Datenflussoptimierung mit dynamischer Allokation von Ressourcen in der Industrie

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Maschinenbau und Verfahrenstechnik der
Universität Duisburg-Essen

zur Erlangung des akademischen Grades

einer

Doktorin der Ingenieurwissenschaften

Dr.-Ing.

genehmigte Dissertation

von

Julia Katharina Rosenberger
aus
Hof

Gutachter Prof. Dr.-Ing. Dr. h.c. Dieter Schramm

Prof. Dr.-Ing. Michael Bühren

Prof. Dr.-Ing. Frank Lobeck

Tag der mündlichen Prüfung: 30. November 2023

Danksagung

Die vorliegende Arbeit entstand während meiner Zeit bei der Bosch Rexroth AG in Lohr am Main. Mein besonderer Dank gilt Herrn Prof. Dieter Schramm und Herrn Prof. Michael Bühren für die wohlwollende Unterstützung und die wertvollen Anregungen, die zum Gelingen dieser Arbeit wesentlich beigetragen haben. Ebenfalls herzlich danken möchte ich Herrn Prof. Frank Lobeck für die Übernahme des Korreferats und die kritische Durchsicht der Arbeit.

Ich danke den zahlreichen Kollegen bei Bosch Rexroth, die mir stets mit Rat und Tat zur Seite standen. Allen voran gilt mein Dank Herrn Dr.-Ing. Andreas Selig und Frau Dr.-Ing. Mirjana Ristic, die mich jederzeit unterstützt haben und mit zahlreichen fachlichen Diskussionen einen wertvollen Beitrag zu der Arbeit geleistet haben. Darüber hinaus danke ich allen Studenten, deren Ergebnisse zur Entstehung dieser Arbeit beigetragen haben. Dies gilt insbesondere für die Arbeiten von Felix Rauterberg.

Weiterhin danken möchte ich den Mitarbeitern am Lehrstuhl für Mechatronik für die kollegiale und offene Zusammenarbeit und die interessanten Diskussionen.

Abschließen möchte ich mit einem ganz besonderen Dank für das Verständnis und die Unterstützung meiner Familie.

Kurzfassung

Daten sind die Basis für neue Geschäftsmodelle und Technologien wie beispielsweise der vorausschauenden Instandhaltung von industriellen Anlagen. Die Erfassung und Verarbeitung von Daten, insbesondere von Datenflüssen in einem Netzwerk, hat deshalb entscheidenden Einfluss auf den Erfolg der Industrie 4.0. Zur Verringerung von Latenzen sowie Kosten und aus Aspekten der Datensicherheit wird die Verlagerung der Datenanalyse an den Ort der Datenerfassung untersucht. Daraus ergeben sich besondere Herausforderungen hinsichtlich limitierter Rechen- und Kommunikationsressourcen, der dezentralen Struktur mit einer Vielzahl heterogener Teilnehmer sowie der Anforderung an hohe Verarbeitungsgeschwindigkeiten. Es ist sicherzustellen, dass relevante Informationen zeitnah verfügbar sind und der Verlust dieser vermieden wird. Zur Lösung der genannten Optimierungsbedarfe werden in der vorliegenden Arbeit drei Ansätze identifiziert und jeweils die Potentiale neuer Technologien untersucht. Der erste Ansatz umfasst die frühzeitige Extraktion relevanter Informationen und Reduktion der Datenmenge unter Einsatz von Anomalieerkennung und Datenkompression. Der zweite Ansatz zielt auf die Schaffung einer Vertrauensbasis zur Absicherung von Daten mittels Distributed Ledger Technologie. Das dritte Potential wird in dem optimierten Nutzen der vorhandenen limitierten Ressourcen gesehen. Aufgrund der dezentralen Struktur des industriellen Internet der Dinge basiert der Lösungsansatz auf intelligenten Multi-Agenten-Systemen. Die Evaluation der einzelnen Bestandteile sowie deren Zusammenspiel in einem Gesamtsystem bestätigt, dass alle ausgearbeiteten Methoden aufgrund ihrer hohen Effizienz nahe der Datenerfassung ausführbar sind, Informationen frühzeitig bereitgestellt werden und Datenverlusten aufgrund von Ressourcenengpässen entgegengewirkt wird. Die Datenflussoptimierungen legen die Grundlage für zukünftige Systeme zur verteilten Verarbeitung von Datenflüssen am Netzwerkrand und sind ein relevanter Treiber für die Industrie 4.0, da sie ermöglichen, bestehende Begrenzungen hinsichtlich der verfügbaren Ressourcen zu überwinden und die Datensicherheit auf dem geforderten Niveau zu gewährleisten.

Abstract

The collection and processing of data, especially streaming data, has a decisive influence on the success of Industry 4.0. They are the basis for new business models and technologies such as condition monitoring or predictive maintenance of industrial plants. To achieve this, it must be ensured that relevant information is available in a timely manner and that the loss of this information is avoided. While the amount of data and the demand for data analysis are increasing, the available resources, especially computing capacity, memory and bandwidth, are still limited.

In this work, essential optimization potentials in streaming data in Industry 4.0 are identified and investigated. In particular, the early extraction of relevant information and the reduction of the amount of data as well as the protection of data are to be mentioned. Specifically, the use of methods of anomaly detection, data compression and distributed ledger technology is examined for this purpose. Furthermore, great potential is seen in optimizing the utilization of limited resources. Due to the decentralized structure of the Industrial Internet of Things, the focus for this is on intelligent multi-agent systems. The evaluation of the individual components proposed in this thesis and their interaction in an overall system shows that the elaborated methods of data processing can be performed closer to the data acquisition due to their high efficiency and thus, information can be provided at an early stage. In addition, the distributed ledger technology IOTA selected in the context of this thesis is evaluated on the basis of two use cases and its applicability on industrial edge devices is confirmed. Furthermore, the optimization of usage of limited resources by multi-agent reinforcement learning systems can significantly enhance the possibility of edge computing in IIoT and prevents data loss due to resource bottlenecks.

The results illustrate that data stream optimization is an important driver for Industry 4.0, as it helps to overcome existing limitations in terms of resources and data safety.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

A2C	Advantage Actor Critic
AAS	Asset Administration Shell
AEC	Agent Environment Cycle
CNN	Convolutional Neural Network
CPS	Cyber-Physical Systems
CRISP-DM	Cross-Industry Standard Process For Data Mining
DAG	Directed Acyclic Graph
dApp	Decentralized Application
DBMS	Data Base Management System
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DIKW	Data, Information, Knowledge, Wisdom
DLT	Distributed Ledger Technology
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DRNG	Distributed Random Number Generator
DSPS	Distributed Stream Processing Systems
DWT	Discrete Wavelet Transform
EEM-KDE	Extended Merging Kernel Density Estimation for Edge-Devices
FLOPS	Floating Point Operations Per Second
FPC	Fast Probabilistic Consensus

HiL	Hardware-in-the-Loop
IoT	Internet of Things
IIoT	Industrial Internet of Things
I4.0	Industrie 4.0
JSP	Job Shop Scheduling
KDE	Kernel Density Estimation
LSTM	Long-Short Term Memory
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent-System
MDP	Markov Decision Process
MG	Markov Game
ML	Machine Learning
MMDP	Multi Agent Markov Decision Process
MPS	Messages Per Second
MSE	Mean Squared Error
MTS	Multivariate Time Series
ONNX	Open Neural Network Exchange
POMDP	Partially Observable Markov Decision Process
POMG	Partially Observable Markov Game
PPO	Proximal Policy Optimization
PoW	Proof of Work
RAMI 4.0	Referenzarchitekturmodell 4.0
RBM	Restricted Boltzman Machine
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RURTS	Restricted Uniform Random Tip Selection
SB3	Stable Baselines 3
SC	Smart Contract
SDK	Software Developer Kit
TPS	Transactions Per Second
URTS	Uniform Random Tip Selection
UTS	Univariate Time Series
WSN	Wireless Sensor Network

KAPITEL 1

Einleitung

In diesem einleitenden Kapitel wird zunächst die Motivation dargelegt und die Problemstellung, die die Datenflussoptimierung löst, beschrieben. Anschließend werden im Stand der Technik bestehende Arbeiten auf dem übergeordneten Gebiet der Systeme zur verteilten Verarbeitung von Datenflüssen zusammengefasst. Aus der Problemstellung und den bestehenden Hindernissen wird das Ziel der Arbeit benannt und die zu untersuchende allgemeine Forschungsfrage gestellt. Es werden drei Lösungsansätze benannt, deren Untersuchung den Kern der vorliegenden Arbeit darstellen. Abgeschlossen wird das Kapitel mit einer Übersicht über den Aufbau der Arbeit.

1.1 Motivation

Die digitale Revolution ist eine weltweite, durch Computer und Digitaltechnik ausgelöste, disruptive Transformation. Sie zeichnet sich durch eine verstärkte Substitution von Technologien aus und beeinflusst und verändert nahezu alle Lebensbereiche (Lexa 2021). Zentraler Inhalt der digitalen Revolution, welche auch auf die Industrie einen starken Einfluss hat und in diesem Kontext auch als vierte industrielle Revolution bezeichnet wird, sind *Daten* (Lu 2017; Peña-Cabrera, Lomas und Lefranc 2019). Das Resultat dieser vierten industriellen Revolution ist eine neue Form der Industrie, die mit dem Begriff Industrie 4.0 (I4.0) bezeichnet wird. Sie zeichnet sich durch durchgängige, integrierte und dennoch offene digitale Prozesse entlang der kompletten industriellen Wertschöpfungskette aus (Milisavljevic-Syed, Thames und Schaefer 2020). Um dies zu erreichen, bedarf es einer Neugestaltung der Organisation, die auf den Prinzipien der Vernetzung, Informationstransparenz, dezentraler Entscheidungen und technischer Assistenz basiert (Hermann, Pentek und

Otto 2016). In allen der genannten Prinzipien spielen Daten und aus diesen abgeleitete Informationen eine wesentliche Rolle. Eng verknüpft mit der Umsetzung dieser Prinzipien ist das Internet der Dinge (engl. Internet of Things (IoT)), welches als virtuelles Netzwerk, in dem Objekte in Form von Netzwerkknoten kontinuierlich große Mengen an Daten über sich selbst und ihre Umgebung übertragen, verstanden wird. Im industriellen Kontext wird das IoT auch als Industrielles IoT (engl. Industrial Internet of Things (IIoT)) bezeichnet (Satyavolu u. a. 2014). Im Rahmen der Digitalisierung werden Daten als Basis für neue Technologiefelder und Geschäftsmodelle erkannt (Nagl und Bozem 2018), weshalb auch in der Industrie ein rapider Anstieg der erfassten Datenmengen zu verzeichnen ist. Die Gründe dafür sind einerseits der vermehrte Einbau von Sensoren sowie die technischen Fortschritte hinsichtlich der Datenerfassung, wie beispielsweise hochauflösende Bildaufnahmen. Andererseits fördert die vernetzte Kommunikation zusätzlich den Austausch von Daten zwischen den verschiedenen Teilnehmern. Dementsprechend handelt es sich bei den Daten in erster Linie um Datenflüsse, insbesondere Zeitreihendaten. Es ist zu erwarten, dass die Datenmenge in industriellen Anlagen und miteinander vernetzten physischen Geräten weiterhin stark steigt und somit mit den aktuell verfügbaren Bandbreiten nicht übermittelt werden kann (Chen, Wan u. a. 2018; Sanabria-Russo u. a. 2019). Als konkrete Beispiele für neue datenbasierte Anwendungen sind die Zustandsüberwachung und die darauf aufbauende prädiktive Instandhaltung zu nennen. Da die Zustandsüberwachung bereits von den frühen Phasen des neuen Industriezeitalters profitieren kann (Shahzad und O'Nils 2018), ist sie ein verbreitetes Technologiefeld in der I4.0. Durch die verstärkte Ausstattung mit Sensoren kann der Zustand des Systems genauer beschrieben werden. Mittels der neuen Erkenntnisse, die aus den Daten gewonnen werden, lassen sich Produkte und Prozesse optimieren. Die datenbasierte Produktion kann dadurch unter anderem die Produktqualität und Produktionseffizienz steigern und dabei gleichzeitig den Energieverbrauch reduzieren (Chen, Wan u. a. 2018). Es wird deutlich, dass Wettbewerbsvorteile heutzutage auch software- und datenbasiert erzielt werden und nicht durch das produzierte Produkt allein.
Die I4.0 zeichnet sich dementsprechend durch schwerwiegende Veränderungen hinsichtlich des Einsatzes von Software und der Kommunikationsstruktur aus. Informationstechnologie, operative Technologie und Kommunikationstechnik verschmelzen zunehmend (Thames und Schaefer 2016; Felser, Rentschler und Kleineberg 2019) und die vorherrschende Darstellung der Architektur in Form der Automatisierungspyramide wird durch neue Architektur-Modelle ergänzt (vgl. Abschnitt 2.1). In der Vision der I4.0 kommunizieren alle Teilnehmer miteinander und lösen beispielsweise Probleme selbstständig auf Basis der zur Verfügung stehenden Daten und Informationen (Chen, Wan u. a. 2018; Huber 2016).

Einer der entscheidenden Faktoren für den Erfolg datenbasierter Strategien in der Industrie ist die Bereitstellung prozesskritischer Informationen in Echtzeit (BITKOM e. V., VDMA e. V. und ZVEI e. V. 2015; Hermann, Pentek und Otto 2016). Durch die zeitnahe Auswertung ist gewährleistet, dass technische Systeme rechtzeitig auf Veränderungen reagieren können. Daraus folgt auch, dass Informations-

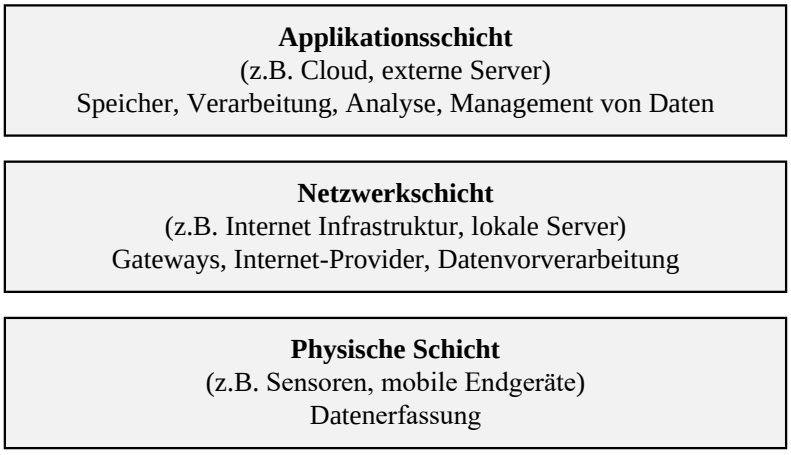


Abbildung 1.1: Drei-Schichten-Architektur von IoT Lösungen (Shahzad und O'Nils 2018)

verluste, beispielsweise durch Bandbreitenengpässe oder mangelnden Speicherplatz, unbedingt zu verhindern sind. Aus diesem Grund führen datenbasierte Strategien zu den genannten architekturellen Änderungen.

Etablierte Systeme zur verteilten Verarbeitung von Datenflüssen (engl. Distributed Stream Processing Systems (DSPS)) erfordern zusätzlich zu der IIoT-Kommunikation zwischen bestehenden Endgeräten eine Cloud-Anbindung. Diese Entwicklung wird auch in Form einer Drei-Schichten-Architektur von IoT-Lösungen beschrieben (Shahzad und O'Nils 2018), Abbildung 1.1. Die physische Schicht entspricht in der Industrie der Sensor-Aktor-Ebene, in der Daten erhoben werden. Die mittlere Schicht beinhaltet die Netzwerkinfrastruktur für den Datentransfer. Die oberste Schicht, die Applikationsschicht, befindet sich in der Cloud oder externen Rechenzentren, in denen die Daten gespeichert, verarbeitet und analysiert werden. DSPS sind der Netzwerkschicht oder auch Middleware, d. h. Software, die die physische Schicht bestehend aus physischen Geräten und Daten(-flüssen) mit den Applikationen zur Datenflussverarbeitung verknüpft, zugeordnet.

Die Datenverarbeitung und Datenanalysen in der Cloud, das sogenannte Cloud Computing, wird zwar einerseits als Innovationstreiber beschrieben (Lemke und Brenner 2015), allerdings hat es auch wesentliche Nachteile. Mit dem aktuellen Vorgehen, bei dem die verteilte Datenflussverarbeitung auf der Applikationsschicht in der Cloud erfolgt, beschränken Latenzen die Einsatzmöglichkeiten. Damit stellt die bereits genannte Anforderung der Echtzeitfähigkeit (BITKOM e. V., VDMA e. V. und ZVEI e. V. 2015) an ein intelligentes Datenmanagement eine Herausforderung dar, die sowohl in bestehenden als auch neuen Architekturen häufig schwer zu bewältigen oder noch ungelöst ist. Außerdem sind, wie bereits beschrieben, Bandbreitenengpässe ein weiteres Problem, das durch Cloud Computing begründet wird. So führt die Übertragung der großen Rohdatenmengen vermehrt zu schwerwiegenden Folgen wie Datenverlusten und Fehlern (Nikoui, Rahmani und Tabarsaied 2019). Eine weitere Herausforderung bei Cloud Anwendungen ist die Gewährleistung der

Datensicherheit. Nach einer Umfrage sehen 78 % der deutschen Unternehmen Sicherheitsrisiken in Cloud Computing (Europäische Wirtschaftsforschung GmbH 2015). Die genannten Aspekte der hohen Latenzen, Bandbreitenengpässe und Sicherheitsrisiken zeigen, dass die bestehende Architektur sowie der Einsatz von Cloud-Lösungen allein nicht ausreichend sind. Sie begründen die wachsende Bedeutung der Datenverarbeitung am Netzwerkrand, dem sogenannten Edge bzw. Fog Computing. Während Edge Computing alle Verarbeitungen durch die Teilnehmer des lokalen Netzwerks, in anderen Worten den Endgeräten innerhalb der Anlagen, umfasst, bezeichnet Fog Computing Auswertungen, die auf weitere, mit dem lokalen Netzwerk verbundene Geräte, beispielsweise lokale Server, verlagert werden (Dautov, Distefano u. a. 2017). Eine aktuelle Entwicklung ist, dass die Vorverarbeitung und Zwischenspeicherung von Daten zunehmend durch die Endgeräte erfolgt und sich somit näher an die Datenerfassung verschiebt, wobei die Datenanalysen selbst weiterhin auf der Cloud-Ebene stattfinden (Nikoui, Rahmani und Tabarsaied 2019). Die stark variierenden und teilweise konkurrierenden Ansprüche – unter anderem die Forderungen nach mehr Daten, minimalen Latenzen, höchster Verfügbarkeit, Energieeffizienz und geringe Kosten für die Kommunikation – sind herausfordernd für die Kommunikationsinfrastruktur und bestehende Informationsverarbeitungssysteme (Wang und Zhang 2020). Die genannten Anforderungen wirken als treibendes Potential für die Weiterentwicklung von Hardware hinsichtlich der Rechen- und Speicherkapazität sowie dem Ausbau der verfügbaren Bandbreiten. Während das Moor'sche Gesetz besagt, dass sich die Anzahl an Transistoren pro Flächeneinheit in regelmäßigen Abständen verdoppelt, und dies für PCs durchaus zutreffend ist, ist nach (Pisani u. a. 2017) und (Schaller 1997) bei eingebetteten Systemen von einer anderen Entwicklung auszugehen. Anstelle der ständigen Erhöhung der Rechenleistung stehen, wie zum Beispiel bei drahtlosen Sensornetzwerken (engl. Wireless Sensor Network (WSN)), auch die Senkung des Energiebedarfs sowie die Reduzierung der Kosten im Vordergrund. Die reine Anpassung der Hardwareressourcen oder das Verlagern in die Cloud ist somit weder effizient noch nachhaltig, da die Anforderungen der Datenanalysen stetig steigen. Stattdessen werden in dieser Arbeit zusätzliche Ansätze, wie die Untersuchung der optimierten Ausnutzung von verfügbaren Ressourcen sowie das frühzeitige Analysieren der Daten näher an der Sensorebene, betrachtet.

Dazu ist es notwendig, sich über das Ziel der *Big Data*-Strategien klar zu werden. Das Interesse gilt nicht den Daten an sich, sondern den Informationen, die in diesen Daten enthalten sind und daraus extrahiert werden können. Da die Informationsmenge wesentlich kleiner als die ursprüngliche Datenmenge ist, stellt jeder Vorgang der Informationsextraktion zugleich eine Datenreduktion dar. Eine langfristige Lösung stellt ein intelligentes, dezentrales und flexibles Datenmanagement dar, welches bei beliebigen Hardware-Gegebenheiten für diese Anforderungen die optimale Kompromisslösung findet, sowie die optimale Ausnutzung der limitierten verfügbaren Ressourcen unterstützt.

1.2 Stand der Technik

Aufgrund der zunehmenden Vernetzung und der steigenden Datenmengen in der I4.0 wird in der Optimierung des Datenflussmanagements und der Bandbreitenauslastung Potential erkannt (Hesse u. a. 2019; Huber und Kaiser 2017; Chen, Wan u. a. 2018). Die in Abschnitt 1.1 beschriebene Situation zeigt, dass eine Reduktion der Datenmenge ohne den Verlust relevanter Informationen erforderlich ist. Aufgrund der genannten Nachteile von Cloud-Lösungen soll dies lokal, innerhalb oder nahe des produzierenden Systems, und somit näher an der Datenerfassung, geschehen. Außerdem wird Optimierungsbedarf bei der Datenweiterleitung gesehen. Die vorliegende Arbeit verfolgt den Ansatz einer Cloud-unabhängigen Datenflussverarbeitung. Sie betrachtet die Verarbeitung nur auf den Endgeräten nahe der Datenerfassung. Der Einsatz von Cloud-Computing soll nur optional sein.

Mit Blick auf die Datenflussverarbeitung kommen auf der Middleware-Schicht in Datenfluss-Systemen zunehmend DSPS zum Einsatz. Aus diesem Grund wird zunächst die Evolution sowie die Merkmale der unterschiedlichen Generationen von DSPS zusammengefasst. Anschließend wird ein Ausblick über die nächste zu erwartende Generation gegeben und bestehende Arbeiten mit Bezug zur Verarbeitung nahe der Datenerfassung vorgestellt.

Wesentliche Merkmale eines DSPS sind die einheitliche Verwaltung der Datenflüsse, Programmierschnittstellen und Vereinfachungen zur Implementierung von Datenfluss-Applikationen. Zusätzlich zeichnen sich einige etablierte DSPS durch ihre horizontale Skalierbarkeit, Fehlertoleranz und Zustandsverwaltung aus.

(Hirzel u. a. 2014) stellen bereits einen umfassenden Katalog mit Optimierungspotentialen für die Datenflussverarbeitung vor. Dabei werden unter anderem die Gebiete der Lastverteilung, Neuanordnung der Operatoren, d.h. Einheiten zur Verarbeitung der eingehenden Daten, oder Entfernen von Redundanzen genannt. Die aus dem Jahr 2014 stammende Arbeit berücksichtigt jedoch noch nicht die relevanten kürzlichen Veränderungen der DSPS, welche im Folgenden beschrieben werden. Das zeigt sich insbesondere darin, dass keine Aussagen zu dem Ort der Datenverarbeitung selbst getroffen und dieser daher nicht als Optimierungspotential betrachtet wird. In dieser Arbeit dagegen stellt die Verlagerung von Analysen von der Cloud zum Netzwerkrand, d.h. die Verarbeitung der Datenflüsse so nahe wie möglich an der Datenerfassung, einen der wesentlichen Ansätze zur Optimierung dar. Dies geschieht mit besonderen Fokus auf industrielle Systeme. Da auf industriellen Endgeräten die verfügbaren Ressourcen limitiert sind, werden konkrete Möglichkeiten der Optimierung für Ressourcenauslastung, Aufgabenverteilung und den Datenfluss-Applikationen untersucht und evaluiert. Diese sind nicht Bestandteil von DSPS selbst (Liu und Buyya 2020), aber stellen eine relevante Grundlage für die Verlagerung der Datenflussverarbeitung an den Netzwerkrand dar. (Liu und Buyya 2020) bestätigen in ihrer Studie den Forschungsbedarf auf dem Gebiet der Ressourcenallokation und Aufgabenverteilung.

Die ersten DSPSs wurden bereits Ende des 20. Jahrhunderts entwickelt und stellen eine zunehmend relevante Alternative zur Verarbeitung von Datensätzen, der

sogenannten Batch-Analyse, dar (Fragkoulis u. a. 2020). Sie werden stetig weiterentwickelt, sodass man bereits heute die in Abbildung 1.2 dargestellten vier Generationen voneinander abgrenzen kann (Dias de Assunção, da Silva Veith und Buyya 2018). Die ersten beiden Generationen sind noch sehr stark an die bis dahin üblichen klassischen Datenbank-Management-Systeme (engl. Data Base Management System (DBMS)) angelehnt. Sie stellen eine Erweiterung von DBMS für die Anwendung auf Datenflüsse dar, wobei sich die zweite Generation bereits durch die dezentralisierte Ausführung der Datenverarbeitung auszeichnet. Die dritte Generation stellt einen wesentlichen Fortschritt in der Skalierbarkeit dar, welche durch Verarbeitungen in Cluster-Umgebungen erreicht wird. Außerdem kann die Datenverarbeitung als benutzerdefinierte Funktion anstelle von SQL-ähnlichen Abfragen programmiert werden. Eine Vielzahl der weit verbreiteten und heute häufig eingesetzten DSPS, wie Apache Flink[1], Apache Storm[2], Spark Streaming[3] und weitere, gehören dieser Generation an. Detaillierte Beschreibungen sowie Vergleiche sind (Dias de Assunção, da Silva Veith und Buyya 2018; Isah u. a. 2019; Nasiri, Nasehi und Goudarzi 2019) zu entnehmen. Während die Datenverarbeitung der ersten drei Generationen komplett in der Cloud stattfindet und keine Analysen auf den Endgeräten selbst vorsieht, zeichnen sich die aktuellen Entwicklungen durch die Ausführung einzelner Verarbeitungsschritte, insbesondere der Vorverarbeitung, auf den Endgeräten am Netzwerkrand aus. Es ist zu erwarten, dass die Datenflussverarbeitung auf den Endgeräten den Einsatz von datengetriebenen Anwendungen unterstützt, da es zur Erfüllung der geforderten Servicequalität beiträgt (Renart, Diaz-Montes und Parashar 2017). Der kombinierte Einsatz von Edge und Cloud, wie es beispielsweise die Open-Source-Software Apache Edgent[4] ermöglicht, ist somit das wesentliche Merkmal der vierten Generation (Dias de Assunção, da Silva Veith und Buyya 2018). Apache Edgent ist ein Laufzeitsystem, welches die zu übertragende Datenmenge durch eine vorgelagerte Analyse auf dem Endgerät reduziert. Nur als relevant eingestufte Ereignisse werden für eine weitere Analyse, Aktion und/ oder zur Speicherung an das cloudbasierte Hauptsystem übermittelt.

Darüber hinaus ist die in dieser Arbeit beschriebene Entwicklung hin zur vollständigen Verarbeitung von Daten auf den Endgeräten ohne obligatorische Cloud zu erwarten, welche eine fünfte Generation von DSPS zur Folge hätte. Die im Rahmen dieser Arbeit entwickelten Ansätze legen hierfür wichtige Grundlagen. Einerseits werden Applikationen vorgestellt, die auch auf den Endgeräten nahe der Datenerfassung ausführbar sind, andererseits wird die intelligente Allokation der begrenzten Ressourcen der Endgeräte beleuchtet.

Bestehende Arbeiten mit dem Fokus auf diesen Ansatz werden im Folgenden vorgestellt. (Pisani u. a. 2017) präsentieren das Rahmenwerk LMC für die plattformübergreifende Ausführung von Datenflussanalysen auf ressourcenlimitierten Endgeräten und vergleichen es mit Apache Edgent. Die Evaluation anhand von drei

[1] http://flink.apache.org/
[2] https://storm.apache.org/
[3] https://spark.apache.org/streaming
[4] https://incubator.apache.org/projects/edgent.html

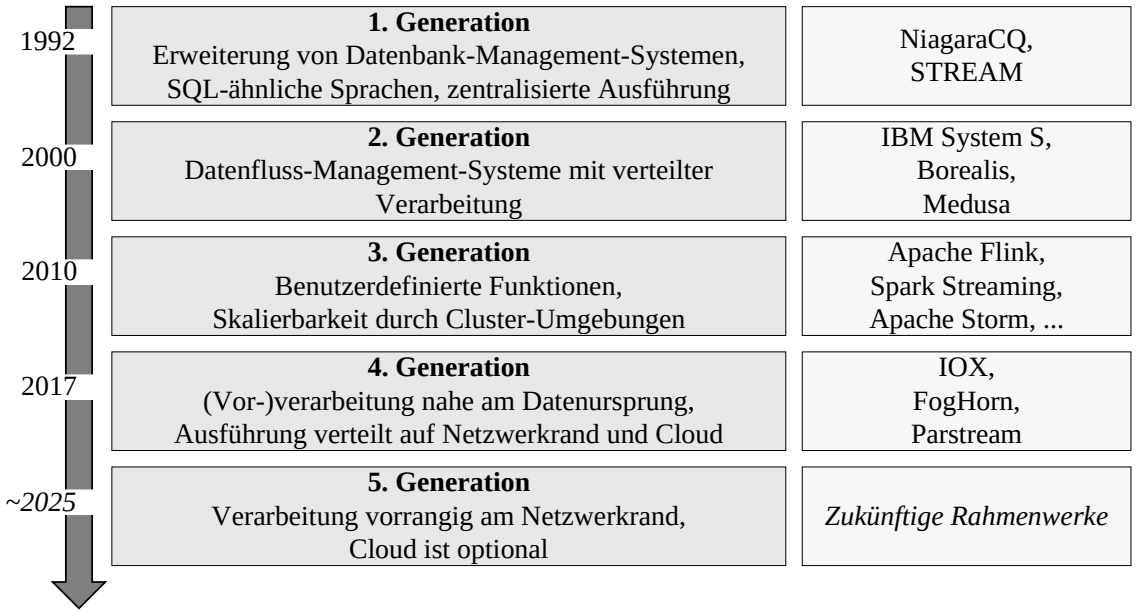


Abbildung 1.2: Generationen von DSPS nach (Dias de Assunção, da Silva Veith und Buyya 2018; Fragkoulis u. a. 2020; Rosenberger, Selig, Ristic u. a. 2023)

einfachen Datenanalysen mittels Filter, schwellwertbasierter Ausreißerkennung und Fast Fourier Transformation, zeigt, dass LMC im Vergleich zu Apache Edgent zu besseren Ergebnissen führt und auf Endgeräten mit geringeren Ressourcen ausgeführt werden kann. Die Experimente finden allerdings nur auf einzelnen Endgeräten statt und es wird keine Aufteilung der Last auf verschiedene Endgeräte innerhalb eines IoT-Netzwerkes betrachtet.

Die Beschränkungen durch die limitierten Ressourcen von IoT-Endgeräten führten zu vielen Fortschritten auf dem Gebiet der mobilen Cloud-Berechnungen, in dem Berechnungen von den mobilen Endgeräten, insbesondere Smartphones, in Clouds verlagert werden. Einen Schritt weiter gehen die Arbeiten von (Hasan, Hossain und Khan 2015; Habak u. a. 2015; Mtibaa, Harras und Fahim 2013), in denen mobile Endgeräte eine lokale Cloud bilden und ihre Rechenkapazitäten gemeinsam in Form von Cluster-Berechnungen einsetzen. Diese Cluster-Umgebungen berücksichtigen jedoch nicht die Anforderungen in industriellen Anlagen und sind nur für die Bearbeitung einzelner Berechnungen, aber nicht für Datenflussverarbeitungen geeignet. Nach (Dautov, Distefano u. a. 2017) mangelt es den modernen DSPS Ansätzen noch an der Unterstützung von Cluster-Berechnungen. (Wang und Peh 2014) untersuchen den Einsatz von mobilen Endgeräten zur verteilten Datenflussverarbeitung, wobei die Anwendung insbesondere auf Smartphones zugeschnitten ist und die Verlässlichkeit der Datenverarbeitung im Fokus steht. Neben der Entwicklung von zwei Kontrollpunkt-Mechanismen wird auch ein Ansatz zur Unterteilung des Netzwerks in kleinere Regionen vorgestellt, innerhalb deren Cluster die Last der Datenverarbeitung verteilt wird. Das Clustern von Smartphones im selben Gebiet ist für Anwendungen, welche auf geographisch verteilten Endgeräten basieren, übertrag-

bar. Beispielsweise wird die Berücksichtigung des Standorts ebenfalls in der Arbeit von (Renart, Diaz-Montes und Parashar 2017) aufgegriffen und, in Kombination mit einer inhaltsbasierten Datenflussverarbeitung, auf das Anwendungsfeld intelligenter Städte angewandt. Der Inhalt der Daten sowie der Standort sind dafür ausschlaggebend, wie und mittels welchen Ressourcen ein Datenfluss verarbeitet wird. Die standortbezogene Verarbeitung ist insbesondere für geographisch weitläufig verteilte Systeme relevant, für industrielle Anlagen ist der Nutzen jedoch eher gering. Einen wesentlichen Fortschritt für Cluster-Berechnungen in IoT-Szenarien stellt die Arbeit von (Dautov, Distefano u. a. 2017) dar. In einer ersten Veröffentlichung stellen die Autoren eine Möglichkeit zur horizontalen Lastverteilung zwischen den Endgeräten in Form von Clustern in einem IoT-Netzwerk vor. Als Middleware zur Datenfluss-Verarbeitung wird die Open-Source-Software Apache Ni-Fi[5] eingesetzt und erweitert. Die Orchestrierung erfolgt hierbei allerdings noch zentralisiert durch einen sogenannten Koordinator-Knoten.

In einer aktuelleren Arbeit (Dautov und Distefano 2020), welche ebenfalls die horizontale Aufgabenverteilung betrifft, wird dieser Nachteil behoben und eine Machbarkeitsstudie vorgestellt, die ebenfalls auf der Cluster-Middleware Apache Ni-Fi basiert, aber keine zentrale Instanz mehr benötigt. Bei dem vorgestellten Rahmenwerk stellt sich allerdings die Problematik, dass vollvermaschte Netzwerke erforderlich sind und dies in der Industrie häufig nicht gegeben ist. Zwei weitere, allerdings bisher nur konzeptionell vorgestellte Ansätze (Dautov und Distefano 2020), werden in (Nastic u. a. 2017) und (Gusev u. a. 2019) beschrieben.

Obwohl DSPS der fünften Generation, also die verteilte Verarbeitung von Datenflüssen nahe der Datenerfassung, eine Lösung für die genannten Probleme wie Risiken der Datensicherheit, Latenzen und Kosten darstellt, existieren bisher nur wenige Arbeiten dazu. Bei keinem der Ansätze findet Maschinelles Lernen (engl. Machine Learning (ML)) Anwendung und der Einsatz im IIoT ist neu.

Das ist darin begründet, dass die Entwicklung von DSPS einer fünften Generation in erster Linie durch fünf Hindernisse erschwert wird:

- Limitierte Rechen- und Kommunikationsressourcen.
- Nutzung vorhandener Infrastruktur statt zusätzlicher Infrastrukturausbau.
- Bedarf an Echtzeitfähigkeit und hohen Verarbeitungsgeschwindigkeiten.
- Dezentrale Systeme mit einer hohen Heterogenität der Teilnehmer.
- Bereits teilweise eingetretene Marktbeherrschung durch die großen, extrem finanzkräftigen Cloud-Anbieter, z.B. Microsoft Azure oder Amazon Web Services.

Den Hindernissen gegenüber stellen sich die Anforderungen und ergeben den Problemraum für die Datenverarbeitung am Netzwerkrand im IIoT:

- Forderung nach ökonomischer Effizienz und Wirtschaftlichkeit.

[5] http://nifi.apache.org/

- Forderung nach Nachhaltigkeit hinsichtlich Energiebedarf, aber auch den Einsatz von Hardware.
- Forderung nach Skalierbarkeit und Erweiterbarkeit.
- Forderung nach Vertrauenswürdigkeit.

1.3 Zielsetzung und Aufbau der Arbeit

Zielsetzung und Forschungsfrage

Diese Arbeit verfolgt das Ziel, Optimierungsbedarfe in industriellen Datenflüssen und deren Verarbeitung zu ermitteln und die Potentiale neuer Technologien für die Lösung der Optimierungsbedarfe zu ermitteln. Dabei liegt besonderer Fokus auf der verteilten Verarbeitung durch die Teilnehmer des IIoT am Netzwerkrand mit dem übergeordneten Ziel, die Entwicklung von DSPS der fünften Generation und den Einsatz neuer, datengetriebener Anwendungen, wie beispielsweise der Zustandsüberwachung, voranzutreiben sowie die zuvor genannten bestehenden Hindernisse zu überwinden.

Abbildung 1.3 bietet einen Überblick über die Motivation und Ausgangssituation sowie der Zielstellung. Aus dieser wird die Forschungsfrage abgeleitet, welche im Rahmen dieser Arbeit beantwortet wird:

Ist die verteilte Verarbeitung von Datenflüssen auf den Endgeräten im industriellen Internet der Dinge unter Berücksichtigung der Wirtschaftlichkeit realisierbar und wie kann eine mögliche Umsetzung gestaltet sein?

Unter Berücksichtigung der genannten Hindernisse werden drei Ansätze zur Lösung unterschiedlicher Aspekte der Problemstellung genauer untersucht:

- Effiziente Algorithmen zur Anwendung auf begrenzten Endgeräten.
- Schaffung einer Vertrauensbasis in dezentralen, heterogenen Netzwerken.
- Optimierte Nutzung der limitierten vorhandenen Ressourcen.

Die Beantwortung der Forschungsfrage legt wichtige Grundlagen für die weitere Entwicklung von DSPS der 5. Generation und somit für die Verarbeitung von Datenflüssen nahe der Datenerfassung. In diesem Zusammenhang wird der Stand der Technik um die folgenden Aspekte erweitert, welche den wissenschaftlichen Beitrag dieser Arbeit darstellen:

- Applikationen zur Datenflussverarbeitung auf industriellen Endgeräten:
  - Entwicklung eines Verfahrens zur Anomalieerkennung in industriellen Datenflüssen.
  - Weiterentwicklung und Evaluation von Methoden des Maschinellen Lernens zur Kompression von Datenflüssen.

  - Vorstellung einer Methode zur Effizienzsteigerung durch den kombinierten Einsatz von Modellen.
- Evaluation der Anwendbarkeit von Distributed Ledger Technology (DLT) im IIoT anhand von zwei relevanten Anwendungsfällen zur dezentralen Absicherung von Daten und Informationen.
- Entwicklung von zwei interagierenden Multi-Agenten-Systemen (engl. Multi-Agent-System (MAS)) basierend auf Bestärkendem Lernen (engl. Reinforcement Learning (RL)) für die Optimierung der Ressourcenauslastung im IIoT.

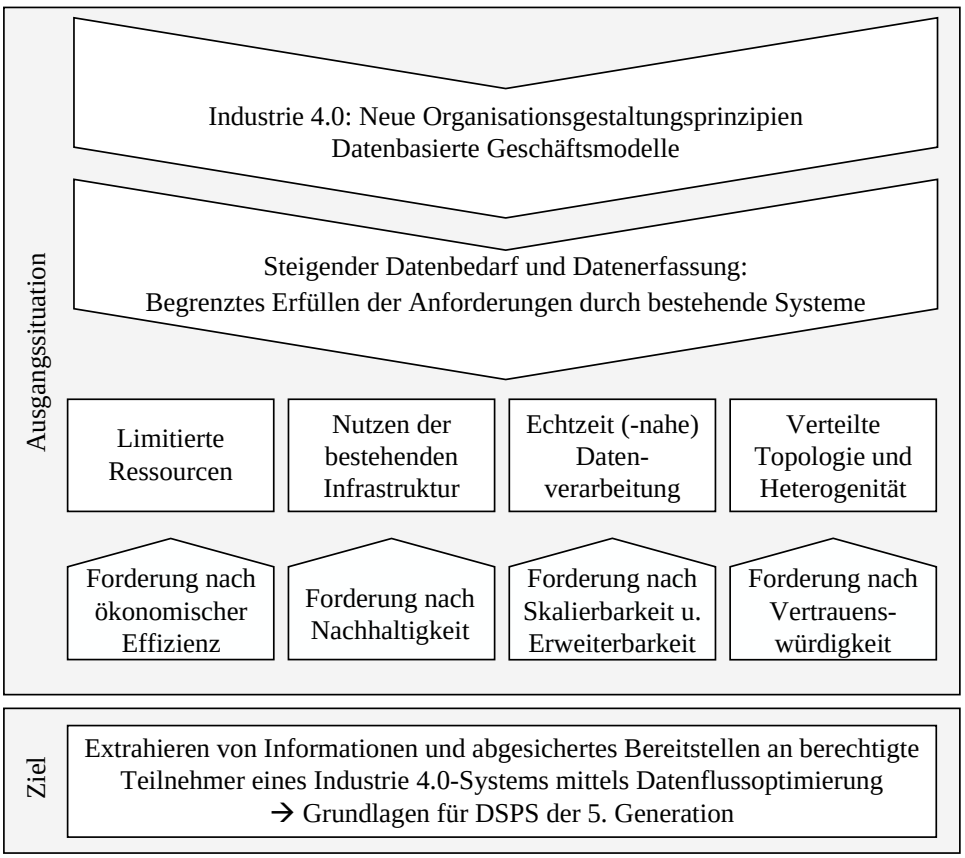


Abbildung 1.3: Zusammenfassung der Ausgangssituation und Ziel der Arbeit

Vorgehen und Aufbau der Arbeit

Bei der Ausarbeitung wird wie folgt vorgegangen: Der Vorstellung der Ausgangssituation, also der Motivation und des aktuellen Stands der Technik zu DSPS, sowie der davon abgeleiteten Zielsetzung für die Arbeit (siehe Kapitel 1), folgt zunächst eine Einführung in die I4.0 und die Klärung von Begrifflichkeiten im Kontext von Daten und datenflussverarbeitenden Systemen (siehe Abschnitte 2.1-2.3). Anschließend werden Technologien untersucht, die Potentiale zur Optimierung der Datenflüsse und deren Verarbeitung besitzen (siehe Abschnitte 2.4-2.7). Diese sind Formen des ML, Datenkompression, Anomalieerkennung und DLT, insbesondere der DLT IOTA. Das methodische Vorgehen ist an das übergreifende Vorgehen Cross-Industry Standard Process For Data Mining (CRISP-DM) angelehnt (siehe Abschnitt 3.1), sodass zunächst die Zielsetzung konkretisiert und spezifische Forschungsfragen abgeleitet werden. Außerdem erfolgt eine Analyse der Randbedingungen und Anforderungen (siehe Kapitel 3). Darauf folgt die Beschreibung der verfolgten Optimierungsansätze.

Der erste Ansatz zielt auf mengenoptimierte Datenflüsse ab und beruht auf Datenreduktion und Informationsextraktion (siehe Kapitel 4). Dazu werden Methoden zur Datenkompression und Anomalieerkennung evaluiert. Aufgrund der Ähnlichkeit der beiden Ansätze wird außerdem der kombinierte Modelleinsatz vorgeschlagen und untersucht.

Der zweite Ansatz zielt auf vertrauenswürdige Datenflüsse ab und betrachtet die Schaffung von Vertrauen zwischen heterogenen Endgeräten im IIoT (siehe Kapitel 5). Hierfür wird der Einsatz der DLT IOTA für die Absicherung von Daten anhand von zwei ermittelten Anwendungsfällen evaluiert. Zum einen wird die Möglichkeit der unveränderlichen Aufzeichnung von Historien beschrieben, zum anderen wird eine Verwaltung von Zugriffsrechten mittels digitaler Verträge (engl. Smart Contracts (SCs)) erläutert.

Der dritte Ansatz zielt auf die ressourcenoptimierte Datenflussverarbeitung ab und ermöglicht somit den Einsatz der vorangegangenen Methoden im IIoT. Hierfür werden zwei interagierende MAS entwickelt, die die verfügbaren limitierten Rechen- und Kommunikationsressourcen für zusätzliche Rechenaufträge wie Datenanalysen oder DLT Knoten allokieren (siehe Kapitel 6).

Abschließend werden die vorgestellten Ansätze als Gesamtsystem in Interaktion mit einer simulierten industriellen Anlage betrachtet, die Inhalte zusammengefasst und die Beantwortung der Forschungsfrage überprüft. Es wird der wissenschaftliche Beitrag herausgestellt und ein Ausblick über den weiteren Forschungsbedarf gegeben (siehe Kapitel 7).

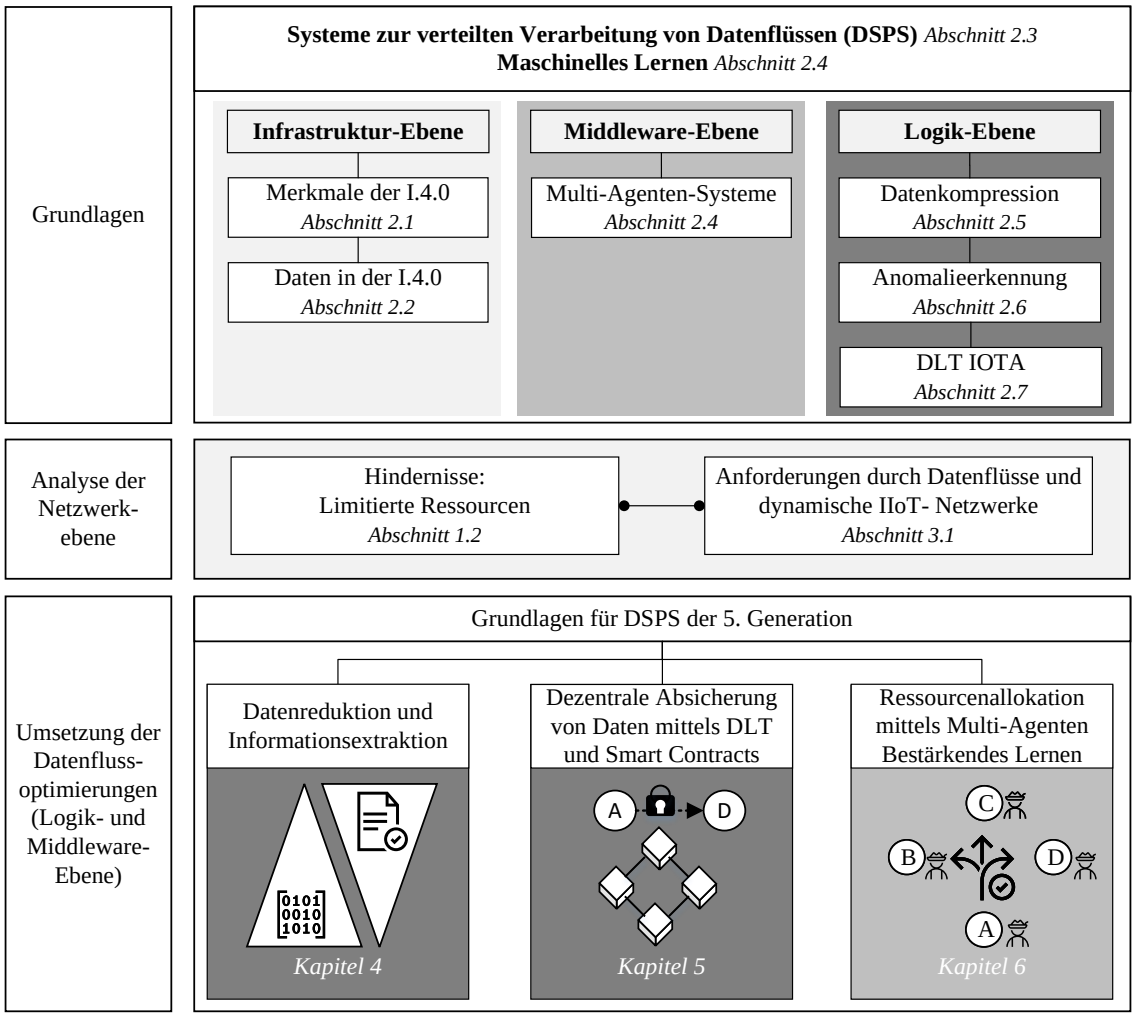


Abbildung 1.4: Aufbau der Arbeit

KAPITEL 2

Grundlagen

Dieses Kapitel gibt eine Einführung in die relevanten Grundlagen für das Verständnis der nachfolgend beschriebenen Optimierungsansätze. Es beleuchtet die Themengebiete Industrie 4.0, Daten sowie Datenflussverarbeitung, Maschinelles Lernen, Datenkompression, Anomalieerkennung und Distributed Ledger Technologie.

2.1 Merkmale der Industrie 4.0

Die Geschichte der Industrie ist von vier industriellen Revolutionen geprägt. Die erste Revolution löste der Einsatz von Wasser- und Dampfkraft in der Industrie in der zweiten Hälfte des 18. Jahrhunderts aus. Mit dem Einsatz von Elektrizität folgte ein Jahrhundert später die zweite Revolution, während die dritte Revolution ab Mitte des 20. Jahrhunderts von der Automatisierung geprägt war. Mit dem Begriff *Industrie 4.0* wurde im Jahr 2011 in Deutschland erstmals die vierte industrielle Revolution betitelt, welche mit der zunehmenden Digitalisierung einhergeht und das Ziel verfolgt, ein höheres Maß an betrieblicher Effizienz und Produktivität sowie einen höheren Automatisierungsgrad zu erreichen (Thames und Schaefer 2016). In der „Umsetzungsstrategie Industrie 4.0“ (BITKOM e. V., VDMA e. V. und ZVEI e. V. 2015) wird der Begriff als „neue Stufe der Organisation und Steuerung der gesamten Wertschöpfungskette über den Lebenszyklus von Produkten“ definiert. Die Ausgangsbasis sind heterogene Daten sowie die Integration von Wissen in die Prozesse (Lu 2017). Mittels der in Abschnitt 1.1 genannten Prinzipien zur Neugestaltung der Organisation durch die I4.0 können die von (BITKOM e. V., VDMA e. V. und ZVEI e. V. 2015) genannten Voraussetzungen, d. h. die Bereitstellung aller für die jeweiligen Teilnehmer relevanten Informationen sowie das Treffen darauf basierender Entscheidungen für einen optimalen Wertschöpfungsfluss, erfüllt werden.

- Die *Vernetzung* ermöglicht es den Teilnehmern Informationen auszutauschen, was die Basis für die Zusammenarbeit zur gemeinsamen Zielerreichung darstellt (Giusto u. a. 2010). In diesem Zusammenhang ist auch der steigende Bedarf an Cybersicherheit zu nennen, da die Manipulation von Informationen zu schwerwiegenden Fehlentscheidungen führen kann.
- Die *Informationstransparenz* umfasst die Aspekte der Datenanalyse sowie der Bereitstellung von Informationen über die gesamte Wertschöpfungskette. Die Vision des Prinzips beschreibt somit eine organisationsübergreifende Transparenz, woraus sich Klärungsbedarf hinsichtlich Datenhoheit und Schutz von geistigem Eigentum ergeben (Ahlfeld u. a. 2022). Es verstärkt sich die Betrachtung von Daten und Informationen als neues, wertvolles Gut. Die Echtzeitfähigkeit beziehungsweise nahe Echtzeitfähigkeit ist eine zusätzliche Anforderung, die mit der Informationstransparenz einhergeht.
- Das Prinzip der *Dezentralen Entscheidungen* basiert auf richtigen Informationen und setzt die zwei zuvor genannten Prinzipien, Vernetzung und Informationstransparenz, voraus. Es wird durch cyber-physische Systeme (engl. Cyber-Physical Systems (CPS)) ermöglicht. CPS können ihre Umwelt selbstständig überwachen und aus den erhaltenen Informationen, z.B. aus Sensorsignalen, Entscheidungen ableiten und die physikalische Umwelt somit autonom steuern (Lee 2008).
- Die *Technische Assistenz* für die menschlichen Mitarbeiter in der I4.0 wird unterteilt in physische und virtuelle Assistenz (Hermann, Pentek und Otto 2016). Während mit physischer Unterstützung beispielsweise die Übertragung von körperlich schweren Arbeiten an Roboter gemeint ist, hat die virtuelle Assistenz zum Ziel, alle relevanten Informationen, die zu der Bearbeitung einer Aufgabe und dem Treffen von Entscheidungen durch den Mitarbeiter erforderlich sind, bereitzustellen.

Die drei Hauptbestandteile der I4.0 sind CPS, intelligente Fabriken, sogenannte Smart Factories, sowie das IoT, im industriellen Kontext auch IIoT genannt (Hermann, Pentek und Otto 2016), wobei auf letztgenanntem im Kontext der Datenflussoptimierung besonderer Fokus liegt.

Industrielles Internet der Dinge

Das IIoT erlaubt Dingen, beispielsweise Sensoren oder Aktoren, durch eindeutige Adressierungsschemata miteinander zu interagieren und kooperieren, um ein gemeinsames Ziel zu erreichen (Giusto u. a. 2010). Das IIoT unterscheidet sich vom IoT im Wesentlichen durch das industrielle Umfeld. Daraus lässt sich ableiten, dass sich die Art der Objekte, z.B. Steuergeräte anstelle von Handys, und eingesetzte Informationstechnologien, aber auch die Anforderungen und die Zielstellung vom IoT unterscheiden. Nach (Boyes u. a. 2018) spielen im IIoT insbesondere Prozess-,

Produkt- und/oder Dienstleistungsinformationen eine wesentliche Rolle. Die Erfassung, Verarbeitung und der Austausch sollen dabei in Echtzeit geschehen und den Gesamtproduktionswert erhöhen.

Die bisherige Struktur der Hierarchie einer Fabrik kann durch das Modell der Automatisierungspyramide nach (DIN EN 62264 2014), die links in Abbildung 2.1 dargestellt ist, gut wiedergegeben werden. Die Struktur ist hierarchisch und jede Ebene verfügt über eigene IT-Systeme, die nur bedingt miteinander interagieren können (Vogel-Heuser 2014). Das führt häufig zu Dateninkonsistenz und hohen manuellen Aufwänden (Hilbrich 2008; Meudt, Pohl und Metternich 2017).

Mit Blick auf die Veränderungen im Kontext von Datenflüssen zeigt sich auf der rechten Seite in Abbildung 2.1, dass sich die durchgängige, lineare Kommunikation der Automatisierungspyramide auflöst. Stattdessen wird eine ebenenübergreifende, zunehmend vernetzte Kommunikation entlang der Lebenszyklus- und Hierarchieebenen angestrebt. Dies spiegelt sich im IIoT mit der Topologie eines vermaschten Netzes wider. Ein vermaschtes Netz zeichnet sich durch seinen dezentralen Aufbau und dadurch erhöhte Ausfallsicherheit aus. Ein Knoten kann dabei einen oder mehrere Nachbarn haben und die Struktur ist nicht vorgegeben.

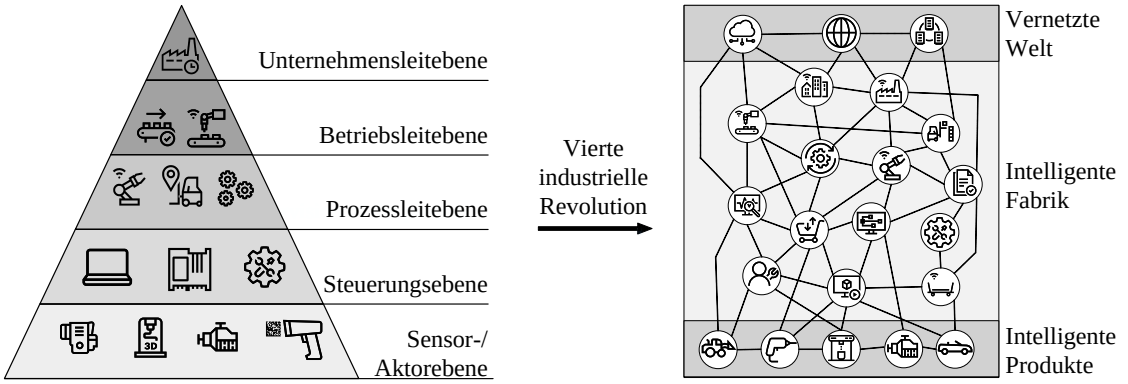


Abbildung 2.1: Übergang der Hierarchien von Industrie 3.0 zu Industrie 4.0 (Salari 2018; Rosenberger, Selig, Ristic u. a. 2023)

Um die I4.0 ganzheitlich unter Berücksichtigung des Produktlebenszyklus, den Schichten der Informations- und Kommunikationstechnologie sowie der Hierarchieebenen, abzubilden, stellt das in (DIN SPEC 91345 2016) beschriebene Referenzarchitekturmodell 4.0 (RAMI 4.0) eine ergänzende Architekturbeschreibung dar. Bei dem RAMI 4.0 handelt es sich, wie in Abbildung 2.2 dargestellt, um ein kubisches Modell. In (Huber 2016) werden die drei Achsen erläutert. Die senkrechte Achse beschreibt die Architektur durch die sogenannten Geschäftsprozessebenen, denen die einzelnen Bestandteile einer Fabrik, von der Hardware bis zum Geschäftsprozess, zugeordnet werden. Die zweite Achse, die Lebenszyklusebenen, beschreibt den Lebenszyklus und Wertestrom. Sie entspricht einer Art zeitlichen Achse für das Produkt und die Produktdaten. Die dritte Achse beschreibt die Hierarchie-Ebenen.

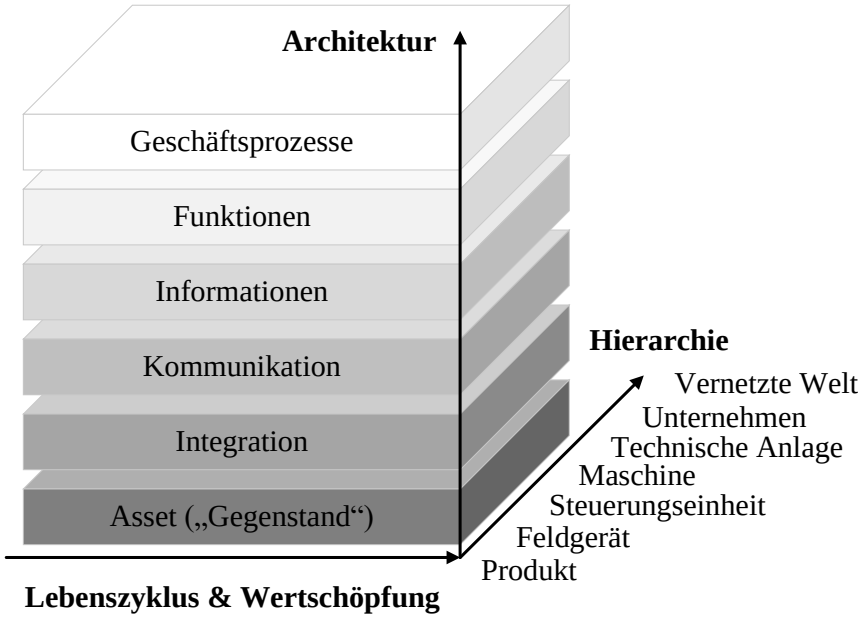


Abbildung 2.2: Referenzarchitekturmodell 4.0 nach (BITKOM e. V., VDMA e. V. und ZVEI e. V. 2015)

Diese Achse ist inhaltlich eng an die Automatisierungspyramide angelehnt und integriert die Norm IEC 62264. Sie wird allerdings durch ihre oberste Ebene, der vernetzten Welt, erweitert, während die Automatisierungspyramide auf der Unternehmensebene endet. Sie stellt eine Bestandsaufnahme aller Ebenen einer Fabrik dar und beschreibt den Verlauf von Daten innerhalb einer Etappe des Lebenszyklus, z.B. innerhalb einer Produktionsstätte. Nach (Huber 2016) ist der Datenaustausch im RAMI 4.0 insbesondere innerhalb einer Schicht der senkrechten Achse möglich und auf zwei benachbarte Schichten ausgeweitet. Ein übergreifender Austausch über mehrere vertikale Schichten hinweg ist somit nicht vorgesehen. Diese beschriebenen Veränderungen durch die vierte industrielle Revolution in den Darstellungen der Hierarchien ist in Abbildung 2.1 nach (Salari 2018) hervorgehoben. Während links die klassische Automatisierungspyramide abgebildet ist, stellt die Grafik rechts eine neuartige Form der Hierarchie in der I4.0 dar. Die funktionale Einteilung der Ebenen der Automatisierungspyramide bleibt weiterhin gültig, allerdings spielt sie durch die ausgeprägte Vernetzung und Dezentralisierung in der I4.0 eine untergeordnete Rolle.

Eng im Kontext zum RAMI 4.0 steht die Erstellung von I4.0-Komponenten. Die I4.0-Komponenten bestehen dabei aus Assets, d. h. aus physischen Gegenständen oder auch digitalen Komponenten (bspw. konzeptionelle Artefakte) (Diedrich und Riedl 2016), und der Verwaltungsschale (engl. Asset Administration Shell (AAS)) (IEC 63278-1 2021), siehe Abbildung 2.3. Die Assets mit den Eigenschaften *Daten*, *Funktionen* und *Kommunikationsfähigkeit* (Huber 2016), sind somit durch die zusätzliche AAS im IIoT ansprechbar. Die Integration von Assets in das IIoT ist in der I4.0 essentiell. Sie wird über die AAS erzielt, die vergleichbar mit einem digita-

len Typenschild ist und relevante Informationen über Software und Hardware der Assets beinhaltet (Hoffmeister u. a. 2021).

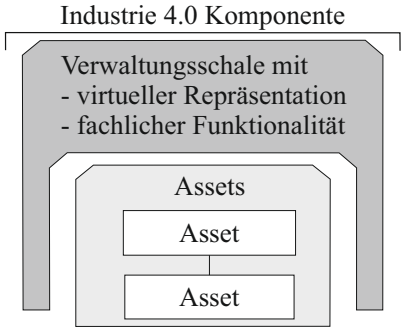


Abbildung 2.3: I4.0 Komponente mit AAS (nach Dönicke u. a. 2017)

2.2 Daten in der Industrie 4.0

Während die Relevanz von Daten für die I4.0 bereits im vorangegangenen Abschnitt aufgezeigt wird, dient dieser Abschnitt der näheren Beschreibung von Daten in der I4.0. In Anlehnung an (BITKOM e. V., VDMA e. V. und ZVEI e. V. 2015) werden industrielle Datenflüsse in horizontale und vertikale Datenflüsse unterteilt. Als horizontale Datenflüsse werden die Daten entlang des Produktlebenszyklus, beispielsweise Produkt- und Entwicklungsdaten, Auftragsdaten, Fertigungsdaten sowie Zeichnungen, kategorisiert. Die Prozessdaten entlang der Achse der hierarchischen Ebenen des RAMI 4.0 repräsentieren dagegen die vertikalen Datenflüsse, die im Kontext der Datenflussoptimierung in dieser Arbeit betrachtet werden. Besondere Herausforderungen stellen sich durch die hohen Senderaten, Echtzeitanforderungen und, bedingt durch die zunehmende Anzahl an Sensoren, das große Datenvolumen.

Der Großteil der Daten in industriellen Anlagen stammt aus Sensorsignalen, auf denen der Fokus dieser Arbeit liegt. In der digitalen Domäne stellen sie fortlaufende Aneinanderreihungen von Daten diskret sequentieller Natur dar. Die Signaldaten können binär, diskret oder kontinuierlich sein und in Abhängigkeit vom Prozess auch periodisch vorkommen. Sensorsignale werden heutzutage meist als Integerwerte übermittelt. Zukünftig ist für höherwertige Sensoren auch die Übermittlung von Fließkommazahlen zu erwarten. Mit dem zunehmenden Einsatz von Kamerasystemen steigt zudem der Austausch von Bild- bzw. Videodaten in industriellen Systemen. Textdaten sind dagegen im Kontext der vertikalen Datenflüsse selten anzutreffen. In der Regel liegen Sensordaten als Zeitreihen vor, da ein Zustand mit einer bestimmten Abtastrate vom Sensor erfasst und als Signal weitergeleitet wird. Nach (Chiarot und Silvestri 2023) sind Zeitreihen eine Datenmenge, deren Elemente in der Reihenfolge ihrer Zeitstempel sortiert sind. Sie werden unterteilt in die eindimensionalen Zeitreihen (engl. Univariate Time Series (UTS)), in denen genau ein Datenpunkt je Zeitstempel vorliegt, und den mehrdimensionalen Zeitreihen (engl. Multivariate Time Series (MTS)), deren Elemente zu einem Zeitstempel aus den

Datenpunkten mehrerer Zeitreihen bestehen. Des Weiteren gibt es auch Daten, die eventbasiert bei Erfüllung einer vorgegebenen Bedingung auftreten und daher nicht sequentieller Natur sind. Beispiele hierfür sind erkannte Anomalien oder Fehlermeldungen. Daten können von strukturierter oder unstrukturierter Natur sein. Strukturierte Daten lassen sich tabellarisch, beispielsweise in einer Datenbank, abbilden. Beispiele für unstrukturierte Daten sind dagegen Texte, Bilder oder Tonaufnahmen. Während es für den menschlichen Betrachter nur geringfügig Unterschiede macht, sind unstrukturierte Daten maschinell deutlich komplizierter auszulesen und zu interpretieren. Die in dieser Arbeit betrachteten Sensorsignale, also Zeitreihendaten, werden den strukturierten Daten zugeordnet. Neben den Signalwerten selbst sind auch die Metadaten relevant für die Interpretation und Weiterverarbeitung. Metadaten können beispielsweise Minimum- bzw. Maximumwerte angeben oder definieren, in welcher Einheit das Signal übertragen wird. Weiterhin können Daten im Zeit-, Zeit-Frequenz- oder Frequenzbereich betrachtet werden. Dabei ist der Informationsgehalt ähnlich, allerdings erlaubt die Änderung der Darstellung den Einsatz unterschiedlicher Synthese- und Analysemethoden und kann dem menschlichen Betrachter die Interpretation und Extraktion von Informationen erleichtern. Ursprünglich liegen Sensorsignale immer im Zeitbereich vor. Für die anderen Darstellungen ist zunächst eine Datenvorverarbeitung, beispielsweise Fourier-Transformation, erforderlich. Im Rahmen dieser Arbeit wird aufgrund der Echtzeitanforderungen weitestgehend auf Vorverarbeitungsschritte verzichtet.

