navigation von MRT-Daten während Neurobiopsieverfahren vor.

Dreidimensionale Daten sind in verschiedenen Formaten verfügbar; beispielsweise Punktfolgen, Tiefenbilder oder geometrischen Gitterdaten, wobei Punktfolgen das am häufigsten verwendete Format darstellen [9, 233, 234, 235]. Im Vergleich zu zweidimensionalen Bildern bieten dreidimensionale Daten Information bezüglich der Entfernung von Objekten im Bild, wodurch sie sich besonders für industrielle Anwendungen eignen [235]. Beispielsweise können Raumtiefendaten in automatisierten Be- und Entladungsvorgänge von Frachtschiffen Verwendung finden [233]. Im Fall einer Punktfolge, exemplarisch dargestellt in Abbildung 3.13, besteht die Datenstruktur aus einer ungeordneten Menge von Punkten in kartesischen Koordinaten, je nach Methode und Format kann Information wie etwa Farbe und Intensität zu einem Punkt hinzugefügt werden. Bis zum Einsatz von tiefen künstlichen neuronalen Netzen basierte die Verarbeitung von Punktfolgen hauptsächlich auf Merkmalen, auch als Punktfolkendeskriptoren bezeichnet, wie etwa das *Ensemble of Shape Functions* (ESF, [236]), das *Point Feature Histogram* (PFH, [237]) oder das *Viewpoint Feature Histogram* (VFH, [238]).

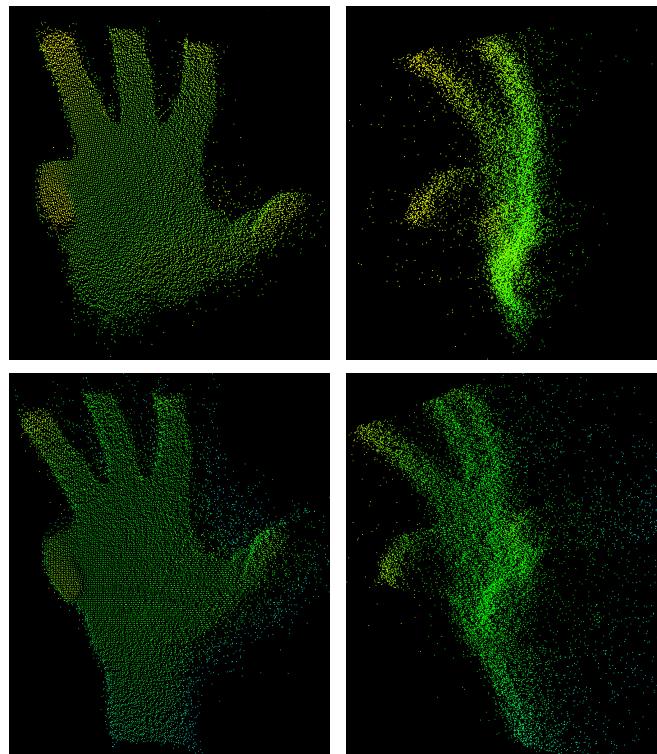


Abbildung 3.13: Eine Veranschaulichung von Raumtiefendaten einer Handgeste in Form von Punktfolgen aus verschiedenen Aufnahmewinkeln, entnommen aus [9].

Im Allgemeinen helfen Raumtiefeninformationen bei der Unterscheidung mehrdeutiger Handgesten, wie von [239] beschrieben. Einige Ansätze beruhen auf der Erkennung bestimmter Handpixel [240] und verwenden Algorithmen oder Zustandsautomaten zur Erkennung dynamischer Gesten [241]. Die Autoren von [242] verwenden einen einzigen Flugzeitsensor und nutzen den VFH-Deskriptor zur Erkennung von Handzeichen, was die Bedeutung geeigneter Punktwolkendeskriptoren hervorhebt. Bessere Ergebnisse, wie sie bei der Zusammenführung von Stereokamerainformationen von Tiefensensoren erzielt wurden, bestätigen den Nutzen der Verwendung eines zweiten Sensors und der Sensorfusion [243]. Die Erkennung von dynamischen Handgesten wirft das Problem der räumlich-zeitlichen Segmentierung auf, wie [244] feststellte.

### 3.2.1 Datensätze

Der Cambridge Gestenerkennungsdatensatz enthält neun Handgestenklassen, die von zwei Personen in 100 Videosequenzen in fünf verschiedenen Beleuchtungskonfigurationen abgetastet wurden [245]. Da die Sequenzlängen des Cambridge Datensatzes variieren und der Datensatz zur Zeit nicht mehr verfügbar ist, bleibt die genaue Anzahl der Samples unklar. [246] führen einen neuen Datensatz mit dynamischen Handgesten ein, der mit Tiefen- und Farbdaten erfasst wurde und in späteren Untersuchungen als NVidia-Benchmark bezeichnet wird. Hier wurden 1.532 Sequenzen von insgesamt 25 unterschiedlichen Handgestenklassen aufgenommen. [247] veröffentlichte den EgoGesture-Datensatz mit insgesamt 3.000.000 Frames von 24.000 Handgesten mit Farb- und Tiefeninformationen in 2.081 Sequenzen mit 83 Handgestenklassen, welche von 50 verschiedenen Personen aufgenommen wurden. Mit dem Beitrag von [248] wurde der Datensatz REHAP (Recognition of Hand Postures) veröffentlicht, welcher insgesamt über eine 1.050.000 unterschiedlicher Aufnahmen von zehn Handgestenklassen in Form von Raumtiefenbildern, dargestellt in Abbildung 3.14, und gliedert sich in zwei Teile: der erste Teil, REHAP-1, verfügt über insgesamt 600.000 einzelne Raumtiefenbilder von 20 verschiedenen Personen, der zweite Teil, REHAP-2, beinhaltet 450.000 Aufnahmen von 15 verschiedenen Personen. Von jeder der zehn Handgestenklassen im REHAP-Datensatz wurden 3.000 Sequenzen aufgenommen. [249] veröffentlichte den ChaLearn IsoGD-Datensatz, der 249 unterschiedlichen Gestenlabels aus 30 verschiedenen Gestensprachen in 1.200.000 Datenpunkten enthält, einschließlich 47.933 manuell beschrifteten dynamischen Handgestensequenzen mit Farb- und Raumtiefeninformationen. Tabelle 3.2 fasst die Metadaten der von [9] untersuchten Handgestendatensätze zusammen.

Datensatz	Bildformat	Anzahl Sequenzen	Anzahl Gestenklassen
Cambridge HGD	RGB	900	9
NVidia Benchmark	RGB+D	1.532	25
EgoGesture	RGB+D	2.081	83
REHAP	RGB+D	30.000	10
ChaLearn IsoGD	RGB+D	47.933	249

Tabelle 3.2: Metadaten der verschiedenen Handgestendatensätze.

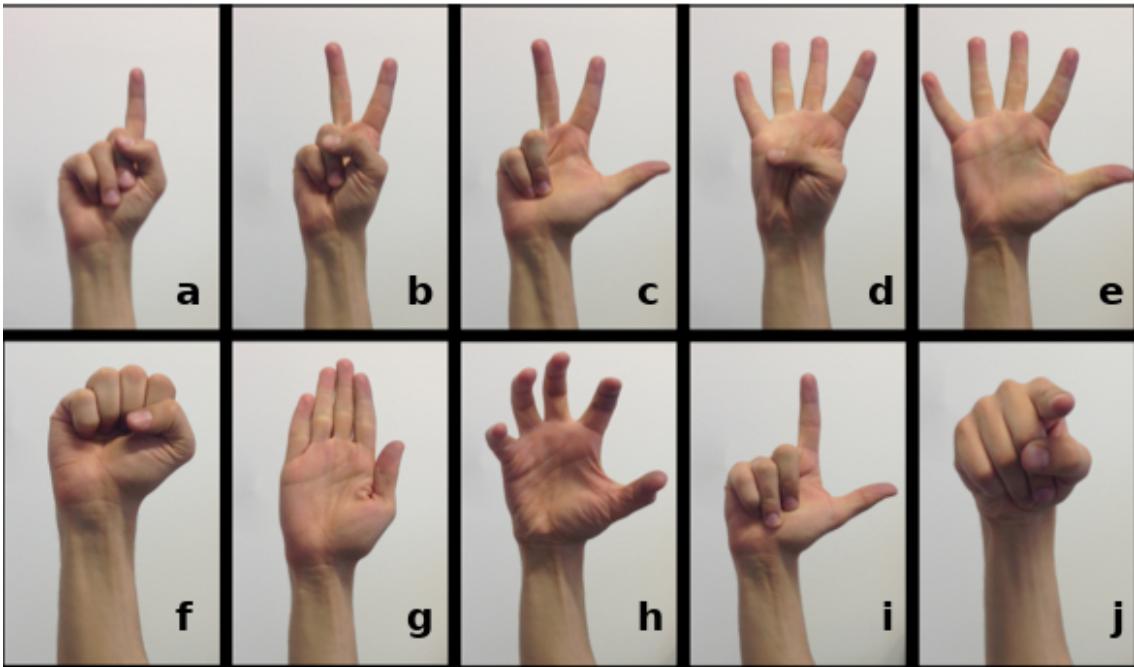


Abbildung 3.14: Die zehn Handgestenklassen aus dem REHAP Datensatz von [248], entnommen aus [9]. Die Handgesten in der oberen Reihe zeigen verschiedene Anzahlen gehobener Finger, was als numerischer Wert interpretiert werden kann. Die Handgestenklassen in der unteren Reihe zeigen andere bedeutungsvolle Gesten, wie beispielsweise eine Faust oder eine flache Hand, welche Start- und Endpunkte einer dynamischen Handgeseitensequenz verwendet werden können.

### 3.2.2 Modelle

Frühe Arbeiten verwendeten keine Deep Learning Methoden für Raumtiefenbilder, bieten aber wertvolle Einblicke in verschiedene Ideen, wie die Handgestenerkennung auf andere Weise angegangen werden kann. Beispielsweise hat [250] ein Erkennungssystem entworfen, welches eine Genauigkeit von bis zu 87,7% auf einem Datensatz der Amerikanischen Gebärdensprache (ASL) erreicht. Dies gilt als das erste datengesteuerte System, welches in der Lage war, Handgesten automatisch zu erkennen, und zwar ohne Deep Learning Methoden, sondern mit Hidden Markov Modellen (HMM). Inspiriert von [251] isolierte der Beitrag von [252] in frühen Experimenten den relevanten Teil der Hand vom Rest des Körpers durch einen Schwellwert und eine Hauptkomponentenanalyse (PCA). Die Autoren von [253] berichten eine Genauigkeit von 95% durch den Einsatz eines

	RGB	Tiefe	RGB Fluss	Tiefe Fluss
Global	43.98%	66.80%	62.66%	58.71%
Fokus	58.09%	73.65%	77.18%	70.12%

Tabelle 3.3: Die Leistungen der einzelnen Kanäle der FOANet-Architektur auf dem NVidia-Benchmarkdatensatz, entnommen von [254].

rekurrenten LSTM-Faltungsnetzwerks zur Klassifizierung dynamischer Handgesten aus dem Cambridge Gestenerkennungsdatensatz. Dieser Ansatz extrahiert die relevantesten Frames mit Hilfe eines auf semantischer Segmentierung basierenden Deep Learning Frameworks, das die relevanten Videoteile in Form von Kachelmustern darstellt. [246] untersuchte ein dreidimensionales neuronales Faltungsnetzwerk mit einer rekurrenten Schicht, wie in Abbildung 3.16 dargestellt. Auf diesem Datensatz erreichte ihr Gestenerkennungssystem eine Genauigkeit von 83,8%. Mit ihrer FOANet-Architektur, dargestellt in Abbildung 3.15, verbesserten [254] die Leistung auf dem ChaLearn IsoGD-Datensatz von einer vorherigen Bestmarke von 67,7% auf 82,1% und auf dem NVidia-Datensatz von 83,8% auf 91,3%. Hierbei verwendet das FOANet keine zeitliche Fusion, sondern optische Flussfelder der RGB- und Tiefenbilder. Diese Architektur besteht aus einem separaten Kanal für jede Fokusregion und Eingangsmodalität. Ein integriertes Focus-of-Attention-Modul (FOA) erkennt Hände, eine Softmax-Score-Schicht fasst 12-Kanäle zusammen und eine Sparse-Fusion-Schicht kombiniert die Softmax-Scores entsprechend der Klassenwahrscheinlichkeiten. Mit dieser Architektur übertreffen [254] sowohl das bisher beste Ergebnis als auch die menschliche Genauigkeit. Die Genauigkeit von FOANet sinkt, wenn die Fusion von spärlichen Netzwerken durch eine durchschnittliche Fusion ersetzt wird. Aus den in Tabelle 3.3 aufgeführten Ergebnissen geht hervor, dass der fokussierte RGB-Flussfeldkanal am meisten zur Klassifikation beiträgt. Außerdem lässt sich ein allgemeiner Vorteil von fokussierten Kanälen gegenüber globalen Kanälen feststellen [254].

### 3.2.3 Diskussion

In der Metastudie zur Handgestenerkennung wurde der aktuellen Stand der Wissenschaft und Technik mit dem Schwerpunkt auf dem Training tiefer künstlicher neuronaler Netzwerke untersucht und insbesondere eine Forschungslinie des Instituts für Informatik der Hochschule Ruhr West konsolidiert. Alle Detailinformation, Handgestendaten und Quelltexte wurden veröffentlicht [9].

Die untersuchten Beiträge bilden die Grundlage für Forschung zur Erkennung von Handgesten im Automobilkontext mit dreidimensionalen Daten von Infrarot Time-of-Flight-Sensoren, um neue Steuerungsmöglichkeiten für die Fahrerassistenz zu schaffen. Im Vergleich der eigenen Ansätze mit verwandten Arbeiten kann festgestellt werden, dass die Leichtbauimplementierungen für mobile Geräte geeignet sind und über eine angemessene Genauigkeit verfügen. Mit einem INTUI-Fragebogen wurde versucht, die Vertrautheit der einzelnen Fahrer mit dem System beim ersten Einsatz zu beurteilen.

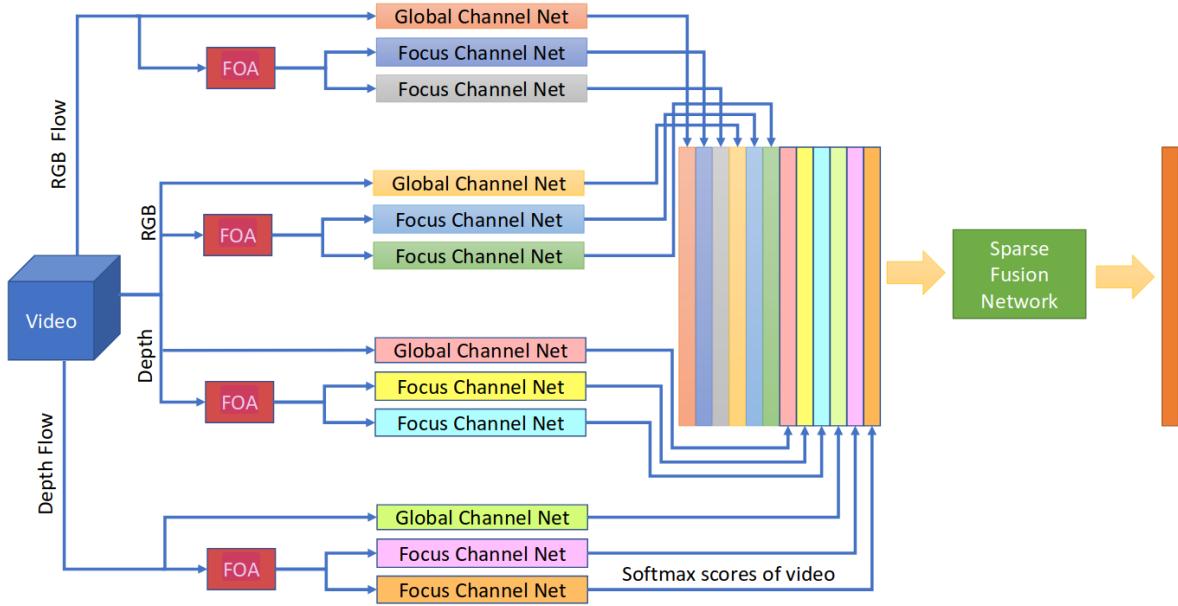


Abbildung 3.15: Die FOANet-Architektur, entnommen aus [254]. Sie besteht aus einem separaten Kanal für jede Fokusregion (global, linke Hand, rechte Hand) und jede Eingabemodalität (RGB, Raumtiefe, RGB-Fluss und Raumtiefenfluss).

Ein standardisierter Spurwechseltest, wie er in der ISO-Norm 26022 beschrieben ist, beleuchtet die Auswirkungen der Technologien auf das Fahrverhalten von Autofahrern. Die Nutzerfreundlichkeit der Freihandgesten wurde nicht mit traditionellen Steuermechanismen verglichen, wie beispielsweise Tasten am Lenkrad; dieser wichtige Vergleich bleibt ebenfalls späterer Forschung überlassen. Auch mehr Nutzerstudien mit größerer Teilnehmerzahl können zu einem detaillierten Ergebnis führen. Mit größeren Rechenressourcen können auch Modelle mit größeren Parameterräumen untersucht werden; aber solche Vergleiche bleiben für die zukünftige Forschung bestehen. Zukünftige Arbeiten könnten in diesem Anwendungsbereich könnten die Generalisierungsmöglichkeiten des LSTM-Modells mittels Transferlernen auf größeren Datensätzen untersuchen. Außerdem könnten neue Modelle gefunden werden, indem die Faltungsarchitektur mit der rekurrenten Schicht rekombiniert wird.

### 3.3 Personenerkennung

Der sich abzeichnende wirtschaftliche Trend der Industrie 4.0 erfordert Software zur Unterstützung menschlicher Arbeit in der Schwerindustrie. Der Beitrag von [10] untersucht verschiedene künstlich intelligente Ansätze zur Personenerkennung im Rahmen des Projektes *DamokleS 4.0*. Das Ziel des Projektes *DamokleS 4.0* ist es, ein System zu entwickeln, welches Mitarbeiterinnen und Mitarbeitern der Schwerindustrie mit moderner Hard- und Software im Arbeitsalltag unterstützt [12, 17, 15].

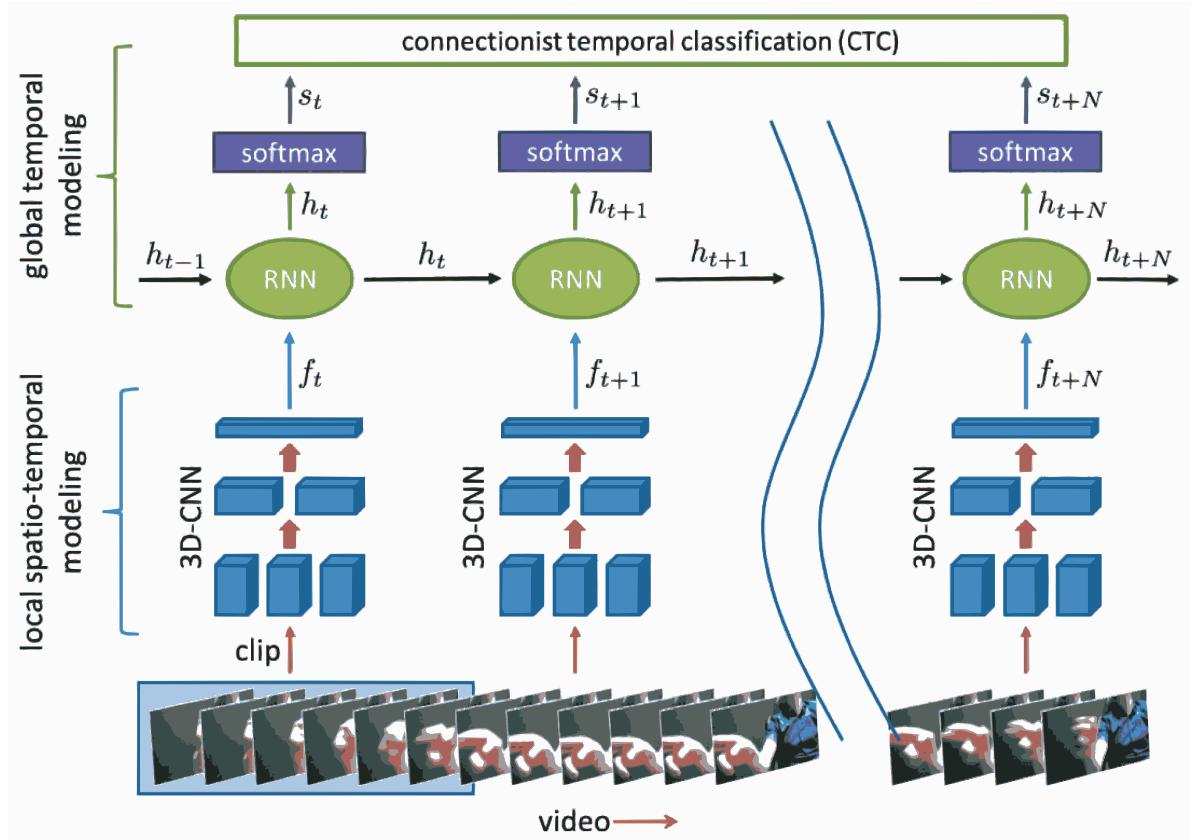


Abbildung 3.16: Die rekurrente dreidimensionale Faltungsarchitektur, entnommen aus [246]. Als Eingabe verwendet das Netzwerk eine dynamische Geste in Form von aufeinanderfolgenden Frames. Es extrahiert lokale räumlich-zeitliche Merkmale über ein 3D-CNN und speist diese in eine rekurrente Schicht ein, welche die Aktivierung über die gesamte Sequenz aggregiert. Anhand dieser Aktivierungen nennt eine Softmax-Schicht dann eine Wahrscheinlichkeitsverteilung über den Klassenraum.

Aus wirtschaftlichen Anforderungen an Produktivität, Flexibilität und Arbeitssicherheit benötigt eine solche Facility Management Anwendung Information über den Produktionsprozess in Echtzeit und kann beispielsweise Realitätserweiterungsbrillen (E: *Augmented Reality*, AR) und andere mobile Geräte nutzen, um den Arbeitnehmerinnen und Arbeitnehmern jederzeit Informationen zur Verfügung zu stellen. Dies soll beispielsweise dabei helfen, die Arbeitszeit effizienter zu nutzen oder im Notfall schnelle und sichere Evakuierungswege zu finden. Die Grundlage eines solchen Systems besteht in einer zuverlässigen Erkennung von Menschen in Fabrikanlagen; die Zusammenfassung der Informationen in einem Kontextmodell ermöglicht dann eine Informationshilfe und eine verbesserte Produktionsplanung. Ein wesentlicher Bestandteil einer Personenerkennungssoftware ist die automatische, künstlich intelligente Verarbeitung von digitalen Farbbildern, welche von Kameras in der Fabrikanlage bereitgestellt werden. Abbildung 3.17 skizziert die Zusammenhänge der in dieser Herausforderung beteiligten Objekte in Form eines UML-Diagramms.

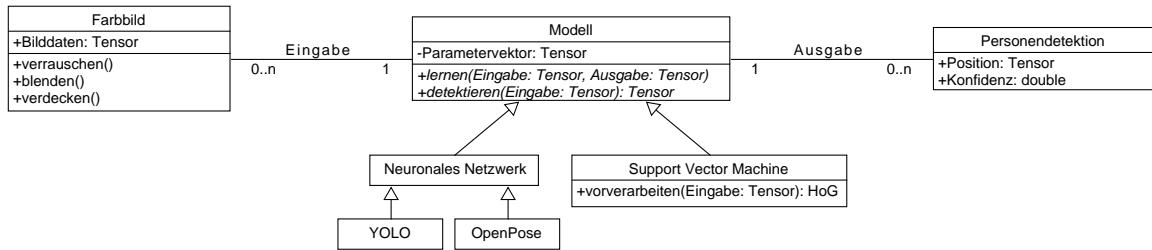


Abbildung 3.17: Ein UML-Diagramm zur Illustration des Zusammenspiels zwischen den Farbbildern, den in dieser Problemstellung untersuchten Modellen zur Personendetektion und der geforderten Ausgabe in Form von Positionen der Menschen im Bild.

Zu diesem Zweck werden verschiedene maschinelle Lernmodelle für die Erkennung des menschlichen Körpers in der Schwerindustrie untersucht. Der Stand der Technik kennt eine Vielzahl von Methoden zur Erkennung menschlicher Körper auf Kamerabildern [10, 255, 256, 53, 257, 258, 51, 52, 259, 260, 261, 54, 56, 55, 262]. Zur Beurteilung der Zuverlässigkeit eines Detektionssystems in industriellen Umgebungen werden Techniken der Merkmalsextraktion über Gradientenorientierungshistogramme mit künstlichen neuronalen Netzwerken verglichen. Um den üblichen Herausforderungen in der Schwerindustrieumgebung gerecht zu werden, wie etwa Staub, schwierige Lichtverhältnisse und teilweise abgedeckte Personen, werden entsprechend programmatische Änderungen an den Bilddaten angewandt und die Genauigkeit der Personenerkennung, der Fußpunktabschätzung und die Tendenz zu Fehlern bewertet. Auf diese Weise werden verschiedene Modelle auf einem Videodatensatz mit simulierten industriellen Arbeiten verglichen, welcher unter Laborbedingungen aufgenommen wurde, um menschliche Körper in einer der Schwerindustrie ähnlichen Umgebung möglichst genau und ausfallsicher zu erkennen.

In einem vorhergehenden Forschungsprojekt wurde ein Videoüberwachungssystem zum Schutz kritischer Infrastrukturen eingeführt, welches die Gradientenorientierungshistogrammmethoden (E: *Histogram of oriented gradients*, HOG) in Kombination mit einem Kalmanfilteralgorithmus verwendet [263]. In diesem Projekt wurde die Softwarearchitektur so konzipiert, dass sie menschliche Nutzer dabei unterstützt, im Falle einer Warnung verdächtige Personen zu erkennen und über die verschiedenen Kameras wiederzuerkennen und zu verfolgen. Dieses System wurde an zwei Referenzflughäfen implementiert, um Erkenntnisse über auftretende Herausforderungen in realen Anwendungen zu sammeln; dabei wurde in Erfahrung gebracht, dass die riesige Menge an Bilddaten, welche in einem Netzwerk von nicht überlappenden Kameras aufgezeichnet wurden, die Wiederherstellung einer einmal erkannten Person erschwert. Die in [263] beschriebenen Szenarien ähneln denen im Kontext der Schwerindustrie in Bezug auf die Herausforderungen durch unterschiedliche Lichtverhältnisse und den Bedarf an schnellen Algorithmen. Bezüglich des Projekts *DamokleS 4.0* beschreibt [264] die Softwarearchitektur und skizziert die wesentlichen Ideen, welche die Testszenarien antreiben, sowie die damit Implementierung für mobile Geräte. Die vorgeschlagenen Szenarien betreffen Anwendungen in den Bereichen Arbeitssicherheit, Produktion und Instandhaltung, der vorgeschlagene Ansatz bietet kontextbezogene Unterstützung für die Mitarbeiter der Fabrik in all diesen Szenarien. Für die Kontexterkennung schlägt [264] die Verwendung von Sensoren für mobile Geräte und externe Sensoren vor, die im Fabrikgebäude montiert sind.

### 3.3.1 Datensatz

Zur Untersuchung der Qualitäten der Personenerkennungsmodelle wurde ein Videodatensatz unter Laborbedingungen aufgezeichnet, welcher die Herausforderungen der Personenerkennung unter realen Industriebedingungen nachstellen soll, wie etwa die teilweise Verdeckung mehrerer sich bewegender menschlicher Körper im Bild. Hierzu wurden vier AVT Prosilica GE1650C Videokameras mit einer Auflösung von  $1600 \times 1200$  Pixel montiert, von denen, für diese Anwendung, nur die Aufnahmen zweier Kameras in die Auswertung aufgenommen wurden. Die simulierten Arbeitsaufgabenszenen wurden in drei Modalitäten aufgenommen. In der ersten Aufnahmemodalität bewegt sich eine einzelne Person durch den Raum, um an einem Computer zu arbeiten. Die zweite Aufnahmereihe zeigt eine Gruppe von zwei Personen, welche den selben Weg entlang gehen. Für die dritte Modalität bewegt sich eine Gruppe von sechs Personen quasi zufällig durch den Raum, währenddessen verschiedene Arbeiten ausgeführt werden. Für die ersten beiden Aufnahmemodalitäten tragen die Akteure Arbeitskleidung, in der Gruppenaufnahme betreten drei Personen in Alltagskleidung die Szene.

Insgesamt wurden etwa 4.000 Bilder aufgenommen, von denen insgesamt 3.557 manuell beschriftet wurden, jede Beschriftung erfolgte mit Begrenzungsrahmen um die einzelnen Personen und um ihre Füße herum. Diese Bilder wurden mit systematischen Störungen, wie in Abbildung 3.18 dargestellt und nachfolgend erläutert, vorverarbeitet und ergeben einen Datensatz mit insgesamt 142.311 Bildern. Anhand dieses Bilddatensatzes wird die Qualität der Personenerkennungsmodelle evaluiert, indem sowohl die Erkennung menschlicher Körper an sich, wie auch die Erkennung derer Fußpunkte, ausgewertet wird.



Abbildung 3.18: Vorverarbeitete Beispielbilder. Oberste Reihe: Das Gauß'sche Rauschen steigt von  $\sigma = 0$  (links) auf  $\sigma = 250$  (rechts). Zweite Reihe: Die Stärke eines blendenden Lichts steigt von 0% einer Lichtkartenaddition (links) auf 100% (rechts). Dritte Reihe: sechs Verdeckungsmodalitäten. Untere Reihe: die kombinierten Störungen, von  $i = 0$  (links) bis  $i = 10$  (rechts). Veröffentlicht in [10].

## Bildrauschen

Um die Qualität der Methoden bei verrauschten Bildern zu bewerten, werden die Eingangsbilder mit Rauschen gemäß der Normalverteilung [265] erzeugt:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.4)$$

Im Rahmen dieser Anwendung beträgt der Mittelwert  $\mu = 0$  und  $\sigma$  wird schrittweise in einem Bereich von  $[0, 250]$  mit einer Schrittweite von 25 erhöht, so dass insgesamt elf Bilder mit jeweils unterschiedlichem Rauschpegel aus jedem Videobild erzeugt werden. Pseudocode 9 legt die Prozedur des Verrauschens der Bilddaten dar.

---

### Algorithmus 9 Bildrauschen

---

```
1: procedure VERRAUSCHEN(Eingabebild  $I$ , Rauschstärke  $\sigma$ )
2:    $w, h \leftarrow \text{Eingabebild.Breite}, \text{Eingabebild.Höhe}$ 
3:   Ausgabebild  $O \leftarrow \text{Eingabebild } I$ 
4:   Rauschen  $\leftarrow \text{Normalverteilung}(\mu = 0, \sigma)[w, h, 3]$ 
5:    $O+ = \text{Rauschen}$ 
6:    $O \leftarrow \text{Begrenzung}(O_\sigma, 0, 255)$ 
7: end procedure
```

---

## Licht

Um das Verhalten unter verschiedenen Lichtverhältnissen zu untersuchen, wird ein blendender Lichteffekt auf einer glänzenden Oberfläche simuliert. Hierzu wird ein Blendlichteffekt, implementiert in [266], auf das statische Hintergrundbild der Kameras angewandt und das ursprüngliche Hintergrundbild subtrahiert, so dass nur die in der Abbildung 3.19 gezeigten Lichtkarten erhalten bleiben. Anschließend werden diese Lichtkarten zu jedem von der jeweiligen Kamera aufgenommenen Bild hinzugefügt, mit einer Stärke von 0% bis 100% in Schritten von 10%, so dass insgesamt elf Blendlichteffektbilder pro Videobild entstehen. Pseudocode 10 illustriert das Hinzufügen von Blendlichteffekten.

---

### Algorithmus 10 Blendlichteffekt

---

```
1: procedure BLENDEN(Eingabebild  $I$ , Lichtkarte  $L$ , Stärke  $S$ )
2:   Ausgabebild  $O \leftarrow \text{Eingabebild } I$ 
3:    $O+ = L \cdot S$ 
4:    $O \leftarrow \text{Begrenzung}(O, 0, 255)$ 
5: end procedure
```

---



Abbildung 3.19: Obere Zeile: Hintergrundbilder der Kameras. Untere Zeile: Die entsprechenden Lichtkarten, welche auf die Kamerabilder addiert werden. Veröffentlicht in [10].

## Verdeckung

Um eine teilweise Bedeckung menschlicher Körper zu simulieren, werden die manuell beschrifteten Begrenzungsrahmen der Körper im Bild mit Blöcken aus Rauschen überzogen, insbesondere um die Qualität der Fußpunktterkennung bei entsprechend verdeckten Körperteilen zu beurteilen. Hierzu werden alle Begrenzungsrahmen in einem Bild gleichzeitig in sechs Modalitäten abgedeckt: den horizontalen linken, mittleren und rechten Teil sowie den vertikalen oberen, mittleren und unteren Teil. Jede Bedeckungsmodalität umfasst ein Drittel des Begrenzungsrahmens entlang der jeweiligen Achse. Innerhalb jedes Begrenzungsrahmens wird der entsprechend verdeckte Bereich mit einem Rechteck aus schwarzen und weißen Pixeln abgedeckt, welche nach dem Zufallsprinzip entsprechend der Normalverteilung erzeugt werden; dem sogenannten Salz- und Pfefferauschen. Es wurden keine künstlichen Objekte zur Verdeckung in das Bild eingefügt, da bereits natürliche Verdeckung durch Tische und andere Hindernisse in den Videodaten enthalten sind. Pseudocode 11 zeigt das teilweise Verdecken mit Rauschblöcken.

---

**Algorithmus 11** Verdeckung

---

```
1: procedure VERDECKEN(Eingabebild  $I$ , Begrenzungsrahmen  $B$ , Modus  $M$ )
2:   Ausgabebild  $O \leftarrow Eingabebild I$ 
3:   for  $(y_{min}, x_{min}, w, h) \in B$  do
4:      $Rauschen_x = SalzUndPfefferrauschen(w, \frac{h}{3})$ 
5:      $Rauschen_y = SalzUndPfefferrauschen(\frac{w}{3}, h)$ 
6:     switch  $M$  do
7:       case Links
8:          $O[x_{min} : x_{min} + w, y_{min} : y_{min} + \frac{h}{3}] \leftarrow Rauschen_x$ 
9:       case Mitte
10:       $O[x_{min} : x_{min} + w, y_{min} + \frac{h}{3} : y_{min} + 2\frac{h}{3}] \leftarrow Rauschen_x$ 
11:      case Rechts
12:       $O[x_{min} : x_{min} + w, y_{min} + h - \frac{h}{3} : y_{min} + h] \leftarrow Rauschen_x$ 
13:      case Oben
14:       $O[x_{min} : x_{min} + \frac{w}{3}, y_{min} : y_{min} + h] \leftarrow Rauschen_y$ 
15:      case Zentrum
16:       $O[x_{min} + \frac{w}{3} : x_{min} + 2\frac{w}{3}, y_{min} : y_{min} + h] \leftarrow Rauschen_y$ 
17:      case Unten
18:       $O[x_{min} + w - \frac{w}{3} : x_{min} + w, y_{min} : y_{min} + h] \leftarrow Rauschen_y$ 
19:    end for
20: end procedure
```

---

**Kombinierte Störungen**

Um einen anspruchsvollen Härtefall auswerten zu können, werden die Störungen durch Rauschen, blendendes Licht und Verdeckung zu einer vierten Modalität kombiniert. Um eine kombinatorische Explosion zu vermeiden werden nur elf Bilder pro Originalbild zu erzeugt, in dem in gleichteiligen Anstiegen hinzugefügt werden; für  $i \in \{0 \dots 10\}$  wird Rauschen der Stärke  $\sigma = 25 \cdot i$  und ein Blendlichteffekt der Stärke  $(10 \cdot i)\%$  auf das Bild angewandt. Ebenfalls wird eine Kombination von Begrenzungsrahmenverdeckungen hinzugefügt, indem jede Person im unteren Drittel in horizontaler Richtung und entlang ihrer vertikalen Mittelachse verdeckt wird.

Die untere Zeile der Abbildung 3.18 zeigt die Ergebnisse dieses kombinierten Störungsprozesses, Pseudocode 12 skizziert den algorithmischen Ablauf derer Erzeugung.

---

#### Algorithmus 12 Kombinierte Bildstörung

---

```

1: procedure STÖREN(Eingabebild  $I$ , Lichtkarte  $L$ , Begrenzungsrahmen  $B$ )
2:   Ausgabebild  $O \leftarrow Eingabebild I$ 
3:   for  $i = 0; i \leq 10; i+ = 1$  do
4:      $O_{i+} = Verrauschen(I, i \cdot 25)$ 
5:      $O_{i+} = Blenden(I, L, \frac{i}{10})$ 
6:      $O_{i+} = Verdecken(I, B, Unten)$ 
7:      $O_{i+} = Verdecken(I, B, Mitte)$ 
8:   end for
9: end procedure

```

---

### 3.3.2 Modelle

Untersucht werden Gradientenorientierungshistogramme (HOG) [267, 256, 268, 260, 269, 270], die neuronale Objektdetektionsarchitektur *You Only Look Once* (YOLO) [258, 51, 52] und das System *OpenPose* (OP) zur Erkennung humanoider Körperhaltungen [255, 259, 261]. Abbildung 3.20 zeigt exemplarische Ausgaben dieser Modelle auf typischen Bildern des Datensatzes. Bei der Erkennung durch HOG und YOLO fehlt Vorwissen über den Aufbau menschlicher Körper und somit die Abschätzung versteckter Körperteile, so dass die Position des Fußpunktes eher ungenau bleibt. Die exemplarische Posenschätzung durch OP zeigt eine gute Antizipation der versteckten Körperteile; so kann der Fußpunkt der Person, welche hinter einem Hindernis steht, zuverlässig lokalisiert werden, wo die anderen Methoden lediglich den Oberkörperteil erkennen würden.



Abbildung 3.20: Exemplarische Erkennungen der drei untersuchten Methoden auf verschiedenen Bildern der simulierten Arbeitsaufgaben aus [10]. Links: OP. Mitte: YOLO. Rechts: HOG.

## Gradientenorientierungshistogramm

Die Gradientenorientierungshistogrammmethode (E: *Histogram of Oriented Gradients*, HOG) liefert Merkmalsbeschreibungen eines Bildes anhand der Orientierungen der Kanten und Linien im Bild. Beliebige Modelle des maschinellen Lernens können dieses Histogramm dann als Merkmalsvektor nutzen, um eine Objekterkennung durchzuführen, zum Beispiel zur Erkennung menschlicher Körper [256]. Wie von [269] ausgearbeitet, berechnet dieses Verfahren zunächst den Gradienten eines Bildes, teilt dann das Bild in benachbarte, nicht überlappende Zellen auf und berechnet für jede Zelle die Gradientenorientierungen, welche abschließend in einem Histogramm zusammengeführt werden. Die Gruppierung dieser Histogramme in größere Blöcke ergibt ein verkettetes Blockmerkmal  $b$  und ermöglicht eine Normalisierung der Blockmerkmale nach der euklidischen Norm:

$$b = \frac{b}{\sqrt{\|b\|^2 + \epsilon}} \quad (3.5)$$

Das Verfahren verkettet dann die normalisierten Blockmerkmale zu einem einzigen HOG-Merkmal, welches dann wieder normalisiert wird. Auf der Grundlage dieses Merkmals kann ein Lernalgorithmus Objekte im Bild erkennen oder klassifizieren, wie es von [267] ausgeführt wird. Für die Personenerkennung in der simulierten Schwerindustrieumgebung wurde die OpenCV-Implementierung mit einer Support-Vektor-Maschine (SVM) verwendet [271, 272].

## You Only Look Once

Das Objekterkennungsnetzwerk *You Only Look Once* (YOLO) führt eine mehrstufige Objekterkennung auf Farbbilddaten durch. YOLO läuft in Echtzeit und kann Menschen sowie alle anderen Objekte, die es gelernt hat, im Bild erkennen und räumlich eingrenzen. Die neuronale Architektur besteht im Wesentlichen aus einer Kaskade von Faltungs- und Max-Pooling-Schichten mit unterschiedlichen Filtergrößen und Pooling-Bereichen. Die Ausgabeschicht dieses Netzwerks ergibt ein Raster von  $S \times S$  Zellen; für jede Zelle prognostiziert die Ausgabeschicht eine Klassenwahrscheinlichkeit und eine Anzahl von Begrenzungsrahmen mit einem Vertrauenswert [52]. Das Verfahren multipliziert die bedingten Klassenwahrscheinlichkeiten und die individuellen Vertrauenswerte, erhält einen klassenspezifischen Vertrauenswert für jede Zelle. Als Grundlage für die Zielfunktion während des Lernvorgangs wird eine Wahrscheinlichkeitsverteilung genutzt, welche auf der Vereinigungsmengenüberschneidung (E: *Intersection over union*, IOU) zwischen vorhergesagten Begrenzungsrahmen und der durch die Daten gegebene Grundwahrheit beruht:

$$P(\text{Klasse}|\text{Objekt}) \cdot P(\text{Objekt}) \cdot \text{IOU}_{\text{Vorhersage}}^{\text{Wahrheit}} = P(\text{Klasse}) \cdot \text{IOU}_{\text{Vorhersage}}^{\text{Wahrheit}} \quad (3.6)$$

Diese neuronale Architektur erfährt fortlaufend Verbesserungen [273, 51, 274, 275]. Die Auswertung für diese Herausforderung nutzt die erste Version, implementiert in [258].