Anhand der in Abbildung 2.4 dargestellten Pyramide werden die Zusammenhänge zwischen Daten, Information, Wissen und Erkenntnis (engl. Data, Information, Knowledge, Wisdom (DIKW)) im industriellen Umfeld kurz erläutert.

- Daten sind lediglich kodierte Werte und haben für sich stehend noch keine Aussagekraft.
- Werden Daten in einen Kontext gesetzt, werden sie als Informationen bezeichnet und sind interpretierbar.
- Wissen beinhaltet Regeln, wie Informationen zu verstehen sind und wie mit ihnen umzugehen ist. Wissen kann explizit vorliegen, beispielsweise in Form von explizit formulierten Regeln, oder implizit, das heißt in Form von Daten, die mit diesen Regeln übereinstimmen. Dementsprechend ist dasselbe Wissen enthalten, nur müssen die Daten erst analysiert werden, um die für den Menschen verständlichere Darstellung zu erhalten. Explizites Wissen ist die Grundlage für klassische Programmierung, wogegen implizites Wissen die Basis für ML ist, auf welches in Abschnitt 2.5 näher eingegangen wird.
- Der Begriff „Erkenntnis“ bezeichnet dagegen das Verständnis über die Ursache, also warum die jeweiligen Daten aufgetreten sind.

Die Zusammenhänge der Begriffe DIKW werden am konkreten Beispiel der Fehlerdiagnose erläutert.
Beispiel: Ein Sensor misst den Öldruck und generiert dadurch Daten. Diese Daten

im Kontext eines weiteren Sensorsignals, z. B. der Pumpendrehzahl, stellt eine Information dar. Die Interpretation, dass der Öldruck zu gering ist für eine bestimmte Drehzahl, stellt Wissen dar und führt zu der Kennzeichnung als Anomalie. Die Beurteilung des Wissens führt somit zu einer Erkenntnis. In diesem Zusammenhang ist sie das Verständnis über die Ursache der Anomalie, beispielsweise ein defekter Sensor, und wird als die eigentliche Fehlerdiagnose bezeichnet.

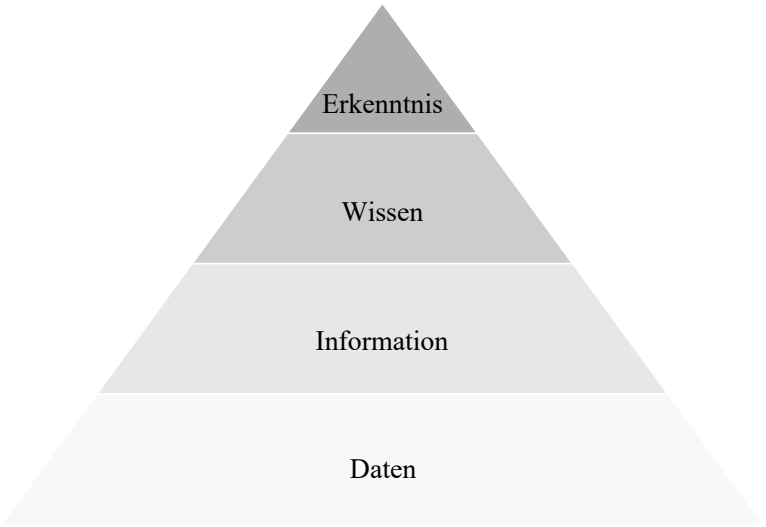


Abbildung 2.4: Darstellung der Daten, Information, Wissen, Verständnis-Pyramide

Aus dem vorangegangenen Beispiel und der Darstellung der DIKW-Pyramide folgt, dass die Menge der relevanten Informationen, aus denen wiederum Wissen und Erkenntnis generiert werden kann, im Allgemeinen im Vergleich zur Rohdatenmenge deutlich reduziert ist. Das Erfassen und Speichern der Rohdaten führt zu enormen Datenmengen, welche durch den Begriff *Big Data* zum Ausdruck gebracht werden. Es folgt eine Zusammenfassung der Merkmale von Big Data sowie deren Bedeutung für Datenverarbeitung in der Industrie. Anschließend wird Big Data von Datenflüssen abgegrenzt. Die (Europäische Wirtschaftsforschung GmbH 2015) definiert Big Data als „die systematische und zeitnahe Auswertung großer unternehmensinterner und externer Datenmengen zur strategischen Unterstützung des Geschäftsbetriebs“. Big Data zeichnet sich durch fünf Merkmale, kurz „5 Vs“, aus.

1. Volumen (engl. Volume): Das Volumen beschreibt die große Menge der Daten.
2. Geschwindigkeit (engl. Velocity): Mit Geschwindigkeit ist die schnelle Erfassung von Daten und Änderung von Werten gemeint.
3. Vielfalt (engl. Variety): Die Vielfalt ist insbesondere auf den Datentyp (un-, semi-, strukturierte Daten) sowie Herkunft der Daten bezogen.
4. Wahrheitsgehalt (engl. Veracity): Der Wahrheitsgehalt bezeichnet die Unsicherheit, d. h. in diesem Kontext die Qualität, also inwieweit die Daten fehlerfrei, vollständig und konsistent sind.

5. Wert (engl. Value): Der Wert, der in den Informationen der Datenmenge beinhaltet ist und durch Weiterverarbeitung der Daten erzielt wird, motiviert Big Data. Er bezeichnet die Erwartung, Vorteile durch die Analyse zu erzielen.

Die Erfassung von Big Data an sich ist daher nur ein Mittel zum Zweck, um aus den Daten interessierende Informationen extrahieren zu können. Dabei ist Big Data und insbesondere die Analyse von Big Data häufig mit großen Nachteilen, wie den bereits in Abschnitt 1.1 beschriebenen Aspekten wie Latenzen, Sicherheit und Kosten, verbunden. Anstelle von Big Data ist in dieser Arbeit deshalb die Optimierung der Analyse von Datenflüssen - lokal und zeitlich nahe der Datenerfassung - das Ziel.

2.3 Datenfluss-Systeme

Der generelle Aufbau von Datenfluss-Systemen ist in Abbildung 2.5 dargestellt. Er umfasst nach (Liu und Buyya 2020) drei Schichten. Die unterste ist die Infrastruktur-Schicht. Sie umfasst die Datenerfassung, die Daten bzw. Datenflüsse sowie die Hardware, welche Ressourcen zur Verfügung stellt. Die Ressourcen sind in der Regel verteilt, aber miteinander vernetzt. Bisher wurden für die Datenflussverarbeitung meist Cloud-Ressourcen benötigt. Mit den aktuellen Entwicklungen (siehe Kapitel 1) steigt auch die Nutzung der Ressourcen am Netzwerkrand, was den Fokus der vorliegenden Arbeit auf Edge Computing begründet. Die mittlere Schicht ist die Middleware-Schicht. Ihr sind unter anderem die DSPS zugeordnet, welche sich in den vergangenen Jahrzehnten zunehmend etablierten. Sie ermöglichen die verteilte Verarbeitung von Datenflüssen in Echtzeit bzw. echtzeitnahe (Li, Xu u. a. 2018). Die Kernaufgaben von DSPS sind in Abbildung 2.5 stichpunktartig zusammengefasst. Aber auch weitere Funktionalitäten wie die Ressourcenallokation oder Aufgabenplanung, die nicht zwangsläufig von DSPS abgedeckt werden, sind der Middleware-Schicht zuzuordnen. Die oberste Schicht ist die Logik-Schicht, welche die Datenfluss-Applikationen, auch Operatoren genannt, umfasst. In dieser Schicht findet die eigentliche Verarbeitung der Datenflüsse statt.

Eine weitere Unterteilung wird nach der Verarbeitungsmethode vorgenommen, typischerweise entweder eine tupelweise Verarbeitung oder die Verarbeitung von Micro-Sequenzen (Dias de Assunção, da Silva Veith und Buyya 2018).

Nach (Liu und Buyya 2020) ist die Container-basierte Bereitstellung der Operatoren, z. B. als Docker-Container, ein offenes Forschungsgebiet und von großer Bedeutung. Diese These wird im Rahmen dieser Arbeit berücksichtigt und die Operatoren daher in Form von Snaps (*Snaps in Ubuntu Core* 2022), einer containerähnlichen Umgebung, zur Verfügung gestellt. Bei der Betrachtung der Infrastruktur-Schicht wird zwischen der horizontalen und vertikalen Verarbeitung unterschieden (Dautov, Distefano u. a. 2017). Die horizontale Verarbeitung entspricht der Verlagerung von Rechenlast auf Servern und ist in DSPS der ersten drei Generationen anzutreffen (siehe Abbildung 1.2). Die vertikale Verarbeitung gewinnt dagegen durch die aktuelle Entwicklung, der Einbindung von IoT Endgeräten, zunehmend an Relevanz und ist Inhalt dieser Arbeit.

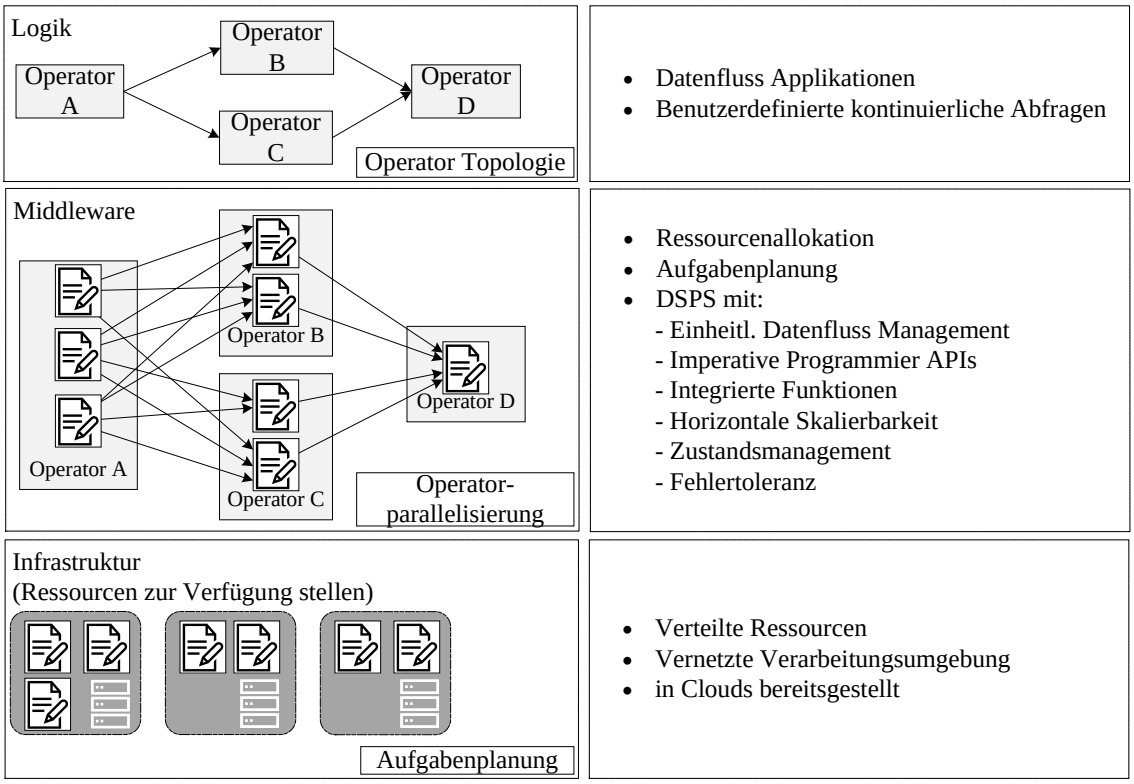


Abbildung 2.5: Allgemeiner Aufbau von Datenfluss-Systemen (nach Liu und Buyya 2020)

2.4 Maschinelles Lernen

2.4.1 Definition und Einteilung der Verfahren

ML ist ein Teilgebiet der Künstlichen Intelligenz, das auf mathematischen Modellen, insbesondere der Statistik, basiert. ML wird als ein Forschungsgebiet definiert, in dem Computern die Fähigkeit verliehen wird, nützliche Aufgaben zu bearbeiten ohne Bedarf einer expliziten Programmierung (Burkov 2019). Eine weitere Definition nach (Mitchell 1997) beschreibt ML als die Verbesserung der Leistung in Bezug auf eine konkrete Aufgabe durch Erfahrung. Im Gegensatz zur klassischen Programmierung ist daher kein explizites Wissen in Form einer Logik erforderlich. Stattdessen wird dem Algorithmus implizites Wissen in Form von Daten für die Bearbeitung einer Aufgabe zur Verfügung gestellt mit dem Ziel der Modellbildung. Im Allgemeinen wird dem Computer ein Optimierungsziel Y vorgegeben, welches iterativ angenähert wird $f : X \rightarrow Y$, um daraus die optimalen Modellparameter zu erlernen. Während die Modellparameter durch die ML-Algorithmen selbst auf Basis der zur Verfügung gestellten Daten erlernt werden, sind zusätzliche für das Modell relevante, übergeordnete Parameter, die sogenannten Hyperparameter wie z.B. die Schrittweite für den Grandientenabstieg, durch den Anwender abzustimmen. Sie haben großen Einfluss auf die Qualität der Ergebnisse, sodass die Hyperparameteroptimierung, d.h. automatisierte Verfahren zur experimentellen Suche nach optimalen Hyperparameterwerten, ein eigenes Forschungsgebiet darstellt. ML lässt sich in überwachtes, unüberwachtes und bestärkendes Lernen einteilen.

Überwachtes Lernen

Überwachtes Lernen wird zur bestätigenden Datenanalyse eingesetzt und setzt für das Training gekennzeichnete Daten voraus. Das bedeutet, die Eingabedaten für das Training sind mit einer oder mehreren Kennzeichnungen versehen, die bestimmte Eigenschaften identifizieren. Somit stehen für das Training sowohl die Eingabewerte als auch die korrekten Ausgabewerte zur Verfügung, sodass der Fehlerabgleich möglich ist. Vertreter dieser Klasse sind beispielsweise Regression und Klassifikation. Abbildung 2.6 stellt schematisch das Vorgehen des überwachten Lernens anhand eines Beispiels für die Klassifikation von Bauteilen dar.

Eine der größten Herausforderungen des überwachten Lernens ist die Kennzeichnung der Daten, da sie mit einem großen zeitlichen, finanziellen und häufig auch manuellem Aufwand verbunden ist. In dieser Arbeit werden daher Algorithmen des unüberwachten Lernens bevorzugt.

Unüberwachtes Lernen

Unüberwachtes Lernen wird vermehrt in der explorativen Datenanalyse eingesetzt und benötigt keine gekennzeichneten Daten. Die Zusammenhänge zwischen den Daten sind von dem Modell selbst zu erkennen, welches dementsprechend nur intrinsische Informationen nutzt. Daher sind diese Verfahren besonders für das Erken-

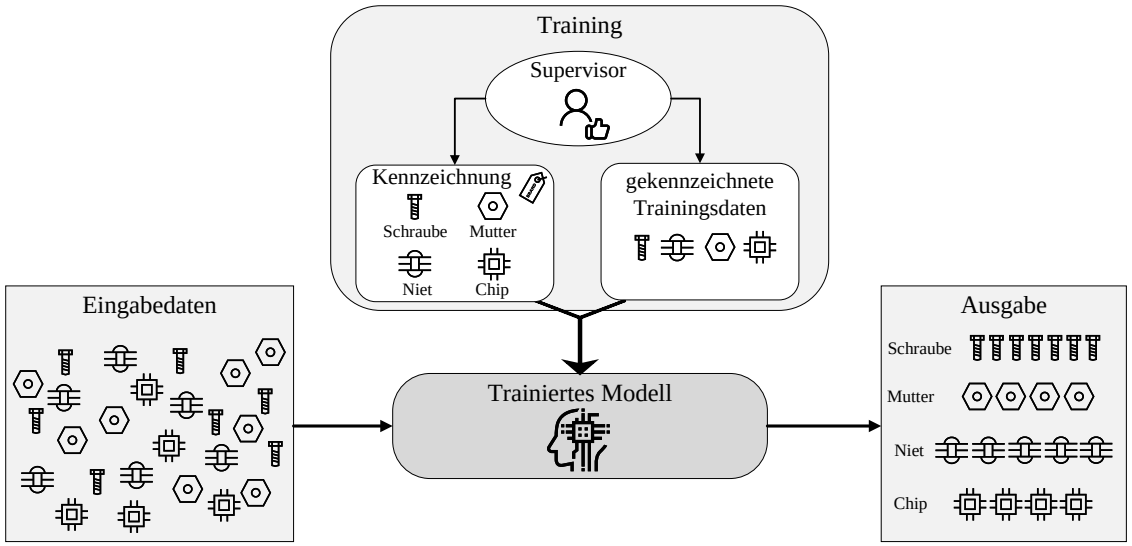


Abbildung 2.6: Allgemeines Vorgehen beim überwachten Lernen

nen von Mustern innerhalb der Daten geeignet, beispielsweise durch Clustering-Verfahren. Abbildung 2.7 abstrahiert die Vorgänge des unüberwachten Lernens anhand eines Beispiels für das Clustering. Die Bauteile werden dabei, im Gegensatz zur Klassifikation, nicht als solche erkannt, aber sie werden vom Algorithmus aufgrund ihrer Ähnlichkeit einem Cluster zugeordnet.

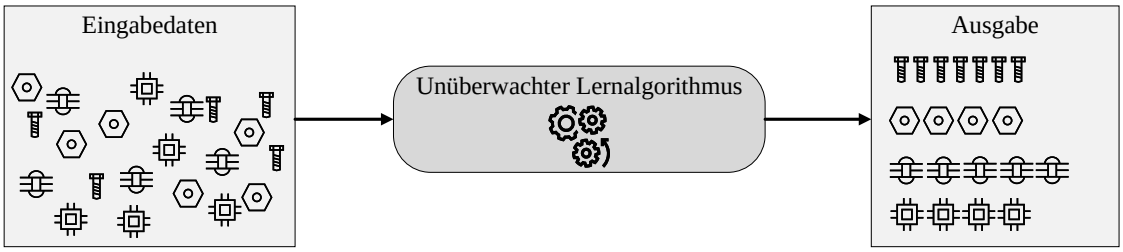
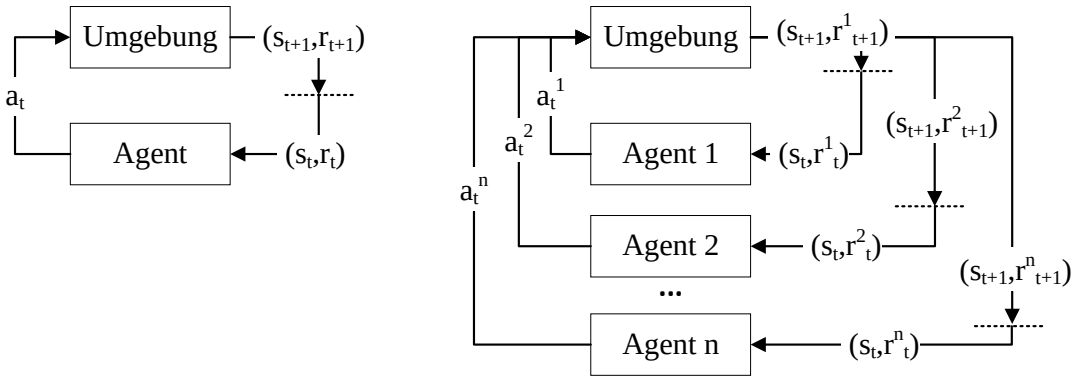


Abbildung 2.7: Allgemeines Vorgehen beim unüberwachten Lernen

Bestärkendes Lernen (engl. RL)

RL ermöglicht einem Software-Agenten eine optimale Strategie (engl. Policy) zur Lösung von Problemen basierend auf sequentieller Entscheidungsfindung zu erlernen. Mittels Trial-and-Error interagiert der Agent mit seiner Umgebung und sammelt Erfahrungen. Abbildung 2.8 a) stellt dar, wie der Agent in Abhängigkeit des aktuellen Zustands s_t der Umgebung eine Aktion a wählt und dadurch den Zustand der Umgebung in s_{t+1} verändert. Für seine Interaktion erhält der Agent eine Belohnung (engl. Reward) r, deren Höhe von der Zielerreichung abhängt. Aus dieser erlernt der Agent eine optimale Strategie, d. h. die optimale Wahl von Aktionen in Abhängigkeit des aktuell beobachteten Zustands der Umgebung, mit dem Ziel der Maximierung

der Belohnung. In der Ausführung richtet der Agent sein Handeln, d.h. die Wahl von Aktionen, nach der im Training erlernten Strategie.



a) Markov Entscheidungsprozess (engl. Markov Decision Process)

b) Markov Spiel (engl. Markov Game)

Abbildung 2.8: Architektur von Single-Agenten (links) und Multi-Agenten-Systemen (rechts) des bestärkenden Lernens (nach Zhang, Yang und Basar 2019)

In dem sogenannten Deep Reinforcement Learning (DRL) wird die Strategie π durch ein neuronales Netz mit mehr als zwei versteckten Schichten abgebildet. (Chen, Qiu u. a. 2021) heben unter anderem die Vorteile der hohen Skalierbarkeit, der Entscheidungsfindung in Echtzeit sowie dem Lernen ohne Vorwissen bei dem Einsatz von DRL Methoden hervor.

2.4.2 Multi-Agenten Bestärkendes Lernen

RL-Systeme bestehend aus mehr als einem Agenten werden als Multi-Agenten Bestärkendes Lernen (engl. Multi-Agent Reinforcement Learning (MARL)) bezeichnet (siehe Abbildung 2.8 b)). Die Interaktion mehrerer Agenten innerhalb einer Umgebung stellt zusätzliche Herausforderungen an das System.

Mathematische Grundlagen

Die einfachste mathematische Beschreibung eines RL-Systems ist der Markov-Entscheidungsprozess (engl. Markov Decision Process (MDP)). Ein MDP wird durch das 5-Tupel (S, A, P, R, γ) beschrieben (Zhang, Yang und Basar 2019; Wiering und Otterlo 2012). Dabei gilt:

- S : Zustandsraum, $s \in S$
- A : Aktionsraum, $a \in A$
- P : Übergangsfunktion, $P : S \times A \times S \rightarrow [0, 1]$

- R : Belohnungsfunktion, $R : S \times A \times S \to \mathbb{R}$
- γ : Diskontierungsfaktor, $\gamma \in [0, 1)$

Die Übergangsfunktion P gibt die Wahrscheinlichkeit an, mit der die Umwelt zum Zeitpunkt t durch die Ausführung der Aktion a_t aus dem Zustand s_t in den Zustand s_{t+1} übergeht. Dabei gilt $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$. Der Agent erhält für diesen Zustandsübergang die Belohnung $R(s_t, a_t, s_{t+1})$. Die Wahl der Aktion durch den Agenten ist in der Strategie π begründet, sodass $a_t \sim \pi(a_t|s_t)$ gilt. (Zhang, Yang und Basar 2019; Frochte 2018)

Voraussetzung für die Anwendbarkeit eines MDP für eine Problemstellung des RL ist die vollständige Sicht des Agenten auf die Zustände der Umgebung. Diese ideale Situation ist in der Realität häufig nicht gegeben, sodass eine Verallgemeinerung durch den teilbeobachteten MPD (engl. Partially Observable Markov Decision Process (POMDP)) Anwendung findet, welcher durch das 7-Tupel $(S, A, P, R, \Omega, O, \gamma)$ definiert ist. (Wiering und Otterlo 2012; Zhang, Yang und Basar 2019) Dabei wird der MDP wie folgt erweitert:

- Ω : Beobachtungsraum, $o \in \Omega$
- O : Beobachtungsfunktion, $O : S \times A \times \Omega \to [0, 1]$

Die Beobachtungsfunktion Ω beschreibt die Wahrscheinlichkeit einer Beobachtung o_t zum Zeitpunkt t, wenn die Umgebung durch die Aktion a_t von dem Zustand s_t in den Zustand s_{t+1} übergeht (Monahan 1982; Wiering und Otterlo 2012). Bei der vollständigen Sicht auf die Umgebung gilt $\Omega = S$.

Sowohl der MDP als auch der POMDP sind nur für die Beschreibung von RL-Systemen mit genau einem Agenten geeignet. Ein MAS beschreibt dagegen ein System, in dem mehrere verteilte und autonome Agenten ($n > 1$) Interaktionen mit einer gemeinsamen Umgebung ausführen (Weiss 1999). Für die mathematische Beschreibung dieser Systeme kommt das Markov Spiel (engl. Markov Game (MG)), auch Stochastisches Spiel genannt (Shapley 1953), zum Einsatz. Der Unterschied zwischen MDP und MG wird in Abbildung 2.8 verdeutlicht. Nach (Zhang, Yang und Basar 2019) wird das MG durch das 6-Tupel $(N, S, \{A^i\}_{i \in N}, P, \{R^i\}_{i \in N}, \gamma)$ beschrieben. Es gilt:

- N : Menge aller Agenten, $N = \{1, ..., n\}$ und $n > 1$
- S : Zustandsraum, $s \in S$
- A^i : Menge aller Aktionen eines Agenten i, $A := A^i \times ... \times A^n$ und $a \in A$
- P : Übergangsfunktion, $P : S \times A \times S \to [0, 1]$
- R^i : Belohnungsfunktion eines Agenten i, $R^i : S \times A \times S \to \mathbb{R}$
- γ : Diskontierungsfaktor, $\gamma \in [0, 1)$

Bei MAS wird zwischen dem globalen Nutzen, der das Ziel aller Agenten im Gesamtsystem beschreibt, sowie dem lokalen Nutzen, der lediglich für jeden Agenten einzeln gilt, unterschieden, Jeder Agent i hat das Ziel, eine optimale Strategie π^i mit $a_t^i \sim \pi^i(a_t^i|s_t)$ zu erlernen. MG zeichnen sich dadurch aus, dass die Agenten ihre Aktionen gleichzeitig ausführen und jeder Agent eine eigene Belohnungsfunktion besitzt.

Wie bei dem POMDP ist auch in einem MAS in Realität meist nicht der gesamte Zustandsraum beobachtbar. Eine eingeschränkte Sicht in einem MARL-System kann durch das teilbeobachtete MG (engl. Partially Observable Markov Game (POMG)) beschrieben werden, welches sich aus dem 8-Tupel (N, S, $\{A^i\}_{i\in N}$, P, $\{R^i\}_{i\in N}$, $\{\Omega^i\}_{i\in N}$, $\{O^i\}_{i\in N}$, γ) zusammensetzt und das MG wie folgt erweitert (Terry, Black u. a. 2020):

- Ω^i : Menge aller Beobachtungen eines Agenten i, $\Omega := \Omega^i \times ... \times \Omega^n$ and $o \in \Omega$
- O^i : Beobachtungsfunktion, $O^i : S \times A \times \Omega \to [0, 1]$

Die Beobachtungsfunktion $O^i(o_{t+1}^i|a_t^i, s_{t+1})$ beschreibt die Wahrscheinlichkeit einer Beobachtung o_t^i eines Agenten i zum Zeitpunkt t, wenn die Umgebung aus dem Zustand s_t in den Zustand s_{t+1} aufgrund der Aktion a_t^i übergeht. Wird angenommen, dass alle Agenten homogen und somit austauschbar sind, und, daraus abgeleitet, die Belohnungsfunktion aller Agenten einheitlich ist $R = R^1 = ... = R^n$, wird das System nach (Zhang, Yang und Basar 2019) auch als Team Spiel, oder Multi Agent Markov Decision Process (MMDP) (Yang und Wang 2020), bezeichnet, welches ein Spezialfall des MG ist.

Das mathematische Modell, welches eine sequentielle anstelle simultaner Ausführung der Aktionen der Agenten berücksichtigt, ist das sogenannte Agent Environment Cycle (AEC) Spiel. Es wird mathematisch durch das 11-Tupel (N, S, $\{A^i\}_{i\in N}$, $\{T^i\}_{i\in N}$, P, $\{\mathcal{R}^i\}_{i\in N}$, $\{R^i\}_{i\in N}$, $\{\Omega^i\}_{i\in N}$, $\{O^i\}_{i\in N}$, γ, v) beschrieben. (Terry, Black u. a. 2020) Die Nächster-Agent-Funktion $v(i'|s_t, i, a_t^i)$ gibt die Wahrscheinlichkeit an, mit der der nächste Agent i' ausgewählt wird. Das Tupel ist zur Vereinheitlichung mit den vorangegangenen Modellen um den Diskontierungsfaktor erweitert. An das POMG werden die folgenden Anpassungen vorgenommen:

- T^i : Übergangsfunktion der Agenten, $T^i : S \times A_i \to S$
- P : Übergangsfunktion der Umgebung, $P : S \times S \to [0, 1]$
- $\mathcal{R}^i$: Menge der möglichen Belohnungen des Agent i, $\mathcal{R}^i \subseteq \mathbb{R}$
- R^i : Belohnungsfunktion des Agenten i, $R^i : S \times N \times A \times S \times \mathcal{R}^i \to [0, 1]$
- v : Nächster-Agent-Funktion, $v : S \times \ N \times A \times N \to [0, 1]$

Abbildung 2.9 fasst die Zusammenhänge der genannten mathematischen Modelle in einem Venn-Diagramm zusammen. (Terry, Grammel u. a. 2020) belegen, dass für jedes POMG ein äquivalentes AEC Spiel existiert.

Des Weiteren werden MAS nach den folgenden Kriterien unterteilt:

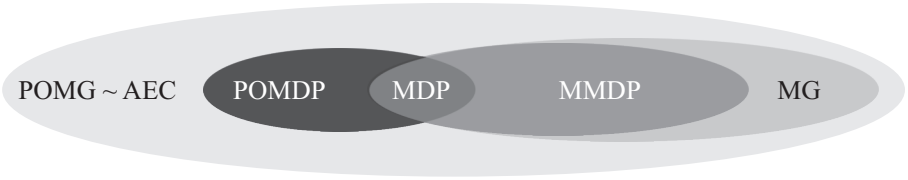


Abbildung 2.9: Venn-Diagramm zur Einordnung mathematischer Modelle nach (Yang und Wang 2020)

Entsprechend dem Venn-Diagramm gilt: MDP $\subset$ MMDP $\subset$ MG $\subset$ POMG sowie MDP $\subset$ POMDP $\subset$ POMG und POMG $\sim$ AEC.

- Topologie (Gronauer und Diepold 2021): zentralisiert vs. vollständig dezentral.
- Abfolge der Ausführung (Terry, Grammel u. a. 2020): Sequentielle vs. simultane Entscheidungsfindung.
- Belohnungsfunktion: individuelle Belohnungsfunktion (jeder Agent besitzt seine eigene Funktion) vs. gemeinsame Belohnungsfunktion (Agenten teilen sich eine einzige Funktion).
- Agent-Typen (Gronauer und Diepold 2021): homogene vs. heterogene Agenten.
- Interaktion (Zhang, Yang und Basar 2019): kooperatives vs. kompetitives Verhalten.
- Kommunikationslevel (Gronauer und Diepold 2021): Ausprägung der Kommunikation zwischen Agenten
- Trainingsarchitektur (Gupta, Egorov und Kochenderfer 2017):
  - zentrales Training + zentrale Ausführung,
  - zentrales Training + dezentrale Ausführung,
  - dezentrales Training + dezentrale Ausführung.

Nach (Gronauer und Diepold 2021) sind bei dezentralen Systemen die drei Aspekte Kommunikation, Koordination und Leistungsvergabe zu beachten. Während Leistungsvergabe in Single-Agenten-Systemen beschreibt, welcher Anteil der Belohnung auf frühere Aktionen entfällt, wird im MARL unterschieden, in welchem Verhältnis die Belohnung auf Teamebene zu der erhaltenen Belohnung des einzelnen Agenten steht. Ansätze zur Vergabe der globalen Belohnung (einheitlich für das gesamte Agententeam) und der lokalen Belohnung (für jeden Agenten individuell) sind unter anderem in (Mao, Gong und Xiao 2020) beschrieben.

Einteilung der Lernalgorithmen

Bei dem RL wird das Ziel verfolgt, eine optimale Strategie π^* für die Interaktion von einem oder mehreren Agenten mit der Umgebung zur Problemlösung zu finden. Die

Strategie ist optimal, wenn die Belohnung langfristig maximiert wird (Ertel 2021). Die zu erwartende kumulierte Belohnung berechnet sich entsprechend der Zustands-Wertefunktion $V^{\pi}(s_t)$, der sogenannten V-Funktion, nach Gleichung 2.1 aus den einzelnen Belohnungen r, die mit dem potenzierten Diskontierungsfaktor γ multipliziert und anschließend aufsummiert werden, sodass die Summe konvergiert. Die Funktion bewertet die Wahl der Aktionen und deren kurzfristige sowie langfristige Auswirkung, beginnend von Zustand s_t. Belohnungen in entfernter Zukunft werden dabei geringer gewichtet als die aktuelle und in naher Zukunft liegenden.

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{2.1}$$

Nach (Ertel 2021) ist die optimale Strategie π^* erreicht, wenn Gleichung 2.2 für alle Zustände s gilt:

$$V^{\pi^*}(s) \geqslant V^{\pi}(s) \tag{2.2}$$

Unter Einsatz der Übergangsfunktion $s_{t+1} = \delta(s_t, a)$ lässt sich die optimale V-Funktion V^* in Gleichung 2.3 definieren. Das bedeutet, dass eine optimale Aktion a im Zustand s von der aktuell vergebenen Belohnung r_t sowie dem V^* im nächsten Zustand s_{t+1} abhängt. (Frochte 2018)

$$V^*(s_t) = r_t + \gamma V^*(\delta(s_t, a)) \tag{2.3}$$

Die optimale V-Funktion V^* ist von der optimalen Strategie π^* abhängig, welche nicht bekannt ist, und daher nicht berechenbar. Bei Q-Learning Verfahren wird aus diesem Grund auf die Aktions-Wertefunktion, die sogenannte Q-Funktion, zurückgegriffen. Die Q-Funktion beschreibt den Nutzen einer im Zustand s durchgeführten Aktion a (Frochte 2018). Sie basiert auf dem Optimalitätsprinzip nach Bellmann (Bellman 1954).

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a)) = r(s, a) + \gamma \cdot \max_{a_{t+1}} Q(\delta(s, a_{t+1})) \tag{2.4}$$

Da Q zunächst nicht bekannt ist, nutzt der Agent den approximierten Wert $\hat{Q}$. Bei dem Q-Learning werden die Zustands-Aktions-Paare jeweils mit dem dazugehörigen $\hat{Q}$-Wert in einer Tabelle abgespeichert und die Approximation $\hat{Q}$ von Q schrittweise optimiert. (Frochte 2018)

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a)) = r(s, a) + \gamma \cdot \max_{a_{t+1}} Q(\delta(s, a_{t+1})) \tag{2.5}$$

Lernverfahren können nach drei Aspekten kategorisiert werden, dem Vorhandensein eines Modells, dem Verfahren zur Sammlung von Erfahrung und dem Optimierungsansatz.

Modell: Wie in Abbildung 2.10 dargestellt, wird zwischen modellbasierten und modellfreien Ansätzen unterschieden. Modellbasierte Ansätze besitzen Kenntnis über die Übergangsdynamiken $P(s_{t+1}|s_t, a_t)$ in Form von Modellen (Wang, Liu u. a. 2020).

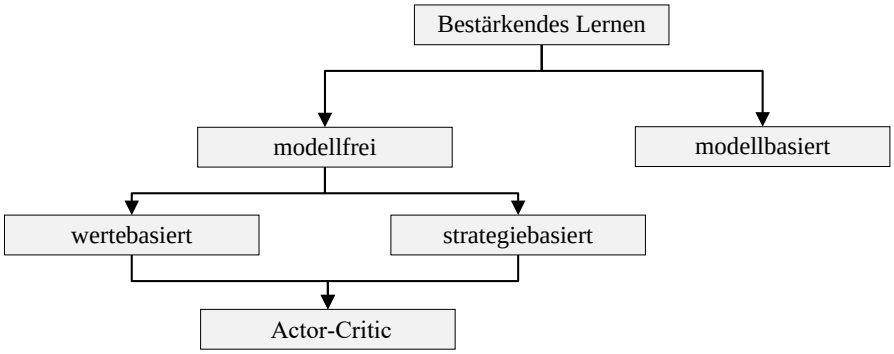


Abbildung 2.10: Klassifizierung von Algorithmen des Bestärkenden Lernens (nach Canese u. a. 2021)

Sammlung von Erfahrungen: Die Verfahren werden nach dem Konzept unterschieden, wie die Erfahrungen gesammelt werden. On-Policy Verfahren beschreiben Verfahren, bei denen die zu optimierende Zielstrategie und die aktuell zum Sammeln der Erfahrung genutzte Verhaltensstrategie identisch sind. Bei den Off-Policy Verfahren dagegen sind Verhaltens- und Zielstrategie zwei unabhängige Strategien. (Zhang, Yang und Basar 2019)

Optimierung: Modellbasierte Ansätze können klassische Verfahren wie Werte- und Strategieiteration nutzen, da auf Basis der Modelle die Berechnung des Bellman-Operators möglich ist. Modellfreie Ansätze basieren dagegen auf wertebasierten, strategiebasierten oder kombinierten Lernverfahren. Wertebasierte Verfahren basieren auf dem Zusammenhang, dass die Strategie optimal ist, wenn die optimale V- bzw. Q-Funktion gefunden ist. Da die V-Funktion bei den modellfreien Verfahren nicht berechnet werden kann, wird bei wertebasierten Verfahren die optimale Q-Funktion gesucht. Beispiele für diese Verfahren sind Q-Learning-Verfahren. Das System ist optimal, wenn die maximale langfristige Belohnung erzielt wird, weshalb gilt (Chen, Qiu u. a. 2021):

$$\pi^*(s_t) = \arg\max_{a \in \mathbb{A}} Q^*(s_t, a), \forall s_t \in \mathbb{S} \tag{2.6}$$

Das klassische Q-Learning-Verfahren nach (Watkins und Dayan 1992) wird in Gleichung 2.7 definiert (Zhang, Yang und Basar 2019). Dabei steht α für die Lernrate.

$$\hat{Q}(s_t, a_t) \leftarrow (1-\alpha)\hat{Q}(s_t, a_t) + \alpha[r + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1})], \quad \alpha > 0 \tag{2.7}$$

Im Gegensatz dazu wird bei den strategiebasierten Verfahren mittels Strategie-Gradienten-Verfahren mit jeder Iteration direkt die Strategie optimiert. Dazu wird der Gradient der Zielfunktion $J(\theta)$ wie folgt berechnet (Chen, Qiu u. a. 2021):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi\theta}[\nabla_\theta \ln \pi_\theta(a|s) G(\tau|s,a)] \tag{2.8}$$

Dabei bezeichnet $G(\tau|s,a)$ die langfristige Belohnung der Trajektorie, die mit (s,a) beginnt. Bekannte strategiebasierte Verfahren sind REINFORCE-Algorithmen (Williams 1992).

Nach (Chen, Qiu u. a. 2021) sind wertebasierte Verfahren robust gegenüber der Parameterinitialisierung und geeignet für das Finden von globalen Extrema. Die strategiebasierten Verfahren überwiegen allerdings aufgrund ihres besseren Konvergierens, ihrer Eignung für sowohl deterministische als auch stochastische Strategien sowie ihrer Eignung für große Aktionsräume.

Um die Vorteile aus beiden Ansätzen zu nutzen, wurden die Akteur-Kritiker-Verfahren entwickelt, bei denen ein wertebasierter Kritiker dem strategiebasierten Akteur eine Einschätzung zu dem aktuellen Zustand gibt, in dem er sich befindet. (Canese u. a. 2021) beschreibt die zwei Bestandteile wie folgt:

- Kritiker: Der Kritiker schätzt die Wertefunktion.
- Akteur: Der Akteur stellt die Strategie dar und aktualisiert diese mittels Strategie-Gradienten-Verfahren entsprechend der Schätzung des Kritikers.

Bei diesen Verfahren ist es das Ziel, sowohl den Werte- als auch den Strategieverlustwert zu minimieren. Sowohl Akteur als auch Kritiker können durch Neuronale Netze abgebildet werden. Bekannte Akteur-Kritiker-Verfahren sind Advantage Actor Critic (A2C) und Proximal Policy Optimization (PPO). Bei A2C wird anstelle der Q-Funktion oder V-Funktion die Vorteilsfunktion $A(s_t,a_t) = Q(s_t,a_t) - V(s_t)$ eingesetzt (Mnih u. a. 2016). PPO basiert ebenfalls auf der Vorteilsfunktion (Schulman, Wolski u. a. 2017) und hat sich aufgrund seiner guten Ergebnisse und seines geringen Aufwands bei dem KI-Forschungs- und Entwicklungsunternehmen OpenAI als das Standardverfahren etabliert (Schulman, Klimov u. a. 2017).

Die Wahl geeigneter Hyperparameter-Werte hat großen Einfluss auf die Qualität der Ergebnisse. Relevante Hyperparameter in einem RL-System sind unter anderem die Lernrate, die Wahl des Lernalgorithmus, Gestaltung und Parametrierung der Belohnungsfunktion und die Trainingsschritte. Kommen Neuronale Netze zum Einsatz sind ebenfalls die Wahl der Aktivierungsfunktion sowie Anzahl und Größe der Schichten als Hyperparameter zu berücksichtigen.

2.4.3 Neuronale Netze und Deep Learning

Ein besonders relevantes Forschungsgebiet des ML sind Künstliche Neuronale Netze, kurz Neuronale Netze, da sie in allen drei genannten Teilgebieten Anwendung finden. Neuronale Netze sind den klassischen ML Algorithmen überlegen, wenn die Datenmenge sehr hoch ist oder die Aufgabenstellung von hoher Komplexität ist.

Die Funktionsweise ist an die des menschlichen Gehirns angelehnt. Ein Neuronales Netz besteht aus mehreren künstlichen Neuronen, die miteinander verknüpft sind und durch die Eingangsdaten aktiviert werden. Zunächst wird in Abbildung 2.11 die Funktionsweise eines einzelnen Neurons dargestellt und im Anschluss in Abbildung 2.12 deren Verknüpfung zu einem einschichtigen Perzeptron, dass die grundlegendste und einfachste Ausführung eines Neuronalen Netzes darstellt.

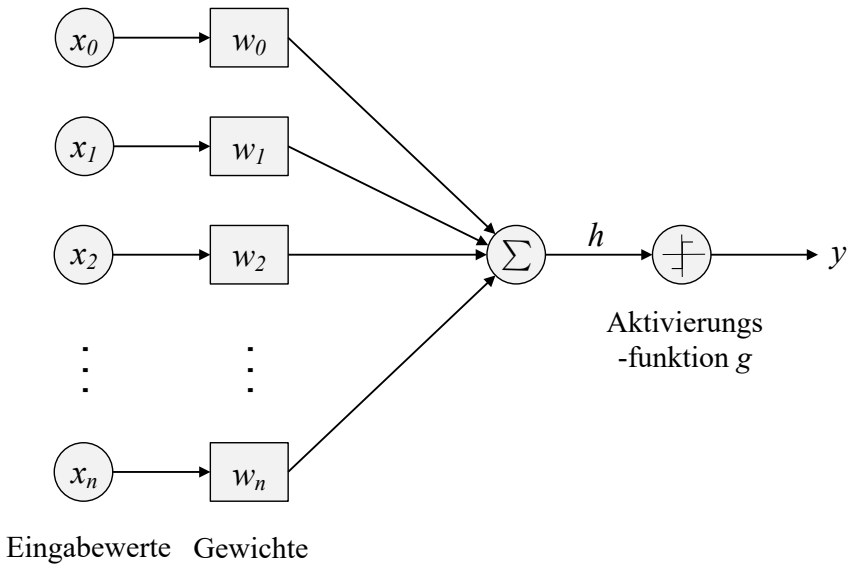


Abbildung 2.11: Mathematisches Modell eines Neurons eines Neuronalen Netzes nach (Frochte 2018)

Jede Eingabe x_i wird mit dem jeweiligen Gewicht $w_i \in \mathbb{R}$ multipliziert. Die Produkte werden aufsummiert, sodass $h(x) = \sum_{n=0}^{N} w_i x_i$ mit $h : \mathbb{R}^n \to \mathbb{R}$ gilt. $h(x)$ ist der Eingangswert für eine Aktivierungsfunktion g. Der Ausgabewert ergibt sich aus $y = g(h(x))$. Die Aktivierungsfunktionen stellen den Zusammenhang zwischen Netzeingabe und Aktivitätslevel eines Neurons dar. Die konkrete Gestalt der Aktivierungsfunktion ist nicht vorgegeben. Häufig werden Sigmoid-, Softmax-, tanh- oder ReLU-Funktionen gewählt (Frochte 2018).

Bei dem einfachsten Fall, den reinen Feedforward-Netzen, sind die Neuronen, wie in Abbildung 2.12 dargestellt, immer mit Neuronen der vorherigen Schicht sowie der direkt darauffolgenden Schicht, allerdings nicht innerhalb einer Schicht oder über mehrere Schichten hinweg, verbunden. Die Hintereinanderausführung solcher Neuronen lässt sich mathematisch als Funktion von Funktionen formalisieren. Die Ausgabe berechnet sich nach: $y = g(h(x))$ mit $h = W \cdot x$. Für das Beispiel in Abbildung 2.12 gilt für y_1 demnach $h(x) = w_{1,1} \cdot x_1 + w_{1,2} \cdot x_2 + w_{1,3} \cdot 1$, wobei 1 den Bias darstellt. (Frochte 2018)

Der Trainingsprozess basiert grundlegend auf der Hebbschen Lernregel. Dazu wird die Ausgabe y mit dem erwarteten Wert $\hat{y}$ abgeglichen. Ziel ist es, die Differenz $error = \hat{y} - y$ durch Anpassung der Gewichte zu minimieren. Dabei werden bei

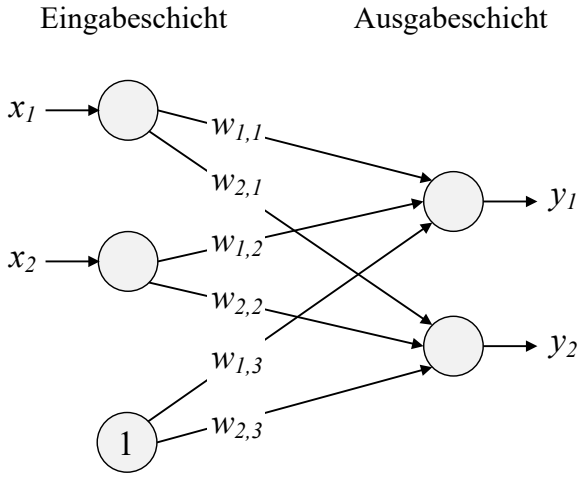


Abbildung 2.12: Schematische Darstellung eines einschichtigen Perzeptrons nach (Frochte 2018)

$\hat{y} > y$ die Gewichte vergrößert und bei $\hat{y} < y$ die Gewichte verringert. Die Größe der Anpassung ist abhängig von der Größe des Gewichtes selbst, dem Unterschied zum erwarteten Wert sowie der Lernrate $\alpha \in]0, 1]$, die sprunghafte Änderungen verhindert. Die Berechnung erfolgt nach Gleichung 2.9:

$$w_j = w_j + \delta w_j \text{ mit } \delta w_j = \alpha \cdot error \cdot x_j \tag{2.9}$$

Diese einfache Berechnung ist nur auf einschichtige Netze anwendbar. Für komplexere Netze erfolgt die schrittweise Anpassung der Gewichte in der Regel durch das Gradientenabstiegsverfahren, die sogenannte Backpropagation. (Frochte 2018)

Neuronale Netze mit mehr als zwei versteckten Schichten werden als tiefe Neuronale Netze (engl. Deep Neural Network (DNN)) und das Gebiet des ML, welches sich mit DNN befasst, als Deep Learning bezeichnet (Frochte 2018). Neben dem einfachen Feedforward-Neuronalen Netzen lassen sich Neuronale Netze noch in weitere Strukturen unterteilen. Faltungsnetze (engl. Convolutional Neural Network (CNN)) sind besonders für rasterförmige Daten geeignet und daher insbesondere in der Bildverarbeitung eingesetzt. Sie besitzen keine gewichteten Verbindungen im klassischen Sinn. Stattdessen werden, wie schematisch in Abbildung 2.13 beschrieben, Filtermatrizen schrittweise über die Eingabedaten gelegt und sogenannte Merkmalskarten (engl. Feature Maps) erstellt. Einer Faltungsschicht folgt meist eine sogenannte Poolingschicht, die die Größe der Merkmalskarten weiter reduziert, indem nur der Maximalwert oder Mittelwert eines Rastersegments beibehalten wird. Mittels sogenanntem Padding lassen sich Eingabedaten bzw. Merkmalskarten am Rand erweitern, um die Größe der Abtastung mit den Filtermatrizen anzupassen. Eine weitere Form sind Rekurrente Neuronale Netze (engl. Recurrent Neural Network (RNN)), welche besonders für die Verarbeitung von Sequenzen ausgelegt sind. Sie sind damit

insbesondere für die Modellierung zeitlicher Zusammenhänge geeignet, welche beispielsweise bei der Verarbeitung natürlicher Sprache relevant sind. Die Ausgabe ist nicht nur direkt von den aktuellen Eingabedaten abhängig, sondern auch von Eingaben zu früheren Zeitpunkten. Ein großer Vorteil von RNN ist, dass die Eingabelänge variieren darf.

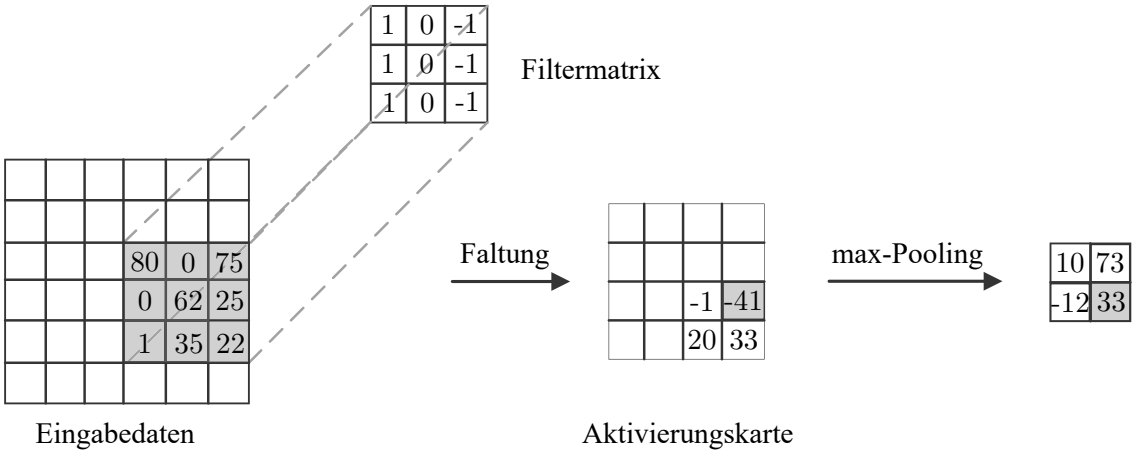


Abbildung 2.13: Beispiel einer Faltungsschicht mit nachfolgendem max-Pooling

2.5 Datenkompression

Eine Datenkompression, oder auch Datenverdichtung, ist die Verringerung der Anzahl der Bits, die eine Datenmenge repräsentieren, durch ein mathematisches Verfahren. Die Motivation dafür liegt in erster Linie in der Verringerung der Aufwände für das Abspeichern und Übermitteln der Daten (Werner 2017). Die Einteilung der Verfahren kann nach drei Kriterien erfolgen (Chiarot und Silvestri 2023):

- Adaptivität an Änderungen in den Daten.
- Symmetrie von Kompression und Dekompression.
- Vorhandensein eines Rekonstruktionsfehlers, d.h. Datenverluste zwischen Originaldaten und dekomprimierten Daten.

2.5.1 Verlustfreie Datenkompression

Die verlustfreie Datenkompression wird auch als Reduktion von Redundanzen bezeichnet. Verlustfrei bedeutet, dass die ursprüngliche Datenmenge durch die Dekodierung vollständig wiederhergestellt werden kann. Die Verfahren zur verlustfreien Datenkompression lassen sich entsprechend Abbildung 2.14 in zwei Kategorien unterteilen: die statistischen und die wörterbuchbasierten Verfahren. Daneben gibt es noch einzelne weitere Verfahren, wie die Lauflängenkodierung und Delta-Kodierung, die sich keiner dieser Kategorie explizit zuordnen lassen. Alle Verfahren haben das Ziel, die ursprüngliche Datenmenge durch insgesamt weniger Bits zu repräsentieren.

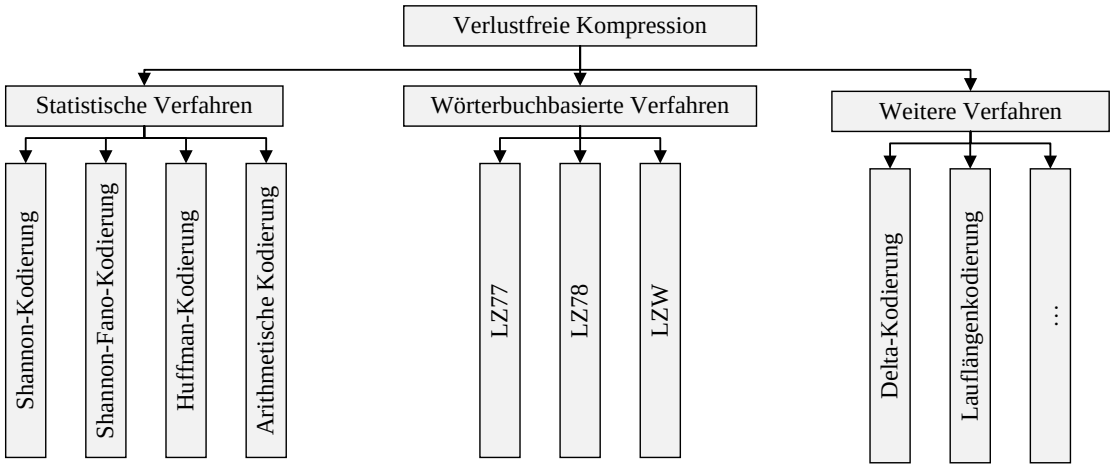


Abbildung 2.14: Einteilung der verlustfreien Verfahren (nach Huang und Chen 2017)

Statistische Verfahren, auch Verfahren der Entropiekodierung genannt, nutzen ein statistisches Modell, um ein Symbol oder eine Symbolkette in Abhängigkeit seiner Auftrittshäufigkeit zu kodieren. Für die Verfahren nach Shannon und Fano, Huffman sowie die Arithmetische Kodierung, welche Grundlage für viele heutige Algorithmen sind, ist Kenntnis über die Auftrittswahrscheinlichkeit der einzelnen Symbole a priori erforderlich. Alle diese Verfahren haben zwei Merkmale gemeinsam:

- Häufig auftretende Symbole werden mit einer geringen und seltene Symbole mit einer größeren Anzahl an Bits kodiert.
- Die Kodierung ist präfixfrei.

Präfixfrei bedeutet, dass keine Kodierung eines Symbols der Präfix der Kodierung eines anderen Symbols ist. Dadurch ist die Kodierung auch ohne den Einsatz von Trennzeichen eindeutig. Statistische Verfahren kodieren im Allgemeinen nur einzelne Symbole, während Wörterbuch-Verfahren auch Symbolketten kodieren. Die Verfahren unterscheiden sich durch das Vorgehen bei der Erstellung der optimalen Kodierung. Bei den Verfahren nach Shannon-Fano und Huffman basiert dies auf der Erstellung von Code-Bäumen, bei denen die Kodierung über den Pfad erstellt wird. Häufig auftretenden Symbolen werden kurze Pfade zugeordnet. Bei der Arithmetischen Kodierung wird am Anfang ein festes Intervall gewählt, welches in Abhängigkeit der Auftrittswahrscheinlichkeit des jeweiligen Symbols weiter geschachtelt wird. Eine Sequenz von Symbolen wird somit als eine einzige Fließkomma-Zahl abgebildet.

Wörterbuch-Verfahren, auch String-Ersatz-Verfahren genannt, legen im Allgemeinen Symbole oder Symbolketten in einem Wörterbuch ab und kodieren den Wörterbucheintrag. Es wird zwischen statischen und dynamischen Wörterbüchern unterschieden. Die Einträge statischer Wörterbücher werden a priori festgelegt. Dynamische Wörterbücher entstehen dagegen sukzessive während der Kodierung. Ein großer Vorteil gegenüber den statistischen Verfahren ist, dass im Vorhinein kein Wissen

über die Auftrittshäufigkeit der Symbole erforderlich ist. Verfahren, denen der Quellensymbolumfang, also die vorkommenden Symbole, nicht bekannt sein muss, sind Verfahren der universellen Quellenkodierung. Das Wörterbuch muss nicht zwischen Kodierer und Dekodierer übermittelt werden, da das Vorgehen bei der Erstellung des Wörterbuchs auf beiden Seiten einheitlich ist. Wie in Abbildung 2.14 dargestellt, sind als relevante Verfahren die in den 1970er Jahren von Abraham Lempel und Jacob Ziv entwickelten Verfahren LZ77 und LZ78 zu nennen. Das LZ77-Verfahren nutzt ein gleitendes Fenster als lokales, dynamisches Wörterbuch, während das LZ78-Verfahren ein globales Wörterbuch über die gesamte kodierte Datensequenz anlegt. Das Code-Wort bei LZ78 setzt sich aus dem Index des Wörterbucheintrags sowie dem aktuell neuen Zeichen zusammen. Das LZW-Verfahren ist eine Weiterentwicklung des LZ78-Algorithmus durch Terry Welch im Jahr 1983 und verkürzt das Code-Wort nur noch auf den Wörterbuchindex. Dafür ist es bei LZW allerdings notwendig, alle vorkommenden Zeichen bereits vor der Kodierung im Wörterbuch einzutragen. Diese drei Verfahren dienen als Grundlage für viele heutige Kompressionsverfahren.

Weitere Verfahren sind die Lauflängen- und Delta-Kodierung. Lauflängenkodierung ist ein einfaches Verfahren, das mehrere gleiche, aufeinanderfolgende Zeichen durch das Zeichen selbst sowie der Anzahl des Zeichens zusammenfasst. Die Effizienz ist abhängig von der Anzahl mehrfach aufeinanderfolgender Zeichen. Bilddateien eignen sich dementsprechend gut für die Lauflängenkodierung, wogegen bei verrauschten Signalen geringe Kompressionsraten zu erwarten sind. Delta-Kodierung bezeichnet die Kodierung von Differenzen. Es kann sich dabei um die Differenz des aktuellen Symbols zum Vorgängersymbol oder zu einem Symbol, welches von einem Modell vorhergesagt wird, handeln. Der Aufbau des prädiktiven Modells kann sehr unterschiedlich sein, beispielsweise als Simulationsmodell oder mathematisches Modell, welches auf Basis vorangegangener Symbole eine Vorhersage ableitet.

2.5.2 Verlustbehaftete Datenkompression

Die verlustbehaftete Datenkompression zeichnet sich durch das Entfernen von Irrelevanzen aus, welches mit einem Datenverlust einhergeht. Die ursprüngliche Datenmenge ist somit nicht vollständig wiederherstellbar, allerdings ist das Ziel, alle relevanten Daten beizubehalten. Verlustbehaftete Verfahren können im Allgemeinen eine deutlich stärkere Verdichtung erzielen als verlustfreie Verfahren. Der Großteil der Methoden zur verlustbehafteten Datenkompression kann den zwei Konzepten funktionale Approximation und ML zugeordnet werden. Daneben bestehen einzelne weitere Verfahren, wie die für Bildkompression geeignete fraktale Kompression oder Kompression mittels prädiktiver Modelle. Das ursprüngliche Haupteinsatzgebiet der verlustbehafteten Datenkompression ist die Kompression von Bild- und Audiodateien. Mittlerweile ist sie aber auch in der Industrie, beispielsweise in WSN, relevant.

Verfahren basierend auf funktionaler Approximation umfassen verschiedene mathematische Methoden zur Approximation von Daten. Beispiele hierfür sind die Hauptkomponentenanalyse und die diskrete Kosinustransformation (engl. Discrete

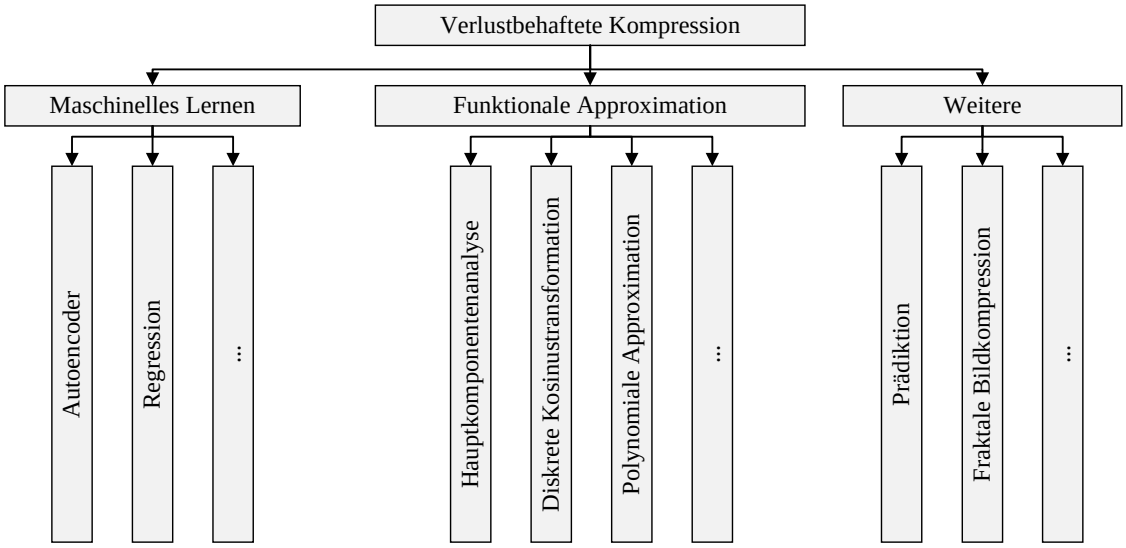


Abbildung 2.15: Einteilung der verlustbehafteten Verfahren zur Datenkompression

Cosine Transform (DCT)) sowie nach (Chiarot und Silvestri 2023) die schrittweise polynomiale Approximation (Eichinger u. a. 2015) und Chebyshev polynomiale Transformation (Hawkins und Darlington 2012).

Transformationen werden durchgeführt, um Daten in einer anderen Form darzustellen, in der die beinhalteten Informationen offensichtlicher werden. Dies kann beispielsweise die Transformation vom Zeitbereich in den Frequenzbereich sein. In der transformierten Darstellung können relevante Daten von irrelevanten Daten besser separiert werden. Die Anzahl der kodierten Datenpunkte wird im Frequenzbereich durch die selbe Anzahl an Koeffizienten dargestellt. Die Koeffizienten der transformierten Darstellung werden nach ihrer Relevanz absteigend sortiert und nur die am höchsten gewichteten beibehalten. Um eine Kompression zu erzielen, werden weniger relevante Koeffizienten, d. h. jene Frequenzanteile von geringerer Energie, entfernt. Die relevanten Koeffizienten und ihre Indizes werden für die Rücktransformation beibehalten. Die Kompressionsrate von Transformationen kann sich auf zwei Weisen ergeben. Zum einen kann sie durch die konkrete Anzahl der Indizes, die bei der Kompression beibehalten werden, vorgegeben werden, sodass gilt $CR = \frac{Indizes\_original}{Indizes\_komp}$. Zum anderen kann die Genauigkeit vorgegeben werden, d. h. das transformierte Signal enthält eine vorgegebene Prozentzahl der Energie der Originalsignale.

Die Funktionsweise der verlustbehafteten Datenkompression mittels Transformation wird anhand der DCT erläutert. Der ursprüngliche Datensatz liegt in der Stützstellendarstellung mit $y_j = f(x_j)$ vor. Nach der Anwendung der DCT liegt er in der Koeffizientendarstellung vor mit $y_j = a_0 \cdot c_0(x_j) + \ldots + a_{n-1} \cdot c_{n-1}(x_j)$. Der Vektor $[a_0 \ldots a_{n-1}]$ beinhaltet die n Gewichte der n Basisfunktionen $c_i(x)$. Bei der DCT werden als Basisfunktionen $c_i(x) = \cos(i \cdot x)$ und als Stützstellen die Werte $x_j = (j + \frac{1}{2}) \cdot \frac{\pi}{n}$ verwendet. Die Anzahl der Koeffizienten a_i ist identisch zu der der Stützstellen x_i. Die Koeffizienten werden nach ihrer Energie sortiert. Eine bestimmte Anzahl der Ko-

effizienten mit dem höchsten Energiegehalt wird beibehalten, die anderen werden verworfen. Je mehr Koeffizienten beibehalten werden, desto exakter ist die Approximation, desto schlechter ist das Kompressionsverhältnis. Anschließend erfolgt die Rücktransformation zur Stützstellendarstellung durch inverse DCT.

Verfahren basierend auf Maschinellem Lernen, insbesondere basierend auf Neuronalen Netzen, kommen aufgrund ihrer guten mathematischen Anpassungsfähigkeit und verallgemeinernden Lernfähigkeit zusätzlich zu den klassischen Verfahren verstärkt zum Einsatz (Rosenberger, Kübel und Rothfuß 2022; Singh u. a. 2020). Beispiele dafür sind Neuronale-Netz-Regression nach (Park, Park und Choi 2018) oder Autoencoder. Autoencoder sind Neuronale Netze von bestimmter Architektur. Sie besitzen eine deutlich geringere Anzahl an Knoten in der mittleren Schicht, der sog. Engpassschicht, als in der Ein- und Ausgabeschicht. Für die Datenkompression wird die Decoder-Seite nicht benötigt, sondern nach der Engpassschicht abgebrochen. Die Ausgabe y der Engpassschicht entspricht der komprimierten Eingabe x_{komp}. Den Lernvorgang stellt die Anpassung der Gewichte der einzelnen Knoten mit der Zielgröße $y = x$ dar. Je weniger Knoten die Engpassschicht umfasst, desto stärker ist die Verdichtung, aber desto größer ist auch die Abweichung zwischen Ein- und Ausgabe, d. h. der mögliche Informationsverlust. Abbildung 2.16 zeigt die Struktur eines Autoencoders bestehend aus voll vernetzten Schichten. Alternativ können auch andere Neuronale-Netz-Strukturen entsprechend Abschnitt 2.4.3 auf Autoencoder angewendet werden.

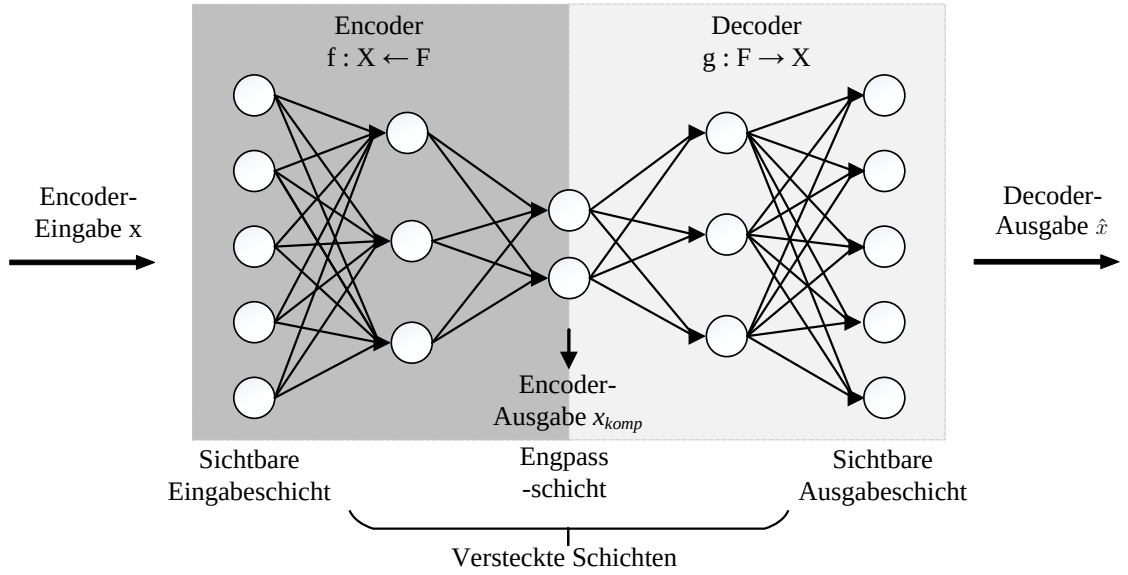


Abbildung 2.16: Schematische Funktionsweise eines vollvernetzten Autoencoders

Während eine Datenkompression immer aus Kodierung und Dekodierung besteht, setzt die Datenreduktion die Dekodierung nicht voraus. Verfahren der Datenreduktion sind immer verlustbehaftet. Beispiele sind Quantisierung, Filterbanken, Modifikation von Abtastraten oder Datenaggregation.

2.6 Anomalieerkennung

Die Anomalieerkennung als Methode der Datenanalyse befasst sich mit der Erkennung von Mustern. Sie ist die Grundlage, um mehr Wissen über das Verhalten und dem Zustand von Systemen zu erlangen und somit die Voraussetzung für relevante Technologiefelder, wie beispielsweise die Zustandsüberwachung und die vorausschauende Instandhaltung von industriellen Anlagen. Anomalien sind Muster in Daten, die nicht mit einem definierten Verhalten übereinstimmen. Sie zeichnen sich durch zwei Charakteristika aus (Goldstein und Uchida 2016):

- Anomalien weichen hinsichtlich ihrer Merkmale von der Norm ab.
- Anomalien treten im Verhältnis zu normalen Instanzen selten auf.

Aus diesen beiden Charakteristika leitet sich ab, dass die Auftrittswahrscheinlichkeit, und somit auch die Erwartung des Auftretens, von anormalen Datenpunkten gering ist. In der Informationstheorie wird der Informationsgehalt eines Datums in Abhängigkeit seiner Wahrscheinlichkeit definiert. Anomalien, die sich durch ihre geringe Auftrittswahrscheinlichkeit auszeichnen, besitzen somit einen hohen Informationsgehalt. Die Ergebnisse der Anomalieerkennung werden entweder als Kennzeichnung oder als Skalenwert, der die Schwere der Anomalie bezeichnet, ausgegeben. Anomalien sind nicht mit Fehlern gleichzusetzen. Erst die Interpretation einer Anomalie entspricht einer Fehlerdiagnose (vgl. Abschnitt 2.2).

2.6.1 Klassifikation der Anomaliearten

Anomalien, auch Ausreißer genannt, werden in vier Arten unterteilt: Punktanomalien, Kontextanomalien, kollektive Anomalien und schleichende Anomalien.

Punktanomalien werden, wie in Abbildung 2.17 dargestellt, als Abweichung einzelner Instanzen von der Mehrheit der Daten definiert (Chandola, Banerjee und Kumar 2009). Die Erkennung von Punktanomalien ist im Vergleich zu den weiteren Anomalien von geringer Komplexität.

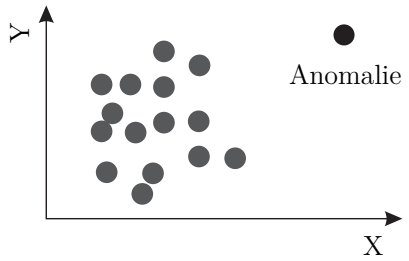


Abbildung 2.17: Punktanomalie (nach Fahim und Sillitti 2019)

Kontextanomalien erfordern für ihre Erkennung, Daten erst in einem bestimmten Kontext, also im mehrdimensionalen Raum, zu betrachten. Die einzelnen Werte sind nicht auffällig, aber in Zusammenhang mit Daten anderer Sensoren (multivariat) oder in Anbetracht der Zeit (univariat) stellen sie eine Anomalie dar. In

Abbildung 2.18 ist ein Beispiel ersichtlich. Die Temperaturwerte zu den Datenpunkten t_1 und t_2 sind identisch. Berücksichtigt man allerdings das zeitliche Auftreten, so wird der Datenpunkt t_1 als normal und t_2 als anormal eingestuft.

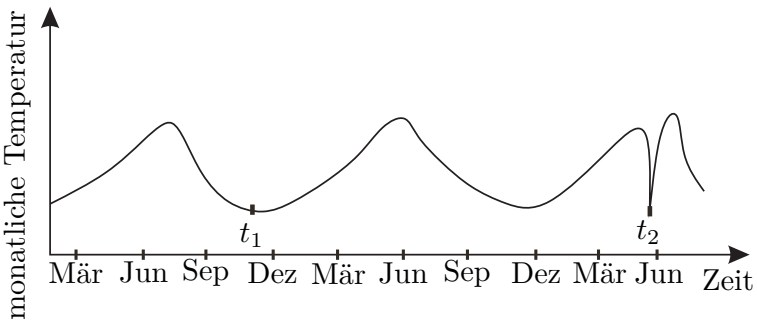


Abbildung 2.18: Kontextanomalie in Anbetracht der Zeit (nach Fahim und Sillitti 2019)

Kollektive Anomalien beziehen sich im Gegensatz zu den beiden bereits genannten Anomalien auf eine Sammlung von Werten. Das Auftreten der einzelnen Datenpunkten stellt dabei keine Anomalie dar, ihr gemeinsames Auftreten oder die Relation mit den anderen Daten des Datensatzes betrachtet jedoch schon. (Chandola, Banerjee und Kumar 2009) Dies ist in Abbildung 2.19 anschaulich dargestellt.

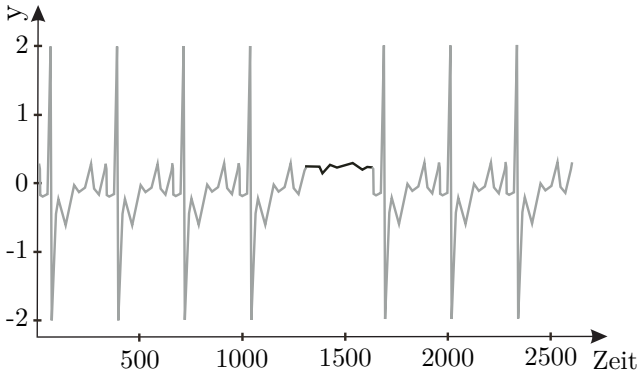


Abbildung 2.19: Kollektive Anomalie (nach Fahim und Sillitti 2019)

Schleichende Anomalien, auch Drifts genannt, sind im Wesentlichen eine besondere Form der Kontextanomalien und werden in der Fachliteratur (Chandola, Banerjee und Kumar 2009; Goldstein und Uchida 2016; Fahim und Sillitti 2019) meist nicht explizit benannt. In dieser Arbeit werden schleichende Anomalien als eigenständige Anomalieart betrachtet, da sie im Maschinenbau in vielen Bereichen, beispielsweise in der Zustandsüberwachung von Getrieben, von großer Bedeutung sind und ihre Erkennung in der Industrie bereits praktiziert wird. Der Kontext ist in diesem Fall die Dimension der Zeit. Abbildung 2.20 zeigt, dass die Anomalien sich durch

die langsame, über eine verhältnismäßig lange Zeitspanne entwickelnde Abweichung von dem erwarteten Wert auszeichnen.

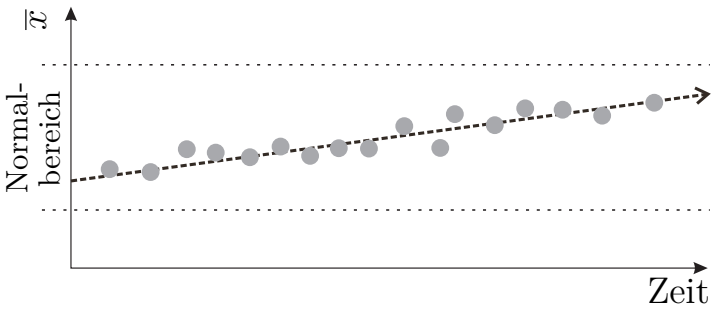


Abbildung 2.20: Schleichende Anomalie

2.6.2 Methoden zur Anomalieerkennung

Die Verfahren zur Anomalieerkennung werden in der Fachliteratur (Goldstein und Uchida 2016; Chatterjee u. a. 2019) häufig in fünf Gruppen eingeteilt.

1. Statistische Verfahren
2. Nächste-Nachbarn-Verfahren
3. Clusterbasierte Verfahren
4. Klassifizierende Verfahren
5. Unterraumbasierte Verfahren

Zusätzlich werden noch zwei weitere, etwas weniger verbreitete Gruppen genannt.

6. Informationstheoretische Verfahren (nach Chandola, Banerjee und Kumar 2009)
7. Verfahren basierend auf Isolation (nach Liu, Ting und Zhou 2008)

Alle Verfahren machen sich für die Erkennung von Anomalien die anfangs genannten zwei Charakteristika von Anomalien, die Abweichung der Merkmale von der Norm sowie das seltene Auftreten, zu Nutze.

Statistische Verfahren basieren auf der Annahme, dass normale Daten in Gebieten hoher Wahrscheinlichkeit auftreten, wogegen Anomalien in Regionen niedriger Wahrscheinlichkeit liegen (Chandola, Banerjee und Kumar 2009). Die Basis für diese Form der Anomalieerkennung sind statistische Modelle, sodass die Erstellung dieser Modelle wesentlicher Bestandteil der Verfahren ist. Dabei wird zwischen parametrischen und nichtparametrischen Techniken unterschieden. Parametrisch bedeutet, dass das statistische Modell a priori durch Parameter definiert werden muss, wie es beispielsweise bei Gauß-Verteilungen mit den Parametern Standardabweichung

σ und Erwartungswert μ der Fall ist. Dafür ist Vorwissen über die zugrundeliegende Verteilung erforderlich (Eskin 2000). Abbildung 2.21 stellt exemplarisch vier unterschiedlich parametrierte Gaußverteilungen dar.

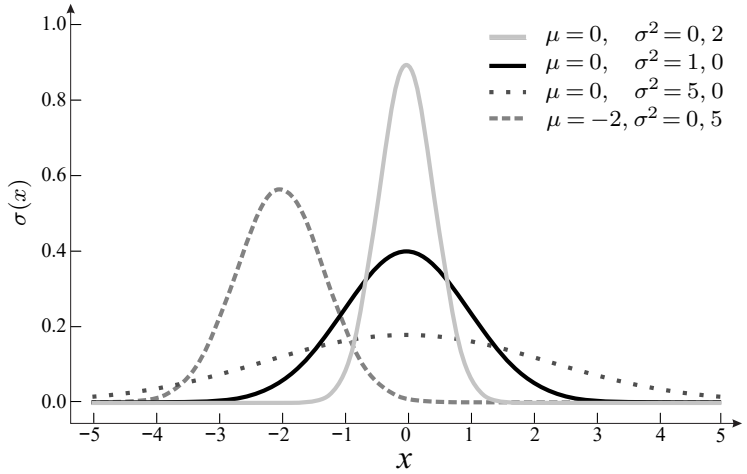


Abbildung 2.21: Gaußverteilungen mit verschiedenen Parametern

Nicht-parametrische Techniken definieren die Modellstruktur dagegen basierend auf gegebenen Daten. Sie setzen keine Kenntnis über die Wahrscheinlichkeitsverteilung voraus und die Parameter sind dementsprechend adaptiv (Desforges, Jacob und Cooper 1998). Ein Beispiel hierfür ist die in Abbildung 2.22 dargestellte Dichteverteilung eines Kerndichteschätzers (engl. Kernel Density Estimation (KDE)).

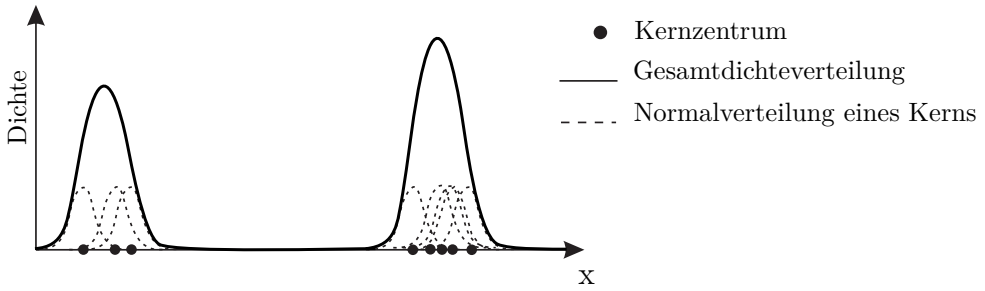


Abbildung 2.22: Dichteverteilung des Kerndichteschätzers (nach Zhou u. a. 2003)

Clusterbasierte Verfahren erkennen zunächst Cluster innerhalb der Daten (siehe Abbildung 2.23) und bewerten im Anschluss das Verhältnis eines Datenpunktes zu den Clustern. Die Verfahren lassen sich in drei Gruppen einteilen. Die erste Gruppe betrachtet die Möglichkeit, einen Datenpunkt einem Cluster zuzuordnen. Eine Anomalie wird erkannt, wenn keine Zuordnung zu einem Cluster möglich ist. Verfahren der zweiten Gruppe untersuchen den Abstand eines Datenpunktes von den Clusterzentren. Weit vom Zentrum entfernte Datenpunkte sind Anomalien. Die dritte

Gruppe berücksichtigt den Aspekt, dass anormale Datenpunkte selbst auch Cluster bilden können. Dabei wird angenommen, dass die Größe anormaler Cluster im Verhältnis zu den anderen Clustern sehr klein ist. (Chandola, Banerjee und Kumar 2009)

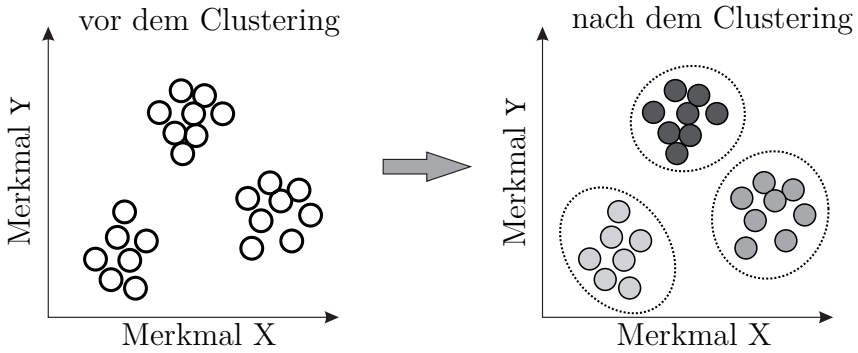


Abbildung 2.23: Distanzbasiertes Clustering mittels K-Means (nach JavaTpoint 2018)

Nächste-Nachbarn-basierte Verfahren basieren auf der Annahme, dass normale Datenpunkte in Gebieten hoher Dichte hinsichtlich der Anzahl der Datenpunkte auftreten. Es liegen dementsprechend viele normale Datenpunkte eng zusammen, während Anomalien weit entfernt von ihren nächsten Nachbarn auftreten. Verfahren, die auf den nächsten Nachbarn basieren, sind den clusterbasierten Verfahren sehr ähnlich. Während clusterbasierte Techniken jeden Datenpunkt in Bezug auf das nächstgelegene Cluster bewerten, werden bei den Nächste-Nachbarn-basierten Techniken die Datenpunkte in Bezug auf ihre lokale Nachbarschaft analysiert (Chandola, Banerjee und Kumar 2009). Dafür sind Dichte- oder Ähnlichkeitsmessungen notwendig. Anomaliewerte können entweder basierend auf der Distanz eines Datenpunkts zu ihren k nächsten Nachbarn wie in Abbildung 2.24 oder auf der relativen Dichte jedes Datenpunktes berechnet werden.

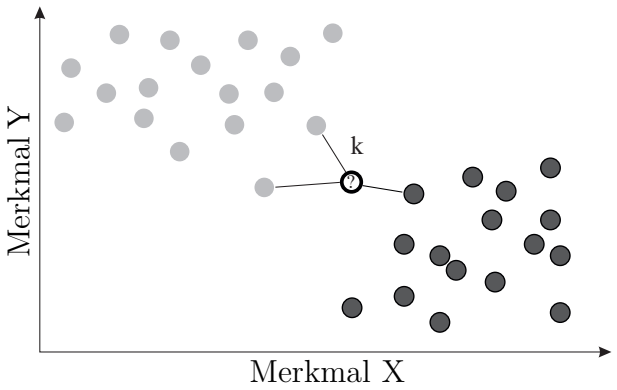


Abbildung 2.24: Abstand zu den k nächsten Nachbarn mit $k = 3$ (nach Mierswa 2017))

Klassifizierende Verfahren nutzen ein Modell, den sog. Klassifikator, um Datenpunkte bereits definierten Klassen zuzuordnen. Während bei den Ein-Klassen-Verfahren die Zuordnung nur in eine einzige Klasse erfolgt, stehen bei den Multi-Klassen-Verfahren mehrere Klassen zur Auswahl (siehe Abbildung 2.25). Ist die Zuordnung nicht möglich, wird der Datenpunkt als anormal gekennzeichnet. Wesentlicher Bestandteil der Verfahren ist die Erstellung des Klassifikators, welche beispielsweise regelbasiert oder an gelabelten Trainingsdaten geschehen kann.

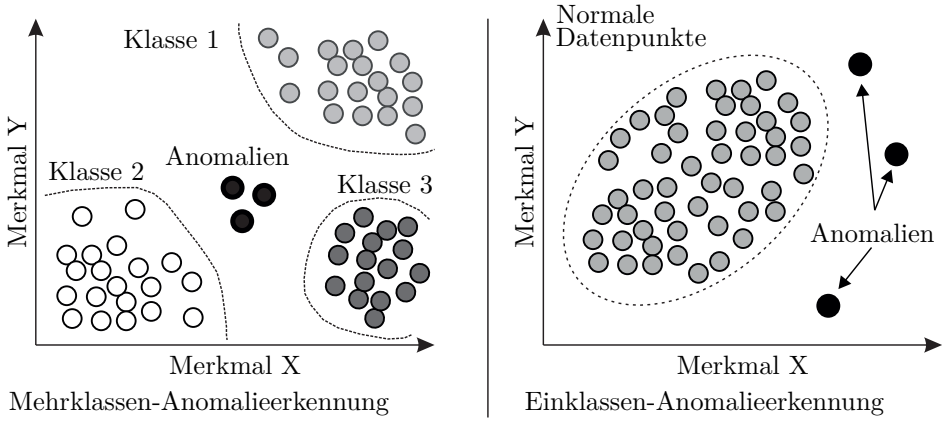


Abbildung 2.25: Unterscheidung von Mehr-Klassen- und Ein-Klassen-Klassifikations-Verfahren (nach Chandola, Banerjee und Kumar 2009)

Unterraumbasierte Verfahren basieren auf der Annahme, dass Anomalien in einem Unterraum mit reduzierten Dimensionen deutlich werden, schematisch in Abbildung 2.26 dargestellt. Daher sind unterraumbasierte Verfahren nur auf mehrdimensionale Datensätze sinnvoll anwendbar. Die Identifikation geeigneter Unterräume stellt die Kernaufgabe der Verfahren dar (Agovic u. a. 2008). Ein bekannter Vertreter ist die robuste Hauptkomponentenanalyse (rPCA, robust Principle Component Analysis).

Informationstheoretische Verfahren basieren auf dem zuvor genannten Grundsatz, dass eine Anomalie ein unerwartetes Ereignis ist. Daher ist der Informationsgehalt von anormalen Daten höher als von normalen Daten. Verfahren dieser Gruppe nutzen informationstheoretische Maße, wie sie unter anderem von (Lee und Xiang 2001) vorgestellt werden, zur Berechnung des Informationsgehalts und entscheiden auf dieser Basis über Anomalien.

Isolationsverfahren sind Verfahren, die Datensätze in einer Baumstruktur (engl. Isolation Tree) abbilden und Anomalien durch ihre durchschnittliche Pfadlänge erkennen. Der Unterschied zu anderen Verfahren ist, dass explizit die anormalen Datenpunkte isoliert werden, ohne dabei die normalen Daten zu definieren. (Liu, Ting

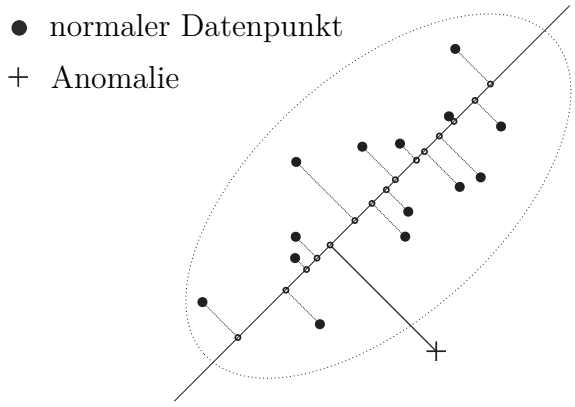


Abbildung 2.26: Unterraumbasierte Verfahren

und Zhou 2008) Das Vorgehen ist in Abbildung 2.27 dargestellt. Ein Datensatz wird zufällig unterteilt. Jeder Vorgang zur Unterteilung erzeugt in einem Baum eine neue Ebene. Der Vorgang wird solange wiederholt, bis alle Datenpunkte in einer eigenen Partition isoliert sind und somit in dem Baum Endknoten darstellen. Die Anzahl der Ebenen von der Wurzel bis zu einem Endknoten entspricht der Pfadlänge. Die Darstellung als Baum wird als Isolationsbaum bezeichnet. Durch mehrfache Wiederholung dieses Vorgehens werden mehrere Bäume erzeugt, die den Isolationswald ergeben. Damit kann eine durchschnittliche Pfadlänge im Isolationswald ermittelt werden. Ein Datenpunkt, der frühzeitig abgespalten wurde, besitzt eine kurze Pfadlänge. Anomalien besitzen deutlich kürzere Pfadlängen als die ermittelte durchschnittliche Pfadlänge des Isolationwalds.

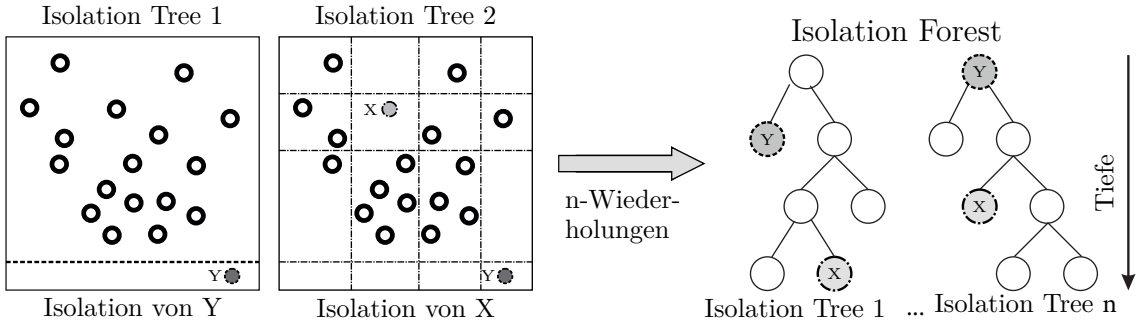


Abbildung 2.27: Erstellen eines Isolationwalds durch Partitionieren eines Datensatzes (nach Köhne 2019)

2.7 Distributed Ledger Technologien (DLT)

2.7.1 Einordnung verschiedener Technologien

DLT ist eine Technologie, die eine besondere Form der elektronischen Datenverarbeitung und Datenspeicherung ermöglicht. Der Hauptbestandteil ist eine dezentrale

Datenbank, das sogenannte *Distributed Ledger* oder verteilte Kassenbuch, über deren Inhalt zwischen den Netzwerkteilnehmern Konsens besteht. Eine DLT benötigt keine zentrale Instanz, die das Schreiben und Lesen von Daten ermöglicht. (Zivic, Ruland und Sassmannshausen 2019) Bevor eine Transaktion durchgeführt wird, also der Vorgang, bei dem ein Wert und/oder eine Nachricht dem verteilten Kassenbuch hinzugefügt wird, muss zwischen den Teilnehmern des Netzwerks Konsens herrschen. In einer Blockchain wird eine Transaktion als neuer Block angefügt. Die Unveränderlichkeit der Blockchain ist über die Verkettung eines Blocks über den kryptographisch sicheren Hash des vorherigen Blocks gegeben (siehe Abbildung 2.28). Zudem sind Veränderungen durch die starke Verteilung der replizierten Kassenbücher, über die Konsens herrscht, erschwert. Die Größe des Netzwerks beeinflusst somit auch dessen Sicherheit.

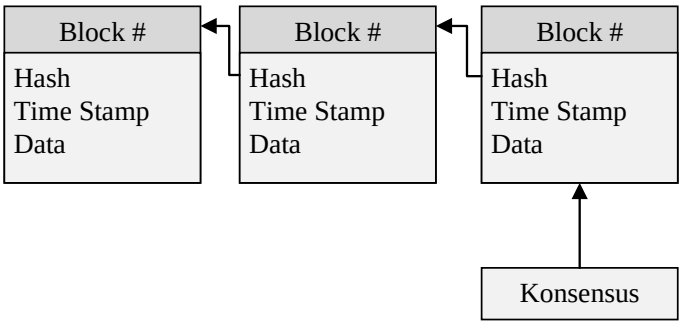


Abbildung 2.28: Prinzip einer Blockchain

Wesentliche Nutzen von DLTs sind (Firouzi, Chakrabarty und Nassif 2020):

- Ermöglichung der Interaktion von Teilnehmern zwischen denen kein Vertrauen besteht ohne Bedarf einer vertrauenswürdigen Drittpartei,
- zuverlässigere Datenspeicherung,
- Nachprüfbarkeit und Überprüfbarkeit durch permanente und unveränderliche Speicherung der Transaktionen,
- Transparenz über die Transaktionen,
- Verfügbarkeit durch Dezentralisierung,
- verbesserte Sicherheit und Integrität,
- Reduktion der Transaktionskosten,
- höhere Geschwindigkeit bei der Durchführung von Transaktionen und
- Ermöglichung von SC.

Bei der Ausprägung von DLTs wird unter anderem zwischen privaten und öffentlichen Netzwerken sowie Netzwerken mit eingeschränkten und uneingeschränkten Rechten unterschieden. In öffentlichen Netzwerken kann jedes Gerät teilnehmen, während in privaten Netzwerken nur ausgewählte Geräte als Teilnehmer hinzugefügt

werden. In einem uneingeschränkten Netzwerk kann jeder Teilnehmer die Einträge des verteilten Kassenbuchs lesen und schreiben. Ist es beschränkt, können Teilnehmer je nach Berechtigung nur schreiben, aber nicht lesen.

Zudem werden DLTs entsprechend der Art ihrer Datenspeicherung unterteilt. Der Großteil der Verfahren lässt sich den Blockchain-basierten oder gerichtet azyklischen Graphen (engl. Directed Acyclic Graph (DAG))-basierten Ansätzen zuordnen (Firouzi, Chakrabarty und Nassif 2020). In (Pervez u. a. 2018) werden der klassische Blockchain-basierte Ansatz und der DAG-basierte Ansatz gegenübergestellt. Die DAG-basierten Ansätze zeichnen sich im Allgemeinen durch ihre Skalierbarkeit, die hohe Geschwindigkeit der Transaktionen, der Abwesenheit von Minern und somit keine bzw. minimale Gebühren sowie quantenresistenten Algorithmen aus. Die in (Han u. a. 2020) vorgestellten Leistungstests an verschiedenen Blockchain-Rahmenwerken - konkret Hyperledger, Ripple, Tendermint und Corda - bestätigen deren mangelnde Skalierbarkeit mit der Forderung nach besseren Lösungen bezüglich des Blockchain-Konsensusproblems.

In (Pervez u. a. 2018) wird ein Vergleich zwischen verschiedenen DAG-basierten DLTs vorgestellt. Die am weitest verbreiteten davon - in Bezug auf die Marktkapitalisierung[1] - sind IOTA, gefolgt von Nano und NXT.

Neben der Skalierbarkeit überzeugt IOTA durch drei wesentliche Vorteile (Pinjala und Sivalingam 2019).

1. Keine Transaktionsgebühren und somit besonders geeignet für Mikrozahlungen.
2. Offline Transaktionen. Ist ein Teil des Netzwerks offline können Transaktionen dennoch ausgeführt werden. Nachdem der getrennte Teil des Netzwerks wieder angeschlossen ist, kann es sich wieder mit dem (Haupt-) Netzwerk verbinden.
3. Quantenresistenz, z.B. gegen Brute-Force-Attacken, durch den Einsatz des Winternitz Signaturverfahrens.

Die DLTs Nano und IOTA weisen bei dem Vergleich in (Pervez u. a. 2018) ähnliche Eigenschaften auf. Beide DLTs sind skalierbar, gebührenfrei, Open Source und besitzen Verfahren zur Verhinderung von Spam-Attacken. Nano hat im Durchschnitt etwas geringere Latenzen bei Transaktionen als IOTA. Die DLT NXT dagegen hat eine höhere durchschnittliche Bestätigungszeit von Transaktionen, ist nicht gebührenfrei und bietet keinen Schutz vor Spam-Attacken. Für den Kontext dieser Arbeit hebt sich IOTA bei dem Vergleich der zusätzlichen Merkmale der vorgestellten DAG-basierten DLTs ab. Diese sind insbesondere der Fokus auf IoT Applikationen, Datentransfer und den Maschine-zu-Maschine Transaktionen (Pervez u. a. 2018). IOTA ist spezialisiert auf den Einsatz im IoT, insbesondere im IIoT, und Maschinenwirtschaft, weshalb auch die Ausführung auf ressourcenlimitierten Endgeräten berücksichtigt wird (Jiang u. a. 2018; Wu, Zhou u. a. 2019; Schweizer u. a. 2020). Die IOTA Foundation kooperiert mit Partnern aus der Industrie und ist Teil der Gemeinschaft

[1] `https://coinmarketcap.com/coins/`. Abgerufen am 06.03.2023

des europäische Projekts „GAIA-X“[23]. Es steht zur Diskussion, ob IOTA ein Standard in der Maschine-zu-Maschine-Kommunikation und Industrie wird (Schweizer u. a. 2020).

Für die Ausarbeitung der genannten Anwendungsfälle wird daher die DLT IOTA ausgewählt. Trotz der genannten Vorteile von IOTA, wurde sich in verschiedenen Arbeiten gegen das bisherige IOTA-Rahmenwerk entschieden (Wu, Zhou u. a. 2019; Raschendorfer u. a. 2019). Grund dafür waren zwei gravierende Nachteile der ersten Version IOTA 1.0 (IRI), nämlich der Bedarf einer zentralen Instanz im Netzwerk und keiner Unterstützung von SC. Die kürzlich veröffentlichten neuen Versionen, IOTA 1.5 (Chrysalis) im Jahr 2020 und IOTA 2.0 (Coordicide) im Jahr 2021, führen zu mehreren Verbesserungen, sodass unter anderem die zwei genannten Mängel beseitigt sind und IOTA an Relevanz gewinnt. Diese aktuelle Entwicklung motiviert, IOTA als mögliches Rahmenwerk für IIoT-Anwendungsfälle in Betracht zu ziehen und den Einsatz von DLT zur Absicherung von Daten experimentell mit IOTA zu evaluieren.

2.7.2 Distributed Ledger Technologie IOTA

Aufgrund der dezentralen Struktur von IIoT-Netzwerken gibt es in der Regel keine zentrale Instanz, die als vertrauenswürdige dritte Partei fungieren könnte. Es ist erwünscht, dass nur bekannte Geräte am Netzwerk teilnehmen, deren Interessen aber nicht zwangsläufig übereinstimmen, beispielsweise Endgeräte unterschiedlicher Hersteller. Es wird daher für die Anwendung im industriellen Umfeld von privaten Netzwerken mit eingeschränkten Berechtigungen ausgegangen. Diese Anforderungen erfüllt die DLT IOTA mit ihrer transaktionsbasierten DAG-Struktur namens *Tangle*. Während bei einer Blockchain nur ein Block dem nächsten folgt, kann bei DAG-basierten Verfahren eine Vielzahl von Verknüpfungen eingegangen werden. Die Gegenüberstellung in Grafik 2.29 zeigt, dass dadurch Engpässe vermieden werden und die Effizienz, ausgedrückt durch die Anzahl der Nachrichten pro Sekunde (engl. Messages Per Second (MPS)), signifikant gesteigert und somit Skalierbarkeit erreicht wird. Eine höhere Aktivität führt bei dem Tangle zu einem höheren Durchsatz, d.h. mehr Validierungen, von Nachrichten. Bei einer Blockchain dagegen hat die Aktivitätsrate keinen Einfluss auf die Validierungsrate, was zu Engpässen führen kann.

IOTA ist als umfassendes Ökosystem zu verstehen, welches insbesondere für industrielle Anwendungsfälle entwickelt wird. Dementsprechend ist eine große Anzahl an Funktionalitäten verfügbar bzw. in Entwicklung. Die erste Version, IOTA IRI, wurde 2016 eingeführt und 2021 von der Version IOTA 1.5 abgelöst, welche eine Übergangslösung bis zu der Einführung von IOTA 2.0 darstellt. Die wesentlichen Merkmale und Unterschiede der bisherigen drei IOTA Versionen 1.0, 1.5 und 2.0 sind in Tabelle 2.1 zusammengefasst. Im Folgenden werden die für die Inhalte dieser Arbeit relevanten einzelnen Bestandteile, d.h. konkrete Typen von Netzwerk-Knoten

[2] `https://gaia-x.eu/`

[3] `https://blog.iota.org/iota-foundation-joins-the-gaia-x-community/`

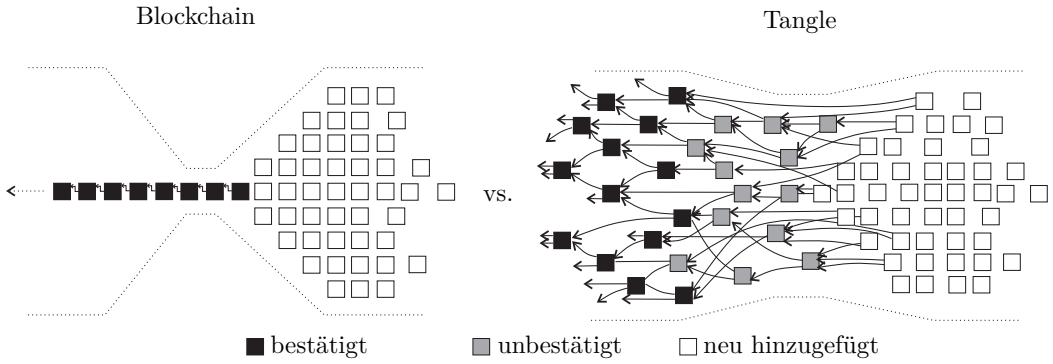


Abbildung 2.29: Gegenüberstellung von Blockchain und DAG Technologie nach (Popov 2018)

und Plug-Ins, der IOTA 1.5 und 2.0 Netzwerke kurz beschrieben. Für weitere Details zu IOTA wird auf die offiziellen IOTA Homepage[4], den IOTA Einsteigerguide[5] sowie den aktuellen, offiziellen und als Open Source verfügbaren Programmcode[6] verwiesen.

Tabelle 2.1: Vergleich der IOTA Versionen 1.0, 1.5 und 2.0 (Stand Juli 2023)

	IOTA 1.0	IOTA 1.5	IOTA 2.0
Name	IRI	Chrysalis	Coordicide
Freigabe Mainnet	2016	2021	noch nicht bekannt
Konsensus	Koordinator	Koordinator	FPC
Peering	manuell	autopeering	autopeering
Zentralisierung	ja	ja	nein
Transaktionsrate	5-20 TPS	$\sim$ 1000 TPS	noch nicht bekannt
Smart Contracts	nein	nein	ja

IOTA Chrysalis

IOTA Chrysalis, auch als IOTA 1.5 bezeichnet, ist eine Übergangsversion. Sie besitzt weiterhin eine zentrale Instanz, welche notwendig ist, um neue Nachrichten zu validieren. Ein Konsensus-Mechanismus ist daher für die Validierung von Nachrichten in dieser Version nicht erforderlich. Dennoch muss ein kryptographisches Rätsel, in

[4]https://blog.iota.org/
[5]https://iota-einsteiger-guide.de/archiv-iota-einsteiger-guide/
[6]https://github.com/iotaledger

diesem Fall Proof of Work (PoW), von dem Teilnehmer, der die Nachricht sendet, gelöst werden. Dies ist ein Mechanismus zur Verhinderung von Spam-Attacken.

Hornet-Knoten Die Hornet-Knoten sind die Basisknoten, die das IOTA 1.5 Netzwerk bilden. Sie können Nachrichten senden und, entsprechend ihrer Berechtigung, den Tangle lesen, mit ihren Nachbarn kommunizieren und das lokale Kassenbuch synchronisieren.

Koordinator Plug-In Die zentrale Instanz, der sogenannte Koordinator, stellt einen Basisknoten mit aktiviertem Koordinator Plug-In dar. Er ist für die Validierung der Nachrichten zuständig, wodurch auf aufwendige Konsensus-Algorithmen auf den Basisknoten verzichtet werden kann und dadurch der Ressourcenbedarf der Basisknoten reduziert ist. Der Koordinator generiert Meilensteine, welche vertrauenswürdige zero-value-Nachrichten, also Nachrichten ohne Wert, darstellen, aber globale, unveränderliche Zeitstempel besitzen. Durch die Meilensteine werden wiederum die weiteren Nachrichten im Netzwerk validiert.

Eintrittsknoten Der Eintrittsknoten (engl. Entry Node) ist sowohl in IOTA 1.5 als auch 2.0 Netzwerken obligatorisch, um neue Teilnehmer mit bestehenden Knoten zu verknüpfen und dadurch das Netzwerk aufzubauen. Dieser Knoten ist die Grundlage für das automatische Verknüpfen von Knoten (engl. autopeering), da er von allen Knoten die Kontaktinformationen besitzt.

Permanode-Funktionalität Da die Größe des Tangles begrenzt ist und dieser regelmäßig gestutzt wird (engl. Pruning), sind die Tangle-Nachrichten zeitlich nicht unbegrenzt verfügbar. Aus diesem Grund stellt IOTA die Permanode Funktionalität bereit, die ein langfristiges, dezentrales und unveränderliches Abspeichern von Nachrichten ermöglicht. Zwei Netzwerk-Bestandteile, der Chronicle-Knoten und die verteilte Datenbank SkyllaDB, sind hierfür notwendig.

Chronicle-Knoten Dieser Knoten ist die Hauptkomponente der Permanode-Funktionalität, da er die Schnittstelle zwischen dem Tangle und der SkyllaDB darstellt und das Schreiben von Daten aus dem Tangle in die Datenbank ermöglicht. Er ist aktuell (Stand 10/2021) nur für IOTA 1.5 verfügbar.

SkyllaDB Die SkyllaDB ist eine verteilte, unveränderliche Datenbank. Sie dient dem langfristigen Abspeichern von Tangle-Nachrichten und ist von dem Pruning nicht betroffen. Die Datenbank ist verteilt, da sie auf mehreren Geräten multipliziert abgelegt werden kann.

IOTA Coordicide

IOTA Coordicide, auch als IOTA 2.0 bezeichnet, ist die aktuelle Version und derzeit (Stand Juli 2023) nur als Entwickler-Version verfügbar. Der zuvor genannte Koordinator entfällt, weshalb ein Konsensus-Algorithmus, in diesem Fall Fast Probabilistic Consensus (FPC), erforderlich ist und von den Basisknoten berechnet wird. Jede neue Nachricht (sogenannter Approver) referenziert auf zwei Vorgänger-Nachrichten (sogenannte Parents), mit der sie sich verknüpft und die sie genehmigt. Noch nicht genehmigte Nachrichten werden Tips genannt. Abgesehen von der vollständigen De-

zentralität ist die SC-Funktionalität ein wesentlicher Fortschritt. Hierfür wird ein zusätzliches, überlagertes Netzwerk aus sogenannten Wasp-Nodes aufgebaut.

GoShimmer-Knoten Diese Knoten sind, analog zu den Hornet-Knoten in IOTA 1.5, die Basisknoten in IOTA 2.0. Allerdings ist die Funktionalität um den wesentlichen Aspekt, dem Validieren von Nachrichten, erweitert. Dies geschieht durch den FPC-Algorithmus.

Smart Contracts SC sind im Gegensatz zu den klassischen, manuell erstellten Rechtsverträgen deterministisch. Ein SC ist ein Programmcode, welcher ausgeführt wird, sobald eine bestimmte Bedingung erfüllt ist. Während es bei klassischen Verträgen zu Rechtsstreiten oder Verzögerungen kommen kann, ist dies bei SC nicht möglich, da sie vollständig automatisiert sind. Dies ist ein wesentliches Merkmal und die Eigenschaft, die SC so relevant für eine Vielzahl von Anwendungen macht. Ein SC ist als eine unveränderliche Zustandsmaschine zu verstehen. Der Zustand eines SC enthält relevante Informationen. Die Ausführung eines SC führt zu einem Zustandsübergang, welcher in einer Kette von Zustandsübergängen aufgezeichnet ist. Die Verknüpfung der Kette mit dem Tangle gewährleistet die Unveränderlichkeit.

Wasp-Knoten Es ist zwischen *on-chain* und *off-chain* SC zu unterscheiden. Während bei der bekannten DLT Ethereum on-chain SC ausgeführt werden und das Hauptnetzwerk dadurch belastet wird, erfolgt die Ausführung bei IOTA durch ein Second-Layer-Protokoll in einem überlagerten Netzwerk aus sogenannten Wasp-Knoten, wobei jeder Wasp-Knoten mit einem Basisknoten des Tangles verknüpft ist. Die Wasp-Knoten bilden untereinander ein Netzwerk. Ein SC wird durch ein Komitee aus Wasp-Knoten ausgeführt. Die Anzahl der Knoten, also die Größe des Komitees, sowie das erforderliche Quorum und die Wahl der Knoten, beispielsweise abhängig von ihrer Reputation, kann der Eigentümer eines SC selbst festlegen. Die Ausführung eines SC geschieht wie folgt: Die Komitee-Knoten hören die Nachrichten auf dem Tangle mit. Wird die durch den SC vorgegebene Bedingung durch eine Transaktion, die sogenannte Anfragetransaktion, erfüllt, entscheidet jeder Knoten für sich, ob er zustimmt. Dies macht den SC deterministisch, da - wie bei einer Blockchain - die nächste Transaktion des Zustands mit der vorherigen Transaktion des Zustands und der Anfragetransaktion verknüpft ist. Das erzeugt eine Kette, aus der nachvollziehbar hervorgeht, welche Anfragetransaktionen zu welchen Zustandsänderungen führten.

KAPITEL 3

Konzept

In Kapitel 3 wird zunächst das methodische Vorgehen beschrieben. Anschließend wird die allgemeine Forschungsfrage konkretisiert. Aus der Analyse der Datenflüsse und des Zielsystems werden Randbedingungen und Anforderungen an die Ausarbeitungen aufgestellt und abschließend die verfolgten Lösungsansätze vorgestellt.

3.1 Methodik

Die Methodik dieser Arbeit folgt dem in Abbildung 3.1 abgebildeten Prozess. Die sechs Schritte werden verallgemeinert, sodass sie auf die Entwicklung von Applikationen zur Optimierung von Datenflüssen anwendbar sind.

1. Der erste Schritt ist das Verständnis über die Problemstellung und die Analyse des zu erreichenden Ziels.
2. Darauf folgt im zweiten Schritt die Analyse der Randbedingungen. Diese erfordert in der Datenverarbeitung Wissen über die Daten selbst und das Ökosystem.
3. Der dritte Schritt beschreibt die Datenvorverarbeitung.
4. Die nachfolgende Modellierung umfasst die Wahl einer geeigneten Methode, das Vorgehen zum Training, Testen und Validieren des Modells, die Erstellung des datenverarbeitenden Modells selbst sowie die technische Modellbewertung.
5. Der fünfte Schritt ist die Evaluation der Ergebnisse und Abgleich mit der Zielsetzung. Das Vorgehen ist als iterativer Prozess zu sehen. Sind die Ergebnisse der Evaluation nicht zufriedenstellend, so ist wieder bei Schritt 1 zu beginnen.
6. Anschließend folgt der reale Einsatz des Modells.

Diese Methodik ist angelehnt an den CRISP-DM, einem branchenübergreifenden Standardprozess für Data Mining (Chapman u. a. 2000).

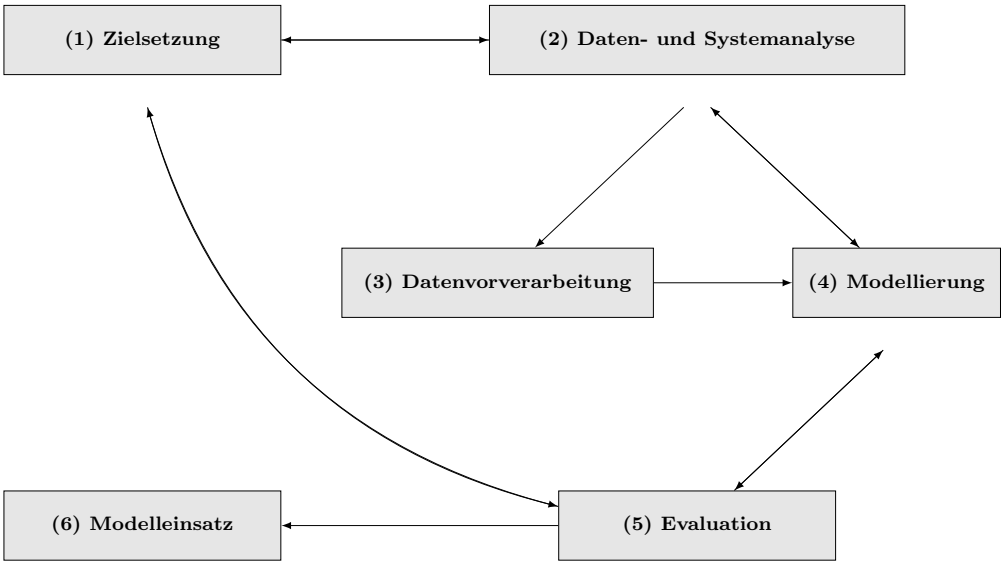


Abbildung 3.1: Vorgehen bei der Erstellung der Arbeit

3.2 Konkretisierung von Forschungsfragen

Der erste Schritt des erläuterten Vorgehens ist das Verständnis der in Kapitel 1 beschriebenen Problemstellung. Dazu wird die allgemeine Forschungsfrage konkretisiert.

In der Industrie steigt die Datenmenge und die Anzahl digitaler Geschäftsmodelle rasant, während die Infrastruktur nicht im gleichen Maß ausgebaut wird. Aus diesem Grund wird die Vermeidung von Ressourcenengpässen mittels Algorithmen als Alternative zu dem Ausbau von Infrastruktur untersucht. Es wird die Datenverarbeitung am Netzwerkrand, d.h. nahe der Datenerfassung, betrachtet, da Cloud-Dienste hinsichtlich Latenzen, Kosten und Datensicherheit nachteilig sind. Latenzen werden reduziert, da gewisse Informationen nur mit zeitlicher Nähe zu ihrer Erfassung relevant sind, um beispielsweise im Rahmen einer Zustandsüberwachung und präventiven Instandhaltung reagieren zu können. Davon abgeleitet werden die folgenden vier Forschungsfragen spezifiziert, deren Beantwortung das Ziel der Arbeit darstellt.

- FF1: Kann durch die Anwendung von informationstheoretischen Methoden auf industriellen Endgeräten der Verlust von Informationen verhindert werden?

- FF2: Können Ressourcenengpässe in datenflussverarbeitenden Systemen am Netzwerkrand mittels ML verhindert werden?

- FF3: Gibt es eine geeignete Technologie, die die Unveränderlichkeit von Daten und Informationen dezentral im IIoT absichern und Vertrauen schaffen kann?
- FF4: Ist es möglich, bei der Verarbeitung von Datenflüssen im IIoT auf Cloud-Dienste zu verzichten?

3.3 Daten- und Systemanalyse

In diesem Abschnitt werden das System, in dem die Datenflussoptimierung stattfindet, sowie die Daten betrachtet. Entsprechend der beschriebenen Drei-Schichten-Architektur in Abschnitt 2.3 ist ein Datenfluss-System in Logik, Middleware und dem Netzwerk einzuteilen. Abbildung 3.2 beschreibt die Bedeutung der einzelnen Schichten für die Problemstellung. Das Netzwerk ist dabei nicht zu verändern, sodass die Randbedingungen, die es an die Datenflussoptimierung stellt, zu untersuchen sind. In der Logik und Middleware sind die Optimierungspotenziale zu analysieren und umzusetzen.

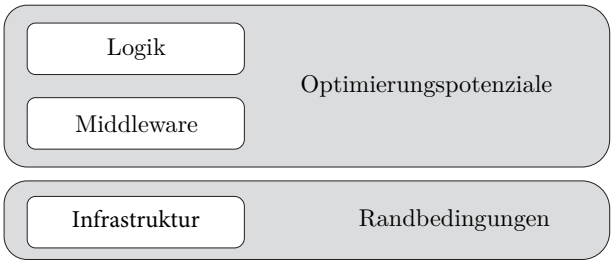


Abbildung 3.2: Einordnung der Datenflussoptimierung in die Architektur von Datenfluss-Systemen

Analyse der Daten

In dieser Arbeit steht im Gegensatz zum Data Mining nicht die Analyse von Big Data, also großen Datensätzen in Datenbanken, sondern die Optimierung von Datenflüssen im Mittelpunkt. Während Big Data eine bereits, beispielsweise in einer Datenbank, abgespeicherte Datenmenge beschreibt, stellen Datenflüsse eine unendliche Folge von Tupeln dar und erzeugen somit Zeitreihen. Unter Datenfluss wird im Rahmen dieser Arbeit eine Menge Daten, die zwischen Teilnehmern in einem Netzwerk übertragen wird, verstanden. Dabei gibt es keine Einschränkung hinsichtlich der Richtung der Kommunikation oder der Anzahl der Teilnehmer in der Kommunikation.

Die in Kapitel 2.2 beschriebenen Merkmale der Analysen von Big Data (5Vs) werden denen der Verarbeitung industrieller Datenflüssen gegenübergestellt.

- Volumen: Bei der Verarbeitung von Datenflüssen, auch nach Aggregation mehrerer Datenflüsse, handelt es sich um ein verhältnismäßig geringes Volumen.

- Vielfalt: Die Vielfalt der Datentypen und -quellen ist gegeben. Sie wird in dieser Arbeit durch die Einschränkung auf strukturierte Sensorsignale reduziert.
- Geschwindigkeit: Die Anforderung an die Geschwindigkeit von Datenerfassung und -verarbeitung ist im industriellen Umfeld besonders hoch, da Signale mit hohen Abtastraten erhoben werden und die Ergebnisse der Auswertungen zeitnah verfügbar sein müssen.
- Wahrhaftigkeit: Unsicherheiten bezüglich der Daten und Datenqualität sind durch die permanente Signalaufnahme mit konstanter Abtastrate und der Betrachtung nahe am Ursprung der Datenerfassung im Allgemeinen gering.
- Wert: Der Wert der Informationen in industriellen Daten ist unter Umständen nur bei einer zeitnahen Verarbeitung gegeben. Daher kann die Verarbeitung von Datenflüssen, welche zeitnah erfolgt, höhere Werte erzielen als Big Data.

Aus den beschriebenen Unterschieden bei der Verarbeitung der transienten und flüchtigen Datenflüsse gegenüber der Verarbeitung von großen Datensätzen, stellen sich die nachfolgenden Anforderungen an die Datenflussverarbeitung (nach Domingos und Hulten 2001; Gomes u. a. 2019):

1. Die benötigte Zeit zur Verarbeitung des Datenpunktes x_t soll möglichst gering und konstant sein. Um Echtzeitfähigkeit zu gewährleisten, muss die Verarbeitung abgeschlossen sein, bevor ein neuer Datenpunkt x_{t+1} zur Verfügung steht. Echtzeitfähig bedeutet hier, dass das System hinsichtlich der Verarbeitungszeit deterministisch ist und immer mit dem aktuellsten Datenpunkt arbeitet.
2. Der Algorithmus muss sich kontinuierlich anpassen können, ohne mehr Speicherplatz für historische Daten zu nutzen. Das heißt, der benötigte Speicherplatz darf nicht steigen.
3. Der Algorithmus muss ohne Zutun von außen, z.B. ohne vorherige Kennzeichnung der Daten und Abstimmung der Parameter, ausführbar sein.
4. Es muss jederzeit ein nutzbares Berechnungsmodell verfügbar sein, da anzunehmen ist, dass der Datenfluss unendlich ist und kein Ende erreicht wird.
5. Der Algorithmus muss sich dynamischen Änderungen anpassen können.
6. Der Algorithmus muss ein sogenannter One-Pass-Algorithmus sein, welcher die Eingabedaten nur einmal liest, und darf daher nur von den aktuell verarbeiteten Daten abhängen.

Analyse des Systems

Das betrachtete Netzwerk für die Datenflussoptimierung in der I4.0 ist das IIoT. Aus den oben genannten Gründen wird das Netzwerk nicht verändert und gibt damit die Randbedingungen vor. Das IIoT hat im Allgemeinen die Topologie eines vermaschten Netzwerks und zeichnet sich durch die Vielzahl an Teilnehmern aus. Die Ressourcen der Endgeräte am Netzwerkrand sind ein limitierender Faktor. Abbildung 3.3 gibt

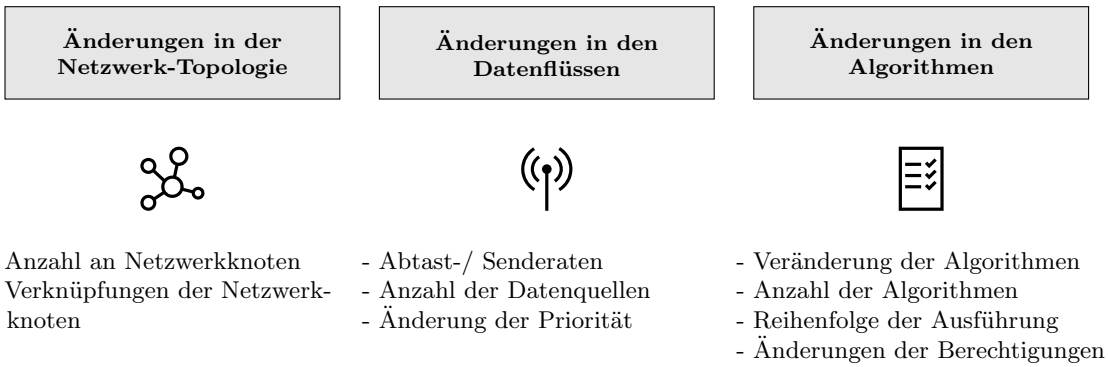


Abbildung 3.3: Systematische Übersicht der dynamischen Änderungen im IIoT (nach Rosenberger, Urlaub, Rauterberg u. a. 2022)

eine systematische Übersicht über die zu berücksichtigen dynamischen Änderungen. Sie sind in drei Kategorien unterteilt. Die erste umfasst dynamische Änderungen in der Topologie, während die zwei weiteren Kategorien die Änderungen der Netzwerkaufgaben, also der Datenübertragung und den auszuführenden Applikationen, berücksichtigen.

Aus der beschriebenen Anwendung in der I4.0-Umgebung ergeben sich, zusätzlich zu den Anforderungen durch die Anwendbarkeit auf Datenflüsse, Anforderungen an die eingesetzten Verfahren:

1. Agnostik hinsichtlich des ausführenden Endgeräts

   a) Komplexität
   Die verfügbare Rechenkapazität der Endgeräte in der Anlage ist beschränkt, weshalb die Anzahl und Komplexität der ausgeführten Rechenoperationen gering sein soll.

   b) Speicherbedarf
   Die Endgeräte in der Anlage sind ebenfalls hinsichtlich der Speicherkapazität begrenzt, sodass nur Algorithmen mit begrenzten Speicherbedarf sinnvoll einsetzbar sind.

2. Anwendbarkeit auf Datenflüsse
   Der Algorithmus soll auf Datenflüsse anwendbar sein und daher die im Abschnitt zur Analyse der Daten genannten Anforderungen erfüllen.

3. Agnostik hinsichtlich der industriellen Anlage
   Die Algorithmen sollen unabhängig von der Art des zu verarbeitenden Signals sowie dem Aufbau der Anlage gültig sein.

4. Geringer manueller Parametrierungsaufwand

Das erste Kriterium ist entscheidend dafür, wieviele Endgeräte zu der Datenflussverarbeitung beitragen können. Je geringer die Komplexität und der Speicherbedarf sind, desto größer ist die Anzahl der Endgeräte, die fähig sind, die Algorithmen auszuführen. Die Kriterien 2 bis 4 sind dagegen grundlegend für den Einsatz in der beschriebenen Gesamtarchitektur.

3.4 Lösungsansätze

Für die Untersuchung und Beantwortung der genannten spezifischen Forschungsfragen sind die in Abbildung 3.4 dargestellten drei Lösungsansätze identifiziert, welche in dieser Arbeit betrachtet werden. Dabei sind insbesondere die in Abschnitt 1.1 genannten Herausforderungen der stark limitierten Ressourcen und begrenzten zulässigen Verarbeitungszeiten zu berücksichtigen.

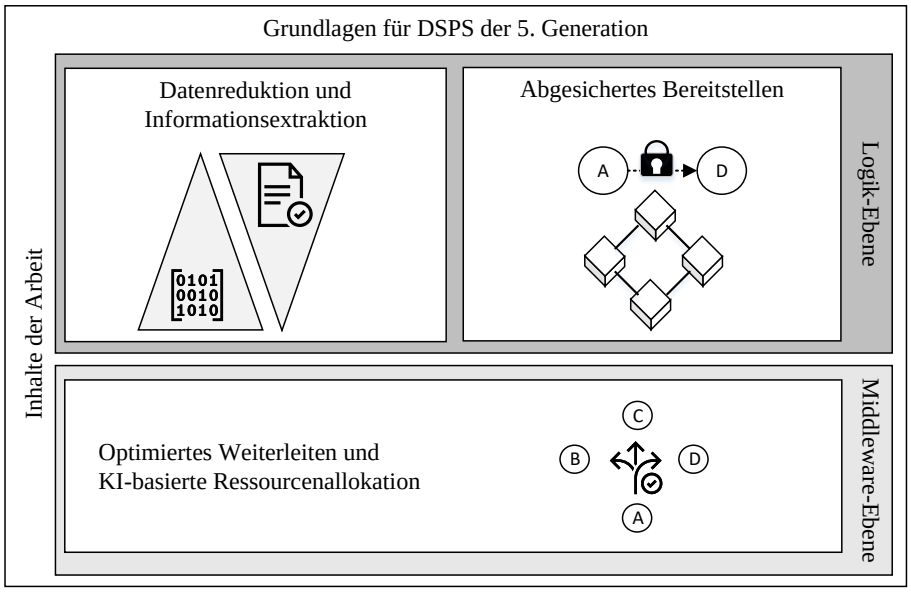


Abbildung 3.4: Inhalte der Arbeit

1. Datenreduktion und Informationsextraktion
   Dieser Ansatz verfolgt die Verringerung der Datenmenge mit dem Ziel Ressourcen zu schonen. Die in den Daten enthaltenen Informationen sind frühzeitig mittels Datenanalysen geringer Komplexität aus den Rohdaten zu extrahieren. Hierfür werden Ansätze der Datenkompression und Anomalieerkennung betrachtet sowie der kombinierte Modelleinsatz für beide Anwendungsgebiete.
2. Absicherung der Daten und Informationen
   Die Schaffung einer Vertrauensbasis, die Absicherung vor unberechtigten Zugriffen sowie die Gewährleistung der Unveränderlichkeit von Daten in dezentralen,

heterogenen Netzwerken werden in dieser Arbeit durch ein DLT System hergestellt. Dazu wird die Anwendbarkeit der explizit für das IoT vorgesehenen DLT IOTA untersucht.