## **OpenPose**

Die neuronale Architektur OpenPose (OP) schätzt zuverlässig die Posen des menschlichen Körpers, indem sie Körpermodellwissen verwendet [276, 255, 259, 261]. Abhängig von der Wahl des Körpermodells, zur Auswahl stehen Körper der Datensätze BODY25 [255], COCO [277] und MPI [278], kann die Laufzeit- und Posenschätzqualität in insgesamt vernünftigen Qualitätsbereichen variieren. OP schätzt Körperhaltungen beliebig vieler Personen in einem Bild auf zweidimensionaler Ebene und kombiniert mehrere Stufen der erlernten Körperteilaffinitätsfelder (E: *Part Affinity Fields*, PAF) und Körperteilkonfidenzkarten (E: *Part Confidence Maps*, PAC) mit einem zweiteiligen Graphenabgleichsverfahren [255]. Das tiefe künstliche neuronale Netzwerk lernt dann, Körperteile mit Individuen im Bild zu assoziieren, kodiert diese in einem globalen Kontext und lernt Teilepositionen und deren Zuordnung abzuschätzen. Die Echtzeitanwendbarkeit ermöglicht es, mehrere Personen in einer Echtzeitvideoübertragung zu erkennen und diese Informationen für eine präzise Fußpunktpositionsbestimmung zu verwenden.

### **3.3.3 Auswertung**

Ein Personenerkennungssystem muss unterschiedliche Anforderungen erfüllen. Erstens müssen die Schwierigkeiten in industriellen Umgebungen antizipiert werden; so müssen etwa Bildstörungen durch Staub, Lichtreflexionen und bewegliche Objekte Berücksichtigung finden, welche die Person vor der Sichtung durch Kameras verbergen können. Zweitens soll eine akkurate Translation der menschlichen Position in ein Weltkoordinatensystem durchgeführt werden. Dies setzt eine präzise Ermittlung der Fußpunktposition voraus. Drittens soll ein solches System ökonomische Anforderungen erfüllen, also in Echtzeit auf möglichst kostengünstiger Hardware laufen. Zur Bewertung der Zuverlässigkeit eines Detektionssystems für menschliche Körper werden drei Qualitätsfaktoren bestimmt: die Genauigkeit bei der Erkennung des menschlichen Körpers an sich, die Genauigkeit der Fußpunktlokalisierung und die Tendenz zu Fehlern, die entweder nicht vorhandene Personen erkennen oder Personen im Bild nicht erkennen. Die Qualitäten der Modelle werden auf einer Reihe von simulierten Arbeitsvideosequenzen ausgewertet, welche programmatisch mit zunehmendem Rauschen, Lichtreflexionen und teilweiser Verdeckung variiert werden.

Zur Erfüllung dieser Anforderungen können die zuverlässigsten Ergebnisse von der neuronalen Architektur OpenPose erwartet werden, da sie Vorwissen über menschliche Körpermodelle mit einbezieht und somit verschiedene Störungen antizipieren kann. Schwierigkeiten werden bei der Fußpunktterkennung durch mittels HOG und YOLO erwartet, da diese nur Begrenzungsrahmen von sichtbaren Körperteilen erkennen können. Von HOG wird, unter guten Bildbedingungen, eine akzeptable Genauigkeit erwartet; aber auch ein rapides Fallen der Genauigkeit bei fortschreitenden Störeffekten. Diese drei Methoden werden bezüglich der drei Anforderungen unter unterschiedlichen Bedingungen, wie in Abbildung 3.26 dargestellt, und ihren Rechenressourcenverbrauch, wie in Tabelle 3.4 aufgeführt, bewertet. Bewertet werden die Erkennungsqualitäten über die Genauigkeit von Personen- und Fußpunktterkennung sowie die allgemeine Tendenz zu Fehlern.

Die Anzahl fehlerhafter Entdeckungen  $f$  beträgt  $f < 0$  für nicht erkannte Personen (Falsch-Negativ) und  $f > 0$  für fälschlicher Weise erkannte nicht existente Personen (Falsch-Positiv). Die Anzahlen der Falsch-Positive und Falsch-Negative werden addiert. So bedeutet ein Wert von  $f = 0$ , dass die Methode dazu neigt, genau so viele Falsch-Positive zu produzieren wie Falsch-Negative. Die gelabelten Farbbilddaten beinhalten Begrenzungsrahmen für die Rümpfe und Füße der Personen im Bild. Um eine Bewertung für Leistungsindikatoren zu erhalten, wird untersucht, wie genau die Methoden die korrekten Positionen der Personenrümpfe und deren Fußpunkte abschätzen. Hierzu wird gezählt, wie oft die vorhergesagten Schlüsselpunkte in den Begrenzungsrahmen liegen. Die Anzahl dieser Erfolgsfälle gibt Auskunft über die Güte der Detektion. In Bezugnahme auf die, von den HOG- und YOLO Modellen vorhergesagten, Begrenzungsrahmen werden die Mittelpunkte der Begrenzungsrahmen aus den Grundwahrheitsdaten für die Personen- und Fußpunktabschätzung genutzt, um eine Aussage über die Abweichung der Erkennungen treffen zu können. Zur Auswertung des OpenPose-Systems wurde das menschliche Körpermodell BODY25 mit Standardparametern verwendet. Von den Schlüsselpunkten dieses menschlichen Körpermodells wird der Halsschlüsselpunkt für die Personenpositionsbestimmung und die geometrische Mitte zwischen dem rechten und linken Fußknöchelschlüsselpunkt für die Bestimmung der Fußpunktpositionen verwendet. Wie in der Abbildung 3.26 gezeigt, hat das OpenPose-System einen deutlichen Vorteil in Bezug auf die Antizipation von Rauschen, Blendlichteffekten und teilweise abgedeckten Körperteilen. Insbesondere bei der Berücksichtigung der Fußpunktpositionen zeigt sich der Vorteil des zugrundeliegenden Körpermodellwissens. Bezuglich der Teilabdeckung wird festgestellt, dass der YOLO-Detektor insbesondere solche Personen nicht im Bild erkennt, die entlang der vertikalen Mittelachse verdeckt sind. Betrachtet man die Tendenzen zu fehlerhaften Erkennungen, kann man feststellen, dass OpenPose durchschnittlich dazu neigt, fälschlicherweise so viele Personen zu sehen, wie es sie nicht sieht. YOLO hingegen tendiert eher dazu, Personen im Bild nicht zu finden. Bei dem Härtefall, welcher alle Störungen kombiniert, fallen alle Methoden schnell in ihrer Qualität, doch auch hier übertrifft OpenPose alle anderen Methoden. Ausgehend von der Auswertung der detaillierten Einzelergebnisse deutet die Tendenz zu fehlerhaften Erkennungen darauf hin, dass OpenPose Personen hinter dem Stuhlstapel falsch erkennt. Es überrascht nicht, dass die HOG-Methode bei allen Bildstörungen am schlechtesten abschneidet.

	HOG	YOLO	OP
FPS	33.5	18.4	11.3
GPU-Speicher	251 MB	1293 MB	1313 MB

Tabelle 3.4: Geschwindigkeit in Bildern pro Sekunde (FPS) und Rechenressourcenverbrauch der drei untersuchten Methoden aus [10].

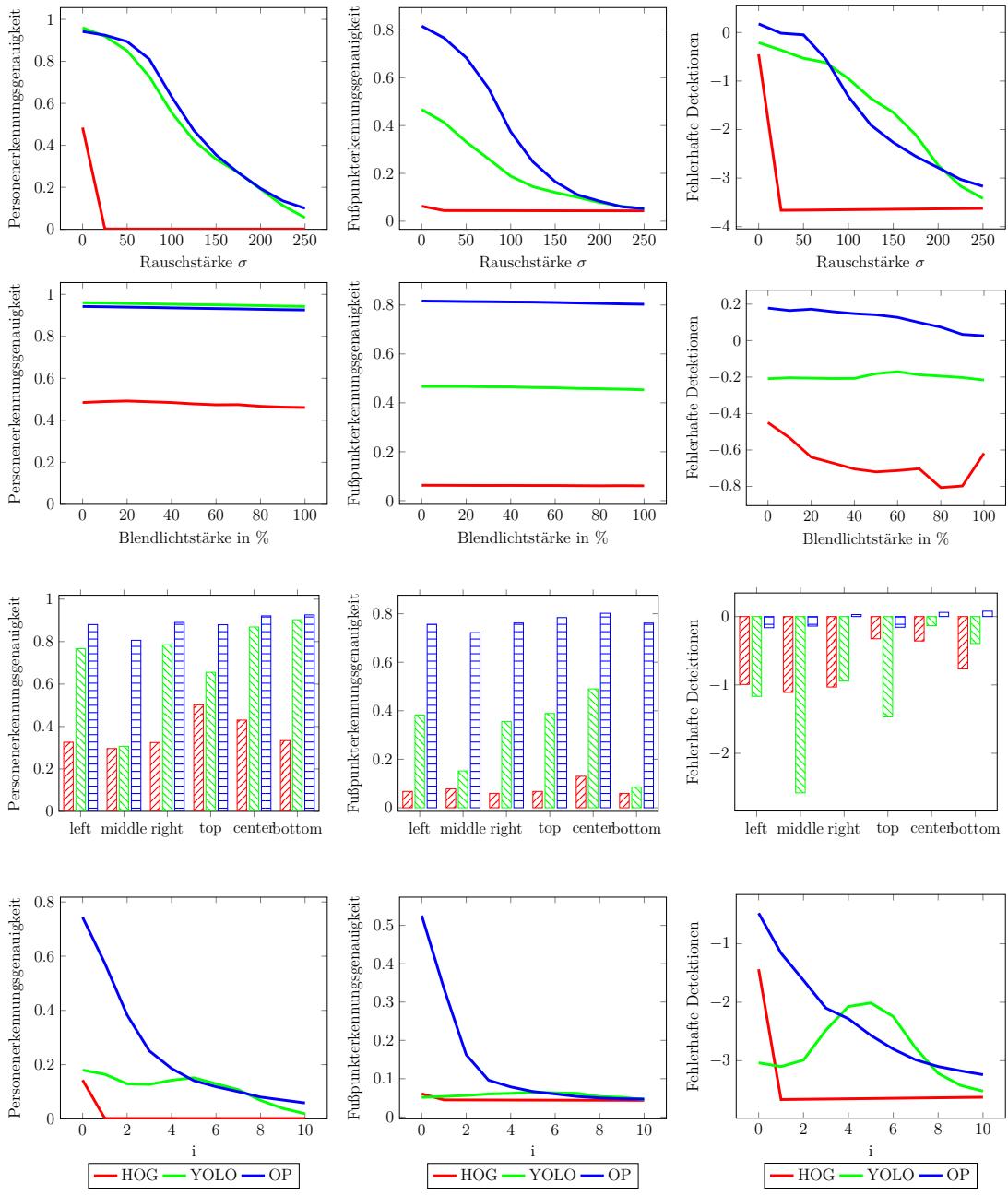


Abbildung 3.21: Obere Reihe: Beständigkeiten gegen Rauschen. Zweite Reihe: Antizipation des Blendlichteffektes. Dritte Reihe: teilweise verdeckte Personen in den sechs Modalitäten. Untere Reihe: der kombinierte Härtefall. Linke Spalte: Genauigkeit der Personenerkennung. Mittlere Spalte: Genauigkeit der Fußpunkterkennung. Rechte Spalte: Fehlerkennungstrend. Veröffentlicht in [10].

### 3.3.4 Diskussion

Die Untersuchungen in dieser Anwendung konzentrieren sich auf die Erkennung von Menschen über RGB-Kameras. Von den untersuchten Detektoren erweist sich OpenPose als am robustesten gegenüber Teildeckungen, verrauschten Bildern und blendenden Lichtverhältnissen. Dennoch kann die YOLO-Architektur nützliche Zusatzinformationen zur Szene liefern, die beispielsweise verwendet werden können, um nicht-menschliche Objekte zu finden. Die HOG-Methode kann sich in Fällen mit wenig Rechenaufwand und perfekten Bildverhältnissen als nützlich erweisen. In den Auswertungen wurden die Standardparameter aller untersuchten Methoden verwendet; eine Feinabstimmung dieser Hyperparameter kann die Ergebnisse verändern. Was YOLO betrifft, so wurden nur die vortrainierten Gewichte aus der ersten Version verwendet. Im Zuge dieser Anwendung wurden weder ausgeklügelte Techniken zur Vervollständigung des menschlichen Körpers implementiert noch Informationen über die zeitliche Korrelation zwischen den Videobildern verwendet, sondern an jedem einzelnen Bild unabhängig gearbeitet. Dabei werden Vorteile verworfen, welche sich beispielsweise aus der Verarbeitung visueller Flussinformationen mit rekurrenten neuronalen Netzwerkarchitekturen ergeben können. Da keine verwandten Arbeiten gefunden werden konnten, welche Herausforderungen für die Erkennung von Menschen in schwerindustriellen Umgebungen über Kamerabilder mit einem entsprechenden Datensatz beschreibt, kann kein belegbarer Vergleich darüber getroffen werden, wie gut der Datensatz andere reale Szenarien simuliert hat.

Ansätze des Transferlernens im Bereich der Arbeitssicherheit helfen, bereits gewonnenes Wissen wiederzuverwenden, um den Rechenbedarf zu senken und die Generalisierungsfähigkeit zu erhöhen [31, 32]. Der Einsatz von *Transfer Meta Learning* kann sinnvoll sein, um die Modelle auf die konkrete industrielle Situation abzustimmen, wenn genügend Daten über das Transferlernen von Objektdetektoren gesammelt wurden. Da nach aktuellem Wissensstand kein solcher Datensatz bekannt ist und im Rahmen des Projektes *Damokles 4.0* [12] kein solcher erstellt werden konnte, bleibt das Erstellen dieses Datensatzes und die Untersuchung mit *Transfer Meta Learning* an dieser Stelle Gegenstand zukünftiger Forschung.

## 3.4 Personenverfolgung

Ausgehend von der Personendetektion auf Kamerabildern eröffnet sich die Frage nach der Verfolgung dieser auf einem Videodatenstrom. Der aktuelle Stand der Forschung untersucht Personenverfolgungstechniken unter verschiedenen Gesichtspunkten mit unterschiedlichen Ansätzen. Zum Beispiel schlägt [279] einen Personenverfolgungsalgorithmus für ein autonomes unbemanntes Luftfahrzeug vor. Bei diesem Ansatz verfolgt eine Drohne mit Überwachungskamera einzelne Personen, was ein sehr flexibles Überwachungssystem mit dem Vorteil einer einfachen Gesichtserkennung ermöglicht. Im Vergleich zu einem stationären, kamerabasierten Ansatz erscheint das autonome Luftfahrzeug für eine industrielle Anwendung unpraktisch, da die fliegende Drohne mit beweglichen Objekten wie etwa Kränen, Fahrzeugen oder sogar anderen Personen kollidieren kann.

Bezüglich Personenidentifizierungstechniken konzentriert sich die aktuelle Forschung auf Methoden des Transferlernens [154, 155, 280, 281, 282]. Diese Methoden ermöglichen die Wiedererkennung von Einzelpersonen und zeigen, dass das Transferlernen die Erkennungsleistung mit sehr wenigen Trainingsdaten erhöhen kann. Für die Anwendung in der Schwerindustrie wurde aus Datenschutzgründen entschieden, Mitarbeiterinnen und Mitarbeiter über die an sie gebundenen, tragbaren Geräte zu identifizieren, anstatt deren Gesichter zu erkennen. Um das Problem der Verfolgung mehrerer Personen zu lösen, verwendet [283] die langsame Merkmalsanalyse (E: *Slow Feature Analysis*, SFA) [284]. Der Beitrag von [285] stellt einen Berechnungsrahmen für die Interpretation von Personentrackingdaten dar, der aus vier Modulen zur Verfolgung von Sofort- und Kurzzeitmerkmalen sowie unbeaufsichtigten und überwachten maschinellen Lerntechniken für höhere Abstraktionsebenen besteht.

Der Beitrag von [11] untersucht Möglichkeiten der Verfolgung von Personen auf Kamerabildern auf Basis der in Sektion 3.3 untersuchten Personendetektionsverfahren mit dem Ziel, einen Beitrag für ein digitales, kontextualisiertes Gebäudemanagementsystem im Rahmen des Projektes *Damokles 4.0* zu leisten. Digitale Gebäudemanagementsysteme helfen, die Produktivität, die Arbeitssicherheit und die Transparenz des Produktionsprozesses zu erhöhen [12, 16, 286, 287, 288, 289]. So können, beispielsweise im Bereich der Arbeitssicherheit, die Arbeitnehmerinnen und Arbeitnehmer in einem Notfall den kürzesten und sichersten Weg zum Ausgang angezeigt bekommen. Beispielsweise kann in einem Stahlwerksstandort ein unsicherer, direkter Weg durch einen potenziell gefährlichen Bereich führen, beispielsweise ein frisch gewalztes, heißes Stahlblech, dessen Temperatur in der Stresssituation nicht korrekt eingeschätzt werden kann. Das Bewegen auf diesem Blech kann dazu führen, dass die Stiefel des Arbeitenden mit dem heißen Stahl verschmelzen, was nicht nur eine Gefahr für das Leben des Arbeiterenden bedeutet, sondern auch mit hohen Kosten für das Unternehmen verbunden ist. Ein intelligentes Arbeitssicherheitssystem kann eine solche Situation verhindern, indem es mit geeigneten Mitteln die richtige Warnung zur richtigen Zeit bereitstellt. Ein solches System kann Kameras und andere mobile Vorrichtungen verwenden, um Personen in gefährlichen Situationen zu lokalisieren und zu identifizieren. Dazu müssen solche Systeme in Echtzeit so viele Informationen wie möglich über den Produktionsprozess sammeln und analysieren, in einen Kontext stellen und die richtigen Aktionen vorschlagen.

Auf einem zuverlässigen Personenerkennungssystem aufbauend wird ein System zur Verfolgung von Personen in der schwerindustriellen Umgebungen mit mehreren Kameras gesucht. Mit Hilfe des künstlichen neuronalen Netzwerks *OpenPose* werden die Fußpunkte der Personen auf jedem Kamerabild individuell lokalisiert und über eine entsprechende dreidimensionale Koordinatentransformation in ein Weltkoordinatensystem überführt, welches ein digitaler Zwilling der Fabrikräumlichkeiten ist. Mit Vorkenntnissen über die Kameraeinstellungen in der Umgebung genügt ein einfaches regelbasiertes System, um zu beurteilen, welche Sensordetections verschmelzen sollen. Durch die Anwendung der Kalmanfilterung wird die Verfolgung stabilisiert. Mit einer variablen Bildstapelgröße kann das nachfolgend erläuterte Verfahren die Genauigkeit erhöhen, wenn es mit zusätzlichen Rechenressourcen ausgestattet ist, indem es mehr Einzelbilder in der selben Zeit verarbeiten kann.

Als Datengrundlage wurde ein weiteres Szenario der Schwerindustrie simuliert, das aufgezeichnete Videomaterial und die durch AR-Brillen aufgezeichneten Positionsdaten dienen als Grundlage für die Auswertung [12, 286]. In diesem speziellen Anwendungsfall werden Kamerabilder verwendet, um Fußpunktlokalisierungen durchzuführen und diese Erkennungen einem umfangreichen Kontextmodell zur Verfügung zu stellen. Die Kenntnis der Standorte und Rollen des Mitarbeiters innerhalb des Produktionsprozesses ermöglicht es einem Kontextmodell, individuelle Informationen anzuzeigen. In den untersuchten Szenarien befragen AR-Geräte das Kontextmodell, um ihrem aktuellen Benutzer unterstützende Ratschläge zu geben [12, 286]. So stellt beispielsweise ein Informationsdienst einer AR-Brille relevante Daten über die aktuelle Aufgabe des Arbeiters zur Verfügung, wie in Abbildung 3.22 dargestellt.



Abbildung 3.22: Beispielhafte *Augmented Reality*, abhängig von der aktuellen Position des Nutzenden. Links: eine Anweisung zum Überprüfen einer Seriennummer. Rechts: Anzeige einer Evakuierungsroute im Notfall. Entnommen aus [11].

Auf diese Weise erhält ein Mitarbeiter kontextsensitive und rollenspezifische Informationen, beispielsweise über den Zustand einer Maschine, zusammen mit einer konkreten Arbeitsanweisung. Bezuglich des *DamokleS 4.0* Projekts [12, 15] beschreibt [264] die gesamte Softwarearchitektur, die dem Kontextmodell [290] zugrunde liegt. Außerdem skizziert [264] die wesentlichen Ideen, welche die Testszenarien bestimmen, sowie die damit verbundenen Prozesse für die Implementierung in mobile Geräte. Die vorgeschlagenen Szenarien betreffen sowohl die Arbeitssicherheit als auch Produktions- und Wartungsanwendungen. Der vorgeschlagene Ansatz bietet kontextbezogene Unterstützung für die Mitarbeiterinnen und Mitarbeiter der Fabrik in all diesen Szenarien. Für die Kontexterkennung schlägt [264] die Verwendung von Sensoren für mobile Geräte und externe Sensoren vor, die im Fabrikgebäude montiert sind, beispielsweise Kameras. In dem Beitrag von [10] wurde eine Vielzahl von Erkennungsmethoden evaluiert und herausgefunden, dass das OpenPose-System [255, 259, 261] die Anforderungen am Besten erfüllt, da es eine äußerst zuverlässige Fußpunkterkennung bietet, selbst unter schwierigen Bildbedingungen.

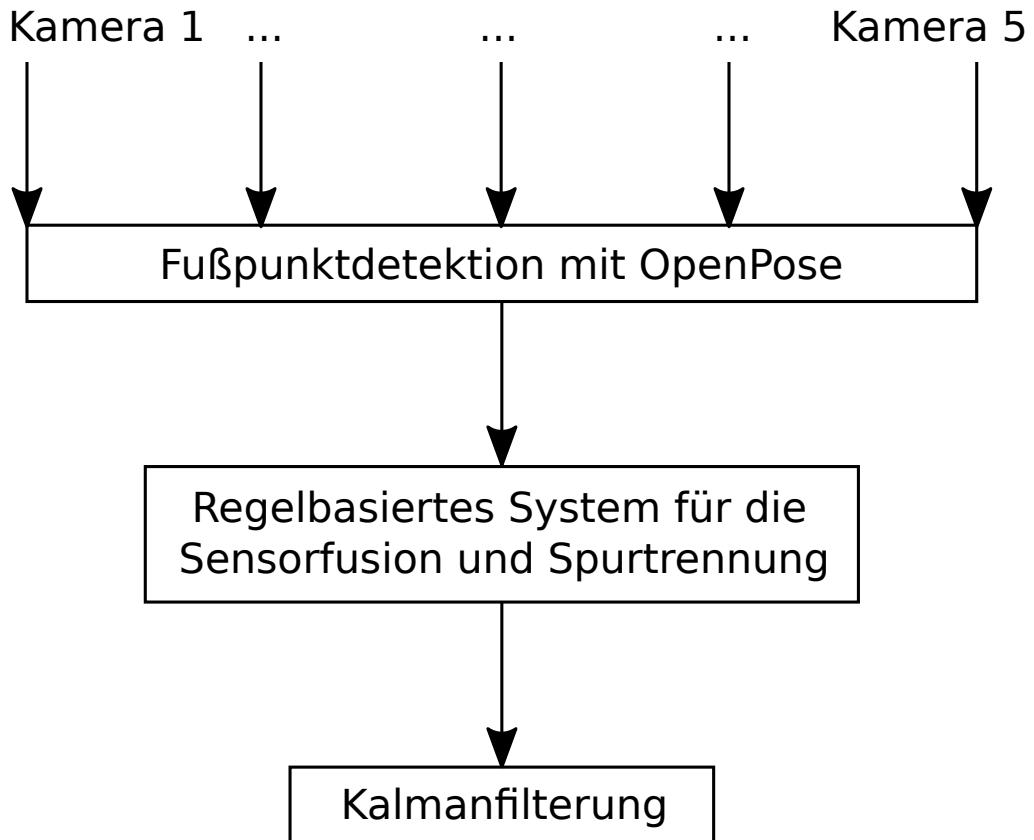


Abbildung 3.23: Das Programmablaufdiagramm des Personenverfolgungsverfahrens aus [11]. Die obere Reihe zeigt Bilder aus dem Datensatz dieser Studie. Von links nach rechts: Kameras  $C_1$  bis  $C_5$ , wie in Abbildung 3.24 dargestellt. Die Person bewegt sich derzeit in Sichtweite der Kameras  $C_1, C_2, C_5$ , aber außer Sichtweite der Kameras  $C_3$  und  $C_4$ .

Anhand der referenzierten Entwicklungen lässt sich feststellen, dass das Zusammenspiel der gesammelten Daten und der sich ständig weiterentwickelnden Algorithmen ein großes Potenzial zur Verbesserung der industriellen Prozesse und des Arbeitsalltags birgt. Die in dieser Anwendung eingesetzte Softwarearchitektur besteht aus drei verschiedenen Modulen, welche in Echtzeit auf einem Videodatenstrom aus mehreren Kameras arbeiten; der Fußpunktterkennung, der Sensorfusion und der Kalmanfilterung. Abbildung 3.23 zeigt das Programmablaufdiagramm der einzelnen Verarbeitungsschritte.

Ausgehend von einem Fußpunkterkennungssystem, für das die OpenPose-Architektur [255, 259, 261] verwendet wird, liefert eine Koordinatentransformation von Bildkoordinaten in Weltkoordinaten die Eingabe für das zweite Modul, einen regelbasierten Sensorfusionsansatz. Das regelbasierte System bereitet auch die Spuren für das dritte Modul, einen Kalmanfilter, vor, indem es die Linienführungen zusammenführt. Alle Verarbeitungsschritte finden auf dem gleichen Bildstapel statt und garantieren die Echtzeitfähigkeit im Kompromiss zwischen Stapelgröße und verfügbaren Rechenressourcen; je mehr Bilder das Programm sieht, desto genauer wird die Personenverfolgung. Die Aufrechterhaltung der Echtzeitfähigkeit hängt nur von den verfügbaren Computerressourcen ab. Als Eingabe wird ein Stapel von  $k$  Bildern pro Kamera präsentiert, wobei  $k$  so gewählt wird, dass das Programm so schnell wie nötig und so genau wie möglich läuft. Die Erhöhung von  $k$  führt zu genaueren Erkennungen auf Kosten einer höheren Rechenzeit. Als Ergebnis liefert das Programm die aktuellen Personenstandorte in Weltkoordinaten, welche an das Kontextmodell auf einem anderen Server gesendet werden können. Das Kontextmodell kann die Standorte mit anderen Daten verknüpfen, beispielsweise, um die verfolgten Personen über intelligente Geräte zu identifizieren.

### 3.4.1 Versuchsaufbau

Zur Aufzeichnung der Bewegungsdaten wurde ein Parcours aufgebaut, welchen die 48 Testpersonen abliefen, während sie den Anweisungen auf der AR-Brille Folge leisteten. Abbildung 3.24 zeigt den Aufbau des Testparcours und die Kamerapositionen. Wie in den Beispielbildern in Abbildung 3.23 ersichtlich, tragen die Testpersonen AR-Brillen und Sicherheitswesten. Sie bewegen sich immer in Sichtweite einer bestimmten Teilmenge der Kameras. Während die AR-Brille die Position in Weltkoordinaten aufzeichnet, zeichnen die Kameras Farbbilder auf. Die Kamerasätze ( $C_1, C_2, C_5$ ) und ( $C_3, C_4$ ) beobachten verschiedene Bereiche der simulierten industriellen Umgebung, wie in der Abbildung 3.24 dargestellt. Jeder Kamerasatz hat einen eindeutigen Kalibrierungsursprung, anhand dessen alle Kameras sowie der Start- und Endpunkt des Parcours über einen Übersetzungsvektor in Beziehung zum Weltkoordinatensystem gesetzt werden können.

### 3.4.2 Fußpunkterkennung in Weltkoordinaten

Das erste Modul der Softwarearchitektur führt eine Fußpunkterkennung und eine Koordinatentransformation auf einem Videodatenstrom aus. Bevor die Fußpunkterkennung durchgeführt wird, werden die Kamerabilder durch einen adaptiven Histogrammausgleich mit einer Kachelrastergröße von acht mal acht Pixeln verbessert [291]. Anschließend werden die Fußpunkte der Person in Kamerabildkoordinaten lokalisiert, unter Verwendung der entsprechenden Fußschlüsselpunkte des COCO-Modells, wie sie von OpenPose [255, 259, 261] bereitgestellt werden. Die ermittelten Fußpunkte in Kamerabildkoordinaten werden über eine geometrische Transformation in Weltkoordinaten übersetzt; hierzu bedarf es einer einmaligen Kalibrierung der Kameras. Der Kalibrierungsprozess geht davon aus, dass sich die Person auf einer ebenen Fläche bewegt, daher wird während der Transformation eine temporäre konstante Höhenkoordinate  $z = 1$  verwendet, welche

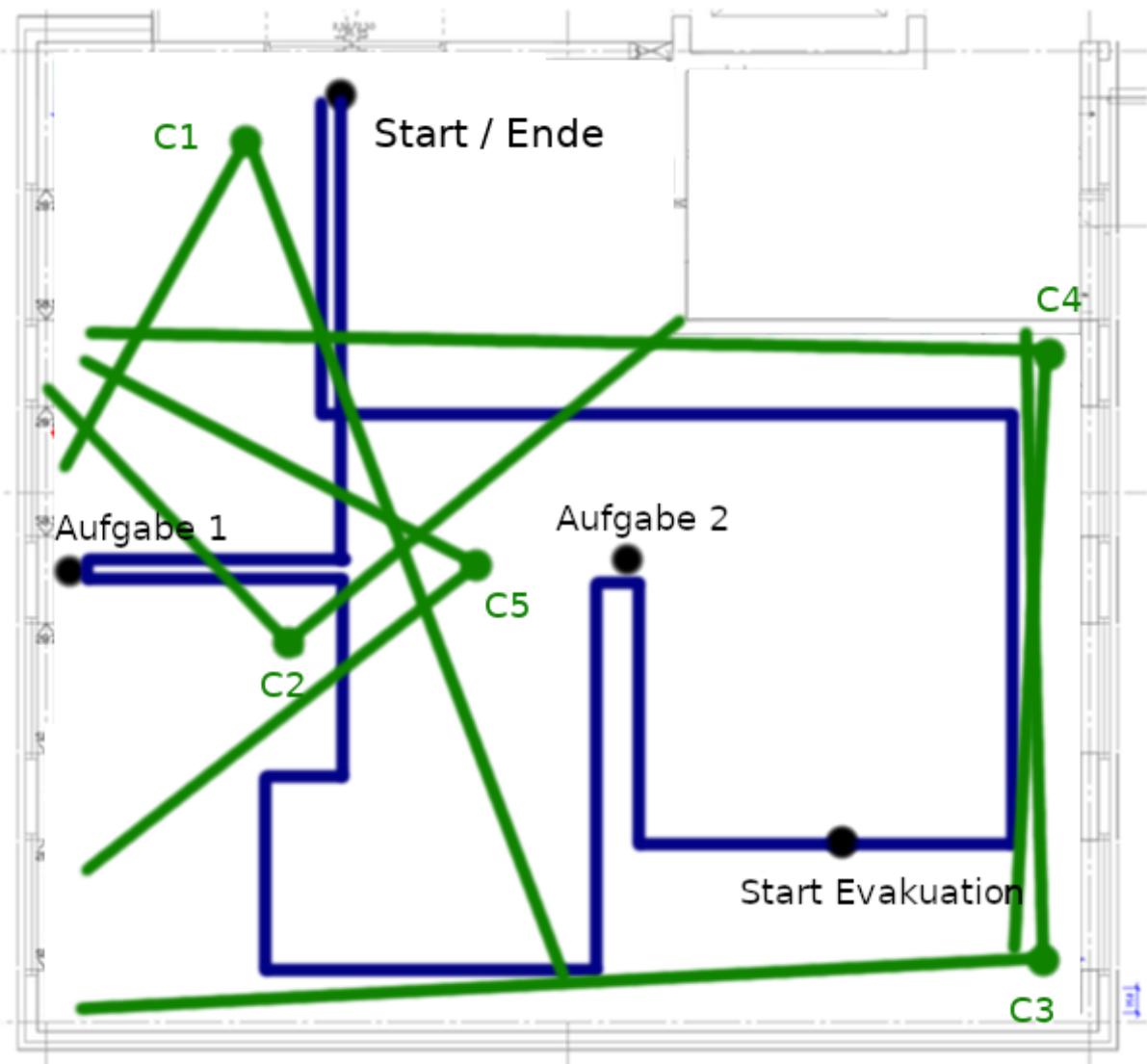


Abbildung 3.24: Der Testparcour der simulierten industriellen Umgebung. Die Testpersonen folgen einem vorgeschriebenen Weg (blau) und lösen eine Reihe von Aufgaben. Fünf Kameras (grün) nehmen währenddessen Videomaterial auf. entnommen aus [11].

nach der Transformation wieder verworfen wird. Die Übersetzung der Kamerakoordinaten  $(x, y, 1)^T$  in Weltkoordinaten  $(p_x, p_y)^T$  nutzt die intrinsische Kameramatrix  $M$ , die Rotationsmatrix  $R$  und den Übersetzungsvektor  $d$ . Der Translationsvektor  $d$  wurde über einen Standardkalibrierprozess mit Schachbrettmustern [292, 293] ermittelt. Mit dem Aufbau einer Hilfsmatrix

$$R' = \begin{pmatrix} R_{0,0}, R_{0,1}, R_{0,2} \\ R_{1,0}, R_{1,1}, R_{1,2} \\ d_0, d_1, d_1, d_2 \end{pmatrix} \quad (3.7)$$

wird eine Koordinatentransformation wie folgt durchgeführt:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} (R' M)^{-1} \quad (3.8)$$

Die auf diese Weise berechneten Weltkoordinaten beziehen sich auf den Ursprung des Schachbrettmusters. Da verschiedene Kameras unterschiedliche Teile der Umgebung erfassen, werden entsprechend viele extrinsische Kalibrierungen durchgeführt. Die Weltkoordinaten werden dann über die Entfernungsvektoren zwischen den verschiedenen Koordinatenursprüngen zueinander in Beziehung gesetzt.

### 3.4.3 Sensorfusion

Die rohen Erkennungen aus dem Lokalisierungsmodul werden im darauffolgenden Modul mit dem Vorwissen über die Kamerapositionen zusammengeführt. Da die Fußpunktlokalisierung aber nicht komplett fehlerfrei alle Schlüsselpunkte erkennt, stellt sich das Problem einzelner fehlender Erkennungen auf wenigen Bildern. Um das Problem dieser fehlenden Erkennungen, durch beispielsweise Rauschen, zu lösen, wird eine Autovervollständigung innerhalb der  $k$  Bilder durchgeführt. Wenn innerhalb eines Farbbildes keine Fußpunkt erkannt wurde, aber im vorherigen und darauffolgenden Bild, dann wird die Position als geometrische Mittel der beiden erfolgreichen Messungen berechnet. Somit kann die Erkennung mit einem Stapel von  $k$  Bildern auf unterschiedlicher Hardware zuverlässig und echtzeitfähig geschehen, wenn ein optimaler Wert für  $k \geq 3$  gefunden wird. Da bekannt ist, auf welchem Bild eine Erkennung vorliegt, wird diese in eine zweidimensionale boolesche Matrix überführt, welche beschreibt, welche Kamera eine kohärente Erkennung innerhalb des Echtzeitfensters liefert. Mit Hilfe dieser Matrix entscheidet ein regelbasiertes System, welche Erkennungen miteinander zusammengeführt werden sollen. Dazu spiegelt ein festgeschriebener Regelsatz Vorwissen über den konkreten Kameraeinsatz in der Umgebung wider. Auf der Grundlage dieses Wissens wird eine Reihe von Konditionalsätzen angewandt, um zu entscheiden, wie die Fusion koordiniert wird; wenn beispielsweise die Kameras  $C3$  und  $C4$  die gleiche Person erkennen, wird ein geometrischer Mittelwert der beiden vorgeschlagenen Weltkoordinaten ermittelt.

### 3.4.4 Spur trennung

Zur Vorbereitung der Positionsdaten für die Kalmanfilterung wird ein Regelsatz aus Wenn-Bedingungen angewandt, welcher jede Position einer eindeutigen Spur zuordnet, indem Vorwissen über den Raum genutzt wird. Jede Spur verfügt somit über eine gleichmäßige Bewegung, was den Filtervorgang vereinfacht. Um die resultierende Spur zu glätten, wird ein Kalmanfilter nach [294] auf jeder der getrennten Spuren angewandt, wie in Abbildung 3.26 dargestellt. Zur Initialisierung des Kalmanfilters wird eine vierdimensionale gleichförmige Bewegungsdynamik angenommen, welche die Personenposition  $(p_x, p_y)^T$  sowie die aktuelle Geschwindigkeit  $(v_x, v_y)^T$  der Personen in Bezug auf einen festen Zeitschritt  $dt$ , entsprechend der Aufnahmefrequenz der Kamera, beschreibt:

$$F = \begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.9)$$

So dass:

$$\begin{pmatrix} p_x \\ p_y \\ v_x \\ v_y \end{pmatrix}_{t+1} = F \cdot \begin{pmatrix} p_x \\ p_y \\ v_x \\ v_y \end{pmatrix}_t \quad (3.10)$$

Zur Initialisierung der Kalmanfilterkovarianzmatrix wird die Einheitsmatrix verwendet. Für die Schätzung wird, über die  $k$  Bilder, jeweils die nachfolgende Positionen iterativ berechnet, damit die Echtzeitanwendbarkeit erhalten bleibt. Die Aufzeichnungen der Laborstudie werden verwendet, um den Ansatz zur Personenortung in einer simulierten industriellen Umgebung zu bewerten. In dieser Studie tragen die Testpersonen im Rahmen des Projektes *DamokleS 4.0* eine AR-Brille, welche sie durch einen Parcour führt [12]. Während dieses Kurses müssen sie drei Aufgaben lösen und im letzten Teil einem Evakuierungsweg zum Ausgang folgen, wie in Abbildung 3.24 dargestellt. Die ursprüngliche Nutzerstudie umfasste zwei verschiedene Navigationsmodalitäten und entsprechende Fragebögen, die darauf abzielen, die Gefühle und Einstellungen der Testperson zu dieser Technologie aus psychologischer Sicht zu bewerten. Für die Anwendung der Personenverfolgung werden nur die gesammelten Videoaufnahmen genutzt.

### 3.4.5 Auswertung

Die Kameras nehmen Videodaten mit acht bis zwölf Bildern pro Sekunde auf, so dass eine Bildstapel der Größe  $k = 4$  verwendet wird, um die Echtzeitfähigkeit mit der verfügbaren Hardware zu gewährleisten. Da die AR-Brille Positionen mit einer Rate von zwei Positionen pro Sekunde aufzeichnet, erzeugen die Kameras mehr Datenpunkt in der gleichen Zeit, da sie mit einer vier bis sechsfachen Frequenz aufnehmen. Abbildung 3.26 zeigt die kompletten Trajektorien, wie sie von der AR-Brille bereitgestellt werden, die rohen Positionsschätzungen nach der Kamerafusion, die getrennten Spuren

und die Endpositionen nach der Kalmanfilterung. Die Statistiken über die zurückgelegte Wegstrecke, Dauer und Geschwindigkeit, wie in Abbildung 3.27 dargestellt, ignorieren die unterschiedlichen zeitlichen Auflösungen, welche durch unterschiedliche Aufzeichnungsrraten verursacht werden. Abbildung 3.25 stellt die mittleren Abweichungen der geschätzten Kamerapositionen zu den Positionsdaten aus der Grundwahrheit dar. Hierbei tritt das Problem verschiedener zeitlichen Auflösungen auf und wird gelöst, indem für jede Kameraposition der räumlich und zeitlich nächstgelegene Punkt in den Positionen der AR-Daten gesucht wird. Der Durchschnitt der mittleren Abweichungen zwischen den rohen Kamerapositionsschätzungen und den Grundwahrheitsdaten der AR-Brille liegt bei etwa  $0,67m$ , während der Durchschnitt der mittleren Abweichungen zwischen den kalmangefilterten Endpositionen und den Grundwahrheitspositionen mit etwa  $0,59m$  etwas niedriger liegt. Dies entspricht den Trajektorien, die nach der Kalmanfilterung der Grundwahrheit näher kommen. Werden die Spuren in der Abbildung 3.26 betrachtet, so kann eine leichte metrische Verzerrung im Start- und Endbereich festgestellt werden, da nur die Kamera C2 diesen Bereich beobachtet. Bei der Betrachtung der Histogramme in der Abbildung 3.27 kann zudem gelesen werden, dass die zurückgelegten Wegstrecken und die durchschnittlichen Geschwindigkeiten nach der Kalmanfilterung der Verteilung der Grundwahrheitspositionen besser entsprechen. Darüber hinaus kann weisen die durchschnittlichen Geschwindigkeiten in den Evakuierungsspuren, also den Spuren neun bis zwölf, höhere Werte auf, da die Personen sich schneller bewegen.

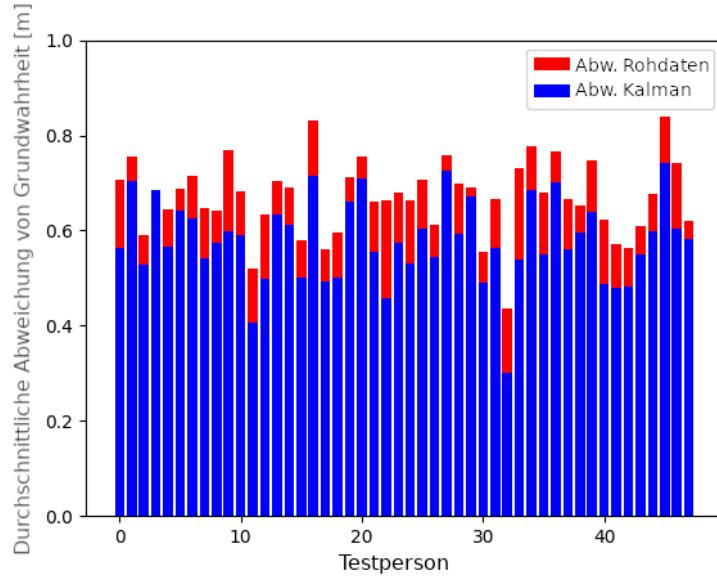


Abbildung 3.25: Die mittleren Abweichungen zwischen den geschätzten Kamerapositionen und den Grundwahrheitspositionen, welche von der AR-Brille für jede Testperson aufgezeichnet wurden. Roten Balken: die mittleren Abweichungen für die rohen Schätzungen. Blaue Balken: die mittleren Abweichungen für die kalmangefilterten Positionen. Entnommen aus [11].