3. Optimierung der Ressourcenausnutzung
   Die optimale Nutzung der verfügbaren limitierten Ressourcen der IIoT-Netzwerkteilnehmer soll die Datenverarbeitung nahe des Datenursprungs ermöglichen. Dazu werden zwei interagierende MAS zur intelligenten Allokation der Ressourcen basierend auf RL vorgestellt.

Der allgemeine dreischichtige Aufbau eines DSPS nach (Liu und Buyya 2020), siehe Abbildung 2.5, ist in Abbildung 3.5 um die Anforderungen der I4.0 und die konkrete Inhalte dieser Arbeit erweitert. Als Infrastruktur wird von einem vermaschten, dynamischen IIoT-Netzwerk ausgegangen. Die Middleware wird durch das MAS um die optimierte, intelligente Ressourcenauslastung erweitert. Die Applikationen der Logik-Schicht bestehen aus vier Bestandteilen: Quelle und Senke, dem Operator sowie dem Datenfluss selbst (Amarasinghe u. a. 2018). In dieser Arbeit werden Operatoren für die Datenkompressionen, Anomalieerkennung sowie DLT und SC für den Einsatz auf der Logik-Schicht, die also für die Anwendung auf Datenflüsse und der Ausführung auf Endgeräten geeignet sind, vorgestellt. Bei der Eignung ist auf die oben genannte Forderung nach Echtzeitfähigkeit zu achten.

Der dritte Schritt des allgemeinen Vorgehens nach CRISP-DM, die Datenaufbereitung, entfällt entsprechend den genannten Anforderungen an eine Datenflussverarbeitung. Die Kapitel 4-6 beschreiben die weiteren Schritte der Modellierung unter Berücksichtigung der Randbedingungen und Anforderungen sowie die Evaluation.

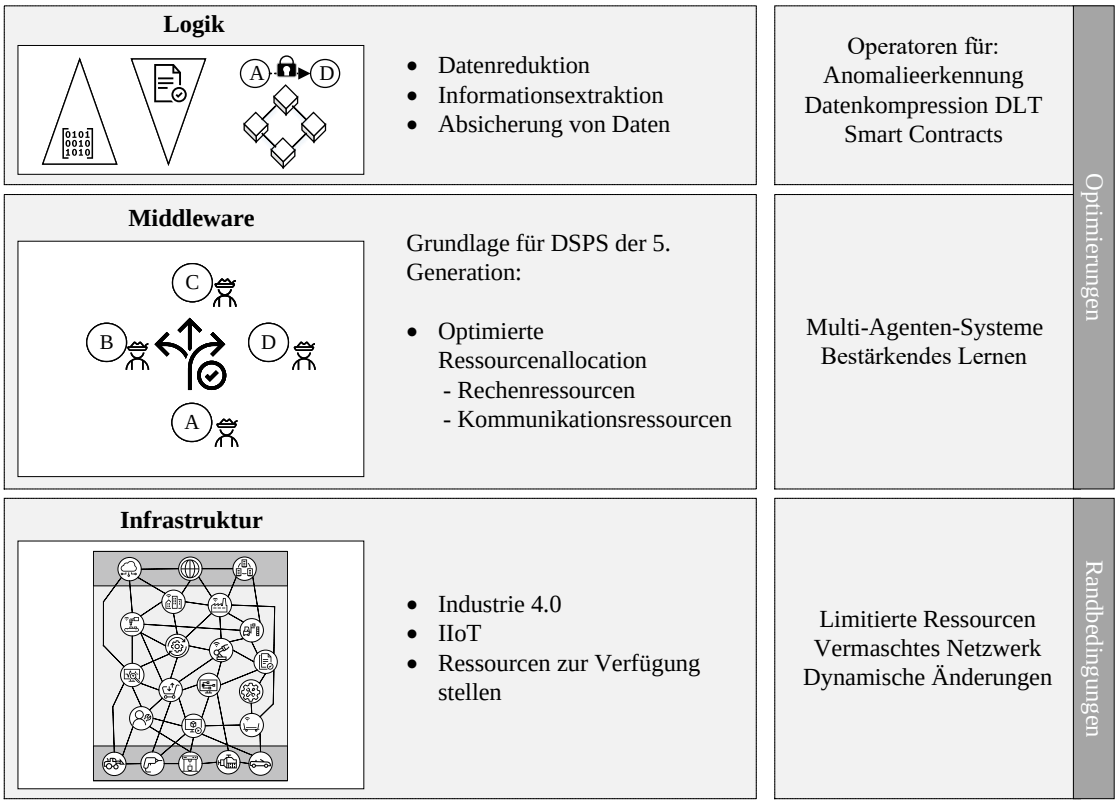


Abbildung 3.5: Einordnung der Inhalte der Arbeit in die Drei-Schichten-Architektur eines Datenflusssystems (nach Liu und Buyya 2020; Rosenberger, Selig, Ristic u. a. 2023)

KAPITEL 4

Datenreduktion und Informationsextraktion

In Kapitel 4 werden Methoden zur Reduktion der Datenmenge unter Vermeidung von Informationsverlusten in der Logik-Ebene vorgestellt. Hierfür wird einerseits die Kompression von Datenflüssen, d.h. die Entfernung von Redundanzen und Irrelevanzen, untersucht und andererseits die Extraktion von Informationen in Form von Anomalien. Außerdem wird die Möglichkeit eines kombinierten Modelleinsatzes erörtert. Die Ergebnisse der Evaluation werden abschließend diskutiert.

4.1 Wahl der Verfahren zur Datenreduktion und Informationsextraktion

Die erfassten Daten beinhalten eine große Menge heterogener und redundanter Daten (Wang und Zhang 2020). Daraus folgt, dass die relevanten Daten nur einen kleinen Teil darstellen, da aus Redundanzen keine neuen Erkenntnisse gewonnen werden können. Das Ziel ist dementsprechend die Reduktion von irrelevanten Rohdaten und die Extraktion von relevanten Informationen zeitnah nach der Erfassung der Rohdaten, also noch in den Datenflüssen und nicht erst in abgespeicherten Datensätzen. Dies ist notwendig, um die Übertragung und die Analysen zu vereinfachen und dadurch Latenzen sowie Kosten zu reduzieren.

Hierfür werden insbesondere zwei relevante Ansätze aus den Gebieten der Datenkompression und Anomalieerkennung identifiziert. Die Reduktion der Datenmenge mittels Kodierung und unter Berücksichtigung der Anforderung, keine Informationen zu verlieren, ist der Inhalt des Gebiets der Datenkompression, welche in dieser

Arbeit auf die Kompression von Datenflüssen durch die Endgeräte in der Anlage angewandt wird. Für die Extraktion relevanter Informationen ist zunächst die Motivation hinter der Erfassung und Analyse der Daten zu betrachten, um zu entscheiden, welche von Relevanz sind und welche nicht. Entsprechend Abbildung 4.1 sind dabei zwei Fälle, die prüfende und die explorative Datenanalyse, zu unterscheiden. Die prüfende Analyse umfasst unter anderem die Zustandsüberwachung und die davon abgeleitete vorausschauende Instandhaltung. Dadurch können der Betrieb von Produktionsanlagen und Prozesse, aber auch die Produkte selbst optimiert werden oder dem Kunden digitale Services angeboten werden. Elementarer Bestandteil der Zustandsüberwachung, also der Überwachung, ob der Zustand im Normalbereich liegt oder sich davon entfernt, ist die Anomalieerkennung. Anomalien sind daher relevante Informationen. Sie zeichnen sich häufig dadurch aus, zeitkritisch zu sein, da die Ursache von Anomalien in Produktionsanlagen zeitnah untersucht und ggf. behoben werden muss, um Produktionsausfälle zu verhindern. Die explorative Datenanalyse dagegen umfasst die Inhalte des sogenannten Data Mining, also der Gewinnung neuer Erkenntnisse durch die Analyse großer Datensätze. Es unterscheidet sich durch seine Anforderungen. Die Analysen basieren auf Big Data und sind weniger zeitkritisch, dafür allerdings sehr zeit- und rechenintensiv und erfolgen meist auf leistungsstarken Servern. Die hierfür benötigten (Roh-)Daten werden daher nicht verworfen, aber die zu übertragende Datenmenge wird mittels Kompression reduziert.

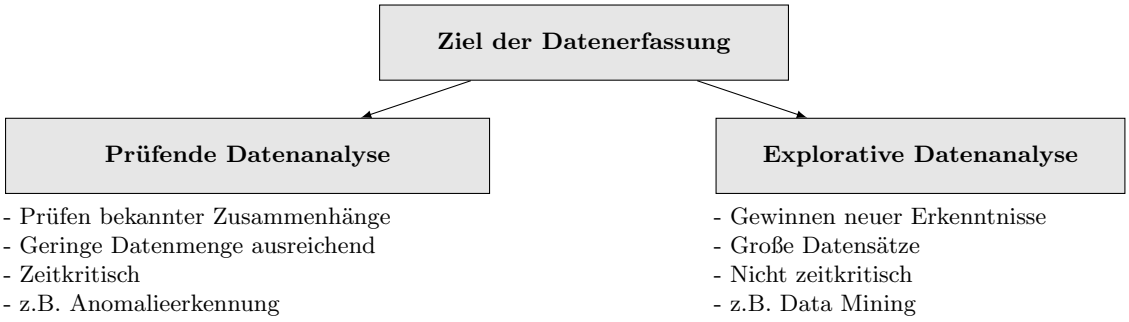


Abbildung 4.1: Einteilung der Ziele der Datenerfassung

Unter Berücksichtigung der in Kapitel 3 aufgestellten Anforderungen werden sowohl für die Anomalieerkennung als auch für die Datenkompression Verfahren des ML gewählt. Die Inhalte dieses Kapitels sind in Abbildung 4.2 zusammengefasst.

4.2 Stand der Technik

Die folgenden zwei Abschnitte umreißen den Stand der Technik zu den gewählten Ansätzen zur Datenreduktion, zum einen der Datenkompression mittels Autoencoder und zum anderen der Anomalieerkennung. Bei beiden Ansätzen liegt besonderer Fokus auf der Eignung für ressourcenlimitierte Endgeräte sowie der Anwendbarkeit auf Datenflüsse.

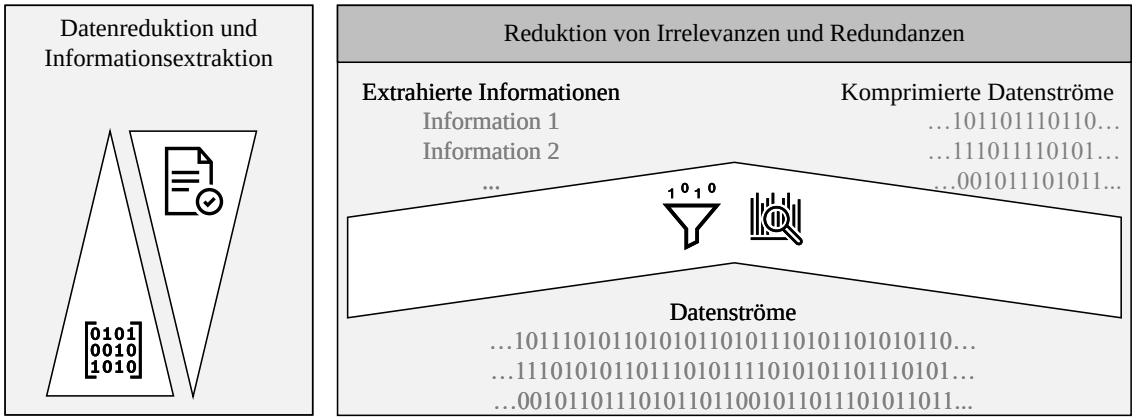


Abbildung 4.2: Datenreduktion und Informationsextraktion in der Logik-Ebene

4.2.1 Datenkompression

Bei der verlustbehafteten Kompression von Datenflüssen mittels Autoencodern ist es möglich, sowohl Korrelationen zwischen mehreren Signalen, aber auch innerhalb eines Signals für die Kompression zu nutzen. Dementsprechend wird ein Ansatz verfolgt, der sowohl auf univariate Datenflüsse als auch multivariate Datenflüsse anwendbar ist. Obwohl (Chiarot und Silvestri 2023) in ihrer Studie über die Datenkompression von Zeitreihen Autoencoder als möglichen Ansatz vorstellen, wird lediglich ein bestehendes Autoencoder-Modell (Hsu 2017) benannt. In diesem Abschnitt wird eine umfassendere Zusammenfassung der bereits existierenden Autoencoder-Modelle, nach der Struktur des Neuronalen Netz sortiert, gegeben.

Die einfachste Art von Neuronalen Netzen sind die vollvernetzen Feedforward-Netze. Ein vollvermaschter, vorwärtsgerichteter Feedforward-Autoencoder wird in (Ben Said u. a. 2017) für die Kompression von unterschiedlichen Datentypen vorgestellt. Dabei wird in der Encoder- und der Decoder-Seite jeweils eine Schicht für jeden Datentypen sowie eine weitere Schicht, in der die Darstellungen der verschiedenen Signale zusammengeführt werden, vorgesehen. Das Autoencoder-Modell ist dabei konkret auf medizinische Daten, genauer gesagt EEG und EMG Daten, ausgelegt und nicht allgemein gültig. Außerdem widerspricht die Spezialisierung auf multimodale Daten dabei grundsätzlich dem Einsatz auf UTS.

Neben dieser einfachen Art von Neuronalen Netzen gibt es weiterhin RNNs, welche sich insbesondere für die sequentielle Datenverarbeitung eignen, und daher für den Forschungsbereich der Zeitreihendaten-Kompression von großer Bedeutung sind. In (Zhang, Lee u. a. 2017) wird gezeigt, dass die Lernfähigkeit eines RNN-Autoencoders von der Varianz der Daten abhängig ist. Der von (Hsu 2017) vorgestellte dynamische Ansatz baut auf diesem Zusammenhang auf. Die Größe der zu komprimierenden Sequenz passt sich entsprechend der zu diesem Zeitpunkt bestehenden Varianz an, um den Rekonstruktionsfehler zu reduzieren und die Qualität der dekomprimierten Daten sicherzustellen.(Hsu 2017) Die Dekompression, d. h. die Rekonstruktion der Originaldaten, ist erforderlich, um die Daten nach der Kompression weiterzuver-

wenden, z. B. Datenanalysen durchzuführen. Das vorgestellte Autoencoder-Modell basiert auf Long-Short Term Memory (LSTM)-RNN. In der Evaluation findet es sowohl auf UTS als auch MTS Anwendung, allerdings ist die dynamische Anpassung der Fenstergröße durch eine iterative binäre Suche gestaltet und somit nicht echtzeitfähig. Die Herausforderung des Problems der verschwindenden beziehungsweise explodierenden Gradienten wird durch das von (Hochreiter und Schmidhuber 1997) vorgeschlagene Konzept des LSTM teilweise behoben. LSTM ist eine spezielle Architektur von RNN, welche eine Art Gedächtnis besitzt, durch die auch zeitliche Zusammenhänge in den Daten gespeichert werden können. LSTM kommt ebenfalls in dem Autoencoder-Modell zur Datenkompression von (Wong und Luo 2018) zum Einsatz, welches allerdings nur für den Einsatz in MTS geeignet ist und dessen Evaluation nur an hochdimensionalen Daten mit über 5000 Datenpunkten je Eingabe erfolgte.

Die dritte Art von Neuronalen Netzen sind CNNs. Die vollverknüpften Schichten werden in diesem Fall durch Faltungsschichten (engl. convolutional layers) erweitert, die speziell für die Extraktion von Mustern aus Daten geeignet sind. In (Yildirim, Tan und Acharya 2018) wird ein mehrschichtiger CNN-Autoencoder, bestehend aus mehreren aufeinanderfolgenden Faltungs- und Pooling-Schichten, vorgestellt. Das Modell ist dabei für die Kompression von EKG-Daten ausgelegt. Im Kontext der Zustandsüberwachung von Infrastruktur wird in (Ni, Zhang und Noori 2019) ebenfalls ein CNN-Autoencoder-Modell beschrieben, welches zur Datenkompression und Anomalieerkennung eingesetzt wird. Das Modell ist allerdings, im Gegensatz zu den Anforderungen dieser Arbeit, nur für eindimensionale Daten ausgelegt und wird mit einem Eingabevektor der Größe 1×2048 evaluiert. Dabei wird bestätigt, dass durch CNN-Autoencoder im Vergleich zu traditionellen verlustbehafteten Algorithmen bei gleichem Kompressionsverhältnis geringere Rekonstruktionsfehler erzielt werden.

Ein weiterer Autoencoder-Typ, welcher insbesondere von (Liu, Chen und Wang 2018) verfolgt wird, sind die Deep Belief Netze (engl. Deep Belief Network (DBN)), die aus Schichten einzelner beschränkter Boltzmann-Maschinen (engl. Restricted Boltzman Machines (RBMs)) aufgebaut sind. RBMs besitzen eine einfache Netzstruktur und zeichnet sich durch ihre Effizienz in der Berechnung aus. In (Liu, Chen und Wang 2018) wird die RBM-Autoencoder-Struktur zur Komprimierung von Sensorsignalen in einem WSN angewendet. Weiter ausgearbeitet wird dieser Ansatz in (Liu, Chen, Yan u. a. 2019), in dem ein gefaltetes DBN in Kombination mit einem Variational-Autoencoder für die Kompression von Signalen in WSN verwendet wird. Das vorgestellte Modell ist allerdings nur für UTS ausgelegt.

Es zeigt sich, dass kein echtzeitfähiger Ansatz existiert, der für die Kompression in Form von Mikrosequenzen aus sowohl für uni- als auch multivariate Datenflüsse geeignet ist. In Abschnitt 4.3 werden sowohl ein auf CNN als auch auf RNN basierende Autoencoder für den Einsatz zur Kompression von industriellen Zeitreihen vergleichend untersucht und auf die Anwendbarkeit von sowohl uni- als auch multivariaten Eingangsignale adaptiert.

4.2.2 Anomalieerkennung

Die Studien von (Chandola, Banerjee und Kumar 2009) und (Goldstein und Uchida 2016) geben einen umfassenden Überblick über ein breites Spektrum von Methoden zur Anomalieerkennung. Die Mehrzahl der existierenden Ansätze ist für die Verarbeitung von Datensätzen und nicht von Datenflüssen ausgelegt. Aufgrund der beschriebenen Unterschiede von Datensätzen und Datenflüssen, siehe Abschnitt 3.3, ist es in den meisten Fällen nicht möglich, diese Ansätze für die Verarbeitung von Datenflüssen in Echtzeit einzusetzen (Ahmad, Lavin u. a. 2017). In einer aktuelleren Studie geben (Salehi und Rashidi 2018) einen Überblick zur Anomalieerkennung in Datenflüssen und berücksichtigt Methoden basierend auf statistischen, clusterbasierten und Nächste-Nachbarn-Verfahren. Bei der Wahl von ML-basierten Verfahren ist generell zwischen überwachten und unüberwachten Techniken des ML zu unterscheiden. Überwachte ML Algorithmen benötigen gekennzeichnete Daten, die Grundwahrheit, zum Training und sind daher für kontinuierliche Lernaufgaben nicht geeignet (Ahmad und Purdy 2016). Für reale Anwendungen und zur Sicherstellung der Portabilität ist ausschließlich die Verfügbarkeit von Daten ohne Kennzeichnung anzunehmen (Goldstein und Uchida 2016; Gomes u. a. 2019). Wie (Ahmad, Lavin u. a. 2017) erwähnen, gibt es nur wenige existierende Ansätze, die die in Abschnitt 3.3 abgeleiteten Kriterien für die Echtzeitverarbeitung der Anomalieerkennung erfüllen.

Zusätzlich erkennen die meisten der existierenden Methoden nur eine Art von Anomalien. Insbesondere kollektive Anomalien unterscheiden sich stark von der Struktur der anderen Anomaliearten (Chandola, Banerjee und Kumar 2009), was die Verwendung eines Modells für alle Arten erschwert. Nichtsdestotrotz gibt es einige wenige Methoden, die in der Lage sind, mehr als einen Anomalietyp zu erkennen (Fisch, Eckley und Fearnhead 2019; Marteau 2021; Hundman u. a. 2018). Allerdings sind diese nicht oder weniger auf Datenflüsse anwendbar und können die oben genannten Kriterien nicht erfüllen. Die Kombination von zwei oder mehr Modellen als sogenannte Hybridmodelle ist ein häufig verwendeter Ansatz, der aber meist auf eine geringere Fehlerrate (Garg u. a. 2019) oder die Verarbeitung hochdimensionaler Daten (Song u. a. 2017) abzielt, anstatt auf die Erkennung mehrerer Anomaliearten. Aufgrund der höheren Komplexität sind Hybridmodelle auf Endgeräten mit begrenzten Ressourcen weniger geeignet.
Bisher gibt es keinen Algorithmus, der in der Lage ist, alle vier Arten von Anomalien durch die Verarbeitung von Datenflüssen auf industriellen Endgeräten zu erkennen. In der vorliegenden Arbeit wird dagegen die Erweiterung eines bestehenden Algorithmus zur Erkennung aller Anomaliearten unter Berücksichtigung der genannten Anforderungen ausgearbeitet.

Zusätzlich wird betrachtet, wie der kombinierte Modelleinsatz für sowohl Datenkompression als auch Anomalieerkennung eine ressourcenschonende, effiziente Erkennung von Anomalien ermöglicht. Das Vorgehen für den gemeinsamen Einsatz eines Modells für die Erzielung zusätzlicher Effizienzsteigerungen (nach Rosenberger, Bühren und Schramm 2021) wird in Abschnitt 4.5 vorgestellt.

4.3 Methoden zur Datenreduktion mittels Datenkompression

Die Datenkompression lässt sich wie in Abschnitt 2.5 genannt in verlustfreie und verlustbehaftete Verfahren unterteilen. Die Charakteristik der verlustbehafteten Kompression stimmt mit der Zielsetzung, nur relevante Informationen anstatt alle verfügbaren Daten zur Verfügung zu stellen, überein. Deshalb wird im Folgenden die Einsatzmöglichkeit von verlustbehafteter Kompression von Datenflüssen mittels ML untersucht und mit klassischen Verfahren verglichen.

Die Kompression von Datenflüssen unterscheidet sich von der Kompression von Datensätzen und birgt zusätzliche Herausforderungen, weshalb viele bestehende Verfahren nicht direkt auf die Kompression von Datenflüssen übertragbar sind. Wie es allgemein für Operatoren in DSPS gilt, gibt es auch bei der Kompression zwei Wege, um mit der Unendlichkeit von Datenflüssen umzugehen, die tupelweise Verarbeitung oder die Verarbeitung von Mikrosequenzen. Iterative Verfahren sind für die Verarbeitung von Datenflüssen nicht geeignet (Lindstrom und Isenburg 2006).

Autoencoder sind nicht für die tupelweise Verarbeitung geeignet, da die Kompression durch ihre Architektur, also der höheren Anzahl an Eingabewerten als Anzahl der Ausgabewerten der Engpassschicht, zustande kommt. Daher werden die eingehenden Daten geringfügig gepuffert, um anschließend die Mikrosequenzen zu komprimieren.

In industriellen Anwendungen, besonders in der Automatisierungstechnik, erfolgen die Datenerfassung und -weiterleitung meist mit hoher Frequenz und liegen im Bereich von Millisekunden. Aus diesem Grund sind die benötigten Zeiten für die Kompression und unter Umständen auch Dekompression zu beachten. Wie in Abschnitt 2.2 erläutert, liegen verschiedene Datentypen in der Industrie vor. Im Kontext von den in dieser Arbeit betrachteten Datenflüssen wird von Fließkommazahlen ausgegangen. Das ist zum einen darin begründet, dass Daten oberhalb der Steuerungsebene als Fließkommazahlen vorliegen und zum anderen in der Erwartung, dass höherwertige Sensoren zukünftig auch Fließkommazahlen übertragen. Die Gesamtzahl der möglichen auftretenden Werte ist bei Fließkommazahlen deutlich größer als beispielsweise bei der Kompression von Buchstaben. Verfahren mit Wörterbüchern dieser Größe können nur schlechte Kompressionsraten erzielen.

In einer vorangegangenen Studie (Rosenberger, Kübel und Rothfuß 2022) wird die Eignung von ML zur Kompression von Datenflüssen evaluiert und bestätigt. Dazu wird die Kombination von modernen Deep Learning Techniken mit der Autoencoder Struktur untersucht und mit einem klassischen Ansatz, der Diskreten Wavelet Transformation (engl. Discrete Wavelet Transform (DWT)), verglichen. Hierfür wurden zwei Verfahren ausgewählt. Das erste Verfahren ist in der Arbeit von (Liu, Chen, Yan u. a. 2019) für die Anwendung auf UTS in WSNs vorgestellt und basiert auf CNN. Der zweite Ansatz nach (Wong und Luo 2018) basiert auf einem RNN-Autoencoder und ist ursprünglich für hochdimensionale industrielle Daten entwickelt. Für beide Autoencoder-Modelle wird die Anwendung auf sowohl uni- als auch multivariate

Datenflüsse untersucht. Die Modelle und die vorgenommenen Anpassungen werden im Folgenden kurz beschrieben. Anschließend werden die Ergebnisse der Evaluation zusammengefasst. Die Modelle wurden im Hinblick auf die Qualität der Kompression mit unterschiedlichen Mikrosequenzlängen und Daten sowohl für den uni- als auch den multivariaten Fall getestet und mit dem Ergebnis der DWT verglichen sowie der Ressourcenverbrauch erfasst.

CBN-VAE

Modell: Das erste Modell (nach Liu, Chen, Yan u. a. 2019) basiert auf CNN, RBM und Variational Autoencoder-Schichten. Der detaillierte Aufbau ist in (Liu, Chen, Yan u. a. 2019) beschrieben und in Abbildung 4.3 zusammengefasst. In Abbildung A.1 im Anhang ist das Vorgehen bei der Kompression einer Einheit *D-CRMB*, welche aus einer gefalteten RBM sowie einer Poolingschicht besteht, schematisch dargestellt. Faltungsschichten werden verwendet, um Informationen und Korrelationen aus den Zeitreihen zu extrahieren. Die Kompression wird durch die Faltungsschrittweite, welche auf die Breite der Filtermatrix eingestellt ist, und die Max-Pooling-Schicht mit einer Max-Pooling-Größe von 2×1 erreicht. Auf mehrere Kompressionseinheiten folgt als Engpassschicht eine Variational Autoencoder-Schicht. Bei einem Variational Autoencoder wird die Wahrscheinlichkeitsverteilung der Eingabedaten erlernt, und es werden Werte aus dieser Verteilung abgetastet. Diese Werte entsprechen dem latenten Raum, der dann dekodiert wird. Der beschriebene Aufbau überzeugt laut (Liu, Chen, Yan u. a. 2019) durch die guten Kompressionsraten und den geringen Rechenaufwand.

Adaptionen: Das erweiterte Modell ist in der Lage, zusätzlich zu UTS auch MTS der Dimension M zu komprimieren. Dabei erhöht sich die Größe der Filtermatrix von $Schrittweite \times 1$ auf $Schrittweite \times M$. Weitere Schritte wie Max-Pooling und die Variational Autoencoder-Schicht skalieren ebenfalls für M-Dimensionen. Je nach der Korrelation innerhalb der Dimensionen könnten bei gleichbleibender Struktur weniger Verteilungen für den Variational Autoencoder verwendet werden, was die Kompressionsrate erhöht. Der angepasste Filter ist in Abbildung 4.4 dargestellt.

Training: Die einzelnen Schichten werden hintereinander verschachtelt und mittels Backpropagation trainiert. Die Verlustfunktion 4.1 setzt sich aus dem mittleren quadratischen Fehler (engl. Mean Squared Error (MSE)) und der Kullback-Leibler-Divergenz zusammen. Die beiden Komponenten werden miteinander verrechnet und ihr Verhältnis zueinander wird mit Hilfe eines Gewichtungsfaktors λ, der so genannten Verlustgewichtung, geregelt, der einen Wert zwischen 0 und 1 annehmen kann. Die endgültige Verlustfunktion sieht wie folgt aus:

$$loss = MSE + \lambda * KL \tag{4.1}$$

Für die Optimierung der Hyperparameter wird Hyperband (Li, Jamieson u. a. 2018) gewählt.

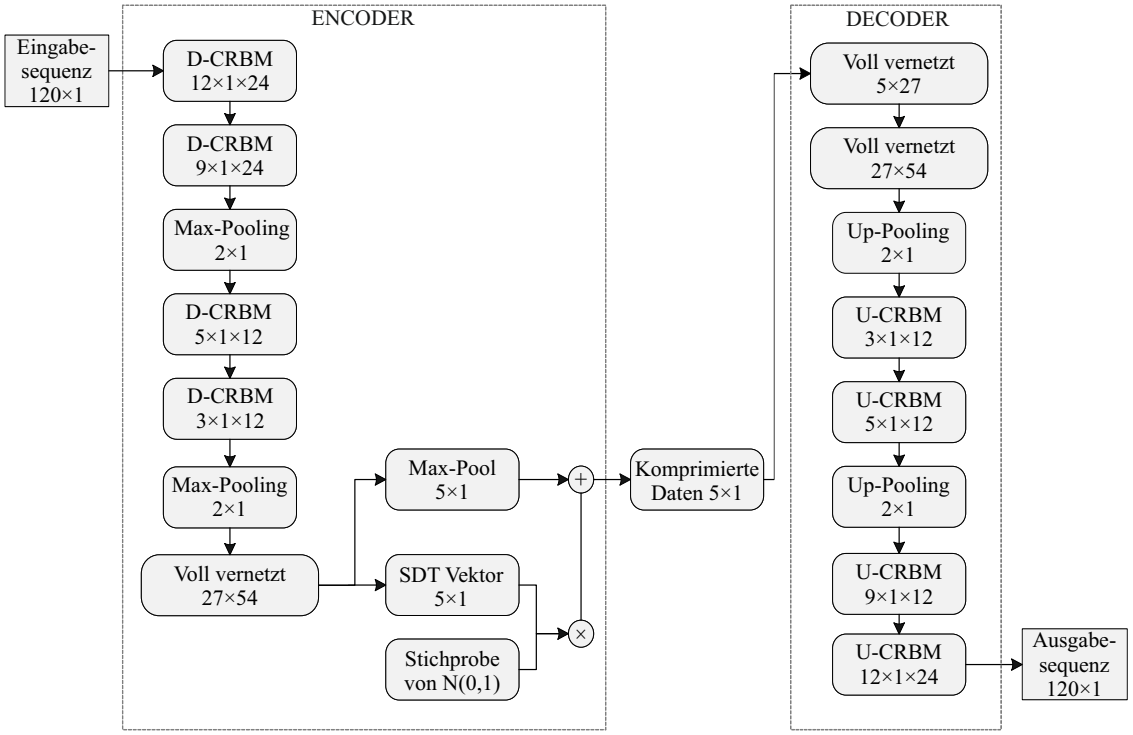


Abbildung 4.3: CBN-VAE-Modell (nach Liu, Chen, Yan u. a. 2019). Der Inhalt in den Klammern der Netzschicht (D-CRBM bzw. U-CRBM und voll vernetzte Schichten) sind die Größe des Filters, und die Inhalte in den Klammern der Eingabe- und Ausgabe sind die Größe des Vektors. Bei Max-Pooling- und Up-Pooling-Schichten entsprechen die Inhalte in den Klammern der Größe der Stichprobe.

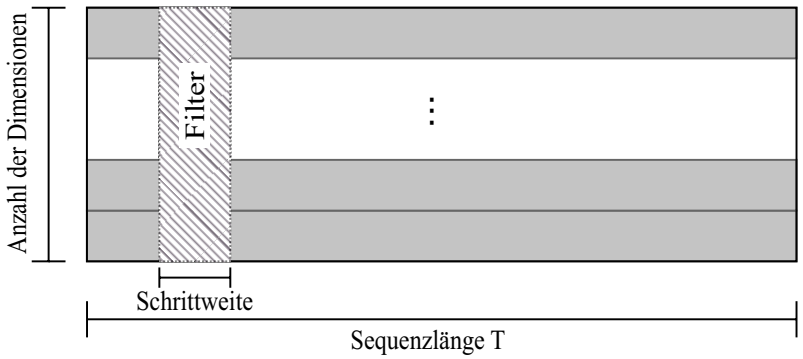


Abbildung 4.4: Filtermatrix bei Kompression von mehrdimensionalen Eingabedaten

LSTM-Autoencoder

Modell: Das im Folgenden eingesetzte LSTM-Autoencoder Modell basiert auf der Arbeit von (Wong und Luo 2018). LSTM ist eine Sonderform von RNN, welches das Problem von verschwindenden bzw. explodierenden Gradienten bei RNN löst. Der LSTM-Autoencoder von (Wong und Luo 2018) wurde ursprünglich für die Analyse von industriellen Sensorsignalen von großem Umfang und dementsprechend für die Anwendung auf multivariate Datenflüsse entwickelt. Die Architektur des Autoencoders wird bei LSTM anders angewandt als bei anderen Arten von neuronalen Netzen und ist in Abbildung 4.5 dargestellt. Dies ist auf die Fähigkeit zurückzuführen, Daten zu komprimieren, indem Korrelationen in den zeitlichen Aspekten der Sequenz statt in ihren verschiedenen Dimensionen identifiziert werden (Wong und Luo 2018). Die Encoder-Schicht eines Netzes gibt nicht für jeden Zeitschritt in der Eingabe die entsprechende Ausgabe zurück, sondern nur für den letzten Zeitschritt der Eingabesequenz, was den Engpass bei der Datenkompression darstellt.

Der beschriebene Aufbau überzeugt laut (Wong und Luo 2018) durch seine geringe Komplexität sowie die Eigenschaft, zeitabhängige Beziehungen erlernen zu können. Abbildung 4.5 stellt den schematischen Aufbau des LSTM-Autoencoders dar.

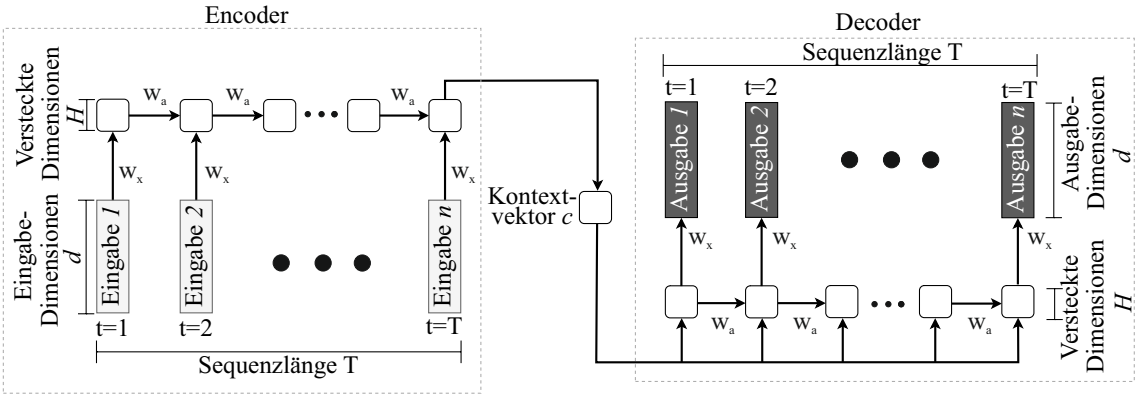


Abbildung 4.5: Schematische Struktur des LSTM-Autoencoder Kompressionsmodells (nach Wong und Luo 2018)

Adaptionen: Das ursprüngliche Modell ist für die Anwendung auf hochdimensionale Eingabedaten ausgelegt. Dennoch ist es möglich, auch eine Eingabe der Dimension 1 zu übergeben. Für die Anwendung auf UTS sind keine Anpassungen erforderlich.

Training: Für das Training des LSTM-RNN wird ADAM (Kingma und Ba 2017) als Trainingsalgorithmus und der MSE als Verlustfunktion gewählt. Die Hyperparameteroptimierung wird ebenfalls mittels Hyperband Optimierung (Li, Jamieson u. a. 2018) durchgeführt.

Der in (Rosenberger, Kübel und Rothfuß 2022) beschriebene experimentelle Vergleich basierend auf zwei öffentlich verfügbaren Datensätzen[1][2] bestätigt, dass auf ML basierende Ansätze unter Umständen zu besseren Ergebnissen als klassische Verfahren führen. Der Vergleich der zwei Autoencoder Modelle resultierte in den folgenden Thesen:

- Obwohl der CBN-VAE Autoencoder hinsichtlich der Parameteranzahl sowie Floating Point Operations Per Seconds (FLOPSs) das weitaus komplexere Modell ist, ist die Verarbeitungszeit geringer. Das wird begründet durch die Parallelisierung von Prozessen. Dies ist bei dem LSTM-Modell nicht möglich, da es zustandsabhängig ist, um den Kontext zwischen zwei Sequenzen zu teilen. In einem realen Produktionssystem ist die Parallelisierung der Prozesse nicht möglich, da die Daten unmittelbar und Sequenz nach Sequenz verarbeitet werden.
- Beide Modelle erzielen schlechtere Ergebnisse bei dem zweiten Datensatz[2], was sich durch den Datensatz selbst begründen lässt. Hochvariate Daten können generell schlechter komprimiert werden aufgrund geringerer Korrelationen.
- Die Kompressionsrate für den mehrdimensionalen Fall ist bei dem CBN-VAE Modell stark angestiegen, da die Engpassschicht unabhängig der Dimension immer die gleiche Anzahl an Merkmalen extrahiert. Dies hat eine deutliche Verschlechterung der Genauigkeit zufolge.
- Da die Evaluation auf zwei unterschiedlichen Datensätzen basierte, wurde die Generalisierbarkeit für beide Modelle nachgewiesen, wobei der LSTM-Autoencoder bessere Generalisierbarkeit über verschiedene Datenstrukturen aufzeigte.
- Bei den genannten Experimenten erzielt der LSTM-Autoencoder zusammenfassend einen 30% höheren Qualitätswert.

Die genauen Ergebnisse sind dem Anhang A.1 zu entnehmen. Da der LSTM-Autoencoder bessere Ergebnisse erzielte, insgesamt von geringerer Komplexität sowie besonders geeignet für Zeitreihendaten ist, wird dieses Modell in der betreuten Arbeit in (Rauterberg 2022b) vertieft und die Anwendung im IIoT geprüft.

In der nachfolgenden Evaluation wird das Verfahren mit dem klassischen Verfahren fpzip (nach Lindstrom und Isenburg 2006), der DWT sowie der Quantisierung verglichen und die Anwendbarkeit auf industrielle Datenflüsse auf einem industriellen Steuergerät überprüft. Die Ergebnisse sind in Abschnitt 4.6.1 zusammengefasst und diskutiert.

Quantisierung

Die Quantisierung ist ein Verfahren der Datenreduktion, jedoch nicht der Datenkompression, da es keinen Algorithmus zur Dekompression erfordert. Die Datenreduktion

[1] Intel Berkeley Research lab data set: http://db.csail.mit.edu/labdata/labdata.html
[2] Distillate flowrate data set: https://openmv.net/info/distillate-flow

erfolgt mittels Reduzierung der Nachkommastellen. In der Arbeit von (Rauterberg 2022b) erfolgt die Quantisierung von 64-bit auf 16-bit Fließkommazahlen und erzielt somit einen Kompressionsfaktor von $CR = 4$. Die Nachkommastellen werden entfernt und es wird nicht gerundet. Die Umsetzung erfolgt durch die Änderung des Datentyps.

DWT

Die DWT ist ein Verfahren der funktionalen Approximation und ist bereits in Abschnitt 2.5.2 genauer erläutert. In dieser Arbeit wird das Wavelet Daubechies 1 (db1) gewählt. In der Arbeit von (Rauterberg 2022b) wird in Prozent vorgegeben, wieviel der Energie des Ausgangssignals beibehalten wird. Die DWT wird in Python mittels der Bibliothek PyWavelets implementiert.

fpzip

Weiterhin wird ein Verfahren vorgestellt und verglichen, welches auch für die verlustfreie Datenkompression geeignet ist. Die Möglichkeit einer verlustfreien Kompression kann für die Übermittlung von Daten für die explorative Datenanalyse relevant sein. Das Verfahren fpzip nach (Lindstrom und Isenburg 2006) ist im Gegensatz zu den klassischen Verfahren, welche in Abschnitt 2.5 vorgestellt werden, insbesondere für die Anwendung auf Fließkommazahlen geeignet. Fließkommazahlen nach (IEEE 754 1985) setzen sich aus Mantisse und Exponent wie folgt zusammen (Lindstrom und Isenburg 2006):

$$(-1)^s 2^{e-2^n e^{-1}-n_m+1}(2^{n_m}+m) \tag{4.2}$$

Dabei gilt für einfache Genauigkeit das Vorzeichen-Bit s, der Exponent e der Länge $n_e = 8$ bit und die Mantisse m der Länge $n_m = 23$ bit beziehungsweise für doppelte Genauigkeit $n_e = 11$ bit und $n_m = 52$ bit.

Das Verfahren fpzip unterteilt sich in drei Schritte – (1) Prädiktion, (2) Berechnung und Weiterverarbeitung der Residuen und (3) Kodierung – welche in Abbildung 4.6 zusammengefasst sind. Für Details wird auf die Veröffentlichung von (Lindstrom und Isenburg 2006) verwiesen.

1. Lorenzo-Prädiktion: Mittels Prädiktion nach (Ibarria u. a. 2003) wird ein Wert basierend auf bereits kodierten Werten vorhergesagt. Anschließend wird der vorhergesagte und der reale Wert in binäre Ganzzahlen umgewandelt.

2. Berechnung und Weiterverarbeitung der Residuen. Zunächst werden die Differenzen (Residuen) zwischen realen und vorhergesagten Werten berechnet. Der mögliche Bereich der Residuen wird in Intervalle variierender Größe unterteilt. Ein Residuum wird durch die Intervallnummer und seine Position im Intervall dargestellt. Das optimierte Vorgehen basiert auf dem zweistufigen Verfahren nach (Chen, Chiang u. a. 2003).

3. Entropiekodierung nach dem Verfahren *Fast Entropy Coding*, einer optimierten Form der Arithmetischen Kodierung basierend auf (Martin 1979; Subbotin 1999).

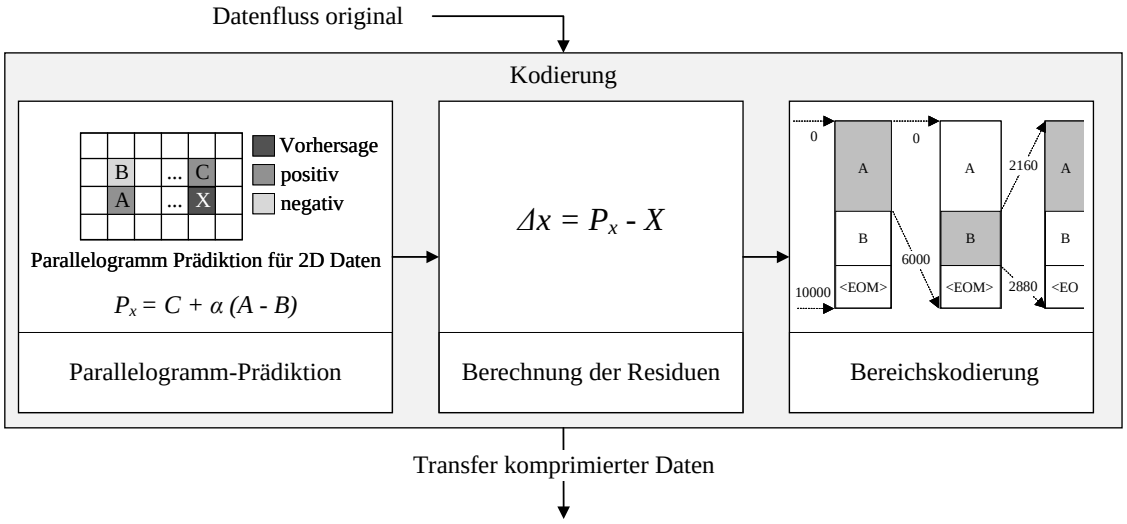


Abbildung 4.6: Datenkompression mittels fpzip

Da die Prädiktion auf Basis der vorangegangenen Werte erfolgt, ist die tupelweise Kompression nicht möglich. Stattdessen kommt ein gleitendes Fenster für die Anwendung auf Datenflüsse zum Einsatz, wobei die Fenstergröße frei wählbar ist. Es können ebenfalls verlustbehaftete Modi gewählt werden, bei denen die am wenigsten relevanten Bits des Exponenten weggelassen werden.

4.4 Informationsextraktion mittels Anomalieerkennung

Datenanalysen verfolgen das Ziel, die in Rohdaten enthaltenen Informationen zu extrahieren. Die Menge der Informationen ist in Relation zu der Rohdatenmenge gering, sodass die Informationsextraktion eine starke Datenreduktion erzielt. In Abschnitt 4.1 ist hergeleitet, dass wesentliche Informationen in Datenflüssen am Netzwerkrand mittels Anomalieerkennung identifiziert und extrahiert werden können. Dazu wird ein nicht-parametrisches statistisches Verfahren vorgestellt, das die in Abschnitt 3.3 genannten Anforderungen einer Datenflussverarbeitung erfüllt. Das Verfahren wird zudem für die Erkennung aller vier vorgestellten Anomaliearten erweitert. Es werden insbesondere die Echtzeitfähigkeit sowie geringer Rechen- und Speicherbedarf berücksichtigt, sodass die Ausführung auf industriellen Endgeräten in der Anlage möglich ist. Die folgenden Ausführungen basieren auf den Inhalten der Veröffentlichung (Rosenberger, Müller u. a. 2022), welche ursprünglich auf der CIRP Conference on Intelligent Computation in Manufacturing Engineering 2021 vorgestellt wurden.

Zunächst werden die verschiedenen, in Abschnitt 2.6.2 einzeln vorgestellten, Ansätze zur Anomalieerkennung hinsichtlich der Anforderungen miteinander verglichen. Die Ergebnisse sind Anhang A.2 zu entnehmen. Die Bewertung zeigt, dass statistische Verfahren für den Anwendungsfall der Datenflussoptimierung am besten geeignet sind. Bei den statistischen Verfahren wird, wie in Abschnitt 2.6.2 beschrieben, zwischen parametrischen und nicht-parametrischen Verfahren unterschieden. Die nicht-parametrischen sind den parametrischen Verfahren hinsichtlich dem geringen Aufwand für die Parametrierung sowie in ihrer Agnostik überlegen. Ein bekannter Vertreter der nicht-parametrischen Statistik ist der Kerndichteschätzer, welcher bereits für die Anomalieerkennung Anwendung findet (Goldstein und Uchida 2016; Rosenberger, Müller u. a. 2022). Die Vorteile sind die geringe Komplexität sowie der geringe und konstante Speicherbedarf. Außerdem ist die Anzahl der Rechenoperationen für jeden neuen Datenpunkt konstant. Da die Anzahl der Rechenoperationen von der Hyperparameterwahl abhängig ist, könnte man das Verfahren durch die Wahl der Hyperparameter adaptiv an die Möglichkeiten des Endgeräts anpassen. Das in (Zhou u. a. 2003) vorgestellte Verfahren Merging Kernels for Density Estimation (M-KDE) über Datenflüsse stellt eine Mischung aus statistischem und dichtebasiertem Nächste-Nachbbarn-Ansatz dar. Aufgrund seiner einfachen Struktur, die sich am besten an die genannten Randbedingungen anpasst, dient es als Grundlage für das im Rahmen dieser Arbeit ausgearbeitete und im Folgenden vorgestellte Verfahren.

Der wesentliche Beitrag in dieser Arbeit ist eine Methode mit der die Erkennung der vier verschiedenen Arten von Anomalien in Datenflüssen auf industriellen Endgeräten möglich ist, die sogenannte *Extended Merging Kernel Density Estimation for Edge-Devices (EEM-KDE)*, die die im Abschnitt 3.3 abgeleiteten Anforderungen erfüllt.

Wie in Abschnitt 4.2.2 beschrieben, sind bestehende Verfahren zunächst nur für die Erkennung einzelner Arten von Anomalien – wie z. B. Punktanomalien – geeignet. Um mehrere Arten von Anomalien zu erkennen werden häufig hybride Systeme eingesetzt. Das bedeutet, dass ein Gesamtsystem die Kombination von zwei oder mehr verschiedenen Verfahren beinhaltet. Dies führt allerdings zu einem Mehrbedarf an Rechenkapazität und Speicherplatz. Da diese in den Endgeräten sehr begrenzt sind, handelt es sich in der hier vorgestellten Lösung dagegen um ein Konzept zur Erweiterung des Grundverfahrens. Vorteile hierdurch sind eine geringe Programmgröße und weniger Ressourcenauslastung. Zunächst wird auf das Verfahren des KDE selbst eingegangen. Im Anschluss werden die Anwendung für die einzelnen Anomaliearten und die entsprechend notwendigen Erweiterungen vorgestellt.

Aufbau des Kerndichteschätzers

Die Anomalieerkennung mittels KDE wird den Algorithmen des ML zugeordnet (Goldstein und Uchida 2016), da die Anomalieerkennung datenbasiert erlernt wird und ohne explizite Programmierung erfolgt. KDE werden genutzt, um eine Wahrscheinlichkeit für einen Datenpunkt x_i auszugeben, wobei die Dichteverteilung nicht

explizit vorgegeben ist. Die Grundlage hierfür sind die sogenannten Kerne und ihre gewichteten Verteilungsfunktionen. Aus der Überlagerung der einzelnen Verteilungsfunktionen ergibt sich, wie in Abbildung 4.7 ersichtlich, die Gesamtdichteverteilung. Ein Kern definiert sich durch die drei Parameter Mittelpunkt μ_i, Bandbreite h_i und Gewichtung ρ_i. Der Mittelpunkt der Verteilungsfunktion (μ_i) stellt das Kernzentrum dar. Die Bandbreite (h_i) beschreibt die Stauchung oder Streckung der Kernverteilung. Die Gewichtung (ρ_i) ist ein Maß für die Bedeutung des Kerns und gibt die Anzahl der Vereinigungen, die der Kern und alle mit ihm verschmolzenen Kerne bereits erfahren hat, an. Des Weiteren besitzt ein Kern den Parameter m_cost_i, der die Vereinigungskosten mit dem nächstliegenden Kern beschreibt. Diese Kosten berechnen sich durch den Abstand zwischen den zwei Kernzentren. Je näher die Kerne zusammenliegen, desto geringer sind die Vereinigungskosten.

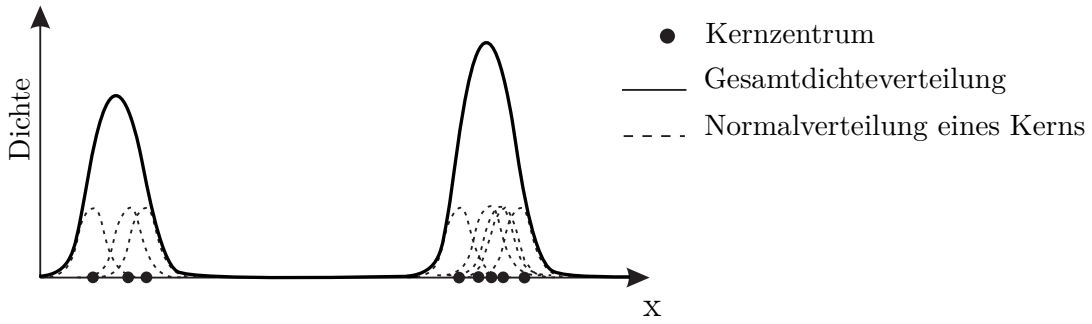


Abbildung 4.7: Aufbau des Kerndichteschätzers (Rosenberger, Müller u. a. 2022)

Das M-KDE Verfahren (Zhou u. a. 2003) ist für die Verarbeitung von Datenflüssen mittels KDE geeignet. Für dieses Verfahren sind lediglich drei Hyperparameter, die Anzahl der Kerne k, die Bandbreite h und die Kernfunktion $K()$, erforderlich. In einem ersten Schritt wird ein Puffer in derselben Größe der Kernanzahl angelegt. Dieser Puffer wird zunächst mit Datenpunkten gefüllt. Ist der Puffer voll, wird jeder neu eingehende Datenpunkt x_i als ein neuer Kern mit der Gewichtung 1 und dem Kernmittelpunkt $\mu_i = x_i$ eingefügt, sodass $k + 1$ Kerne vorliegen. Der Puffer wird nach den Mittelpunkten sortiert und die jeweiligen Vereinigungskosten berechnet. Anschließend wird eine Vereinigung zwischen den zwei Kernen im Puffer mit den geringsten Vereinigungskosten durchgeführt. Somit liegen wieder k Kerne im Puffer vor und das Vorgehen wird mit dem nächsten eingehenden Datenpunkt wiederholt. Die einzelnen, bei Eingang eines neuen Datenpunktes auszuführenden Rechenschritte sind in nachfolgendem Pseudocode 1 dargestellt. Grafisch ist das Vorgehen im Puffer in Abbildung 4.8 und anhand der Dichteverteilung in Abbildung 4.9 ersichtlich. Es umfasst die beschriebenen Schritte 10-15 im Pseudocode.

Mathematisch ist der Vorgang der Vereinigung in Formel 4.3 beschrieben. Die neue Gewichtung ist die Summe der Einzelgewichtungen und die Bandbreite sowie der Mittelpunkt ergeben sich jeweils aus dem gewichteten Mittelwert der Werte der

Algorithmus 1 Pseudocode des univariaten M-KDE (angelehnt an Zhou u. a. 2003)

Eingabe Datenfluss $(x_1, x_2, ...)$
Ausgabe Wahrscheinlichkeit je Datenpunkt entsprechend Dichteverteilung
Algorithmus DensityOverDataStream()

```
Iniitiere einen Puffer mit m Einträgen
while Datenfluss hat nicht geendet do
    Lese Datenpunkt x_i aus dem Datenfluss aus
    Fülle das Element [μ_i, h_i, ρ_i, m_cost_i] mit [x_i, h, 1, m_cost]
    Füge das Element in den Puffer ein
    Sortiere den Puffer nach Mittelpunkten μ
    Aktualisiere die Spalte m_cost
    if Puffer ist voll then
        Finde den Eintrag des geringsten m_cost-Wertes im Puffer
        Vereine den Eintrags des geringsten m_cost mit der Zeile darüber
        Füge den vereinten Wert in den Puffer ein
        Entferne die zwei Einträge, die vereinigt wurden
        Sortiere den Puffer nach μ
        Aktualisiere die Spalte m_cost
    end if
    Ausgabe des Puffers und Aufbau der Dichtefunktion bei Bedarf
end while
```

beiden Kerne vor der Vereinigung. Die Kernfunktionen können frei gewählt werden und sind in Tabelle A.4 beispielhaft dargestellt.

$$\frac{\rho_i^*}{h_i^*} \cdot K(\frac{x-\mu_i^*}{h_i^*}) + \frac{\rho_j^*}{h_j^*} \cdot K(\frac{x-\mu_j^*}{h_j^*}) \simeq \frac{\rho_i^* + \rho_j^*}{h_u^*} \cdot K(\frac{x-\mu_u^*}{h_u^*}) \tag{4.3}$$

ρ_i^*, ρ_j^*	Gewichtungen der beiden Kerne
h_i^*, h_j^*	Bandbreiten
μ_i^*, μ_j^*	Kernmittelpunkte
$K(...)$	Kernfunktion
h_u^*	Neue Bandbreite
μ_u^*	Neuer Mittelpunkt

Die Parameterwerte des neuen Kerns werden wie folgt berechnet:

- Die Gewichtung ρ ist wie in Gleichung 4.4 dargestellt die Summe der zwei Einzelgewichtungen.

$$\rho_{(t+1)}(k_u) = \rho_t(k_i) + \rho_t(k_j) \tag{4.4}$$

- Die Bandbreite h wird als Hyperparameter vom Anwender festgelegt und ist für alle Kerne identisch. Sie ändert sich deshalb durch die Vereinigung der Kerne nicht.

- Der neue Kernmittelpunkt μ_u^* ist der gewichtete Mittelwert aus den zwei vereinigten Instanzen.

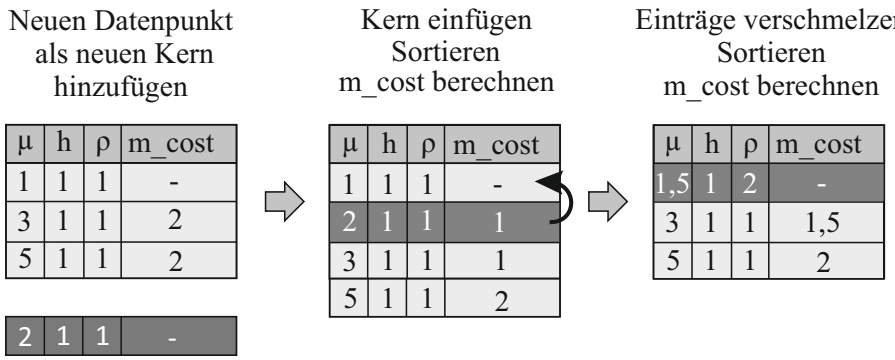


Abbildung 4.8: Vorgehen im Puffer

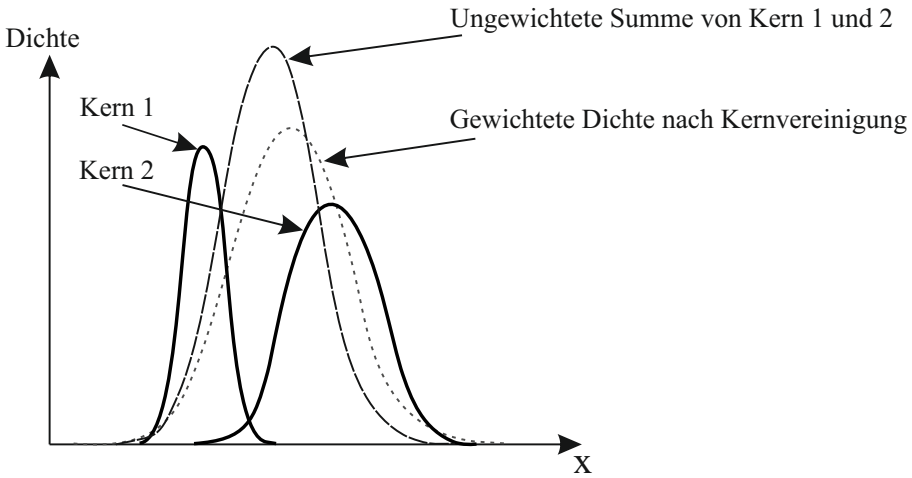


Abbildung 4.9: Auswirkung einer Kernvereinigung (nach Zhou u. a. 2003)

Die Wahrscheinlichkeit eines Datenpunktes x_i wird anhand Formel 4.5 ermittelt.

$$f_n^*(x) = \frac{1}{k} \sum_{j=1}^{k} \frac{\rho_j^*}{h_j^*} \cdot K(\frac{x - \mu_j^*}{h_j^*}) \text{ mit: } n = \sum_{j=1}^{k} \rho_j^* \tag{4.5}$$

Im Folgenden wird der Einsatz des KDE zur Erkennung verschiedener Anomalien beschrieben.

Erkennung von Punktanomalien

Der KDE generiert entsprechend vorangegangener Beschreibung eine Gesamtdichteverteilung. Für die Erkennung von Punktanomalien wird nur für den eingehenden Datenpunkt x seine Auftrittswahrscheinlichkeit $\hat{f}_n^*(x)$ entsprechend dem KDE-Modell und Gleichung 4.5 berechnet. Wie in Abbildung 4.10 ersichtlich liegt eine

Anomalie vor, wenn ein Datenpunkt x_i auftritt, dem eine Wahrscheinlichkeit unter einem festgelegten Schwellwert σ_s zugeordnet ist.

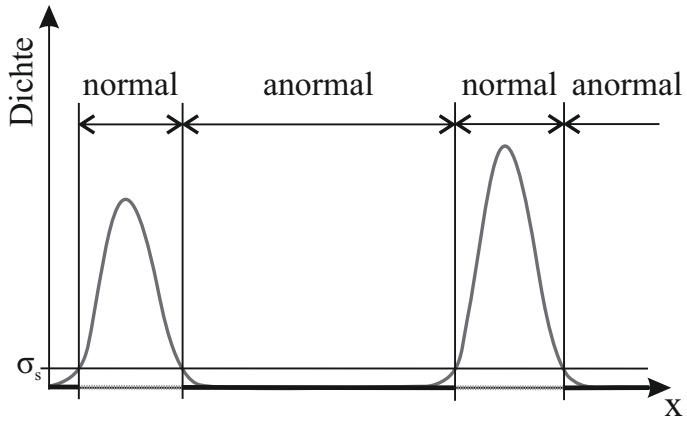


Abbildung 4.10: Erkennung von Punktanomalien in Abhängigkeit eines Schwellwertes

Für die Erkennung der anderen genannten Arten von Anomalien muss das Verfahren M-KDE im Rahmen der vorliegenden Arbeit erweitert werden. Diese Erweiterungen werden in den folgenden drei Unterabschnitten beschrieben.

Erkennung schleichender Anomalien

Schleichende Anomalien (Drifts) werden in dieser Arbeit über die Bewegung der einzelnen Kerne erkannt. Dafür werden k Driftkennzahlen eingeführt, die ein Maß für die Änderung des Parameters Kernmittelpunkt darstellen. Die Adaptivität des KDE wird durch die Anpassung an die eingehenden Daten erzielt. Wie bereits beschrieben, führt jeder neu eingehende Datenpunkt zu einer Vereinigung von zwei Kernen und somit einer Aktualisierung der Position eines Kernmittelpunktes. Diese Verschiebung ist in Abbildung 4.11 bildlich dargestellt, wobei $\mu_{1(t)}$ dem Zentrum von Kern 1 vor der Verschmelzung und $\mu_{1(t+1)}$ dem Mittelpunkt von Kern 1 nach der Verschmelzung entspricht. Die Driftkennzahl eines Kerns berechnet sich aus der Summe der Differenz $\pm\Delta x$, die sich zwischen dem Mittelpunkt des Kerns $\mu_{i(t)}$ vor der Vereinigung und dem Mittelpunkt $\mu_{i(t+1)}$ nach der Vereinigung ergibt. Ändern sich die Daten über einen langen Zeitraum, so bewegen sich die Kernmittelpunkte konstant in eine Richtung. Der Betrag der Driftkennzahl wird größer. Ist ein bestimmter Schwellwert erreicht, wird eine schleichende Anomalie erkannt. Bleibt die Wahrscheinlichkeitsverteilung dagegen konstant, so bewegen sich die Kernmittelpunkte um einen festen Ort und die Driftkennzahl schwankt um einen festen Wert. Bei eindimensionalen Daten ist die Kennzahl ein Skalar, der die Größe der Positionsänderung angibt, bei mehrdimensionalen Daten ein Vektor, der die Größe und Richtung der Positionsänderung angibt.

Außerdem kann die Driftkennzahl als Maß für den stabilen Zustand des Modells dienen. Zu Beginn ist eine starke Bewegung der Kerne zu erwarten, da die Dichtever-

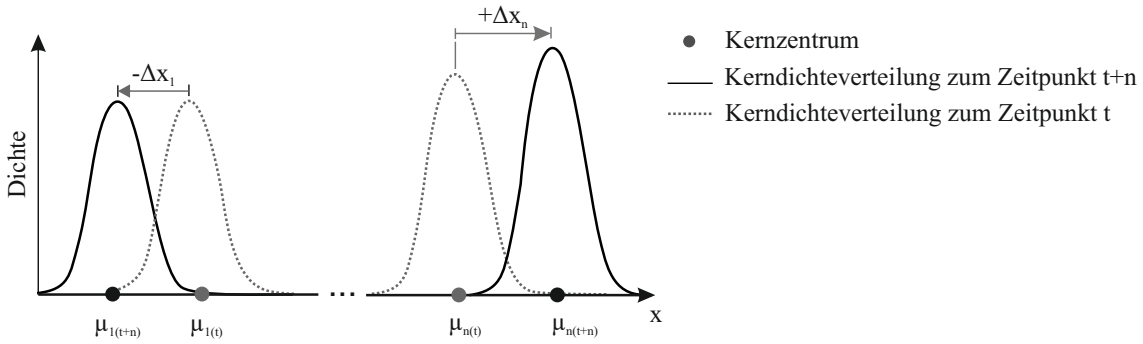


Abbildung 4.11: Darstellung der Positionsänderung von Kernmittelpunkten (Rosenberger, Müller u. a. 2022)

teilung erst auf wenigen Datenpunkten basiert und somit noch nicht repräsentativ für die reale Verteilung sein kann. Die Gewichte der Kerne sind zudem noch sehr gering, sodass durch Vereinigungen große Bewegungen hervorgerufen werden können. Die Verteilung ist stabil, wenn die Driftkennzahlen nur in einem bestimmten Bereich schwankt und die einzelnen Bewegungen bei der Vereinigung gering sind. Die Erkennung des stabilen Zustands ist Voraussetzung für die Anomalieerkennung.