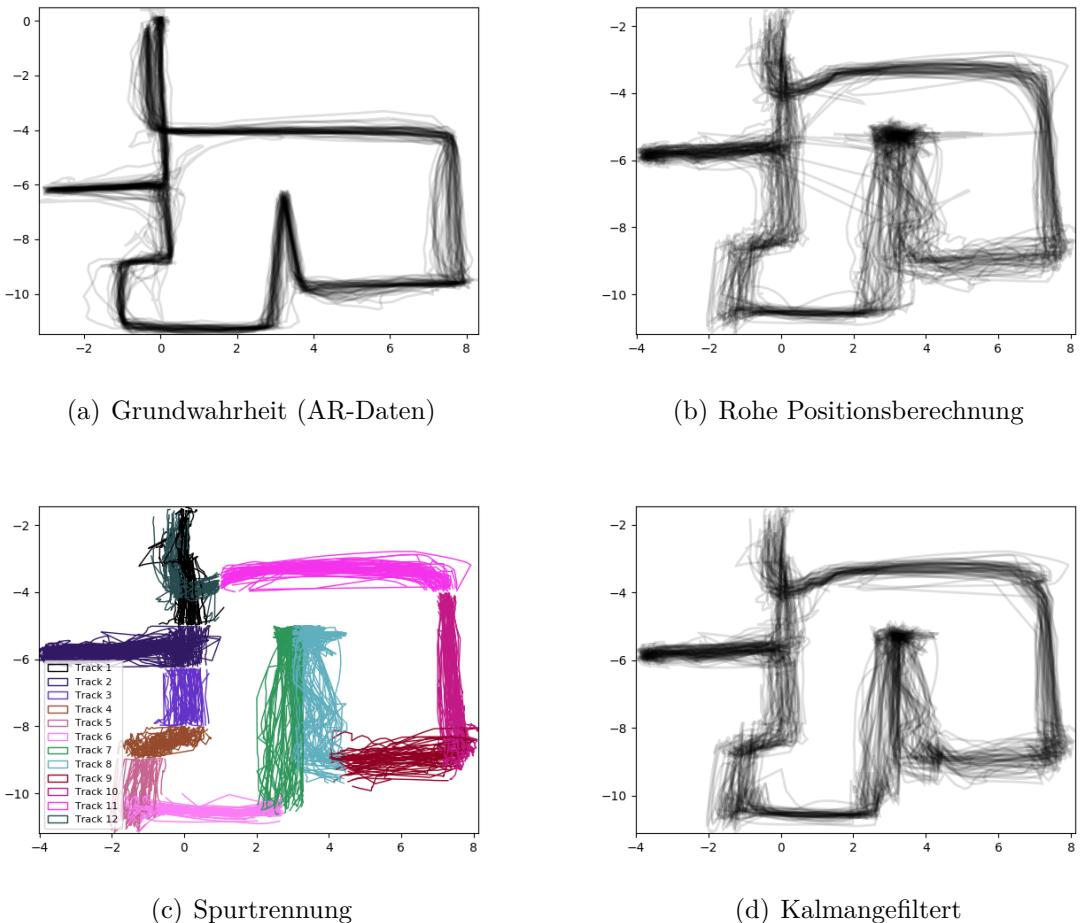


Abbildung 3.26: Die berechneten und gemessenen Spuren der Testpersonen, alle Skalen in Metern. Oben links: die Grundwahrheitsdaten, wie sie von den AR-Brillen bereitgestellt werden. Oben rechts: die Rohdaten, also die Spuren nach der Erkennung von Fußpunkten und der Anwendung des ersten regelbasierten Systems. Unten links: die Ergebnisse nach der Spurtrennung. Unten rechts: die Spuren nach der Kalmanfilterung. Diese ähneln weitgehend den Grundwahrheitspositionen. Es zeigt sich eine metrische Verzerrung; besonders der Bereich um den Start- und Endpunkt herum ist gestreckt, weil er von nur einer Kamera beobachtet wird. Entnommen aus [11].

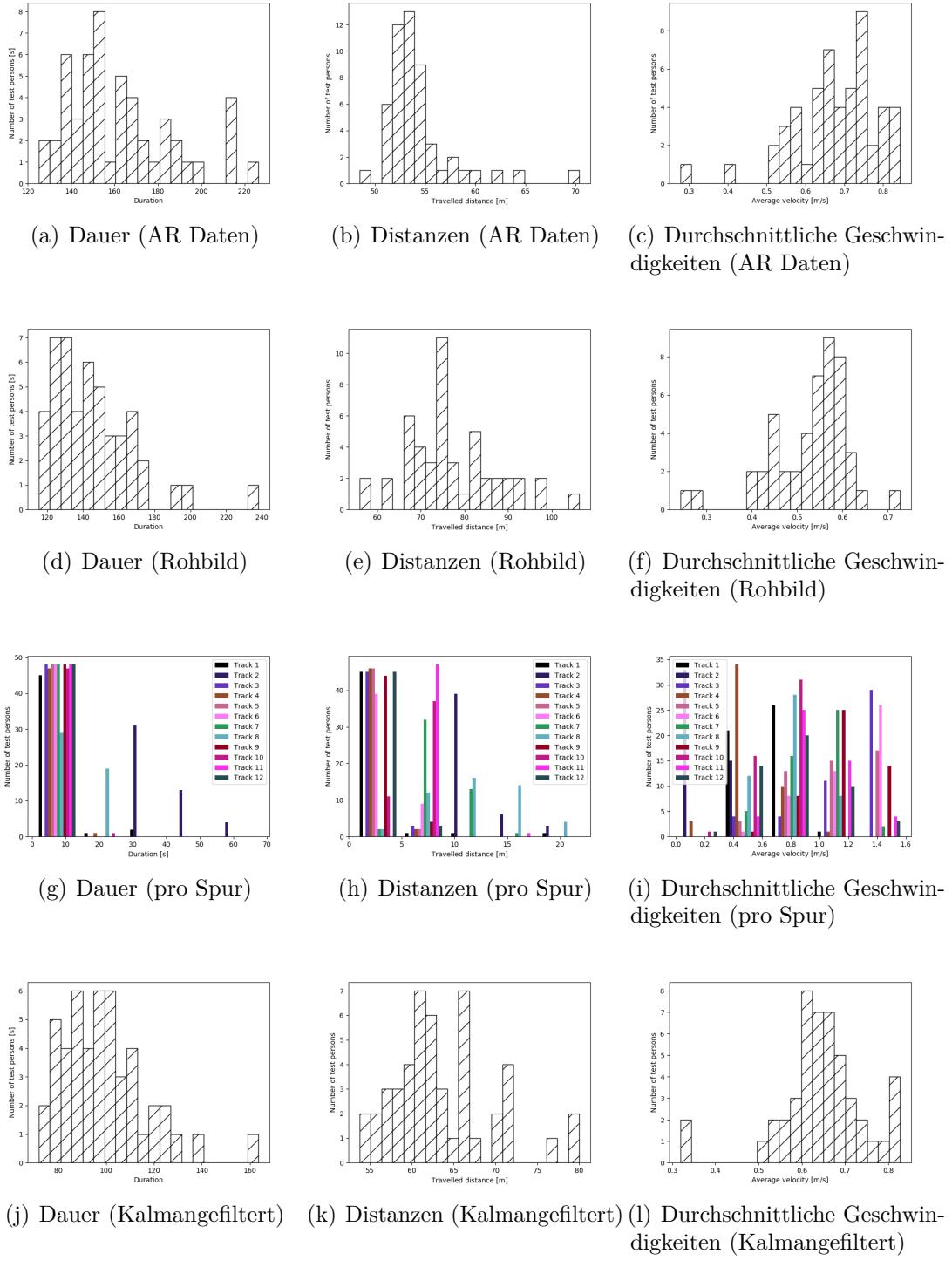


Abbildung 3.27: Von links nach rechts: Dauer in Sekunden, zurückgelegte Wegstrecken in Metern, durchschnittliche Geschwindigkeiten in Metern pro Sekunde. Von oben nach unten: AR-Daten, rohe Erkennungen, Statistiken pro einzelne Spur, die kalman- gefilterten Endergebnisse. Entnommen aus [11].

### 3.4.6 Diskussion

Aus der metrischen Verzerrung kann geschlossen werden, dass sich eine Person für eine zuverlässige Standortbestimmung in Sichtweite von mindestens zwei kalibrierten Kameras bewegen muss. Um genaue Verfolgungsergebnisse bei Kamerabildern zu erhalten, scheint die Verwendung zusätzlicher Mittel wie Kalmanfilter angebracht. Aus Datenschutzgründen wurde entschieden, Personen über ein mitgeführtes Smartphone in einem Kontextmodell zu identifizieren, anstatt Gesichtserkennung auf Kamerabildern zu verwenden. In diesen Auswertungen werden nur Standardparameter für alle Drittanbieterprogramme, wie das *OpenPose* Netzwerk und die Kamerakalibrierungstoolbox von *MatLab* verwendet. Die Änderung dieser Hyperparameter kann die Ergebnisse verbessern. Mehr Rechenressourcen ermöglichen es, bessere Ergebnisse zu liefern und gleichzeitig die Echtzeitfähigkeit zu erhalten, indem nur ein einziger Parameter, nämlich die Größe des Bildstapels, erhöht werden muss. Die Sensorfusion mit Hilfe eines regelbasierten Systems basiert auf Vorwissen, ermöglicht aber aufgrund ihres transparenten Regelsatzes einfache Änderungen. Mit dem Weltkoordinatenmodell wird die Annahme getroffen, dass sich die Personen auf einer ebenen Fläche bewegen, so dass keine unterschiedlichen Höhenstufen voneinander unterscheiden werden. Der regelbasierte Ansatz ist für den Benutzer transparent und leicht zu ändern, verfehlt aber die Flexibilität, einfach auf andere, etwa dreidimensionale Weltkoordinatenkonfigurationen, übertragen werden zu können. Für die Spur trennung besteht die gleiche Herausforderung.

Um diesen Ansatz weiterzuentwickeln, könnte eine flexiblere Streckenzuweisung einen hohen Mehrwert beitragen. So kann beispielsweise ein Verstärkungslernagent lernen, Spurteile dynamisch zu öffnen und zu schließen. Auch könnte ein Kalmanfilter auf die Detektionen in Bildkoordinaten angewandt werden, um die Detektionen weiter zu stabilisieren. Mit Hilfe von Transferlernen könnte untersucht werden, mit welchem Aufwand die Modelle an ähnliche Situationen angepasst werden können.

## 3.5 Bildklassifikationstransfer

Die Lernprozesse tiefer künstlicher neuronaler Netzwerke als ein Teilbereich der künstlichen Intelligenz im Allgemeinen und des maschinellen Lernens im Besonderen, ermöglicht es Computern, Strukturen in beispielhaften Daten automatisch zu erkennen [6]. Transferlernen für die Klassifikation von Bilddaten mit Unterstützung von Grafikprozessoren wird in vielen Anwendungen immer beliebter und bietet viele Vorteile; zum Beispiel in medizinischen Anwendungen [295], bei der Fernerkundung von beispielsweise Waldbränden oder Landnutzungen [296] oder generell zur Auswertung von optischen Satellitenbildern [297], aber auch bei der Erkennung bedrohter Tierarten mit Mobilgeräten [36]. Üblicher Weise wird ein vielversprechendes vortrainiertes künstliches neuronales Netzwerk ausgewählt und auf eine konkrete Aufgabe angepasst. Bei der Auswahl eines vortrainierten Modells für Anwendungen der Bildklassifikation gibt es jedoch einige Fallstricke, wenn sich nur auf Genauigkeit konzentriert wird und andere wichtige Parameter, wie Trainings- und Abfragezeiten oder Speicherbedarfe, außer Acht gelassen werden.

Transferlernen mit auf Bilddaten vortrainierten Modellen, meistens großen faltenden neuronalen Netzwerken, wird in der Regel so durchgeführt, dass die auf großen Standarddatensätzen, wie etwa ImageNet, vortrainiert und dann für die neue Zielaufgabe nachtrainiert werden. Dies lässt sich leicht realisieren, indem beispielsweise bestimmte Schichten durch andere aufgabenspezifische Schichten ersetzt werden und nur in diesen eine Gewichtsänderung für die Zielaufgabe stattfindet. Die Leistung des resultierenden transfergelernten künstlichen neuronalen Netzwerks hängt jedoch von dem verwendeten vortrainierten Modell ab. So bieten unterschiedliche Frameworks, wie etwa PyTorch [298], verschiedene vortrainierte Modelle an, welche die Anwendung dieser Technik vereinfachen. Der Stand der Technik umfasst viele Architekturen mit jeweils eigenen Stärken und Schwächen, welche für Anwendungen mit faltenden neuronalen Netzwerken geeignet sind. Ohne auf die konkrete Wiederverwendung eines dieser Modelle einzugehen wird deutlich, dass es bei der Auswahl des Modells einen Spielraum gibt. Nach [37] können die Größe und die Ähnlichkeit des Ziel- und Quelldomäne einen Ausgangspunkt für die Auswahl eines vortrainierten Modells darstellen. ImageNet ist beispielsweise ein populärer Datensatz natürlicher Bilder, doch als Ausgangspunkt für beispielsweise Spektrogramme oder Audiodaten denkbar ungeeignet. Eine Analyse von [299] vergleicht Ansätze aus gesammelter Literatur bei einem Lernvorgang auf ImageNet und hat ermittelt, dass der Stromverbrauch unabhängig von der Stapelgröße und der Architektur ist, die Genauigkeit und Inferenzzeit in einem hyperbolischen Verhältnis stehen, die Energiebeschränkung eine Obergrenze für die maximal erreichbare Genauigkeit und Modellkomplexität darstellt und die Anzahl der Operationen eine zuverlässige Schätzung der Inferenzzeit bietet [299]. In ähnlicher Weise stellt [300] eine Auswertung vor, in welcher Modelle auf ImageNet anhand mehrerer Indizien verglichen werden, wie etwa Genauigkeit, Rechenkomplexität, Speicherverbrauch und Inferenzzeit.

Die Auswahl des besten vortrainierten Modells ist also ein vielschichtige Frage, welche in ihrer Tiefe gut verstanden werden muss, damit für eine Anwendung die geeigneten Einstellungen gefunden werden können. Als Vorstudie zum *Transfer Meta Learning* wurden umfangreiche Experimente zur Klassifizierung von fünf Datensätzen mit elf vortrainierten Modellen durchgeführt. Hierbei wurde ein Überblick über die getesteten Modelle und Datensätze verschafft, die Leistungen anhand verschiedener Metriken bemessen und Einblicke und nützliche Richtlinien für das Transferlernen mit vortrainierten Modellen zur Bildklassifikation gewonnen.

### 3.5.1 Datensätze und Modelle

Die Auswertung wird sowohl auf Standard- als auch auf Nicht-Standarddatensätzen durchgeführt, namentlich CIFAR10 [301], MNIST [302], Hymenoptera [303] und einem Smartphonebilddatensatz aus einer Vorstudie von [26]. Bei der Nichtstandardaufgabe Hymenoptera handelt sich um einen kleinen Bilddatensatz aus einem PyTorch Tutorial zum Transferlernen, welcher die Unterscheidung von Ameisen und Bienen als Klassifikationsaufgabe stellt, wie in Abbildung 3.28 dargestellt. Dieser besteht aus nur 245 Trainingsbildern und 153 Testbildern und ist somit der kleinste Datensatz dieser Anwendung. Bei dem Nichtstandarddatensatz mit Smartphonebildern handelt es sich mit 654

Bilder ebenfalls um einen relativ kleinen Datensatz mit zwölf Smartphonklassen, wie ausschnittsweise in Abbildung 3.28 zu sehen ist. Im Rahmen dieser Anwendung wird eine Datenerweiterung vorgenommen, um das Volumen des Datensatzes und die damit verbundene Generalisierungsfähigkeit zu vergrößern. Hierzu wurde eine Rotationsoperation in Kombination mit zunehmendem Rauschen angewendet; für die Rotationsoperation werden die Bilder um  $r \in \{45^\circ, 135^\circ, 225^\circ, 315^\circ\}$  gedreht, für die Rauschoperation wird Rauschen in Prozenten  $p \in \{10\%, 25\%, 50\%\}$  aus einer diskreten Gleichverteilung  $\{0 \dots 255 \cdot p\}$  hinzugefügt. Dies führt zu insgesamt zwölf Augmentierungsoperationen für jedes Bild, sodass eine Gesamtzahl von 8502 Bildern im augmentierten Datensatz enthalten ist, einschließlich der unveränderten 654 Originalbilder.



Abbildung 3.28: Beispielbilder aus dem Datensatz Hymenoptera [303] (oben) und dem Smartphonedatensatz [6, 26] (unten).

Die von PyTorch [298] bereitgestellten Modelle sind auf ImageNet [304] vortrainiert. Die Entwicklung neuer faltender neuronaler Netzwerkarchitekturen stellt einen fortlaufenden Prozess in der aktuellen Forschungslandschaft des maschinellen Lernens dar. Im Rahmen dieser Anwendung finden mehrere dieser Architekturen Verwendung, sodass eine kurze Beschreibung der Eigenheiten dieser Architekturen dem Verständnis zuträglich ist. Abbildung 3.29 fasst die Chronologie der Modelle zusammen.

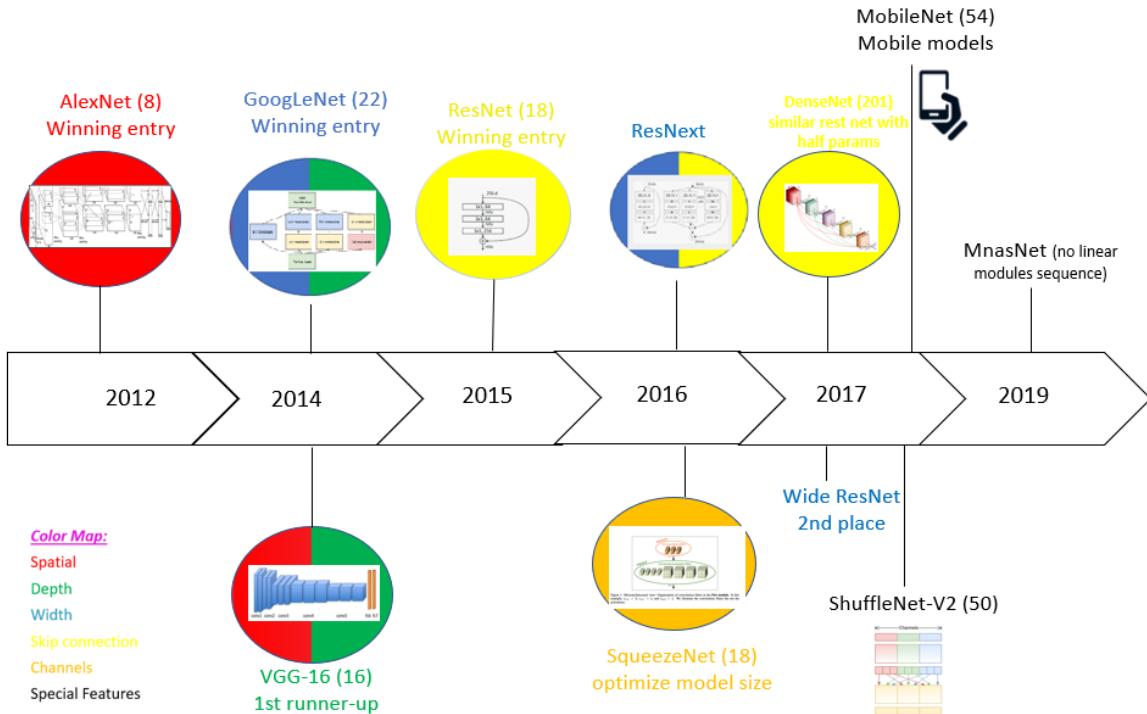


Abbildung 3.29: Chronologie der Entwicklung von tiefen faltenden neuronalen Netzwerken zur Bildklassifikation, entnommen aus [6].

## AlexNet

Das von [305] entwickelte Netzwerk erlangte Popularität, als es im Jahr 2012 die ImageNet Large Scale Visual Recognition Challenge gewann. Das Modell wurde auf etwa  $1,3 \cdot 10^6$  Bildern trainiert und verfügt über  $6,2 \cdot 10^7$  lernbare Parameter. Die Eingabeschicht besteht  $227 \times 227 \times 3$  Neuronen, gefolgt von fünf Faltungsschichten und drei vorwärtsverbundenen Schichten. Im Vergleich zu früheren Architekturen wurden nicht nur mehr und breitere Schichten verwendet, sondern auch eine Regularisierung über Dropout angewandt, um Überanpassung zu vermeiden. AlexNet verwendet standardmäßig die Aktivierungsfunktion *ReLU*.

## VGG -16

Die von der Visual Geometry Group (VGG) an der Universität Oxford entworfene Architektur wurde zu einem beliebten Modell, als es beim ILSVRC-Wettbewerb 2014 den ersten Platz belegt [306]. Dieses Modell erhöhte Tiefe des Netzwerks um eine hohe Genauigkeit zu erreichen. VGG-16 besteht aus insgesamt 13 Faltungsschichten und drei vorwärtsverbundenen Schichten. Die homogene Topologie des Netzwerks besteht aus fünfmaliger Wiederholung eines Musters von  $3 \times 3$  Filtern und  $2 \times 2$  Max-Pooling. Die Eingabeschicht wurde ursprünglich für RGB-Bilder der Größe  $224 \times 224$  entworfen. VGG verwendet die *ReLU* Aktivierungsfunktion und verbessert die Architektur gegenüber

AlexNet, indem es die Kernelgrößen  $11 \times 11$  und  $5 \times 5$  durch eine Kernelgröße von  $3 \times 3$  ersetzt. Dies zu höherer Auflösung in den Zwischenschichten aber auch einer höheren Anzahl von  $13,8 \cdot 10^7$  lernbaren Parametern führt. Das Kaskadieren mehrerer kleinerer Filter erhöht die Tiefe des künstlichen neuronalen Netzwerks. Bezuglich der Rechenkosten vergünstigt dies das Erlernen von mehr repräsentativen Merkmalen, als es bei größeren Filtern der Fall wäre. Der Erfolg von VGG beruht auf dem Kaskadieren kleinerer Kernel, was zur Vorlage beim Entwurf späterer Architekturen wird.

## Inception V-1

Nach dem Erfolg von VGG wurde Inception-V1, auch als GoogLeNet bekannt, von [307] aus vorgestellt. Inspiriert von LeNET [308] gewann diese Architektur die ILSVRC 2014 Klassifikationssherausforderung. Dieses Modell gründet auf dem Konzept, sowohl die Tiefe als auch die Breite des Netzwerks zu erhöhen; was jedoch aufgrund der erhöhten Anzahl von lernbaren Parametern und der umfangreichen Nutzung von Rechenressourcen zu einem Problem der Überanpassung führen kann. Um die Nachteile dieses Ansatzes zu überwinden, wurde das naive Inception-Konzept als Split-Transform-Merge eingeführt, welches aus drei Blöcken besteht. Ein grundlegender Faltungsblock verwendet abwechselnd  $1 \times 1$ -Filter und Max-Pooling-Schichten als Merkmalsextraktoren mit niedrigen Parametern. Ein Inception-Modul nimmt mehrere Extraktionen gleichzeitig vor; drei verschiedene Filtergrößen werden parallel innerhalb des gleichen Zweiges verwendet, um räumliche Informationen auf verschiedenen Skalen zu erhalten. Als dritter Block wurde ein Hilfsklassifikator eingefügt, um das Lernverfahren sowohl zu stabilisieren als auch zu beschleunigen. Insgesamt spart dieses Modell mit nur etwa  $5 \cdot 10^6$  lernbaren Parameter Ressourcen.

## ResNet18

Das Residual Network (ResNet) ist mit  $1,1 \cdot 10^7$  lernbaren Parametern der Gewinner des ILSVRC 2015 [309]. ResNet-18 verfügt über 18 Faltungsschichten und mit einer Kernelgröße von  $3 \times 3$ . Obwohl es sich bei ResNet um ein sehr tiefes künstliches neuronales Netzwerk handelt, welches mit zunehmender Anzahl versteckter Schichten grundsätzlich der Herausforderung verschwindender Gradienten gegenübersteht, liegt ein besonderer Vorteil bei dieser Architektur im systematischen Überspringen synaptischer Verbindungen. Dies wird über Restmodule realisiert, welche über Abkürzungen verfügen, die es erlauben, eine oder mehrere Schichten zu überspringen. Diese Verknüpfungen erleichtern den Gradientenabstieg und führen zu einem wesentlich schnelleren Training.

## SqueezeNet

SqueezeNet zeichnet sich dadurch aus, dass es die Genauigkeit von AlexNet mit 50 mal weniger Parametern beibehält [310]. Die Architektur wurde verbessert, indem  $3 \times 3$  Filter durch  $1 \times 1$ -Filter ersetzt wurden und die Anzahl der Eingangskanäle unter Verwendung von Zusammenführungsschichten verringert wurde. Durch das Einspeisen in eine Expansionsschicht wiederhergestellt wurde, welche eine Kombination aus  $1 \times 1$ -

und  $3 \times 3$  Filtern enthält, wurden die reduzierten Merkmale wiederhergestellt. Dieses Netzwerk erhöht die Anwendbarkeit von faltenden künstlichen neuronalen Netzwerken in Systemen mit begrenzten Ressourcen, wie etwa FPGAs oder eingebettete Systeme, indem es eine hohe Genauigkeit bei geringerem Rechenaufwand bietet.

## DenseNet

Aufbauend auf der Idee von ResNet entstand DenseNet mit einem schichtenübergreifendem Verbindungs muster, welches einen maximalen Informationsfluss gewährleistet [311]. Kürzere Verbindungen zwischen den Schichten in der Nähe des Signaleingangs und des Signalausgangs mildern das Problem verschwindender Gradienten, da das erneute Erlernen redundanter Merkmale verhindert wird. Diese synaptischen Verbindungsmatrizen werden so konstruiert, dass jede Schicht mit allen vorhergehenden Schichten verbunden wird, also jede Schicht die Merkmale aller vorhergehenden Schichten als Eingangssignale liest. Dies erhöht die Tiefe des Netzwerks, sodass DenseNet mit über  $2 \cdot 10^7$  lernbaren Parametern über den größten Parametervektor, der im Rahmen dieser Arbeit untersuchten künstlichen neuronalen Faltungsnetzwerke, verfügt. Es zeigt sich, dass diese dichten Verbindungen eine Regularisierung darstellen und der Überanpassung entgegenwirken.

## ResNext

Das Netzwerk ResNext erreichte den zweiten Platz der ILSVRC 2016 [312]. Bei dieser Architektur handelt es sich um eine verbesserte Version von ResNet, mit einer geringeren Anzahl von Parametern, welche bei gleichem Rechenaufwand die Genauigkeit erhöht. ResNext führt eine Kardinalitätsdimension ein, welche die Menge der Transformationen mit der gleichen Topologie darstellt und sich als effektiver als die Konstruktion eines vergleichbar tieferen und breiteren faltenden neuronalen Netzwerks erweist. Das Eingabesignal durchläuft mehrere eindimensionale Faltungen und dann eine Reihe von spezialisierten  $3 \times 3$  Filtern, bevor aus Ausgangssignal als Summe zusammengeführt wird.

## MobileNet

Smartphones und andere Geräte mit begrenzten Rechenressourcen eignen sich derzeit kaum bis gar nicht für tiefe neuronale Lernverfahren. MobileNet wurde entworfen, um den Einsatz dieser Verfahren auf mobilen Plattformen zu ermöglichen [313]. Die Nutzung von trennbaren Filtern durch den Einsatz eindimensionaler Faltungen motiviert mehrere Forscher; MobileNet liegt die Idee zu Grunde, tiefenweise trennbare Filter zu nutzen, welche die Standardfaltung in eindimensionale und tiefenweise Faltung faktorisieren. Die tiefenweise Faltung wendet einen einzigen Kernel für jeden Kanal an, der davon unabhängig auch mehrmals eindimensional gefaltet wird. Da die Kernel getrennt aufgeteilt werden verringert die Anzahl der Parameter auf nur  $3 \cdot 10^6$  für den Merkmalsextraktor, was weniger anfällig für eine Überanpassung ist, verfügt aber über etwa  $10^7$  Parameter im Klassifikatorteil. MobileNet stellt einen Kompromiss zwischen Genauigkeit und der Rechenkosten dar.

## **Wide ResNet**

Wide ResNet verbessert die ResNet-Architektur, indem es die Tiefe der Restnetzwerke zugunsten einer größeren Breite verringert [314]. Es verwendet Merkmale wieder und übertrifft die ResNet sowohl in Genauigkeit als auch in den Rechenkosten und zeigt damit, dass tiefere Netzwerke nicht notwendigerweise besser sind. Wide ResNet besteht aus 16 Schichten mit, je nach Version,  $6, 9 \cdot 10^7$  oder  $12, 7 \cdot 10^7$  lernbaren Parametern. Verglichen mit einem dünnen tiefen Netz mit 1.000 Schichten aber gleicher Parameterzahl zeigt sich, dass Wide ResNet um ein Vielfaches schneller lernt.

## **ShuffleNet**

Frühere Architekturen verbessern Genauigkeit durch Erhöhung der Anzahl der Schichten und Kanäle, was die Fließkommaoperationen pro Sekunde erhöht und in Bereiche  $\geq 10^9$  bringt; ShuffleNet reduziert diese Rechenkosten durch geschicktes Umstrukturieren der Operationen [315]. Anstatt viele Gruppenfaltungen übereinander zu stapeln, wie durch die Vorlage AlexNet, ermöglicht ShuffleNet die simultane Gruppenfaltung, um Eingabedaten aus verschiedenen Gruppen zusammenzuführen und alle Kanäle vollständig miteinander zu verknüpfen, anstatt zu komprimieren oder eine Darstellung in geringerer Bitanzahl zu verwenden. Indem es die Reihenfolge der Kanäle ändert und simultane Gruppenfaltung verwendet, übertrifft ShuffleNet seine Vorlage AlexNet mit einer etwa 13-fachen Beschleunigung bei ähnlicher Genauigkeit.

## **MnasNet**

Mobile Neural Architecture Search (MnasNet) wurde für mobile Geräte entworfen, welche trotz beschränkter Ressourcen schnell neuronale Modelle ausführen müssen [316]. Dabei folgt MnasNet der Vorlage MobileNet, verbessert aber die Genauigkeit und Rechenkosten mit dem Vorschlag eines faktorisierten hierarchischen Suchraums, welcher eine Vielzahl von Schichtstrukturen entlang des Netzwerks erlaubt. Das bedeutet, dass der Suchraum in verschiedene Blöcke unterteilt ist, wobei jeder Block einen bestimmten Satz von Schichten mit ähnlichen Abmessungen, Filtergrößen und Operationen aufweist. Durch die direkte Nutzung der Hardware des Smartphones rechnet MnasNet in Datenworte anstelle von Fließkomazahlen und bietet eine auf die Smartphonehardware optimierte neuronale Architektur.

### **3.5.2 Auswertung**

Die Lösung dieser Herausforderung besteht in einer Empfehlung von geeigneten Bildklassifikationsmodellen mit einem zugehörigen Transferlernverfahren, welche mit geringem Lernaufwand zu hoher Klassifikationspräzision führen. Hierzu wird eine Auswertung aller Kombinationen von Modellen und Aufgaben vorgenommen und deren Lernerfolge gemessen, um zu einem aussagekräftigen Ergebnis zu gelangen. Die elf vortrainierten Modelle werden in insgesamt vier Modalitäten auf alle fünf Aufgaben mit der Transferlernmethode der Feinabstimmung trainiert.

Die erste Transferlernkonfiguration iteriert in zehn Episoden, sodass jedes Beispielbild zehn mal gesehen wird. In der zweiten Konfiguration wird jeder Transferlernvorgang mit nur einer Episode ausgeführt, also jedes Bild aus dem Trainingsdatensatz nur einmal präsentiert. In jeder dieser beiden Konfigurationen wird sowohl die Feinabstimmung des designierten Klassifikatorteils des künstlichen neuronalen Netzwerks als auch des vollständigen Netzwerks auf jede Zielaufgabe ausgeführt. Verfügt ein Netzwerk über keinen expliziten Klassifikatorteil, so wird die letzte vollverbundene Schicht als solcher angenommen. Für die Auswertung wird zunächst eine einfache Betrachtung der genauesten Lösungen vorgenommen, aber auch weiterführende Metriken für eine detaillierte Untersuchung genutzt. Hierbei zeigt sich der Nutzen menschlichen Expertenwissens bezüglich die Kenntnis der Eigenheiten der einzelnen Modelle und deren Transferlerneigenschaften sowie der Bedarf nach einer systematischen Abbildung dieser Zusammenhänge über einen Metalernprozess.

### **Einfache Betrachtung**

Um eine einfache Auswahlempfehlung ableiten zu können, wird die Anzahl der überlegenen Testergebnisse der Architekturen gezählt. Durch einfaches Zählen derjenigen Fälle, in denen eine bestimmte Konfiguration die höchste Genauigkeit über beide Episodenstellungen und alle Aufgaben hinweg erreicht hat, findet sich die höchste Punktzahl für ein vollständig feinabgestimmtes *Densenet* mit drei Erfolgsfällen in Tabelle 3.5. Da die maximale Anzahl von Erfolgsfällen zehn beträgt, ergibt sich, dass die Nutzung einer vollständig abgestimmten *Densenet*-Architektur in nur 30% der Fälle zu einem optimalen Transferlernvorgang führt. Weiterhin berücksichtigt diese einfache Betrachtung nicht die hohe Anzahl der lernbaren Parameter dieser Architektur und die damit einhergehende hohe Trainingszeit, was die Suche nach aussagekräftigeren Transferlernempfehlungen eröffnet.

#Erfolge	Modell
3	<i>Densenet (voll)</i>
2	<i>Resnext50_32x4d (voll)</i>
1	<i>Resnet18 (voll)</i>
1	<i>Densenet (Klassifikator)</i>
1	<i>Resnext50_32x4d (Klassifikator)</i>
1	<i>Shufflenet (voll)</i>
1	<i>Googlenet (voll)</i>

Tabelle 3.5: Die Anzahl der Erfolgsfälle, in denen das Modell die höchste Punktzahl in der Genauigkeit über die fünf Aufgaben und zwei Konfigurationen der Episodenanzahl erreicht hat. Zehn Erfolgsfälle stellen das theoretische Maximum dar, die beste Lösung erreicht davon drei Erfolge.

## Detaillierte Untersuchung

Im Rahmen der Lösung dieser Herausforderung, also der Ableitung von zielführenden Transferlernprozessen, werden die Leistung der Modelle anhand von vier Maßstäben gemessen: die Genauigkeit  $G$ , die Parameterzahl  $n_p$ , die Genauigkeitsdichte  $\rho$  und die Trainingszeit  $t$ . Die Genauigkeit  $G$  misst, wie viele Beispiele korrekt klassifiziert wurden. Die Anzahl der Parameter  $n_p$  nennt die Komplexität des Modells, also die Zahl der während des Transferlernvorgangs veränderbaren synaptischen Gewichte. Die Genauigkeitsdichte  $\rho$  teilt die Genauigkeit durch die Anzahl der Parameter; je höher der Wert, desto effizienter ist das Modell im Hinblick auf die Nutzung der synaptischen Gewichte während des Transferlernvorgangs:

$$\rho = \frac{G}{n_p} \quad (3.11)$$

Zur Auswertung werden die Genauigkeiten  $G$  der Modelle in allen Aufgaben und die Trainingszeiten  $t$  in Sekunden gemessen, zusammen mit der Anzahl der lernbaren Parameter  $n_p$ . Die Anzahl der Parameter ist wichtig, um die Speichernutzung für jedes Modell zu bestimmen, also den zugewiesenen Speicher für das jeweilige Netzwerkmodell und die zugehörige Verarbeitung der Bilddatenstapel auf einer Grafikkarte.

Wie in Abbildung 3.30 dargestellt, zeigt sich bei Betrachtung der durchschnittlichen Genauigkeitsdichten über alle Datensätze, dass ein vollständig feinabgestimmtes SqueezeNet die höchste Genauigkeitsdichte aufweist. Dieses Ergebnis bestätigt die ursprüngliche Hypothese bei der Entwicklung von SqueezeNet, welche mindestens die Genauigkeit von AlexNet bei 50 mal weniger Parametern und weniger als 0,5 MB Modellgröße beibehält [310]. Die Ergebnisse bezüglich der Feinabstimmung der Klassifikatorteile, welche in Abbildung 3.30 dargestellt sind, unterscheiden sich geringfügig in Bezug auf die Dichte der Genauigkeit und der Reihenfolge der Modelle; jedoch zeigt sich auch ein großer Unterschied in Bezug auf die Genauigkeitsdichte von ResNet18. Die Ergebnisse der Lernprozesse mit zehn Episoden in Abbildung 3.30 zeigen, dass die Reihenfolge der Modelle in Bezug auf die Genauigkeitsdichten im Vergleich zu den Transferlernvorgängen mit nur einer Episode nicht beeinflusst wird und die Genauigkeitsdichten in geringem Maße höher sind. Abbildung 3.31 zeigt Zusammenhänge zwischen Genauigkeit, Modellgröße und Trainingszeit für alle Aufgaben und Modelle nach der Feinabstimmung der Gesamtparametervektoren, gemittelt über alle Aufgaben. Hierbei kann festgestellt werden, dass ResNet18 wieder ein zufriedenstellendes Ergebnis als das effizienteste Modell in Bezug auf die Genauigkeitsdichte liefert, also seine Parameter am effizientesten nutzt. Abbildung 3.32 zeigt die Modellgröße und die durchschnittliche Genauigkeit gegenüber der Trainingszeit für alle Aufgaben und Modelle, nachdem nur die Klassifizierungsschichten aller Datensätze feinabgestimmt wurden. Hierbei zeigt MnasNet die schlechteste Leistung, was es zum fehlertechnisch ungünstigsten Modell macht, aber es zeigt auch eine geringe Modellkomplexität und eine kurze Trainingszeit. Das komplexeste Modell in allen Untersuchungen ist VGG16, was sich in den Zahlen zur Genauigkeitsdichte widerspiegelt. Die detaillierten Ergebnisse aller vortrainierten Modelle auf allen Aufgaben stehen, für den Transferlernprozess mit einer Episode, in den Abbildungen 3.33 und 3.34.

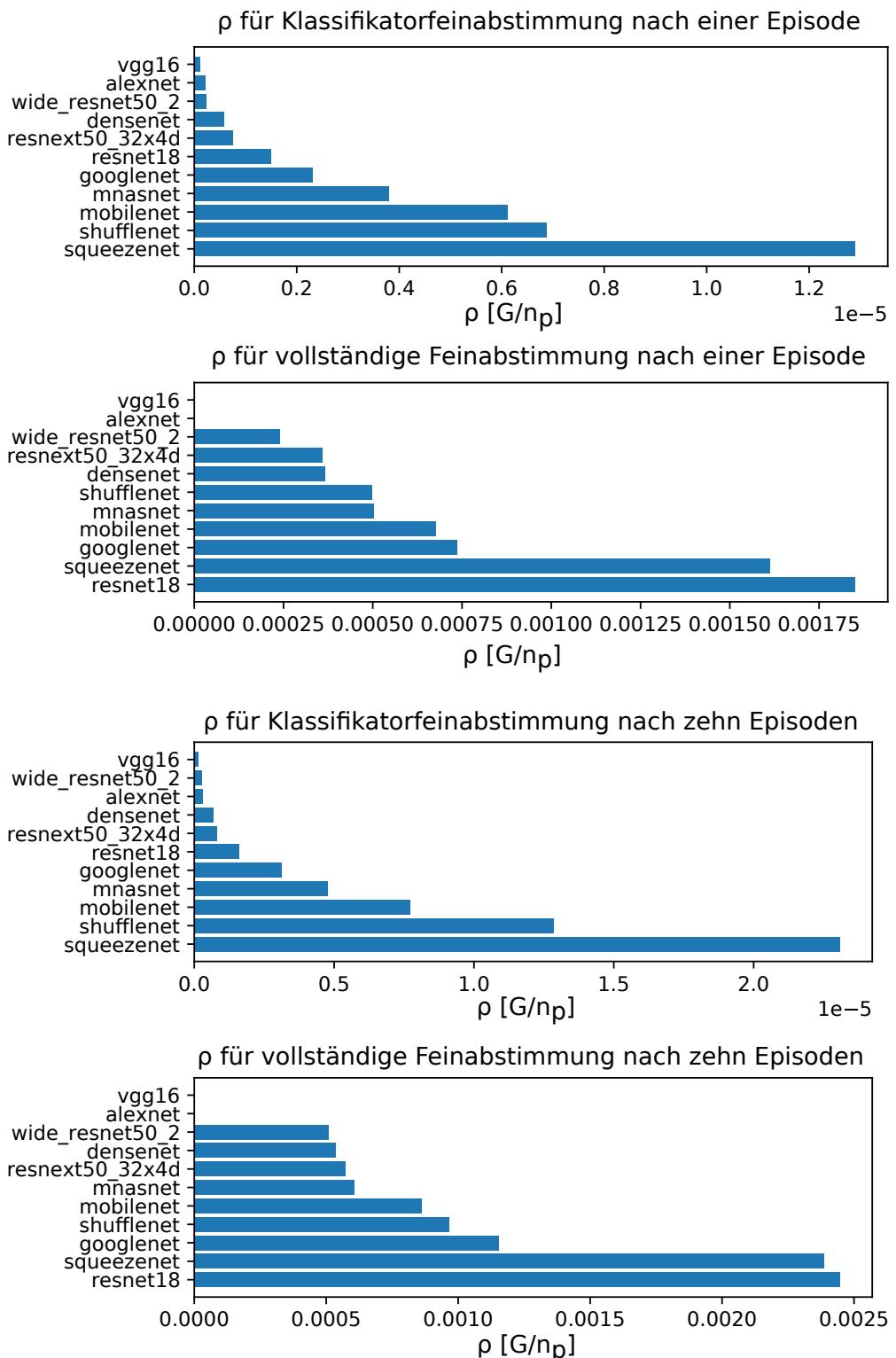


Abbildung 3.30: Durchschnittliche Genauigkeitsdichten nach einer Episode (oberen beiden) und nach zehn Episoden (unteren beiden), jeweils nach Feinabstimmung aller Schichten und der Klassifizierungsschichten, entnommen aus [6].