Verschmelzen zwei Kerne mit hoher Gewichtung und entsteht damit durch den neuen Datenpunkt ein neuer Kern mit sehr geringer Gewichtung, wird für diesen zunächst ein starker Drift erkennbar sein. Aus diesem Grund werden Kerne erst ab einer bestimmten Gewichtung relativ zur durchschnittlichen Gewichtung aller Kerne berücksichtigt, um falsch-negative Ausgaben von Anomalien zu vermeiden. Die Driftkennzahl wird mit jedem eingehenden Datenpunkt nicht für alle Kerne, sondern jeweils nur für den neu vereinigten Kern aktualisiert, um den Aufwand zu minimieren. Eine weitere mögliche Vereinfachung stellt die reine Betrachtung der Vorzeichenfolge anstelle der Größe der Bewegung dar. Bewegt sich ein Kern nur in eine Richtung, also immer mit demselben Vorzeichen, liegt ein Drift vor.

Erkennung von kollektiven Anomalien

Kollektive Anomalien bezeichnen, wie in Abschnitt 2.6.1 beschrieben, Werte, die alleinstehend als normal eingestuft werden, aber durch das gemeinsame Auftreten eine Anomalie darstellen. Im industriellen Umfeld tritt der Großteil der kollektiven Anomalien als Abweichung von einem Produktionszyklus auf. In anderen Bereichen, beispielsweise der Cybersicherheit, sind auch zyklenunabhängige anormale Muster anzutreffen. Für den Anwendungsbereich der I4.0 wird die Erkennung von kollektiven Anomalien auf Anomalien in zyklischen Daten eingeschränkt. Zunächst ist es notwendig festzustellen, ob die eingehenden Daten zyklisch sind. Ist ein Zyklus erkannt, so werden die Sequenzen miteinander verglichen und Abweichungen als Anomalien ausgegeben. Die eingehenden Daten werden in Abhängigkeit von ihrem Wert einem Rastersegment zugeordnet. Für jedes Rastersegment wird eine Kodierung hinterlegt. Bei jeder Zuordnung eines Datenpunktes in die Rastersegmente wird also

jeweils eine Kodierung an die Sequenz angefügt. Dies ist in Abbildung 4.12 ersichtlich. Der erste kodierte Wert im Puffer, im Beispiel in Abbildung 4.12 blaues „A“, wird als Initialwert bezeichnet. Im Puffer werden so lange kodierte Werte gespeichert, bis der Initialwert erneut auftritt. Dann werden die nachfolgend eingehenden Kodierungen mit den Werten im Puffer, die auf den Initialwert folgen, abgeglichen. Haben bei der Überprüfung alle eingehenden kodierten Werte mit den gepufferten Werten übereingestimmt (grüne Pfeile), so wird die im Puffer zwischengespeicherte Sequenz als Zyklus erkannt. Kommt es dagegen zu der Situation, dass ein eingehender Wert nicht mehr mit dem zu vergleichenden Wert im Puffer übereinstimmt (rote Pfeile), so wird die Prüfung abgebrochen und der Puffer um die eingehenden Werte erweitert bis erneut der Initialwert eingeht. Im Beispiel ist zu erkennen, dass erst bei der dritten Prüfung nach Auftreten eines Initialwerts ein Zyklus vollständig erkannt und als Referenzzyklus festgelegt wird. Durch die Einteilung in Raster und anschließende Kodierung ist das Verfahren stabiler als bei dem Abgleich der konkreten Werte. Bevor die Anomalieerkennung beginnt, muss der Referenzzyklus noch ein weiteres Mal durch den darauffolgenden Zyklus bestätigt werden, der zu einem bestimmten Prozentsatz mit dem Referenzzyklus übereinstimmen muss.

Danach erfolgt die Anomalieerkennung durch den Vergleich der eingehenden Sequenz mit dem bestätigten Referenzzyklus. Ist Anzahl der abweichenden kodierten Werte größer als ein festgelegter prozentualer Schwellwert, wird der Zyklus als Anomalie erkannt.

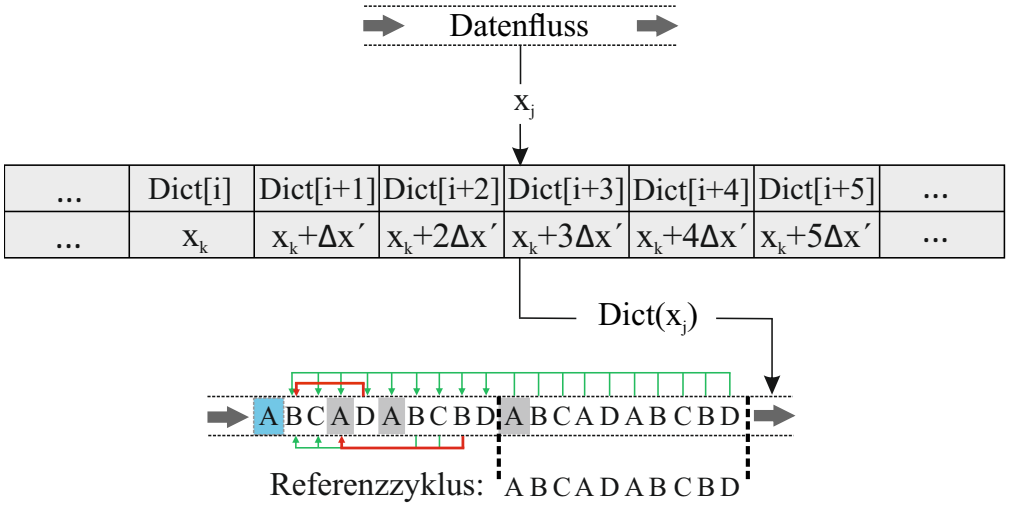


Abbildung 4.12: Ablauf der Zyklenerkennung (Rosenberger, Müller u. a. 2022)

Um die Qualität der Zyklenerkennung zu steigern, wird eine adaptive Rasterung erstellt. Adaptiv bedeutet, dass keine feste Segmentlänge, sondern eine Rasterung in Abhängigkeit der Auftrittshäufigkeit verwendet wird. Dieses Vorgehen erreicht eine erhöhte Genauigkeit, da bei einer hohen Wahrscheinlichkeit die Rasterung feiner und bei einer geringen Wahrscheinlichkeit gröber ist. Dazu wird die Rasterung über den Flächeninhalt unter der normierten Dichteverteilung in dem jeweiligen

$$numTSeg_k(\Delta x_k) = A_k \cdot maxSeg \tag{4.6}$$

$numTSeg_k$	Anzahl an Teilsegmenten im Segment Δx_k
A_k	Normierter Flächeninhalt von Segment Δx_k
$maxTSeg$	Gesamtanzahl an zu verteilenden Teilsegmenten
k	Iterator für Segmente Δx

$$x_g(l) = x_k + j \cdot \frac{\Delta x}{numTSeg_k} \text{ mit: } j = [0; numTSeg_k] \tag{4.7}$$

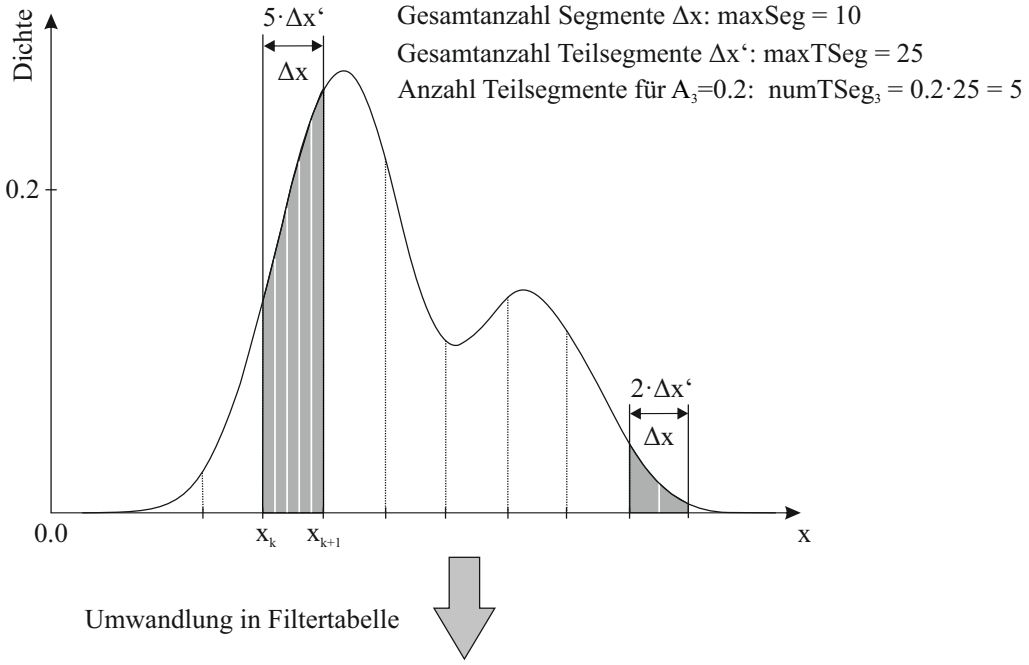
x_k	Startwert des aktuellen Segmentes $k \cdot \Delta x$
j	Iterator für Teilsegmente $\Delta x'$
l	Iterator für Grenzarray

Segment vorgenommen. Da die Breite über alle Segmente gleich groß ist, ist die Verteilung der Teilsegmente nur von dem mittleren normierten Dichtewert je Segment abhängig. Näherungsweise wird hierfür der Dichtewert in der Segmentmitte erfasst. In Formel 4.6 ist die Berechnung der Anzahl der Teilsegmente $numTSeg_k$ in einem Segment Δx_k beschrieben und in Formel 4.7 die jeweiligen Grenzen der Teilsegmente, denen entsprechend ein Datenpunkt in ein Segment eingeordnet wird.

Mit der Einordnung in ein Teilsegment erfolgt eine Kodierung durch ein einzelnes Symbol. Zur Anomalieerkennung im Ausführungssystem wird die Kodierung des aktuellen Zyklus mit der des vorhergehenden Zyklus verglichen. Der Abgleich mit einem Musterzyklus könnte aufgrund der Adaptivität des Verfahrens langfristig zu schlechteren Ergebnissen führen. Das Vorgehen zur Definition der Grenzen der Teilsegmente und der Zuordnung einer Kodierung ist in Abbildung 4.13 dargestellt.

Erkennung von Kontextanomalien

Kontextanomalien erfordern zur Erkennung die Betrachtung der Daten in einem Kontext. Dieser Kontext ergibt sich bei industriellen Anwendungsfällen durch das gemeinsame Betrachten von mehreren Signalen oder den Kontext der Zeit. Daraus folgt mathematisch die Notwendigkeit, mehrere Dimensionen zu berücksichtigen. Um die Komplexität zu reduzieren und somit die Performanz zu verbessern, kann im Rahmen der Datenvorverarbeitung eine Dimensionsreduktion vorangestellt werden. Die Methode zur Erkennung von Punktanomalien im eindimensionalen Raum ist dementsprechend nicht mehr ausreichend und muss auf den mehrdimensionalen Raum ausgeweitet werden. Konkret bedeutet das für den KDE die mathematische Änderung der univariaten Kernfunktionen, die bei der ursprünglichen Variante des KDE zur Erkennung von Punktanomalien Anwendung finden, zu multivariaten Kernfunktionen. Die Eingangsgröße, ursprünglich ein einzelner Datenpunkt x eines Datenflusses als Skalar, ändert sich zu einem Vektor der Länge m, der jeweils einen Datenpunkt von m Signalen zum selben Zeitpunkt t umfasst. Außerdem ändert sich



...	Dict[i]	Dict[i+1]	Dict[i+2]	Dict[i+3]	Dict[i+4]	Dict[i+5]	...
...	x_k	$x_k+\Delta x'$	$x_k+2\Delta x'$	$x_k+3\Delta x'$	$x_k+4\Delta x'$	$x_k+5\Delta x'$	...

Abbildung 4.13: Kodierung mittels adaptivem Raster (Rosenberger, Müller u. a. 2022)

die Bandbreite h von einem Skalar zu einer $m \times m$ Matrix. Die Rechenoperationen zur Berechnung der Vereinigungskosten und das Sortieren der Puffereinträge ist im mehrdimensionalen Raum deutlich komplexer. Im Gegensatz zu der univariaten Variante besitzt die $k \times 3$ Puffermatrix in diesem Fall nur die drei Spalten [Mittelpunkt, Bandbreite, Gewichtung] und k Zeilen entsprechend der Anzahl der Kerne. Die Grundlage für die Vereinigungskosten ist eine zusätzliche $k \times k$ Kostenmatrix mit den Abständen zwischen den Kernen, die die vektorielle Entfernung zwischen den Kernen als Skalar enthält. Ein neu eingehender Datenvektor wird mit einer zusätzlichen Zeile an das Ende des Puffers eingefügt. Auch die Kostenmatrix wird zu einer $k + 1 \times k + 1$ Matrix erweitert und die Vereinigungskosten in Bezug auf die bereits bestehenden k Kerne berechnet. Die Indizes der Vereinigungskosten in der Kostenmatrix stimmen mit dem der Kerne der Puffermatrix überein. Das kleinste Element der Kostenmatrix wird gesucht und entsprechend der Indizes diese beiden Kernen in der Puffermatrix vereinigt. Das Ergebnis, der neu vereinte Kern, wird in einer weiteren Zeile an den Puffer angefügt, sodass $k + 2$ Zeilen vorhanden sind. Nun müssen sowohl in der Kostenmatrix als auch der Puffermatrix die Einträge der beiden Kerne vor der Vereinigung entfernt werden, sodass wieder je eine $k \times k$ und $k \times 3$ Matrix entsteht (siehe Abbildung 4.14). Zur Optimierung in Bezug auf Leistung und Speicherplatz wird die Berechnung der Vereinigungskosten angepasst. Anstatt bei jeder neuen Dateninstanz nach dem Abstand zwischen den Kernen *m_cost* zu sortieren, wird die komplette Berechnung im mehrdimensionalen Raum nur einmal

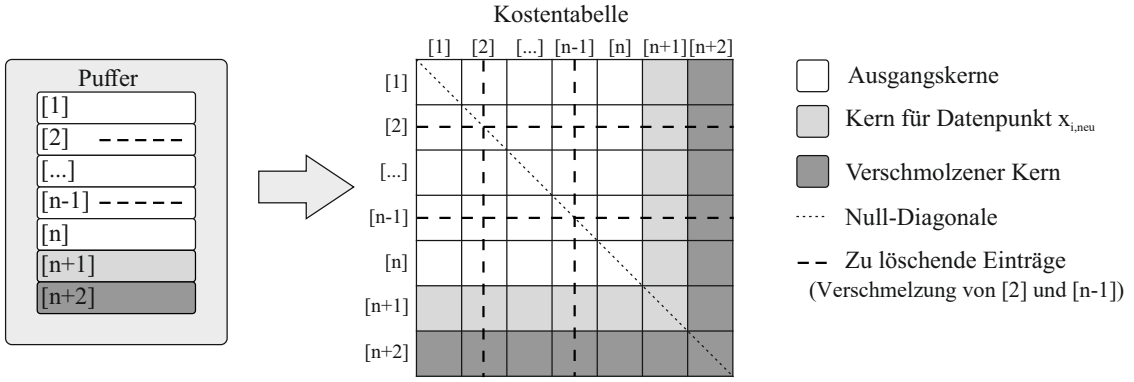


Abbildung 4.14: Effiziente Berechnung der Kostenmatrix (Rosenberger, Müller u. a. 2022)

Wahrscheinlichkeitsberechnung mit multivariaten Kernen

$$\hat{f}_n(x) = \frac{1}{n \cdot |H|^{0.5}} \cdot \sum_{i=1}^{n} K(H^{-0.5} \cdot (x - X_i)) \tag{4.8}$$

$\hat{f}_n(x)$	Dichtefunktion an der Stelle x
H	Bandbreitenmatrix
$\|H\|$	Determinante der Bandbreitenmatrix
n	Anzahl an Kernen im Puffer
μ_i	Kernmittelpunkt
$K(...)$	Kernfunktion

zu Beginn durchgeführt. Danach werden nur noch die Abstände des neu eingefügten Kerns berechnet. Gleichung 4.8 zeigt die Berechnung der Dichtefunktion $\hat{f}_n(x)$ für den m-dimensionalen Raum unter Verwendung der im Puffer gespeicherten Kerne k_i. Die Erkennung der Anomalie erfolgt darauf basierend analog zur Punktanomalie über die Auftrittswahrscheinlichkeit des Vektors. Ist diese sehr gering, wird der Vektor als Anomalie ausgegeben.

Tabelle A.4 im Anhang gibt einen Überblick über häufig eingesetzte Kernfunktionen und bietet einen Vergleich zwischen den uni- und multivariaten Ausprägungen. In dieser Arbeit wird der Gauß-Kern verwendet, welcher sich für UTS nach Gleichung 4.9 und für MTS nach Gleichung 4.10 berechnet.

$$(\sqrt{2\pi})^{-1} \exp(-0.5x^2) \tag{4.9}$$

$$(2\pi)^{-0.5d} \exp(-0.5||x||^2) \tag{4.10}$$

Das beschriebene Vorgehen zur Erweiterung von Verfahren zur Erkennung von Punktanomalien auf weitere Anomaliearten ist nicht nur für den KDE als statis-

tisches Verfahren geeignet, sondern auch auf die anderen Ansätze übertragbar. In Anhang A.4 sind die entsprechenden Anpassungen je Anomalieart auf die Parameter der weiteren Ansätze übertragen.

4.5 Effizienzsteigerung durch kombinierten Modelleinsatz

Die folgenden Ausführungen basieren auf den Inhalten, welche in Veröffentlichung (Rosenberger, Bühren und Schramm 2021) vorgestellt wurden.

Zeitreihendaten zeichnen sich nach (Chiarot und Silvestri 2023) durch Redundanz, Annäherbarkeit und Vorhersagbarkeit aus. Anomalien treten dagegen nach (Rosenberger, Müller u. a. 2022; Goldstein und Uchida 2016) in Relation zu normalen Daten selten auf. Sie besitzen weder den selben Ursprung wie normale Daten, d. h. sie sind beispielsweise nicht in demselben Anlagenverhalten begründet, noch entsprechen sie der Erwartung. Dementsprechend stellen sie das Gegenteil von normalen Zeitreihendaten dar. Wie in Abschnitt 4.1 abgeleitet, besitzen Anomalien einen hohen Informationsgehalt. Die Erkennung von Anomalien entspricht einer Informationsextraktion und kann somit auch als eine Form der verlustbehaftete Datenkompression angesehen werden, welche nach (Meyer 2019) auch als Reduktion von Irrelevanzen definiert ist.

Sowohl die Datenkompression als auch Anomalieerkennung können somit dasselbe Ziel, das Verhältnis von Informationsgehalt zu Datenmenge zu maximieren, verfolgen. Unter dieser Annahme ist es naheliegend, dass auch der Einsatz der gleichen Modelle möglich ist und diese nur unterschiedlich interpretiert werden. Aus den genannten Charakteristika von Anomalieerkennung und Datenkompression werden drei Annahmen getroffen, denen der gemeinsame Einsatz von Modellen zugrundeliegt (Rosenberger, Bühren und Schramm 2021):

1. Anomalien treten selten auf und sind nicht redundant. Sie sind daher schlecht zu komprimieren.

2. Anomalien treten unerwartet auf und sind daher schwer vorherzusagen oder zu approximieren.

3. Anomalien sind relevante Informationen.

Tabelle 4.1 beschreibt in Anlehnung an die Einteilung der Verfahren in Grundlagenkapitel 2.5 typische Ansätze und nennt sowohl für die Anomalieerkennung als auch die Datenkompression Beispiele der zu betrachtenden Kenngrößen, Beispiele für konkrete Verfahren sowie Verweise auf Literatur, in der die Ansätze bereits zu Einsatz kommen.

Tabelle 4.1: Kombinierter Modelleinsatz für Datenkompression und Anomalieerkennung: Beispiele für die zu betrachtenden Kenngrößen, konkrete Verfahren sowie Verweise auf Literatur

Ansatz	**Kompression**	**Anomalieerkennung**
Statistische Modelle	Kodierung nach Auftrittswahrscheinlichkeit, z.B. Huffman-Kodierung, (Huffman 1952)	Auftrittswahrscheinlichkeit z.B. Gaußsche Modelle, (Guttormsson u. a. 1999)
Neuronale Netze	Komprimierte Darstellung durch Engpassschicht, z.B. Autoencoder, (Wong und Luo 2018)	Rekonstruktionsfehler, z.B. Autoencoder, (Thompson u. a. 2002)
Prädiktive Modelle	Kodierung von Differenzen, z.B. Delta-Kodierung (Nibali und He 2015)	Abgleich mit vorhergesagten Wert, z.B. Abweichung zw. Simulation u. realer Welt, (Weiss und Hirsh 1998)
Funktionale Approximation	Approximierte Darstellung, z.B. DCT, (Sparka u. a. 2017)	Betrachtung der Frequenzanteile, z.B. DCT, (Thill, Konen und Bäck 2017; Din und Qazi 2018)
Spektral-zerlegung	Dimensionsreduktion, z.B. Hauptkomponententransformation (Liu, Mohandes u. a. 2018)	Distanz oder Abweichung z.B. zu Hauptkomponenten oder Eigenvektor, (Idé und Kashima 2004; Parra, Deco und Miesbach 1996)
Informations-theoretische Ansätze	Informationsgehalt, z.B. Entropie, (Liu, Towsley u. a. 2005)	Informationsgehalt, z.B. Kolmogorow-Komplexität, (Keogh, Lonardi und Ratanamahatana 2004)

Der Nutzen des kombinierten Modelleinsatzes kann auf unterschiedliche Weise erzielt werden. Ein allgemeines Vorgehen zur Steigerung von Effizienz und Qualität in ressourcenlimitierten Systemen mittels kombiniertem Modelleinsatz bei unterschiedlichen Voraussetzungen wird in dem Ablaufdiagramm 4.15 beschrieben.

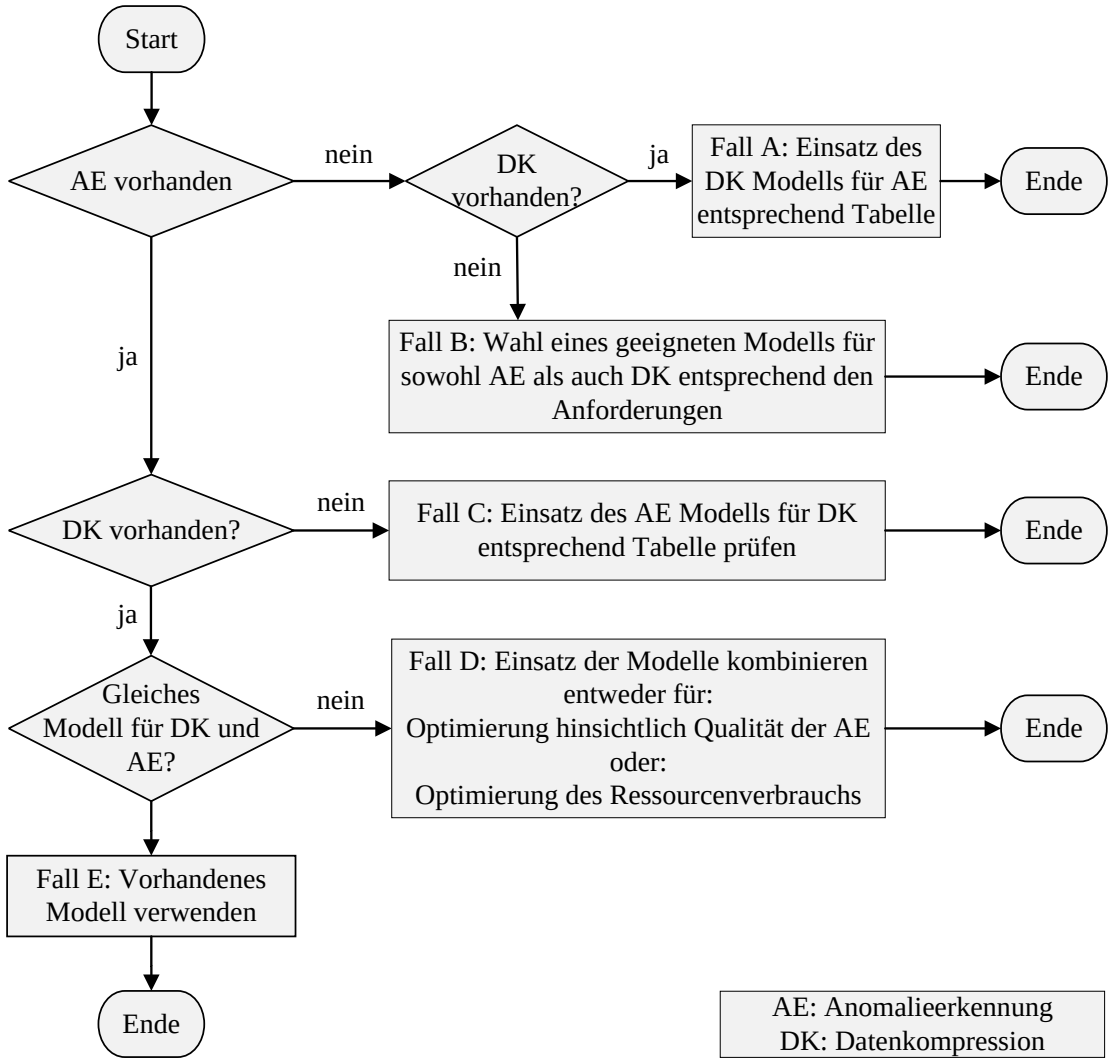


Abbildung 4.15: Allgemeines Vorgehen zum kombinierten Einsatz von Modellen

Sind bereits sowohl ein Modell für die Anomalieerkennung als auch die Datenkompression vorhanden (Fall D), ist zu entscheiden, worauf der Fokus der Optimierung liegt. Es kann einerseits der Ressourcenverbrauch reduziert werden, wenn eines der beiden Modelle für die Anwendung auf beide Aufgaben ausgewählt wird. Andererseits kann die Qualität der Anomalieerkennung durch die doppelte Prüfung auf falsch-positive und falsch-negative Ergebnisse oder die Erkennung zusätzlicher Arten von Anomalien verbessert werden. Dies wird an drei Beispielen anschaulich erläutert.

- Statistisches Modell:
  Ausgangssituation: Es ist ein statistisches Modell in Form einer statistischen Verteilung der Daten vorhanden. Das Modell wird zur Erstellung der Code-Wörter für die Kompression eingesetzt. Seltene Werte erhalten lange Code-Wörter, häufig auftretende Werte werden mit kurzen Codes kodiert.

  Optimierung:
  Erkennung von Punktanomalien: Punktanomalien können entweder über die Auftrittswahrscheinlichkeit erkannt werden, sofern die statistische Verteilung explizit vorliegt. Falls dies nicht der Fall ist, kann die Erkennung implizit über die Länge des Code-Wortes erfolgen. Sehr lange Code-Wörter sind ein Hinweis auf Anomalien.
  Erkennung von Drifts: Drifts können bei statischer, nicht-adaptiver Kodierung durch eine langsame Verschlechterung des durchschnittlichen Kompressionsverhältnisses festgestellt werden.

- Vorhersage-Modell:
  Ausgangssituation: Für die Einsparung von Bandbreite werden sowohl auf der Sender- als auch der Empfängerseite zwei identische Vorhersage-Modelle ausgeführt. Lediglich die Abweichung des Vorhersagewerts vom Realwert wird kodiert und übermittelt.

  Optimierung:
  Erkennung von Punktanomalien: Eine überdurchschnittlich hohe Abweichung, das Delta, ist ein Hinweis auf eine Punktanomalie.
  Erkennung von Drifts: Eine langfristig zunehmende Größe des Deltas zeigt, dass der Realwert von dem Vorhersagemodell stetig abweicht und ein Drift vorliegen kann.

- Autoencoder:
  Ausgangssituation: Der kodierende Teil eines Autoencoders wird für die Kompression von Daten eingesetzt.

  Optimierung: Erkennung von Kontextanomalien: Auf dem selben Endgerät soll eine Anomalieerkennung durchgeführt werden. Die zu betrachtende Metrik für Anomalien ist der Rekonstruktionsfehler nach der Dekodierung. Ein überdurchschnittlich hoher Wert bedeutet, dass das trainierte Modell diese Eingabe nicht erwartet hat, woraus sich ableiten lässt, dass diese nicht im Training aufgetreten ist. Eine seltene, nicht erwartete Eingabe deutet entsprechend den oben genannten Kriterien auf eine Anomalie hin.

4.6 Evaluation

Die vorgestellten Ansätze zur Datenkompression (Abschnitt 4.3) und Anomalieerkennung (Abschnitt 4.4) werden auf der Grundlage von Experimenten bewertet und die Ergebnisse in Abschnitt 4.7 zusammengefasst.

Die Experimente wurden auf einem repräsentativen industriellen Endgerät, einzuordnen in die Steuerungsebene der Automatisierungspyramide, mit nachfolgenden Hardware-Spezifikationen durchgeführt. Die Programmiersprache ist Python und der Code ist im Snap-Format (*Snaps in Ubuntu Core* 2022).

- ctrlX CORE virtual der Firma Bosch Rexroth
- CPU: 64 bit Quad Core AMD-V CPU
- RAM: 4 GB RAM
- Speicher: 4 GB eMMC Memory
- OS: Linux Ubuntu Core 18

4.6.1 Evaluation der Datenkompression

Evaluation der Kompressionsleistung

Die Eignung des vorgestellten LSTM-Autoencoders für die Ausführung auf dem genannten industriellen Endgerät wird im Folgenden evaluiert. Die Ausarbeitungen beruhen auf den Ergebnissen der betreuten Studienarbeit „Performancevergleich von Datenkompressionsalgorithmen auf industriellen Edge-Devices“ (Rauterberg 2022b). Es wird ein Vergleich zu den Verfahren Quantisierung, fpzip und DWT angestellt. Hierzu werden die sequentiellen, dimensionalen und intervallischen Abhängigkeiten der Verfahren in Bezug auf Ressourcenverbrauch und Kompressionsleistung gesetzt.

- Sequentielle Abhängigkeit: Einfluss der Mikrosequenzlänge n auf Kompressionsrate bzw. Rekonstruktionsfehler.
- Dimensionale Abhängigkeit: Einfluss der Dimension m auf Kompressionsrate bzw. Rekonstruktionsfehler.
- Intervallische Abhängigkeit: Einfluss der Senderate auf den Ressourcenverbrauch.

Eindimensionale Daten, also Daten der Dimension $m = 1$, werden als Vektor übergeben. Die Länge des Vektors entspricht der Sequenzlänge n. Mehrdimensionale Daten mit der Dimension m werden dagegen als zweidimensionale Matrix mit der Größe $m \times n$ übergeben.

Für die Evaluation werden unterschiedliche Sequenzlängen ($n = 20, 100$ und 500 Datenpunkte), Senderate (5 ms, 20 ms und 50 ms) sowie Dimensionen ($m = 1, 2$ und 3) gewählt. Die Evaluation erfolgt mit dem Datensatz einer Spritzgussanlage, welcher ebenfalls in der Arbeit von (Sparka u. a. 2017) zur Evaluation einer verlustfreien Datenkompression verwendet wurde.

Die Kompressionsleistung wird nach drei Metriken beurteilt. Die erste ist das Kompressionsverhältnis CR, welches in Gleichung 4.11 definiert ist. Es gibt Verfahren, deren Architektur die Kompressionsrate bereits vorgibt. Sie ergibt sich beim Autoencoder aus der Anzahl der Neuronen der Eingabeschicht im Verhältnis zu der

Anzahl der Engpassschicht. Bei der Quantisierung ergibt sie sich aus dem Verhältnis des ursprünglichen Dateiformats zu dem reduzierten Dateiformat, in diesem Fall float64 zu float16. Bei der DWT wird in dieser Arbeit vorgegeben, dass die 10 % der Koeffizienten mit der größten Energie beibehalten werden. Dadurch ist die Kompressionsrate stark beeinflusst, allerdings hängt sie noch von weiteren Faktoren ab, wie z.B. den Metainformationen, die ebenfalls übertragen werden.

$$CR = \frac{1}{\rho} = \frac{s_{orig}}{s_{comp}} = \frac{d_{orig}\ n_{orig}\ m_{orig}\ b}{d_{comp}\ n_{comp}\ m_{comp}\ b} \tag{4.11}$$

mit
- ρ - Kompressionsfaktor
- s_{orig} - Größe der Originaldaten
- s_{comp} - Größe der komprimierte Daten
- d - Anzahl der Bits eines jeden Datenpunktes
- b - Losgröße
- n - Sequenzlänge
- m - Anzahl der Dimensionen

Die zweite Metrik ist der Rekonstruktionsfehler *RE*, welcher die Genauigkeit der Dekompression beschreibt. Je geringer der Fehler, desto exakter geben die rekonstruierten Daten die Originaldaten wieder. Für verlustfreie Kompressionsverfahren gilt $RE = 0$. In dieser Arbeit wird dafür die Wurzel der mittleren quadratischen Abweichung *RMSE* nach Gleichung 4.12 gewählt. Für die bessere Bewertung der Ergebnisse wird dabei die prozentuale Abweichung *RMSPE* nach Gleichung 4.13 als Metrik betrachtet.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(x_i - \hat{x_i}\right)^2} \tag{4.12}$$

$$RMSPE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{x_i - \hat{x_i}}{d_{max} - d_{min}}\right)^2} \tag{4.13}$$

mit
- x_i - Originaldatum
- $\hat{x_i}$ - Rekonstruiertes Datum
- n - Stichprobengröße
- d_{max} - Maximalwert
- d_{min} - Minimalwert

Bei verlustbehafteten Verfahren führt ein hohes Kompressionsverhältnis im Allgemeinen auch zu höheren Ungenauigkeiten. Um die evaluierten Verfahren besser vergleichen zu können und diesen Kompromiss zu berücksichtigen, wird eine dritte Metrik einbezogen, der Qualitätswert *QS* entsprechend Gleichung 4.14. Da die Kompressionsrate möglichst hoch sein soll und der Rekonstruktionsfehler dabei möglichst gering, ist ein hoher *QS* wünschenswert. Da für die verlustfreien Verfahren immer

$RMSE = 0$ gilt, ist der QS für diese Verfahren nicht anwendbar. Der Vergleich von verlustfreien Verfahren erfolgt allein über das Kompressionsvehältnis.

$$QS = \frac{CR}{RMSPE} \tag{4.14}$$

Daneben ist außerdem die Geschwindigkeit der Kompression eine relevante Größe, die besonders bei der Kompression von Datenflüssen von Bedeutung ist. Es ist zu prüfen, dass die Kompression schneller erfolgt als die Aufzeichnung der Datenpunkte, sodass der Speicher nicht überläuft. Die dafür gewählte Metrik sind Bytes pro Sekunde $[B/s]$ nach (Salomon und Motta 2010) unter Angabe des genutzten Prozessors. Alternativ könnte das prozessorunabhängige Maß CPU Zyklen/ Byte $[CPB]$, d. h. die Anzahl der notwendigen Zyklen für die Kompression eines Bytes, gewählt werden (Chiarot und Silvestri 2023).

In Abhängigkeit des Anwendungsfalls ist auch die Dekompressionsrate relevant. Werden die Daten lediglich abgespeichert oder unabhängig vom Echtzeittakt weiterverarbeitet werden, ist das nicht der Fall. Ist die Kompression allerdings nur für das Weiterleiten der Daten, d. h. zum Einsparen von Bandbreite, relevant und werden die komprimierten Daten noch im IIoT auf einem Endgerät im Echtzeittakt dekomprimiert, ist auch die Dekompressionsrate zu betrachten.

Die Grafiken 4.16 und 4.17 stellen exemplarisch die deutlich vorhandene Sequenzlängenabhängigkeit des Algorithmus fpzip und eine nur geringfügig vorhandene Sequenzlängenabhängigkeit des Autoencoders dar.

Die vergleichenden Ergebnisse der Kompressionsleistung der Verfahren LSTM-Autoencoder, fpzip, DWT und Quantisierung sind in Tabelle 4.2 zusammengefasst. Die detaillierten Ergebnisse der Experimente sowie weitere Visualisierungen der drei Arten von Abhängigkeiten sind (Rauterberg 2022b) zu entnehmen.

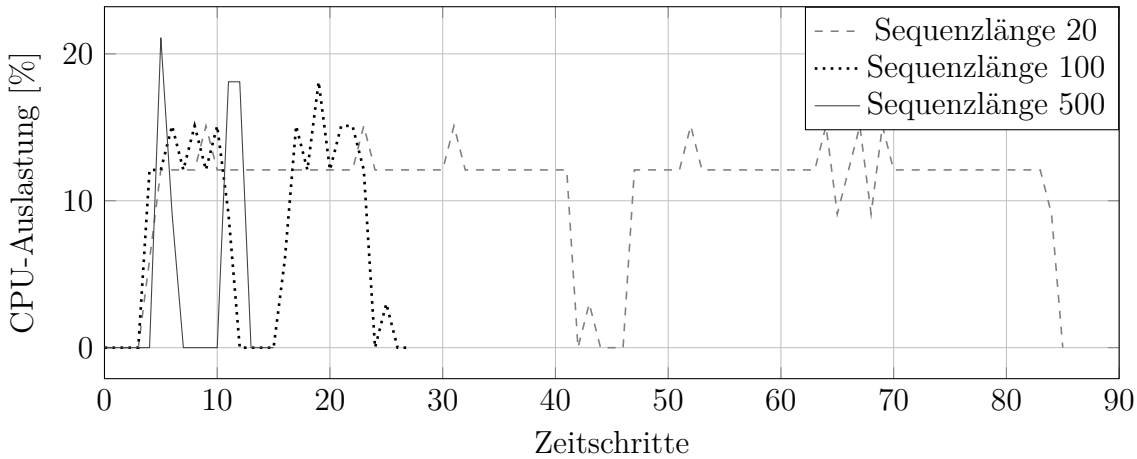


Abbildung 4.16: CPU-Auslastung mit Algorithmus Autoencoder: im Vergleich zu Algorithmus fpzip geringfügige Abhängigkeit von Sequenzlänge (vgl. Abbildung 4.17) (Rauterberg 2022b)

Tabelle 4.2: Vergleich der Kompressionsleistung für unterschiedliche Parametrierungen (Rauterberg 2022b)

m	Algorithmus	$n = 20$			$n = 100$			$n = 500$		
		CR []	RMSPE [%]	QS []	CR []	RMSPE [%]	QS []	CR []	RMSPE [%]	QS []
1	Autoencoder	20,00	76,44	0,26	100,00	76,4	1,31	500,00	76,13	6,57
1	DWT	5,71	0,21	27,19	8,86	0,19	46,63	10,16	0,19	53,47
1	fpzip	1,18	0,00	n.a.	1,53	0,00	n.a.	1,65	0,00	n.a.
1	Quantisierung	4,00	0,02	200,00	4,00	0,02	200,00	4,00	0,02	200,00
2	Autoencoder	20,00	27,96	0,72	100,00	28,03	3,57	500,00	28,5	17,54
2	DWT	2,45	8,14	0,30	2,64	6,71	0,39	2,69	5,95	0,45
2	fpzip	1,01	0,00	n.a.	1,41	0,00	n.a.	1,33	0,00	n.a.
2	Quantisierung	4,00	0,01	400,00	4,00	0,01	400,00	4,00	0,01	400,00
3	Autoencoder	20,00	25,14	0,80	100,00	25,21	3,97	500,00	25,67	19,48
4[a]	DWT	3,83	20,11	0,19	4,33	20,47	0,22	4,46	20,54	0,22
3	fpzip	1,30	0,00	n.a.	1,45	0,00	n.a.	1,44	0,00	n.a.
3	Quantisierung	4,00	0,01	400,00	4,00	0,01	400,00	4,00	0,01	400,00

[a]Aufgrund von internen Formaten der PyWavelet-Bibliothek kann die DWT nicht auf dreidimensionale Daten angewendet werden. Aus diesem Grund werden vierdimensionale Daten verwendet.

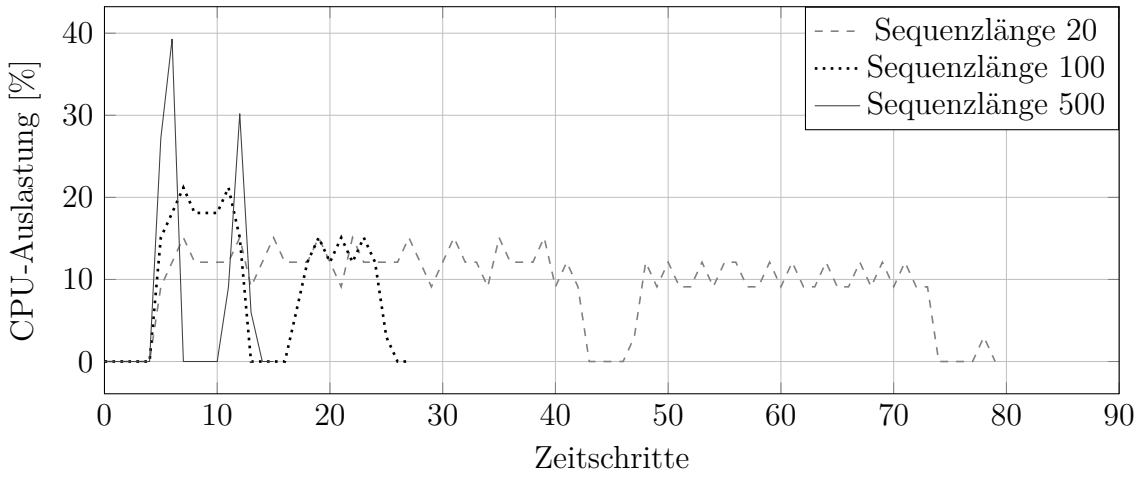


Abbildung 4.17: CPU-Auslastung mit Algorithmus fpzip: im Vergleich zu Algorithmus Autoencoder höhere Abhängigkeit von Sequenzlänge (vgl. Abbildung 4.16) (Rauterberg 2022b)

Aus den Ergebnissen lassen sich folgende Schlüsse über die sequentielle, dimensionale und intervallische Abhängigkeit ziehen (Rauterberg 2022b):

- Sequentielle Abhängigkeit: Der Algorithmus fpzip hat eine deutliche Abhängigkeit zwischen CPU-Auslastung und Sequenzlänge gezeigt. Grund dafür ist die punktweise Verarbeitung anstelle der Verarbeitung von Mikrosequenzen, d.h. vektor- bzw. matrixbasiert, wie bei Autoencoder, DWT und Quantisierung.
- Dimensionale Anhängigkeit: Die Rechenintensität von Autoencoder, DWT und Quantisierung sind nur geringfügig von der Dimension der Daten abhängig, wogegen bei der Kompression mittels fpzip eine deutliche Abhängigkeit besteht.
- Intervallische Abhängigkeit: Die intervallische Abhängigkeit beschreibt den Einfluss der Senderate auf die Rechenintensität. Dieser Zusammenhang ist bei allen Algorithmen festzustellen.

Evaluation des Ressourcenverbrauchs

Der Ressourcenverbrauch der Verfahren wird gegenübergestellt und die Anwendbarkeit auf einem industriellen Endgerät überprüft. Dazu werden der Verbrauch von CPU und RAM sowie die Geschwindigkeit der Kompression und Dekompression gemessen. Die Ergebnisse sind in den Tabellen 4.3-4.5 dargestellt.

Alle untersuchten Modelle weisen eine hohe Geschwindigkeit bei der Datenflusskompression auf und sind für die Anwendbarkeit auf Datenflüsse und den Einsatz auf industriellen Endgeräten geeignet. Für die Bewertung der Echtzeitfähigkeit wird geprüft, ob die Zeitspanne $t_{buffering}$ für das Puffern der Daten einer Sequenz größer ist als die benötigte Zeit für das Komprimieren $t_{compression}$ der Daten. Es gilt:

$$t_{buffering} = \sum_{i=0}^{N} t_i$$

Tabelle 4.3: Ressourcenverbrauch der Kompressionsalgorithmen (Rauterberg 2022b)

Algorithmus	CPU max. [%]	RAM max. [%]	Konfiguration[a]
Kompression			
Autoencoder	45,2	3,97	I: 5 ms, S: 100, D: 3
DWT	69,6	3,93	I: 5 ms, S: 20, D: 2
fpzip	72,6	3,88	I: 5 ms, S: 500, D: 3
Quantisierung	51,4	3,80	I: 5 ms, S: 500, D: 3
Dekompression			
Autoencoder	51,4	3,97	I: 5 ms, S: 100, D: 3
DWT	66,6	3,93	I: 5 ms, S: 20, D: 2
fpzip	51,4	3,88	I: 5 ms, S: 500, D: 3
Quantisierung	9,1	3,80	I: 5 ms, S: 500, D: 3

[a]I: Intervall, S: Sequenzlänge, D: Dimensionalität

Tabelle 4.4: Laufzeiten der Kompressionsalgorithmen (Rauterberg 2022b)

		$n = 20$		$n = 100$		$n = 500$	
m	Algorithmus	t_{avg} [ms]	t_{max} [ms]	t_{avg} [ms]	t_{max} [ms]	t_{avg} [ms]	t_{max} [ms]
1	Autoencoder	5,65	9,14	6,24	9,0	9,39	15,5
1	DWT	8,53	27,82	8,99	13,68	13,19	16,03
1	fpzip	5,69	9,8	8,09	18,06	16,29	19,48
1	Quantisierung	5,31	19,76	5,93	9,26	10,33	12,9
2	Autoencoder	5,9	11,18	6,93	10,12	13,15	15,93
2	DWT	11,02	23,02	13,23	17,34	25,42	28,14
2	fpzip	6.39	9,99	10,58	13,25	30,96	34,85
2	Quantisierung	5,66	16,27	7,24	10,58	16,03	19,04
3	Autoencoder	5,88	11,43	7,97	13,22	16,44	19,17
4[a]	DWT	13,69	42,15	15,03	24,10	33,61	36,28
3	fpzip	6,99	11,17	12,91	19,17	41,2	47,18
3	Quantisierung	5,92	15,41	8,35	11,12	21,49	25,27

[a]Aufgrund von internen Formaten der PyWavelet-Bibliothek kann die DWT nicht auf dreidimensionale Daten angewendet werden. Aus diesem Grund werden vierdimensionale Daten verwendet.

Tabelle 4.5: Laufzeiten der Dekompressionsalgorithmen (Rauterberg 2022b)

m	Algorithmus	$n = 20$		$n = 100$		$n = 500$	
		t_{avg} [ms]	t_{max} [ms]	t_{avg} [ms]	t_{max} [ms]	t_{avg} [ms]	t_{max} [ms]
1	Autoencoder	5,69	11,2	6,47	9,43	10,32	12,53
1	DWT	5,98	11,64	6,87	10,48	10,18	13,06
1	fpzip	5,5	20,92	6,42	10,32	12,62	14,63
1	Quantisierung	4,58	19,21	4,67	8,78	5,9	8,59
2	Autoencoder	5,9	10,4	7,39	9,82	14,45	17,89
2	DWT	9,09	13,46	11,45	15,18	21,96	24,84
2	fpzip	6,01	16,88	8,28	12,37	21,13	23,71
2	Quantisierung	4,51	14,11	5,12	11,37	7,73	13,95
3	Autoencoder	5,98	10,66	8,2	12,28	19,15	21,91
4[a]	DWT	11,49	25,62	13,31	17,59	30,86	35,23
3	fpzip	5,94	10,52	9,63	14,66	27,84	32,29
3	Quantisierung	4,59	16,71	5,69	15,24	8,58	10,93

[a]Aufgrund von internen Formaten der PyWavelet-Bibliothek kann die DWT nicht auf dreidimensionale Daten angewendet werden. Aus diesem Grund werden vierdimensionale Daten verwendet.

Betrachtet man die für die Datenkompression benötigten durchschnittlichen Geschwindigkeiten je Sequenz t_{avg} nach Tabelle 4.4, so sind alle Verfahren für Senderaten von 1 ms je Datenpunkt, für Sequenzlängen $n = 500$ sogar für Senderaten von 0,1 ms je Datenpunkt geeignet. Unter Berücksichtigung der maximalen Zeitspannen t_{max} für die Kompression sind die Geschwindigkeiten der meisten Verfahren hoch genug, um in dem beschriebenen Versuchsaufbau Senderaten von 2 ms je Datenpunkt einzuhalten. Einzige Ausnahme ist die DWT für vier Dimensionen und einer Sequenzlänge $n = 20$, bei der die maximale Verarbeitungszeit (42,15 ms) in den Experimenten bei durchschnittlich 2,11 ms pro Datenpunkt und somit bei dem Dreifachen der durchschnittlichen Verarbeitungszeit lag.

4.6.2 Evaluation der Anomalieerkennung

Das Verfahren zur Anomalieerkennung EEM-KDE wird auf die bekannten und öffentlich verfügbaren Numenta Anomaly Benchmark (NAB)-Datensätze für Datenflüsse (Lavin und Ahmad 2015) sowie auf Datensätze eines industriellen Demonstrators (test bed (TB)) angewendet, die in Tabelle 4.6 aufgeführt sind.

In den Experimenten werden drei Aspekte untersucht, um die Anomalieerkennung zu bewerten.

Tabelle 4.6: Für die Evaluation der Anomalieerkennung ausgewählte Datensätze

Datensatz	NAB 1.1-1.5	NAB 2.1-2.6	NAB 3.1-3.5	TB 1-3
Ursprung	synthetisch	synthetisch	real	Demonstrator
Anomalie enthalten	nein	ja	ja	ja
Zyklische Daten	3 von 5	4 von 6	1 von 5	3 von 3
Anzahl der Signale	1	1	1	3

1. Wahl der drei grundlegenden Hyperparameter:
   Wie die Experimente von (Zhou u. a. 2003) zeigten, wird die Fehlerrate von der Anzahl der Kerne beeinflusst. Mehr Kerne können die reale Werteverteilung besser abbilden und die Ergebnisse verbessern, erfordern aber - anders als eine Erhöhung der Bandbreite - auch eine linear steigende Anzahl von Berechnungen. Standardmäßig wird in dieser Arbeit eine Kernanzahl von $k = 10$ gewählt, um eine geringe Rechenzeit zu garantieren. Die Bandbreite dagegen variiert von Datensatz zu Datensatz. Die Wahl der Kernfunktion kann ebenfalls einen Einfluss auf die Ergebnisse haben. In dieser Arbeit wird durchgehend der Gauß-Kern eingesetzt.

2. Anzahl der Iterationen bis zum Beginn der Anomalieerkennung:
   Das Erreichen des stabilen Zustands und damit eine ausreichend gute Annäherung an die reale Dichteverteilung ist Voraussetzung für die Erkennung von Anomalien (siehe Abschnitt 4.4). Andernfalls kann keine gültige Aussage über Anomalien getroffen werden. Für den stabilen Zustand wird die Bedingung gestellt, dass der Driftwert über einen Bereich von 15 Signalwerten 20-mal eine Schwelle von 15% nicht überschreiten darf. Der stabile Zustand kann somit nicht vor 300 Signalwerten erreicht werden. Für die Erkennung kollektiver Anomalien muss zusätzlich der Zyklus erkannt und verifiziert werden.

3. Falsch-positive oder falsch-negative Erkennung von Anomalien und Zyklen.

Die Ergebnisse der Anomalieerkennung für die in Tabelle 4.6 genannten Datensätze, sind in den Tabellen 4.7-4.9 zusammengefasst. Aufgrund der stark unterschiedlichen Wertebereiche zwischen den Datensätzen wurde die Anzahl der Kerne und die Bandbreite angepasst.

Der stabile Zustand wird durchschnittlich nach 497 Datenpunkten erreicht. Der Algorithmus erkennt je nach Datensatz nach durchschnittlich 4 bis 6 Zyklen, ob zyklische Daten vorliegen oder nicht. Verrauschte Daten erschweren die Zyklenerkennung. In dem Datensatz NAB 3.1 konnte aufgrund eines durchschnittlichen Rauschens von $\pm$ 9% in einer Region mit höherer Dichte, und somit auch kleineren Segmenten im adaptiven Gitter, mit der ursprünglichen Einstellung kein Zyklus erkannt werden. Ein Tiefpassfilter über zehn Werte und die Verwendung einer statischen Ras-

terung führen zur Erkennung des Zyklus. Bei Anwendung auf weniger verrauschte Datensätze zeigt sich jedoch, dass dieses Vorgehen vermehrt zu falsch-positive Anomalien führt und somit keine allgemeingültige Lösung darstellt. Kollektive Anomalien (NAB 3.2 und 3.5), die in nicht-zyklischen Daten auftreten, werden entsprechend der Natur des Algorithmus (Erkennung von kollektiven Anomalien auf Basis der Abweichung vom Zyklus) nicht erkannt. Die Relevanz der Parameter „Kernanzahl“ und „Bandbreite“ zeigt sich in NAB 3.4. Hier wird eine Punktanomalie nicht detektiert aufgrund einer zu groß gewählten Bandbreite.

In den industriellen Demonstrator-Datensätzen TB1 - TB3 werden drei Anomalien erkannt. Die erste ist auf eine Veränderung des Arbeitspunktes zurückzuführen, die sowohl als Drift als auch kollektive Anomalie sichtbar ist. Bei der zweiten handelt es sich um eine anormale Spitze innerhalb eines Signals, und die dritte hat ihren Ursprung in Spiel und Drift. Aufgrund der sehr langen Zyklen in TB3 dauerte es nur 3,12 Zykluslängen, bis die Erkennung der kollektiven Anomalie begann, aber die absolute Anzahl der Iterationen ist jedoch im Vergleich zu den kürzeren Zyklen in den NAB-Datensätzen deutlich länger.

Tabelle 4.7: Ergebnisse der Evaluation der Anomalieerkennung (NAB 1-2) (nach Rosenberger, Müller u. a. 2022)

Datensatz	Anzahl der Kerne k	Bandbreite h	erkannte Anomalien
NAB 1.1 - 1.5	10	15	0 von 0 Anomalien
NAB 2.1 - 2.6	10	20	8 von 8 Anomalien
Stabiler Zustand erreicht nach ∅ 453 Iterationen			
Falsch-positive Anomalien: 0			
Falsch-negative Anomalien: 0			

Der zweite Teil der Evaluation betrifft die Ressourcenauslastung (CPU und RAM) des Endgeräts sowie die benötigte Verarbeitungszeit. Die Ergebnisse für drei unterschiedliche Konfigurationen der Kernanzahl sind in Tabelle 4.10 zusammengefasst.

4.7 Ergebnisse und Diskussion

Zusammenfassend wurden verschiedene Modelle für die Datenreduktion und Informationsextraktion in Datenflüssen auf industriellen Endgeräten vorgestellt. Die Algorithmen sind für die Anwendung auf der Logik-Ebene von Datenfluss-Systemen ausgelegt und deren Anwendbarkeit auf Endgeräten der Steuerungsebene bestätigt.

Tabelle 4.8: Ergebnisse der Evaluation der Anomalieerkennung (NAB 3) (nach Rosenberger, Müller u. a. 2022)

Datensatz	Anzahl der Kerne k	Bandbreite h	erkannte Anomalien
NAB 3.1	10	20	3 von 3 Punktanomalien, 2 von 2 Drifts
NAB 3.2	10	15	0 von 1 kollektive Anomalie, 1 von 1 Drift
NAB 3.3	10	15	3 von 3 Punktanomalien
NAB 3.4	20	20	3 von 4 Punktanomalien, 1 von 1 kollektiven Anomalien
NAB 3.5	10	15	2 von 2 Punktanomalien, 2 von 3 kollektiven Anomalien

Stabiler Zustand erreicht nach ∅ 603 Iterationen.
Zyklus erkannt (falls vorhanden) nach 5,20 Zykluslängen.
Falsch-positive Anomalien: 3
Falsch-negative Anomalien: 1 Punkt- und 2 kollektive Anomalien (nicht-zyklisch)

Tabelle 4.9: Ergebnisse der Evaluation der Anomalieerkennung (TB) (nach Rosenberger, Müller u. a. 2022)

Datensatz	Anzahl der Kerne k	Bandbreite h	erkannte Anomalien
TB 1	10 (1-D) 250 (3-D)	15	1 kollektive Anomalie, 1 Drift
TB 2	10 (1-D) 250 (3-D)	15	1 Punktanomalie, 1 kollektive Anomalie, 1 Kontextanomalie
TB 3	15 (1-D) 250 (3-D)	20	1 Punktanomalie, 1 kollektive Anomalie, 1 Kontextanomalie

Stabiler Zustand erreicht nach ∅ 435 Iterationen
Zyklus erkannt (falls vorhanden) nach 3,12 Zykluslängen
Falsch-positive Anomalien: 0
Falsch-negative Anomalien: 0

Tabelle 4.10: Ergebnisse der Evaluation der Performanz des EEM-KDE Verfahrens (Rosenberger, Müller u. a. 2022)

Ausführungsform	∅ Verarbeitungszeit [ms]	max. CPU [%]	max. RAM [%]
10 Kerne (eindimensional)	2,73	27,01	6,20
20 Kerne (eindimensional)	6,17	28,47	6,17
10 Kerne (eindimensional) und 250 Kerne (mehrdimensional)	326,27	28,88	7,20

Datenreduktion mittels Datenkompression
Verschiedene bestehende Verfahren wurden für die Anwendung auf UTS und MTS angepasst und deren Einsatz auf industriellen Endgeräten hinsichtlich Ressourcenverbrauch und Kompressionsqualität evaluiert. Die Evaluation der Datenkompression zeigt, dass...

1. die Quantisierung ein sehr simples, aber effektives Verfahren zur Datenreduktion darstellt. Sie hat über alle Experimente hinweg den höchsten Qualitätswert QS erzielt.

2. das verlustfreie Verfahren fpzip für den Anwendungsfall geeignet ist und eine Alternative darstellt, wenn ein Datenverlust nicht akzeptabel ist. Dabei stellt die verlustfreie Kompression einen Kompromiss mit verhältnismäßig geringen Kompressionsraten dar.

3. die DWT im eindimensionalen Raum höhere Kompressionsraten erzielt als die Quantisierung bei einem Rekonstruktionsfehler von unter 1 %. Im mehrdimensionalen Raum dagegen nimmt die Kompressionsrate ab, bei gleichzeitig stark ansteigendem Rekonstruktionsfehler. Die DWT ist somit nicht für den mehrdimensionalen Raum empfehlenswert.

4. der LSTM-Autoencoder zu deutlich schlechteren Ergebnissen führt als die anderen Verfahren. Die in (Rauterberg 2022b) vorgestellte Evaluation des LSTM-Autoencoders ist kritisch zu betrachten. Eine mögliche Begründung kann die Umwandlung des Modells in ein TensorFlow Lite-Modell sein, welches zwar explizit für den Einsatz auf Endgeräten und Mikrocontrollern empfohlen wird, allerdings für Modelle basierend auf RNN zu schlechteren Ergebnissen führen kann, da der Zustand der LSTM-Zellen regelmäßig zurückgesetzt wird. Dennoch ist ein Trend zu erkennen, dass der Autoencoder im mehrdimensionalen Raum bei längeren Sequenzlängen bessere Ergebnisse erzielt.

Während die Ergebnisse der Arbeit (Rauterberg 2022b) gegen den Einsatz des LSTM-Autoencoders sprechen, ist die Kompressionsqualität nochmal mit dem origi-

nalen TensorFlow- anstelle des TensorFlow Lite-Modells zu prüfen. Außerdem ist der verlustfreie Modus von fpzip mit den anderen verlustbehafteten Verfahren schlecht vergleichbar, weshalb zusätzlich ein verlustbehafteter Modus zu evaluieren ist.

Informationsextraktion mittels Anomalieerkennung
Die Experimente bestätigen, dass das entwickelte EEM-KDE Verfahren für die Erkennung von Anomalien in industriellen Datenflüssen geeignet ist. Die gewählten Hyperparameter führten zu guten Ergebnissen. Generell hat sich gezeigt, dass eine Anzahl von zehn bis fünfzehn univariaten Kernen und die Anpassung deren Bandbreite an den Wertebereich häufig ausreichend ist. Die Kernanzahl k und ihre Bandbreite h sind so zu wählen, dass ihr Produkt in etwa dem Wertebereich des Signals entspricht. Bei der Erkennung von Kontextanomalien stellte sich heraus, dass die Anzahl der Kerne deutlich erhöht werden muss, um eine Unteranpassung zu vermeiden. Für drei Dimensionen führte erst ein Standardwert von mindestens 250 Kernen und eine erhöhte Bandbreite zu zufriedenstellenden Ergebnissen. Der stabile Zustand wurde bereits nach wenigen hundert Iterationen erreicht, was für das industrielle Umfeld als ausreichend erachtet wird. Eine hohe Anzahl von Anomalien wurde korrekt erkannt. Die Erkennung kollektiver Anomalien ist bisher auf zyklische Daten beschränkt, könnte aber mit einem gleitenden Mittelwert über die Dichteverteilung auf andere Fälle ausgedehnt werden. Bei verrauschten Signalen verbessert ein Tiefpassfilter die Zykluserkennung.

Insgesamt lässt sich zusammenfassen, dass das EEM-KDE Verfahren eine gute Leistung bei der Erkennung von Anomalien in einer einzigen Dimension erzielt. Es ist für Abtastraten von wenigen Millisekunden mit nur zehn Kernen einsetzbar. Mit weniger als 30% maximaler CPU- und 8% maximaler RAM-Auslastung kann die Anomalieerkennung auf Industriesteuerungen betrieben werden, während noch Kapazität für die Hauptaufgaben wie die Prozesssteuerung vorhanden ist. Allerdings verlangsamt eine wesentlich höhere Anzahl von Kernen für die Anomalieerkennung im mehrdimensionalen Raum den Prozess. Daher wird die mehrdimensionale Anomalieerkennung mit EEM-KDE auf leistungsfähigeren Endgeräten, z. B. Industrie-PCs, empfohlen, und die Anzahl der Dimensionen ist auf das Notwendigste zu reduzieren, z. B. durch Merkmalsextraktion.

Das EEM-KDE Verfahren stellt eine Lösung für eine umfassende Anomalieerkennung auf Endgeräten in der Industrie auf der Logik-Ebene dar. Sie erfüllt die spezifizierten Anforderungen wie Echtzeitfähigkeit, Verarbeitung von Datenflüssen und Generik. Weiteres Optimierungspotenzial besteht in der Erweiterung der kollektiven Anomalieerkennung für nicht-zyklische Daten sowie in der Reduktion der Komplexität der Erkennung von Kontextanomalien, um mehr Dimensionen zu berücksichtigen und die Rechenlast zu reduzieren. Ein alternativer Ansatz für die Zykluserkennung, bei dem die Abfolge der Vereinigung der Kerne kodiert wird, ist weiterzuverfolgen. Zusätzliche Untersuchungen mit weiteren industriellen Daten sowie Tests in realen Anlagen und auf weiteren Endgeräten sind anzustellen.

Kombinierter Modelleinsatz
Der in der Arbeit (Rosenberger, Bühren und Schramm 2021) vorgeschlagene kombinierte Modelleinsatz wurde im Rahmen dieser Arbeit nicht evaluiert. Es wird vorgeschlagen, das Konzept zukünftig exemplarisch an dem zuvor beschriebenen LSTM-Autoencoder zu evaluieren. Da das EEM-KDE Verfahren im mehrdimensionalen Bereich sehr rechenintensiv ist, könnte der Autoencoder besonders für MTS eine performante Alternative darstellen.
Der Einsatz für die Anomalieerkennung erfordert zusätzlich zu der Kompression auch die Dekompression und die Berechnung des Rekonstruktionsfehlers auf dem selben Endgerät. Die zusätzliche CPU Nutzung erfolgt im Anschluss an die Kompression und muss daher nicht als zusätzliche Last aufaddiert werden, allerdings verlängert sich die Zeitspanne der Berechnungen um die Zeit für die Dekompression und Berechnung des Rekonstruktionsfehlers, $t_{buffer} < (t_{compression} + t_{decompression} + t_{RMSE})$, sodass geprüft werden muss, ob bzw. ab welcher Sequenzlänge der Echtzeittakt eingehalten werden kann.

Die vorgestellten Ausarbeitungen zur Mengenoptimierung von Datenflüssen können in folgenden realen Szenarien Anwendung finden.

- Reduktion der zu transferierenden Datenmenge innerhalb einer Anlage oder zwischen Anlage und Rechenzentren zur Verhinderung von Datenverlusten durch Bandbreitenengpässe sowie Verringerung des benötigten Speicherplatzes.
- Extraktion von relevanten Informationen, beispielsweise über den Zustand von Anlage, Komponenten oder Produkt, als Grundlage für eine vorausschauende Instandhaltung. Die Verarbeitung am Netzwerkrand minimiert die Latenzen zwischen Datenerfassung und Bereitstellung der Information, sodass sie für zeitkritische Datenanalysen geeignet ist. Außerdem ist gegenüber der Verarbeitung in der Cloud eine höhere Datensicherheit gegeben und die Verarbeitung am Netzwerkrand somit für die Analyse von sensiblen Produktionsdaten prädestiniert.