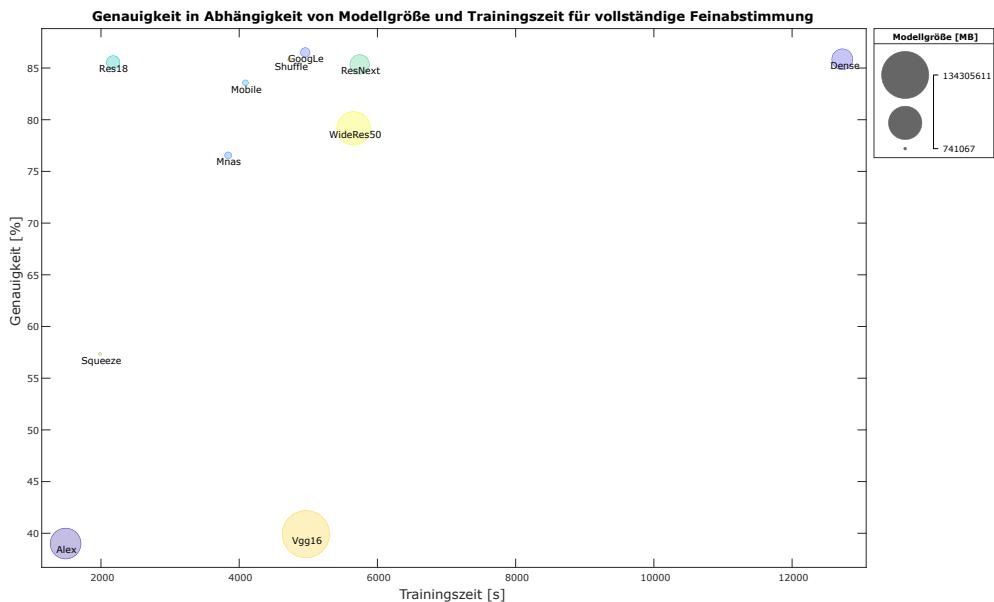


Abbildung 3.31: Modellgrößen und Genauigkeiten im Vergleich zur Trainingszeit für alle Aufgaben und Modelle nach der Feinabstimmung aller Schichten, entnommen aus [6].

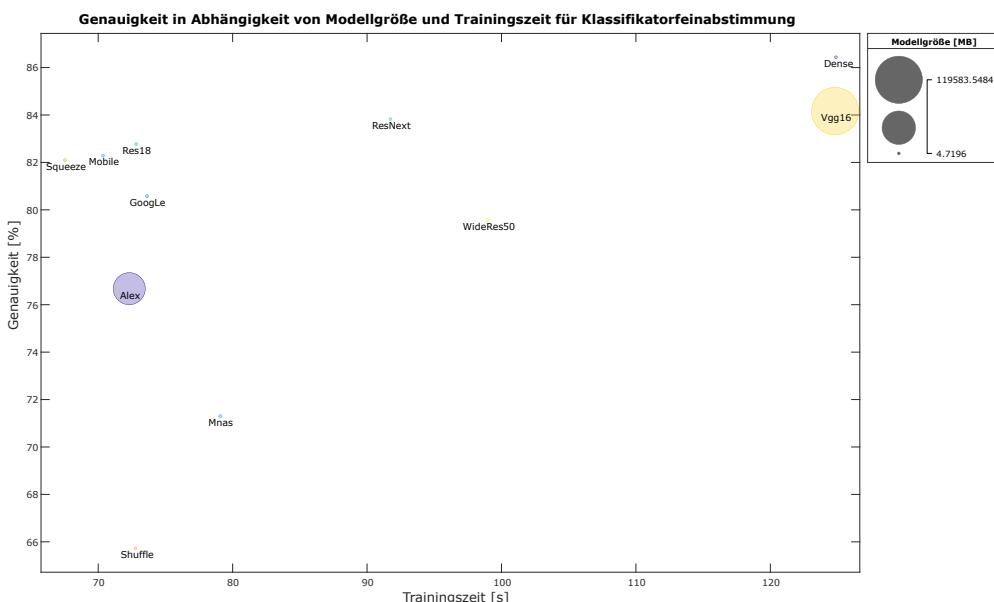


Abbildung 3.32: Modellgrößen und Genauigkeiten im Vergleich zur Trainingszeit für alle Aufgaben und Modelle nur nach Feinabstimmung der Klassifizierungsschichten, entnommen aus [6].

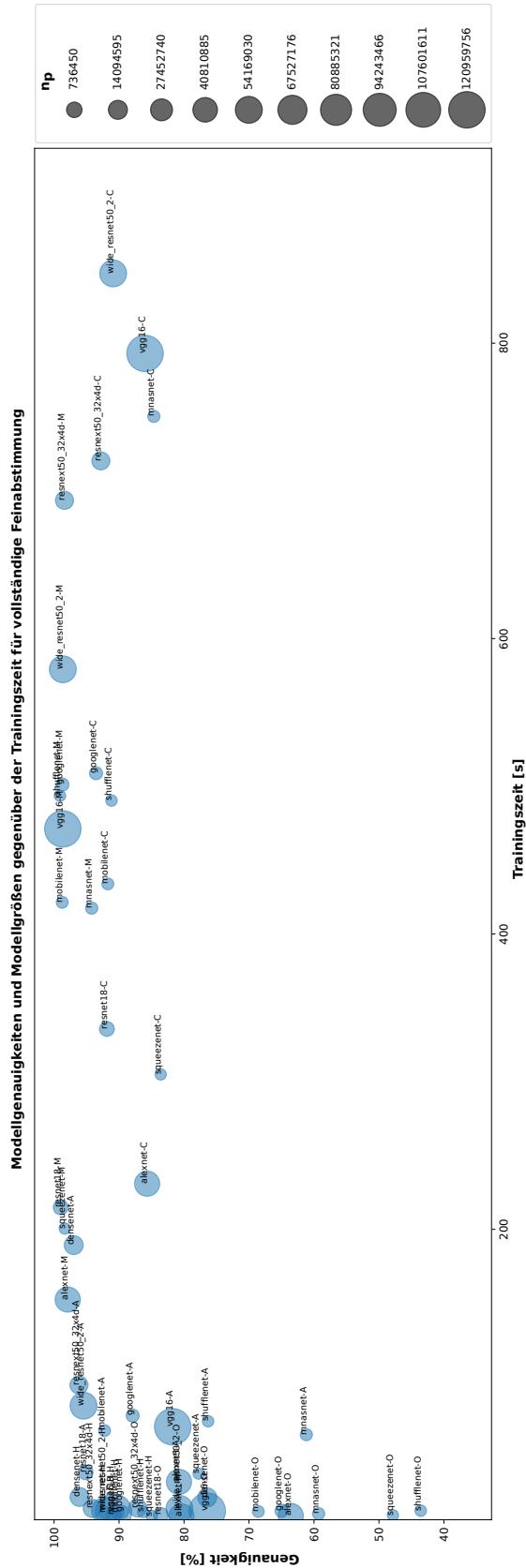


Abbildung 3.33: Detaillierte Metadaten für alle Modell- und Aufgabenkombinationen für die vollständige Feinabstimmung. Die Suffixe kürzen die verschiedenen Aufgaben ab: A: Smartphones (augmented), C: CIFAR10, H: Hymenoptera, M: MNIST, O: Smartphones (original).

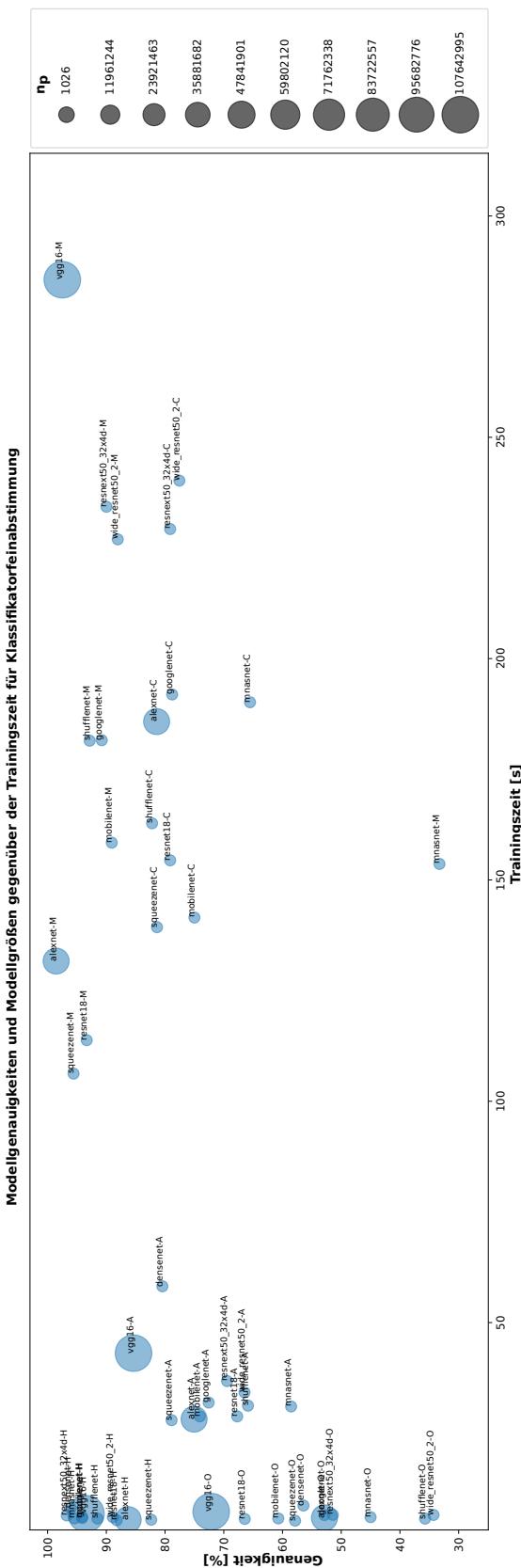


Abbildung 3.34: Detaillierte Metadaten für alle Modell- und Aufgabenkombinationen für die Klassifikatorfeinabstimmung. Die Suffixe kürzen die verschiedenen Aufgaben ab: A: Smartphones (augmented), C: CIFAR10, H: Hymenoptera, M: MNIST, O: Smartphones (original).

### 3.5.3 Diskussion

Die Leistung von tiefen künstlichen neuronalen Netzwerken zur Bildklassifikation hat sich in den letzten Jahren in vielerlei Hinsicht verbessert; nichtsdestotrotz gibt es kritische Parameter, die definieren, welches vortrainierte Modell zu konkreten Anforderungen einer Anwendung passt. Im Rahmen dieser Herausforderung wurde eine umfassende Bewertung der Transferlernenschaften von elf beliebten vortrainierten Modellen anhand von fünf Beispieldatensätzen vorgenommen, welche als Leitfaden für die Auswahl eines geeigneten Modells vor dem Einsatz in einer Anwendung dient. Es wurden zwei verschiedene Transferlernexperimente durchgeführt; das Transferlernen mit einer Episode und das Transferlernen mit zehn Episoden, wobei, je nach Modalität, eine Feinabstimmung nur der Klassifizierungsschichten und eine vollständige Feinabstimmung aller Schichten durchgeführt wurde. Es hat sich gezeigt, dass ein Transferlernprozess mit nur einer Episode genügt, um ein Bildklassifikationsproblem hinreichend zu lösen.

Die bisherigen Ergebnisse liefern einige Anhaltspunkte für die Auswahl des richtigen Modells für die Feinabstimmung der Klassifizierungsschichten; für Anwendungen, die eine hohe Genauigkeit erfordern, sind GoogLeNet, DenseNet, ShuffleNet-V2, ResNet-18, ResNext die besten Kandidaten. Bei Anwendungen, wo der Schwerpunkt auf der Genauigkeitsdichte liegt, eignet sich SqueezeNet. Für zeitkritische Anwendungen bietet sich AlexNet für die kürzeste Trainingszeit an. Wird eine kleine Modellgröße benötigt, beispielsweise für eingebettete Systeme, so sind SqueezeNet, ShuffleNet, MobileNet, MnasNet und GoogLeNet quasi gleichwertig geeignet. Andererseits können auch einige Richtlinien abgeleitet werden, wenn nur die Klassifizierungsschichten fein abgestimmt werden sollen. DenseNet erreicht die höchste Genauigkeit, ResNet18 die höchste Genauigkeitsdichte und SqueezeNet die kürzeste Trainingszeit. Obwohl die Richtlinien einige Hinweise geben, lässt sich kein endgültiges Urteil finden, sondern lediglich eine Entscheidungshilfe für die Auswahl des richtigen vortrainierten Modells auf der Grundlage der Aufgabenanforderungen. So kann die Auswahl des richtigen vortrainierten Modells für bestimmte Anwendungsbedingungen eine Herausforderung darstellen, da Kompromisse zwischen Trainingszeit, Modellkomplexität und Genauigkeit als Entscheidungsfaktoren für ein erfolgreiches Ergebnis erforderlich sind.

Bei dieser Forschungsarbeit zeigt sich, dass menschliches Expertenwissen notwendig ist, um einen idealen Transferlernprozess zu bestimmen. Besonders die detaillierten Ergebnisse in den Abbildungen 3.33 und 3.34 zeigen den Bedarf nach einer Systematisierung auf Grundlage der Metadaten, um in dieser unklaren Sachlage menschliches Expertenwissen durch maschinelle Prognosen zu unterstützen. Diese Anforderung bildet die Grundlage des, im Rahmen dieser Dissertation entwickelten, Konzeptes des *Transfer Meta Learning*. In weiterführenden Untersuchungen werden, auf Basis der Untersuchungen dieser Herausforderung, weitere Bewertungsmetriken mit nützlichen Parametern eingeführt, um die Entscheidungsfindung bei der Auswahl des optimalen Modells für die Feinabstimmung zu erleichtern. Auf diese Weise wird systematisch die Verwendbarkeit aller verfügbaren a priori und a posteriori Metadaten für die Schätzung nützlicher Hyperparameter des Transferlernprozesses untersucht, wie im nachfolgenden Kapitel zum Konzept des *Transfer Meta Learning* ausführlich beschrieben.

# 4 Transfer Meta Learning

Das Ziel des Transferlernens ist es, zuvor erworbenes Wissen über eine Quellaufgabe wiederzuverwenden, um das Lernen einer Zielaufgabe zu erleichtern. Die Methode des *Transfer Meta Learning* lernt Metadaten von Transferlernprozessen, um die wahrscheinlichsten Einstellungen für einen erfolgreichen maschinellen Wissenstransfer zu ermitteln [5]. Dies geschieht auf Grundlage vorher bekannter Modell- und Aufgabendaten, wie beispielsweise der Parameter- und Datenpunktzahlen, jedoch ohne Kenntnis konkreter Daten aus der Zielaufgabe. Hierfür werden zuvor bekannte Metadaten über die Quelldomäne, die Zieldomäne und das voreingestellte Modell genutzt, um eine Abbildung der Transferlernparameter zur Genauigkeit des resultierenden transferbelehrten Modells abzuschätzen.

Zur Demonstration des Ansatzes wurden über 15.000 Metadaten und Modellparameter von Transferlernverfahren zu einem Datensatz zusammengefasst, welcher verwendet wird, um Metamodelle zu lernen. Diese Metamodelle schätzen die *a posteriori* Information, vor allem die letztendliche Genauigkeit des jeweiligen Transferprozesses, auf Grundlage von *a priori* Information, wie etwa der Anzahl der Epochen, der Lernrate und des Optimierungskriteriums. Als Metamodellarchitektur wird ein mehrschichtiges Perzepron verwendet und mit einem linearen Modell und dem Einsatz der besten bekannten Konfiguration verglichen, um die Frage nach der Güte der Abschätzung des Transferlernprozessergebnisses entscheiden zu können. Mit dem Einsatz des mehrschichtigen Perzepron zeigt sich, dass die Methode des *Transfer Meta Learning* effektive Hyperparameter für das Transferlernen zur Bildklassifikation auf Grundlage von Metadaten finden kann. Da dies bedeutet, dass weniger Rechenressourcen benötigt werden, um mittels eines systematisch gewählten Transferlernprozess gute Ergebnisse zu erzielen, bedeutet dies auch, dass weniger Energie für das Training tiefer künstlicher neuronaler Netzwerke verbraucht werden muss. Der gewinnbringende Gedanke hierbei besteht in einer Verringerung der verursachten Kohlendioxidemissionen; doch es steht zur Befürchtung, dass mit effizienteren Methoden nicht weniger Energie verbraucht, sondern lediglich mehr Lösungen in der selben Zeit berechnet werden.

Die in Abbildung 4.1 zusammengefasste Methodik wird eingesetzt, um die Forschungsfrage zu beantworten, ob mit einem systematischen Metalearnverfahren Vorschläge für Transferlernprozesse generiert werden können, welche zu besseren Ergebnissen führen als die einfache Verwendung der bisher besten Konfiguration oder eine Vorhersage durch ein lineares Regressionsmodell. Zur Beantwortung der Forschungsfrage wird zunächst ein Metadatensatz aus 15.972 Transferlernprozessergebnissen erstellt und anschließend die Lernbarkeit der erfassten Zusammenhänge durch insgesamt 10.402 Metalearkonfigurationen untersucht, welche ausgewertet und mit den beiden einfachen Ansätzen verglichen werden.

Es zeigt sich, dass das Konzept des *Transfer Meta Learning* einen vielversprechenden Beitrag darstellt. Nach aktuellem Kenntnisstand handelt es sich hierbei um einen neuartigen Ansatz, welcher auf der grundlegenden Idee beruht, zunächst nützliche Metadaten aus Transferlernprozessen zu sammeln und diese dann zu nutzen, um bessere Hyperparameter zu schätzen.

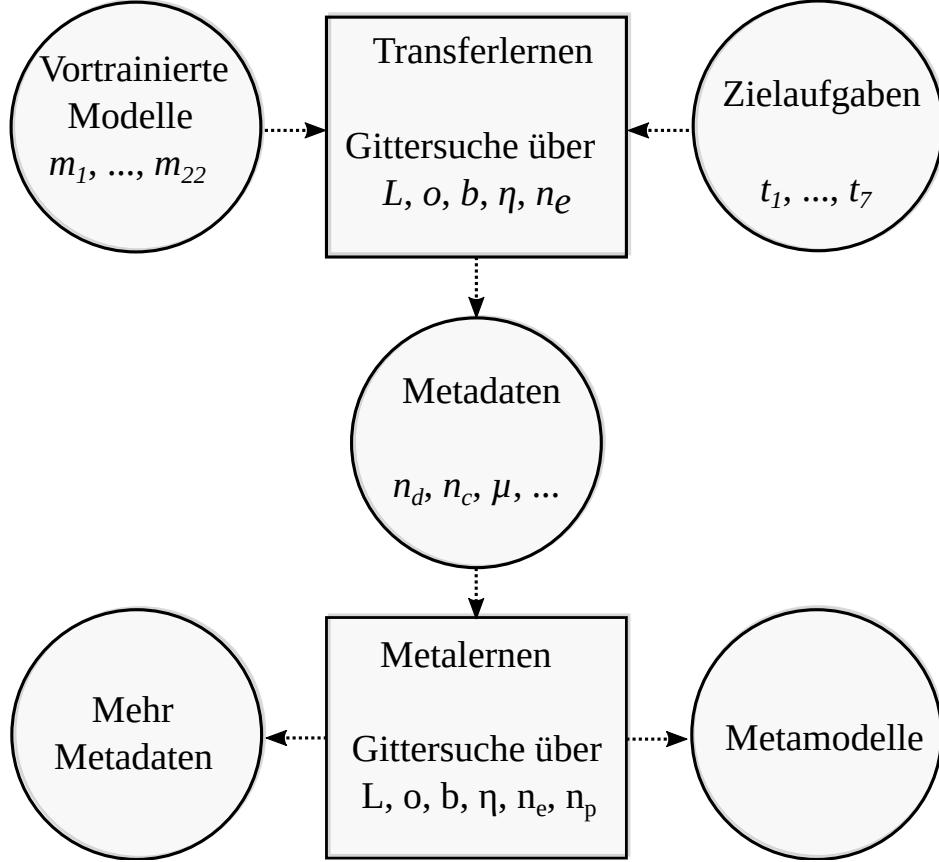


Abbildung 4.1: Der methodische Ablauf des *Transfer Meta Learning* als Flussdiagramm. Mit ausgewählten Aufgaben  $t$  und vortrainierten Modellen  $m$  werden in einer Gittersuche über die Verlustfunktionen  $L$ , die Optimierer  $o$ , die Stapelgrößen  $b$ , die Lernraten  $\eta$  Metadaten erzeugt. Mit Eigenschaften der Aufgaben, etwa der Anzahl der Klassen  $n_c$  und Datenpunkte  $n_d$  und den stochastischen Momenten des Labelraumes  $\mu$ , werden Metalerner trainiert. Um einen optimalen MLP Metalerner zu finden, wird erneut eine Gittersuche durchgeführt, diese zusätzlich mit verschiedenen Epochenzahlen  $n_e$  und Neuronenzahlen  $n_p$ . Veröffentlicht in [5].

## 4.1 Motivation

Motiviert durch die offenen Fragen aus der Herausforderung des Bildklassifikationstransfers wird ein systematischer Ansatz zur Optimierung von Transferlernrozessen gesucht [6]. Gesucht wird ein Programm, welches die Fragen der Modellauswahl und zugehörigen Transferlernparametern mit hoher Zuverlässigkeit beantworten kann. Da nicht in jedem Fall konkrete Daten aus der Aufgabe zum Zeitpunkt der Modellsuche bekannt sind, kann es von Vorteil sein, diese Fragen allein mit der Kenntnis von Metadaten zu beantworten, also ohne Kenntnis von tatsächlichen Proben aus dem Datensatz der Zielaufgabe. Dieses Konzept reduziert den Bedarf an Expertenwissen für die Anwendung von Transferlernen und kann automatisiert Vorschläge beitragen, mit deren Hilfe bessere Lernprozesse zu finden sind. Für eine Darlegung der Machbarkeit beschränkt sich die Problemklasse zunächst auf die Bildklassifikation und die Transferlernmethode der Feinabstimmung; jedoch sind Erweiterungen des zugrunde liegenden Ansatzes auf andere Herausforderungen des maschinellen Lernens denkbar.

Die verwandte Forschung kennt zur Zeit noch keine Methode, um mittels eines Metalern-ansatzes optimale Transferlernprozesse abzuschätzen. Im Sinne des maschinellen Lernen beschäftigen sich die Konzepte des Metalernens damit, wie eine Maschine maschinelles Lernen lernt, also lernt, welche Methoden mit welchen Hyperparametern zu guten Ergebnissen führen [134, 317]. Die allgemeinen Konzepte der Metadatenanalyse finden auch in vielen anderen Bereichen Anwendung; in den Naturwissenschaften beispielsweise sparen Anwendungen, die lernen, Simulationseinstellungen aus Metadaten zu übertragen, Rechenzeit und Energie [41, 43, 318]. Auch der aktuelle Stand der Technik im maschinellen Lernen kennt bereits mehrere Ansätze, welche die Konzepte von Transferlernen und Metalernen auf verschiedene Weisen zusammenführen. Im Kontext des Trainings tiefer künstlicher neuronaler Faltungsnetzwerke kann das Lernen verallgemeinerter Metarepräsentationen helfen, hochgradig übertragbare Parametervektoren für Zero-Shot- und Few-Shot-Ansätze zu finden [39, 40, 42, 319], so dass das Lernen von Metarepräsentationen selbst auch als eine Methode des Transferlernens betrachtet werden kann. Die Suche nach einem guten Repräsentationslernalgorithmus hat zu einem Benchmark für Transferlernen geführt, der Visual Task Adaptation Benchmark (VTAB) [320]. In einer Anwendung zur adaptiven Strahlformungsoptimierung für ein Signalverarbeitungsproblem hilft die Kombination von Transfer- und Metalernen, das Problem der Leistungsverschlechterung zu lösen, wenn sich die Testumgebung ändert [34]. Ein anderer Ansatz nutzt die Idee von Metalernker-nen zur Verkettung von Transformationen, um einem Transferlerner zu helfen, schnell neue Zielaufgaben zu lernen [29]. Eine Demonstration eines Ensembles von Transfer- und Metalernern mit schnellen sigmoidalen Regressionsmodellen, welche die modernsten Ansätze auf einem bestimmten Datensatz übertreffen, verwendet hierarchisch entwickelte Ensembles als Bausteine [30]. Informationstheoretische Überlegungen untersuchen die Grenzen des *Transfer Meta Learning* und kommen zu dem Schluss, dass obere Grenzen für die empirische Risikominimierung in der durchschnittlichen Generalisierungslücke und den hochwahrscheinlichen bayes'schen Grenzen liegen [35].

## 4.2 Konzept

In diesem Abschnitt wird das grundlegende Konzept anhand klassischer Paradigmen der Informatik ausgeführt, was für das Verständnis der Methodik des *Transfer Meta Learning* hilfreich ist. Die Umsetzung der Methodik des *Transfer Meta Learning* in eine objektorientierte Software vereinfacht die Verwaltung der Datenmengen, die Anwendbarkeit für den Nutzer sowie die Wartbarkeit der Software und erlaubt eine einfachere Weiterentwicklung für weitere Konzepte; also andere Arten von Aufgaben, wie etwa bestärkendes Lernen, und andere Arten von Modellen, wie beispielsweise SVM oder Entscheidungsbäume. Eine Aufgabe ist hierbei zunächst von allgemeiner Natur und definiert sich im Speziellen über den konkreten Aufgabentypen; beispielsweise kann sich eine Aufgabe als Instanz einer überwachten, unüberwachten oder einer Verstärkungslernaufgabe darstellen. Im Sinne des maschinellen Lernens verfügt jede Aufgabe über eine Datenlage, auf deren Grundlage ein Optimierungsverfahren eine Zielfunktion zu erfüllen versucht. So entstammen beispielhafte Eingabevektoren  $(x, y)$  der Datenlagen einer Aufgabe  $A$ ; diese können im Prinzip beliebige Daten enthalten. Für eine Aufgabe des überwachten maschinellen Lernens kann  $x$  beispielsweise eine Menge von Bildern und  $y$  die Menge zugehöriger Label enthalten, bei einer unüberwachten Lernaufgabe existieren keine Label, hier also entspricht  $x = y$ . Hier gilt das Ziel, beispielsweise mit einem Autoencoder oder einer Hauptkomponentenanalyse, wesentliche Strukturen in den Daten zu erkennen, beispielsweise in Nachbarschaften oder Clustern. Im Rahmen eines Verstärkungslehrproblems entstammen die Daten einem markov'schen Entscheidungsprozess unterschiedlicher Beobachtbarkeit, hier würde  $x$  aus den Zuständen der Umgebung und den zugehörigen ausgeführten Aktionen bestehen,  $y$  aus den darauf folgenden Belohnung.

Die Formulierung der Konzeption des *Transfer Meta Learning* bedient sich des EVA-Prinzips und nutzt dynamische Bindung für polymorphe Datenstrukturen zur Gestaltung mannigfaltiger Aufgaben- und Optimierungsfunktionsklassen. Das EVA-Prinzip, also die Gliederung des Programms in einen Eingabeteil, einen Verarbeitungsteil und einen Ausgabeteil stellt ein ausführlich untersuchtes Grundprinzip der Datenverarbeitung dar. Die klare Einteilung nach einem bewährten Standard gestaltet das Zusammenspiel der Module übersichtlicher und vereinfacht die Handhabung.

Die objektorientierten polymorphen Datenstrukturen erlauben die Formulierung verschiedener Datenverarbeitungs- und Optimierungsverfahren als Erben einer Basisklasse. Dies können nicht nur künstliche neuronale Netzwerke oder evolutionäre Algorithmen sein, sondern beispielsweise auch beliebige Merkmalsextraktoren, latente Repräsentationen, Dimensionierungsänderungen, Clusteringverfahren oder Signalfilter. Die softwaretechnische Umsetzung kann beständig erweitert werden, sodass zukünftige Entwickler auf dem bereits vorliegenden Quelltext aufbauen können, um die Software für weitere konkrete Anwendungen zu optimieren, indem weitere Optimierungs- und Transferlernverfahren oder neue Aufgabenklassen hinzugefügt werden. An dieser Stelle können die in den Grundlagen erläuterten weiterführenden Methoden des Transferlernens eingebaut werden. Zur programmiertechnischen Verwirklichung wird die Programmiersprache PyTorch genutzt, da eine Vorstudie ergeben hat, dass diese sich aufgrund der konsequenten Objektorientierung und lebendigen Entwicklungsgemeinschaft am Besten eignet [13].

#### 4.2.1 Modellierung nach EVA-Prinzip

Das EVA-Prinzip modelliert ein Programm mit Abläufen für Eingabe, Verarbeitung und Ausgabe. In Sinne dieses Modellierungsprinzips besteht die Eingabekomponente aus den Methoden zum Einlesen und Aufbereiten der Metadaten der bereits bekannten und der unbekannten Aufgabenklassen, sowie der Aufbereitung der Metadaten der vortrainierten Modelle. Der Verarbeitungsteil im Konzept des *Transfer Meta Learning* besteht im Lernprozess, welcher die Zusammenhänge der Metadaten der Transferlernprozesse abschätzen soll. Diese Methode setzt sich aus der Gesamtheit der Methoden des Metamodells zusammen, also dessen Lern- und Vorhersagefunktionalitäten. Die Verarbeitung nach dem EVA-Prinzip besteht also aus der Findung geeigneter Metadatenzusammenhänge zwischen Aufgaben, Modellen und Transferlernhyperparametern zur Abschätzung eines Optimierungsprozesses für die ungesehene, neue Aufgabe. Dies stellt die Grundlage für einen Algorithmus dar, welcher die neue Aufgabe mit einem möglichst geringen Rechenaufwand unter Verwendung der bereits bekannten Lösungen anderer Aufgaben zu lösen imstande ist. Die Ausgabe im Sinne des EVA-Prinzips besteht aus der Inferenz des Metamodells, also der Abfrage zur Findung optimal transferbelehrbarer Modelle und deren Hyperparameterkonfigurationen für die neue Eingabeaufgabe. Die Metadaten der nunmehr gelösten Aufgabe können, mitsamt ihrer zugehörigen Lösung, daraufhin als weitere Eingabe, beziehungsweise als ein weiteres Ausgangsmodell, für ein fortwährendes *Transfer Meta Learning* genutzt werden. Abbildung 4.2 illustriert die Anwendung des EVA-Prinzips zur Umsetzung der Methodik des Transferlernens mit Hilfe eines Metamodells. Das Metamodell lernt, die Metadaten abzubilden, indem es die Zusammenhänge zwischen a priori und a posteriori Informationen modelliert. Als Ausgabe für eine neue Aufgabe  $A_n$  kann nun eine Inferenz über den Eingabedatenraum derart durchgeführt werden, dass diejenigen Modelle und Hyperparameter gefunden werden, welche beispielsweise eine maximale Genauigkeit versprechen.

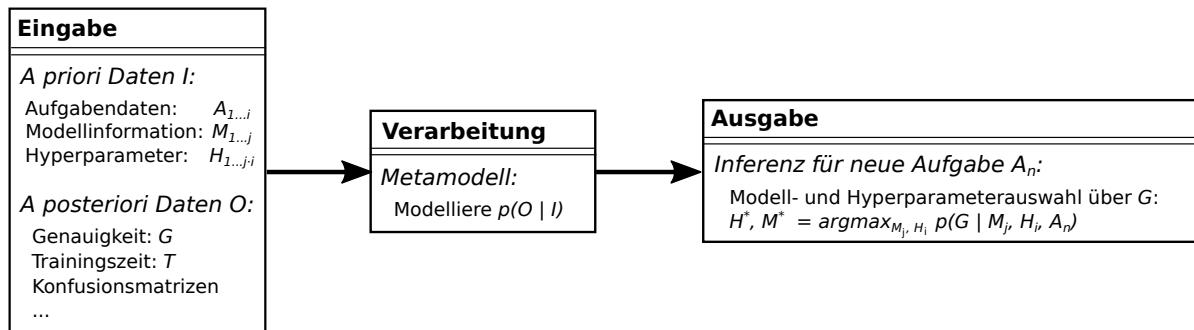


Abbildung 4.2: Darstellung der Methodik zur Anwendung von Metamodellwissen für Transferlernaufgaben nach dem EVA-Prinzip. Dem Metamodell werden verschiedene Aufgabendaten  $A$ , Modelldaten  $M$ , Hyperparameter  $H$  mitsamt der zugehörigen Transferlernresultate, beispielsweise die Genauigkeit  $G$ , die Trainingszeit  $T$  oder weitere a posteriori Information, als Eingabe präsentiert.

Der Anwender kann Aufgaben- und Modellinstanzen zum Metamodell hinzufügen und auch anzuwendende Transferlernmethoden wählen. Aufgrund verschiedener Randbedingungen der einzelnen Optimierungsverfahren und Datenlagen gestaltet sich ein generischer Ansatz als Herausforderung in abstrakter Formulierung von Ein- und Ausgabespezifikationen; so müssen beim Hinzufügen einer neuen Aufgabe die vom Anwender vorgegebenen Transferlernmethoden und Ausgangsmodelle verwendet und auf einer Metalebene in Zusammenhang gebracht werden. Für eine neue Transferlernaufgabe müsste geprüft werden, inwiefern die Randbedingungen der einzelnen Aufgaben mit denen der Modelle übereinstimmen. Stimmen Randbedingungen überein, so kann das Metamodell genutzt werden, um ein optimales Modell  $M^*$  mit optimalen Transferlernhyperparametern  $H^*$  aus den bekannten Modellen so zu wählen, dass die Vorhersage auf den Daten der neuen Aufgabe  $A_n$  maximal genau ist; also das bestmögliche Modell als Grundlage für die zu erbringende Transferleistung ausgewählt wird. Dies kann über eine Betrachtung der Vorhersagegenauigkeit auf den Daten der neuen Aufgabe geschehen, wenn die Aufgabeninstanzen vom selben Typ sind und sich die Dimensionalitäten der Ein- und Ausgabedaten ineinander überführen lassen. Ausgehend von  $M^*$  kann ein passendes Transferlernverfahren  $H^*$  angewandt und eine Lösung der neuen Aufgabe  $A_n$  gefunden werden. Das auf diese Weise gefundene Modell  $M^*$  und dessen tatsächliche Leistung auf der neuen Aufgabe  $A_n$  unter Anwendung eines Transferlernverfahrens mit den Hyperparametern  $H^*$  kann nun als weitere Eingabe dem Metamodell zugeführt werden, was Anwendungen in einem fortwährenden Lernprozess ermöglicht.

### 4.2.2 Objektorientierte Umsetzung

Die Umsetzung des Konzeptes des *Transfer Meta Learning* orientiert sich an den Paradigmen der Objektorientierung, da dieser moderne Programmierungsansatz die Wieder verwendbarkeit und Wartbarkeit einzelner Komponenten erleichtert und das Erfassen der Zusammenhänge vereinfacht [198]. Polymorphie ist ein objektorientiertes Konzept, welches einem Objekt erlaubt, zur Laufzeit eine von mehreren möglichen Gestalten anzunehmen. Ein solcher Polymorphismus erlaubt verschiedenen erbenden Unterklassen Methoden und Attribute gleichen Namens zu besitzen, wobei sich die konkrete Implementierung unterscheiden kann. Dies kann realisiert werden, indem eine erbende Klasse eine Methode ihrer Basisklasse abändert, um beispielsweise ein anderes Optimierungsverfahren zu realisieren. Die überladene Optimierungsmethode verfügt über die selbe Signatur wie die Optimierungsmethoden anderer Modelle und kann aus Sicht eines Metamodells somit einfach ausgetauscht werden. So kann beispielsweise durch den Aufruf einer *fit()*-Methode eines neuronalen Netzwerkmodells ein anderes Optimierungsverfahren angewandt werden als durch den Aufruf der *fit()*-Methode einer SVM. Ebenso können *get()* und *set()*-Methoden für das Setzen und Auslesen von Parametervektoren überladen werden, welche dann in den *save()* und *load()*-Methoden der Basisklasse für das Laden und Speichern von persistenten Daten aufgerufen werden können.

Wenn eine überladene Methode eines polymorphen Objektes aufgerufen wird, dann gibt es also mehrere mögliche Kandidaten; die Methode der Basisklasse sowie die überladenen Methoden der, entsprechend der Hierarchie, erbenden Klassen. Dies erlaubt eine Formu-

lierung abstrakter Modell- und Aufgabenklassen. Dynamische Bindung zu nutzen heißt hier, mit Referenzen auf den konkreten Unterklassen zu arbeiten, um die entsprechend gewünschten Methoden auszuführen. Die dynamische Bindung erlaubt, zur Laufzeit den tatsächlichen Datentyp eines Objektes aufzulösen, um die jeweilige überladene Methode aufzurufen. Abbildung 4.3 skizziert die Grundidee der Polymorphie und der dynamischen Bindung der Klassenhierarchien, welche in dieser Methodik Anwendung finden, und wie diese in einem Metamodell zusammengeführt werden.

## Aufgabenklassen

Eine Aufgabe im Sinne des maschinellen Transferlernens besteht aus einer Datengrundlage und einem zugehörigen Optimierungsproblem. Die Natur der Datengrundlagen kann sich verschieden ausgestalten; für eine Aufgabe des überwachten maschinellen Lernens besteht diese beispielsweise aus Paaren von Ein- und Ausgabedaten, für eine Aufgabe des unüberwachten maschinellen Lernens lediglich aus einem zu ergründendem Datensatz und für eine Aufgabe des Verstärkungslernens aus Daten eines markov'schen Entscheidungsprozesses. Allen diesen verschiedenen Ausgestaltungen ist gemein, dass die Daten innerhalb der Aufgabeninstanz vollständig bekannt sind. Neue Daten werden, im Sinne des *Transfer Meta Learning*, über neue Aufgabeninstanzen bereitgestellt.

Abbildung 4.4 stellt die erste Ebene der Vererbungshierarchie der Aufgabenklassen dar. Die abstrakten Basisklasse *Task*, von welcher selbst keine Instanzen erstellt werden können, gibt das Vorhandensein einer Initialisierungsmethode *init()* und einer Methode zur Abfrage der Metadaten *getMetadata()* vor. Die Methode *init()* soll vor der erstmaligen Verwendung jeder Aufgabeninstanz genutzt werden, um beispielsweise einen Datensatz herunterzuladen, Labelinformationen einzulesen und andere notwendige Vorbereitungen zu treffen, die Erben der Basisklasse definieren die konkreten Initiierungen; die Methode *getMetadata()* soll die relevanten Metadaten der jeweiligen Aufgabe für das Metalernen zur Verfügung stellen. Die Art des maschinellen Lernverfahrens bestimmt die Aufteilung der Datengrundlage; bei überwachten und unüberwachten Lernverfahren liegen die Datensätze als Tensoren vor, bei interaktiven Aufgaben, wie beispielsweise einem markov'schen Entscheidungsprozess, wird die zu optimierende Funktion als *step()*-Methode angenommen, welche aufgrund einer Aktion und des internen Zustandes eine zu maximierende Vergütung als Fließkommazahl zurückgibt. Von diesen Klassen können nun weitere Klassen erben, wie in Abbildung 4.3 skizziert. So erbt etwa eine Klasse *Classification* von *Supervised* und von dieser wiederum konkrete Bildklassifikationsaufgaben, wie etwa *MNIST*, *CIFAR10* oder *Places365*.