KAPITEL 5

Absicherung von Daten mittels IOTA

Dieses Kapitel beschreibt Einsatzmöglichkeiten der DLT IOTA zur Schaffung von Vertrauen durch Absicherung von Daten in der I4.0. Dazu werden zwei Anwendungsfälle, zum einen die unveränderliche Aufzeichnung von Historien, zum Anderen eine Zugriffsrechteverwaltung unter Einsatz von SC, erarbeitet. Im Anschluss wird die Umsetzung beider Anwendungsfälle in einem repräsentativen IIoT-Netzwerk evaluiert und die Ergebnisse diskutiert.

5.1 Auswahl der Technologie

Im Rahmen digitaler Geschäftsmodelle ist die Absicherung von Informationen unabdingbar. Um Vertrauen in einer zunehmend anonymen, automatisierten und digitalisierten Geschäftswelt zu schaffen, steigt der Bedarf an der Bereitstellung von unanfechtbaren Nachweisen, beispielsweise der Einhaltung einer vertraglich zugesicherten Leistungserbringung. Dies stellt eine neue, nicht-triviale Aufgabe dar. Zudem ist sicherzustellen, dass wertvolle Daten und Informationen nur berechtigten Teilnehmern zur Verfügung gestellt werden. Die Absicherung von Informationen betrifft im Kontext dieser Arbeit dementsprechend zwei Aspekte. Eine Zusammenfassung ist in Abbildung 5.1 dargestellt.

1. Abgesicherte Aufzeichnung von ausgewählten, kritischen Information
2. Abgesichertes Bereitstellen

Der erste Anwendungsfall, die Absicherung von Aufzeichnungen, bezeichnet in diesem Kontext primär die *Unveränderlichkeit*, *Dezentralität* und *Transparenz* einer

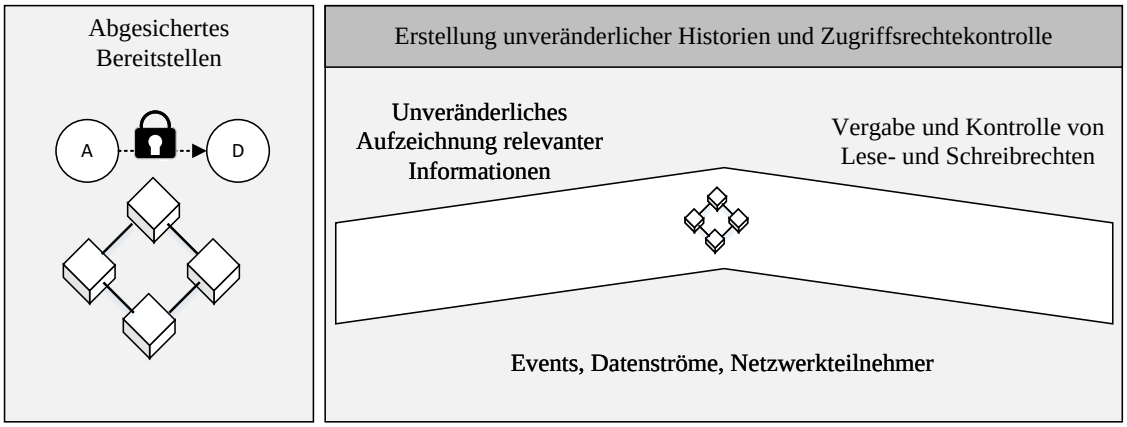


Abbildung 5.1: Schwerpunkte zur Absicherung von Daten und Informationen

Aufzeichnung und wird in Abschnitt 5.3 behandelt. Diese Anforderungen überschneiden sich stark mit den in Abschnitt 2.7 beschriebenen Eigenschaften von DLT. Die Konvergenz von IoT und DLT wird bereits mehrfach hervorgehoben (Farahani, Firouzi und Luecking 2021; Pinheiro u. a. 2018; Wu, Dai und Wang 2021) und DLT als eine der Kerntechnologien der I4.0 bezeichnet (ElMamy u. a. 2020), da sie in der Lage ist, wesentliche Sicherheitsmängel zu beheben, die Datenqualität zu erhöhen und den Einsatz von datenbasierten Diensten im IIoT voranzutreiben. Eine wichtige Voraussetzung für die Anwendung in Produktivsystemen ist die Bewertung des Ressourcenbedarfs und der Mehraufwände durch die Nutzung von IIoT-Geräten als Knoten in einem DLT-Netzwerk. Während viele Anwendungsfälle identifiziert und die Vorteile von DLT im IIoT ausführlich argumentiert sind (Farahani, Firouzi und Luecking 2021; Wu, Zhou u. a. 2019; Raschendorfer u. a. 2019; Lupascu, Lupascu und Bica 2020), fehlt es an ausreichenden Experimenten zur Anwendbarkeit in industriellen Automatisierungssystemen, insbesondere hinsichtlich der Performanz.
Der zweite Anwendungsfall, die Absicherung der Bereitstellung, betrifft die Prüfung der Berechtigung, insbesondere Lese- und Schreibrechte, der einzelnen Teilnehmer (siehe Abschnitt 5.4). Eine mögliche Lösung stellt der Einsatz von in Software eingebetteten Vertragsklauseln, welche an eine DLT angebunden sind, den sogenannten *SC*, dar. Die Anwendbarkeit von SC in der ressourcenlimitierten IIoT Umgebung wird in dieser Arbeit ebenfalls evaluiert.

Aus den in Abschnitt 2.7.1 genannten Argumenten wurde für die Ausarbeitungen in dieser Arbeit die DLT IOTA gewählt. Aufgrund der Aktualität der IOTA Versionsupdates ist im Rahmen dieser Arbeit eine der ersten Studien zu IOTA 1.5 und IOTA 2.0 für den Einsatz in der Industrie entstanden (Rosenberger, Rauterberg und Schramm 2021), welche auf der IEEE Global Conference for Artificial Intelligence and Internet of Things 2021 vorgestellt wurde. Dazu wurde die Anwendbarkeit der neuen Versionen IOTA 1.5 und 2.0, sowie der Einsatz von SC im IIoT Kontext mit besonderem Fokus auf die Ressourcenauslastung der Endgeräte und zusätzli-

che Bandbreitenauslastung durch die DLT evaluiert. Die folgenden Ausführungen basieren auf den Inhalten dieser Veröffentlichung.

5.2 Stand der Technik

Dieser Abschnitt beschreibt den Stand der Technik zu dem Einsatz von DLT in der Industrie, insbesondere der Evaluation von bestehenden DLT unter Beachtung der besonderen Anforderungen im IIoT.

Verschiedene Studien untersuchen die Leistung, wie unter anderem Messungen der möglichen Transaktionen pro Sekunde (engl. Transactions Per Second (TPS)), mit Bezug auf die DLT selbst, z.B. für Aussagen über die Skalierbarkeit der Technologie. Dafür werden verschiedene Benchmark-Systeme, darunter BlockBench[1], Hyperledger Caliper[2] und DAGbench[3], entwickelt, von denen das Letztgenannte für Leistungstests auf DAG-basierten Rahmenwerken geeignet ist.

Unter anderem zeigen die Studien von (Fan, Ghaemi u. a. 2020) und (Han u. a. 2020), dass sich die meisten Studien auf Hyperledger konzentrieren, während für IOTA nahezu keine Analysen durchgeführt werden. Mögliche Gründe dafür können einerseits die Verfügbarkeit des Hyperledger Caliper Benchmarks für Hyperledger und andererseits die bereits erwähnten Mängel der ersten Version von IOTA darstellen. (Kusmierz u. a. 2019) bestätigen die Neuheit und das Fehlen von umfangreichen Studien für IOTA.

Eine der wenigen Ausnahmen ist die Studie von (Fan, Khazaei u. a. 2019), die eine empirische Evaluation des IOTA 1.0-Rahmenwerks für Smart Home-Anwendungen erarbeiteten. Für die Tests besteht das Netzwerk aus 40 Knoten. Die gewählte Hardware verfügt über mehr Rechenleistung und deutlich mehr Speicherkapazität als das in dieser Arbeit gewählte repräsentative industrielle Endgerät. Darüber hinaus werden in (Jiang u. a. 2018) detaillierte Ergebnisse von empirischen Leistungstests mit verschiedenen Hardware-Aufbauten vorgestellt, allerdings liegt der Fokus der Experimente auf der vorgestellten Consortium-Blockchain, einer Kombination von Hyperledger und IOTA. Abgesehen davon ist die Anzahl der IOTA-Knoten nur vier und damit zu klein für ein realistisches IIoT-Szenario. (Jogunola u. a. 2020) weisen nach, dass IOTA die verbreitete Blockchain Ethereum in einer Anwendung zum Energie-Handel übertrifft. Da IOTA zum Zeitpunkt der Durchführung der Studie allerdings noch über keine SC-Funktionalität verfügte, wurde Hyperledger für die SC verwendet. Die zunehmende Bedeutung von SC zeigt sich auch in verschiedenen Studien (Rouhani und Deters 2019; Aldweesh u. a. 2018; Zheng u. a. 2018). Aufgrund der Neuheit von IOTA 2.0 wird IOTA im Kontext von SC allerdings noch nicht in bestehenden Arbeiten berücksichtigt.

Insgesamt sind nur wenige Studien zu der Performanz von IOTA veröffentlicht. Alle genannten Leistungstests basieren auf IOTA 1.0 und sind daher veraltet. Dar-

[1] https://github.com/ooibc88/blockbench
[2] https://hyperledger.github.io/caliper/
[3] https://github.com/InnoLabsAU/DAGBENCH

über hinaus zielen die Evaluationen meist auf die Leistung des DLT-Rahmenwerks selbst ab, ohne die Anwendbarkeit auf das gesamte Zielsystem zu bewerten. Besonders IIoT-Geräte sind in ihren Ressourcen wie Rechenleistung und Speicher, sowie teilweise auch Energie, begrenzt, weshalb diese relevant für die Evaluation sind.
Bisher existiert dementsprechend keine aktuelle Studie, die den Einsatz von IOTA 1.5 und IOTA 2.0 im industriellen Umfeld evaluiert. Die in dieser Arbeit vorgestellte umfassende empirische Leistungsanalyse anhand von zwei industriellen Anwendungsfällen mit Fokus auf die Auswirkungen auf die Ressourcen der IIoT-Geräte ist neu.

5.3 Aufzeichnung von Historien

5.3.1 Beschreibung des Anwendungsfalls

Unverfälschliche Nachweise sind in vielerlei Hinsicht relevant. Beispielsweise gibt es rechtliche Anforderungen, die eine Vorlage der Dokumentation bestimmter Abläufe und Prüfungen noch viele Jahre später verlangen. Ein Beispiel sind Wartungs- und Instandhaltungsmaßnahmen an Schienenfahrzeugen. Eine weitere Anwendung ist die Absicherung durch den Nachweis über die Einhaltung von vertraglichen Vereinbarungen, z. B. als Qualitätsnachweis über den Durchlauf aller Qualitätsprüfpunkte in einem Prozess oder als Nachweis der Fertigungsprozesstreue. Insbesondere für Produkte, bei denen im Schadensfall eine Personengefährdung vorliegt, kann dies von hoher Tragweite sein. Kommt es beispielsweise zu einem Schadensfall durch ein defektes Bauteil, kann dem Auftraggeber gegenüber nachgewiesen werden, dass die Produktion des Bauteils entsprechend den vereinbarten Vorgaben stattgefunden hat und der Defekt nicht dem Produzenten anzulasten ist.

Trotz der fortschreitenden Digitalisierung werden solche relevanten Nachweise häufig nicht automatisiert und teilweise noch in Papierform aufbewahrt. Der Einsatz von DLT bietet hier enormes Verbesserungspotential, da Aufzeichnungen unverfälschbar, automatisiert, effizient und transparent gespeichert werden. Im Gegensatz zu Nachweisen auf Papier können weder Datums-/Zeitangaben verfälscht werden und der Verlust der abgespeicherten Daten ist erschwert, sodass die Vertrauenswürdigkeit der Daten steigt. (Rosenberger, Rauterberg und Schramm 2021)

Im Falle von IOTA findet das sogenannte *Pruning* statt. Das heißt, nach einer festgelegten Anzahl von Meilensteinen werden ältere, bereits bestätigte Nachrichten aus dem Tangle entfernt. Aus diesem Grund ist der Tangle selbst nicht für die Anforderungen der unveränderlichen Aufzeichnung von Historien geeignet, da die enthaltenen Nachrichten nicht für unbegrenzte Zeit zur Verfügung stehen. Eine Lösung dafür stellt die sogenannte Permanode-Funktionalität von IOTA dar. Dazu ist ein zusätzlicher Knoten-Typ, der Chronicle-Knoten, sowie eine verteilte Datenbank, die SkyllaDB, erforderlich. Beides ist nach aktuellem Stand (Juli 2023) nur mit IOTA 1.5 möglich.

Der erste Anwendungsfall beschreibt das langfristige, unveränderliche Absichern von ausgewählten Daten mittels der IOTA Permanode-Funktionalität durch ein DLT Netzwerk auf industriellen Endgeräten.

5.3.2 Beschreibung der Architektur

In diesem Abschnitt wird die Architektur des erstellten IOTA 1.5 Netzwerkes für den Anwendungsfall 1 beschrieben. Wie in Abbildung 5.2 ersichtlich, besteht die Architektur aus drei Schichten, nämlich der Netzwerk-, der Kommunikations- und der Applikationsschicht.

Die Netzwerkschicht beinhaltet einerseits den sogenannten Tip Selection Algorithmus, konkret Uniform Random Tip Selection (URTS). Da der Tangle als DAG im Gegensatz zu Blockchain-basierten DLT nicht linear aufgebaut ist, müssen bei einer neuen Nachricht zunächst zwei bestehende, noch unbestätigte Nachrichten bestätigt werden. Der URTS Algorithmus ist für die Wahl von genau zwei unbestätigten Nachrichten (Tips) aus allen bestehenden unbestätigten Nachrichten zuständig. Des Weiteren beinhaltet IOTA 1.5 noch einen Konsensusalgorithmus, welcher allerdings nicht für die Bestätigung von Nachrichten selbst, sondern zur Verhinderung von Spam-Attacken eingesetzt wird. In diesem Fall wird ein PoW Algorithmus ausgeführt. Die Berechnung des PoW erzeugt für das sendende Endgerät einen zusätzlichen Rechenaufwand, welcher die Anzahl der Nachrichten, die in kurzer Zeit gesendet werden können, begrenzt und somit die Gefahr von Spam-Attacken reduziert. Mit dem Parameter der PoW-Schwierigkeit kann die Schwierigkeit des zu lösenden kryptographischen Rätsels beeinflusst werden. In der Evaluation werden unterschiedliche Schwierigkeitsgrade gewählt. Weiterhin gehört die für die Permanode-Funktionalität erforderliche verteilte Datenbank SkyllaDB zu der Netzwerkschicht.

In der Kommunikationsschicht sind die Nachrichten selbst sowie die Netzwerkknoten enthalten. Die sogenannten Hornet-Knoten sind die Basisknoten, aus welchen das Netzwerk aufgebaut ist. Die Validierung von Nachrichten geschieht bei IOTA 1.5 noch nicht durch Konsensusalgorithmen, sondern durch die zentrale Instanz, den Koordinator, welche das zusätzliche Koordinator Plug-In erfordert. Für den Anwendungsfall wird, wie oben beschrieben, weiterhin ein Chronicle-Knoten eingesetzt, der als Schnittstelle zur verteilten SkyllaDB Datenbank dient. Durch ein weiteres Plug-In, dem Spammer, wird für die Evaluation eine künstliche Netzwerklast in Form von MPS geschaffen.

Die Applikationsschicht besteht aus dem Client-SDK (engl. Software Developer Kit (SDK)), welches für die Interaktion von Software mit dem Tangle benötigt wird. Weiterer Bestandteil ist die Firefly Wallet, deren Aufgabe das sichere Aufbewahren von MIOTA Token, der IOTA Währung, ist. Alle weiteren dezentralen Applikationen (engl. Decentralized Applications (dApps)), welche mit dem Tangle interagieren, beispielsweise Dashboards, werden ebenfalls der Applikationsschicht zugeordnet.

5.4 Zugriffsrechteverwaltung mittels Smart Contracts

5.4.1 Beschreibung des Anwendungsfalls

Die Zugriffsrechte für das Schreiben und Lesen von Daten werden exemplarisch durch ein Lizenzmanagement automatisiert und ohne weitere menschliche oder sonstige

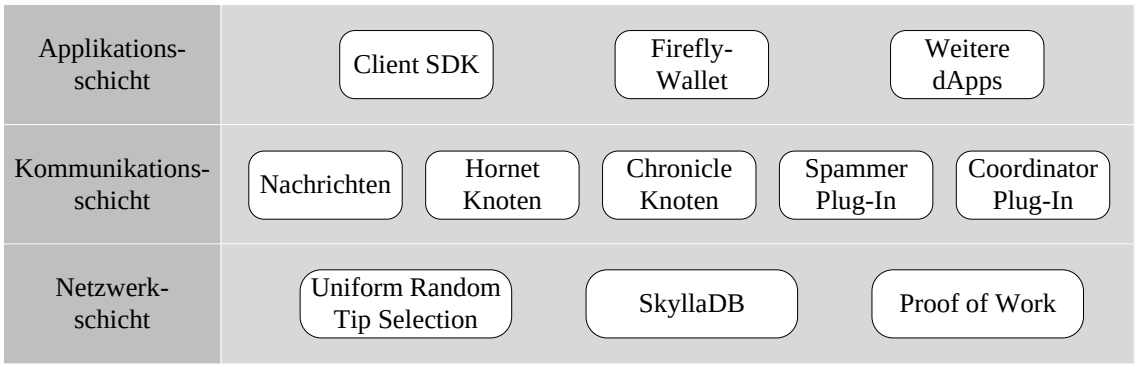


Abbildung 5.2: Drei-Schichten-Modell des aufgebauten IOTA Chrysalis Netzwerks (Rosenberger, Rauterberg und Schramm 2021)

durch zentrale Drittpartei durchgeführte Überprüfung vergeben. Im industriellen Kontext ist dies ein weiterer Schritt, um digitale Geschäftsmodelle zu ermöglichen. Im Fall der Datenflussoptimierung betrifft dies beispielsweise die Leserechte von Datenflüssen selbst oder den Analyseergebnissen von digitalen Services wie der bereits genannten Anomalieerkennung. Wird für einen Service gezahlt, im einfachsten Fall direkt mittels MIOTA, wird diese Transaktion im Tangle festgehalten. Ein SC prüft die Bedingung, nämlich den Zahlungseingang, und gewährt bei Erfüllung die Leserechte oder stellt eine Lizenz zur Verfügung. Die Ausführung wird ebenfalls im Tangle als Nachricht erfasst und ist somit nachweisbar, transparent und unveränderlich. Ist die Bedingung nicht mehr erfüllt, beispielsweise wenn der Zeitraum, für den die Gebühr bezahlt wurde, endet, werden die Rechte wieder entzogen.

Der zweite Anwendungsfall beschreibt eine Verwaltung von Zugriffsrechten auf Daten und Informationen durch die automatisierte Code-Ausführung in Form von SC in einem IOTA 2.0 Netzwerk auf industriellen Endgeräten.

5.4.2 Beschreibung der Architektur

Analog zu der Netzwerkarchitektur für Anwendungsfall 1, ist diese für das IOTA 2.0 Netzwerk dreischichtig. In Abbildung 5.3 wird eine Übersicht über die konkret eingesetzten Bestandteile für Anwendungsfall 2 gegeben.

In der Netzwerkschicht kommt auch in diesem Fall ein Tip Selection-Algorithmus zum Einsatz. Dieser ist eine weiterentwickelte Version, nämlich Restricted Uniform Random Tip Selection (RURTS).

Da IOTA 2.0 vollständig dezentral ist und es keinen Koordinator mehr gibt, der Nachrichten validiert, ist ein neuer Konsensus-Algorithmus, der FPC Algorithmus in Kombination mit dem Distributed Random Number Generator (DRNG), eingeführt.

Die Kommunikationsschicht beinhaltet auch hier alle Nachrichten sowie die unterschiedlichen Netzwerkknoten. Da die SC-Funktionalität nur in IOTA 2.0 verfügbar ist, kommen GoShimmer-Knoten anstelle der Hornet-Knoten als Basisknoten zum

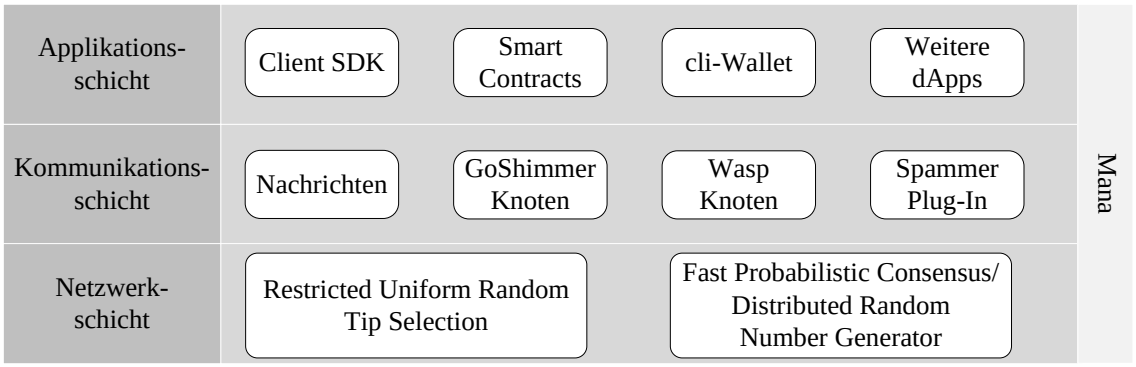


Abbildung 5.3: Drei-Schichten-Modell des aufgebauten IOTA Coordicide Netzwerks (Rosenberger, Rauterberg und Schramm 2021)

Einsatz. Für die SC-Funktionalität wird ein überlagertes Netzwerk (off-chain) aus Wasp-Knoten gebildet wie es in Abbildung 5.4 dargestellt ist.

Die Applikationsschicht beinhaltet wie bei IOTA 1.5 das Client SDK, eine Wallet, in diesem Fall die sogenannte cli-Wallet, die SCs sowie weitere dApps.

Über alle Schichten hinweg spielt Mana eine wichtige Rolle. Mana ist ein Reputationstoken, also ein Mittel, um die Vertrauenswürdigkeit von Knoten zu beziffern. Je mehr Mana ein Knoten besitzt, desto vertrauenswürdiger wird er eingeschätzt.

Die Zugriffsrechte werden in dieser Implementation über einen SC-basiertes Lizenzmanagement verwaltet. Die Verhaltensweise des Lizenzmanagements, d.h. der Inhalt des SC, ist in Pseudocode 2 beschrieben.

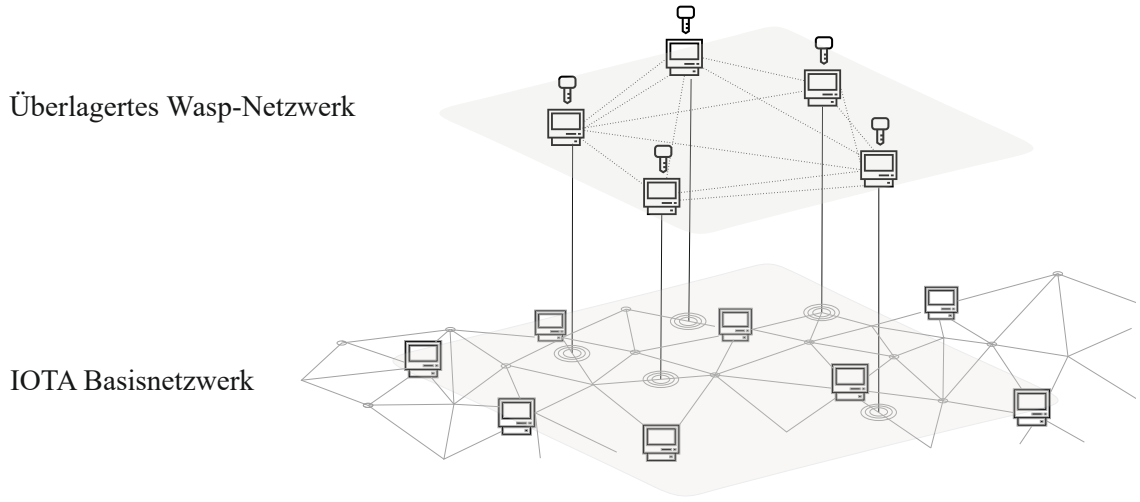


Abbildung 5.4: Darstellung des überlagerten Wasp-Netzwerks für SC

Algorithmus 2 Smart Contract für Zugriffsrechteverwaltung (nach Rosenberger, Rauterberg und Schramm 2021)

```
1: def function buyLicence():
2: mengeIOTA = ScContext.getIncomingIota()
3: if mengeIOTA > 0 then
4:     Lizenz = generateLicence()
5:     Dauer = Menge MIOTA Token / Preis pro Sekunde
6:     Ablaufdatum = Zeitstempel + Dauer
7: else
8:     Fehlerausgabe "Keine Übertragung von MIOTA Token"
9: end if
10: print(licence)
11: storeLicence(aktueller Zeitstempel, callerID, Lizenz, Ablaufdatum)
12: end function
13:
14: def function getLicences():
15: Lizenznummer = ScContext.getLicenceNumber()
16: Zeitstempel, callerID, Lizenz, Ablaufdatum = getLicence(Lizenznummer)
17: return Zeitstempel, callerID, Lizenz, Ablaufdatum
18: end function
```

5.5 Evaluation

Die Experimente werden genauso wie die in Kapitel 4 vorgestellten weiteren Algorithmen der Logik-Ebene auf repräsentativen industriellen Endgeräten durchgeführt. Für den Aufbau des IOTA Netzwerks werden virtualisierte Steuereinheiten ctrlX CORE virtual der Firma Bosch Rexroth AG mit einer 64 bit Quad Core AMD CPU, 4 GB RAM und 4 GB eMMC Speicher eingesetzt. Das Betriebssystem ist in beiden Fällen Linux Ubuntu Core 18, welches die Applikationen im Snap-Format erwartet. Der direkte Einsatz von bereits verfügbaren IOTA-Knoten als Docker-Container wäre möglich, wenn man die Docker-Container wiederum in einem geeigneten Snap betreibt. Aufgrund des nicht abschätzbaren zusätzlichen Ressourcenverbrauchs und um die Ergebnisse nicht zu verfälschen, wird sich gegen die Docker-Lösung entschieden und die Knoten stattdessen direkt als Snap betrieben. Dafür werden für die Hornet-, GoShimmer- und Wasp-Knoten die von IOTA verfügbaren vorkompilierten Binaries genutzt. Nach offizieller Dokumentation (Foundation 2022) bedarf ein Endgerät für die Ausführung des Chronicle-Knoten mindestens 32 GB Festplattenspeicher und 4 GB RAM. Durch das Kompilieren der Binaries und das nachträgliche Entfernen all der Dateien, die nur für die Erstellung des Snaps erforderlich waren, kann der benötigte Speicher auf unter 30 MB reduziert werden und der Einsatz auf einem industriellen Endgerät mit nur 4 GB Speicher ermöglicht werden. Für den Betrieb der SkyllaDB ist das gewählte Endgerät laut den Spezifikationen (SkyllaDB 2022) mit seinen verfügbaren Ressourcen an den unteren zulässigen Grenzen

einzuordnen. Allgemein werden Endgeräte mit mehr RAM und Festplattenspeicher empfohlen. Nichtsdestotrotz wird auch die Einsatzmöglichkeit einer Steuereinheit für den Betrieb der SkyllaDB evaluiert.

Zu Evaluationszwecken finden zwei weitere IOTA-Funktionalitäten Einsatz:
Spammer Plug-In[4] Dieses Plug-In lässt Basisknoten synthetisch Nachrichten mit vorgegebener Senderate generieren und erzeugt dadurch eine synthetische Netzwerklast. Somit können die Performanz und die Ressourcenverbräuche in Abhängigkeit zu der Netzwerklast, z.B. MPS, gemessen werden.
Faucet-Knoten [5] Der Faucet-Knoten kann zu Entwicklungszwecken Überweisungen von MIOTA Token tätigen.

Die Evaluation wird an privaten Netzwerken unterschiedlicher Größe, konkret mit 5, 10 und 20 Knoten, sowie unterschiedlichen Netzwerkauslastungen durchgeführt. Es wird der Ressourcenverbrauch für vier unterschiedliche Knotentypen, die auf der Testhardware ausgeführt werden, und ihre Abhängigkeit zu der Netzwerklast erhoben. Jedes Steuergerät betreibt dabei genau einen Knoten. Der Verbrauch des Festplattenspeicher wird nicht gemessen, da für die Evaluation die maximale Größe des Tangles durch den Pruning Parameter auf 3 GB begrenzt wird.

5.5.1 Evaluation der Ressourcenauslastung durch die Basis-Knoten

Die Ergebnisse des untersuchten Ressourcenverbrauchs durch die Basisknoten, Hornet- bzw. GoShimmer-Knoten, für die Anwendungsfälle 1 und 2 sind in den Tabellen 5.1 und 5.2 aufgeführt.

Zusätzlich wird der Ressourcenverbrauch des sendenden Endgeräts gemessen, welches sich durch die Ausführung eines Basisknotens mit Spammer Plug-In auszeichnet. Für das IOTA 1.5 Netzwerk werden zusätzlich die Werte des Hornet-Knotens mit Koordinator Plug-In aufgezeichnet (siehe Tabelle 5.3). Die Evaluation wird mit der Software Prometheus (Prometheus 2022) durchgeführt. Die folgenden Metriken werden berücksichtigt:

- *process_cpu_seconds_total*,
- *go_memstats_alloc_bytes*,
- *iota_node_messages_per_second*.

Für die Aufzeichnung der Netzwerklast in IOTA 1.5 werden zusätzlich die Metriken *tx_bytes* (gesendete Datenmenge) und *rx_bytes* (empfangene Datenmenge) der Linux Netzwerk Schnittstellen-Statistiken (*/sys/class/net/eth0/statistics/*) einmal pro Sekunde erfasst. Für die Evaluation der GoShimmer-Knoten stehen weiterhin die nachfolgend ausgewählten Metriken des Prometheus Monitoring Tools zur Verfügung:

- *process_cpu_seconds_total*,

[4] https://github.com/iotaledger/hornet/tree/if-deployment/plugins/spammer
[5] https://github.com/iotaledger/chrysalis-faucet

Tabelle 5.1: Performanz von Hornet-Knoten betreibenden Endgeräten (Rosenberger, Rauterberg und Schramm 2021)

Netzwerk-größe[a]	MPS [mxs/s]	CPU ∅ [%]	CPU max. [%]	RAM ∅ [MB]	RAM max. [MB]	Traffic[b] (ein) ∅ [kB/s]	Traffic[b] (aus) ∅ [kB/s]
5 Knoten	10	1,87	2,80	24,26	31,79	4,23	0,77
5 Knoten	30	3,47	4,40	22,08	31,29	0,09	0,08
5 Knoten	50	4,23	5,80	25,50	32,79	0,10	0,07
10 Knoten	10	3,14	15,6	24,33	31,56	0,17	0,10
10 Knoten	30	5,59	7,80	25,95	35,78	39,82	30,82
10 Knoten	50	8,07	46,84	26,58	38,01	0,11	0,90
20 Knoten	10	2,89	4,00	23,37	30,55	18,24	16,01
20 Knoten	30	4,40	6,20	25,15	31,73	17,39	8,54
20 Knoten	50	8,08	31,00	28,06	46,18	38,61	23,46

[a]Zusätzlich zu der Anzahl der Basisknoten wird je ein Hornet-Knoten mit Koordinator Plug-In bzw. mit Spammer Plug-In in dem Netzwerk betrieben. Die Ergebnisse beziehen sich auf Hornet-Knoten ohne Plug-In.

[b]Die Messung über die Linux Statistiken ist evtl. fehlerhaft und nicht aussagekräftig.

Tabelle 5.2: Performanz von GoShimmer-Knoten betreibenden Endgeräten (Rosenberger, Rauterberg und Schramm 2021)

Netzwerk-größe[a]	MPS [mxs/s]	CPU ∅ [%]	CPU max. [%]	RAM ∅ [MB]	RAM max. [MB]	Traffic (ein) [kB/s]	Traffic (aus) [kB/s]
5 Knoten	10	8,78	31,55	190,16	249,93	234,38	948,06
5 Knoten	30	24,86	81,08	190,35	329,18	249,52	962,37
5 Knoten	50	30,59	84,81	194,33	317,42	265,05	977,94
10 Knoten	10	6,45	8,13	64,22	87,86	182,15	710,93
10 Knoten	30	12,69	16,14	88,73	121,94	215,73	814,82
10 Knoten	50	24,73	64,48	147,55	221,98	169,10	881,73
20 Knoten	10	5,20	6,07	61,59	76,85	28,45	24,04
20 Knoten	30	20,72	30,14	125,21	170,90	244,18	131,50
20 Knoten	50	31,61	53,61	123,88	162,34	261,41	147,42

[a]Zusätzlich zu der Anzahl der GoShimmer-Basisknoten werden fünf Wasp-Knoten sowie ein GoShimmer-Knoten mit Spammer Plug-In in dem Netzwerk betrieben. Die Ergebnisse beziehen sich auf GoShimmer-Knoten ohne Plug-In.

- *process_mem_usage_bytes*,
- *tangle_message_total_count*,
- *traffic_gossip_inbound_bytes*,
- *traffic_gossip_outbound_bytes*.

Tabelle 5.3: Performanz von Nachrichten sendenden oder validierenden Endgeräten (Rosenberger, Rauterberg und Schramm 2021)

Knotentyp	MPS [mxs/s]	CPU ∅ [%]	CPU max. [%]	RAM ∅ [MB]	RAM max. [MB]	Traffic (ein) [kB/s]	Traffic (aus) [kB/s]
Hornet (sendend)	5	69,68	85,21	36,42	47,93	7,73	11,31
Hornet (sendend)	10	94,12	107,13	32,35	55,55	9,28	16,09
Koordinator (valid.)	10	6,24	23,65	21,15	28,12	17,02	16,58
Koordinator (valid.)	30	9,46	16,59	26,73	36,40	60,85	64,55
Koordinator (valid.)	50	14,21	24,30	26,74	35,89	83,30	90,43
GoShimmer (sendend)	10	6,97	9,00	189,74	260,02	11,97	11,97
GoShimmer (sendend)	30	23,64	33,20	110,30	188,42	82,77	82,77
GoShimmer (sendend)	50	30,15	42,60	183,59	299,70	82,79	82,79

5.5.2 Evaluation von Permanode- und Smart Contract-Funktionalität

Von besonderer Relevanz für den ersten Anwendungsfall ist der Ressourcenverbrauch des Schnittstellen-Knotens, dem Chronicle-Knoten. Dieser ist in Tabelle 5.4 in Abhängigkeit zu der Rate der Datenbankzugriffe zusammengefasst. Zusätzlich wird die Anwendbarkeit der SkyllaDB auf IIoT Endgeräten betrachtet und anhand unterschiedlicher Raten der abzuspeichernden Daten evaluiert. Die Ergebnisse sind in Tabelle 5.5 aufgeführt.

Tabelle 5.4: Performanz eines Chronicle-Knoten betreibenden Endgeräts (Rosenberger, Rauterberg und Schramm 2021)

Zugriffsrate [mxs/s]	CPU ∅ [%]	CPU max. [%]	RAM ∅ [MB]	RAM max. [MB]
10	0,85	2,30	1,71	1,90
30	2,34	5,61	1,50	1,70
50	5,13	12,02	1,72	1,90

Tabelle 5.5: Performanz eines SkyllaDB betreibenden Endgeräts (Rosenberger, Rauterberg und Schramm 2021)

MPS [mxs/s]	CPU ∅ [%]	RAM [MB]	Speicher ∅ [B/s][a]	Traffic (ein) [kB/s]	Traffic (aus) [kB/s]
10	2,14	19,51	98,10	8,66	0,76
30	4,52	17,51	320,93	29,27	2,36
50	8,11	15,69	405,60	56,71	2,78

[a]Der Speicherbedarf erhöht sich schrittweise in Abhängigkeit der zu speichernden Datenmenge. Daher sind die Ergebnisse in [B/s] in Abhängigkeit der zu speichernden Nachrichtenrate aufgeführt.

Im Gegensatz zu Anwendungsfall 1, ist für die automatisierte Zugriffsrechteverwaltung der Ressourcenverbrauch des Wasp-Knoten-Netzwerks relevant. Tabelle 5.6 zeigt den durchschnittlichen Ressourcenverbrauch eines Wasp-Knotens in unterschiedlich großen Wasp-Netzwerken bei unterschiedlicher Anzahl von SC pro Minute. Zusätzlich erfolgt eine Evaluation der Ausführung der SC selbst. Hierfür werden von (Zheng u. a. 2018) unterschiedliche Metriken vorgeschlagen, von denen in dieser Arbeit die durchschnittliche Antwortzeit *Average Response Delay ($\overline{RD}$)* als die für industrielle Applikationen aussagekräftigste Metrik gewählt wird. Sie wird entsprechend Gleichung 5.1 berechnet. Zusätzlich wird der Ressourcenverbrauch während der Ausführung des SC gemessen.

$$\overline{RD} = \frac{\sum_{i=1}^{N}(t_{mx\,confirmed} - t_{mx\,input})}{N}(s) \tag{5.1}$$

mit $t_{mx\,confirmed}$ - Zeitpunkt des bestätigten SC
$t_{mx\,input}$ - Zeitpunkt der den SC auslösenden Nachricht
N - Anzahl der bestätigten SC

Tabelle 5.6: Performanz eines Wasp-Knoten betreibenden Endgeräts (Rosenberger, Rauterberg und Schramm 2021)

Anzahl der Wasp-Knoten	SC Rate [/min]	CPU ∅ [%]	CPU max. [%]	RAM ∅ [MB]	RAM max. [MB]	$\overline{RD}$ [s]
5 Knoten	1	1,09	17,01	4,12	8,32	24,00
5 Knoten	2	1,34	18,01	4,29	8,77	24,73
5 Knoten	4	1,89	21,10	4,24	8,63	24,44
10 Knoten	1	2,03	70,01	4,98	10,07	29,86
10 Knoten	2	3,22	47,09	5,36	10,63	29,98
10 Knoten	4	4,67	49,01	5,14	10,15	35,89

5.6 Ergebnisse und Diskussion

Aus der Evaluation zeigt sich, dass die Hornet-Knoten zu einem geringen Ressourcenverbrauch führen, der nur geringfügig mit der Netzwerkgröße und Nachrichtenrate ansteigt. Dieses Ergebnis ist schlüssig, da der Koordinator für die Validierung der Nachrichten verantwortlich ist und für die Validierung somit kein zusätzlicher Ressourcenbedarf bei den Hornet-Knoten, sondern nur bei dem Koordinator entsteht. Die Messungen des Traffics der Hornet-Knoten korrelieren entgegen der Erwartung nicht mit der MPS-Rate. Für IOTA 1.5 stehen im Gegensatz zu IOTA 2.0 in Prometheus keine Metriken für die Traffic-Messungen zur Verfügung, weshalb die Linux Netzwerk Schnittstellen-Statistiken für die Traffic-Messungen in IOTA 1.5 herangezogen wurden. Möglicherweise lag hier ein Messfehler vor und die Ergebnisse sind nicht aussagekräftig.

Die GoShimmer-Knoten verhalten sich wegen der Dezentralisierung dementsprechend anders. Sie besitzen aufgrund der Abwesenheit des Koordinators einen höheren Ressourcenverbrauch, da sie die Validierung von Nachrichten mittels FPC selbst durchführen. Bei dem Versenden von Nachrichten steigt der Ressourcenverbrauch der Hornet-Knoten signifikant an, was in der Berechnung des PoW durch das sendende Endgerät zur Verhinderung von Spam-Attacken begründet ist. In dieser Evaluation ist die PoW-Schwierigkeit für das IOTA 1.5-Netzwerk auf 100 gesetzt und für das IOTA 2.0 Netzwerk nur auf 2. Die Auswirkung auf den Ressourcenverbrauch ist direkt erkennbar. Tabelle 5.3 zeigt, dass GoShimmer-Knoten dadurch die fünffache Senderate bei einem Drittel der CPU-Auslastung im Vergleich zu den Hornet-Knoten erzielen können. Die Verringerung der Schwierigkeit ist unter bestimmten Umständen akzeptabel. Diese können beispielsweise weniger sicherheitskritische Daten betreffen oder der Einsatz in privaten Netzwerken mit ausschließlich bekannten Geräten und einem dadurch verringerten Risiko von Spam-Attacken sein.

Zusammenfassend zeigt sich, dass sowohl Hornet- als auch GoShimmer-Knoten für die Ausführung auf IIoT Endgeräten mit limitierten Ressourcen geeignet sind.

In Abhängigkeit der bestehenden Auslastung durch weitere Aufgaben der Endgeräte ist die MPS-Rate in IOTA 2.0 Netzwerken allerdings begrenzt. Der Ressourcenverbrauch des Chronicle-Knotens ist wie zu erwarten sehr gering, da dieser lediglich eine Schnittstelle zwischen Tangle und der SkyllaDB darstellt.

In den Experimenten wird bewiesen, dass auch die Ausführung der SkyllaDB auf den limitierten Endgeräten möglich ist. Allerdings ist anzumerken, dass in Abhängigkeit der langfristig abzuspeichernden Datenmenge, der Festplattenspeicher zu erweitern ist. Dies ist je nach Endgerät möglich, beispielsweise bei der eingesetzten ctrlX CORE durch eine microSD Karte. Es ist zudem hervorzuheben, dass die Einschränkung der SkyllaDB auf ausgewählte CPUs (aktuell Intel atom und Intel core, AMD low power und AMD standard) die Anzahl der geeigneten Endgeräte begrenzt. In dem experimentellen Aufbau dieser Arbeit ist somit nur die ctrlX CORE virtual geeignet, nicht aber die reale ctrlX CORE.

Der Ressourcenverbrauch der Wasp-Knoten ist vergleichbar mit dem der GoShimmer-Knoten. Sie sind daher ebenfalls für das IIoT geeignet. Die Ausführung von SC führt zu keinem signifikanten Anstieg der CPU-Auslastung. Eine durchschnittliche Zeitspanne von $\overline{RD} \approx 24\,\mathrm{s}$ in einem Netzwerk aus fünf Wasp-Knoten sowie $\overline{RD} \approx 32\,\mathrm{s}$ in einem Netzwerk aus zehn Wasp-Knoten wurde gemessen. Der $\overline{RD}$ steigt mit steigender Wasp-Netzwerkgröße, da mehr Wasp-Knoten zu dem Quorum beitragen müssen und den SC damit sicherer machen. Die Bewertung dieser Zeitspannen ist sehr abhängig von der Anwendung. Im Rahmen der Zugriffsrechteverwaltung und digitalen Services kann sie durchaus ausreichend sein. Andere Anwendungen können allerdings einer schnelleren Ausführung bedürfen. Beispielsweise steht eine Zeitspanne von $\sim 30\,\mathrm{s}$ bis zur Ausführung eines SC nicht im Verhältnis für digitale Services mit sehr kurzen Laufzeiten (wenige Sekunden). Ein weiteres Beispiel wäre ein Service-Mitarbeiter, der Zugriffsrechte für seine Arbeit an einer Anlage benötigt und dessen Erwartungshaltung es nicht entspricht, dass er 30 s warten muss. Anpassungen in der Parametrierung, z.B. Anzahl der benötigten Wasp-Knoten für die Validierung, Größe des Quorums, etc., können zur Erhöhung der Geschwindigkeit beitragen.

Die maximale CPU-Auslastung von 70,01 % eines Wasp-Knotens in einem Netzwerk aus zehn Knoten und einer SC-Rate von 1 SC/min wird als Ausreißer betrachtet. Die anderen Verbrauchsspitzen in diesem Aufbau lagen bei 46,23 %, was den Trend der leicht ansteigenden maximalen CPU-Auslastung analog zu der geringfügig ansteigenden durchschnittlichen CPU-Auslastung in Abhängigkeit einer steigenden SC-Rate, wie auch einer steigenden Anzahl an Netzwerk-Knoten aufzeigt. Begründet ist dieser erhöhte Ressourcenverbrauch bei größeren Netzwerken durch die höhere Anzahl der Nachbarn und somit der höheren Anzahl an Peering-Partnern.

Zusammenfassend sind die folgenden Aussagen zu dem Einsatz der DLT IOTA im IIoT zu treffen:

- IOTA Chrysalis Version 1.5: Die Ausführbarkeit von privaten Netzwerken in der noch zentralisierten IOTA 1.5-Version in einem Netzwerk aus repräsentativen IIoT Endgeräten mit limitierten Ressourcen wird nachgewiesen.

- IOTA Coordicide Version 2.0: Die Ausführbarkeit von privaten Netzwerken in der vollständig dezentralen IOTA 2.0-Version in einem Netzwerk aus repräsentativen IIoT Endgeräten mit limitierten Ressourcen wird nachgewiesen. Aufgrund der Notwendigkeit, Konsensus-Algorithmen zu berechnen, begrenzen die Ressourcen der Endgeräte die erreichbaren MPS-Raten.
- Permanode-Funktionalität: Anwendungsfall 1 beschreibt den Bedarf an einer langfristigen, unveränderlichen und verteilten Ablage von Daten und Informationen in der Industrie. Da dies auf dem IOTA-Tangle selbst aufgrund von Pruning nicht möglich ist, wird die zur Verfügung gestellte Permanode-Funktionalität mit IOTA Chrysalis evaluiert. Es wird gezeigt, dass der Betrieb eines Chronicle-Knotens und der SkyllaDB auf limitierten IIoT Endgeräten möglich ist. Wichtig für den Einsatz in einer realen Anwendung ist es, eine sehr gute Netzwerkverbindung herzustellen und den Speicherplatz auf dem Endgerät an die abzuspeichernde Datenmenge in der Datenbank anzupassen, um Datenverluste zu vermeiden.
- SC-Funktionalität: Der Einsatz von SC zur Verwaltung von Zugriffsrechten, insbesondere den Lese- und Schreibrechten auf Daten und Datenflüsse, kann, wie an Anwendungsfall 2 gezeigt, die I4.0 weiter vorantreiben. Die durchgeführten Experimente konnten bestätigen, dass SCs mit IOTA Coordicide durch ein überlagertes Wasp-Netzwerk auf IIoT Endgeräten möglich sind.

Weitere Experimente mit der SkyllaDB mit unterschiedlichen Endgeräten und Ressourcen, Replikationsfaktoren und Synchronisierungsparameter sind erforderlich, um präzisere Aussagen zu der Einsatzfähigkeit der SkyllaDB im IIoT zu machen. Ein weiterer Aspekt, der in Zukunft zu berücksichtigen ist, ist die Evaluation der Permanode-Funktionalität, sobald diese für die IOTA Version 2.0 verfügbar ist.

Die Untersuchung der DLT IOTA für die Optimierung der Vertrauenswürdigkeit von Datenflüssen zeigt, dass die Absicherung von Daten durch DLT und SC auf industriellen Endgeräten möglich ist. Der sich daraus ergebende Mehraufwand ist gegen den Nutzen, exemplarisch dargestellt an den zwei Anwendungsfällen der Aufzeichnung von unveränderliche Historien sowie der Zugriffsrechteverwaltung, für die jeweiligen Daten(flüsse) abzuwägen.

KAPITEL 6

Ressourcenallokation mittels Multi-Agenten Bestärkendes Lernen

Das sechste Kapitel untersucht einen Optimierungsansatz, der auf der Middleware-Ebene eines DSPS anwendbar ist. Das Ziel ist, die vorhandenen, aber limitierten Ressourcen der IIoT Endgeräte optimal für die Bearbeitung von zusätzlichen Rechenaufträgen in Netzwerken mit dynamischen Veränderungen zu nutzen. Für die optimierte Zuweisung von Rechen- bzw. Kommunikationsressourcen werden zwei auf RL basierende, interagierende Agentensysteme vorgestellt.

6.1 Wahl des Verfahrens

Die begrenzten Ressourcen der industriellen Endgeräte eines IIoT Netzwerks selbst, konkret Rechenkapazität, Hardware- und Arbeitsspeicher, und in bestimmten Fällen, z.B. in WSN, auch Energie, sowie die begrenzte Bandbreite zwischen den Teilnehmern stellen die größten Einschränkungen hinsichtlich der Datenverarbeitung auf den Endgeräten sowie der Datenübertragung dar. Aus diesem Grund ist die optimale Ausnutzung dieser limitierten Größen eine notwendige Voraussetzung für den Einsatz der in den zwei vorangegangenen Kapiteln beschriebenen Technologien, d. h. der Anwendung der entwickelten Algorithmen der Logik-Ebene eines DSPS der fünften Generation.

Ein IIoT Netzwerk besteht aus mehreren dezentralen Teilnehmern, welche jeweils über eigene Ressourcen verfügen. Es ist davon auszugehen, dass die Teilnehmer durch ihre Kernaufgabe, z.B. Steuerungsaufgabe, bereits zu einem gewissen Grad

ausgelastet sind. Mittels intelligenter Ressourcenzuweisung können die verbleibenden Ressourcen optimal für die weiteren Aufgaben der Datenverarbeitung eingesetzt werden. Es wird ein System vorgestellt, welches die zu übermittelnden Datenflüsse intelligent, unter Beachtung der gestellten Anforderungen, auf die verfügbaren Ressourcen aufteilt. Die Inhalte sind angelehnt an dem bereits veröffentlichten Artikel (Rosenberger, Urlaub, Rauterberg u. a. 2022) sowie dem Konferenzbeitrag (Rosenberger, Bühren und Schramm 2021). Eine zusammenfassende Übersicht der Inhalte zeigt Abbildung 6.1.

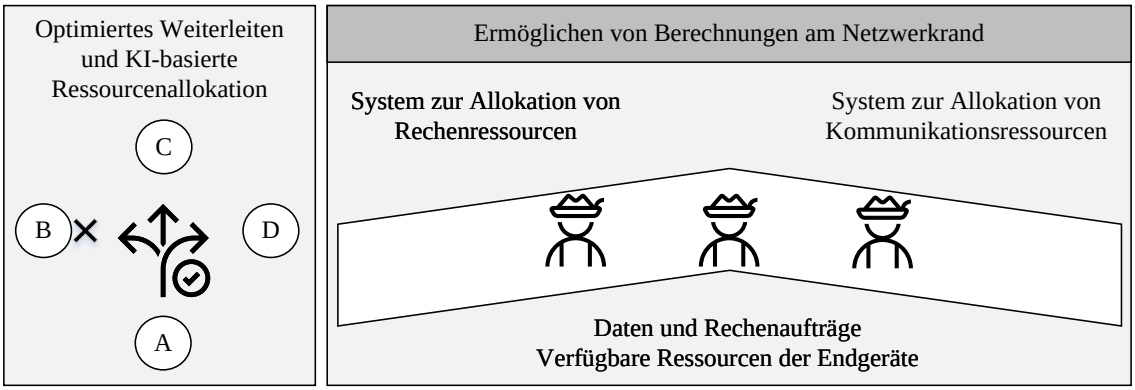


Abbildung 6.1: Schwerpunkte zur Ressourcenallokation mittels Multi-Agenten

In großen dezentralen und dynamischen Systemen, wie es das IIoT darstellt, ist die klassische Programmierung nicht mehr ausreichend. Generell ist RL eine geeignete Technik für komplexe Problemstellungen, die durch sequenzielle Entscheidungsfindung zu lösen sind. MAS sind unter anderem aufgrund ihrer Skalierbarkeit besonders für dezentrale Systeme wie IoT- und IIoT-Netzwerke geeignet. Es wird daher ein Ansatz basierend auf MARL gewählt.

Für die intelligente Ressourcenallokation besteht in dieser Arbeit die konkrete Aufgabe der Multi-Agenten darin, einerseits den verfügbaren Endgeräten die durchzuführenden Aufgaben der Datenverarbeitung zuzuweisen und andererseits den optimalen Weg zur Datenübertragung zwischen Datenquelle und Datensenke, d. h. dem ausführenden Endgerät, zu wählen. Es wird zudem in Abschnitt 6.8 ein Vergleich zwischen Single-Agenten, zentralen und dezentralen Multi-Agenten angestellt. Diese werden in Bezug auf bestimmte Kriterien bewertet.

Eine Gesamtübersicht über das vorgeschlagene MAS für die Datenflussoptimierung wird schematisch in Abbildung 6.2 gegeben, wobei der Gesamtrahmen ein exemplarisches IIoT-Netzwerk mit der Struktur eines vermaschten Netzes darstellt. Weiß hinterlegt sind die intelligenten IIoT-Endgeräte, die aus einer Recheneinheit bestehen, auf der unter anderem RL-Agenten sowie exemplarische Rechenaufträge entsprechend Kapitel 4 und 5 ausgeführt werden.

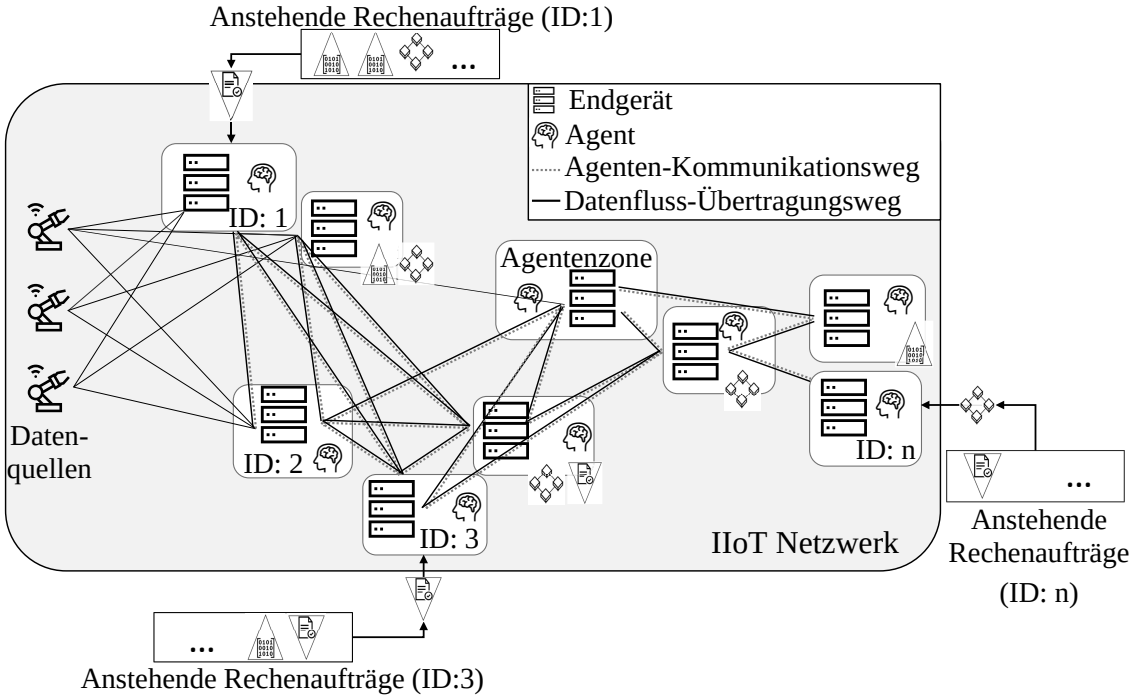


Abbildung 6.2: Schematische Architektur des vollständig dezentralen Systems (in Anlehnung an Rosenberger, Urlaub, Rauterberg u. a. 2022)

6.2 Stand der Technik

Dieser Abschnitt beschreibt den Stand der Technik auf dem Gebiet der Ressourcenallokation mittels RL im industriellen Kontext. Ressourcen können dabei einerseits physischer Art sein, beispielsweise Rohmaterialen für die Produktion, oder abstrakter Natur, beispielsweise Rechenressourcen. Ein wichtiger Einsatzbereich von RL in der Produktion ist die Planung der Maschinenauslastung (engl. Job Shop Scheduling (JSP)). Ein Ansatz zur gleichmäßigen Maschinenauslastung mittels dynamischen JSP basierend auf DRL wird von (Wang, Hu, Wang u. a. 2021) beschrieben. Zwei weitere Beispiele mit dem Fokus auf energieoptimierter Fertigung sind in (Bakakeu u. a. 2020) und (Roesch u. a. 2019) genannt. Der Schwerpunkt der Arbeit von (Luo, Zhang und Fan 2021) liegt dagegen auf den Echtzeitanforderungen, welche durch ein System aus heterogenen Multi-Agenten erreicht wird.

Mit Blick auf den Schwerpunkt dieser Dissertation, den IIoT Netzwerken, wird weiterhin auf Ansätze zum Umgang mit den begrenzten Rechenressourcen eingegangen. Einer der am weitesten verbreiteten Ansätze ist das Mobile Edge Computing, bei dem die Rechenlast auf lokale Server (Fog), d. h. zwischen Netzwerkrand (Edge) und Cloud, verlagert wird. Für die Entscheidung über die Lastverlagerung wird zunehmend der Einsatz von RL Agenten untersucht. Es gibt eine Vielzahl von Single-Agenten Ansätzen basierend auf DRL wie beispielsweise in (Chen, Liu u. a. 2021; Xiong u. a. 2020; Wang, Zhao u. a. 2019; Wang, Zhao u. a. 2021). Ein allge-

meiner Multi-Agenten Ansatz für IoT Netzwerke wird in (Liu, Yu u. a. 2020) vorgestellt. Ausarbeitungen zu MARL, die explizit für das IIoT bestimmt sind, werden von (Ren, Sun und Peng 2021) und (Cao u. a. 2020) beschrieben. Während bei der Lastverlagerung von abgeschlossenen Aufgaben, d.h. von bekanntem und endlichem Rechenaufwand und Dauer, ausgegangen wird, unterscheiden sich die Anforderungen im Kontext der Datenflussverarbeitung. Diese werden bisher nur in wenigen Arbeiten berücksichtigt. (Li, Xu u. a. 2018) stellen einen zentralisierten Ansatz vor. Der Ansatz ist nicht für Systeme dynamischer Größe geeignet, da die Anzahl der Aktionen von der Anzahl der Recheneinheiten abhängt und zu Beginn fest vorgegebenen wird. In (Russo u. a. 2018) wird ein Ansatz größerer Flexibilität, basierend auf einem MAS, vorgestellt. Das Verfahren ist jedoch modellbasiert und setzt ein Modell der Umgebung voraus.

Abgesehen von der Verwaltung von Produktionsmaschinen- und Rechenressourcen ist die Bedeutung von DRL in Kommunikation und Netzwerken hervorzuheben (Luong u. a. 2019). Dabei stellen industrielle drahtlose Netzwerke (Chen, Qiu u. a. 2021) ein relevantes Einsatzgebiet für RL Applikationen dar. Der Einsatz wird zwischen der Allokation von Netzwerkressourcen und Routing-Aufgaben unterschieden.

Die Allokation der Netzwerkressourcen, beispielsweise die Wahl des Frequenzbandes, erfolgt in der Fahrzeug-zu-Fahrzeug-Kommunikation (Ye, Li und Juang 2019) oder Gerät-zu-Gerät-Kommunikation (Li und Guo 2020) jeweils durch einen Agenten je Netzwerkteilnehmer, d.h. einem Fahrzeug bzw. Gerät. In (Gong u. a. 2021) wird ebenfalls ein MAS aufgebaut und im Rahmen von zukünftigen industriellen 6G Netzwerken das Ziel verfolgt, den Energieverbrauch und die Latenzen zu minimieren, wobei die Agenten Einfluss auf die Aufgabenplanung, Übertragungsleistung und CPU-Zyklenfrequenz nehmen können.

Das zweite Anwendungsgebiet ist das Routing, wobei der beste Übertragungsweg für Daten von einer Datenquelle zu einem vorgegebenen Ziel gesucht wird (Murudkar und Gitlin 2019). Aus Gründen der Skalierbarkeit und Flexibilität werden für das Routing häufig MAS eingesetzt, sodass der Weg mittels Multi-Hop-Routing ermittelt wird (Zhang, Liu u. a. 2021; You u. a. 2019; Ding, Yang u. a. 2020). Ein essentieller Unterschied der behandelten Problemstellung dieser Arbeit ist, dass das Ziel nicht in Vorhinein bekannt ist, sondern ein geeignetes Endgerät zur Datenflussverarbeitung zur Laufzeit zu ermitteln ist.

Nur wenige Ansätze adressieren sowohl die Optimierung der Rechenleistung als auch der Netzwerkressourcen. (Wang, Zhao u. a. 2021) präsentieren einen Single-Agenten für die Zuweisung von Rechenressourcen als auch ein Routing in Abhängigkeit der Netzwerklast, mit dem Ziel die Latenzen zu minimieren. In (Cao u. a. 2020) wird dagegen ein Ansatz für die I4.0 vorgestellt, bei dem mehrere Agenten die Entscheidungen über Aufgabenverlagerung und Kanalzuweisung treffen.

Da die kombinierte Optimierung der Allokation von sowohl Rechen- als auch Kommunikationsressourcen von besonderem Neuheitsgrad ist, wird der in dieser Arbeit vorgestellte Ansatz insbesondere gegenüber den Arbeiten von (Wang, Zhao u. a. 2021) und (Cao u. a. 2020) abgegrenzt:

- Architektur: Im Gegensatz zu der hier vorgestellten vollständig dezentralen Lösung basieren die bestehenden Ansätze auf einem Single-Agenten (Wang, Zhao u. a. 2021) oder einem zentralisierten MAS (Cao u. a. 2020).
- Dynamische Veränderungen: Die genannten zentralisierten Strukturen sind nicht für die Anpassung an dynamische Änderungen geeignet. Die Anpassung an dynamische Änderungen ist ein wesentliches Merkmal des hier vorgestellten Systems.
- Anwendungsbereich: Lediglich der in (Cao u. a. 2020) beschriebene Ansatz ist explizit für die Anwendung im industriellen Kontext entwickelt.
- Daten: Beide bestehenden Algorithmen sind nicht für die Ressourcenallocation zur Verarbeitung von Datenflüssen entwickelt.
- Zielsetzung: Während in (Wang, Zhao u. a. 2021; Cao u. a. 2020) die Latenzen im Routing und der Verarbeitung minimiert werden, steht in dieser Arbeit die Maximierung der Datenverarbeitung am Netzwerkrand unter Einsatz bestehender Ressourcen im Fokus.

Da insbesondere in drahtlosen Netzwerken die begrenzt verfügbare Energie eine große Einschränkung sowohl für die Rechen- als auch Übertragungsleistung darstellt, gibt es zunehmend Veröffentlichungen, die RL für die Energieoptimierung beschreiben (Mao, Zhang und Letaief 2016; Zhang, Liu u. a. 2021; Jin u. a. 2021; Yang, Alphones u. a. 2020).

Des Weiteren wird in einigen Fällen Wert auf eine gleichmäßige Lastverteilung gelegt. In (Wang, Hu, Lin u. a. 2021) kommt hierfür Föderales Lernen (engl. Federated Learning) zum Einsatz. Nach (Chen, Hu u. a. 2022) werden die Entscheidungen in bestehenden Systemen häufig zentralisiert getroffen, sodass die Anwendung in großen verteilten Netzwerken in ihrer Qualität und Effektivität beschränkt ist. Obwohl eine Multi-Edge Kooperation vorgeschlagen wird, basiert der Ansatz auf einem Single-Agenten. Auf MARL beruhende Ausarbeitungen sind in (Sun u. a. 2020) und (Li, Tang u. a. 2017) nachzuschlagen. Obwohl die gleichmäßige Lastverteilung in dieser Dissertation nicht explizit als Ziel definiert ist, wird sie dennoch erreicht. Unter der Annahme, dass großteils ein Mangel an Ressourcen vorliegt, werden diese maximal zulässig zugewiesen, sodass sich die Auslastung der Endgeräte dem zulässigen Maximalwert annähert und über das gesamte System ähnelt.

Zusammenfassend zeigt sich, dass keiner der existierenden Ansätze eine Ressourcenallokation der verfügbaren Ressourcen der IIoT Endgeräte selbst vorsieht. Ein besonderes Merkmal des vorgestellten Ansatzes ist die Anpassungsfähigkeit an dynamische Systemveränderungen, welche in vielen bestehenden Arbeiten eine Einschränkung darstellt. Außerdem ist kein Ansatz bekannt, in dem zwei MAS miteinander interagieren, um gemeinsam sowohl Netzwerk- als auch Rechenressourcen optimal auszulasten. Die hohe Anzahl der jüngsten Veröffentlichungen belegen die Signifikanz und Aktualität sowie die Erwartung an dieses Forschungsgebiet.

6.3 Problemdefinition und Beschreibung der Lösung

6.3.1 Beschreibung der Problemstellung

Die Möglichkeiten zur Bearbeitung von Rechenaufträgen am Netzwerkrand sind in erster Linie durch die stark limitierten Ressourcen Rechenkapazität, RAM, Festplattenspeicher und Bandbreite begrenzt. Das vorgestellte MAS soll die verfügbaren Ressourcen optimal für die Verarbeitung von Rechenaufträgen zuweisen.

$$\text{Rechenauftrag} = \begin{cases} \text{Algorithmus + Datenfluss, z.B. Datenkompression} \\ \text{Algorithmus, z.B. DLT Knoten} \end{cases}$$

Diese Aufgabenstellung ist in Abbildung 6.3 abstrahiert dargestellt.

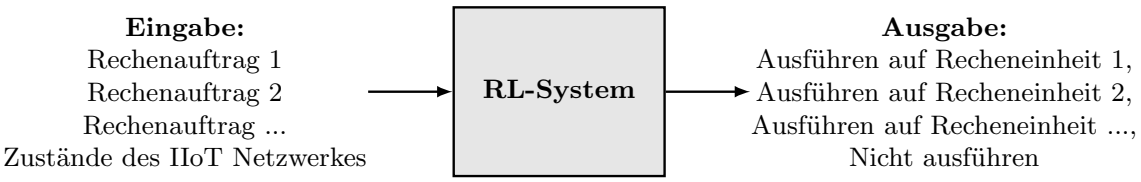


Abbildung 6.3: Abstraktion der Problemstellung (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Die folgenden Anforderungen sind von dem System zu erfüllen:

- A1: Das System soll sich adaptiv an dynamische Änderungen im Netzwerk anpassen.
- A2: Datenverlust durch Überlastung soll vermieden werden.
- A3: Die Ressourcenallokation soll keine zusätzliche Hardware bedürfen.
- A4: Die Anzahl und der Vielfalt der Rechenaufträge darf variieren.
- A5: Das System soll agnostisch für vers. Sensorsignale und Rechenaufträge sein.
- A6: Das System soll für die Verarbeitung von Datenflüssen geeignet sein.
- A7: Der manuelle Parametrierungsaufwand soll gering sein.

6.3.2 Beschreibung der Problemlösung

Der vorgeschlagene Lösungsansatz unterteilt sich in zwei Bestandteile. Das erste System, im Folgenden *MAS1* benannt, ist verantwortlich für die Allokation von Ressourcen der Endgeräte, beispielsweise der Rechenkapazität. Das zweite System, im Folgenden *MAS2* benannt, ist verantwortlich für die Allokation von Kommunikationsressourcen im Netzwerk, in diesem Fall der Bandbreite.

Beide Systeme sind vollständig dezentrale MAS. Da die Agenten ihre Entscheidungen sequentiell treffen, sind MAS1 und MAS2 als AEC-Spiele zu klassifizieren. Innerhalb des jeweiligen MAS sind alle Agenten kooperativ und homogen, besitzen

dieselbe Belohnungsfunktion R und denselben Beobachtungsraum S und sind somit austauschbar, sodass sie den MMDP zugeordnet werden.

Es folgt eine detaillierte Beschreibung, wie der Lösungsansatz die genannten Anforderungen im Einzelnen erfüllt. Das vollständig dezentrale System ist durch seine Struktur fähig, sich adaptiv an Änderungen in der Netzwerktopologie, sowohl in der Anzahl der Teilnehmer als auch ihrer Verknüpfungen, anzupassen (A1). Die Schwierigkeit dabei stellt die Strategie der Agenten dar, die als Neuronale Netze abgebildet ist. Die Anzahl der Ein- und Ausgabewerte eines Neuronalen Netzes darf sich zur Laufzeit nicht ändern. Bestehende Lösungen, die dieses Problem betrachten, sind unter anderem DeepSets (Zaheer u. a. 2017), Pointer Netzwerke (Vinyals, Fortunato und Jaitly 2015) und das Sequenz-zu-Sequenz-Rahmenwerk (Vinyals, Bengio und Kudlur 2016). In dieser Arbeit wird ein anderer Ansatz von geringerer Komplexität verfolgt. Die Problemstellung, wie sie in Abbildung 6.3 dargestellt ist, wird in kleinstmögliche Einheiten unterteilt, sodass die Anzahl der Ein- und Ausgabewerte je Agent je Wahl einer Aktion statisch ist. Zur Anpassung an Veränderungen im Netzwerk wird die Anzahl der kleinsten Einheiten erhöht oder verringert. Das bedeutet konkret, dass jede Recheneinheit durch jeweils einen Agenten von MAS1 und MAS2 verwaltet wird. Ein Agent entscheidet nicht über alle Rechenaufträge gleichzeitig, sondern nacheinander. Somit ist die Anzahl der Rechenaufträge variabel gestaltbar (A4).

Das Training der Agentensysteme erfolgt mit variierenden Rechenaufträgen, d.h. variierende Lasten durch die Ausführung der Berechnungen sowie variierende Senderaten der Datenflüsse, aber auch variierenden Grundauslastungen der Recheneinheiten selbst. Somit sind die Agentensysteme generisch einsetzbar und nicht auf konkrete Anwendungsfälle trainiert (A5, A7). Die Systeme erhalten die notwendigen Informationen über ihre Umwelt durch Abfragen, die sie selbstständig tätigen. Weitere Parametrierungen sind daher nicht notwendig (A7).

Die Agentensysteme sind leichtgewichtig und für den Einsatz auf Endgeräten entwickelt, sodass es für die Ausführung der Agenten keiner zusätzlichen Hardware bedarf (A3). Die Agenten sind als TensorFlow Lite-Modelle exportiert, welche insbesondere für den Einsatz auf ressourcenarmen Endgeräten entwickelt wurden.

Dem System werden keine Informationen über die Gesamtlast des Rechenauftrags (Rechenaufwand, Datengröße) übergeben. Es wird angenommen, dass die zu verarbeitenden Datenflüsse unendlich sind (A6).

Ziel der Agenten ist es, die zusätzlichen Rechenaufträge mit den Ressourcen der Endgeräte auszuführen. Dies hat eine Steigerung der Auslastung der verfügbaren Ressourcen zur Folge. Um eine Überlastung zu vermeiden, werden die Agenten darauf trainiert, die Auslastung unterhalb eines vorgegebenen Schwellwerts zu halten (A2). Zusätzlich wird die Priorität der Rechenaufträge auf einen geringeren Wert gesetzt als die Hauptaufgaben der Recheneinheit. Somit besteht im Falle einer Überlastung kein Risiko für den Anlagenbetrieb.

Des Weiteren besitzen zunächst alle Endgeräte die erforderlichen Berechtigungen die Daten zu lesen und zu verarbeiten, sodass eine Abfrage über diese Berechtigungen entfällt.

6.4 System zur Allokation von Rechenressourcen

Die Agenten des MAS1 verfolgen das Ziel, die Rechenressourcen ihres eigenen Endgeräts optimal zu verwalten und die Auslastung unter einem definierten Schwellwert zu halten, um die Bearbeitung einer möglichst großen Anzahl zusätzlicher Rechenaufträge zu ermöglichen. Der Agent kann lediglich die Verarbeitung von Rechenaufträgen starten, jedoch nicht wieder stoppen, sodass die Datenverarbeitung auf den Endgeräten indirekt maximiert wird. Auch die gleichmäßige Lastverteilung wird angestrebt, aber nur indirekt erzielt. Unter der Annahme, dass die Ressourcen im IIoT immer limitiert sind und maximal ausgeschöpft werden, ist zu erwarten, dass sich der Ressourcenverbrauch über alle Endgeräte an den vorgegebenen Schwellwert annähert.

Der folgende Abschnitt definiert detailliert den MMDP für MAS1, gefolgt von der Beschreibung des Trainingsprozesses von MAS1 in Algorithmus 3 mit den in Tabelle 6.1 aufgeführten Trainingsparametern.

Tabelle 6.1: Trainingsparameter für MAS1 und MAS2.

Parameter	**MAS1**	**MAS2**
Gesamtzahl der Trainingsschritte	1 000 000	1 000 000
Trainingsschritte je Episode	100	100
Lernrate α	0,003	0,003
Diskontierungsfaktor γ	0,99	0,99
RL Algorithmus	PPO	PPO
Anzahl der Agenten	3	3

- **Umgebung**: Der Rechenauftrag soll auf einem der unterschiedlichen Endgeräte d_i im IIoT ausgeführt werden, weshalb das IIoT Netzwerk die Umgebung der Agenten darstellt.
- **Agentenraum**: In Abhängigkeit von der Netzwerkgröße, d.h. von der Anzahl D der möglichen IIoT Endgeräte, variiert die Anzahl der Agenten des MAS1, da gilt $N^{MAS1} = D$. Für MAS gilt: $N = \{1, 2, ..., n\}$ und $n > 1$.
- **Zustandsraum**: Der Beobachtungsarray der einzelnen Agenten besteht aus zwei Zuständen $s = \{s^1, s^2\}$ mit $0 < s^1, s^2 \leqslant 1$ und $s \in S$. s^1 ist die beobachtete CPU-Auslastung des Endgeräts d_i und s^2 die durchschnittliche CPU-Auslastung aller benachbarter Endgeräte. Es gilt:

$$s^2 = \frac{1}{n} \sum_{i=0}^{n} s_1^{1,i} \tag{6.1}$$

- **Aktionsraum**: Der Aktionsraum besteht aus zwei diskreten Aktionen: Ausführung des Rechenauftrags $a = 1$ oder keine Ausführung $a = 0$. Für die Aktionen $a \in A$ gilt $A = \{0, 1\}$.

- **Belohnungsfunktion**: Alle Agenten sind homogen und verfügen über die selbe Belohnungsfunktion.

$$R(s,a) = R^1(s,a) = R^2(s,a) = ... = R^n(s,a) \tag{6.2}$$

Die Überschreitung des vorgegebenen Schwellwerts, aber auch Inaktivität werden bestraft. Die Zuweisung von Rechenaufträgen unterhalb des Schwellwerts wird dagegen belohnt.

$$R^{MAS1}(s,a) = \begin{cases} \text{Bestrafung } r = -10 & \text{wenn } a = 1 \text{ und } s_{t+1} > \text{ Schwellwert} \\ \text{Bestrafung } r = -10 & \text{wenn } a = 0 \text{ und } s_{t+1} < \text{ Schwellwert} \\ \text{Belohnung } r = 1 & \text{sonst} \end{cases} \tag{6.3}$$

6.5 System zur Allokation von Kommunikationsressourcen

Der zweite Bestandteil der Gesamtlösung ist das mit MAS1 interagierende System MAS2 für das intelligente Weiterleiten von Rechenaufträgen in Abhängigkeit der verfügbaren Bandbreite. Es ersetzt die Nächste-Agenten-Funktion v des AEC Spiels durch einen intelligenten Mechanismus, der ebenfalls auf DRL basiert, mit dem Ziel Datenverluste aufgrund von Bandbreitenengpässen zu verhindern. Es folgen die Beschreibungen des MMDP für die MAS2-Agenten sowie des Trainingsprozesses. Die Trainingsparameter sind bereits in Tabelle 6.1 aufgeführt.

- **Umgebung**: Auch hier ist die Umgebung das IIoT.
- **Agentenraum**: Analog zu MAS1 ist die Anzahl der Agenten identisch zu der Anzahl verfügbarer Recheneinheiten im IIoT Netzwerk $N^{MAS2} = D$.
- **Zustandsraum**: Die Beobachtung eines jeden Agenten umfasst folgende Zustände:
  - s^1: maximal verfügbare Bandbreite zwischen dem Agenten-Endgerät und dem betrachteten Nachbarn j
  - s^2: aktuell genutzte Bandbreite zwischen dem Agenten-Endgerät und dem betrachteten Nachbarn j
  - s^3: durchschnittliche aktuell genutzte Bandbreite zu allen k Nachbarn des Agenten
  - s^4: erwarteter Bandbreitenverbrauch des zuzuweisenden Rechenauftrags
  - s^5: aktuelle CPU-Auslastung des betrachteten Nachbarn j
  - s^6: aktuelle RAM-Auslastung des betrachteten Nachbarn j
  - s^7: erwartete CPU-Auslastung des zuzuweisenden Rechenauftrags

Algorithmus 3 Ablauf des Trainings für MAS1 (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Eingabe Warteschlange $W = w_1, w_2, ..., w_n, ...$

while Trainingsschritte $t \neq 0$ **do**
  Agenten $1, 2, ..., n$ überwachen die Zustände $s_1, s_2, ...s_n$
  for Alle Agenten $1, ..., n$ **do**
    Übergebe den Zustand s_t in das neuronale Netz als Eingabe
    Erhalte Aktion a_t als Ausgabe
    $t = t - 1$
    $i = i + 1$
  end for
  Übergebe die gewählten Aktionen aller Agenten zur `step`-Funktion
  for Alle Agenten $1, ..., n$ **do** ▷ Im Training hat jeder Agent exakt einen Nachbarn
    if Aktion $a_t = 1$ **then**
      for Addiere Rechenlast zu CPU-Auslastung des Endgeräts i (s^1_{t+1}) **do**
        if $s^1_{t+1} >$ threshold **oder** Länge(Q) < N **then**
          Setze `done-flag`
        end if
        Neuberechnung der durchschnittlichen CPU-Auslastung (s^2_{t+1})
      end for
    else if Aktion $a_t = 0$ **then**
      Erweitere Warteschlange W um verworfenen Rechenauftrag w_i
    end if
    Belohne den Agent i
    Abfrage `done-flag`
    Agent überwacht den neuen Zustand s_{t+1}
    $i = i + 1$
  end for
  `step`-Funktion übergibt `done`, Beobachtungen s_{t+1} und Belohnungen r zum Trainingsalgorithmus
  Training des Neuronalen Netz
  if done-flag **then**
    Ende des Trainings
  end if
end while

– s^8: erwartete RAM-Auslastung des zuzuweisenden Rechenauftrags

$s = \{s^1, s^2, ..., s^8\}$ mit $s^1, s^2, s^3, s^4 \in \mathbb{R}^+$ und $0 < s^5, s^6, s^7, s^8 \leqslant 1$ und $s \in S$

$$\text{und } s^3 = \frac{1}{k} \sum_{j=0}^{k} s^{1,j} \tag{6.4}$$

- **Aktionsraum**: Übermittlung des Rechenauftrags an Nachbarn j ($a = 1$) oder keine Übermittlung, sondern Entscheidung über den nächsten Nachbarn $j + 1$ ($a = 0$). Für Aktion $a \in A$ gilt $A = \{0, 1\}$.
- **Belohnung**: Alle Agenten besitzen die gleiche Belohnungsfunktion.

$$R(s, a) = R^1(s, a) = R^2(s, a)... = R^n(s, a) \tag{6.5}$$

Ähnlich zu der bereits beschriebenen Belohnung in MAS1 wird die Überschreitung des Schwellwerts oder Inaktivität durch die gewählten Aktionen bestraft. Die Allokation von Ressourcen für die Weiterleitung von Rechenaufträgen ohne Schwellwertüberschreitung wird belohnt. Die Belohnung setzt sich aus drei Bestandteilen zusammen.

$$R^{MAS1}(s, a) = R_1^{MAS2}(s, a) + R_2^{MAS2}(s, a) + R_3^{MAS2}(s, a) \quad \text{mit} \tag{6.6}$$

$$R_1^{MAS2}(s, a) = \begin{cases} \text{Belohnung } r\text{= Bandbreitenverbrauch [\%]} & \text{wenn } s_{t+1} < \text{Schwellwert} \\ \text{Belohnung } r\text{= Schwellwert} & \text{sonst} \end{cases}$$

$R_2^{MAS2}(s, a) = (1 - s_{t+1}^5) \cdot 50$ mit s_{t+1}^5 - CPU-Auslastung des Nachbarn j bei $t + 1$

$R_3^{MAS2}(s, a) = (1 - s_{t+1}^6) \cdot 50$ mit s_{t+1}^6 - RAM Auslastung des Nachbars j bei $t + 1$

6.6 Interaktion der Systeme

Die Interaktion und Ausführung der beiden unabhängig voneinander trainierten Agentensysteme MAS1 und MAS2 ist in Algorithmus 5 zusammengefasst.

Durch das gewählte Setup, in dem die Trainingsprozesse der beiden Agentensysteme getrennt sind, ist die Vergabe einer globalen Belohnung nicht möglich. In dem Fall, dass beide Systeme bereits gemeinsam trainiert werden, ist zu erwarten, dass eine zusätzliche globale Belohnung das Lernverhalten optimiert, da sie den Agenten Feedback über die Erreichung des Gesamtziels gibt. Eine globale Belohnung könnte somit die Interaktion der Systeme verbessern, weshalb ihre mögliche Gestaltung kurz skizziert wird. Eine mögliche Belohnungsfunktion ist in Gleichung 6.7 definiert. Sie basiert auf der Belohnung erfolgreich ausgeführter Rechenaufträge und die Anzahl der benötigten Hops h zwischen dem ersten und dem ausführenden Endgerät. Die Belohnung wird dementsprechend erst am Ende einer Spielrunde vergeben. In dieser Arbeit ist eine Spielrunde definiert von ihrem Beginn mit der ersten Entscheidung

Algorithmus 4 Ablauf des Trainings von MAS2 (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Eingabe Warteschlange $W = w_1, w_2, ..., w_n, ...$

while Trainingschritte $t \neq 0$ **do**
  Agenten $1, 2, ..., n$ beobachten Zustände $s_1, s_2, ...s_n$
  for Alle Agenten $1, ..., n$ **do**
    Übergebe den Zustand s_t in Neuronales Netz als Eingabe
    Erhalte Aktion a_t als Ausgabe
    Übergebe die gewählte Aktion zur `step`-Funktion des Agenten
    $t = t - 1$
    $i = i + 1$
  end for
  for Alle Agenten $1, ..., n$ **do**
    if Aktion $a_t = 1$ **then**
      Addiere Aufgabenlast zu der Bandbreitenauslastung der entsprechenden Verknüpfung (s^1_{t+1})
      if $s^1_{t+1} >$ max. Bandbreite **or** Länge(Q) < N **then**
        Setze `done-flag`
      end if
      Neuberechnung der durchschnittlichen Bandbreitenauslastung aller Verknüpfungen in der Umgebung (s^2_{t+1})
    else if Aktion $a_t = 0$ **then**
      Erweitere Warteschlange W um verworfenen Rechenauftrag w_i
    end if
    Belohne den Agenten i
    Abfrage der `done-flags`
    Agent überwacht den neuen Zustand s_{t+1}
    i=i+1
  end for
  `step`-Funktion übergibt `done`, Beobachtungen s_{t+1} und Belohnungen r an den Trainingsalgorithmus
  Training des neuronalen Netzes
  if done-flag **then**
    Ende des Trainings
  end if
end while

Algorithmus 5 Interaktion von MAS1 und MAS2 (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Nächster anstehender Rechenauftrag w_i in Warteschlange W_i für Endgerät d_i
Abfrage der Zustände des Endgeräts und der Nachbarn
if Agent i^{MAS1} entscheidet für Ausführung ($a = 1$) **then**
    Agent triggert die Ausführung auf dem Endgerät
    Die Route des Rechenauftrags wird erweitert um Agent i
    Anhängen der Metadaten des Rechenauftrags an lokale Liste
    done
else if Agent i^{MAS1} entscheidet gegen die Ausführung ($a = 0$) **then**
    Übergibt Rechenauftrag an den Agenten i^{MAS2}
    for Nachbarn 1...k des Endgeräts i **do**
        if Agent i^{MAS2} entscheidet für Weiterleiten an den Nachbarn j **then**
            Füge Rechenauftrag w_i zu Warteschlange W_j für Endgerät d_j hinzu
            Erweitere die Route des Rechenauftrags um j
            $i = j$
            go to *top*
        else if Agent i^{MAS2} entscheidet gegen das Weiterleiten an Nachbarn j **then**
            if Nachbar $j == k$ **then**
                Verwerfen des Rechenauftrags und Entfernen aus Warteschlange
                Zurücksenden zu initialem Endgerät d_i entsprechend Metadaten
                Erweitere Ende der Warteschlange um Rechenauftrag
                done
            end if
            $j = j + 1$
        end if
    end for
end if

über einen neuen anstehenden Rechenauftrag von einem Agenten 1^{MAS1}, also von dem Routenbeginn, bis zur Entscheidung $a^{MAS1} = 1$ über die Ausführung des Rechenauftrags durch Agent i^{MAS1} oder der finalen Entscheidung den Rechenauftrag zu verwerfen $a^{MAS2} = 0$ durch einen Agenten i^{MAS2}, d.h. den Rechenauftrag an keinen der Nachbarn weiterzuleiten. Für die Einführung dieser beschriebenen Belohnung ist für die MAS1-Agenten die zusätzliche Beobachtung der Hops erforderlich, da diese sonst nicht aus der Belohnung lernen können. Im Gegensatz dazu ist die Erweiterung des Beobachtungsraums der MAS2-Agenten um die Anzahl der bisher zurückgelegten Hops aus zwei Gründen nicht zielführend. Zum einen haben die MAS2-Agenten keinen Einfluss auf die Entscheidung, ob ein Rechenauftrag weitergeleitet werden soll, da über die Notwendigkeit des Weiterleitens des jeweiligen Rechenauftrags ein MAS1-Agent entscheidet. Zum anderen sollen die MAS2-Agenten nicht dazu ermutigt werden, einen Rechenauftrag gar nicht weiterzuleiten, da dies dem Gedanken der Belohnungsfunktion 6.5 widerspricht.

$$R^{glob}(s,a) = \begin{cases} \text{Belohnung } r_1^{glob} = 80 \cdot 0.9^h & \text{Ausführen des Rechenauftrags} \\ \text{Bestrafung } r_2^{glob} = -80 \cdot 1.1^h & \text{Endgültiges Verwerfen des Rechenauftrags} \end{cases} \tag{6.7}$$

Voraussetzung, um den Verlust von Rechenaufträgen zu verhindern, ist die ausschließliche Weiterleitung an Endgeräte, deren Ressourcen ebenfalls durch Agenten überwacht werden. Zusätzlich ist es relevant, die Berechtigungen eines Endgeräts für das Empfangen, Lesen und Verarbeiten von Daten zu prüfen (siehe Anwendungsfall 2 in Abschnitt 5.4). Für beides stellt die AAS eine Lösung dar. Dafür müssen die relevanten Informationen in der AAS der I4.0 Komponente hinterlegt sein, über die sie im IIoT ansprechbar sind.

Das Flussdiagramm 6.4 beschreibt die Interaktion der beiden kooperierenden Systeme auf einem Endgerät.

6.7 Limitierungen der vorgestellten Methode

Die nachfolgenden Limitierungen sind bei dem aktuellen Ausarbeitungsgrad der vorgestellten Methode zu berücksichtigen.

- Ressourcen: Bisher sind nur die CPU- und Bandbreiten-Auslastung berücksichtigt. Weitere Ressourcen wie RAM und Festplattenspeicher sowie Energieverbrauch dagegen nicht. Die Agenten sind nicht für den Betrieb auf Endgeräten mit limitierter Energie, z.B. bei Batteriebetrieb in WSN, optimiert.
- Priorität: Alle Rechenaufträge haben die selbe Priorität. Die Agenten unterscheiden aktuell nicht zwischen unterschiedlich priorisierten Rechenaufträgen.
- Berechtigungen: Alle Netzwerkteilnehmer sind berechtigt die Rechenaufträge auszuführen, Daten zu lesen und zu verarbeiten.
- Hyperparameter-Optimierung: Die Hyperparameter sind manuell gewählt. Bisher sind keine Methoden zur Optimierung von Hyperparametern angewandt.

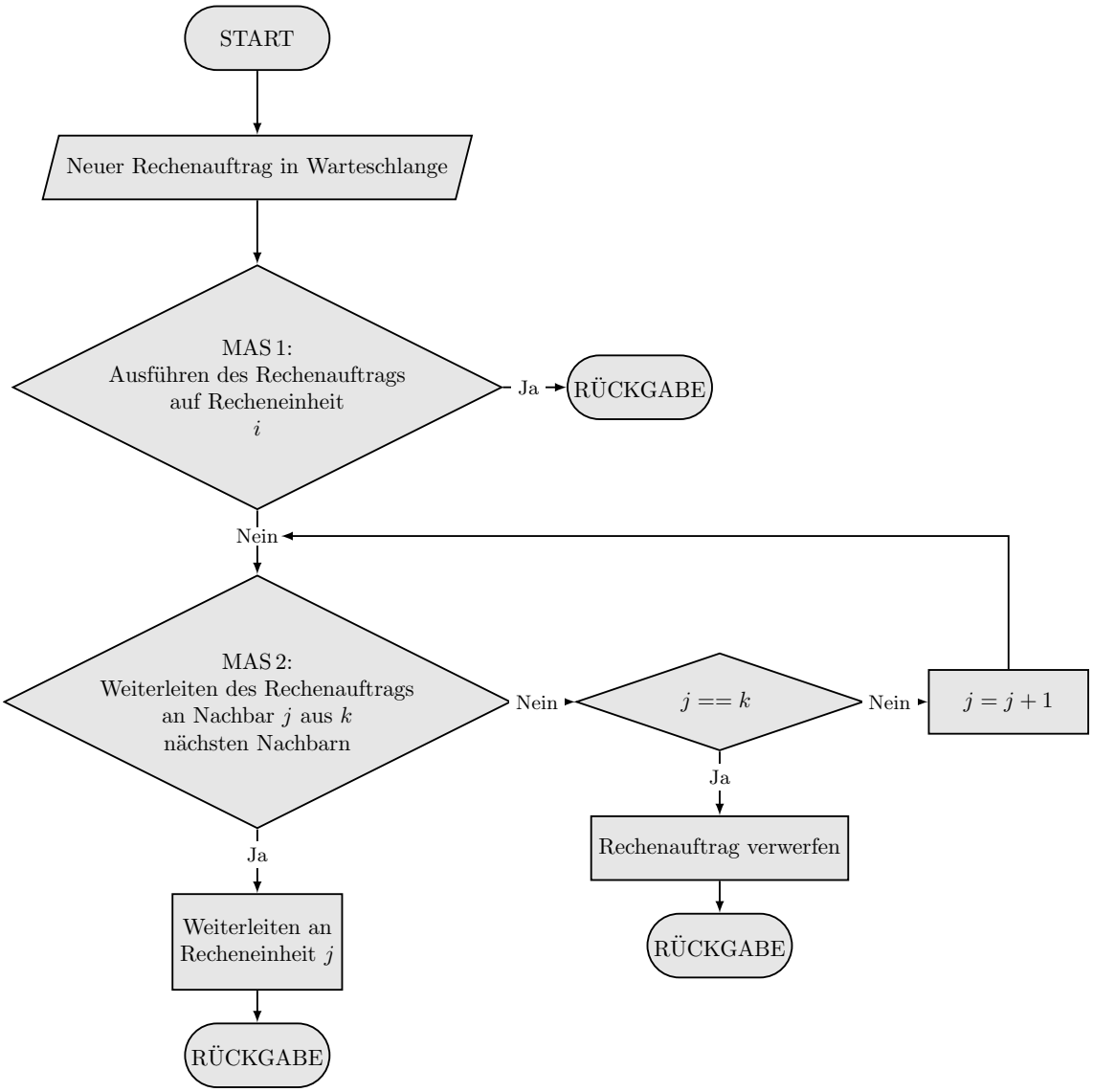


Abbildung 6.4: Interaktion der Agentensysteme MAS1 und MAS2 (Rosenberger, Urlaub, Rauterberg u. a. 2022)

- Datenverluste: Datenverluste sind während der Routenfindung akzeptiert. Der Ansatz ist nicht geeignet für Applikationen, in denen keine Datenverluste zulässig sind.
- Aktionsraum: Die Agenten entscheiden über die Erhöhung der Auslastung durch Ausführung oder Weiterleiten von Rechenaufträgen. Eine Reduzierung der Last durch Beendigung laufender Rechenaufträge ist nicht möglich.
- Anwendungsgebiet: Der Trainingsprozess ist möglichst generisch gestaltet, sodass die Agenten für den Einsatz in einer Vielzahl industrieller Anlagen geeignet sind. Daraus folgt, dass die Strategie der Agenten nicht auf spezifische Anwendungsfälle trainiert ist. Spezifische Anwendungsfälle, beispielsweise das Erlernen von Langzeitzusammenhängen und zyklischen Prozessen, erfordert Training mit spezifischen Daten.

6.8 Vergleich unterschiedlicher Architekturen

In diesem Abschnitt werden drei wesentliche Architekturen von Agentensystemen, dem Single-Agenten sowie zwei kooperativen MAS, zentralisiert und vollständig dezentral, gegenübergestellt. Dazwischen gibt es zahlreiche Mischformen, die in diesem Rahmen nicht weiter betrachtet werden.

Single-Agenten-System

Die Anwendung eines Single-Agenten-Systems auf die beschriebene Problemstellung (Abschnitt 6.3.1) ist in Abbildung 6.5 schematisch skizziert.

Der Single-Agenten-Ansatz ist durch ein MDP mit den nachfolgenden Werten beschrieben.

- **Umgebung**: Die Umgebung stellt das IIoT-Netzwerk dar.
- **Menge der Agenten**: Es existiert nur ein einziger Agent $N = 1$.
- **Zustandsraum**: Die beobachteten Zustände sind die CPU-Auslastungen aller Recheneinheiten $d_i \in D$ in % in der Umgebung sowie die erwartete zusätzliche CPU-Last des jeweiligen Rechenauftrags. Der Zustandsraum besitzt damit die Mächtigkeit $S = |D + 1|$ mit $s_0 =$ erwartete Rechenlast, $s_i =$ CPU-Auslastung von Recheneinheit d_i.
- **Aktionsraum**: Der Aktionsraum umfasst alle beobachteten Recheneinheiten D, die für die Ausführung eines Rechenauftrags ausgewählt werden können $a_{t,1}, a_{t,2}, ..., a_{t,D}$ sowie die zusätzliche Aktion, einen Rechenauftrag nicht auszuführen $a_{t,0}$ und besitzt somit die Mächtigkeit $A = |D + 1|$.
- **Belohnungsfunktion**: Die Allokation von Ressourcen für die Bearbeitung eines Rechenauftrags wird belohnt, solange der vorgegebene Schwellwert nicht

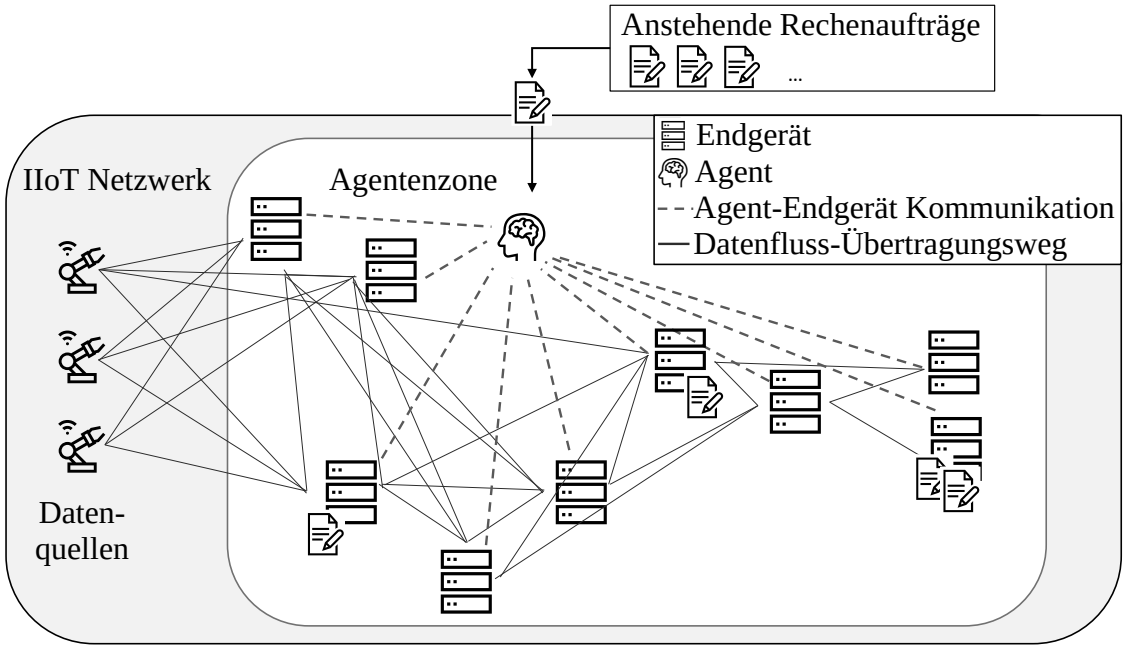


Abbildung 6.5: Schematische Architektur des Single-Agenten Systems (in Anlehnung an Rosenberger, Urlaub, Rauterberg u. a. 2022)

überschritten wird. Keine Zuweisung oder Zuweisung mit Schwellwertüberschreitung werden bestraft. Für die Belohnungsfunktion $R(s, a)$ gilt:

$$R(s,a) = \begin{cases} \text{Belohnung } r = 20 & \text{wenn } a_t = 1 \text{ und } s_{j,t+1} < \text{Schwellwert} \\ \text{Bestrafung } r = -20 & \text{wenn } a_t = 1 \text{ und } s_{j,t+1} > \text{Schwellwert} \\ \text{Belohnung } r = 20 & \text{wenn } a_t = 0 \text{ und alle } s_{i,t+1} > \text{Schwellwert} \\ \text{Bestrafung } r = -5 & \text{wenn } a_t = 0 \text{ und min. ein } s_{i,t+1} < \text{Schwellwert} \\ \text{Bestrafung } r = -10 & \text{wenn } a_t = 0 \text{ und alle } s_{i,t+1} < \text{Schwellwert} \end{cases}$$

Zentralisiertes, hierarchisches MAS

Der zweite Ansatz beschreibt, wie in Abbildung 6.6 schematisch dargestellt, ein MAS, welches hierarchisch aufgebaut ist und einen übergeordneten Agenten besitzt. Dieser Agent stellt eine zentrale Instanz in dem System dar. Die Gesamtmenge der Agenten ist daher nicht homogen. Der zentrale Agent kommuniziert mit allen Sub-Agenten, jedoch nicht direkt mit den Recheneinheiten. Die Sub-Agenten kommunizieren untereinander nicht. Sie überwachen jeweils einen kleinen Teil des Netzwerks und sind entsprechend dem oben beschriebenen MDP des Single-Agenten aufgebaut. Die mathematische Beschreibung des übergeordneten Agenten ist wie folgt:

- **Umgebung**: Die Umgebung entspricht dem IIoT-Netzwerk mit den Sub-Agenten.
- **Menge der Agenten**: Es gibt genau einen zentralen Agenten $N = 1$.

- **Zustandsmenge**: Die Zustandsmenge besteht aus den Durchschnittswerten der CPU-Auslastung der m Endgeräte in den Netzwerkzonen, die durch die Sub-Agenten i^{sub} überwacht werden. Die Mächtigkeit des Zustandsraums ist daher $S = |N^{sub} + 1|$ mit $s \in S$.

$$s^{sub_i} = \frac{1}{d^{sub_i}} \sum_{m=0}^{n} s^{1,m} \tag{6.8}$$

- **Aktionsmenge**: Die Aktionsmenge des zentralen Agenten entspricht der Anzahl an Sub-Agenten $A = |N^{sub}|$. Der Agent wählt je Rechenauftrag genau einen Sub-Agenten, dem er den Rechenauftrag übergibt.

- **Belohnungsfunktion**: Die Belohnungsfunktion belohnt die Zuweisung an Sub-Agenten von gering ausgelasteten Netzwerkregionen.

$$R(s,a) = \begin{cases} \text{Belohnung } r = 20 & \text{wenn Aktion } a_{t,i} = 1 \text{ und } s_t^{sub_i} < s_t^{sub_j} \\ \text{Bestrafung } r = -20 & \text{sonst} \end{cases}$$

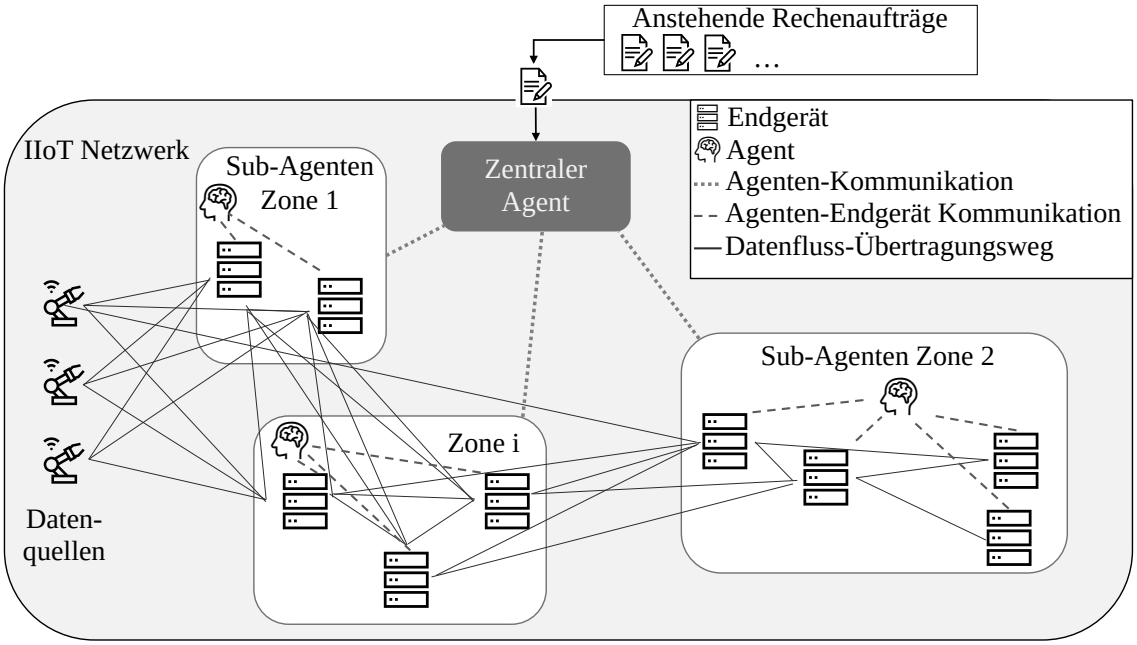


Abbildung 6.6: Schematische Architektur des zentralisierten, hierarchischen Systems (in Anlehnung an Rosenberger, Urlaub, Rauterberg u. a. 2022)

Vollständig dezentrales MAS

Das vollständig dezentrale System entspricht dem in Abschnitt 6.4 vorgestellten MAS1. Der schematische Aufbau ist in Abbildung 6.2 dargestellt.

Gegenüberstellung

Der Single-Agenten Ansatz überzeugt durch seinen uneingeschränkten Beobachtungsraum. Durch die direkte Kommunikation zu den Recheneinheiten sind Ungewissheiten minimiert. Ein Single-Agent hat jedoch nur eine begrenzte Rechenkapazität und das neuronale Netz kann nicht unendlich groß gestaltet werden, sodass die Anzahl der Ein- und Ausgaben limitiert ist. Der zentrale, übergeordnete Agent im zentralisierten MAS dagegen ist auf die Darstellung der Beobachtung durch die Sub-Agenten angewiesen. Die Unterteilung des Netzwerkes in einzelne Bereiche, die wiederum von den Sub-Agenten verwaltet werden, macht das System in begrenztem Maß skalierbar. Dennoch kann der zentralen Agenten keine unendlich große Anzahl an Sub-Agenten beobachten. Der wesentliche Vorteil des dezentralen MAS ist seine Adaptivität bezüglich dynamischen Änderungen in der Umgebung, wie sie bereits in Abschnitt 6.3 beschrieben ist. Durch die festgelegte Anzahl der Ein- und Ausgabewerte sowohl beim Single-Agenten und als auch beim zentralen MAS ist die Adaptivität dort nicht gegebenen.

Der Kommunikations-Mehraufwand ist bei dem dezentralen MAS abhängig von der Implementierung. Während eine $n : m$ Kommunikation zwischen den Agenten zu einem enormen Mehraufwand führt, ist die als MAS1 implementierte Kommunikation $1 : n^{local}$, d.h. ein Agent kommuniziert lediglich zu seinen nächsten Nachbarn, auf ein Minimum begrenzt. Neben der Anzahl der Kommunikationsteilnehmer ist auch der Kommunikationsinhalt und die Kommunikationsmenge zu berücksichtigen. Diese ist wiederum für alle drei vorgestellten Ansätze abhängig von der Implementierung. Unterschiedliche Implementierungen werden in (Alagha 2019) beschrieben.

Die beiden zentralisierten Ansätze, Single-Agent und zentralisiertes MAS, besitzen wesentliche Nachteile hinsichtlich ihrer Ausfallsicherheit. Die zentralen Instanzen stellen eine einzelne Fehlerstelle (single-point of failure) dar.

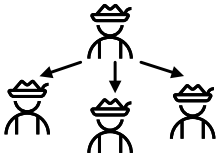
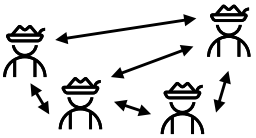
In Tabelle 6.2 werden die vorgestellten Systeme qualitativ verglichen.

6.9 Evaluation

6.9.1 Versuchsaufbau

Die Experimente werden an einem Testnetzwerk, bestehend aus realen und virtualisierten industriellen Endgeräten, durchgeführt. Die Endgeräte der Firma Bosch Rexroth sind eine physische ctrlX CORE X3 mit einer 64 bit Quad Core ARM A53 CPU, 1 GB RAM und 4 GB eMMC Festplattenspeicher sowie virtualisierte Steuereinheiten ctrlX CORE virtual mit einer 64 bit Quad Core AMD CPU, 4 GB RAM and 4 GB eMMC Festplattenspeicher, betrieben als virtuelle Maschinen auf einem Laptop mit

Tabelle 6.2: Bewertung verschiedener Architekturen von Agentensystemen (Rosenberger, Urlaub, Rauterberg u. a. 2022)

	Single-Agent	Zentralisiertes MAS	Dezentrales MAS
			
Skalierbarkeit	nicht gegeben	begrenzt	hoch
Adaptivität	nein	nein	ja
Beobachtung	voll	teilweise	gering
Kommunikation	mittel	mittel	niedrig/mittel/hoch*
Ausfallrisiko	hoch	hoch	gering

* Kommunikationslast abhängig von der Implementierung (siehe Alagha 2019)

einem Intel Xeon E3-1503M v5 Prozessor und 32 GB RAM. Das Betriebssystem ist in beiden Fällen Linux Ubuntu Core 18, welches die Applikationen im Snap-Format erwartet. Das Netzwerk besteht aus einer physischen und sechs virtuellen Steuereinheiten, sodass jeweils sieben MAS1 und MAS2 Agenten vorhanden sind. Als Netzwerkstruktur wird die Gitter-Struktur gewählt (siehe Abbildung 6.7), eine Sonderform des vermaschten Netz, die sich dynamisch ändern kann.

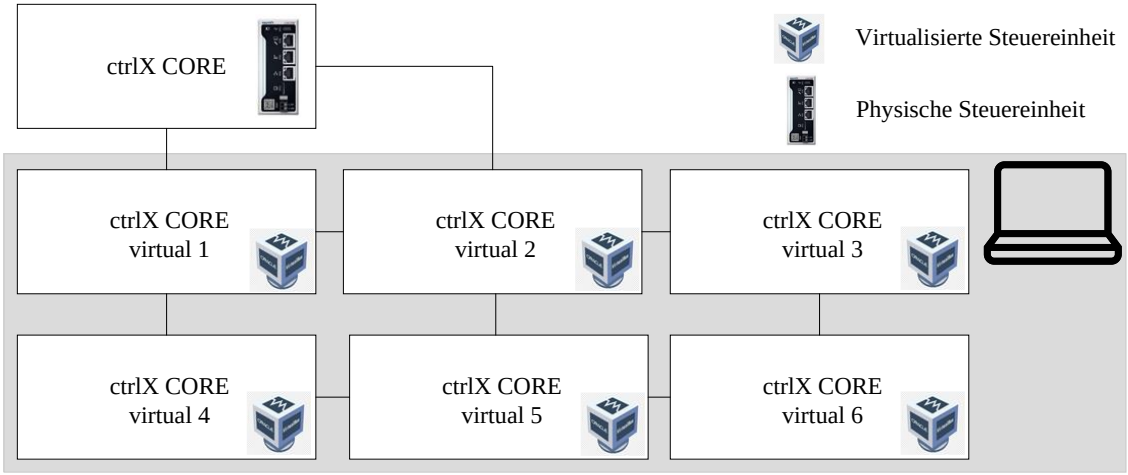


Abbildung 6.7: Schematische Struktur des Evaluationsnetzwerks (grau hinterlegt: Netzwerk aus virtualisierten Steuereinheiten ausgeführt auf einem Notebook)

Für die Experimente werden beide MAS in Python 3.8.10 implementiert sowie die nachfolgend aufgeführten Python-RL-Bibliotheken genutzt.

- OpenAI Gym[1]: Die OpenAI Gym ist eine Python Bibliothek von OpenAI[2], welche episodenbasiertes RL unterstützt. Sie ermöglicht eine Abstraktion des POMDP für Single-Agenten.
- PettingZoo[3] (Version 1.15.0): Analog zur OpenAI Gym für Single-Agenten stellt die Bibliothek PettingZoo eine Umgebung für Multi-Agenten zur Verfügung, die eine Abstraktion von AEC Spielen darstellen (Terry, Black u. a. 2020).
- SuperSuit (Version 3.3.3): In Kombination mit PettingZoo wird das Paket SuperSuit eingesetzt, um die Ausführung der Umgebungen zu parallelisieren, sodass POMG, welche sich äquivalent zu AEC Spielen verhalten, implementiert werden können.
- Stable Baselines 3 (SB3)[4] (Version 1.1.0): Die SB3 Bibliothek des Deutschen Zentrum für Luft- und Raumfahrt beinhaltet vortrainierte Implementierungen von modellfreien RL-Algorithmen, welche in den zuvor genannten Umgebungen eingesetzt werden können, um die optimale Strategie zu finden.

Für die Snaps werden die SB3-Modelle als TensorFlow Lite-Modelle exportiert, welche explizit für den Einsatz auf ressourcenlimitierten Endgeräten geeignet sind.

6.9.2 Trainingsergebnisse

In diesem Abschnitt werden die separat, nach Algorithmus 3 und 4 mit den Trainingsparametern nach Tabelle 6.1 trainierten Modelle evaluiert. Die Abbildungen 6.8 und 6.9 zeigen die Ergebnisse des MAS1 bzw. des MAS2 nach dem durchgeführten Training.

An den Grafiken ist erkennbar, dass die Agenten-Systeme im Training das gewünschte Verhalten erlernen. Die Agenten (durchgezogene Linien) allokieren verfügbare Ressourcen für zusätzliche Rechenaufträge und beachten dabei den vorgegebenen Schwellwert von 80 % (gestrichelte Linie), sodass keine Überlastung entsteht.

6.9.3 Evaluation des Ressourcenverbrauchs durch die Agenten

Tabelle 6.3 fasst die Ergebnisse der Messung des Ressourcenverbrauchs, d.h. CPU-Auslastung, RAM-Auslastung, Datenverkehr und Festplattenspeicher, eines Snaps, der beide Agenten vom Typ MAS1 und MAS2 enthält, auf dem physischen Endgerät (reale ctrlX CORE) zusammen. Die Zeit für die Ausführung der Agenten, d.h. der Entscheidungsfindung selbst, ist Tabelle 6.4 zu entnehmen.

[1] OpenAI Gym Documentation `https://www.gymlibrary.ml/`
[2] OpenAI: `https://openai.com/`
[3] PettingZoo: `https://www.pettingzoo.ml/`
[4] Stable-Baselines3 Docs: `https://stable-baselines3.readthedocs.io/en/master/`

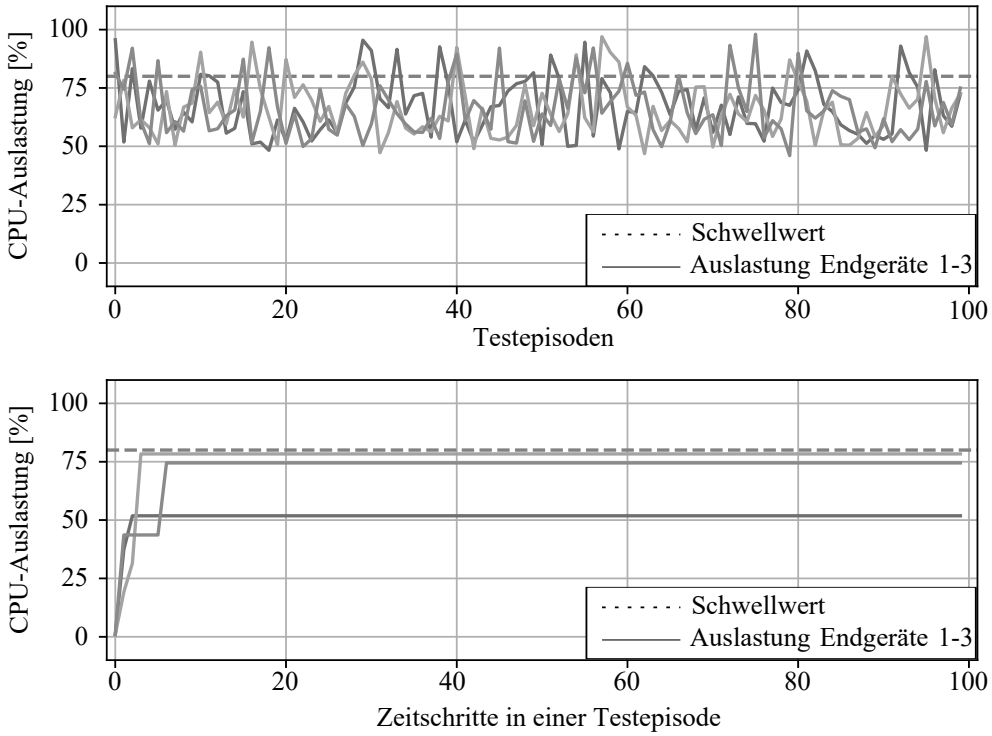


Abbildung 6.8: Evaluation nach dem Training des MAS1 mit drei Agenten (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Tabelle 6.3: Ressourcenverbrauch des Agenten-Snaps (Rosenberger, Urlaub, Rauterberg u. a. 2022)

	CPU ∅ [%]	**CPU max.** [%]	**Festplatten-speicher** [MB]	**RAM**[a] ∅ [%]	**RAM**[a] **max.** [%]
Agenten-Snap	0,81	6,60	53,8	4,6	4,7

[a] RAM wird lediglich für die Warteschlange sowie die Liste der ausgeführten Rechenaufträge und ihren Metadaten benötigt.

Der zusätzliche Datenverkehr aufgrund der Kommunikation zwischen den Agenten besteht aus drei Inhalten, welche im Netzwerk im JSON-Format übermittelt werden.

- Beschreibung des Rechenauftrags,
- Abfragen der Agenten, um die Beobachtungen zu erhalten und
- Informationen über die gefundene Route.

Der Datenverkehr berechnet sich aus der Anzahl der Entscheidungen und Hops sowie der Größe der Inhalte. Die drei Bestandteile werden aufsummiert.

- Anzahl der weitergeleiteten Rechenaufträge multipliziert mit der Größe der Aufgabenbeschreibung ($\sim$ 217 bytes),

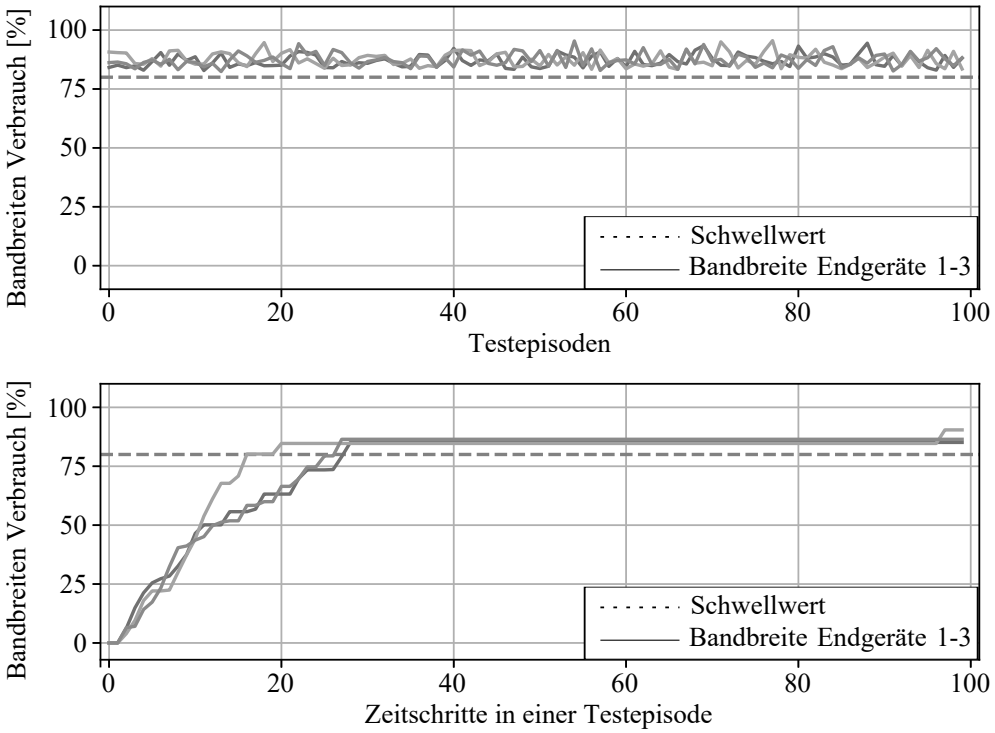


Abbildung 6.9: Evaluation nach dem Training des MAS2 mit drei Agenten (Rosenberger, Urlaub, Rauterberg u. a. 2022)

- Anzahl der Entscheidungen über das Weiterleiten des Rechenauftrags multipliziert mit der Größe der Zustände des Nachbarn (~ 50 bytes)
- Anzahl der gefundenen Routen multipliziert mit der Anzahl der Hops multipliziert mit der Information über die Route (~ 149 bytes).

Tabelle 6.4: Laufzeiten der Agenten des MAS1 und MAS2 (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Agent	$\mathbf{t_{avg}}$[ms]	$\mathbf{t_{max}}$[ms]
MAS1-Agent	0,74	1,16
MAS2-Agent	0,81	0,98

6.9.4 Evaluation der Inferenz der Agentensysteme

Der zweite Teil der Evaluation betrachtet die Inferenz des Agenten-Systems. Die intelligente Ressourcenallokation wird dazu in einem Testaufbau untersucht. Dabei wird insbesondere die Performanz der Agenten betrachtet. Weitere Evaluationen finden im Rahmen der Betrachtung des Gesamtsystems (siehe Abschnitt 7.1) statt. Abbildung 6.10 stellt die CPU-Auslastung der Agenten-Snaps sowie den Bandbreiten-

und die CPU-Auslastung von zwei benachbarten, durch Agenten verwalteten Endgeräten mit intelligenter Ressourcenallokation dar.

Die Ergebnisse ähneln den Trainingsergebnissen (vgl. Abbildung 6.9). Bei der Evaluation hat nur Agent 1 Rechenaufträge zugewiesen bekommen. Da das Endgerät 1 bereits bis zum Schwellwert ausgelastet wird, leitet der Agent 1 Rechenaufträge an Agent 2 (von Endgerät 2) weiter. Es wird bestätigt, dass das in der Trainingsumgebung erlernte Verhalten der Agenten auf die Testumgebung übertragbar ist. Ein bestehendes Problem ist der Fall, wenn die Auslastung des Endgeräts an den Schwellwert grenzt. In diesem Fall werden die Rechenaufträge weitergeleitet oder zurückgewiesen und wieder in der Warteschlange angestellt. Dadurch muss der Agent in sehr schneller Zeit sehr viele Rechenaufträge weiterleiten oder zurückweisen und das Endgerät wird überlastet. Ein Mechanismus, der dem Agenten bei einer Gesamtauslastung um den Schwellwert keine neuen Rechenaufträge zuweist, wäre eine mögliche Lösung.

Die Analyse der Ergebnisse hinsichtlich Latenzen und benötigten Hops von erfolgreich zugewiesenen Rechenaufträgen ist in Tabelle 6.5 erfasst.

Tabelle 6.5: Auswertung von Latenzen und Hops bei erfolgreicher Weiterleitung (Rosenberger, Urlaub, Rauterberg u. a. 2022)

Zeit[a] ∅ [s]	**Zeit**[a] **max.** [s]	**Hops** h ∅ []	**Hops** h **max.** []
4,84	11,87	1,84	3

[a] Zeit zwischen erster Entscheidung und erfolgreichem Routing zu ausführendem Endgerät.
Die Werte sind stark von der Gesamtauslastung des Netzwerks abhängig.

6.9.5 Limitierungen der experimentellen Evaluation

- Implementierung: Die Agenten sind als Snap implementiert. Die gewählte Programmiersprache ist Python. Das Modell wird in ein TensorFlow Lite-Modell konvertiert und ist insbesondere für Endgeräte mit limitierten Ressourcen geeignet. Als allgemeineres Format könnte das Open Neural Network Exchange (ONNX)-Format in Betracht gezogen werden.
- Netzwerk Topologie: Es ist nur ein Netzwerk in Gitter-Topologie, die eine Sonderform eines vermaschten Netzwerks darstellt, evaluiert. Da vermaschte Netzwerke die komplexeste Topologie darstellen, wird davon ausgegangen, dass die meisten Netzwerkformen abgedeckt sind.
- Netzwerkgröße: Das Evaluationsnetzwerk ist im Vergleich zu realen industriellen Netzwerken klein. Die Evaluation in heterogenen und größeren Netzwerken wäre wünschenswert, um genauere Aussagen über das Verhalten der Agentensysteme und das volle Potential der vorgestellten Methode zu treffen.

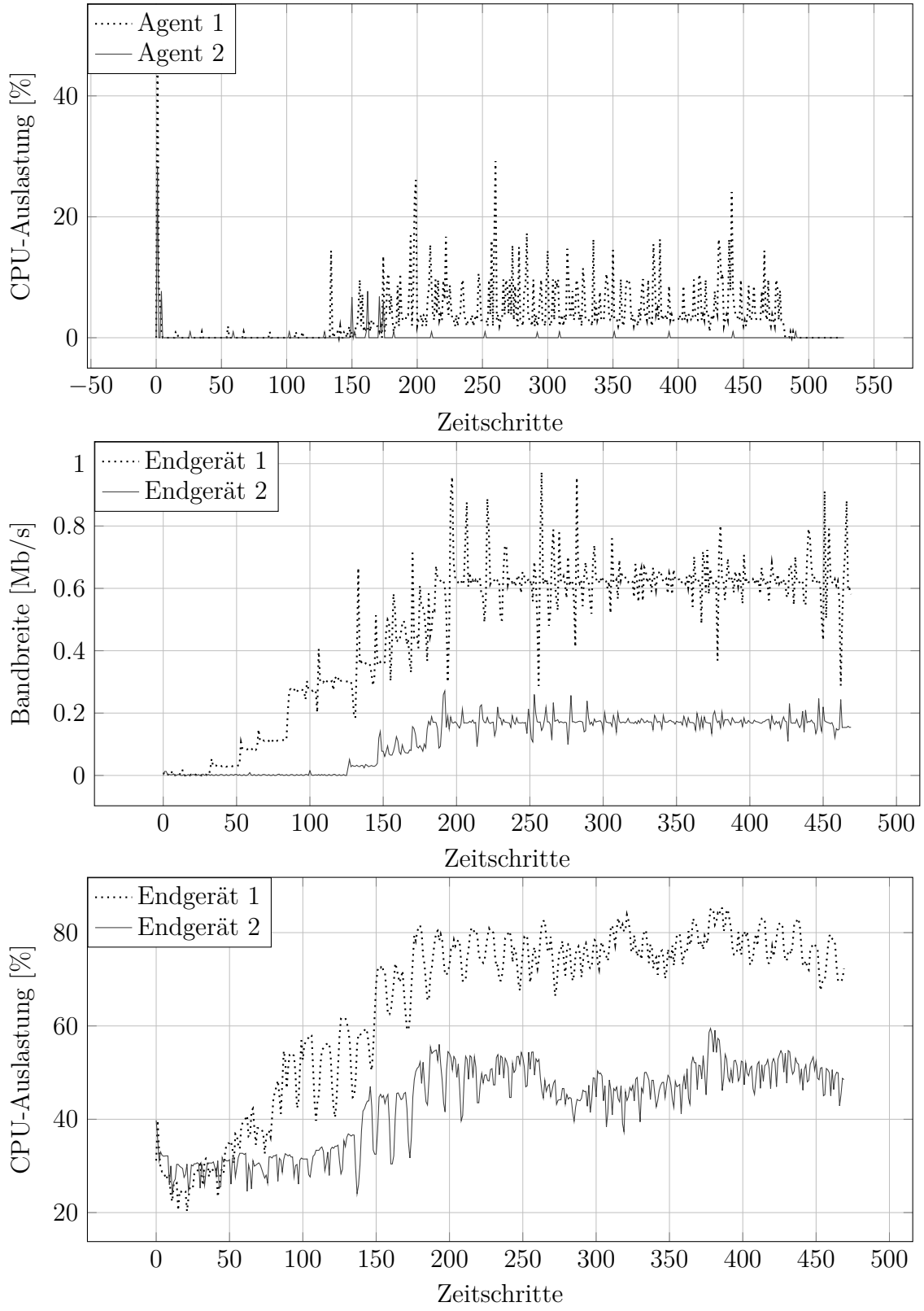


Abbildung 6.10: Ressourcenallokation durch MAS1 und MAS 2 in der Testumgebung

- Anwendungsbereich: Der Anwendungsbereich des IIoT konnte in der Evaluation nur in begrenztem Umfang getestet werden. Es wurden nur homogene Endgeräte auf Steuerungsebene in einem Testaufbau evaluiert.
- Adaptivität: Die Adaptivität an dynamische Änderungen wurde in diesem Testaufbau nicht experimentell evaluiert. Sie ist Bestandteil der Betrachtung des Gesamtsystems in Abschnitt 7.1.
- Metriken: Ausgewertet werden der Verbrauch von Rechen- und Kommunikationsressourcen des Agenten-Snaps sowie die Latenzen und die Hops der erfolgreich gefundenen Routen. Zusätzliche Metriken wie der Kontext der gesamten Ressourcennutzung im Netzwerk und der Energieverbrauch der Agenten sind bisher nicht berücksichtigt.

6.10 Ergebnisse und Diskussion

Das vorgestellte MARL-System kann die Rechen- und Kommunikationsressourcen von Endgeräten für die Bearbeitung zusätzlicher Rechenaufträge allokieren. Hervorzuheben ist die Eignung für Endgeräte mit begrenzten Ressourcen in sich dynamisch ändernden Netzwerken. Die Adaptivität wird einerseits durch die Unterteilung des Systems in kleinste Einheiten und das sequentielle Treffen von Entscheidungen erreicht, sodass die Anzahl der Rechenaufträge, Netzwerkknoten und Verknüpfungen variieren kann. Andererseits wird der Trainingsprozess generisch gestaltet, sodass die Agenten nicht auf bestimmte Verarbeitungsalgorithmen oder Anlagen beschränkt sind.

Drei Architekturen mit unterschiedlichen Ausprägungen der Zentralität werden gegenübergestellt und bewertet. Dabei wird der vollständig dezentrale Ansatz für die gegebenen Randbedingungen und Anforderungen als der Zielführendste ausgewählt. Dennoch können in Abhängigkeit der Anforderungen auch andere Architekturen für andere Anwendungen besser geeignet sein.

Die Evaluation hat die Anwendbarkeit von MARL auf ressourcenlimitierten industriellen Endgeräten bestätigt. Die Experimente bestätigen, die geringe Komplexität der Berechnungen, die sehr geringe Größe der Modelle und das schnelle Treffen von Entscheidungen durch die Agenten. Obwohl die Agenten innerhalb von Millisekunden entscheiden, beträgt die gesamte Latenz zwischen der ersten Entscheidung eines Agenten und der erfolgreich gefundenen Route mehrere Sekunden. Dies ist durch die Anzahl der Entscheidungen und besonders durch die Zeitspannen zur Abfrage des aktuellen Zustands der Umgebung begründet. Sind alle Endgeräte stark ausgelastet, sind die Agentensysteme nicht in der Lage, Ressourcen zuzuweisen, sondern verursachen stattdessen einen Mehraufwand durch das Weiterleiten von Anfragen sowie der Ausführung der Agenten selbst. In diesem Fall bietet das MARL-System keine Vorteile, sondern führt zu einer zusätzlichen Belastung des Netzwerks.