## Modellklassen

Eine Modellklasse im Sinne des *Transfer Meta Learning* beschreibt zunächst ein Optimierungsverfahren im Allgemeinen. Ein solches Optimierungsverfahren besteht sowohl aus dem zugrundeliegenden Optimierungsalgorithmus, wie etwa dem stochastischen Gradientenabstieg, als auch dem zu optimierenden Parametertensor, wie beispielsweise den synaptischen Gewichten eines tiefen künstlichen neuronalen Netzwerks. Das maschinelle

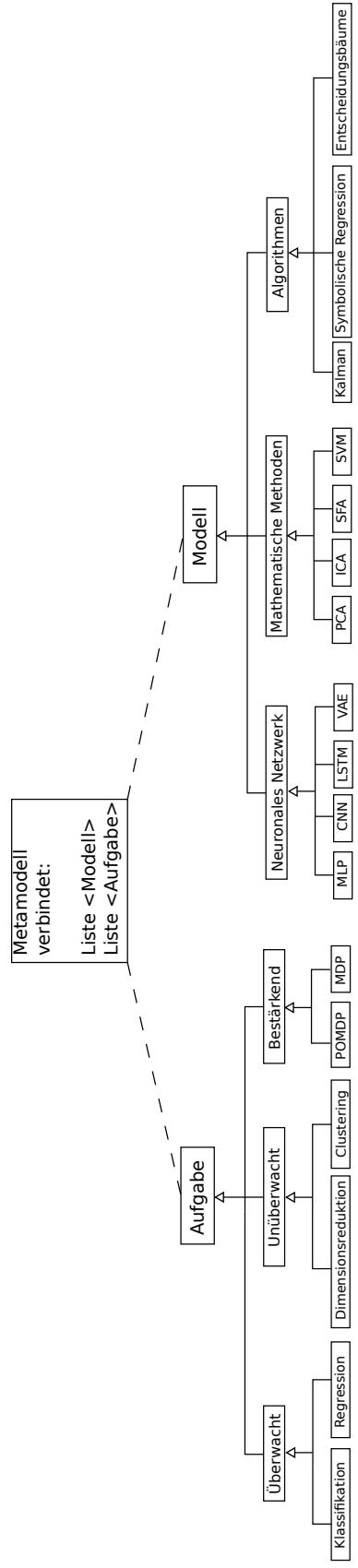


Abbildung 4.3: Eine Skizze zur Illustration der Polymorphie und dynamischen Bindung. Von den Basisklassen zur Modellierung beispielhafter konkreter Aufgaben oder Optimierungsverfahren erben verschiedene Unterklassen, welche das jeweilige Optimierungsmodell beziehungsweise die jeweilige Datenlage konkretisieren. In einer Metamodellklasse werden mit dynamischer Bindung die Referenzen auf die jeweiligen Unterklassen in einer Indexstruktur, beispielsweise einer Liste, zusammengeführt.

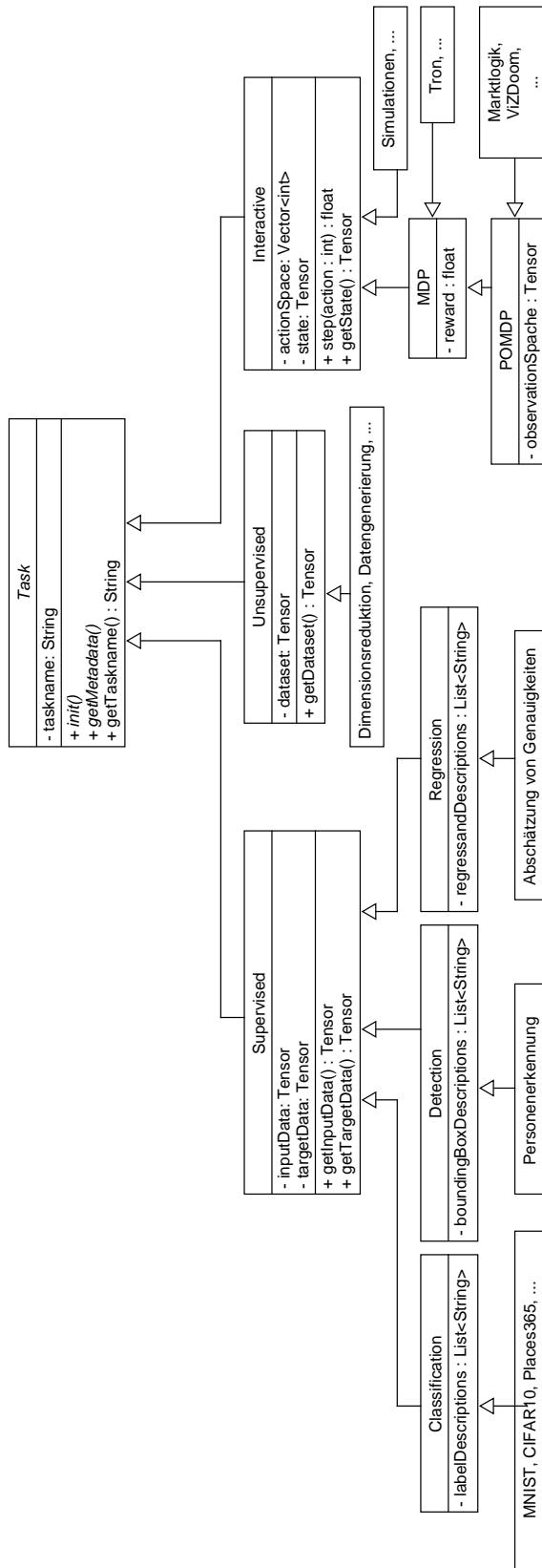


Abbildung 4.4: Ein UML-Klassendiagramm zur Veranschaulichung des Aufgabenkonzeptes. Eine abstrakte Klasse *Task* dient als Basisklasse für verschiedene maschineller Lernaufgaben (überwachtes Lernen, unüberwachtes Lernen, Verstärkungslernen). Die Blätter des dargestellten Graphen deuten verschiedene Formen von maschinellen Lernaufgaben an, welche im Rahmen dieser Arbeit untersucht wurden oder Erwähnung fanden.

Lernen kennt eine Vielzahl verschiedener Optimierungsverfahren, die aufgrund ihrer unterschiedlichen Randbedingungen nicht beliebig ausgetauscht werden können; beispielsweise können evolutionäre Algorithmen eingesetzt werden, wenn kein Gradient genutzt werden kann. Allen Optimierungsverfahren ist aber gemein, dass sie aufgrund von Daten irgendeine Form von Mustererkennung, Vorhersage, Rekonstruktion, Klassifikation oder Regression vornehmen können.

Abbildung 4.5 stellt die Vererbungshierarchie im Modellkonzept dar. Eine abstrakte Basisklasse *Model* verlangt die Implementierung einer *fit()*-Funktion zum Lernen und einer *infer()*-Funktion zur Modellinferenz in den erbenden Klassen. Weiterhin wird die Existenz eines Parametertensors angenommen, welcher jedoch ebenfalls in den erbenden Klassen genau ausgestaltet werden muss. Die konkrete Ausgestaltung der Parameter-tensoren in den erbenden Klassen wird dann von den *save()*- und *load()*-Methoden der Basisklasse zur Sicherung des Modells auf Persistenzebene genutzt. Die verschiedenen Optimierungsverfahren implementieren dann ihre Eigenheiten in den Methoden und Attributen der jeweiligen erbenden Klasse. Die im Deep Learning verwendeten tiefen künstlichen neuronalen Netzwerke stellen sich in der Klasse *NeuralNetwork* dar und bieten dem Anwender die Möglichkeit, Hyperparameter wie etwa Stapelgröße und Lernraten zu modifizieren. Zur Verfeinerung der Randbedingungen können weitere Vererbungen hinzugefügt werden, beispielsweise Erben der Klasse *NeuralNetwork* für Variational Autoencoder (VAE), Long Short-Term Memory (LSTM) oder Generative Adversarial Networks (GAN). Weitere denkbare Modellklassen würden die Modellierung einer Wahrscheinlichkeitsverteilung mit Gauß'schen Mixturen oder naiven Bayesansätzen, den Einsatz evolutionärer Algorithmen, besondere Formen des stochastischen Gradientenabstiegs oder Algorithmen der Bild- und Signalverarbeitung umfassen.

## Metamodellklasse

Die Metamodellklasse implementiert die Verwaltung der Aufgaben und Modelle, das Einlesen von sowie das Lernen aus Metadaten und die Modellauswahl. Die Aufgabe dieser Klasse besteht in der Zuordnung von Modellklassen, Aufgabenklassen und zugehörigen Metadaten über eine dynamisch gebundene Indexstruktur. Ein Anwender kann die Inferenzmethode des Metamodells nutzen um, unter Wahrung der Randbedingungen der bekannten Aufgaben- und Modellinstanzen, einen Vorschlag für Parameter eines prinzipiell beliebigen Wissenstransfers zu erhalten. Abbildung 4.6 stellt die Metamodellklasse in Relation mit den Aufgaben- und Modellklassen dar. Indexstrukturen, wie beispielsweise verkettete Listen, erlauben eine Verwaltung der polymorphen Aufgaben- und Modellinstanzen über dynamische Bindung.

Das Metamodell setzt bekannte Optimierungsverfahren und Aufgaben in Verbindung und bietet Lern- und Inferenzmethoden, um auf Grundlage bekannter Modelle ein neues Modell für eine neue Aufgabe auswählen und optimale Hyperparameter für den Transferlernvorgang ableiten zu können. Dies adressiert die Herausforderung der Modellauswahl, also die Auswahl desjenigen Modells samt Hyperparametern, welches die zugrundeliegenden Daten einer gegebenen Aufgabe am Besten erklärt, beziehungsweise die genauesten Vorhersagen trifft.

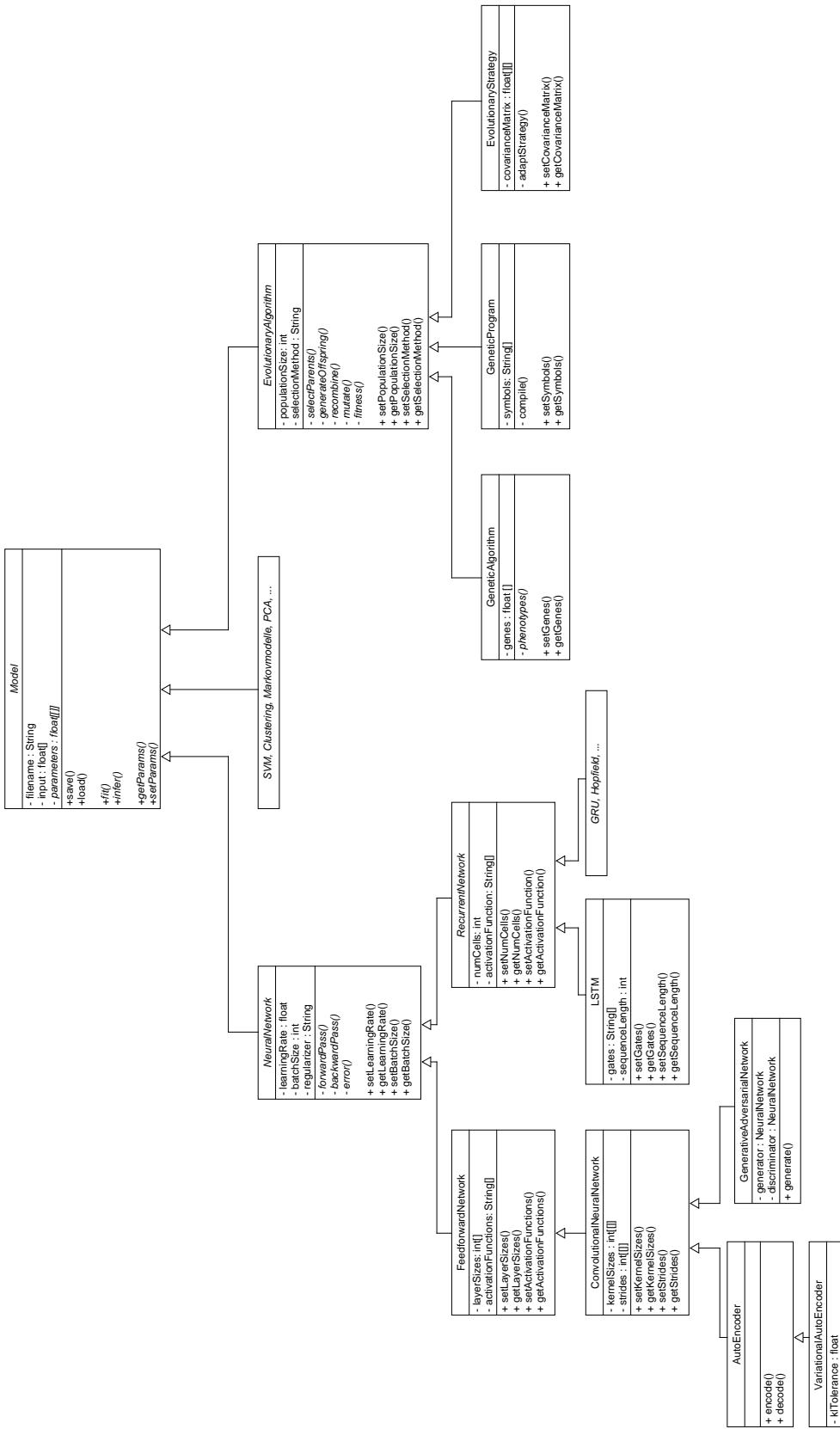


Abbildung 4.5: Ein UML-Klassendiagramm zur Vererbungshierarchie der einzelnen Modellklassen. Alle Optimierungsverfahren erben von der Basisklasse `Model`. Zur besseren Übersichtlichkeit wurden in dieser Darstellung sind nicht alle denkbaren Optimierungsverfahren ausformuliert, werden aber angedeutet.



Abbildung 4.6: Ein UML-Klassendiagramm zur Beschreibung des Metamodells. Diese Klasse verfügt über Listen bekannter Modelle und Aufgaben und bietet eine Methode zum Einlesen zugehöriger Metadaten, zum Lernen aus diesen und zur Inferenz für eine Eingabeaufgabe.

Hierzu kann das Metamodell unterschiedliche Ein- und Ausgabedaten aus vorliegenden Aufgabendatensätzen nutzen, deren Lösungsqualität mit bekannten Modellen abbilden und somit eine Wissensbasis für Transferlernmethoden zur Verfügung stellen.

## 4.3 Methode

Die Methode des *Transfer Meta Learning* beginnt mit einer Reihe von vortrainierten Modellen, welche mittels Transferlernen auf eine Reihe von Aufgaben abgestimmt werden und endet mit Metamodellen, welche verwendet werden können, um die gesammelten *a priori* und *a posteriori* Metadaten abzubilden. Die gelernten Metamodelle können systematisch Vorschläge für die Einstellungen von Transferlernvorgängen beitragen, welche zu optimalen Wissensübertragungsprozessen führen. An dieser Stelle würde sonst menschliches Expertenwissen über die bisher als beste bekannte Konfiguration Einsatz finden. Als Alternative zur Abschätzung durch Expertenwissen, oder durch ein mehrschichtiges Perzepron, wird auch die Abschätzung durch eine lineare Regression verglichen, wobei sich zeigt, dass das tiefe künstliche neuronale Netzwerk diese Zusammenhänge am zuverlässigsten abbilden kann.

Zur Auswertung der Methodik wird eine Rastersuche von Transferlernprozessen durchgeführt, wobei die einzelnen Ergebnisse und relevanten Metadaten erfasst werden. Abbildung 4.1 gibt einen kurzen Überblick über den Gesamtprozess und die Zusammenhänge der Modelle, Aufgabendaten und Lernprozesse. Auf der Grundlage der gesammelten Metadaten wird mittels Metalernen ein Metamodell berechnet, welches genutzt werden kann, um die *a posteriori* Information über die Transferlernprozesse, wie beispielsweise die Genauigkeiten auf den jeweiligen Trainings-, Validierungs- und Testdatensätzen der Zielaufgabe, auf Grundlage von gegebenem *a priori* Wissen, wie zum Beispiel die Identität des Modells, die Anzahl der Parameter, statistische Eigenschaften der Aufgabe und Hyperparametereinstellungen, abzuschätzen. Die allgemeine Methode kann auf die Verwendung weiterer *a posteriori* Variablen ausgedehnt werden, denkbar wären unter anderem die Trainings- und Inferenzzeiten oder Konfusionsmatrizen.

### 4.3.1 Aufgaben und Modelle

Als Quellaufgabe für alle vortrainierten Modelle dient ImageNet [304]. In den Transferlernprozessen werden die folgenden Zielaufgaben  $\{t_1 \dots t_7\}$  verwendet, um einen Wissenstransfer von der Quellaufgabe zur Zielaufgabe zu berechnen:

- $t_1$ : CIFAR10 [301]
- $t_2$ : MNIST [302]
- $t_3$ : FashionMNSIT [321]
- $t_4$ : Places365 [322]
- $t_{5,6}$ : Smartphones (original und erweitert) [26]
- $t_7$ : Hymenoptera (aus einem PyTorch Tutorial [298, 303])

Für die Transferlernprozesse werden die folgenden vortrainierten Modelle  $\{m_1 \dots m_{22}\}$  verwendet, wie sie in PyTorch [298] implementiert und mit der Ausgangsaufgabe vortrainiert wurden:

- $m_1$ : AlexNet [305, 323]
- $m_2$ : VGG-16 [306]
- $m_3$ : GoogLeNet [324]
- $m_4$ : ResNet18 [309]
- $m_5$ : SqueezeNet [310]
- $m_6$ : DenseNet [311]
- $m_7$ : ResNext [312]
- $m_8$ : MobileNetV2 [313]
- $m_9$ : Wide ResNet [314]
- $m_{10}$ : ShuffleNet [315]
- $m_{11}$ : MnasNet [325]

Für jedes dieser Modelle wird die Feinabstimmung des gesamten Parametervektors und nur des Klassifikatorteils untersucht, so dass die Modelle  $m_{12:22}$  die Klassifikatorteile der ursprünglichen Modelle enthalten. Wenn einem Modell kein separater Klassifikatorteil zugewiesen ist, wird stattdessen die letzte vollständig verbundene Schicht verwendet.

### 4.3.2 Rastersuche

Für die Metaanalysen wird ein Datensatz mit einer Rastersuche über den Transferlernprozesshyperparameterraum zusammengestellt, welcher die Metadaten und Resultate der Transferlernprozesse in den verschiedenen Wertebereichen enthält. Der erste Teil des Datensatzes besteht aus dem Aufgabenvektor  $\{t_1, t_2, t_3, t_5, t_7\}$ , mit welchem alle Modelle, unter Verwendung der Optimierungsalgorithmen **Adam** [326], **RMSProp** [112] und einfachem stochastischen Gradientenabstieg in allen Kombinationen mit sowohl Kreuzentropie als auch negativem Log-Likelihood-Verlust als Optimierungskriterien, transferbelehrt wurden. Hierzu wurde jede dieser Kombinationen mit Lernraten  $\eta \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ , Epochenzahlen  $n_e \in \{1, 2, 3\}$  und Stapelgrößen  $b \in \{25, 50\}$  berechnet. Dieser Teil des Datensatzes beinhaltet Metadaten, Ergebnisse und Modelldateien von insgesamt 11.880 Transferlernprozessen. Für jeden dieser Prozesse werden auch die resultierenden Parametervektoren des neuronalen Netzwerks gespeichert, was insgesamt etwa  $1,6TB$  an Modelldaten für zukünftige Forschungen ergibt. Für einen anderen Teil des Datensatzes wurden weitere Metadaten gesammelt, indem jedes der 22 Bildverarbeitungsmodelle auf dem Aufgabenvektor  $\{t_1, t_2, t_3, t_5, t_6, t_7\}$  mit verschiedenen Lernraten  $\eta \in \{1, 0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}\}$  und Epochenzahlen  $n_e \in \{1, 3, 5, 7, 10\}$  transferbelehrt wurden, aber nur unter Verwendung des Optimierers **Adam** mit Kreuzentropieverlust, was zu insgesamt 3.960 weiteren Transferlerndatenpunkten führt. Um zur Überprüfung eines Härtefalls eine kaum übertragbare Aufgabe zu haben, wurden außerdem Daten aus Transferlernprozessen zu der harten Zielaufgabe  $t_4$  (Places365) hinzugefügt, welche mit Lernraten  $\eta \in \{1, \dots, 10^{-5}\}$  in einer einzigen Episoden unter Verwendung des Optimierers **Adam** und des Kreuzentropieverlustes berechnet wurden, womit dem Datensatz weitere 132 Stichproben für Testzwecke hinzugefügt wurden.

### 4.3.3 Metadatensatz

Für jeden Transferlernprozess werden Metadaten gesammelt, die sowohl die Einstellungen der Transferlernhyperparameter, die resultierenden Trainings- und Validierungsgenauigkeiten wie auch die entsprechende Trainings- und Testzeit in Sekunden, die Konfusionsmatrizen und den Ressourcenverbrauch auf dem Gerät enthalten. Zu den Transferlernhyperparametern gehören die Verlustfunktion  $L$ , der Optimierer  $o$ , die Stapelgröße  $b$ , die normalisierte Anzahl der Epochen  $n_e$  und die normalisierte Anzahl der Modellparameter  $n_p$  sowie deren Modellidentität  $m_i$ . Aus den Metadaten der Aufgabe und des Modells ist auch die Anzahl der Parameter bekannt, sowie die Statistiken über die Aufgabe, wie etwa die Anzahl der Datenpunkte  $n_d$ , die Anzahl der Klassen  $n_c$  und die stochastischen Momente des Labelraums  $\mu_{1,\dots,4}$ .

### 4.3.4 Metalernen

Dieser Metadatensatz mit 15.972 Datenpunkten aus Transferlernprozessen wird in einen Trainings- und Validierungsdatensatz unterteilt und derart organisiert, dass jede der Aufgaben auch eine völlig unbekannte Testaufgabe darstellt, welche auf Grundlage des

Lernens der Metadaten aus anderen Aufgaben abgeschätzt werden soll, um so die Generalisierungsfähigkeit auf Daten zu testen, die nicht in den Metalernprozess einbezogen sind. Aus dem Trainingsdatensatz werden zufällig Stapel der Größe 100 gezogen und dem Metalernprozess zugeführt. Hierbei wird die a-posteriori Variable der Validierungsgenauigkeit in der Zielaufgabe bei verschiedenen Eingabekonfigurationen des Metamodells als Grundlage der Abschätzung herangezogen, dies aber in zahlreichen Kombinationen von Optimierungsalgorithmen, Kriterien und Metalernhyperparametern. Für eine statistisch aussagekräftige Studie wurde jeder Metalernversuch zehnmal mit verschiedenen zufälligen Anfangsgewichtungsvektoren vorgenommen, um daraus Durchschnittswerte zu berechnen, zur Auswertung der maximalen Genauigkeit wird das beste Metamodell aus den zehn Versuchen herangezogen. Um eine Überanpassung beim Metalernen zu vermeiden findet ein frühzeitiges Stoppen Anwendung, wenn in zwei aufeinanderfolgenden Lernepochen der Fehler auf den Validierungsdaten steigt aber der Fehler auf den Trainingsdaten sinkt. Abbildung 4.7 skizziert die Architektur des Metamodells im Allgemeinen.

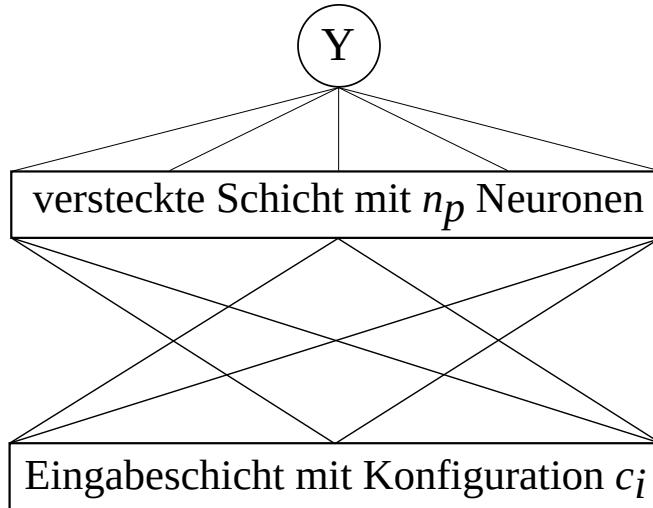


Abbildung 4.7: Das grundlegende Schema aller in den Untersuchungen aus [5] verwendeten mehrschichtigen Perzeptrone.

Für das Ausgangsneuron  $Y$ , welches versucht die Validierungsgenauigkeit zu nennen, wird eine sigmoidale Aktivierungsfunktion verwendet. In der verdeckten Schicht, deren Neuronenzahl in den Experimenten variiert, findet  $\tanh()$  als Aktivierungsfunktion Einsatz. Die Eingabeschicht enthält die Metadaten, welche in verschiedenen Eingabekonfigurationen untersucht werden können. Zur Eingabe des Optimierungsalgorithmus und das Optimierungskriterium werden binäre One-Hot-Encodings genutzt, zum Beispiel liest sich die Kodierung der drei Optimierungsalgorithmen als:

$$o_i = \begin{cases} [0, 0, 1] : & Adam \\ [0, 1, 0] : & RMSProp \\ [1, 0, 0] : & SGD \end{cases} \quad (4.1)$$

In gleicher Weise wird für die Darstellung des Optimierungskriteriums sowie für jedes der 22 Modellidentitäten verfahren; für  $m_1$  wird das erste Bit des Vektors auf 1 gesetzt, für Modell  $m_{22}$  das zweiundzwanzigste Bit. Die Anzahl der Modellparameter, Datenpunkte, Klassen, Stapelgrößen und Epochen wird über das Maximum aller entsprechenden Werte im Trainingsdatensatz normalisiert.

## 4.4 Auswertung

Um die Forschungsfrage nach der maschinellen Lernbarkeit von Wissensübertragungsprozessen zu beantworten, werden Versuche mit mehreren Aufgaben- und Modellklassen durchgeführt und ausgewertet. Die Auswertung einer Reihe von Experimenten soll mehrere Fragen beantworten; zum Einen, wie gut sich der Einsatz eines mehrschichtigen Perzeptrons als Metamodell im Vergleich zu einfacheren Ansätzen verhält; zum Anderen die Frage nach dem Nutzen verschiedener Eingabekonfigurationen für die Metamodelle sowie die Frage nach universell guten Parametern für die Anforderungen des Metalerenns mit Hilfe eines mehrschichtigen Perzeptrons. Da keine der bekannten aktuellen Methoden in dieser Form auf dieses Problem anwendbar ist, werden die Vorschläge des mehrschichtigen Perzeptrons mit einfachen Ansätzen der linearen Regression und mit dem besten aus der Trainingsmenge bekannten Transferlernverfahren verglichen. Es werden grundsätzlich zwei Möglichkeiten des Transferlernens betrachtet: die Feinabstimmung nur der Klassifizierungsschichten, bei der die Gewichte des gesamten Modells mit Ausnahme der letzten Schicht unverändert bleiben, sowie die Feinabstimmung aller Schichten, bei welcher sich die Gewichte im gesamten künstlichen neuronalen Netzwerk ändern. Verfügt das jeweilige künstliche neuronale Netzwerk über keine explizite Klassifizierungsstruktur, so wird stattdessen die letzte vollverbundene Schicht des Netzwerks angepasst.

### 4.4.1 Versuchsaufbau

Es werden Versuchsaufbauten mit verschiedenen Eingabekonfigurationen ausgewertet. Die unterschiedlichen Eingabekonfigurationen umfassen verschiedene a priori Variablen, welche bereits vor Beginn des entsprechenden Transferlernprozesses bekannt sind. Diese Evaluationsmethode dient als umfassender Vergleich des mehrschichtigen Perzeptrons mit der einfachen Verwendung des besten bisher bekannten Modells oder der Approximation mit einer linearen Regression. Auf Grundlage gleicher Eingabedatenlagen soll eine Aussage darüber getroffen werden, welche Metadaten besonders wertvoll für die Abschätzung eines Transferlernprozesses sind.

Alle Eingabekonfigurationen  $c_i$  beinhalten die normalisierte Stapelgröße  $b$ , die normalisierte Anzahl der Epochen  $n_e$  und die normalisierte Anzahl der Parameter des Transferlernmodells  $n_p$ . Außerdem enthält jede Eingabekonfiguration  $c_i$  die Lernrate  $\eta$  als Fließkommazahl sowie die Verlustfunktion  $L$ , den Optimierer  $o$  und die Modellidentität  $m_i$ , kodiert als binäre One-Hot-Vektoren. Es wird festgestellt, dass die Nichtberücksichtigung jeder dieser Variablen zu schlechteren Ergebnissen als mit ihnen führt, so dass eine Minimalkonfiguration in  $c_0 = \{b, n_e, n_p, \eta, L, o, m_i\}$  gefunden wird.

Für die Untersuchungen über den Effekt der stochastischen Momente der Labelverteilungen wird der Mittelwert der Quellaufgabe als  $\mu_{1_s}$  bezeichnet und den Mittelwert der Zielaufgabe als  $\mu_{1_t}$ . Ebenso werden die Standardabweichungen als  $\mu_{2_s}$ ,  $\mu_{2_t}$ , die Schiefe als  $\mu_{3_s}$ ,  $\mu_{3_t}$  und die Kurtosis als  $\mu_{4_s}$  oder  $\mu_{4_t}$  bezeichnet. In Kurzform werden die stochastischen Momente als  $\mu_{1,2,3,4}$  und die Anzahl der Datenpunkte und Klassen als  $n_c$  und  $n_d$  zusammengefasst, da die entsprechenden Werte der Quell- und der Zielaufgabe immer zusammen verwendet werden. Während die Konfiguration  $c_1$  zusätzlich das Verhältnis der Anzahl der Klassen und Datenpunkte  $r_c$  und  $r_d$  zwischen Quell- und Zielaufgabe enthält, enthält  $c_2$  stattdessen die Anzahl der Datenpunkte und Klassen in Quell- und Zielaufgabe  $n_{ds}$ ,  $n_{dt}$ ,  $n_{cs}$ ,  $n_{ct}$ . Die Konfiguration  $c_3$  enthält beide Darstellungen dieser Größen. Es wird die Kurtosis zu  $c_2$  hinzugefügt, um  $c_4$  zu erhalten, und die Schiefe zu  $c_4$  hinzugefügt, um  $c_5$  zu erhalten. Zusätzlich enthält  $c_6$  die Verhältnisse des Klassen und Datenpunkte,  $c_7$  das erste stochastische Moment, das an dieser Stelle als Kodierung für die Aufgabenidentität interpretieren werden kann. Die Konfiguration  $c_8$  berücksichtigt auch die Standardabweichung,  $c_9$  erweitert diese wieder um die Daten und Klassenverhältnisse. Tabelle 4.1 fasst die Unterschiede zwischen den Eingabekonfigurationen zusammen, die in den Metalernversuchen untersucht werden.

Als Qualitätsmaß für die Metalerner wird die Verlustreduktion in Prozenten genutzt:

$$\delta = 100 \frac{L_{start}}{L_{end}} - 100[\%] \quad (4.2)$$

Die durchschnittliche Verlustreduktion auf den Validierungs-, Trainings- und Testmetadatensätzen gilt als Maß für die Qualität des Metalernverfahrens. Hätte das Metalernverfahren den Verlust nicht reduziert, also  $L_{start} = L_{end}$ , würde dies zu  $\delta = 0\%$  führen; bei einer Verlustreduktion von  $L_{start} = 1$  auf  $L_{end} = 0.5$  wäre  $\delta = 100\%$ . Es wird die Verlustreduzierungsrate  $\delta$  anstelle des mittleren durchschnittlichen prozentualen Fehlers (MAPE) gewählt, weil die Qualität der Metalernprozesse selbst bewertet werden soll, unabhängig vom Definitionsbereich der untersuchten Verlustfunktionen.

#### 4.4.2 Ergebnisse

Zunächst werden die Maximalwerte von  $\delta$  in der Tabelle 4.3 betrachtet um herauszufinden, welche Metamodellkonfigurationen und Hyperparameter eine erhebliche Verlustreduktion und somit eine maximale Genauigkeit erreichen. Für die Trainingsmenge wird die Metamodellkonfiguration mit der größten Verlustreduktion in  $c_5$  gefunden. Im Einzelnen erreicht die Konfiguration  $c_5$  die maximale Verlustreduktion  $\delta_{train_{max}} = 209,71\%$  mit einer Verlustfunktion  $L = L2$ , einer Lernrate  $\eta = 10^{-3}$ , einer Anzahl von Epochen  $n_e = 20$  und einer Anzahl von versteckten Neuronen  $n_p = 100$  mit dem Optimierer  $o = RMSProp$  auf einer Trainingsmenge bestehend aus den Aufgaben *Hymenoptera*, *MNIST*, *FashionMNIST*, *Smartphones* und *Places365*. Für den Validierungsmetadatensatz wird ein Optimum bei  $\delta_{val_{max}} = 188,12\%$  mit der Eingabekonfiguration  $c = c_4$  mit den Metalernhyperparametern  $\{L = L2, \eta = 10^{-3}, n_e = 30, n_p = 100, o = RMSProp\}$  gefunden, trainiert auf Metadaten

<i>Konfiguration</i>	Zusätzliche EingabevARIABLEn
$c_1$	$r_c, r_d$
$c_2$	$n_c, n_d$
$c_3$	$n_c, n_d, r_c, r_d$
$c_4$	$\mu_4, n_c, n_d$
$c_5$	$\mu_3, \mu_4, n_c, n_d$
$c_6$	$\mu_3, \mu_4, n_c, n_d, r_c, r_d$
$c_7$	$\mu_1, \mu_3, \mu_4, n_c, n_d$
$c_8$	$\mu_1, \mu_2, \mu_3, \mu_4, n_c, n_d$
$c_9$	$\mu_1, \mu_2, \mu_3, \mu_4, n_c, n_d, r_c, r_d$

Tabelle 4.1: Die verschiedenen a priori Metadateneingabekonfigurationen  $c_1 \dots c_9$  bilden die Eingabeschichten der untersuchten Metamodelle.

der Aufgaben *MNIST*, *Hymenoptera*, *Smartphones*, *Places365* und *FashionMNIST*, validiert auf *CIFAR10*. Bei der Suche nach der maximalen Verlustreduktion auf den Testdatensätzen erreichte die Metamodellkonfiguration  $c = c_5$  eine Verlustreduktion  $\delta_{val_{max}} = 201.96\%$  mit  $\{L = L2, \eta = 10^{-3}, n_e = 20, n_p = 100, o = RMSprop\}$  für die Testaufgabe *FashionMNIST* und die Trainingsdatensätze mit *CIFAR10*, *MNIST*, *Hymenoptera*, *Smartphones* und *Places365*. Aus dieser Beobachtung kann geschlussfolgert werden, dass die Metamodelle in diesem Versuchsaufbau den Erfolg eines Transferlernprozesses für die Zielaufgabe *FashionMNIST* am zuverlässigsten abschätzen. Dies bedeutet, dass sich die Eigenschaften dieser Aufgabe gut aus den anderen ergeben. Die Metamodelle mit den Maximalwerten in der Verlustreduktion empfehlen die Transferlernprozesse, welche später im Vergleich verwendet werden. Weiterhin wird versucht die Frage zu beantworten, welche Hyperparameter in der Regel zu einem erfolgreichen Metalernverfahren führen, indem die durchschnittlichen Verlustreduktionen betrachtet werden. Auf der Suche nach einem Metalernverfahren mit guter Generalisierungsfähigkeit wird nicht nach Maximalwerten gesucht, sondern nach robusten Hyperparametern, die in allen Kombinationen von Trainings-, Validierungs- und Testdatensätzen gut abschneiden. In diesem Sinne bestätigt die Tabelle 4.2 überdurchschnittliche Leistungen der Eingabekonfigurationen  $c_4$  und  $c_5$ . Dies deutet darauf hin, dass die normalisierte Anzahl von Klassen und Datenpunkten zu besseren Vorhersagen führt als die Verwendung der entsprechenden Verhältnisse. Darüber hinaus kann gesehen werden, dass die Verwendung der dritten und vierten stochastischen Momente als EingabevARIABLEn einen positiven Effekt auf die Reduktion des mittleren Verlustes hat, aber dass die ersten und zweiten stochastischen Momente den Lernprozess zu verzerrten scheinen, da tatsächlich Verlustzunahmen in  $c_7$ ,  $c_8$  und  $c_9$  gefunden werden, wenn Mittelwert und Standardabweichung als Eingaben hinzukommen. Dies kann als eher kontraintuitiv gesehen werden, da vermutetet werden kann, dass die Mittelwerte die Aufgaben implizit so kodieren würden, dass die Metamodelle absurd genau werden könnten, aber stattdessen wird festgestellt, dass ein großer Eingabewert den Gradientenabstieg negativ beeinflusst.

$c$	$\overline{\delta_{val}}$	$\overline{\delta_{train}}$	$\overline{\delta_{test}}$
$c_4$	<b>31.01%</b>	42.27%	10.87%
$c_5$	29.32%	<b>42.29%</b>	<b>11.56%</b>
$c_2$	29.32%	38.67%	9.31%

Tabelle 4.2: Die drei höchsten durchschnittlichen Verlustreduktionen.

$c$	$\delta_{val_{max}}$	$\delta_{train_{max}}$	$\delta_{test_{max}}$
$c_4$	<b>188.12%</b>	200.55%	182.2%
$c_5$	162.07%	<b>209.71%</b>	<b>201.96%</b>
$c_2$	137.23%	138.37%	152.79%

Tabelle 4.3: Die drei höchsten maximalen Verlustreduktionen.

Um allgemeingültige optimale Hyperparameter für die mehrschichtigen Perzeptrone, welche die Metadaten lernen, zu finden, werden in den Tabellen 4.4 und 4.5 die vielversprechendsten Metalernhyperparameter und Architekturen aufgeführt. Tabelle 4.4 zeigt, dass das Optimierungskriterium  $L = L2$  und der Optimierungsalgorithmus  $o = RMSProp$  im Durchschnitt aller Trainings- und Validierungsversuche die höchste Punktzahl erreichen. Überraschender Weise erreicht der Optimierungsalgorithmus  $o = SGD$  mit dem Optimierungskriterium  $L = L2$  die höchsten Mittelwerte auf dem Testdatensatz. Es kann vermutet werden, dass dem geringen Unterschied keine statistische Signifikanz zuzuschreiben ist, da die Leistung auf den Trainings- und Validierungsdaten deutlich schlechter ist und es sich um einen glücklichen Zufall handeln könnte. Bei der weiteren Suche danach, welche neuronalen Architekturen zu Metamodellen mit hoher Generalisierungskraft führen, können in Tabelle 4.5 die besten Ergebnisse auf dem Trainings-, Validierungs- und Testdatensatz für 100 versteckte Neuronen mit einer Trainingszeit von 30 Epochen bei einer Lernrate von  $10^{-4}$  gesehen werden, gemittelt über alle Optimierungskriterien, Optimierungsalgorithmen und Eingabekonfigurationen. Interessanter Weise wird festgestellt, dass eine Erhöhung der Anzahl der Neuronen in der versteckten Schicht des Metamodells nicht zu höheren Lernerfolgen führt, da Metamodelle mit 100 Neuronen die Sachverhalte dieses Versuchsaufbaus bereits bestmöglich repräsentieren.

#### 4.4.3 Vergleich

Um die Validität der Metamodelle zu überprüfen und sie mit der Auswahl des bisher besten Lernverfahrens und des Vorschlags einer linearen Regression zu vergleichen, wird eine einfache Testaufgabe (*FashionMNIST*) und eine schwierige Testaufgabe (*Places365*) ausgewählt, um die Metamodelle für einen Transferlernprozess mit einer einzigen Episode abzufragen. Die Metamodelle, die für die Inferenz verwendet werden, hatten keine Daten der jeweiligen Testaufgabe im Trainings- oder Validierungsdatensatz gesehen und haben die höchste Verlustreduktion auf dem Testdatensatz.

$L$	$o$		$\overline{\delta_{val}}$	$\overline{\delta_{train}}$	$\overline{\delta_{test}}$
L2	Adam		-11.05%	-6.323%	-22.6%
L2	RMSProp		<b>31.82%</b>	<b>46.29%</b>	6.92%
L2	SGD		24.12%	30.48%	<b>8.23%</b>
L1	Adam		-4.45%	-2.4%	-15.43%
L1	RMSProp		18.45%	25.02%	1.67%
L1	SGD		12.04%	15.33%	-0.99%

Tabelle 4.4: Mittlere Verlustreduzierung in Prozent für alle Optimierungskriterien  $L$  und Optimierungsalgorithmen  $o$ , gemittelt über alle zehn Versuche und jede Metamodellarchitektur.

$n_p$	$n_e$	$\eta$	$\overline{\delta_{val}}$	$\overline{\delta_{train}}$	$\overline{\delta_{test}}$
100	10	$10^{-3}$	10.6%	17.6%	-6.4%
100	10	$10^{-4}$	23.8%	31.4%	8.9%
100	10	$10^{-5}$	11.6%	13.5%	3.5%
100	20	$10^{-3}$	14.2%	22.4%	-5.8%
100	20	$10^{-4}$	28.4%	38.3%	10.6%
100	20	$10^{-5}$	14.1%	16.8%	3.4%
100	30	$10^{-3}$	16.0%	23.4%	-3.9%
100	30	$10^{-4}$	<b>30.8%</b>	<b>41.9%</b>	<b>10.9%</b>
100	30	$10^{-5}$	16.1%	19.7%	4.9%
500	10	$10^{-3}$	-6.6%	-2.0%	-21.5%
500	10	$10^{-4}$	18.3%	26.6%	2.3%
500	10	$10^{-5}$	15.3%	18.6%	3.5%
500	20	$10^{-3}$	-3.2%	1.4%	-19.7%
500	20	$10^{-4}$	23.0%	33.0%	3.5%
500	20	$10^{-5}$	19.4%	24.3%	5.0%
500	30	$10^{-3}$	-2.9%	1.5%	-19.4%
500	30	$10^{-4}$	24.6%	35.7%	4.4%
500	30	$10^{-5}$	23.3%	29.7%	7.1%
1000	10	$10^{-3}$	-22.6%	-20.5%	-33.2%
1000	10	$10^{-4}$	11.3%	19.0%	-5.6%
1000	10	$10^{-5}$	16.4%	20.4%	4.3%
1000	20	$10^{-3}$	-22.2%	-19.9%	-33.8%
1000	20	$10^{-4}$	16.6%	26.2%	-2.9%
1000	20	$10^{-5}$	21.4%	27.4%	6.5%
1000	30	$10^{-3}$	-21.9%	-19.5%	-33.1%
1000	30	$10^{-4}$	19.1%	29.9%	-1.2%
1000	30	$10^{-5}$	24.5%	32.0%	8.1%

Tabelle 4.5: Mittlere prozentuale Verlustreduktion für alle Neuronenzahlen  $n_p$ , Epochenzahlen  $n_e$  und Lernraten  $\eta$ , gemittelt über alle zehn Versuche und jeden Metalernhyperparameter.

Die Genauigkeit wird für Lernraten  $\eta \in \{0.1, 10^{-3}, 10^{-4}, 10^{-5}\}$  geschätzt, und zwar für alle Modelle  $m_{1:22}$ , alle Optimierungsalgorithmen und Optimierungskriterien, unter Berücksichtigung der erforderlichen und vorher bekannten Metadaten, also der Schiefe und Kurtosis des Labelraums, der normalisierten Anzahl der Klassen und Daten beziehungsweise der Klassen- und Datenverhältnisse. Da Parameter für einen Transferlernprozess mit nur einer Episode abgeleitet werden sollen, wird  $n_e = \frac{1}{10}$  gesetzt, da ein Maximalwert von zehn für die Anzahl der Epochen in den Trainingsdaten bekannt ist und die Leistung nach einer Episode abgeschätzt werden soll. Der Wert für die normalisierte Stapelgröße wird auf  $b = 1.0$  gesetzt und die Vorhersagen des Metamodells mit den tatsächlichen Transferlernergebnissen aus dem Testdatensatz verglichen, wie in Abbildung 4.9 gezeigt. In den entsprechenden Graphen in Abbildung 4.9 kann gesehen werden, dass das Metamodell die erwartete Genauigkeit in Bezug auf Lernraten und Modellidentitäten für die leichte Aufgabe besser differenziert als für die schwierige Aufgabe. Außerdem sind glatte Vorhersagen erkennbar, was als Bestätigung für den Erfolg des Metalernens angesehen werden kann, da keine Sprünge in den Vorhersagen auftreten. Dies zeigt zwar einen erfolgreichen Metalernprozess, aber andererseits auch, dass die Vorhersagequalität mit zunehmender Komplexität der Zielaufgabe abnimmt. Werden die vorhergesagten und tatsächlichen Genauigkeiten direkt für eine Wahl des Optimierers und des Kriteriums verglichen, so kann in Abbildung 4.9 gesehen werden, dass das Metamodell die Modellgenauigkeiten über die Lernraten für die einfache Aufgabe (*FashionMNIST*) in praktisch brauchbarer Weise reproduzieren konnte; beispielsweise für die Trajektorien  $m_1$  und  $m_2$  im Vergleich zur Trajektorie  $m_{22}$ . Die schwierige Aufgabe (*Places365*) enthält einen Sprung bei einer Lernrate von  $\eta = 10^{-3}$ , welchen das Metamodell nicht antizipieren konnte, so dass die Vorhersagen kaum zu den Daten passen.

Um schließlich die *Transfer Meta Learning* Methode mit der einfachen Verwendung der besten Einstellung, die aus dem Trainingsmetadatensatz bekannt ist, und einem Vorschlag durch lineare Regression, der ebenfalls auf dem Trainingsmetadatensatz trainiert wurde, zu vergleichen, sind in den Tabellen 4.6 und 4.7 die Testergebnisse für alle Optimierungsalgorithmen und Kriterien aufgeführt. Aus Tabelle 4.6 kann gelesen werden, dass das Metamodell in der Tat eine gute Konfiguration vorgeschlagen hat, die zu einem höheren Testrang als bei den anderen Ansätzen führt. Das Ergebnis für die schwierige Aufgabe in Tabelle 4.7 bestätigt den Vorteil der Verwendung einer MLP-Vorhersage anstelle einer linearen Regression oder einfach des besten bekannten Ansatzes, da es auch in diesem Fall besser abschneidet als die Baselines. Um einen fairen Vergleich zu ermöglichen, enthält die lineare Regression dieselben EingabevARIABLEN wie die Konfiguration  $c_5$ . Als einfache Basislinie wird die beste Konfiguration verwendet, die im Trainingsdatensatz zu finden ist. Beide Baselines kennen keine Daten aus dem Testdatensatz.

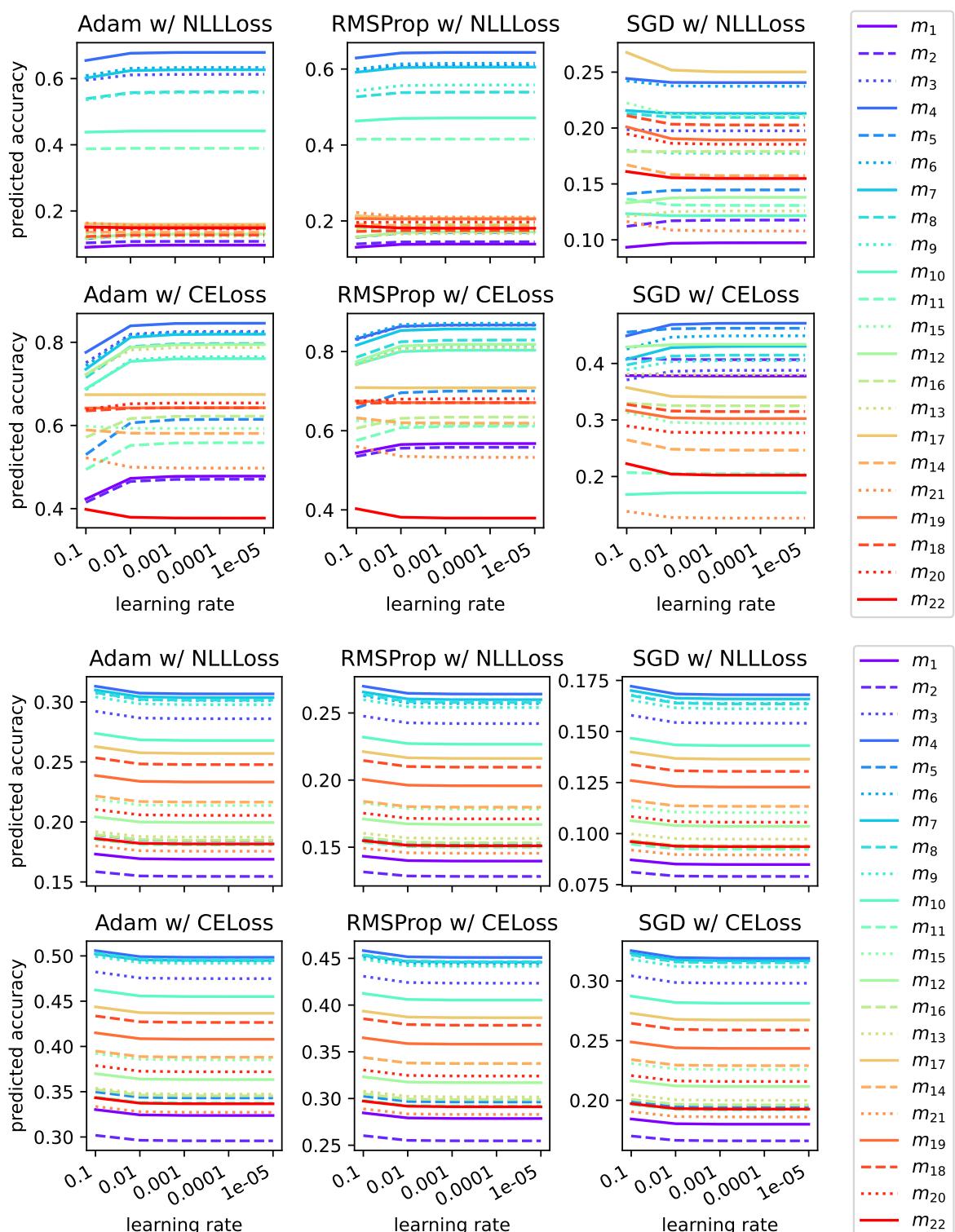


Abbildung 4.8: Metamodellinferenzen für eine leichte Aufgabe (*FashionMNIST*, oberes Raster) und eine schwere Aufgabe (*Places365*, unteres Raster), basierend auf [5].

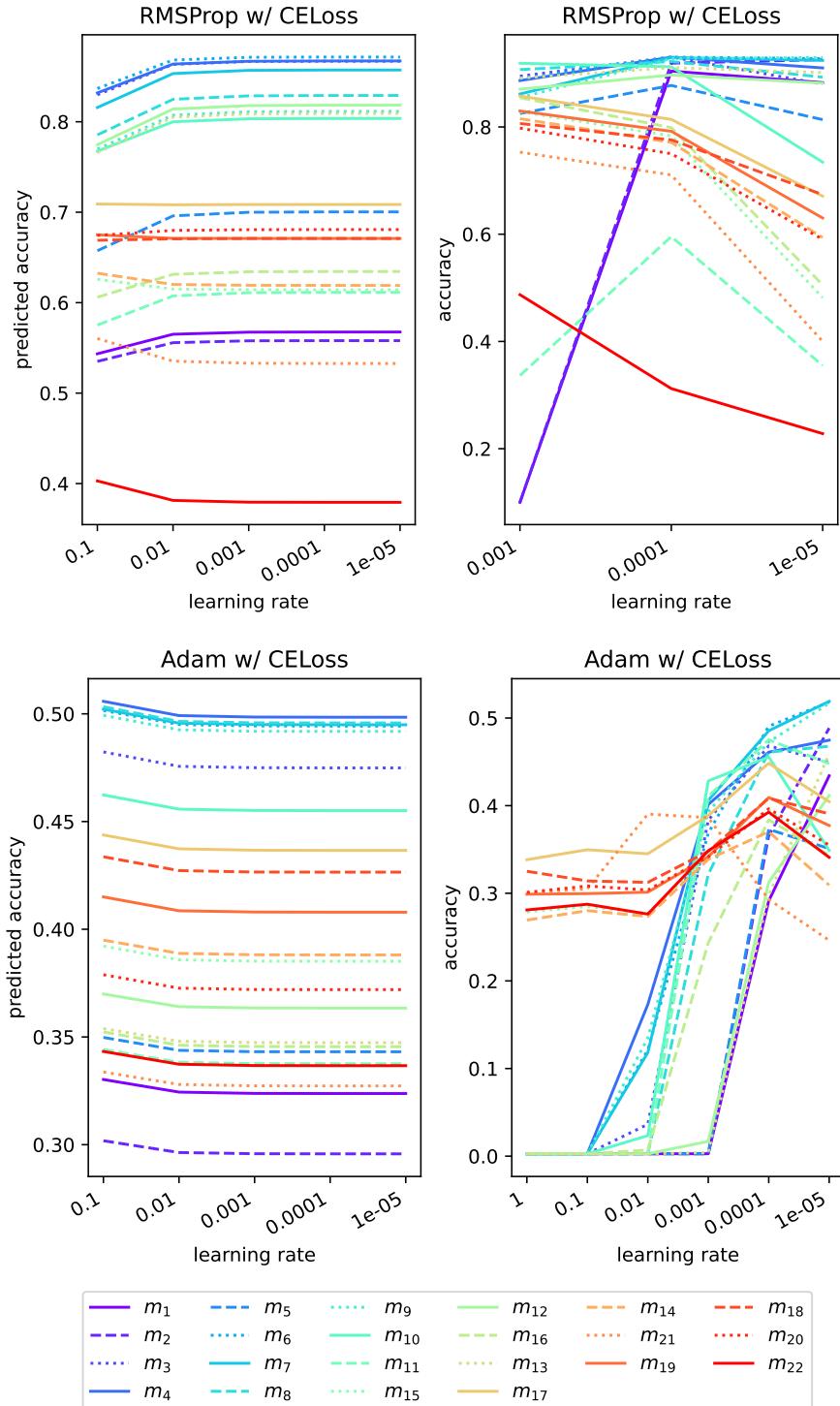


Abbildung 4.9: Metamodellinferenzen (links) im Vergleich zu den tatsächlichen Modellgenauigkeiten (rechts) bei den entsprechenden Testaufgabendaten von *FashionMNIST* (oben) und *Places365* (unten), entnommen aus [5].

<i>Rang</i>	<i>G</i>	$m_i$	$\eta$	<i>o</i>	<i>L</i>
1	93.7%	$m_6$	$10^{-5}$	<i>Adam</i>	<i>CE</i>
2	93.5%	$m_7$	$10^{-4}$	<i>Adam</i>	<i>CE</i>
3	93.3%	$m_9$	$10^{-5}$	<i>Adam</i>	<i>CE</i>
...					
<b>24</b>	<b>92.6%</b>	<b><math>m_2</math></b>	<b><math>10^{-5}</math></b>	<b>RMS</b>	<b>CE</b>
...					
<b>46</b>	<b>91.0%</b>	<b><math>m_4</math></b>	<b><math>10^{-5}</math></b>	<b><i>RMS</i></b>	<b><i>CE</i></b>
...					
<b>58</b>	<b>90.3%</b>	<b><math>m_{13}</math></b>	<b><math>10^{-5}</math></b>	<b><i>Adam</i></b>	<b><i>CE</i></b>
...					
538	7.7%	$m_{10}$	$10^{-4}$	<i>SGD</i>	<i>CE</i>

Tabelle 4.6: Testergebnisse für die Aufgabe *FashionMNIST*, geordnet nach Genauigkeit. Der Modellvorschlag des MLP in fetter Schrift, der beste bekannte Ansatz aus dem Trainingsset in blau und der lineare Regressionsvorschlag in türkis.

<i>Rang</i>	<i>G</i>	$m_i$	$\eta$	<i>o</i>	<i>L</i>
1	51.91%	$m_7$	$10^{-5}$	<i>Adam</i>	<i>CE</i>
2	51.81%	$m_6$	$10^{-5}$	<i>Adam</i>	<i>CE</i>
3	51.68%	$m_9$	$10^{-5}$	<i>Adam</i>	<i>CE</i>
...					
<b>7</b>	<b>47.5%</b>	<b><math>m_4</math></b>	<b><math>10^{-5}</math></b>	<b><i>Adam</i></b>	<b><i>CE</i></b>
...					
<b>13</b>	<b>45.95%</b>	<b><math>m_{13}</math></b>	<b><math>10^{-5}</math></b>	<b><i>Adam</i></b>	<b><i>CE</i></b>
...					
<b>85</b>	<b>17.2%</b>	<b><math>m_4</math></b>	<b><math>10^{-4}</math></b>	<b><i>Adam</i></b>	<b><i>CE</i></b>
...					
141	0.26%	$m_{10}$	1.0	<i>Adam</i>	<i>CE</i>

Tabelle 4.7: Tatsächliche Testergebnisse für die Aufgabe *Places365*, sortiert nach Genauigkeit. Der Modellvorschlag des MLP in fetter Schrift, der beste bekannte Ansatz aus dem Trainingsset in blau und der lineare Regressionsvorschlag in türkis.

# 5 Diskussion

Die untersuchten wissenschaftlichen Grundlagen des maschinellen Lernens, insbesondere des Trainings tiefer künstlicher neuronaler Netzwerke, wurden in der Praxis zur Lösung verschiedener Problemstellungen angewandt. Die konkreten Anwendungen, also der automatisierte Handel mit Differenzkontrakten, die Metastudie zur Handgestenerkennung, die Personenerkennung und Personenverfolgung sowie das Transferlernen zur Bildklassifikation bieten praktisch nützliche Programmvorlagen und Orientierungshilfen für die Lösung praktischer Herausforderungen. Dabei zeigen sich die Vorteile des Transferlernens und der Bedarf nach einer systematischen Herangehensweise, um zielführende Transferlernprozesse mit geringem Forschungsaufwand, also ohne menschliches Expertenwissen, nur auf Basis von Metadaten zu finden. Aus diesen Herausforderungen ergibt sich die, nach aktuellem Stand der Wissenschaft neue, Methode des *Transfer Meta Learning*.

Das Konzept des *Transfer Meta Learning* erlaubt eine systematische Aufarbeitung aller im Rahmen der Anwendungen angefallenen Metadaten in objektorientierten Strukturen und zeigt eine Möglichkeit auf, die gewonnenen Erkenntnisse methodisch auf neue Aufgaben zu übertragen. Die durchgeführten Experimente zur Überprüfung der Methodik hätten ausführlicher gestaltet werden können, indem beispielsweise auch Bildsegmentierung oder Objektdetektion untersucht hätten werden können. Dies übersteigt jedoch den Rahmen der Verhältnismäßigkeit, der durch die Zeit- und Rechenressourcen zur Verfügung steht; denn auch so zeigt sich hier die Machbarkeit eines vielversprechenden, neuen Ansatzes. Aber auch diese Methode unterliegt informationstheoretischen Limitationen, welche im entsprechenden Abschnitt näher betrachtet werden. Die wissenschaftliche Arbeit an diesem Thema wird fortgeführt und ein Ausblick auf kommende Forschung ist lohnenswert, um die Richtung der Entwicklung abzuschätzen. Abschlussarbeiten der akademischen Grade Bachelor und Master haben an dieser Stelle Vorarbeiten geleistet, welche in weiteren Forschungsprojekten fortgeführt werden können [233, 327, 328, 329]. Bei allen Anwendungen der Informatik, insbesondere der künstlich nachgebildeten Intelligenz, welche das Potential tiefgreifender gesellschaftlicher Änderungen bergen, sind ethische Betrachtungen von essentiellem Wert, um Gefahren und Missbrauchspotentiale im Vorhinein zu erfassen und die Technik idealerweise zum Wohle der Menschheit einzusetzen. Fairness in künstlich intelligenten Anwendungen ist daher ein stark diskutiertes Thema, da solche Systeme stets in Zusammenhang mit gesellschaftlichen Leitfragen, wirtschaftlichen Privilegien und sozioökonomischen Konzepten stehen [330, 331].

## 5.1 Limitationen

Alle Optimierungsverfahren unterliegen mathematischen Grenzen bezüglich ihrer Verlustfunktion; das dem *Transfer Meta Learning* zugrundeliegende Deep Learning System unterliegt zunächst denselben informationstheoretischen Grenzen wie alle anderen Deep Learning Systeme. Es wird also angenommen, dass das Modell mit der Datenkomplexität skaliert und, wenn es mit unzureichenden oder falschen Metadaten trainiert wird, entsprechend ineffiziente Transferlernmethoden vorschlagen würde, welche diese Trainingsdaten widerspiegeln. Insbesondere bei Anwendungen des kontinuierlichen Lernens kann diese allgemeine Schwäche zu einem selbstverstärkenden Problem führen. Die oberen Grenzen der Methode liegen in den oberen Grenzen der universellen Funktionsannäherung im Allgemeinen, beziehungsweise der universellen Funktionsannäherung mehrschichtiges Perzeptrons im Besonderen, da die Metalernoperationen ein solches künstliches neuronales Netzwerk benutzt.

Aus informationstheoretischen Überlegungen heraus wird festgestellt, dass die oberen Schranken, wie sie in [35] ausgearbeitet wurden, auch für diese Methode gelten. Vereinfacht zusammengefasst findet sich die obere Grenze für die durchschnittliche Generalisierung des Metalerns im Erwartungswert der Verlustfunktion für eine durchschnittliche Informationsdichte  $I$  mit  $N$  Trainingsmetadaten und  $M$  Metadaten pro Aufgabe für eine zufällige Zielaufgabe  $T$ , unter der Annahme dass sich die Daten der Zielaufgabe aus den Daten der Trainingsaufgaben abbilden lassen:

$$\mathbb{E}[L] \leq \frac{1}{N} \sum_{i=1}^N \sqrt{2\sigma^2 I} + \mathbb{E}\left[\sum_{j=1}^M \sqrt{2\sigma_T^2 I}\right] \quad (5.1)$$

Bezüglich der Limitationen der durchgeführten Experimente wird die geringe Anzahl von Aufgaben in der Zieldomäne als eine Einschränkung gesehen, insbesondere da nur eine Quellaufgabe verwendet wurde. Obwohl die wenigen Aufgabenkombinationen bereits zahlreiche Transferlernprozesse hervorgebracht haben, deren Metadaten mit akzeptablen Ergebnissen für Metalernmethoden genutzt werden konnten, würden entsprechend der oberen informationstheoretischen Grenzen mehr Quell- und Zielaufgaben die Metalernprozesse verbessern. Da der vorgestellte Ansatz zunächst nur Aufgaben aus der Zieldomäne der Bildklassifikation adressiert, wird die fehlende Erweiterung der Methodik auf verwandte Lernprobleme nicht als Einschränkung des Ansatzes, sondern als zukünftige Arbeit für weitere Forschungsarbeit betrachtet. Eine weitere Einschränkung der Untersuchungen liegt darin, dass nur die Genauigkeit auf dem Validierungsset der Zielaufgabe vorhergesagt wurde; denn es wurden auch die Genauigkeit auf der Trainingsdatenmenge, die Lernzeiten, die Inferenzzeiten und die Speicherverbräuche erfasst, aber die Untersuchung der Vorhersage dieser Werte wird ebenfalls zukünftigen Forschungen überlassen. Weiterhin sollte an dieser Stelle erwähnt werden, dass inkonsistente Modelltrainingszeiten beobachtet wurden; mehr oder weniger ähnliche Lernprobleme erforderten sehr unterschiedliche Rechenzeiten. Es wird vermutet, dass die Ursache für dieses Problem in der Verwendung paralleler Datenspeicher auf verschiedenen GPUs liegt.

## 5.2 Ausblick

Zunächst soll in zukünftiger Forschung die Anwendbarkeit des *Transfer Meta Learning* auf verwandte Lernprobleme der Bildverarbeitung, wie beispielsweise Bildsegmentierung [332] oder Bildrekonstruktion [333], untersucht werden. In diesen zukünftigen Studien können auch andere Architekturen von künstlichen neuronalen Netzwerken mit einbezogen werden, wie beispielsweise Variationale Autoencoder (VAE), Long Short-Term Memory (LSTM) oder Generative Adversarial Networks (GAN). Zukünftige Forschung sollte andere Transferlernmethoden als die Feinabstimmung einbeziehen, wie beispielsweise die im Stand der Wissenschaften erläuterte Elastic Weight Consolidation [23], das Incremental Moment Matching [25], die progressiven neuronalen Netzwerke [46] oder der Transfer über Weltmodellwissen [19]. Darüber hinaus wurden auch Trainingsgenauigkeiten, Konfusionsmatrizen, Ressourcennutzungen und Inferenzzeitinformationen aus den Transferlernprozessen gesammelt, sodass zukünftige Forschungsarbeiten die Auswirkungen der Vorhersage dieser Daten als zusätzliche Verlustfunktionen untersuchen können. Nachdem all diese Effekte und Methoden in einem *Transfer Meta Learning* Verfahren untersucht wurden, kann geplant werden, wird eine Basisstudie mit anderen Benchmarks durchzuführen, wie zum Beispiel VTAB [320], Causalworld [334] oder Erkennung von Herz-Kreislauf-Erkrankungen [335]. Bei diesen Studien kann auch der Frage nachgegangen werden, inwieweit die Anwendung des Metalernens auf die von *Transfer Meta Learning* erstellten Metadaten helfen könnte, genauere Vorhersagen zu treffen. Offen bleibt die grundsätzliche Frage, ob und wie ein maschinelles Lernverfahren selbstständig neue Wissensquellen erschließen kann, ohne dass dies des Wissens eines Menschen bedarf; ein kontinuierlicher Lernprozess könnte *Transfer Meta Learning* nutzen, um selbstständig einen Raum von Aufgaben und Modellen zu ordnen, welche für die Erschließung neuer Datenquellen verwendet werden können.

Weitere Forschungsarbeiten können die Vorteile der Methode bei der Lösung von Aufgaben des auf visuellen Verstärkungslernens examinieren, wie beispielsweise auf den verschiedenen Szenarien der Umgebung ViZDoom [182]. Vorarbeiten hierzu wurden in der Bachelorarbeit von [327] geleistet, in welcher ein Weltmodelltransfer nach dem Vorbild von [19] in der ViZDoom-Umgebung zur Lösung einer Ausweichaufgabe untersucht und bestätigt wurde, dass ein solcher Wissenstransfer mit geringdimensionalen Kontrollnetzwerken modellierbar ist. Aufbauend auf einer Implementierung des Spiels Tron von [14], welche prinzipiell beliebige Feldgrößen, Spielerzahlen und Belohnungssysteme erlaubt, untersucht die Masterarbeit von [328] die Auswirkungen der Belohnungssysteme auf die gelernte Strategie und kommt zu dem Schluss, dass bereits geringe Änderungen in den Belohnungssignalen zu stark unterschiedlichen Strategien führen können. Die Interpretation dieser Änderungen der Umgebungsbedingungen und Belohnungssysteme als einzelne Aufgabeninstanzen eröffnet eine nützliche Anwendung von *Transfer Meta Learning*. Mit den auf diese Weise erhobenen Metadaten kann herausgefunden werden, bei welchen Belohnungssystemen welche Anpassungen von vortrainierten Modellen mit welchen Hyperparameter zum schnellen Transferlernen von optimalen Strategien führen. In ähnlicher Weise kann die Forschung an simulierten Finanzmärkten die Methode des *Transfer Meta Learning* verwenden, indem das Handeln eines jeweiligen Wertes als eigene

Aufgabe verstanden wird. In der Bachelorarbeit von [329] wurden verschiedene Modelle und Hyperparameter im Kontext des bestärkenden Lernens für Differenzkontraktionshandel untersucht, welche hierzu als Datengrundlage dienen können. Ferner kann zur Beurteilung der Transferlernfähigkeiten in zukünftigen Experimenten versucht werden, Wissen aus dem REHAP Handgestendatensatz auf ein andere Handgestendatensätze zu übertragen. Im Sinne des *Transfer Meta Learning* kann die zukünftige Forschung einheitliche und systematische Vergleiche über alle Handgestenerkennungssysteme und den jeweiligen Datensätzen erbringen, die in den letzten Jahren entstanden sind. Dies verspricht ein detailliertes Bild, welche Teile des Systems geändert werden könnten, um eine intuitivere, menschlichere Maschineninteraktion zu ermöglichen. An dieser Stelle können auch Metadaten über andere Systeme, welche mit Raumtiefenbildern arbeiten, in den Lernprozess integriert werden. So stellen beispielsweise die in der Bachelorarbeit von [233] untersuchten Raumtiefenfilter und deren Hyperparameter eine Datengrundlage für Metalernprozesse dar, die unter Umständen für Transferlernen anderer Raumtiefenbildsysteme weiterverwendet werden können.

Bezüglich der Personenerkennung und Personenverfolgung kann die Methode des *Transfer Meta Learning* genutzt werden, um die Positionsbestimmung und Spurverfolgung für neue Umgebungen einfacher zu implementieren. Wenn Metadaten über Modelle und Kameraeinstellungen bekannt sind, kann aus diesen systematisch eine Konfiguration abgeleitet werden, welche in anderen Blickwinkeln und Sichtverhältnissen eine robuste Detektion liefert. Da die meisten Personenerkennungsmodelle, wie beispielsweise OpenPose, im Prinzip schon fertig gelernt sind, erübrigt sich an dieser Stelle die Notwendigkeit des Transferlernens zur Menschenerkennung. Allerdings können Transferlernmethoden genutzt werden, um mit Objektdetektionsnetzwerken, wie etwa YOLO, schnell neuartige Objekte lernen und in einem Kamerabild wiederfinden zu können.

## 5.3 Ethische Betrachtungen

Informatik im Allgemeinen und künstliche Intelligenz, beziehungsweise maschinelles Lernen, im Besonderen, erlaubt es, immer mehr arbeitsintensive Abläufe in der menschlichen Gesellschaft zu automatisieren. Algorithmen des maschinellen Lernens können dabei stets nur so gut sein, wie die Daten, aus denen sie lernen. Wenn sich vorurteilsbehaftete Diskriminierung bereits in den Trainingsdaten manifestiert, führt dies zu Programmen und Modellen, welche diese Diskriminierung aufrechterhalten und somit systematisch gegen Menschen- und Persönlichkeitsrechte verstößen können [330]. Die von Menschen getroffene Auswahl und Kozeptionierung der Trainingsdaten ist daher von entscheidender Bedeutung für maschinelle Lernsysteme, die zum Wohle der Gesellschaft eingesetzt werden sollen. Nach der Sapir-Whorf-Hypothese, einer Annahme der Linguistik, beeinflusst die Sprache das Denken; sodass nur wahrgenommen wird, wofür Konzepte existieren und nur Konzepte für etwas existieren, wofür es sprachliche Ausdrücke gibt [336]. Für datengetriebene maschinelle Lernsysteme bedeutet dies, dass nur aus dem gelernt werden kann, was präsentiert wird und nur das präsentiert werden kann, wofür sprachliche Kategorien bekannt sind.

Die Auswahl der Datensätze und somit die maschinell gefestigten Vorurteile hängen also stark von der Denkweise der Menschen ab, welche diese Systeme implementieren. Insofern stellen ethische Debatten essentielle Fragen an den wissenschaftlichen Umgang mit und den technischen Einsatz von maschinellen Lernsystemen in Bezug auf die Humanität [331]. An dieser Stelle wird auf die Bedeutung der Erklärung von Toronto hingewiesen, welche Entscheidungsträger dazu aufruft, technischen Fortschritt nicht auf Kosten der Menschenrechte zu fördern.

Als konkrete Lösungsvorschläge, welche es Entwicklern ermöglichen, vorurteilsfreie Datensätze und Implementierungen zu verwirklichen, bieten sich zunächst unternehmensinterne Schulungen und Checklisten an. Weiterhin existieren Ansätze zur systematischen Verbesserung der Fairness von künstlich intelligenten Systemen durch statistisch validierte Abmilderungen von Vorurteilen in Daten. Mit den Erkenntnissen von [337] wird gezeigt, dass die Fairnesskalibrierung nur mit einer einzigen Fehlerbedingung nicht besser ist als die Zufallsauswahl von Vorhersagen eines bestehenden Klassifikators. Der Beitrag von [338] schlägt ein konkaves Optimierungsverfahren für das Lernen einer Datenumwandlung mit drei Zielen vor: die Kontrolle von Diskriminierung, die Begrenzung der Verzerrung in einzelnen Datenproben bei Erhaltung des Nutzens der Systeme. Der Metalernalgorithmus von [339] bezieht eine Klasse von Fairnessbedingungen in Bezug auf mehrere nicht disjunkte sensible Attribute mit ein, welche mit beweisbaren Garantien ausgestattet sind, indem zunächst eine Menge von Klassifizierungsproblemen mit konkavem Bedingungen entwickelt und dann gezeigt wird, dass Klassifikationsprobleme mit allgemeinen Arten von Fairnessnebenbedingungen auf die ausgeführten Probleme reduziert werden können. Die von [340] durchgeführten Untersuchungen verknüpfen ungleiche Auswirkungen mit einem Maß für Klassifizierungsgenauigkeit und schlagen einen Test auf unterschiedliche Auswirkungen vor, welcher darauf basiert, wie gut eine geschützte Attributsklasse, beispielsweise Geschlecht oder Hautfarbe, aus anderen Attributen vorhergesagt werden kann. Die Studien von [341] identifizieren Bedingungen für gestörte Daten, mit denen eine Abmilderung der Verzerrung eines Klassifikators durch einen Quotenausgleich garantiert wird. Im Artikel von [342] werden zwei Lösungen für eine diskriminierungssichere Klassifizierung präsentiert, die weder eine Datenänderung noch eine Anpassung des Klassifikators erfordern, indem Zurückweisungsoptionen von probabilistischen Klassifikatoren und Unstimmigkeitsregionen genutzt werden, um Diskriminierung zu reduzieren. In der Ausarbeitung von [343] werden drei Ursachen für Unfairness beim maschinellen Lernen erörtert und anschließend ein Regularisierungsansatz vorgeschlagen, der auf jeden Vorhersagealgorithmus mit probabilistischen diskriminativen Modellen anwendbar ist. Von der Sklaverei der Vorurteile befreite Gedanken, welche letztendlich in lernenden Maschinen zu verborgenen gesellschaftlichen Grundsätzen werden können, sind kein Selbstzweck sondern führen zu konkreten Veränderungen gesellschaftlicher Bedingungen und leisten Beiträge zur Vermeidung von gewaltsamen Konflikten und zu einer menschlicheren Gesellschaft. Künstlich intelligente Methoden finden in militärischen Bereichen Anwendung [344, 345], die Entwicklung tödlicher autonomer Waffensysteme birgt ein enormes Gefahrenpotential für die Menschheit [346]; aber auch bestehende zivile Anwendungsfälle, wie beispielsweise Fälschungen im Allgemeinen oder Deep Fakes im Besonderen, sowie gezielte Marktmanipulationen oder Unterstützungen bei Straftaten

gegen Persönlichkeitsrechte stellen zunehmend zivilrechtliche Verstöße dar [347]. Auf der Mikroebene kann gezielte manipulative politische Werbung demokratische Willensbildung verzerren [348, 349]. Haftungsfragen bei künstlich intelligenten Systemen sind in vielen Fällen ungeklärt und Gegenstand juristischer Diskussion. Artikel Zwölf des Übereinkommens der Vereinten Nationen über die Verwendung elektronischer Kommunikationsmittel bei internationalen Verträgen besagt, dass eine Person, in deren Auftrag eine Rechenmaschine programmiert wurde, letztlich für jede von der Maschine erzeugte Nachricht verantwortlich ist [350].

Vor diesem Hintergrund werden mögliche Konsequenzen der Forschungsarbeiten, welche im Rahmen der in dieser Dissertation beschriebenen Herausforderungen als Lösungen präsentiert werden konnten, auf die Gesellschaft diskutiert. Das Verstärkungslernen auf Marktdaten kann zur wirtschaftlichen Manipulation genutzt werden, indem als Optimierungskriterium nicht eigener Profit sondern wirtschaftlicher Niedergang eines Zielunternehmens gewählt wird. Die Techniken zur Personenerkennung und Personenverfolgung könnten als Grundlage für Massenüberwachungssysteme verwendet werden. Obwohl sich die Methode des *Transfer Meta Learning* nicht direkt mit kriminellen Anwendungen, Massenüberwachung oder tödlichen autonomen Waffensystemen befasst, muss bedacht werden, dass sich böswillige Akteure durch den Einsatz der zugrundeliegenden Methode in solchen Anwendungen einen Vorteil verschaffen könnten, indem sie mehr Lösungen in derselben Zeit berechnen. Da sich diese Methode mit grundlegenden Überlegungen zur Optimierung des Rechenaufwands befasst und an sich keine personenbezogenen Daten verwendet, bestehen zunächst keine Einwände gegen die Verwendung der Daten und Algorithmen in datenschutzrechtlich sensiblen Bereichen oder sicherheitskritischer Infrastruktur. In diesem Sinne verletzen die vorgestellten Methoden und Modelle nach besten Wissen nicht die Menschenrechte, die Autonomie oder die Würde der Menschen. Es werden keine negativen Auswirkungen auf die Umwelt festgestellt; im Gegenteil kann argumentiert werden, dass die Methode des *Transfer Meta Learning* dazu beitragen könnte, die Energiekosten und die damit verbundenen Kohlendioxidemissionen zu senken, wenn bestehende Probleme mit weniger Rechenaufwand gelöst werden.

# 6 Zusammenfassung

Diese Dissertation untersucht die wissenschaftlichen Grundlagen selbstlernender Systeme und des maschinellen Wissenstransfers. In praktischen Anwendungen werden diese Grundlagen genutzt, um mit aktuellen maschinellen Lernverfahren konkrete Herausforderungen zu lösen. Hierbei zeigt sich der Nutzen von Transferlernmethoden zur Komplexitätsreduktion. Um ein systematisches Herangehen an das Transferlernen zu ermöglichen, wird die neue Methode des *Transfer Meta Learning* eingeführt, welche anhand von Metadaten automatisch optimale Hyperparameter für Transferlernprozesse ermitteln kann. Insbesondere benötigt dieser Ansatz nur Metadaten und keine tatsächlichen Stichproben aus dem Datensatz, um Transferlerneinstellungen vorzuschlagen, die besser sind als die einfache Verwendung der besten bekannten Einstellungen oder die Herleitung von Einstellungen durch eine lineare Regression.

Die Herausforderungen umfassen ein bestärkend lernendes Agentensystem für den Differenzkontrakt Handel, eine Metastudie zu Handgestenerkennungssystemen, Systeme zur Erkennung und Verfolgung von Personen auf Kamerabildern und eine Anwendung von Transferlernen zur Bildklassifikation im Bereich der Kreislaufwirtschaft. Bei allen Anwendungen entstanden verschiedene vortrainierte Modelle samt zugehöriger Metadaten, welche die Grundlage für die Anwendung von *Transfer Meta Learning* in zukünftiger Forschung bilden können. Beim automatischen Handelssystem entstammen die Metadaten aus Handelsaufgaben ausgewählter Basiswerte mit unterschiedlichen Beobachtungsbedingungen. Die Metadaten der Handgestenerkennungssysteme beschreiben Leistungen diverser Architekturen auf den entsprechenden Handgestendatensätzen, die Metadaten der Personenerkennungs- und Verfolgungssysteme Erkennungsgenauigkeiten und Ressourcenverbrauch. In der Bildklassifikationstranferanwendung zeigt die Notwendigkeit einer systematischen Herangehensweise, die entstandenen Metadaten bilden einen Teil der Datengrundlage für die Machbarkeitsstudie des *Transfer Meta Learning*.

Dieser neue Ansatz zum Lernen des Transferlernens durch Metalernen wurde am Beispiel der Bildklassifikation ausgeführt. Die Ergebnisse zeigen, dass Metamodelle dabei helfen Rechenkosten zu senken, indem sie geeignete Einstellungen für Transferlernprozesse bezüglich einer Zielaufgabe vorschlagen können. Auf Grundlage der beschriebenen Herausforderungen wurde der Einsatz dieses Ansatzes in zukünftiger Forschung diskutiert. Es wurde diskutiert, dass sich weitere Forschungen zur Übertragbarkeit dieses Ansatzes auf verwandte Probleme, wie etwa Bildsegmentierung oder Verstärkungslernen, als lohnenswert erweisen könnten, sodass diesem Weg in zukünftigen Arbeiten weiter gefolgt werden kann. Was die ethischen Aspekte betrifft, so wurden keine Bedenken hinsichtlich des Missbrauchs der angefallenen Datensätze gefunden und eine Diskussion über die Nützlichkeit der Methodik zur Erleichterung problematischer Anwendungen angeregt. Alle Quelltexte und Datensätze wurden veröffentlicht [351, 352, 353, 354, 355, 356, 357].

# Danksagung

Zu guter Letzt möchte ich mich bei jenen Menschen bedanken, welche mich bei der Anfertigung meiner Dissertation unterstützt haben. Mein erstes Dankeswort gilt, post mortem, Herrn Rolf Würtz, dessen Wirken ich die Wahl des Themas und die Annahme als Doktorand verdanke. Sein herzliches Wesen, sein Denken und Handeln sowie die unzähligen Spieleabende bleiben unvergessen. Mein besonderer Dank gilt weiterhin Herrn Uwe Handmann und Herrn Tobias Glasmachers für die hervorragende Betreuung und die enorme Unterstützung bei der Durchführung und Umsetzung der gesamten Arbeit. Weiterhin möchte ich mich bei Frau Asja Fischer bedanken, welcher sich binnen kurzer Zeit als Zweitgutachter zur Verfügung gestellt hat. An dieser Stelle gilt ein weiteres Dankeswort Herrn Jörg Bornschein, welcher mich in den frühen Phasen meines Studiums sehr für dieses Thema begeistern konnte. In gleicher Weise gilt mein Dank Herrn Laurenz Wiskott, welche mich in interessanten Seminaren und Vorlesungen immer auf neue Gedanken brachte. Seinen Lehrveranstaltungen verdanke ich tiefgründige Einblicke in die Mathematik, welche selbstlernenden Systemen zugrunde liegt. Ebenso möchte ich den Kolleginnen und Kollegen der Hochschule Ruhr West Dank aussprechen, die mich auf meinem Weg mit Rat, Anregungen und produktiven Gesprächen begleitet haben. Für das Gegenlesen der Dissertationsschrift und die Korrekturvorschläge danke ich meiner Mutter, Kathi Großmann, meiner langjährigen Freundin, Isabelle Sandow, sowie Frau Stephanie Lakner. Des Weiteren danke ich meinen Eltern, meinen Freunden sowie meinen Brüdern und Schwestern für die Geduld, Ermutigungen und guten Gespräche während des Studiums und der Arbeit an dieser Dissertation. Zum Schluss gilt mein Dank Dir, unbekannte lesende Person, denn in gewisser Weise macht die ganze Arbeit erst dann wirklich Sinn, wenn Du sie auch gelesen hast.

# **Eidesstattliche Erklärung**

Ich versichere an Eides statt, dass ich die eingereichte Dissertation selbstständig und ohne unzulässige fremde Hilfe verfasst, andere als die in ihr angegebene Literatur nicht benutzt und dass ich alle ganz oder annähernd übernommenen Textstellen sowie verwendete Grafiken, Tabellen und Auswertungsprogramme kenntlich gemacht habe. Außerdem versichere ich, dass die vorgelegte elektronische mit der schriftlichen Version der Dissertation übereinstimmt und die Abhandlung in dieser oder ähnlicher Form noch nicht anderweitig als Promotionsleistung vorgelegt und bewertet wurde.

---

Bochum, 23.05.2022  
Ort, Datum

---

Nico Zengeler

# Literaturverzeichnis

- [1] J. von Neumann, “Eine Axiomatisierung der Mengenlehre.” *Journal für die reine und angewandte Mathematik*, vol. 1925, no. 154, pp. 219–240, 1925. [Online]. Available: <https://doi.org/10.1515/crll.1925.154.219>
- [2] K. Gödel, “Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I,” *Monatshefte für Mathematik und Physik*, vol. 38, no. 1, pp. 173–198, 1931.
- [3] A. M. Turing *et al.*, “On computable numbers, with an application to the Entscheidungsproblem,” *Journal of Math*, vol. 58, no. 345-363, p. 5, 1936.
- [4] A. M. Turing, “Intelligent machinery,” *National Physical Laboratory. Mathematics Division. Technical report.*, 1948.
- [5] N. Zengeler, T. Glasmachers, and U. Handmann, “Transfer Meta Learning,” in *2022 International Conference on Pattern Recognition*, 2022.
- [6] N. Abou Baker, N. Zengeler, and U. Handmann, “A Transfer Learning Evaluation of Deep Neural Networks for Image Classification,” *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 22–41, 2022. [Online]. Available: <https://www.mdpi.com/2504-4990/4/1/2>
- [7] N. Zengeler and U. Handmann, “Contracts for Difference: A Reinforcement Learning Approach,” *Journal of Risk and Financial Management*, vol. 13, no. 4, p. 78, 4 2020. [Online]. Available: <https://www.mdpi.com/1911-8074/13/4/78>
- [8] M. Wehrmann, N. Zengeler, and U. Handmann, “Observation Time Effects in Reinforcement Learning on Contracts for Difference,” *Journal of Risk and Financial Management*, vol. 14, no. 2, 2021. [Online]. Available: <https://www.mdpi.com/1911-8074/14/2/54>
- [9] N. Zengeler, U. Handmann, and T. Kopinski, “Hand Gesture Recognition in Automotive Human Machine Interaction,” *Sensors*, vol. 19, no. 1;59, pp. 1–28, 2019, mDPI, Basel, Switzerland. [Online]. Available: <https://www.mdpi.com/1424-8220/19/1/59>
- [10] N. Zengeler, M. Grimm, C. Borgmann, M. Jansen, S. Eimler, and U. Handmann, “An Evaluation of Human Detection Methods on Camera Images in Heavy Industry Environments,” in *2019 IEEE 14th Conference on Industrial Electronics and Applications (ICIEA)*, 2019.

- [11] N. Zengeler, A. Arntz, D. Keßler, M. Grimm, Z. Qasem, M. Jansen, S. Eimler, and U. Handmann, “Person Tracking in Heavy Industry Environments with Camera Images,” in *S-CUBE 2019 10th EAI International Conference on Sensor Systems and Software*, 2019.
- [12] N. Zengeler, M. Grimm, A. Arntz, D. Keßler, M. Jansen, S. Eimler, U. Handmann, K. Hermsen, and C. Grubert, “DamokleS 4.0 Interner Report,” *Hochschule Ruhr West*, 2019.
- [13] F. Wafo, I. Mabou, D. Heilmann, N. Zengeler, and U. Handmann, “An Evaluation of Machine Learning Frameworks,” in *IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*, 2021, pp. 1411–1416. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9516253>
- [14] N. Zengeler, “The Game of Tron as A Reinforcement Learning Environment,” in *Science Fiction vs. Science Facts*. Hochschule Ruhr West, Germany, 2020.
- [15] Hochschule Ruhr West, “DamokleS 4.0 - IKT für Cyber Physical Systems,” <https://www.damokles40.eu/>, zugegriffen: 14.04.2022.
- [16] K. Hermsen, S. Eimler, A. Arntz, D. Keßler, M. Jansen, Z. Qasem, M. Grimm, N. Zengeler, and U. Handmann, “Dynamic, Adaptive and Mobile System for Context-Based and Intelligent Support of Employees in the Steel Industry,” in *4th ESTAD (European Steel Technology and Application Days)*, Düsseldorf, Germany, 2019. [Online]. Available: [https://www.metec-estad\\_programmflyer\\_a5q\\_web-5.pdf](https://www.metec-estad2019.com/files/190619_metec-estad_programmflyer_a5q_web-5.pdf)
- [17] U. Handmann, M. Jansen, K. Hermsen, J. Bons, M. Grimm, K. Ohler-Martins, and M. Erdogan, “Damokles 4.0 - Dynamic, adaptive and mobile system for context-based and intelligent support of employees in the heavy industry,” in *Vgb Kongress 2017*. VGB PowerTech e. V., 2017. [Online]. Available: [https://www.vgb.org/kongress\\_2017\\_kurzfassungen\\_e2\\_2.html](https://www.vgb.org/kongress_2017_kurzfassungen_e2_2.html)
- [18] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, “Carbon emissions and large neural network training,” *arXiv preprint arXiv:2104.10350*, 2021.
- [19] D. Ha and J. Schmidhuber, “World models,” *arXiv preprint arXiv:1803.10122*, 2018.
- [20] P. F. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, “Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model,” *CoRR*, vol. abs/1610.03518, 2016. [Online]. Available: <http://arxiv.org/abs/1610.03518>
- [21] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, “DARLA: Improving Zero-Shot Transfer in

Reinforcement Learning,” in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: Pmlr, 06–11 Aug 2017, pp. 1480–1490. [Online]. Available: <http://proceedings.mlr.press/v70/higgins17a.html>

- [22] S. Barnett and S. J. Ceci, “When and where do we apply what we learn? A taxonomy for far transfer,” *Psychological Bulletin*, vol. 128, pp. 612–637, 07 2002.
- [23] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” *CoRR*, vol. abs/1612.00796, 2016. [Online]. Available: <http://arxiv.org/abs/1612.00796>
- [24] R. French, “Catastrophic Forgetting in Connectionist Networks,” *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.
- [25] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, “Overcoming Catastrophic Forgetting by Incremental Moment Matching,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4652–4662. [Online]. Available: <http://papers.nips.cc/paper/7051-overcoming-catastrophic-forgetting-by-incremental-moment-matching.pdf>
- [26] N. Abou Baker, P. Szabo-Müller, and U. Handmann, “Feature-fusion transfer learning method as a basis to support automated smartphone recycling in a circular smart city,” in *EAI S-CUBE 2020 - 11th EAI International Conference on Sensor Systems and Software*, online, 2020. [Online]. Available: <https://builder.eai.eu/share/52890>
- [27] I. Radosavovic, P. Dollár, R. B. Girshick, G. Gkioxari, and K. He, “Data Distillation: Towards Omni-Supervised Learning,” *CoRR*, vol. abs/1712.04440, 2017. [Online]. Available: <http://arxiv.org/abs/1712.04440>
- [28] U. C. Allard, C. L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin, “Deep Learning for Electromyographic Hand Gesture Signal Classification by Leveraging Transfer Learning,” *CoRR*, vol. abs/1801.07756, 2018. [Online]. Available: <http://arxiv.org/abs/1801.07756>
- [29] F. Aiolfi, “Transfer learning by kernel meta-learning,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 81–95.
- [30] P. Kordik and Cerny, Jan and Fryda, Tomas, “Discovering predictive ensembles for transfer learning and meta-learning,” *Machine learning*, vol. 107, no. 1, pp. 177–207, 2018.

- [31] Y. Guo, H. Niu, and S. Li, "Safety monitoring in construction site based on unmanned aerial vehicle platform with computer vision using transfer learning techniques," in *Proceedings of the 7th Asia-Pacific Workshop on Structural Health Monitoring, APWSHM 2018, 12-15 November 2018, Hong Kong SAR, China*, 2018.
- [32] J. Shen, X. Xiong, Y. Li, W. He, P. Li, and X. Zheng, "Detecting safety helmet wearing on construction sites with bounding-box regression and deep transfer learning," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 2, pp. 180–196, 2021.
- [33] L. Alzubaidi, M. A. Fadhel, O. Al-Shamma, J. Zhang, J. Santamaría, Y. Duan, and S. R Oleiwi, "Towards a better understanding of transfer learning for medical imaging: a case study," *Applied Sciences*, vol. 10, no. 13, p. 4523, 2020.
- [34] Y. Yuan, G. Zheng, K.-K. Wong, B. Ottersten, and Z.-Q. Luo, "Transfer learning and meta learning-based fast downlink beamforming adaptation," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 1742–1755, 2020.
- [35] S. T. Jose, O. Simeone, and G. Durisi, "Transfer meta-learning: Information-theoretic bounds and information meta-risk minimization," *IEEE Transactions on Information Theory*, 2021.
- [36] D. Choe, E. Choi, and D. K. Kim, "The Real-Time Mobile Application for Classifying of Endangered Parrot Species Using the CNN Models Based on Transfer Learning," *Mobile Information Systems*, vol. 2020, pp. 1–13, 2020.
- [37] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. Muller, "Transfer learning for time series classification," *CoRR*, vol. abs/1811.01533, Dec 2018. [Online]. Available: <http://arxiv.org/abs/1811.01533>
- [38] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," 2019. [Online]. Available: <http://arxiv.org/pdf/1902.00751v2>
- [39] J. W. Soh, S. Cho, and N. I. Cho, "Meta-Transfer Learning for Zero-Shot Super-Resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [40] Q. Sun, Y. Liu, Z. Chen, T.-S. Chua, and B. Schiele, "Meta-Transfer Learning through Hard Tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.
- [41] Y.-C. Chuang, T. Chen, Y. Yao, and D. S. H. Wong, "Transfer learning for efficient meta-modeling of process simulations," *Chemical Engineering Research and Design*, vol. 138, pp. 546–553, 2018.

- [42] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, “Meta-Transfer Learning for Few-Shot Learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [43] M. Ashouri and A. Hashemi, “A transfer learning metamodel using artificial neural networks applied to natural convection flows in enclosures,” 2020.
- [44] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big Data*, vol. 3, no. 1, p. 9, May 2016. [Online]. Available: <https://doi.org/10.1186/s40537-016-0043-6>
- [45] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.
- [46] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, “Sim-to-Real Robot Learning from Pixels with Progressive Nets,” *CoRR*, vol. abs/1610.04286, 2016. [Online]. Available: <http://arxiv.org/abs/1610.04286>
- [47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [48] P. Cunningham, M. Cord, and S. J. Delany, “Supervised learning,” in *Machine learning techniques for multimedia*. Springer, 2008, pp. 21–49.
- [49] S. Becker, “Unsupervised learning procedures for neural networks,” *International Journal of Neural Systems*, vol. 2, no. 01n02, pp. 17–33, 1991.
- [50] H. B. Barlow, “Unsupervised learning,” *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.
- [51] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv*, 2018.
- [52] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: <http://arxiv.org/abs/1506.02640>
- [53] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian Detection: An Evaluation of the State of the Art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, April 2012.
- [54] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer, “SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving.” in *CVPR Workshops*, 2017, pp. 446–454.
- [55] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, “Towards Reaching Human Performance in Pedestrian Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 973–986, April 2018.

- [56] S. Wu, S. Wang, R. Laganière, C. Liu, H. Wong, and Y. Xu, “Exploiting Target Data to Learn Deep Convolutional Networks for Scene-Adapted Human Detection,” *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1418–1432, March 2018.
- [57] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 161–168.
- [58] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, “A Closer Look at Few-shot Classification,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=HkxLXnAcFQ>
- [59] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.
- [60] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, “Adversarial autoencoders,” *arXiv preprint arXiv:1511.05644*, 2015.
- [61] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [62] J. An and S. Cho, “Variational autoencoder based anomaly detection using reconstruction probability,” *Special Lecture on IE*, vol. 2, no. 1, 2015.
- [63] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*, 1st ed. Cambridge, MA, USA: MIT press Cambridge, 1998, vol. 135.
- [64] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.
- [65] ———, “Efficient reinforcement learning through evolving neural network topologies,” in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, 2002, pp. 569–577.
- [66] M. S. Parsons, “Interpretation of machine-learning-based disruption models for plasma control,” *Plasma Physics and Controlled Fusion*, vol. 59, no. 8, p. 085001, 2017.
- [67] J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, “Predicting disruptive instabilities in controlled fusion plasmas through deep learning,” *Nature*, vol. 568, no. 7753, pp. 526–531, 2019.
- [68] M. Ghoreishi and A. Happonen, “Key enablers for deploying artificial intelligence for circular economy embracing sustainable product design: Three case studies,” in *AIP Conference Proceedings*, vol. 2233, no. 1. AIP Publishing LLC, 2020, p. 050008.

- [69] N. A. Baker, P. Szabo-Mýller, and U. Handmann, “Transfer learning-based method for automated e-waste recycling in smart cities,” *EAI Endorsed Transactions on Smart Cities*, vol. 5, no. 16, 4 2021.
- [70] A. Ramesh, C. Kambhampati, J. R. Monson, and P. Drew, “Artificial intelligence in medicine.” *Annals of the Royal College of Surgeons of England*, vol. 86, no. 5, p. 334, 2004.
- [71] P. Hamet and J. Tremblay, “Artificial intelligence in medicine,” *Metabolism*, vol. 69, pp. S36–s40, 2017, insights Into the Future of Medicine: Technologies, Concepts, and Integration. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S002604951730015X>
- [72] Y. Ding, J. H. Sohn, M. G. Kawczynski, H. Trivedi, R. Harnish, N. W. Jenkins, D. Lituiev, T. P. Copeland, M. S. Aboian, C. Mari Aparici *et al.*, “A deep learning model to predict a diagnosis of Alzheimer disease by using 18F-FDG PET of the brain,” *Radiology*, vol. 290, no. 2, pp. 456–464, 2019.
- [73] N. Goenka and S. Tiwari, “Deep learning for Alzheimer prediction using brain biomarkers,” *Artificial Intelligence Review*, vol. 54, no. 7, pp. 4827–4871, 2021.
- [74] W. Wang, J. Yang, J. Xiao, S. Li, and D. Zhou, “Face recognition based on deep learning,” in *International Conference on Human Centered Computing*. Springer, 2014, pp. 812–820.
- [75] L. Deng, G. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: An overview,” in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 8599–8603.
- [76] H. Buehler, L. Gonon, J. Teichmann, and B. Wood, “Deep hedging,” *Quantitative Finance*, vol. 19, no. 8, pp. 1271–1291, 2019.
- [77] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [78] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica *et al.*, “Exascale deep learning for climate analytics,” in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. Ieee, 2018, pp. 649–660.
- [79] S. Rasp, M. S. Pritchard, and P. Gentine, “Deep learning to represent subgrid processes in climate models,” *Proceedings of the National Academy of Sciences*, vol. 115, no. 39, pp. 9684–9689, 2018.

- [80] S. Scher, “Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning,” *Geophysical Research Letters*, vol. 45, no. 22, pp. 12–616, 2018.
- [81] S. Ardabili, A. Mosavi, M. Dehghani, and A. R. Várkonyi-Kóczy, “Deep learning and machine learning in hydrological processes climate change and earth systems a systematic review,” in *International conference on global research and education*. Springer, 2019, pp. 52–62.
- [82] J. Shi, S. Chen, Y. Lu, Y. Feng, R. Shi, Y. Yang, and J. Li, “An approach to cryptography based on continuous-variable quantum neural network,” *Scientific reports*, vol. 10, no. 1, pp. 1–13, 2020.
- [83] W. Kinzel and I. Kanter, “Neural cryptography,” in *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP’02.*, vol. 3. Ieee, 2002, pp. 1351–1354.
- [84] A. Klimov, A. Mityagin, and A. Shamir, “Analysis of neural cryptography,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2002, pp. 288–298.
- [85] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *arXiv preprint arXiv:1410.5401*, 2014.
- [86] A. Yadav and K. Pasupa, “Augmenting Differentiable Neural Computer with Read Network and Key-Value Memory,” in *2021 25th International Computer Science and Engineering Conference (ICSEC)*. Ieee, 2021, pp. 262–266.
- [87] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.
- [88] C. Darwin, *The origin of species*. PF Collier & son New York, 1909.
- [89] T. Bäck, D. Fogel, and Z. Michalewicz, “Introduction to evolutionary algorithms,” *Evolutionary computation*, vol. 1, pp. 59–63, 2000.
- [90] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, “Optimizing Deep Learning Hyper-Parameters through an Evolutionary Algorithm,” in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, ser. Mlhpc ’15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: <https://doi.org/10.1145/2834892.2834896>
- [91] P. Tabacof and E. Valle, “Exploring the space of adversarial images,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 426–433.

- [92] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. LI, “Adversarial attacks on deep learning models in natural language processing: A survey,” *arXiv preprint arXiv:1901.06796*, 2019.
- [93] J. H. Holland, “Genetic algorithms and the optimal allocation of trials,” *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.
- [94] D. Whitley, “A genetic algorithm tutorial,” *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.
- [95] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic programming*. Springer, 1998.
- [96] J. R. Koza and R. Poli, “Genetic programming,” in *Search methodologies*. Springer, 2005, pp. 127–164.
- [97] W. B. Langdon, “Genetic programming and data structures: genetic programming+ data structures= automatic programming!” *Univ. Birmingham, UK*, 1998.
- [98] H.-G. Beyer and H.-P. Schwefel, “Evolution strategies—A comprehensive introduction,” *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.
- [99] I. Rechenberg, “Evolution strategy: Nature’s way of optimization,” in *Optimization: Methods and applications, possibilities and limitations*. Springer, 1989, pp. 106–126.
- [100] J. Jägersküpper, “Rigorous runtime analysis of the (1+ 1) ES: 1/5-rule and ellipsoidal fitness landscapes,” in *International Workshop on Foundations of Genetic Algorithms*. Springer, 2005, pp. 260–281.
- [101] N. Hansen and A. Ostermeier, “Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation,” in *Proceedings of IEEE international conference on evolutionary computation*. Ieee, 1996, pp. 312–317.
- [102] H.-G. Beyer and B. Sendhoff, “Covariance matrix adaptation revisited—the CMSA evolution strategy—,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2008, pp. 123–132.
- [103] C. Igel, N. Hansen, and S. Roth, “Covariance matrix adaptation for multi-objective optimization,” *Evolutionary computation*, vol. 15, no. 1, pp. 1–28, 2007.
- [104] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.
- [105] M. M. Waldrop, “News Feature: What are the limits of deep learning?” *Proceedings of the National Academy of Sciences*, vol. 116, no. 4, pp. 1074–1077, 2019. [Online]. Available: <https://www.pnas.org/content/116/4/1074>

- [106] X. Ren, Z. Xing, X. Xia, D. Lo, X. Wang, and J. Grundy, “Neural Network-based Detection of Self-Admitted Technical Debt: From Performance to Explainability,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 3, pp. 1–45, 2019.
- [107] Z. Yang, A. Zhang, and A. Sudjianto, “Enhancing explainability of neural networks through architecture constraints,” *arXiv preprint arXiv:1901.03838*, 2019.
- [108] G. F. Elsayed, I. J. Goodfellow, and J. Sohl-Dickstein, “Adversarial Reprogramming of Neural Networks,” *CoRR*, vol. abs/1806.11146, 2018. [Online]. Available: <http://arxiv.org/abs/1806.11146>
- [109] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *CoRR*, vol. abs/1312.6199, 2013. [Online]. Available: <http://arxiv.org/abs/1312.6199>
- [110] P. Neekhara, S. Hussain, S. Dubnov, and F. Koushanfar, “Adversarial Reprogramming of Sequence Classification Neural Networks,” *CoRR*, vol. abs/1809.01829, 2018.
- [111] S. Ruder, “An overview of gradient descent optimization algorithms,” *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: <http://arxiv.org/abs/1609.04747>
- [112] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” 2012. [Online]. Available: [http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\\_slides\\_lec6.pdf](http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)
- [113] Z. Zhang, “Improved adam optimizer for deep neural networks,” in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. Ieee, 2018, pp. 1–2.
- [114] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: <http://arxiv.org/abs/1212.5701>
- [115] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
- [116] B. Recht, C. Re, S. Wright, and F. Niu, “Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent,” in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 693–701. [Online]. Available: <http://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf>
- [117] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

- [118] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, “On the Number of Linear Regions of Deep Neural Networks,” 2014. [Online]. Available: <http://arxiv.org/pdf/1402.1869v2>
- [119] K. Kawaguchi, J. Huang, and L. P. Kaelbling, “Effect of Depth and Width on Local Minima in Deep Learning,” *Neural Computation*, vol. 31, no. 7, pp. 1462–1498, Jul 2019. [Online]. Available: <http://arxiv.org/pdf/1811.08150v4>
- [120] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.
- [121] S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, apr 1998. [Online]. Available: <https://doi.org/10.1142/S0218488598000094>
- [122] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway Networks,” 2015. [Online]. Available: <http://arxiv.org/pdf/1505.00387v2>
- [123] J. Hu, L. Shen, and G. Sun, “Squeeze-and-Excitation Networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Ieee, 2018, pp. 7132–7141.
- [124] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: [https://www.researchgate.net/publication/13853244\\_Long\\_Short-term\\_Memory](https://www.researchgate.net/publication/13853244_Long_Short-term_Memory)
- [125] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
- [126] D. N. Perkins, G. Salomon *et al.*, “Transfer of learning,” *International encyclopedia of education*, vol. 2, pp. 6452–6457, 1992.
- [127] M. Kaboli, “A Review of Transfer Learning Algorithms,” Technische Universität München, Research Report, Aug. 2017, transfer Learning Algorithms. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01575126>
- [128] Z. Li and D. Hoiem, “Learning without Forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2017.
- [129] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010. [Online]. Available: <http://dx.doi.org/10.1109/TKDE.2009.191>
- [130] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A Survey on Deep Transfer Learning,” *CoRR*, vol. abs/1808.01974, 2018. [Online]. Available: <http://arxiv.org/abs/1808.01974>

- [131] Y. Wang and Q. Yao, “Few-shot Learning: A Survey,” *CoRR*, vol. abs/1904.05046, 2019. [Online]. Available: <http://arxiv.org/abs/1904.05046>
- [132] F. Zenke, B. Poole, and S. Ganguli, “Continual Learning Through Synaptic Intelligence,” in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: Pmlr, 06–11 Aug 2017, pp. 3987–3995. [Online]. Available: <http://proceedings.mlr.press/v70/zenke17a.html>
- [133] ———, “Improved multitask learning through synaptic intelligence,” *CoRR*, vol. abs/1703.04200, 2017. [Online]. Available: <http://arxiv.org/abs/1703.04200>
- [134] J. Vanschoren, “Meta-Learning: A Survey,” *CoRR*, vol. abs/1810.03548, 2018. [Online]. Available: <http://arxiv.org/abs/1810.03548>
- [135] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: <http://arxiv.org/abs/1503.02531>
- [136] R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng, “Zero-Shot Learning Through Cross-Modal Transfer,” 2013.
- [137] Y. Xian, B. Schiele, and Z. Akata, “Zero-Shot Learning – The Good, the Bad and the Ugly,” 2020.
- [138] C. H. Lampert, H. Nickisch, and S. Harmeling, “Attribute-Based Classification for Zero-Shot Visual Object Categorization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 453–465, 2014.
- [139] Z. Zhang and V. Saligrama, “Zero-Shot Learning via Semantic Similarity Embedding,” 2015.
- [140] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, “Label-Embedding for Image Classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, p. 1425–1438, Jul 2016. [Online]. Available: <http://dx.doi.org/10.1109/TPAMI.2015.2487986>
- [141] E. Bart and S. Ullman, “Cross-generalization: learning novel classes from a single example by feature replacement,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, 2005, pp. 672–679 vol. 1.
- [142] M. Fink, “Object Classification from a Single Example Utilizing Class Relevance Metrics,” in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005. [Online]. Available: <https://proceedings.neurips.cc/paper/2004/file/ef1e491a766ce3127556063d49bc2f98-Paper.pdf>

- [143] T. Tommasi and B. Caputo, “The More You Know, the Less You Learn: From Knowledge Transfer to One-shot Learning of Object Categories,” in *Bmvc*, 2009.
- [144] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell, “Multi-Content GAN for Few-Shot Font Style Transfer,” 2017.
- [145] B. Liu, X. Wang, M. Dixit, R. Kwitt, and N. Vasconcelos, “Feature Space Transfer for Data Augmentation,” 2019.
- [146] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei, “Label Efficient Learning of Transferable Representations across Domains and Tasks,” 2017.
- [147] V. J. Prakash and L. M. Nithya, “A Survey on Semi-Supervised Learning Techniques,” *CoRR*, vol. abs/1402.4645, 2014. [Online]. Available: <http://arxiv.org/abs/1402.4645>
- [148] X. Zhu, “Semi-Supervised Learning Literature Survey,” 2006.
- [149] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, “Deep Visual Analogy-Making,” in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1252–1260. [Online]. Available: <http://papers.nips.cc/paper/5845-deep-visual-analogy-making.pdf>
- [150] S. Ruder, “Neural Transfer Learning for Natural Language Processing,” Ph.D. dissertation, National University of Ireland, Galway, 2019.
- [151] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng, “Zero-shot learning through cross-modal transfer,” in *Advances in neural information processing systems*, 2013, pp. 935–943.
- [152] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, “Zero-shot learning by convex combination of semantic embeddings,” *arXiv preprint arXiv:1312.5650*, 2013.
- [153] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever, “Better language models and their implications,” *OpenAI Blog* <https://openai.com/blog/better-language-models>, 2019.
- [154] M. Gómez-Silva, E. Izquierdo, A. de la Escalera, and J. M. Armingol, “Transferring learning from multi-person tracking to person re-identification,” *Integrated Computer-Aided Engineering*, pp. 1–16, 04 2019.
- [155] H. Chen, Y. Wang, Y. Shi, K. Yan, M. Geng, Y. Tian, and T. Xiang, “Deep Transfer Learning for Person Re-Identification,” in *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*, 09 2018, pp. 1–5.

- [156] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive Neural Networks,” *CoRR*, vol. abs/1606.04671, 2016. [Online]. Available: <http://arxiv.org/abs/1606.04671>
- [157] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a Few Examples: A Survey on Few-Shot Learning,” *ACM Comput. Surv.*, vol. 53, no. 3, jun 2020. [Online]. Available: <https://doi.org/10.1145/3386252>
- [158] Z. N. K. Swati, Q. Zhao, M. Kabir, F. Ali, Z. Ali, S. Ahmed, and J. Lu, “Brain tumor classification for MR images using transfer learning and fine-tuning,” *Computerized Medical Imaging and Graphics*, vol. 75, pp. 34–46, 2019.
- [159] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, “Spottune: transfer learning through adaptive fine-tuning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4805–4814.
- [160] N. Brunel and J. P. Nadal, “Mutual information, Fisher information, and population coding.” *Neural Comput.*, vol. 10, no. 7, pp. 1731–1757, Oct 1998.
- [161] R. Pascanu and Y. Bengio, “Revisiting natural gradient for deep networks,” *arXiv preprint arXiv:1301.3584*, 2013.
- [162] K. Hambardzumyan, H. Khachatrian, and J. May, “Warp: Word-level adversarial reprogramming,” *arXiv preprint arXiv:2101.00121*, 2021.
- [163] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” *arXiv preprint arXiv:1804.07461*, 2018.
- [164] E. W. Xiang, B. Cao, D. H. Hu, and Q. Yang, “Bridging Domains Using World Wide Knowledge for Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 6, pp. 770–783, 2010.
- [165] R. Dale, “GPT-3: What’s it good for?” *Natural Language Engineering*, vol. 27, no. 1, pp. 113–118, 2021.
- [166] C. Payne, “MuseNet, 2019,” URL <https://openai.com/blog/musenet>, 2019.
- [167] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model. journal of machine learning research 3,” *Feb (2003)*, pp. 1137–1155, 2003.
- [168] F. Jelinek, “Interpolated estimation of Markov source parameters from sparse data,” in *Proc. Workshop on Pattern Recognition in Practice, 1980*, 1980.
- [169] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: <http://arxiv.org/abs/1706.03762>

- [170] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [171] C. M. Bishop, “Mixture density networks,” in *Unpublished technical report*. Birmingham: Aston University, 1994, Technical Report. [Online]. Available: <http://publications.aston.ac.uk/id/eprint/373/>
- [172] M. Van Otterlo and M. Wiering, “Reinforcement learning and markov decision processes,” in *Reinforcement Learning*. Springer, 2012, pp. 3–42.
- [173] J. E. Staddon and D. T. Cerutti, “Operant conditioning,” *Annual review of psychology*, vol. 54, no. 1, pp. 115–144, 2003.
- [174] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992. [Online]. Available: <http://dx.doi.org/10.1007/BF00992698>
- [175] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning,” *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: <http://arxiv.org/abs/1602.01783>
- [176] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wiersta, and M. A. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: <http://arxiv.org/abs/1312.5602>
- [177] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992. [Online]. Available: <http://dx.doi.org/10.1007/BF00992696>
- [178] A. Nowé, P. Vrancx, and Y.-M. D. Hauwere, “Game theory and multi-agent reinforcement learning,” in *Reinforcement Learning*. Springer, 2012, pp. 441–470.
- [179] A. Rajeswaran, I. Mordatch, and V. Kumar, “A game theoretic framework for model based reinforcement learning,” in *International conference on machine learning*. Pmlr, 2020, pp. 7953–7963.
- [180] A. Newell, “The Chess Machine: An Example of Dealing with a Complex Task by Adaptation,” in *Proceedings of the March 1-3, 1955, Western Joint Computer Conference*, ser. AFIPS ’55 (Western). New York, NY, USA: Acm, 1955, pp. 101–108. [Online]. Available: <http://doi.acm.org/10.1145/1455292.1455312>
- [181] K. Shao, D. Zhao, N. Li, and Y. Zhu, “Learning battles in vizdoom via deep reinforcement learning,” in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. Ieee, 2018, pp. 1–4.
- [182] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, “Vizdoom: A doom-based ai research platform for visual reinforcement learning,” in *2016 IEEE conference on computational intelligence and games (CIG)*, Ieee. Santorini, Greece: Ieee, Sep 2016, pp. 1–8, the best paper award. [Online]. Available: <http://arxiv.org/abs/1605.02097>

- [183] G. Lample and D. S. Chaplot, “Playing FPS Games with Deep Reinforcement Learning,” *CoRR*, vol. abs/1609.05521, 2016. [Online]. Available: <http://arxiv.org/abs/1609.05521>
- [184] M. Smith, S. Lee-Urban, and H. Muñoz-Avila, “RETALIATE: Learning winning policies in first-person shooter games,” in *Aaaai*, 2007, pp. 1801–1806.
- [185] D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. E. Schapire, “Exploratory Gradient Boosting for Reinforcement Learning in Complex Domains,” *CoRR*, vol. abs/1603.04119, 2016. [Online]. Available: <http://arxiv.org/abs/1603.04119>
- [186] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [187] X. Wang, J. Song, P. Qi, P. Peng, Z. Tang, W. Zhang, W. Li, X. Pi, J. He, C. Gao *et al.*, “SCC: an efficient deep reinforcement learning agent mastering the game of StarCraft II,” in *International Conference on Machine Learning*. Pmlr, 2021, pp. 10 905–10 915.
- [188] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, “StarCraft II: A New Challenge for Reinforcement Learning,” *arXiv preprint arXiv:1708.04782*, Aug. 2017. [Online]. Available: <https://arxiv.org/abs/1708.04782v1>
- [189] K. Arulkumaran, A. Cully, and J. Togelius, “Alphastar: An evolutionary computation perspective,” in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 314–315.
- [190] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver, “AlphaStar: Mastering the Real-Time Strategy Game StarCraft II,” <https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/>, 2019.
- [191] K. Zhang and W. Pan, “The two facets of the exploration-exploitation dilemma,” in *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. Ieee, 2006, pp. 371–380.

- [192] M. Tokic and G. Palm, “Value-difference based exploration: adaptive control between epsilon-greedy and softmax,” in *Annual conference on artificial intelligence*. Springer, 2011, pp. 335–346.
- [193] A. Masadeh, Z. Wang, and A. E. Kamal, “Reinforcement learning exploration algorithms for energy harvesting communications systems,” in *2018 IEEE International Conference on Communications (ICC)*. Ieee, 2018, pp. 1–6.
- [194] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.
- [195] G. Tesauro, “Temporal Difference Learning and TD-Gammon,” *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995. [Online]. Available: <http://doi.acm.org/10.1145/203330.203343>
- [196] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, no. 1, pp. 99–134, 1998. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S000437029800023X>
- [197] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized Experience Replay,” *CoRR*, vol. abs/1511.05952, 2015. [Online]. Available: <http://arxiv.org/abs/1511.05952>
- [198] P. Wegner, “Concepts and paradigms of object-oriented programming,” *ACM Sigplan OOPS Messenger*, vol. 1, no. 1, pp. 7–87, 1990.
- [199] H. Balzert, *Lehrbuch Grundlagen der Informatik: Konzepte und Notationen in UML 2, Java 5, C++ und C#; Algorithmik und Software-Technik; Anwendungen*. Springer, 2005.
- [200] K. Venkataraman, “Automated versus floor trading: An analysis of execution costs on the Paris and New York exchanges,” *Journal of Finance*, vol. 56, no. 4, pp. 1445–1485, 8 2001. [Online]. Available: <https://onlinelibrary.wiley.com/doi/full/10.1111/0022-1082.00375><https://onlinelibrary.wiley.com/doi/abs/10.1111/0022-1082.00375><https://onlinelibrary.wiley.com/doi/10.1111/0022-1082.00375>
- [201] P. Aghion, B. Jones, and C. Jones, “Artificial intelligence and economic growth,” National Bureau of Economic Research, Tech. Rep., 2017.
- [202] A. Golub, J. Glattfelder, and R. B. Olsen, “The Alpha Engine: Designing an Automated Trading Algorithm,” *SSRN Electronic Journal*, 4 2018. [Online]. Available: <https://papers.ssrn.com/abstract=2951348>
- [203] G. Jeong and H. Y. Kim, “Improving financial trading decisions using deep Q-learning: Predicting the number of Shares, action Strategies, and transfer learning,” *Expert Systems with Applications*, vol. 117, pp. 125–138, 3 2019.

- [204] M. Kearns and L. Ortiz, “The Penn-Lehman Automated Trading Project,” pp. 22–31, 11 2003.
- [205] J. B. Chakole, M. S. Kolhe, G. D. Mahapurush, A. Yadav, and M. P. Kurhekhar, “A Q-learning agent for automated trading in equity stock markets,” *Expert Systems with Applications*, vol. 163, p. 113761, 1 2021.
- [206] J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.
- [207] M. Vargas, B. D. Lima, and A. Evsukoff, “Deep learning for stock market prediction from financial news articles,” in *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. Ieee, 2017, pp. 60–65.
- [208] X. Ding, Y. Zhang, T. Liu, and J. Duan, “Deep learning for event-driven stock prediction,” in *Twenty-fourth international joint conference on artificial intelligence*, 2015.
- [209] K. Chen, Y. Zhou, and F. Dai, “A LSTM-based method for stock returns prediction: A case study of China stock market,” in *2015 IEEE International Conference on Big Data (Big Data)*. Ieee, 2015, pp. 2823–2824.
- [210] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, “Deep learning for stock prediction using numerical and textual information,” in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. Ieee, 2016, pp. 1–6.
- [211] J. Brogaard *et al.*, “High frequency trading and its impact on market quality,” *Northwestern University Kellogg School of Management Working Paper*, vol. 66, 2010.
- [212] I. Aldridge, *High-frequency trading: a practical guide to algorithmic strategies and trading systems*. John Wiley & Sons, 2013, vol. 604.
- [213] J. E. Moody and M. Saffell, “Reinforcement learning for trading,” in *Advances in Neural Information Processing Systems*, 1999, pp. 917–923.
- [214] M. A. H. Dempster and Y. S. Romahi, “Intraday FX trading: An evolutionary reinforcement learning approach,” in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2002, pp. 347–358.
- [215] C. Gold, “FX trading via recurrent reinforcement learning,” in *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, 2003, pp. 363–370.
- [216] Y.-S. Lim and D. Gorse, “Reinforcement Learning for High-Frequency Market Making.” in *Esann*, 2018.

- [217] H. Kaiming, Z. Xiangyu, R. Shaoqing, and S. Jian, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” *CoRR*, vol. abs/1502.01852, 2015. [Online]. Available: <http://arxiv.org/abs/1502.01852>
- [218] C. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
- [219] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” *arXiv preprint arXiv:1511.05952*, 2015.
- [220] Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: <http://arxiv.org/abs/1605.02688>
- [221] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri *et al.*, “Lasagne: First release.” Aug. 2015. [Online]. Available: <http://dx.doi.org/10.5281/zenodo.27878>
- [222] J. Suarez and R. R. Murphy, “Hand gesture recognition with depth images: A review,” in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, Sept 2012, pp. 411–417.
- [223] S. Yang, P. Premaratne, and P. Vial, “Hand gesture recognition: An overview,” in *2013 5th IEEE International Conference on Broadband Network Multimedia Technology*, Nov 2013, pp. 63–69.
- [224] F. F. M. Ma’asum, S. Sulaiman, and A. Saparon, “An Overview of Hand Gestures Recognition System Techniques,” *IOP Conference Series: Materials Science and Engineering*, vol. 99, no. 1, p. 012012, 2015. [Online]. Available: <http://stacks.iop.org/1757-899X/99/i=1/a=012012>
- [225] R. Z. Khan and N. A. Ibraheem, “Hand gesture recognition: a literature review,” *International journal of artificial Intelligence & Applications*, vol. 3, no. 4, p. 161, 2012.
- [226] J. Suarez and R. R. Murphy, “Hand gesture recognition with depth images: A review,” in *Ro-man, 2012 Ieee*. Ieee, 2012, pp. 411–417.
- [227] Z. Ren, J. Meng, and J. Yuan, “Depth camera based hand gesture recognition and its applications in human-computer-interaction,” in *Information, Communications and Signal Processing (ICICS) 2011 8th International Conference on*. Ieee, 2011, pp. 1–5.
- [228] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, “Vision-based hand-gesture applications,” *Communications of the ACM*, vol. 54, no. 2, pp. 60–71, 2011.

- [229] T. Kopinski, F. Sachara, A. Gepperth, and U. Handmann, “Free-hand gesture recognition with 3D-CNNs for in-car infotainment control in real-time,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 959–964.
- [230] T. Kopinski, S. Geisler, and U. Handmann, “Gesture-based human-machine interaction for assistance systems,” in *2015 IEEE International Conference on Information and Automation*, Aug 2015, pp. 510–517.
- [231] T. Kopinski, J. Eberwein, S. Geisler, and U. Handmann, “Touch versus mid-air gesture interfaces in road scenarios - measuring driver performance degradation,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2016, pp. 661–666.
- [232] M. G. Jacob and J. P. Wachs, “Context-based hand gesture recognition for the operating room,” *Pattern Recognition Letters*, vol. 36, pp. 196–203, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167865513002225>
- [233] P. Student, “Automatische Ermittlung von Parametern von Scandaten auf Basis von tiefen neuronalen Netzen,” Bachelorarbeit, Hochschule Ruhr West, 2021.
- [234] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li, “Deep learning on 3D point clouds,” *Remote Sensing*, vol. 12, no. 11, p. 1729, 2020.
- [235] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, “Deep learning for 3d point clouds: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [236] W. Wohlkinger and M. Vincze, “Ensemble of shape functions for 3D object classification,” in *2011 IEEE International Conference on Robotics and Biomimetics*, Dec 2011, pp. 2987–2992.
- [237] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning point cloud views using persistent feature histograms,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2008, pp. 3384–3391.
- [238] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, “Fast 3D recognition and pose using the Viewpoint Feature Histogram,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 2155–2162.
- [239] E. Kollarz, J. Penne, J. Hornegger, and A. Barke, “Gesture recognition with a time-of-flight camera,” *International Journal of Intelligent Systems Technologies and Applications*, vol. 5, no. 3, pp. 334–343, 2008.
- [240] M.-H. Yang, N. Ahuja, and M. Tabb, “Extraction of 2d motion trajectories and its application to hand gesture recognition,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 8, pp. 1061–1074, 2002.

- [241] A. Ramamoorthy, N. Vaswani, S. Chaudhury, and S. Banerjee, “Recognition of dynamic hand gestures,” *Pattern Recognition*, vol. 36, no. 9, pp. 2069–2081, 2003.
- [242] T. Kapuściński, M. Oszust, and M. Wysocki, “Hand Gesture Recognition Using Time-of-Flight Camera and Viewpoint Feature Histogram,” in *Intelligent Systems in Technical and Medical Diagnostics*, J. Korbicz and M. Kowal, Eds. Berlin, Heidelberg: Springer, 2014, pp. 403–414.
- [243] Y. Wen, C. Hu, G. Yu, and C. Wang, “A robust method of detecting hand gestures using depth sensors,” in *Haptic Audio Visual Environments and Games (HAVE), 2012 IEEE International Workshop on*. Ieee, 2012, pp. 72–77.
- [244] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, “A unified framework for gesture recognition and spatiotemporal gesture segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 9, pp. 1685–1699, 2009.
- [245] T.-K. Kim and R. Cipolla, “Canonical Correlation Analysis of Video Volume Tensors for Action Categorization and Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1415–1428, Aug. 2009. [Online]. Available: <http://dx.doi.org/10.1109/TPAMI.2008.167>
- [246] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz, “Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4207–4215.
- [247] Y. Zhang, C. Cao, J. Cheng, and H. Lu, “EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition,” *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1038–1050, May 2018.
- [248] T. Kopinski, A. Gepperth, and U. Handmann, “A time-of-flight-based hand posture database for human-machine interaction,” in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Nov 2016, pp. 1–6.
- [249] J. Wan, S. Z. Li, Y. Zhao, S. Zhou, I. Guyon, and S. Escalera, “ChaLearn Looking at People RGB-D Isolated and Continuous Datasets for Gesture Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2016, pp. 761–769.
- [250] A. Kurakin, Z. Zhang, and Z. Liu, “A real time system for dynamic hand gesture recognition with a depth sensor,” in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 1975–1979.
- [251] S. Oprisescu, C. Rasche, and B. Su, “Automatic static hand gesture recognition using ToF cameras,” in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*. Ieee, 2012, pp. 2748–2751.

- [252] T. Kopinski, A. Gepperth, and U. Handmann, “A Real-Time Applicable Dynamic Hand Gesture Recognition Framework,” in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sept 2015, pp. 2358–2363.
- [253] V. John, A. Boyali, S. Mita, M. Imanishi, and N. Sanma, “Deep Learning-Based Fast Hand Gesture Recognition Using Representative Frames,” in *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov 2016, pp. 1–8.
- [254] P. Narayana, J. R. Beveridge, and A. Draper, “Gesture Recognition : Focus on the Hands,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.
- [255] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” in *Cvpr*, 2017.
- [256] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, June 2005, pp. 886–893 vol. 1.
- [257] B. V. S. Girish, “Method and apparatus for human detection in images,” 2019, uS Patent 10,255,490.
- [258] J. Redmon, “Darknet: Open Source Neural Networks in C,” <http://pjreddie.com/darknet/>, 2013–2016.
- [259] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand Keypoint Detection in Single Images using Multiview Bootstrapping,” in *Cvpr*, 2017.
- [260] T. Surasak, I. Takahiro, C. Cheng, C. Wang, and P. Sheng, “Histogram of oriented gradients for human detection in video,” in *2018 5th International Conference on Business and Industrial Research (ICBIR)*, May 2018, pp. 172–176.
- [261] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” in *Cvpr*, 2016.
- [262] S. Zhang, J. Yang, and B. Schiele, “Occluded Pedestrian Detection Through Guided Attention in CNNs,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [263] S. Hommel, M. Grimm, D. Malysiak, and U. Handmann, “APFel - fast multi camera people tracking at airports, based on decentralized video indexing,” *Science<sup>2</sup> - Safety and Security*, vol. 2, pp. 48–55, 2014, hOMELAND SECURITY UG, Hemer, Germany.
- [264] Z. Qasem, J. Bons, C. Borgmann, S. Eimler, and M. Jansen, “Dynamic, Adaptive, and Mobile System for Context-Based and Intelligent Support of Employees in Heavy Industry,” in *2018 Sixth International Conference on Enterprise Systems (ES)*, 10 2018.

- [265] P. Z. Peebles, *Probability, random variables, and random signal principles*. McGraw-Hill New York, NY, USA:, 2001, vol. 3.
- [266] “GNU Image Manipulation Program Documentation,” <https://docs.gimp.org/2.10/en/plug-in-lighting.html>, zugegriffen: 09.11.2018.
- [267] H. Bristow and S. Lucey, “Why do linear SVMs trained on HOG features perform so well?” *CoRR*, vol. abs/1406.2419, 2014. [Online]. Available: <http://arxiv.org/abs/1406.2419>
- [268] V. Prisacariu and I. Reid, “fastHOG - a real-time GPU implementation of HOG,” Department of Engineering Science, Tech. Rep. 2310/09, 2009.
- [269] C. Tomasi, “Histograms of oriented gradients,” *Computer Vision Sampler*, pp. 1–6, 2012.
- [270] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, “Fast Human Detection Using a Cascade of Histograms of Oriented Gradients,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 2, June 2006, pp. 1491–1498.
- [271] Itseez, “Open Source Computer Vision Library,” <https://github.com/itseez/opencv>, 2015.
- [272] *The OpenCV Reference Manual*, 2nd ed., Itseez, April 2014.
- [273] J. Sang, Z. Wu, P. Guo, H. Hu, H. Xiang, Q. Zhang, and B. Cai, “An improved YOLOv2 for vehicle detection,” *Sensors*, vol. 18, no. 12, p. 4272, 2018.
- [274] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [275] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A Review of Yolo Algorithm Developments,” *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.
- [276] S. Aubry, S. Laraba, J. Tilmanne, and T. Dutoit, “Action recognition based on 2D skeletons extracted from RGB videos,” in *MATEC Web of Conferences*, vol. 277. EDP Sciences, 2019, p. 02034.
- [277] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.
- [278] E. Volkova, S. De La Rosa, H. H. Bülthoff, and B. Mohler, “The MPI emotional body expressions database for narrative scenarios,” *PloS one*, vol. 9, no. 12, p. e113647, 2014.

- [279] O. Surinta and S. Khruahong, “Tracking People and Objects with an Autonomous Unmanned Aerial Vehicle using Face and Color Detection,” in *2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON)*, 01 2019.
- [280] P. Peng, T. Xiang, Y. Wang, M. Pontil, S. Gong, T. Huang, and Y. Tian, “Unsupervised cross-dataset transfer learning for person re-identification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1306–1315.
- [281] Y.-J. Li, F.-E. Yang, Y.-C. Liu, Y.-Y. Yeh, X. Du, and Y.-C. Frank Wang, “Adaptation and re-identification network: An unsupervised deep transfer learning approach to person re-identification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 172–178.
- [282] S. M. Ahmed, A. R. Lejbolle, R. Panda, and A. K. Roy-Chowdhury, “Camera on-boarding for person re-identification using hypothesis transfer learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12144–12153.
- [283] T. Hao, Q. Wang, D. Wu, and J. Sun, “Multiple person tracking based on slow feature analysis,” *Multimedia Tools and Applications*, vol. 77, 09 2017.
- [284] L. Wiskott and T. J. Sejnowski, “Slow Feature Analysis: Unsupervised Learning of Invariances,” *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002. [Online]. Available: <https://doi.org/10.1162/089976602317318938>
- [285] S. Amin and J. Burke, “OpenMoves: A System for Interpreting Person-Tracking Data,” in *Proceedings of the 5th International Conference on Movement and Computing*, 06 2018, pp. 1–4.
- [286] A. Arntz, D. Keßler, N. Borgert, N. Zengeler, M. Jansen, U. Handmann, and S. Eimler, “Navigating a Heavy Industry Environment Using Augmented Reality - A Comparison of Two Indoor Navigation Designs,” in *Virtual, Augmented and Mixed Reality - Industrial and Everyday Life Applications, 12th International Conference, VAMR 2020 - Held as Part of the 22nd HCI International Conference, HCII 2020*, Springer Nature, Cham, Switzerland, 2020. [Online]. Available: [https://link.springer.com/chapter/10.1007/978-3-030-49698-2\\_1](https://link.springer.com/chapter/10.1007/978-3-030-49698-2_1)
- [287] A. Bunte, A. Fischbach, J. Strohschein, T. Bartz-Beielstein, H. Faeskorn-Woyke, and O. Niggemann, “Evaluation of Cognitive Architectures for Cyber-Physical Production Systems,” *CoRR*, vol. abs/1902.08448, 2019. [Online]. Available: <http://arxiv.org/abs/1902.08448>
- [288] M. Nouiri, D. Trentesaux, and A. Bekrar, “EasySched: a multi-agent architecture for the predictive and reactive scheduling of Industry 4.0 production systems based

- on the available renewable energy,” *CoRR*, vol. abs/1905.12083, 2019. [Online]. Available: <http://arxiv.org/abs/1905.12083>
- [289] W. Hasselbring, S. Henning, B. Latte, A. Möbius, T. Richter, S. Schalk, and M. Wojcieszak, “Industrial DevOps,” *CoRR*, vol. abs/1907.01875, 2019. [Online]. Available: <http://arxiv.org/abs/1907.01875>
- [290] A. K. Dey, “Understanding and Using Context,” *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Jan. 2001. [Online]. Available: <http://dx.doi.org/10.1007/s007790170019>
- [291] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. T. H. Romeny, and J. B. Zimmerman, “Adaptive Histogram Equalization and Its Variations,” *Comput. Vision Graph. Image Process.*, vol. 39, no. 3, pp. 355–368, Sep. 1987. [Online]. Available: [http://dx.doi.org/10.1016/S0734-189X\(87\)80186-X](http://dx.doi.org/10.1016/S0734-189X(87)80186-X)
- [292] MathWorks, “Camera Calibration,” <https://mathworks.com/>, 2017.
- [293] J. Heikkila and O. Silven, “A Four-step Camera Calibration Procedure with Implicit Image Correction,” in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, ser. Cvpr '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 1106–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=794189.794489>
- [294] G. Welch, G. Bishop *et al.*, “An introduction to the Kalman filter,” 1995.
- [295] A. S. Lundervold and A. Lundervold, “An overview of deep learning in medical imaging focusing on MRI,” *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019, special Issue: Deep Learning in Medical Physics. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0939388918301181>
- [296] R. Pires de Lima and K. Marfurt, “Convolutional Neural Network for Remote-Sensing Scene Classification: Transfer Learning Analysis,” *Remote Sensing*, vol. 12, no. 1, p. 86, 2020.
- [297] M. Zou and Y. Zhong, “Transfer Learning for Classification of Optical Satellite Image,” *Sensing and Imaging*, vol. 19, no. 1, 2018.
- [298] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>

- [299] A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” 2017.
- [300] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark Analysis of Representative Deep Neural Network Architectures,” *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018. [Online]. Available: <http://dx.doi.org/10.1109/ACCESS.2018.2877890>
- [301] R. Zaheer and H. Shaziya, “A Study of the Optimization Algorithms in Deep Learning,” in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, 2019, pp. 536–539.
- [302] O. Kaziha and T. Bonny, “A Comparison of Quantized Convolutional and LSTM Recurrent Neural Network Models Using MNIST,” in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. Ieee, 11/19/2019 - 11/21/2019, pp. 1–5.
- [303] S. Chilamkurthy, “Transfer Learning for Computer Vision Tutorial,” [https://pytorch.org/tutorials/beginner/transfer\\_learning\\_tutorial.html](https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html), zugriff am: 14.02.2022.
- [304] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [305] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, Eds., *ImageNet Classification with Deep Convolutional Neural Networks*, 2012.
- [306] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” [Online]. Available: <http://arxiv.org/pdf/1409.1556v6>
- [307] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 6/7/2015 - 6/12/2015, pp. 1–9.
- [308] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [309] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2015. [Online]. Available: <http://arxiv.org/pdf/1512.03385v1>
- [310] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,” 2016. [Online]. Available: <http://arxiv.org/pdf/1602.07360v4>

- [311] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 7/21/2017 - 7/26/2017, pp. 2261–2269.
- [312] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, “Aggregated Residual Transformations for Deep Neural Networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 7/21/2017 - 7/26/2017, pp. 5987–5995.
- [313] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017. [Online]. Available: <http://arxiv.org/pdf/1704.04861v1>
- [314] S. Zagoruyko and N. Komodakis, “Wide Residual Networks,” 2017. [Online]. Available: <http://arxiv.org/pdf/1605.07146v4>
- [315] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices,” 2017. [Online]. Available: <http://arxiv.org/pdf/1707.01083v2>
- [316] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “MnasNet: Platform-Aware Neural Architecture Search for Mobile,” 2019.
- [317] H. Latapie, O. Kilic, G. Liu, Y. Yan, R. Kompella, P. Wang, K. R. Thorisson, A. Lawrence, Y. Sun, and J. Srinivasa, “A Metamodel and Framework for Artificial General Intelligence From Theory to Practice.” [Online]. Available: <http://arxiv.org/pdf/2102.06112v1>
- [318] D. J. Skinner and R. Maulik, “Meta-modeling strategy for data-driven forecasting.” [Online]. Available: <http://arxiv.org/pdf/2012.00678v1>
- [319] S.-J. Park, S. Han, J.-W. Baek, I. Kim, J. Song, H. B. Lee, J.-J. Han, and S. J. Hwang, “Meta Variance Transfer: Learning to Augment from the Others,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. Pmlr, 13–18 Jul 2020, pp. 7510–7520. [Online]. Available: <https://proceedings.mlr.press/v119/park20b.html>
- [320] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, and N. Houlsby, “The Visual Task Adaptation Benchmark,” *CoRR*, vol. abs/1910.04867, 2019. [Online]. Available: <http://arxiv.org/abs/1910.04867>
- [321] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” *CoRR*, vol. abs/1708.07747, 2017. [Online]. Available: <http://arxiv.org/abs/1708.07747>

- [322] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million Image Database for Scene Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.
- [323] A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks,” *CoRR*, vol. abs/1404.5997, 2014. [Online]. Available: <http://arxiv.org/abs/1404.5997>
- [324] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: <http://arxiv.org/abs/1409.4842>
- [325] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. Le V, “MnasNet: Platform-Aware Neural Architecture Search for Mobile,” *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. [Online]. Available: <http://arxiv.org/pdf/1807.11626v3.pdf>
- [326] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Iclr 2015*, 2015.
- [327] M. Türkoglu, “Modellbasiertes Verstärkungslernen,” Bachelorarbeit, Hochschule Ruhr West, 2021.
- [328] M. Wehrmann, “Einfluss des Belohnungssystems auf das Verhalten von Verstärkungslernen,” Masterarbeit, Hochschule Ruhr West, 2021.
- [329] A. Schweer, “Untersuchung der Anwendbarkeit von bestärkendem Lernen auf Differenzkontraktionshandel,” Bachelorarbeit, Hochschule Ruhr West, 2020.
- [330] R. Allen and D. Masters, “Artificial Intelligence: the right to protection from discrimination caused by algorithms, machine learning and automated decision-making,” in *ERA Forum*, vol. 20, no. 4. Springer, 2020, pp. 585–598.
- [331] N. Bostrom and E. Yudkowsky, “The ethics of artificial intelligence,” *The Cambridge handbook of artificial intelligence*, vol. 1, pp. 316–334, 2014.
- [332] S. Ghosh, N. Das, I. Das, and U. Maulik, “Understanding deep learning techniques for image segmentation,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–35, 2019.
- [333] A. Raj, Y. Bresler, and B. Li, “Improving robustness of deep-learning-based image reconstruction,” in *International Conference on Machine Learning*. Pmlr, 2020, pp. 7932–7942.
- [334] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, and S. Bauer, “Causalworld: A robotic manipulation benchmark for causal structure and transfer learning,” *arXiv preprint arXiv:2010.04296*, 2020.

- [335] M. Boulares, T. Alafif, and A. Barnawi, “Transfer learning benchmark for cardiovascular disease recognition,” *IEEE Access*, vol. 8, pp. 109 475–109 491, 2020.
- [336] H. Hoijer, “The Sapir-Whorf Hypothesis,” *Language in culture*, pp. 92–105, 1954.
- [337] G. Pleiss, M. Raghavan, F. Wu, J. Kleinberg, and K. Q. Weinberger, “On fairness and calibration,” *Advances in neural information processing systems*, vol. 30, 2017.
- [338] F. Calmon, D. Wei, B. Vinzamuri, K. Natesan Ramamurthy, and K. R. Varshney, “Optimized pre-processing for discrimination prevention,” *Advances in neural information processing systems*, vol. 30, 2017.
- [339] L. E. Celis, L. Huang, V. Keswani, and N. K. Vishnoi, “Classification with fairness constraints: A meta-algorithm with provable guarantees,” in *Proceedings of the conference on fairness, accountability, and transparency*, 2019, pp. 319–328.
- [340] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, “Certifying and removing disparate impact,” in *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 259–268.
- [341] P. Awasthi, M. Kleindessner, and J. Morgenstern, “Equalized odds postprocessing under imperfect group information,” in *International Conference on Artificial Intelligence and Statistics*. Pmlr, 2020, pp. 1770–1780.
- [342] F. Kamiran, A. Karim, and X. Zhang, “Decision theory for discrimination-aware classification,” in *2012 IEEE 12th International Conference on Data Mining*. Ieee, 2012, pp. 924–929.
- [343] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, “Fairness-aware classifier with prejudice remover regularizer,” in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2012, pp. 35–50.
- [344] R. Trifonov, O. Nakov, and V. Mladenov, “Artificial Intelligence in Cyber Threats Intelligence,” in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, 2018, pp. 1–4.
- [345] B. Hallaq, T. Somer, A.-M. Osula, K. Ngo, and T. Mitchener-Nissen, “Artificial intelligence within the military domain and cyber warfare,” in *Eur. Conf. Inf. Warf. Secur. ECCWS*, 2017, pp. 153–157.
- [346] S. Russell, S. Hauert, R. Altman, and M. Veloso, “Ethics of artificial intelligence,” *Nature*, vol. 521, no. 7553, pp. 415–416, 2015.
- [347] T. C. King, N. Aggarwal, M. Taddeo, and L. Floridi, “Artificial Intelligence Crime: An Interdisciplinary Analysis of Foreseeable Threats and Solutions,” *Science and Engineering Ethics*, vol. 26, 2020. [Online]. Available: <https://doi.org/10.1007/s11948-018-00081-0>

- [348] R. Rathi, “Effect of Cambridge Analytica’s Facebook ads on the 2016 us presidential election,” *Towards Data Science*, 2019.
- [349] D. Kreiss and S. C. McGregor, “The “arbiters of what our voters see”: Facebook and Google’s struggle with policy, process, and enforcement around political advertising,” *Political Communication*, vol. 36, no. 4, pp. 499–522, 2019.
- [350] P. Čerka, J. Grigienė, and G. Sirbikytė, “Liability for damages caused by artificial intelligence,” *Computer Law & Security Review*, vol. 31, no. 3, pp. 376–389, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S026736491500062X>
- [351] N. Zengeler, “Quelltext und Datensätze zu automatisierten CfD Handelssystemen,” [https://gitlab.hs-ruhrwest.de/nico.zengeler/cfd\\_arla](https://gitlab.hs-ruhrwest.de/nico.zengeler/cfd_arla), 2020, zugegriffen: 23.05.2022.
- [352] ——, “Quelltext und Datensätze zur Handgesenerkennung,” [https://gitlab.hs-ruhrwest.de/nico.zengeler/hgrahmi\\_source/](https://gitlab.hs-ruhrwest.de/nico.zengeler/hgrahmi_source/), 2019, zugegriffen: 23.05.2022.
- [353] ——, “Quelltext und Datensätze zur Personenerkennung,” <https://gitlab.hs-ruhrwest.de/nico.zengeler/human-detection>, 2018, zugegriffen: 23.05.2022.
- [354] ——, “Quelltext und Datensätze zur Personverfolgung,” <https://gitlab.hs-ruhrwest.de/nico.zengeler/detectionprocessing>, 2018, zugegriffen: 23.05.2022.
- [355] ——, “Quelltext und Datensätze zum Bildklassifikationstransfer,” <https://gitlab.hs-ruhrwest.de/nico.zengeler/midas>, 2021, zugegriffen: 23.05.2022.
- [356] ——, “Quelltext Transfer Meta Learning,” <https://gitlab.hs-ruhrwest.de/nico.zengeler/transfer-meta-learning>, 2022, zugegriffen: 23.05.2022.
- [357] ——, “Quelltext und Datensätze zur virtuellen Umgebung Tron,” <https://gitlab.hs-ruhrwest.de/nico.zengeler/gym-tron>, 2020, zugegriffen: 23.05.2022.
- [358] S. D. et al., “Recurrent Layers – Lasagne 0.2.dev1 documentation,” <http://lasagne.readthedocs.io/en/latest/modules/layers/recurrent.html>, zugegriffen: 07.06.2017.

# Formelverzeichnis

Formel 2.1 Überführung der Genotypen in Phänotypen bei evolutionären Algorithmen . . . . .	7
Formel 2.2 Fitnessfunktion in evolutionären Algorithmen . . . . .	7
Formel 2.3 Universelle Approximationseigenschaft des MLP . . . . .	11
Formel 2.4 Ableitung der Sigmoidfunktion . . . . .	13
Formel 2.5 Änderungen des Parametervektors als Ableitung der Verlustfunktion nach den Gewichten . . . . .	14
Formel 2.7 RMSProp Lernratenanpassung . . . . .	15
Formel 2.2 ADAM Gewichtsveränderungen . . . . .	15
Formel 2.21 Funktionsvorschrift des Ausgabeneuron am Beispiel MLP . . . . .	15
Formel 2.23 Aktivierungen in der versteckten Schicht am Beispiel MLP . . . . .	16
Formel 2.24 Verlustfunktion am Beispiel MLP mit einem Ausgabeneuron . . . . .	16
Formel 2.28 Verlustfunktionen bezüglich versteckter Neuronen am Beispiel MLP . . . . .	17
Formel 2.32 Synaptische Gewichtsänderungen am Beispiel MLP . . . . .	17
Formel 2.33 Anwendung der Kettenregel im mehrschichtigen Perzeptron . . . . .	17
Formel 2.34 Faltungsoperation . . . . .	18
Formel 2.35 Aktivierung eines Neurons in einer Faltungsschicht . . . . .	18
Formel 2.36 LSTM CEC ([124]) . . . . .	20
Formel 2.38 Funktionsvorschriften der Ein- und Ausgangstore in LSTM . . . . .	20
Formel 2.4 Funktionsvorschriften der Erinnerungszelle in LSTM . . . . .	20
Formel 2.45 LSTM Funktionsvorschriften (Nach Implementierung von [358]) . . . . .	20
Formel 2.46 Verlustfunktion elastische Gewichtskonsolidierung . . . . .	24
Formel 2.47 A-posteriori-Verteilung bei elastischer Gewichtskonsolidierung . . . . .	26
Formel 2.48 Transferaktivitäten in progressiven neuronalen Netzwerken . . . . .	26
Formel 2.49 Gegnerische Programmmaske für negativen Transfer . . . . .	28
Formel 2.5 Modulierte Eingabe für gegnerische Umprogrammierung . . . . .	28
Formel 2.51 Wahrscheinlichkeitsverteilung einer Symbolreihe für Sprachmodelltransfer . . . . .	30
Formel 2.52 Definition des Zustandswertes . . . . .	33
Formel 2.53 Bestimmung der bestmöglichen Aktion . . . . .	33
Formel 2.56 Aktualisierung des Zustandswertes nach erhaltener Belohnung . . . . .	34
Formel 2.57 Diskontierte Zustandwertaktualisierung . . . . .	34
Formel 2.58 Aktualisierung des Q-Wertes . . . . .	34
Formel 2.59 Der Zustandswert im Q-Learning . . . . .	35
Formel 2.6 Optimale Aktion im Q-Learning . . . . .	35
Formel 2.61 REINFORCE Lernregel nach [177] . . . . .	37
Formel 2.62 A3C Gradientenabstieg . . . . .	37

Formel 3.1	Mittelwertbereinigte Zustandpräsentation der simulierten Derivate-	
	marktumgebung . . . . .	40
Formel 3.2	Aktionsraum der simulierten Derivatemarktumgebung . . . . .	41
Formel 3.3	Erweiterung der Stop-Loss und Take-Profit-Werte für die Machbar-	
	keitsstudie der bestärkenden Lerner in einer realen Handelsumgebung.	51
Formel 3.4	Normalverteilung . . . . .	68
Formel 3.5	Blocknormalisierung in Gradientenorientierungshistogrammen . . .	72
Formel 3.6	Grundlage der Abschätzung von YOLO für Klassen eines Objektes	
	in einem Bezugsrahmen . . . . .	72
Formel 3.7	Hilfsmatrix für die Koordinatentransformation . . . . .	82
Formel 3.8	Koordinatentransformationsmatrix von Kamerabildkoordinaten zu	
	Weltkoordinaten . . . . .	82
Formel 3.9	Kalmanfiltermatrix . . . . .	83
Formel 4	Kalmanfilterung . . . . .	83
Formel 3.2	Genaugkeitsdichte . . . . .	95
Formel 4.1	One-Hot-Encoding der Optimisierungsalgorithmen . . . . .	115
Formel 4.2	Qualitätsmaß für Metalerner . . . . .	117
Formel 5.1	Vereinfachte obere Grenze der Metalerngeneralisierung nach [35]. .	126

# Abbildungsverzeichnis

2.1	Eine Programmablaufskizze genetischer Algorithmen. . . . .	8
2.2	Eine Zusammenstellung der, im Kontext tiefer künstlicher neuronaler Netzwerke, häufig verwendeten Aktivierungsfunktionen. Die verschiedenen neuronalen Aktivierungsfunktionen $\sigma(x)$ weisen einer Anregung entlang einer Dimension $x$ eine beschränkte Ausgabeaktivierung zu. Mit Ausnahme der <i>ReLU</i> Funktion sind alle dargestellten Aktivierungsfunktionen stetig differenzierbar. . . . .	13
2.3	Ein einfaches MLP mit einer Eingabeschicht $\underline{x}$ , einer versteckten Schicht $\underline{z}$ und einer Ausgabeschicht, welche in diesem Beispiel aus nur einem einzigen Neuron $y$ besteht. Die konstanten Biaswerte betragen 1 und dienen als additiver Term. Die synaptischen Gewichte zwischen den Schichten präsentieren sich als Matrizen $\underline{W}^{(1),(2)}$ . . . . .	16
2.4	Eine Zelle der rekurrenten neuronalen Long Short-Term Memory Architektur mit Vergessenstoren, veröffentlicht in [125]. . . . .	21
2.5	Eine konzeptionelle Skizze der Feinabstimmung des Klassifikator Teils eines beispielhaften faltenden neuronalen Netzwerks zur Bildklassifikation; das Lernen relevanter Merkmale geschieht über Faltungsschichten. Links: das vortrainierte Netzwerk. Rechts: die Feinabstimmung. Die blauen Schichten (Eingabeschicht und Merkmalsextraktion) bleiben unverändert, die rote Schicht (Klassifikation) wird feinabgestimmt. . . . .	23
2.6	Konzeptionelle Skizze der elastischen Gewichtskonsolidierung (EWC) für einen beispielhaften zweidimensionalen Parametervektor $\Theta = (\theta_1, \theta_2)^T$ . Für zwei Aufgaben $A, B$ liegen die jeweils optimalen Parametervektoren $\Theta_A^*, \Theta_B^*$ im Minimum der jeweiligen Fehlerregion der Aufgabe ( $A$ blau, $B$ rot). Ein einfaches Neutrainieren eines für Aufgabe $A$ konditionierten Netzwerks führt zum Optimum für Aufgabe $B$ (gestrichelte Linie). Die Methode der EWC führt hingegen zu einem Optimum $\Theta_{AB}^*$ , welches bei gleicher Parameteranzahl beide Aufgaben gut lösen kann. . . . .	25
2.7	Skizze eines progressiven neuronalen Netzwerks nach dem Konzept aus [156]. Die einzelnen neuronalen Netzwerke optimieren jeweils eine Aufgabe $A, B$ oder $C$ . . . . .	27

2.8	Eine Beispielanwendung der gegnerischen Umprogrammierung, entnommen aus [108]. Ein ImageNet Klassifikator soll umprogrammiert werden, sodass er anstelle der gelernten Klassifikation vielmehr die Anzahl weißer Vierecke auf einer schwarzen Fläche zählt. Zunächst wird eine Neuzuordnung zwischen gelernten Klassen und der Anzahl von weißen Vierecken vorgenommen (a). Das zu lernende gegnerische Programmbettet die Nutzdaten der neuen Aufgabe in eine, bezüglich der Neuzuordnung optimierte, Pixelmenge ein; den Angriffsvektor (b). Werden nun die Eingabedaten der neuen Aufgabe mitsamt des gegnerischen Programms präsentiert, so nennt das derart umprogrammierte Netzwerk die Klasse, welcher nach der Neuzuordnung der Anzahl der weißen Vierecke entspricht (c) . . . . .	29
2.9	Darstellung des Weltmodellkonzepts für visuelles Verstärkungslernen, entnommen aus [19]. Zunächst werden die Beobachtungen in eine latente Variable $z_t$ encodiert. Das Weltmodell $M$ modelliert eine Wahrscheinlichkeitsverteilung für die Folgezustände $z_{t+1}$ , ausgehend vom aktuellen Zustand $z_t$ und einer Aktion $a$ . Die Transferleistung erbringt der Controller $C$ , welcher auf Grundlage der neuronalen Aktivierung im Weltmodell und der Beobachtung des aktuellen Zustands der Umgebung die optimale Strategie bezüglich für den jeweiligen Aktionsraum approximiert. . . . .	31
3.1	Ein Sequenzdiagramm zur Veranschaulichung des Ablaufs eines Lernschrittes in simulierten Marktumgebung. Die beteiligten Objekte, also der Agent und die Umgebung, kommunizieren über Zustand, Aktion und Belohnung.	41
3.2	Das mehrschichtige Perzeptron, entnommen aus [7]. . . . .	44
3.3	Das rekurrente neuronale Netzwerk, entnommen aus [7]. . . . .	44
3.4	Das rekurrente Faltungsnetzwerk, entnommen aus [8]. . . . .	45
3.5	Verlauf des Indexwertes <i>US500</i> während Aufzeichnung der historischen Marktdaten im Jahr 2020 für die simulierte Derivatemarktumgebung, entnommen aus [8]. . . . .	48
3.6	Auswertungsergebnisse aus der Machbarkeitsstudie, basierend auf den Untersuchungen aus [7]. Oben: Aktionsverteilung. Mitte: Kapitalentwicklung. Unten: Profitverteilung. . . . .	50
3.7	Die neuronale Architektur für die Anwendung im Echtzeittest, entnommen aus [7]. . . . .	52
3.8	Eine Lerndynamik des LSTM-Modells im Echtzeittest, entnommen aus [7]. X-Achse: Anzahl Lernschritte in Tausend, Y-Achse: durchschnittliche Belohnung (blau), maximale Belohnung (grün), minimale Belohnung (rot).	52
3.9	Die Auswertung der erweiterten LSTM-Architektur im Echtzeittest aus [7]. Links: Die erzielten Gewinne. Rechts: Die Eigenkapitalentwicklung. . . . .	53

3.10 Links: Die Anzahl der positiven Ergebnisse in allen Versuchen einer bestimmten Beobachtungsdauer. Das Maximum von zehn positiven Versuchen findet sich bei einer Beobachtungsdauer von zehn Minuten, das zweitbeste Ergebnis bei 45 Sekunden. Rechts: Die Gesamtzahl der positiven Ergebnisse pro Basiswert, addiert über alle Zeitspannen. Beide Graphen sind entnommen aus [8]. . . . .	55
3.11 Ein UML-Diagramm zur Skizze der Zusammenhänge zwischen Sensordaten, neuronalen Netzwerken und Handgesten. Die Sensordaten entstammen entweder Farb- oder Raumtiefenkameras. Verschiedene neuronale Netzwerke versuchen aus diesen Daten eine Abbildung zu Handgesten zu lernen, welche sich über eine Ganzzahl als Klasse eines Datensatzes klassifizieren lassen. . . . .	57
3.12 Die in der Metastudie von [9] zur Handgestenerkennung untersuchten Beiträge in chronologischer Reihenfolge. . . . .	58
3.13 Eine Veranschaulichung von Raumtiefendaten einer Handgeste in Form von Punktwolken aus verschiedenen Aufnahmewinkeln, entnommen aus [9]. . . . .	59
3.14 Die zehn Handgestenklassen aus dem REHAP Datensatz von [248], entnommen aus [9]. Die Handgesten in der oberen Reihe zeigen verschiedene Anzahlen gehobener Finger, was als numerischer Wert interpretiert werden kann. Die Handgestenklassen in der unteren Reihe zeigen andere bedeutungsvolle Gesten, wie beispielsweise eine Faust oder eine flache Hand, welche Start- und Endpunkte einer dynamischen Handgestensequenz verwendet werden können. . . . .	61
3.15 Die FOANet-Architektur, entnommen aus [254]. Sie besteht aus einem separaten Kanal für jede Fokusregion (global, linke Hand, rechte Hand) und jede Eingabemodalität (RGB, Raumtiefe, RGB-Fluss und Raumtiefenfluss). . . . .	63
3.16 Die rekurrente dreidimensionale Faltungsarchitektur, entnommen aus [246]. Als Eingabe verwendet das Netzwerk eine dynamische Geste in Form von aufeinanderfolgenden Frames. Es extrahiert lokale räumlich-zeitliche Merkmale über ein 3D-CNN und speist diese in eine rekurrente Schicht ein, welche die Aktivierung über die gesamte Sequenz aggregiert. Anhand dieser Aktivierungen nennt eine Softmax-Schicht dann eine Wahrscheinlichkeitsverteilung über den Klassenraum. . . . .	64
3.17 Ein UML-Diagramm zur Illustration des Zusammenspiels zwischen den Farbbildern, den in dieser Problemstellung untersuchten Modellen zur Personendetektion und der geforderten Ausgabe in Form von Positionen der Menschen im Bild. . . . .	65
3.18 Vorverarbeitete Beispielbilder. Oberste Reihe: Das Gauß'sche Rauschen steigt von $\sigma = 0$ (links) auf $\sigma = 250$ (rechts). Zweite Reihe: Die Stärke eines blendenden Lichts steigt von 0% einer Lichtkartenaddition (links) auf 100% (rechts). Dritte Reihe: sechs Verdeckungsmodalitäten. Untere Reihe: die kombinierten Störungen, von $i = 0$ (links) bis $i = 10$ (rechts). Veröffentlicht in [10]. . . . .	67

3.19 Obere Zeile: Hintergrundbilder der Kameras. Untere Zeile: Die entsprechenden Lichtkarten, welche auf die Kamerabilder addiert werden. Veröffentlicht in [10]. . . . .	69
3.20 Exemplarische Erkennungen der drei untersuchten Methoden auf verschiedenen Bildern der simulierten Arbeitsaufgaben aus [10]. Links: OP. Mitte: YOLO. Rechts: HOG. . . . .	71
3.21 Obere Reihe: Beständigkeiten gegen Rauschen. Zweite Reihe: Antizipation des Blendlichteffektes. Dritte Reihe: teilweise verdeckte Personen in den sechs Modalitäten. Untere Reihe: der kombinierte Härtefall. Linke Spalte: Genauigkeit der Personenerkennung. Mittlere Spalte: Genauigkeit der Fußpunktterkennung. Rechte Spalte: Fehlerkennungstrend. Veröffentlicht in [10]. . . . .	75
3.22 Beispielhafte <i>Augmented Reality</i> , abhängig von der aktuellen Position des Nutzenden. Links: eine Anweisung zum Überprüfen einer Seriennummer. Rechts: Anzeige einer Evakuierungsroute im Notfall. Entnommen aus [11]. . . . .	78
3.23 Das Programmablaufdiagramm des Personenverfolgungsverfahrens aus [11]. Die obere Reihe zeigt Bilder aus dem Datensatz dieser Studie. Von links nach rechts: Kameras $C_1$ bis $C_5$ , wie in Abbildung 3.24 dargestellt. Die Person bewegt sich derzeit in Sichtweite der Kameras $C_1, C_2, C_5$ , aber außer Sichtweite der Kameras $C_3$ und $C_4$ . . . . .	79
3.24 Der Testparcour der simulierten industriellen Umgebung. Die Testpersonen folgen einem vorgeschriebenen Weg (blau) und lösen eine Reihe von Aufgaben. Fünf Kameras (grün) nehmen währenddessen Videomaterial auf. entnommen aus [11]. . . . .	81
3.25 Die mittleren Abweichungen zwischen den geschätzten Kamerapositionen und den Grundwahrheitspositionen, welche von der AR-Brille für jede Testperson aufgezeichnet wurden. Roten Balken: die mittleren Abweichungen für die rohen Schätzungen. Blaue Balken: die mittleren Abweichungen für die kalmangefilterten Positionen. Entnommen aus [11]. . . . .	84
3.26 Die berechneten und gemessenen Spuren der Testpersonen, alle Skalen in Metern. Oben links: die Grundwahrheitsdaten, wie sie von den AR-Brillen bereitgestellt werden. Oben rechts: die Rohdaten, also die Spuren nach der Erkennung von Fußpunkten und der Anwendung des ersten regelbasier-ten Systems. Unten links: die Ergebnisse nach der Spur trennung. Unten rechts: die Spuren nach der Kalmanfilterung. Diese ähneln weitgehend den Grundwahrheitspositionen. Es zeigt sich eine metrische Verzerrung; besonders der Bereich um den Start- und Endpunkt herum ist gestreckt, weil er von nur einer Kamera beobachtet wird. Entnommen aus [11]. . . . .	85
3.27 Von links nach rechts: Dauer in Sekunden, zurückgelegte Wegstrecken in Metern, durchschnittliche Geschwindigkeiten in Metern pro Sekunde. Von oben nach unten: AR-Daten, rohe Erkennungen, Statistiken pro einzelne Spur, die kalmangefilterten Endergebnisse. Entnommen aus [11]. . . . .	86
3.28 Beispielbilder aus dem Datensatz Hymenoptera [303] (oben) und dem Smartphonedatensatz [6, 26] (unten). . . . .	89

3.29	Chronologie der Entwicklung von tiefen faltenden neuronalen Netzwerken zur Bildklassifikation, entnommen aus [6]. . . . .	90
3.30	Durchschnittliche Genauigkeitsdichten nach einer Episode (oberen beiden) und nach zehn Episoden (unteren beiden), jeweils nach Feinabstimmung aller Schichten und der Klassifizierungsschichten, entnommen aus [6]. . . . .	96
3.31	Modellgrößen und Genauigkeiten im Vergleich zur Trainingszeit für alle Aufgaben und Modelle nach der Feinabstimmung aller Schichten, entnommen aus [6]. . . . .	97
3.32	Modellgrößen und Genauigkeiten im Vergleich zur Trainingszeit für alle Aufgaben und Modelle nur nach Feinabstimmung der Klassifizierungsschichten, entnommen aus [6]. . . . .	97
3.33	Detaillierte Metadaten für alle Modell- und Aufgabenkombinationen für die vollständige Feinabstimmung. Die Suffixe kürzen die verschiedenen Aufgaben ab: A: Smartphones (augmented), C: CIFAR10, H: Hymenoptera, M: MNIST, O: Smartphones (original). . . . .	98
3.34	Detaillierte Metadaten für alle Modell- und Aufgabenkombinationen für die Klassifikatorfeinabstimmung. Die Suffixe kürzen die verschiedenen Aufgaben ab: A: Smartphones (augmented), C: CIFAR10, H: Hymenoptera, M: MNIST, O: Smartphones (original). . . . .	99
4.1	Der methodische Ablauf des <i>Transfer Meta Learning</i> als Flussdiagramm. Mit ausgewählten Aufgaben $t$ und vortrainierten Modellen $m$ werden in einer Gittersuche über die Verlustfunktionen $L$ , die Optimierer $o$ , die Stapelgrößen $b$ , die Lernraten $\eta$ Metadaten erzeugt. Mit Eigenschaften der Aufgaben, etwa der Anzahl der Klassen $n_c$ und Datenpunkte $n_d$ und den stochastischen Momenten des Labelraumes $\mu$ , werden Metalerner trainiert. Um einen optimalen MLP Metalerner zu finden, wird erneut eine Gittersuche durchgeführt, diese zusätzlich mit verschiedenen Epochenzahlen $n_e$ und Neuronenzahlen $n_p$ . Veröffentlicht in [5]. . . . .	102
4.2	Darstellung der Methodik zur Anwendung von Metamodellwissen für Transferlernaufgaben nach dem EVA-Prinzip. Dem Metamodell werden verschiedene Aufgabendaten $A$ , Modelldaten $M$ , Hyperparameter $H$ mit- samt der zugehörigen Transferlernresultate, beispielsweise die Genauigkeit $G$ , die Trainingszeit $T$ oder weitere a posteriori Information, als Eingabe präsentiert. . . . .	105
4.3	Eine Skizze zur Illustration der Polymorphie und dynamischen Bindung. Von den Basisklassen zur Modellierung beispielhafter konkreter Aufgaben oder Optimierungsverfahren erben verschiedene Unterklassen, welche das jeweilige Optimierungsmodell beziehungsweise die jeweilige Datenlage konkretisieren. In einer Metamodellklasse werden mit dynamischer Bindung die Referenzen auf die jeweiligen Unterklassen in einer Indexstruktur, beispielsweise einer Liste, zusammengeführt. . . . .	108

4.4 Ein UML-Klassendiagramm zur Veranschaulichung des Aufgabenkonzeptes. Eine abstrakte Klasse <i>Task</i> dient als Basisklasse für verschiedener maschineller Lernaufgaben (überwachtes Lernen, unüberwachtes Lernen, Verstärkungslernen). Die Blätter des dargestellten Graphen deuten verschiedene Formen von maschinellen Lernaufgaben an, welche im Rahmen dieser Arbeit untersucht wurden oder Erwähnung fanden. . . . .	109
4.5 Ein UML-Klassendiagramm zur Vererbungshierarchie der einzelnen Modellklassen. Alle Optimierungsverfahren erben von der Basisklasse <i>Model</i> . Zur besseren Übersichtlichkeit wurden in dieser Darstellung sind nicht alle denkbaren Optimierungsverfahren ausformuliert, werden aber angedeutet.	111
4.6 Ein UML-Klassendiagramm zur Beschreibung des Metamodells. Diese Klasse verfügt über Listen bekannter Modelle und Aufgaben und bietet eine Methode zum Einlesen zugehöriger Metadaten, zum Lernen aus diesen und zur Inferenz für eine Eingabeaufgabe. . . . .	112
4.7 Das grundlegende Schema aller in den Untersuchungen aus [5] verwendeten mehrschichtigen Perzeptrone. . . . .	115
4.8 Metamodellinferenzen für eine leichte Aufgabe ( <i>FashionMNIST</i> , oberes Raster) und eine schwere Aufgabe ( <i>Places365</i> , unteres Raster), basierend auf [5]. . . . .	122
4.9 Metamodellinferenzen (links) im Vergleich zu den tatsächlichen Modellgenauigkeiten (rechts) bei den entsprechenden Testaufgabendaten von <i>FashionMNIST</i> (oben) und <i>Places365</i> (unten), entnommen aus [5]. . .	123

# Verzeichnis der Algorithmen

1	(1 + 1)-Evolutionsstrategie nach Rechenbergs $\frac{1}{5}$ -Regel . . . . .	10
2	$(\mu, \lambda)$ -Evolutionsstrategie mit Kovarianzmatrixadaption . . . . .	10
3	Fehlerrückführung . . . . .	14
4	Q-Lernschritt . . . . .	35
5	Hinzufügen zum priorisierten Erinnerungsspeicher . . . . .	36
6	Aktualisieren des priorisierten Erinnerungsspeichers . . . . .	36
7	Simulierte Marktlogik . . . . .	43
8	Training in der Marktsimulation . . . . .	47
9	Bildrauschen . . . . .	68
10	Blendlichteffekt . . . . .	68
11	Verdeckung . . . . .	70
12	Kombinierte Bildstörung . . . . .	71

# Abkürzungsverzeichnis

**AI** Artificial Intelligence

**ML** Machine Learning

**DL** Deep Learning

**MLP** Multilayer Perceptron

**LSTM** Long Short-Term Memory

**CEC** Constant Error Carousel

**CNN** Convolutional Neural Network

**DNC** Differentiable Neural Computer

**OP** OpenPose

**YOLO** You Only Look Once

**HOG** Histogram of oriented gradients

**EWC** Elastic Weight Consolidation

**IMM** Incremental Moment Matching

**MDP** Markov Decision Process

**POMDP** Partially Observable Markov Decision Process

**ARP** Action Replay Process

**CfD** Contract for Difference

**IOU** Intersection over Union

**CE** Circular Economy

**UML** Unified Modeling Language