In zukünftigen Experimenten ist die Erfolgsrate des Systems in Abhängigkeit der durchschnittlichen Gesamtauslastung im Netzwerk zu erfassen und der Umgang mit

dynamischen Änderungen zu evaluieren. Für aussagekräftigere Bewertungen des Systems sind weitere Untersuchungen in realen IIoT Netzwerken mit mehr Endgeräten und Verknüpfungen anzustellen.

Für eine bessere Einordnung der Ergebnisse könnte ein Vergleich zu manueller, statischer Allokation in Betracht gezogen werden. Wünschenswert wäre ein Vergleich der Allokation durch die MASs zu einem System, dass dem Stand der Technik entspricht. Da die Datenverarbeitung bisher - falls überhaupt vorhanden - meist in der Cloud ausgeführt wird und es nach aktuellem Stand der Technik kein Verfahren für eine Ressourcenallokation im Netzwerk auf den Endgeräten im IIoT gibt (siehe Abschnitt 6.2), kann kein quantitativer Vergleich zu der Performanz eines bestehenden Verfahrens gezogen werden.

In weiteren Ausarbeitungen sind die folgenden Aspekte bei der Gestaltung der MASs in Betracht zu ziehen. Der Zustandsraum des MAS1 kann um RAM, Festplattenspeicher und die bereits zurückgelegten Hops erweitert werden. Bei dem MAS2 ist die Erweiterung des Zustandsraums um die Priorität der Daten oder des Rechenauftrags und die Zugriffsrechte des jeweiligen Endgeräts für Lesen bzw. Verarbeiten der Daten zu betrachten. Außerdem ist es möglich, dass die Einführung einer globalen Belohnung bei kombiniertem Training beider Systeme das Verhalten der Agenten verbessert (siehe Abschnitt 6.6).

Da die Optimierung von Flüssen oft mit dezentralen Systemen einhergeht, ist zu erwähnen, dass sich MARL-Systeme generell für Probleme der Flussoptimierung eignen und somit nicht nur auf Datenflüsse, sondern z. B. auch auf Verkehrs-, Energie- oder Materialflüsse, anwendbar sind und der hier vorgestellte Ansatz auch für diese Anwendungsgebiete von Interesse sein kann.

Die Wahl von Parameterwerten hat großen Einfluss auf die Qualität der Lernergebnisse. In dieser Arbeit werden insbesondere die Lernrate sowie die Belohnungsfunktion manuell optimiert. Die Beschreibung und Ergebnisse der Tests mit verschiedenen Parameterwerten sind der Veröffentlichung (Urlaub und Rosenberger 2022) und Abschnitt A.5 im Anhang zu entnehmen. Die grundlegenden Parameter wie die Größe der Neuronalen Netze wurden nicht optimiert, da die ausgewählten, vortrainierten SB3-Modelle bereits zu zufriedenstellenden Ergebnissen führten.

Der dritte Lösungsansatz, die Ressourcenoptimierung, stellt für die zuvor in Kapitel 4 und 5 beschriebenen Optimierungsansätze eine wesentliche Grundlage dar, da die Rechen- und Kommunikationsressourcen der limitierende Faktor für die Datenverarbeitung am Netzwerkrand sind. Die vorgestellte dezentrale Allokation von Ressourcen durch interagierende Agentensysteme ist hierfür besonders auf den Einsatz in sich dynamisch verändernden Netzwerken, wie es im IIoT der Fall ist, ausgelegt.

KAPITEL 7

Zusammenfassung und Ausblick

Dieses Kapitel umfasst die Betrachtung des Gesamtsystems, eine Zusammenfassung der wesentlichen Inhalte der Arbeit und die Beantwortung der zuvor gestellten Forschungsfragen. Die Arbeit wird kritisch gewürdigt und der wissenschaftliche Beitrag benannt. Das Kapitel schließt mit einem Ausblick und Überblick zu weiterem Forschungsbedarf.

7.1 Betrachtung des Gesamtsystems

In den Kapiteln 4-6 wurden Optimierungsansätze vorgestellt, die den Anspruch haben, unabhängige Datenflussoptimierungen darzustellen. Deren Umsetzung und Anwendbarkeit wurden daher zunächst einzeln experimentell nachgewiesen und evaluiert, weshalb abschließend eine exemplarische Betrachtung des gemeinsamen Einsatzes innerhalb eines Gesamtsystems folgt. Das Gesamtsystem beschreibt in diesem Rahmen sowohl das Zusammenspiel der einzelnen Optimierungsansätze als auch deren Interaktion mit einer industriellen Anlage.

In dieser prototypischen Betrachtung wird hierzu eine Simulation einer industriellen Anlage mittels Hardware-in-the-Loop (HiL) Kopplung mit einem Netzwerk aus Endgeräten eingesetzt. Als Simulationsumgebung wird iPhysics (machineering GmbH & Co. KG 2022) gewählt. Das verwendete Simulationsmodell ist die in Abbildung 7.1 dargestellte Verpackungsanlage.

Das Modell wird mit vier Endgeräten gekoppelt. Das Netzwerk besteht aus drei physischen und neun virtualisierten Steuereinheiten, die über Ethernet sowie einen Router und einen Switch verbunden sind. Die Kommunikation erfolgt über TCP/IP. Eine Übersicht des Versuchaufbaus ist in den Abbildungen 7.2 und 7.3 dargestellt.

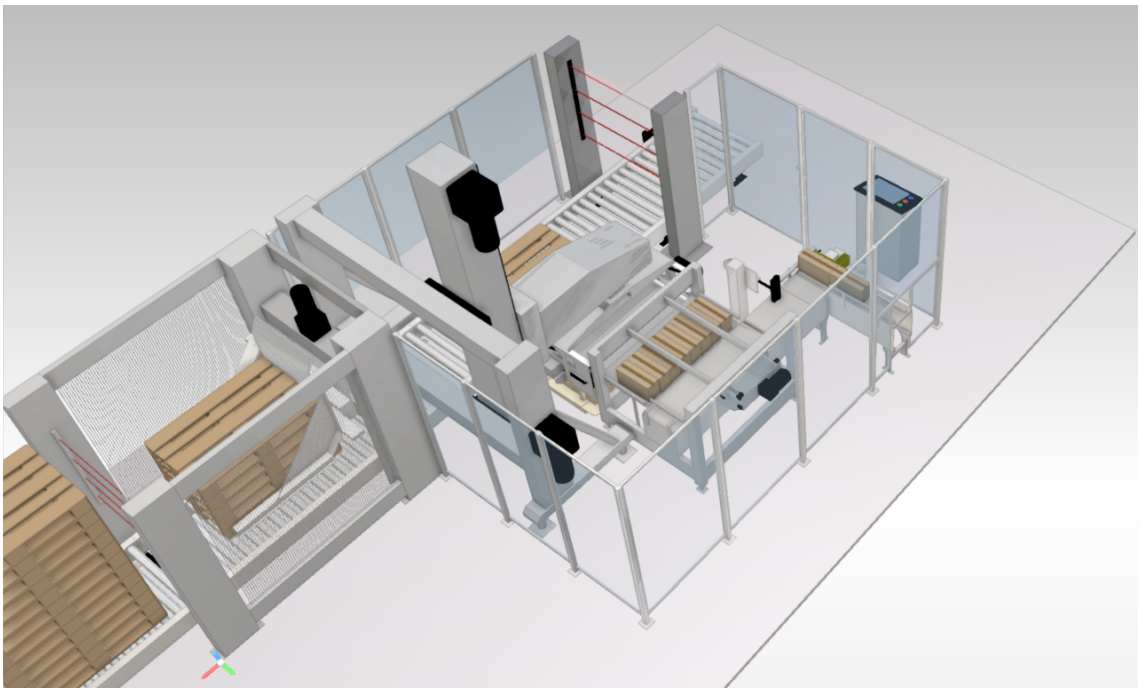


Abbildung 7.1: Simulationsmodell für die Evaluation des Gesamtsystems

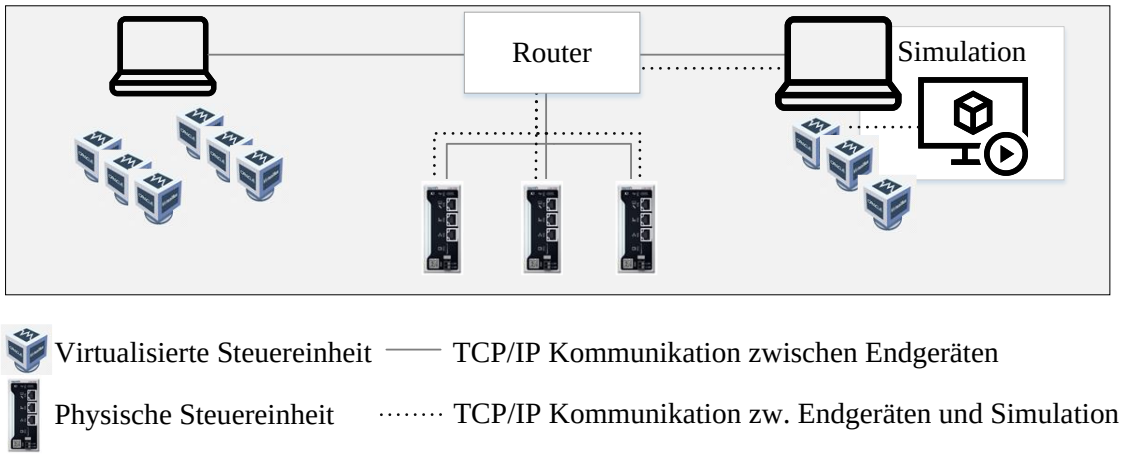


Abbildung 7.2: Versuchsaufbau für die Evaluation des Gesamtsystems (Rosenberger, Selig, Ristic u. a. 2023)

Für die HiL Kopplung werden Ausgangs- und Eingangsvariablen verschiedener Datentypen ausgewählt. Die zwischen der Simulation und den Endgeräten ausgetauschten Variablenwerte stellen die Datenflüsse dar. Eine Übersicht der Datenflüsse, auf die die Optimierungsansätze angewendet werden, ist Tabelle A.10 im Anhang zu entnehmen.

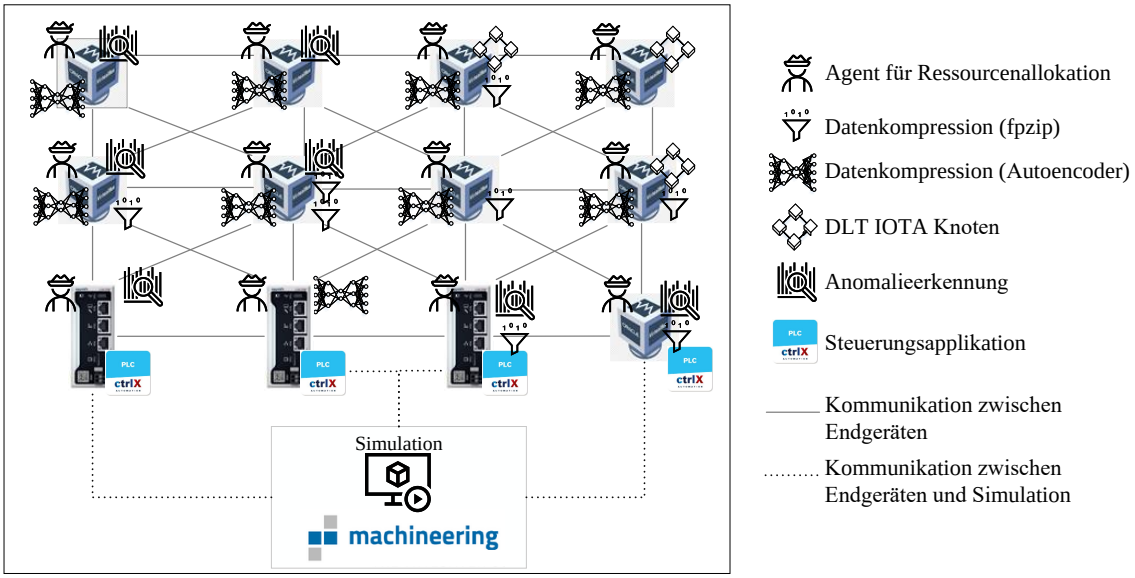


Abbildung 7.3: Darstellung des aufgebauten Gesamtsystems mit Elementen der Middleware- und Logik-Ebene (Rosenberger, Selig, Ristic u. a. 2023)

In dem aufgebauten Gesamtsystem wird geprüft, ob die einzelnen Datenflussoptimierungen auch gemeinsam ausführbar und auf industrielle Datenflüsse anwendbar sind. Dazu ist auf den Endgeräten das System aus Multi-Agenten aktiv, welche für die Ausführung der Aufgaben der Datenkompression und Anomalieerkennung geeignete Endgeräte auswählen. Parallel zum MAS wird das DLT Netzwerk IOTA betrieben, indem dieselben Endgeräte IOTA GoShimmer Knoten ausführen. In Abbildung A.11 im Anhang ist die Visualisierung des aufgebauten IOTA-Tangles sowie das GoShimmer Dashboard eines vernetzten IOTA-Knotens ersichtlich.

Den primären Einsatzzweck der Steuergeräte stellt die Steuerung von Industrieanlagen dar, wogegen die Datenverarbeitung an zweiter Stelle steht. Um dies abzubilden, sind exemplarische Steuerungsaufgaben in diesem Gesamtsystem ebenfalls umgesetzt. Vier Steuergeräte führen Steuerungsskripte nach IEC 61131-3 Standard für die Steuerung der simulierten Anlage aus. Die über HiL gekoppelten Steuergeräte dienen als Zugangspunkte für die Datenflüsse, d. h. als Datenquellen für die Algorithmen der Logik-Ebene. Aus diesem Grund werden die ausgewählten Steuerungsskripte auf vier Steuergeräte verteilt, um mehr als eine Quelle für Datenflüsse zu schaffen. In Abbildung 7.4 sind die mittels HiL gesteuerten Komponenten im Simulationsmodell markiert. Die Ausgabewerte werden mittels Hardware-in-the-Loop Kopplung wieder an die Simulation übergeben. Die dazugehörigen Steuerungsskripte, die Konfigurationsdateien für die Kommunikation zwischen Simulationsmodell

und Steuereinheiten sowie das Simulationsmodell selbst sind der GitHub Ablage unter `https://github.com/JuliRosenberger/iPhysicsEvaluation` zu entnehmen.

Zur Betrachtung des Gesamtsystems werden verschiedene Experimente umgesetzt, welche in der Tabelle 7.1 aufgeführt sind.

Es wird nachgewiesen, dass die entwickelten Algorithmen der Logik-Ebene (Datenkompression und Anomalieerkennung (siehe Kapitel 4), IOTA GoShimmer-Knoten (siehe Kapitel 5)) sowie Algorithmen der Middleware-Ebene (MAS (siehe Kapitel 6)) im Verbund einsetzbar sind. Ebenfalls konnte die Zuweisung von Algorithmen zu Endgeräten mit freien Kapazitäten bestätigt werden. Dabei haben die Messungen des Ressourcenverbrauchs gezeigt, dass die verfügbaren Ressourcen der Endgeräte für zusätzliche Datenflussanalysen eingesetzt werden, ohne dabei eine Überlastung der Endgeräte herbeizuführen.

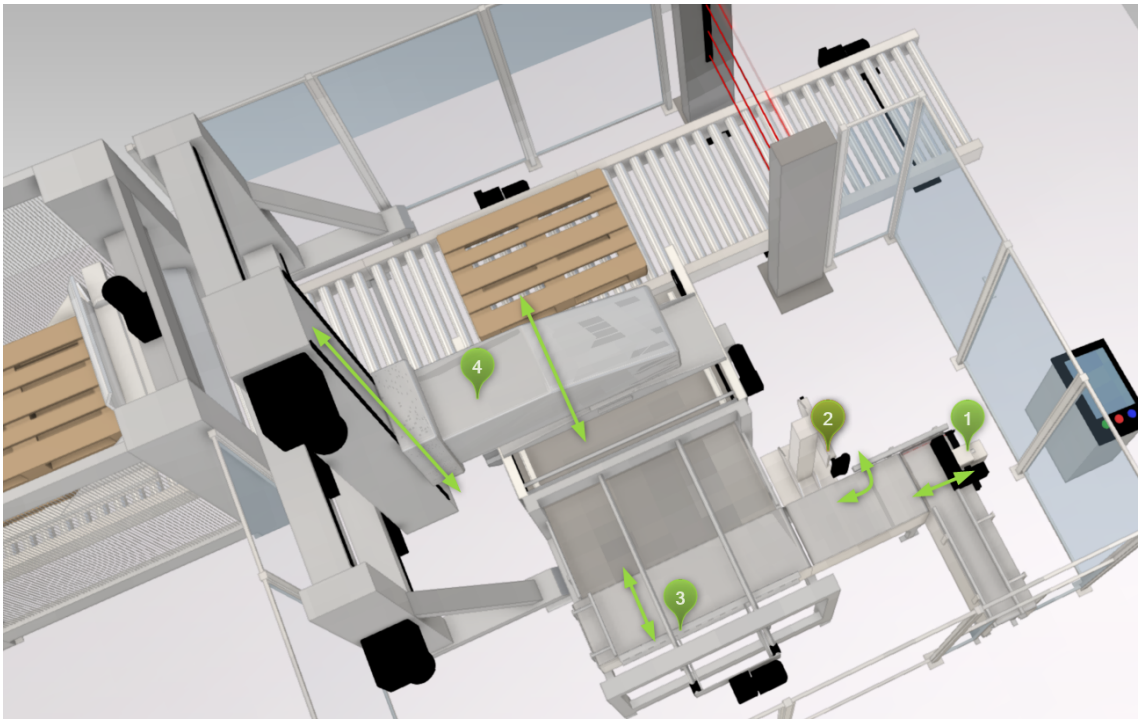


Abbildung 7.4: Simulationsmodell mit markierten Bewegungen von Bauteilen, die mittels HiL über die Steuergeräte gesteuert werden (grün) (Rosenberger, Selig, Ristic u. a. 2023)

Tabelle 7.1: Durchgeführte Experimente zur Betrachtung des Gesamtsystems (Rosenberger, Selig, Ristic u. a. 2023)

Experiment	Beschreibung	Durchgeführt
	Gesamtsystem	
Evaluation des Gesamtsystems	Alle Algorithmen laufen gemeinsam und das System interagiert mit einer simulierten Anlage	✓
• Ausführung der PLC Applikationen	Steuerung des Simulationsmodells über reale/ virtualisierte Steuereinheiten	✓
• Ausführung des MAS	Agenten, Watchdog, Ressourcenmonitoring laufen als Snaps auf den Steuereinheiten	✓
• Ausführung von IOTA GoShimmer	Faucet, Masterpeer, Replica Knoten laufen als Snaps auf den Steuereinheiten	✓
• Ausführung d. Anomalieerkennung	Algorithmus zur Anomalieerkennung *EEM-KDE* läuft als Snap auf den Steuereinheiten	✓
• Ausführung d. Datenkompression	Autoencoder und fpzip als Snaps auf den Steuereinheiten	✓
	Middleware-Ebene	
Ressourcenallokation	Agenten entscheiden über die Ausführung oder das Weiterleiten von Rechenaufträgen abhängig von den verfügbaren Ressourcen	✓
Belastungstest	Watchdog befreit Ressourcen und sendet Aufgaben zur erneuten Zuweisung zurück in die Warteschlange	✓
Adaptiv bei dynam. Änderungen	Hinzufügen/ Entfernen von Agenten o. Datenflüssen zur Laufzeit	✓
	Logik-Ebene	
Anomalieerkennung	Erkennung von manuell initiierten Anomalien	✓
Datenkompression	Autoencoder und fpzip komprimieren u. dekomprimieren Daten	✓
IOTA		✓
• Adaptiv bei dynam. Änderungen	Hinzufügen/ Entfernen von Knoten zur Laufzeit	✓
• Tangle	Tangle ist aufgebaut und synchronisiert	✓
• SCs	Ausführung von SCs, die mit der Simulation interagieren	geplant

In der Simulationsumgebung können Werte durch den Anwender zur Laufzeit überschrieben werden und somit manuell Anomalien ausgelöst werden. Mit diesem Vorgehen wird die Funktionsweise der Anomalieerkennung mittels EEM-KDE überprüft. Eine Visualisierung sowie der Ablauf ist den Abbildungen A.8 und A.9 im Anhang zu entnehmen. Dabei hat ein Agent einem Steuergerät die Ausführung der Anomalieerkennung für einen Datenfluss zugewiesen. Das Steuergerät empfing den Datenfluss und verarbeitete ihn schrittweise. In der simulierten Anlage wurde manuell eine Anomalie ausgelöst und dieser anormale Datenwert im Datenfluss an das Steuergerät übermittelt. Das Steuergerät hat die Anomalie korrekt erkannt und angezeigt.

Analog dazu wurde die erfolgreiche Kompression von Datenflüssen mittels fpzip und Autoencoder sowie deren Zuweisung durch das Agentensystem (siehe Abbildungen A.10 im Anhang) nachgewiesen.

Außerdem konnte ein Nachweis über die Adaptivität bei Netzwerkänderungen erbracht werden. Bei Entfernen bzw. Hinzufügen von Endgeräten und somit dem Entfall oder Hinzukommen von Agenten oder IOTA-Knoten kommt es zu keinen Fehlermeldungen und das MAS sowie IOTA-Netzwerk laufen stabil weiter.

Aktuell noch nicht implementiert ist ein Mechanismus, der bei Entfernen eines Endgeräts die ausgeführten Rechenaufträge der Logik-Ebene an ein benachbartes Endgerät übergibt oder wieder zurückweist, sodass sie vom Agentensystem neu vergeben werden können.

Insgesamt bestätigt sich, dass die entwickelten Algorithmen einzeln oder gemeinsam zu Optimierungen in den Datenflüssen der I4.0 beitragen können und robust gegenüber Änderungen im Netzwerk sind sowie die Ausführung auf industriellen Endgeräten möglich ist.

7.2 Zusammenfassung

Diese Arbeit befasst sich mit der Untersuchung von Optimierungspotenzialen in industriellen Datenflüssen. Sie verfolgt das Ziel, den Teilnehmern in I4.0-Umgebungen, unter Berücksichtigung der vorhandenen Berechtigungen, die für sie relevanten Informationen bereitzustellen und abzusichern.

Die Datenflussoptimierung ist ein relevantes Gebiet, um die Entwicklung der I4.0 voranzutreiben, da Lösungen zur Beseitigung bestehender Defizite aufgezeigt und evaluiert werden. Dazu gehören insbesondere die Vermeidung von Informationsverlusten, die Absicherung von Daten und Informationen sowie die direkte Verarbeitung auf den Endgeräten.

Zusammenfassend erzielen die in dieser Arbeit präsentierten Ausarbeitungen folgende Optimierungen:

1. Vermeidung von Ressourcenengpässen:
   Die Algorithmen zur Datenkompression erzielen eine Verringerung der zu übermittelnden und abzuspeichernden Datenmenge, sodass ein kostenintensiver

Ausbau der Infrastruktur oder Datenverluste durch Bandbreitenengpässe vermieden werden können. Es wird ein LSTM-Autoencoder für die Kompression von uni- und multivariaten Datenflüssen vorgestellt und mit klassischen Verfahren wie der Quantisierung, fpzip und DWT verglichen.

2. Verringerung von Latenzen:
   Die Entwicklung von Algorithmen zur frühzeitigen Informationsextraktion, in dieser Arbeit exemplarisch an einer Anomalieerkennung auf einem industriellen Endgerät gezeigt, reduziert die Latenzen zwischen Datenerfassung und Analyse der Daten. Im Rahmen der Zustandsüberwachung und präventiven Instandhaltung kann somit die Gesamtzeit zwischen der aufgetretenen Anomalie und der Reaktion darauf signifikant verringert und mögliche Schäden oder Fehler in der Produktion vermieden werden.

3. Schaffung einer Vertrauensbasis:
   In der Wirtschaft und Industrie ist Vertrauen ein hohes Gut, da es die Grundlage für Geschäftsmodelle darstellt. Obwohl in der zunehmend digitalen Geschäftswelt Vertrauen immer relevanter ist, macht die gleichzeitig zunehmende Anonymität die Schaffung von Vertrauen besonders herausfordernd. Eine Instanz, die transparent und unveränderlich die Einhaltung der Geschäftsbedingungen überwacht und somit Vertrauen schafft, kann die notwendige Basis für digitale Geschäftsmodelle darstellen. In dieser Arbeit fungiert ein DLT-Netzwerk basierend auf IOTA als diese Instanz. Anhand von zwei Anwendungsfällen wird der Einsatz demonstriert und evaluiert. In dem ersten Anwendungsfall wird Vertrauen durch die Gewährleistung von Manipulationssicherheit von Daten erzielt. Es wird die Möglichkeit einer unveränderbaren Aufzeichnung von Historien aufgezeigt. Der zweite Anwendungsfall schafft Vertrauen durch die Ausführung automatisierter, deterministischer, digitaler Verträge, sogenannte SC. Dies wird am Beispiel einer Verwaltung von Zugriffsrechten auf die Daten, Datenflüsse und Informationen demonstriert.

4. Optimierte Nutzung der verfügbaren, begrenzten Ressourcen:
   Das IIoT besteht aus einer Vielzahl von Endgeräten. Es wird angenommen, dass die Endgeräte durch ihre Kernaufgabe nicht voll ausgelastet sind, sondern noch über limitierte Rechenressourcen verfügen, die für weitere Rechenaufträge, hier exemplarisch die Ausführung von Datenkompression, Anomalieerkennung oder DLT-Knoten, genutzt werden können. Mittels RL entscheidet ein MAS darüber, auf welchen Endgeräten welche zusätzlichen Rechenaufträge ausgeführt und über welche Übertragungswege die Datenflüsse bis zur Verarbeitung weitergeleitet werden. Dadurch kann eine zunehmende Verlagerung der Datenflussverarbeitung in die Nähe der Datenerfassung ohne Ausbau der Infrastruktur erreicht werden.

Die benannten Methoden zur Optimierung wurden auf repräsentativen IIoT-Endgeräten bzw. -Netzwerken evaluiert, deren Eignung bestätigt und Grenzen hinsicht-

lich Ressourcenbedarf und Echtzeitfähigkeit aufgezeigt. Abbildung 7.5 fässt die ausgearbeiteten Datenflussoptimierungen zusammen.

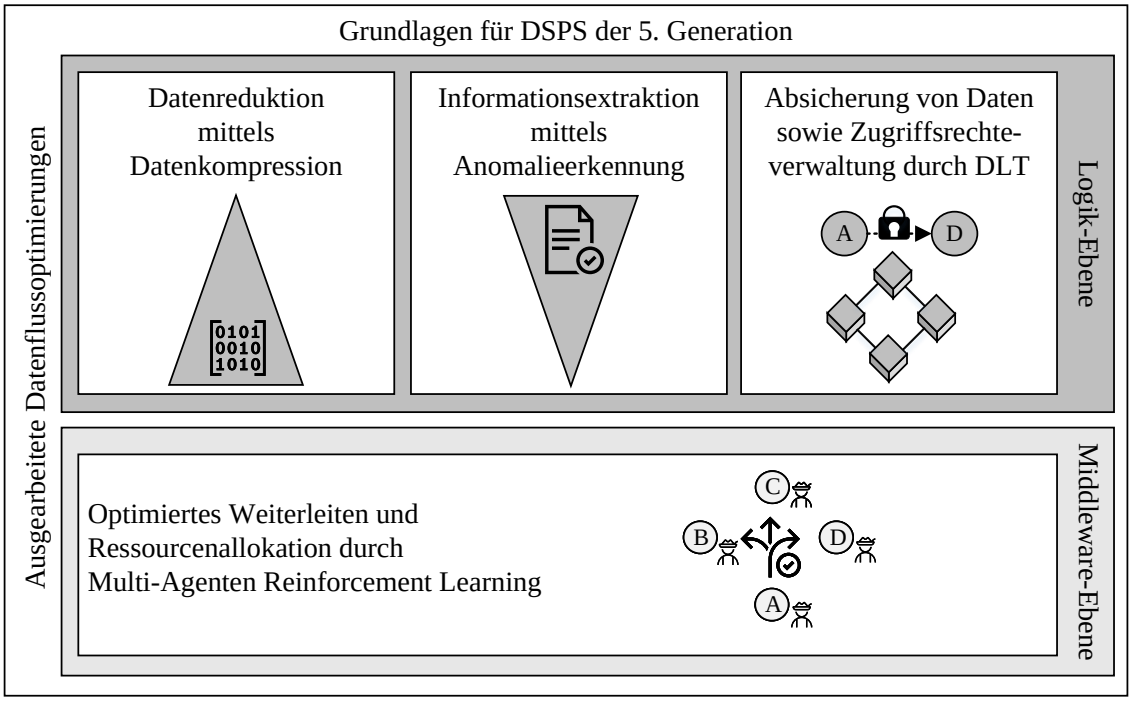


Abbildung 7.5: Zusammenfassung der ausgearbeiteten Datenflussoptimierungen. Dunkelgrau: Optimierungen in der Logik-Ebene von Datenfluss-Systemen, hellgrau: Optimierungen in der Middleware-Ebene von Datenfluss-Systemen

7.3 Beantwortung der Forschungsfragen

In diesem Abschnitt werden zunächst die spezifischen Forschungsfragen anhand der in den vorherigen Kapiteln vorgestellten Erkenntnisse beantwortet, um anschließend auf die zentrale Forschungsfrage einzugehen.

- FF1: Kann durch die Anwendung von informationstheoretischen Methoden auf industriellen Endgeräten der Verlust von Informationen verhindert werden?

  Es wird untersucht, wie Ressourcenengpässen trotz steigender Datenmenge entgegengewirkt werden kann. Dabei stellt die Anwendung von Software, besonders von informationstheoretischen Methoden, eine nachhaltige und kostengünstigere Ergänzung zu hardwareseitigen Lösungen, z. B. Austausch oder Aufrüstung von Hardware, dar. Die Ausarbeitungen in Kapitel 4 konnten bestätigen, dass der Verbrauch von Bandbreite und Speicherplatz durch den Einsatz von Datenkompression und Anomalieerkennung in Datenflüssen auf industriellen Endgeräten reduziert werden kann. Während die Datenkompression die Datenmenge

durch Reduktion von Irrelevanzen und Redundanzen verringert, dient die Anomalieerkennung der Extraktion der relevanten Informationen. Findet sie nahe der Datenerfassung statt, kann zusätzlich der Verlust von zeitkritischen Informationen, die durch zu große Latenzen in der Analyse ihre Relevanz verlieren, verhindert werden. Ein kombinierter Modelleinsatz kann zu Effizienzsteigerungen führen.

- FF2: Können Ressourcenengpässe in datenflussverarbeitenden Systemen am Netzwerkrand mittels ML verhindert werden?

  Ausgewählte Methoden des ML haben sich auch auf ressourcenlimitierten IIoT-Geräten als anwendbar erwiesen. Dabei wurden ML-Algorithmen sowohl auf Logik-Ebene zur Datenflussverarbeitung als auch auf der Middleware-Ebene zum Ermöglichen der Datenflussverarbeitung eingesetzt und evaluiert. Diese Möglichkeiten sind sowohl in der Weiterentwicklung der Endgeräte (De Blasi und Engels 2020) als auch der Programmiersprachen (*TensorFlow Lite* 2022) begründet. ML-Algorithmen sind dabei nicht als Ersatz, aber als Ergänzung zu bestehenden Verfahren anzusehen. Beispielsweise kann auch ein sehr einfaches klassisches Verfahren wie die Quantisierung zielführend sein.

- FF3: Gibt es eine geeignete Technologie, die die Unveränderlichkeit von Daten und Informationen dezentral im IIoT absichern und Vertrauen schaffen kann?

  Die Untersuchung der neuesten Versionen der DLT IOTA hat ergeben, dass sie die Anforderungen, die das ressourcenlimitierte IIoT stellt, erfüllen können und geeignete Möglichkeiten bieten, dezentral Entscheidungen zu treffen und Vertrauen in dem zunehmend anonymen, digitalen I4.0-Umfeld zu schaffen. Dennoch bedeutet der Einsatz von DLT im IIoT einen gewissen Mehraufwand hinsichtlich Kommunikations-, Rechen- und Speicherressourcen, welcher durch die Nutzen der entsprechenden Anwendungsfälle gerechtfertigt werden muss.

- FF4: Ist es möglich, bei der Verarbeitung von Datenflüssen im IIoT auf Cloud-Dienste zu verzichten?

  Die Frage ist nicht pauschal zu beantworten, da die Verarbeitung von mehreren Faktoren abhängt. Zu beantwortende Fragen sind beispielsweise:

  – Was soll berechnet werden?
  – Wie stark sind die Ressourcen der Endgeräte bereits durch die Kernaufgabe erschöpft?
  – Welche Zykluszeiten sind bei der Verarbeitung der Datenflüsse erfordert?
  – Sind Datenverluste und Latenzen zulässig und in welchem Rahmen?

  Insgesamt lässt sich sagen, dass einige bestätigende Analysen auf den Endgeräten selbst ausgeführt werden können und, aufgrund der genannten Vorteile, auch sollten. Lokale Server (sogenanntes Fog Computing) können ebenfalls Bestandteil des IIoT sein und die Rechenkapazitäten erweitern, sodass auch Möglichkeiten für die rechenintensiven explorativen Datenanalysen, Trainingsprozesse

oder die Optimierung von Hyperparametern bestehen, für die es andernfalls Cloud-Anbindungen bedarf. Es ist zu erwarten, dass es zukünftig ein Zusammenspiel aus Datenverarbeitung am Netzwerkrand (Edge) und in den internen (Fog), aber auch externen Rechenzentren (Cloud) gibt. Daraus ergibt sich eine neue Fragestellung, die es zu untersuchen gilt: *Welche Datenanalysen sind wo auszuführen?*.

Beantwortung der zentralen Forschungsfrage

Ist die verteilte Verarbeitung von Datenflüssen auf den Endgeräten im industriellen Internet der Dinge unter Berücksichtigung der Wirtschaftlichkeit realisierbar und wie kann eine mögliche Umsetzung gestaltet sein?

Die Ausarbeitungen (Kapitel 4-6) zeigen, dass sich die zentrale Forschungsfrage prinzipiell mit „Ja“ beantworten lässt. Die konkrete Umsetzung ist jedoch von den Anforderungen und Randbedingungen des jeweiligen Anwendungsfalls abhängig. Für die verteilte Verarbeitung von Datenflüssen auf industriellen Endgeräten des IIoT sind verschiedene Aspekte zu berücksichtigen. Eine Übersicht dazu ist Abschnitt 3.3 zu entnehmen. Hierbei haben sich in erster Linie die begrenzten Rechen- aber auch Kommunikationsressourcen sowie die zeitlichen Anforderungen an die Verarbeitung als Herausforderung erwiesen. Leichtgewichtige Algorithmen zur Datenflussverarbeitung mit dem Ziel der Datenkompression und Anomalieerkennung können Ressourcenengpässe sowie Informationsverluste reduzieren (siehe FF1). Die Wirtschaftlichkeit wird in den vorgestellten Ausarbeitungen berücksichtigt, indem als Randbedingung die Ausführung der Algorithmen auf der bestehenden Infrastruktur gestellt ist. Ein Nachrüsten oder Ausbau von Endgeräten oder des Netzwerks ist nicht vorgesehen. Um die Ausführung zusätzlicher Rechenaufträge zu ermöglichen, ist aus diesem Grund die optimale Ausnutzung der begrenzten Ressourcen unabdingbar. Ein möglicher Lösungsansatz stellt der in Kapitel 6 beschriebene intelligente Multi-Agenten Ansatz dar.

7.4 Wissenschaftlicher Beitrag der Arbeit

Die Optimierung von industriellen Datenflüssen ist ein Gebiet von hohem Neuheitsgrad, welches durch die aktuellen Entwicklungen von I4.0 und IIoT sowie Datenverarbeitung am Netzwerkrand an Bedeutung gewinnt. Es ist keine existierende Arbeit zu der umfassenden Betrachtung von Optimierungspotentialen in industriellen Datenflüssen bekannt. Die vorliegende Arbeit zeigt die vielfältigen Ansatzpunkte für Optimierungen sowie mögliche Lösungen dafür. Die Teillösungen werden in die Architektur von Datenfluss-Systemen eingeordnet und die Anwendung in einem Gesamtsystem der Datenflussoptimierung in Form einer Machbarkeitsstudie belegt. Es gibt eine enorme Anzahl an Arbeiten, die einzelne Aspekte bereits betrachten. Das Zusammenwirken und gegenseitige Beeinflussen von Datenreduktion, Anomalieerkennung, Kommunikation und der Bestätigung einer Diensterbringung, insbesondere

unter Berücksichtigung der für die I4.0 relevanten Anforderungen, ist dagegen neu. Des Weiteren stellen die aufgezeigten Optimierungen einen wesentlichen Beitrag für die Weiterentwicklung der DSPS dar. Während die aktuellen Arbeiten zur Entwicklung einer vierten Generation von DSPS zwar den kombinierten Einsatz von Edge und Cloud Ressourcen aufzeigen, legt diese Arbeit bereits die Grundlagen für die Verlagerung der überwiegenden Verarbeitung auf die Endgeräte am Netzwerkrand und somit für eine fünfte Generation von DSPS.

In Zusammenhang mit der Untersuchung der Methoden zur Datenflussoptimierung sind weiterhin die folgenden Ausarbeitungen und Weiterentwicklungen entstanden, in denen jeweils ein eigener wissenschaftlicher Beitrag erbracht wird.

1. Optimierungen in der Datenverarbeitung
    - Für die Datenreduktion auf Endgeräten mit begrenzten Ressourcen werden zwei bestehende Methoden des ML, konkret CBN-VAE (nach Liu, Chen, Yan u. a. 2019) und LSTM-Autoencoder (nach Wong und Luo 2018), verglichen und für den Einsatz entsprechend der Anforderungen erweitert (Rosenberger, Kübel und Rothfuß 2022). Der zu besseren Ergebnissen führende LSTM-Autoencoder wurde weiterhin auf einem industriellen Endgerät evaluiert und mit klassischen Verfahren verglichen.
    - Als eine Methode zur Optimierung der Datenverarbeitung wird ein neues Verfahren *EEM-KDE* für die Erkennung verschiedener Anomaliearten vorgestellt, welches auf Datenflüsse anwendbar und für die Ausführung auf Endgeräten mit begrenzten Ressourcen geeignet ist (Rosenberger, Müller u. a. 2022).
    - Des Weiteren wird für eine effizientere Ressourcenauslastung bzw. Verbesserung der Qualität der Ansatz des kombinierten Modelleinsatzes für sowohl Anomalieerkennung als auch Datenkompression vorgeschlagen und die exemplarische Umsetzung für verschiedene Verfahren beschrieben (Rosenberger, Bühren und Schramm 2021).

2. Optimierung hinsichtlich Absicherung von Daten und Zugriffsrechten
    - Der Einsatz der DLT IOTA sowie SC wird anhand von zwei relevanten Anwendungsfällen evaluiert. Die zwei aktuellsten Versionen IOTA Chrysalis und IOTA Coordicide wurden erst kürzlich veröffentlicht, sodass die im Rahmen dieser Arbeit durchgeführte Evaluation die erste experimentelle Bewertung des Einsatzes im industriellen Umfeld darstellt (Rosenberger, Rauterberg und Schramm 2021).

3. Optimierung der Ressourcenausnutzung
    - Es wird eine Methode aufgezeigt, die eine optimierte Auslastung der begrenzten Ressourcen mittels MARL ermöglicht und auf den Endgeräten des IIoT ausführbar ist (Rosenberger, Urlaub, Rauterberg u. a. 2022).

- Es wird eine Methode basierend auf zwei interagierenden, kooperierenden MASs vorgestellt.
- Das vorgestellte System dient der gleichzeitig optimierten Allokation von sowohl Rechen- als auch Kommunikationsressourcen.
- Das vorgestellte System ist für den Einsatz in Netzwerken mit dynamischen Veränderungen wie variierender Anzahl an Teilnehmern und Rechenaufträge geeignet.

4. Simulation für verteilte Systeme und Berechnungen am Netzwerkrand
   - In dem Artikel (Rosenberger, Selig, Ristic u. a. 2023) wird eine Methode vorgestellt, Simulation auf Berechnungen am Netzwerkrand und verteilte Systeme in der I4.0 anzuwenden. Der Hardware- bzw. Software-in-the-Loop-Ansatz beschränkt sich nicht auf einen einzelnen Algorithmus auf einem einzelnen Endgerät, sondern auf das komplexe Zusammenspiel in einem System, das in einer industriellen Umgebung läuft. Der Einsatz der Simulation aus Sicht der operativen Technologie zur realitätsnahen und kosteneffizienten virtuellen Inbetriebnahme, Validierung und Verifikation von IT-Anwendungen im IIoT wird anhand verschiedener Anwendungsfälle erläutert und bewertet. Diese Methode wird in dieser Arbeit für die Betrachtung des Gesamtsystems eingesetzt. Aufgrund von nicht vernachlässigbaren Risiken bei der Einführung neuer Algorithmen in eine laufende Produktionsumgebung kann diese Methode künftig einen hohen praktischen Nutzen stiften.

7.5 Ausblick und kritische Würdigung

Die in dieser Arbeit vorgestellten Optimierungen können zukünftig in realen Produktionsanlagen Einsatz finden.

- Datenkompression: Die Ausarbeitungen zu den klassischen Verfahren fpzip, DWT und Quantisierung können direkt auf geeigneten industriellen Endgeräten zur Kompression von Datenflüssen zur Ressourceneinsparung wie Bandbreite oder Speicherplatz ausgeführt werden. Der vorgestellte LSTM-Autoencoder erfordert vor dem Einsatz zusätzlich ein Training mit geeigneten Daten.
- Anomalieerkennung: Das EEM-KDE Verfahren kann auf industriellen Endgeräten in realen Produktionssystemen für die Erkennung von verschiedenen Anomaliearten in Echtzeit angewendet werden und damit zu einer Zustandsüberwachung beitragen. Das Verfahren ist selbstlernend und erfordert nur wenige durch den Anwender festzulegende Parameter. Für Kontextanomalien ist leistungsfähige Hardware erforderlich und es existieren für den mehrdimensionalen Raum effizientere Verfahren.

- Die in Kapitel 5 aufgebauten IOTA 1.5 und 2.0 Netzwerke sowie das für SC erforderliche Wasp-Netzwerk sind ohne Anpassungen in eine reale Produktionsumgebung zu übertragen. Je nach Anwendungsfall ist zu entscheiden, welche Daten als relevant eingestuft werden, um sie langfristig unveränderlich in die dezentrale Datenbank abzuspeichern. Der konkrete Inhalt der automatisiert ausführbaren, digitalen Verträge (SC) ist ebenfalls vom jeweiligen Anwendungsfall abhängig.
- Die MAS wurden generisch trainiert und sind somit allgemeingültig in unterschiedlichen Anlagen einsetzbar. Bei dem Einsatz in einer konkreten Anlage ist die Kommunikation zwischen den Agenten sowie das Starten einer Datenverarbeitungsaufgabe zu beachten, da diese Anlagen- bzw. Endgeräte-spezifisch sind. Eine zu klärende Fragestellungen ist beispielsweise, wie ein neues Endgerät als neuer Nachbar eines bestehenden Agenten bekanntgemacht wird.

  Weiterhin ist die Übertragbarkeit auf analoge Problemstellungen in Quelle-Senke-Systemen, wie beispielsweise dem Energiemanagement, zu untersuchen.

Die vorangegangenen Ausarbeitungen zeigen, wie das IIoT und die Daten selbst erst durch das Zusammenspiel verschiedener neuen Technologien ihr volles Potential entfalten können. In dieser Arbeit kommen dabei insbesondere ML, DLT, MAS, Virtualisierung und Simulation zum Tragen. Weiterhin sind aber auch die Verwaltungsschale und Digitale Zwillinge, Kommunikationstechnologien sowie Safety- und Security-Technologien eng mit den Daten und dem IIoT verknüpft.

Die bereits erzielten Optimierungen und Ergebnisse können unter Beachtung der nachfolgenden Aspekte weiter ausgebaut werden.

Im Rahmen dieser Arbeit wurden die Optimierung auf vertikale Datenflüsse eingegrenzt. Die Anwendbarkeit der ausgearbeiteten Methoden ist perspektivisch auch für horizontale Datenflüsse zu prüfen. Aufgrund der Unterschiede zwischen vertikalen und horizontalen Daten ist zu erwarten, dass nicht alle Methoden der vertikalen Datenflüsse direkt auf horizontale Datenflüsse übertragbar sind. Gleichzeitig können sich aber bei der Betrachtung der horizontalen Datenflüsse weitere neue Optimierungspotentiale ergeben. Für die Methoden zur Datenreduktion und Informationsextraktion ist zu untersuchen, ob sie neben Zeitreihendaten auch für weitere Eingabedatentypen mit Relevanz für industrielle Applikationen, wie Bilddateien oder Frequenzmessungen, anwendbar sind oder gegebenenfalls analoge Verfahren entwickelt werden können.

Hinsichtlich der Evaluation des Einsatzes von DLT ist bei IOTA Chrysalis darauf hinzuweisen, dass diese lediglich als Übergangsversion angedacht ist. Die Evaluation der Permanode-Funktionalität war zu dem Zeitpunkt der Durchführung der Experimente noch nicht in IOTA Coordicide verfügbar, weshalb zunächst IOTA Chrysalis eingesetzt wurde. Die Umstellung des Hauptnetzes auf IOTA Coordicide ist weiterzuverfolgen und die Permanode-Funktionalität in der aktuellsten Version zu evaluieren. Außerdem bedarf es weiterer Untersuchungen zu der Ausführung der SkyllaDB auf ressourcenlimitierten Endgeräten (siehe Abschnitt 5.6).

Insbesondere für die Ressourcenallokation mittels MAS werden noch Optimierungspotenziale gesehen, die weiter zu untersuchen sind. Für das MAS2 ist der Beobachtungsraum um die Priorität des Rechenauftrags zu erweitern. Dadurch kann die Weiterleitung von Daten schneller erfolgen und z. B. zeitkritische Informationen schneller extrahiert werden. Ebenfalls ist die Übergabe von Zugriffsberechtigungen als Erweiterung zu berücksichtigen, sodass die Weiterleitung von Datenflüssen nur an berechtigte Nachbar-Endgeräte erfolgt. Für MAS1 kann der Beobachtungsraum um die Auslastung von RAM und ggf. Festplattenspeicher erweitert werden. Wie bereits in Abschnitt 6.6 beschrieben, ist zu erwarten, dass die Vergabe einer globalen Belohnungsfunktion das Lernverhalten der Agenten verbessern kann. Ein weiterer bisher unberücksichtigter Ansatz ist die Adaptivität auf Basis der verfügbaren Ressourcen in und zwischen den Endgeräten. Die Algorithmen könnten so gestaltet werden, dass in Abhängigkeit der verfügbaren Ressourcen, die Algorithmen unterschiedlich rechenintensiv ausgeführt werden. Ein Beispiel ist die Anzahl der Kerne bei dem KDE-Verfahren. Je höher die Kernanzahl gewählt wird, desto exakter beschreibt die erlernte Dichteverteilung die reale Datenlage, aber desto rechenintensiver ist die Anomalieerkennung mittels KDE. Auch diese Wahl der Rechenintensität könnte durch das MAS geschehen.

Weitere Verbesserungen der Algorithmen des ML, wie sie in der Anomalieerkennung, der Datenkompression und dem MAS anzutreffen sind, können durch eine geeignete Optimierung der Hyperparameter erzielt werden. Die Hyperparameterwerte in dieser Arbeit wurden manuell bestimmt. Eine große Anzahl an Literatur bestätigt den Nutzen von Algorithmen der automatisierten Hyperparametersuche. Automatisierte Verfahren von verhältnismäßig geringer Komplexität wie die Rastersuche und Zufällige Suche können bereits Verbesserungen erzielen. Für Optimierungen im kontinuierlichen Raum sind beispielsweise kontinuierliche genetische Algorithmen, die den evolutionären Algorithmen zugeordnet werden, geeignete Verfahren.

Für die beschriebenen Ausarbeitungen in Kapitel 4 und 6 wird die Programmiersprache Python genutzt. Um die Geschwindigkeit zu erhöhen, sollten andere Sprachen, beispielsweise Julia oder Rust, betrachtet werden. Außerdem hat die Wahl der Programmiersprache, neben dem Betriebssystem selbst, ebenfalls die Einfluss auf die Einhaltung der Echtzeitfähigkeit. Hierfür sind beispielsweise C, C++ oder Real-Time Java zu wählen. Zudem hat die Wahl der Datenformate Einfluss die Ergebnisse und auch die statische Allokation von Speicher ist noch zu untersuchen.

Weiterhin besteht Bedarf an vertiefenden Experimenten. Die Experimente zu der Anomalieerkennung und der Datenkompression wurden auf einem repräsentativen IIoT-Endgerät, einer Steuereinheit, durchgeführt. Generell sind für alle Ausarbeitungen Experimente in einer realen Anlage, über einen längeren Zeitraum und mit höher variablen Daten, auf heterogenen Endgeräten (auch außerhalb der Steuerungsebene) wünschenswert.

Die Experimente zu dem IOTA und den MARL Systemen erfordern zur Evaluation ganze IIoT-Netzwerke. Im Rahmen dieser Arbeit wurde ein IIoT-Netzwerk, bestehend aus virtuellen bzw. realen Steuereinheiten, künstlich aufgebaut. Näher an realen Anwendungen wäre ein größeres System mit einer Vielzahl heterogener

Endgeräte mit unterschiedlichen Betriebssystemen und Rechenressourcen. Die Betrachtung des Gesamtsystems erfolgte anhand einer simulierten industriellen Anlage. Eine Betrachtung des Zusammenspiels in einer realen Anlage ist bisher noch nicht erfolgt.

Die entstandenen Ausarbeitungen können auch in weiteren Szenarien außerhalb der reinen Datenflussoptimierung in der I4.0 Einsatz finden. Unter Berücksichtigung der aktuellen Entwicklungen hinsichtlich zunehmender Virtualisierung von Endgeräten ergibt sich die Frage, welche Algorithmen auf einem realen Endgerät ausgeführt werden müssen und welche auf die virtualisierten Instanzen verlagert werden können. Das in dieser Arbeit beschriebene MAS kann für diese Problemstellung adaptiert werden. Hierbei ist auch die gleichmäßige Lastverteilung eine relevante Aufgabe. Unter Anpassung der Belohnungsfunktion könnte das beschriebene MAS1 auch hierfür eingesetzt werden.

Kürzliche Entwicklungen in Richtung virtualisierte industrielle Endgeräte, beispielsweise virtualisierte Industriesteuerungen, erfordern eine Neubetrachtung des Einsatzes der Datenflussoptimierung. Bestehende Begrenzungen hinsichtlich Rechenressourcen könnten durch virtualisierte, skalierende Lösungen behoben werden, wogegen die Datenübertragung noch zusätzlich gefordert wird. Andererseits bestehen die beschriebenen Nachteile der Cloud weiterhin und gelten auch, bzw. insbesondere, für virtualisierte Endgeräte, da diese ein besonderes Sicherheitsrisiko darstellen.

Die vorgestellten Ausarbeitungen stellen einen wesentlichen Beitrag für die Weiterentwicklung der DSPS hin zu einer fünften Generation dar, da sie auf der Logik- und Middleware-Ebene Grundlagen für die Verlagerung der Datenflussverarbeitung an den Netzwerkrand in die Nähe der Datenerfassung legen. Dennoch sind für den Einsatz einer fünften Generation von DSPS weitere Forschungen notwendig, da wesentliche Funktionalitäten von DSPS, beispielsweise die Umsetzung des Zustandsmanagements, einheitliches Datenflussmanagement, etc. nicht Inhalt dieser Arbeit sind.

ANHANG A

Anhang

A.1 Gegenüberstellung der Autoencoder-Modelle

Die Tabellen A.1 und A.2 geben eine Übersicht über die Ergebnisse der experimentellen Evaluation der Autoencoder-Modelle CBN-VAE und LSTM-Autoencoder. Tabelle A.1 zeigt den Ressourcenverbrauch bei ein- und mehrdimensionaler Datenflusskompression. Tabelle A.2 beinhaltet die Ergebnisse der Kompressionsleistung der beiden Autoencoder sowie einen Vergleich durch einen Qualitätswert.

Tabelle A.1: Gegenüberstellung des Ressourcenbedarfs

Modell	**FLOPS**	**Speicher [B]**	**Parameter**	$\mathbf{t_{enc}}$ **[ms]**	$\mathbf{t_{dec}}$ **[ms]**
Dimension $m = 1$, Batch Größe $b = 1$					
CBN-VAE	10.299.640	37.428	5237	44	236
LSTM-RNN	2.700	260	24	622	1092
Dimension $m = 4$, Batch Größe $b = 1$					
eCBN-VAE	15.837.880	91.860	6.965	67	321
LSTM-RNN	26.037	1.808	288	403	809

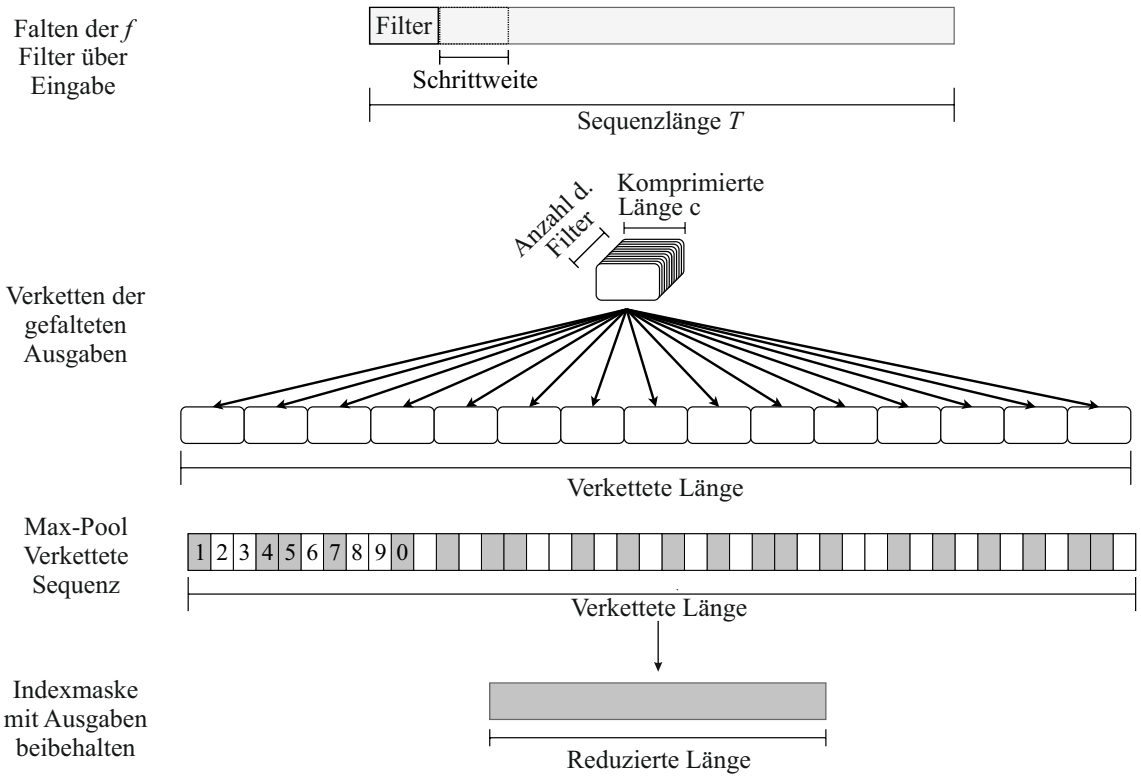


Abbildung A.1: Datenkompression mittels Faltungs- und Max-Pooling-Schichten

Tabelle A.2: Gegenüberstellung der Kompressionsqualität

	UTS (Dimension $m = 1$)			MTS (Dimension $m = 4$)		
Modell	**CR**	**RE**	**QS**	**CR**	**RE**	**QS**
	Intel Datensatz					
DWT	20	45,2	0,44	20	36,9	0,54
LSTM-RNN	20	5,1	7,9	20	10,8	3,7
eCBN-VAE	48	8,8	5,5	192	46,4	4,1
	Distillate flowrate Datensatz ($m = 1$)					
LSTM-RNN	20	18,8	2,1	n.a.	n.a.	n.a.
eCBN-VAE	48	53,9	0,9	n.a.	n.a.	n.a.

CR - Kompressionsrate, RE - Rekonstruktionsfehler, QS - Qualitätswert

A.2 Wahl eines geeigneten Ansatzes zur Anomalieerkennung

Dieser Abschnitt gibt einen Überblick über die Bewertung verschiedener Kategorien zur Anomalieerkennung nach Rosenberger, Müller u. a. 2022. Die Bewertung ist Grundlage für die Wahl des in Abschnitt 4.4 vorgestellten nicht-parametrische statistische Verfahren. Sie erfolgt unter Betrachtung der folgenden Kriterien:

- A-prioiri-Wissen: Kenntnisse über die Datenwerte sind für die zielführende Anwendung der Algorithmen erforderlich.
- Tupelweise Verarbeitung: Die Verarbeitung einzelner Datenpunkte ist möglich. Sie ist nicht auf Datensätze oder Mini-Sequenzen beschränkt.
- Speicherbedarf: Der Speicherbedarf soll gering und möglichst konstant sein.
- Rechenkomplexität: Für die Ausführung auf Endgeräten ist eine geringe Komplexität erforderlich.
- Parametrieraufwand: Die Parametrierung soll möglichst einfach sein.
- Echtzeitfähigkeit: Die Berechnungen müssen deterministisch sein.
- Agnostik: Die Algorithmen sollen unabhängig von dem Zielsystem sein, also Datenflüsse verschiedenster Anlagen oder Maschinen anwendbar sein.
- Adaptivität: Die Algorithmen sollen sich zur Laufzeit an Änderungen in den Datenwerten anpassen können.
- Unüberwachtes Lernen: Die Algorithmen sollen unabhängig von menschlichem Zutun und gelabelten Daten trainiert werden können.

Obwohl statistische Ansätze rechnerisch leichtgewichtig sind (Ahmad, Lavin u. a. 2017), sind sie typischerweise nicht auf hochdimensionale Datensätze anwendbar (Hu u. a. 2020). Außerdem haben sie meist den Nachteil, dass sie a-priori-Wissen erfordern, welches nur selten verfügbar ist (Salehi und Rashidi 2018). Ausnahmen sind nicht-parametrische Methoden wie Histogramme oder Kernel-Funktionsbasierte Modelle, z. B. die Parzen-Fenster-Schätzung (Chandola, Banerjee und Kumar 2009).

Die meisten Clustering-Methoden sind einer Gruppe von Methoden zuzuordnen, die aus einem Online-Teil zur Anomalieerkennung und einem Offline-Teil zum Lernen bestehen (Ahmad, Lavin u. a. 2017; Salehi und Rashidi 2018) und somit die Anforderungen nicht erfüllen. In (Salehi und Rashidi 2018) wird für diese Gruppe nur eine Ausnahme (Chenaghlou u. a. 2018) genannt, die Anomalien in Echtzeit detektiert.

Nächste-Nachbbarn-Methoden können entweder abstandsbasiert oder dichtebasiert sein und gelten als intuitiv und einfach zu implementieren (Salehi und Rashidi 2018), aber die grundlegenden Verfahren haben eine hohe Rechenkomplexität (Chandola, Banerjee und Kumar 2009). Die distanzbasierten Ansätze sind nur in der Lage,

globale Ausreißer zu erkennen und eignen sich nicht für inhomogene Dichten (Salehi und Rashidi 2018). Im Gegensatz dazu erkennen die dichtebasierten Ansätze lokale Ausreißer. Ein bekanntes Beispiel ist der inkrementelle lokale Ausreißerfaktor (iLOF) (Pokrajac, Lazarevic und Latecki 2007). Optimierungen hinsichtlich Speicher und Verarbeitungszeit sind in MiLOF (Salehi, Leckie u. a. 2016) implementiert.

Die Klassifikation basiert meist auf überwachtem Lernen und ist daher in diesem Kontext nicht anwendbar. Es gibt wenige Ausnahmen des unüberwachten Trainings der Klassifikatoren, welchen es allerdings laut (Hu u. a. 2020) häufig an theoretischen Grundlagen mangelt.

Unterraumbasierte Verfahren, oder auch Verfahren der spektralen Dekomposition, basieren auf Verfahren der Dimensionsreduktion und sind nur für mehrdimensionale Daten sinnvoll. Somit sind sie nur für die Erkennung von Kontextanomalien geeignet und stellen kein ausreichendes Verfahren für die Datenflussoptimierung dar.

Informationstheoretische Verfahren erkennen Anomalien durch ihren überdurchschnittlich hohen Informationsgehalt. Die Verfahren überschneiden sich teilweise mit denen der Datenkompression, was einem kombinierten Modelleinsatz entgegenkommt. Die Verfahren gehören den unüberwachten Lernmethoden an und sie agieren unabhängig einer zugrundeliegenden statistischen Verteilung der Daten. Chandola, Banerjee und Kumar 2009

Isolationsbasierte Verfahren sind eine zunehmend relevante Klasse für die Anomalieerkennung, aber typischerweise nicht für Datenflüsse geeignet. Es gibt einen Ansatz für Streaming-Daten (Ding und Fei 2013), der aber auf einem gleitenden Fenster basiert, was die Anforderung an die tupelweise Verarbeitung nicht erfüllt und zu einem erhöhten Speicherbedarf führt.

Eine zusammenfassende Übersicht der Bewertung der unterschiedlichen Ansätze zur Anomalieerkennung ist in Tabelle A.3 gegebenen. Es ist darauf hinzuweisen, dass es einzelne Verfahren geben kann, die in einzelnen Kriterien von der Bewertung der allgemeinen Ansätze abweichen.

Tabelle A.3: Bewertung der Ansätze zur Anomalieerkennung nach (Chandola, Banerjee und Kumar 2009; Hofmockel 2019)

	Statistisch p.[a]	**Statistisch n.-p.**[b]	**Clusterbasiert**	**Nächste-Nachbarn**	**Klassifizierend**	**Unterraumbasiert**	**Informationstheoret.**	**Isolationsverfahren**
A priori Kenntnis notwendig	✓	✗	✗	✗	✓	✗	✗	✗
Tupelweise Verarbeitung	✓	✓	✓	✓	✓	✗	✗	✗
Konstanter Speicherbedarf	✓	✓	✓[c]	✗	✓	✓	✓	✗
Geringe Rechenkomplexität	✓	✓	✗	✗	✓	✗	✓[d]	✓
Geringer Parametrieraufwand	✗	✓	✓	✓	✓	✓	✗	✓
Echtzeitfähig	✓	✓	✗	✗	✓	✓	✓	✗
Agnostisch hinsichtl. industr. Anlage	✓	✓	✓	✓	✓	✓	✓	✓
Adaptiv	✓	✓	✓	✓	✗	✓	✓	✓
Unüberwachtes Lernen	✓	✓	✓	✓	✗	✓	✓	✓

[a] Parametrische statistische Verfahren.
[b] Nicht-parametrische statistische Verfahren.
[c] Sofern die Anzahl der Cluster-Daten konstant bleibt, z.B. bei offline-Training und online-Ausführung (nicht-adaptiv).
[d] Abhängig von dem gewählten Algorithmus.

A.3 Kernfunktionen für Kerndichteschätzer

Tabelle A.4: Beispiele für Kernfunktionen in uni- und multivariater Form nach (Nedden 2012)

Univariate Kernfunktion	**Multivariate Kernfunktion**[3]
Gauß-Kern	
$(\sqrt{2\pi})^{-1}\exp(-0.5x^2)$	$(2\pi)^{-0.5d}\exp(-0.5\|\|x\|\|^2)$
Dreieckskern	
$K(x)=\begin{cases}1-\|x\| & \text{falls } \|x\|<1\\ 0, & \text{sonst}\end{cases}$	$K(x)=\begin{cases}c_d^{-1}(d+1)(1-\|\|x\|\|^2), & \text{falls } \|\|x\|\|<1\\ 0, & \text{sonst}\end{cases}$
Rechteckskern	
$K(x)=\begin{cases}\frac{1}{2} & \text{falls } \|x\|<1\\ 0, & \text{sonst}\end{cases}$	$K(x)=\begin{cases}c_d^{-1}, & \text{falls } \|\|x\|\|<1\\ 0, & \text{sonst}\end{cases}$
Epanechnikov-Kern	
$K(x)=\begin{cases}\frac{3}{4}(1-x^2), & \text{falls } \|x\|<1\\ 0, & \text{sonst}\end{cases}$	$K(x)=\begin{cases}\frac{1}{2}c_d^{-1}(d+2)(1-\|\|x\|\|^2), & \text{falls}\|\|x\|\|<1\\ 0, & \text{sonst}\end{cases}$

[3] Volumen der d-dimensionalen Einheitskugel: $c_d := \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)}$

A.4 Transfer des Vorgehens auf weitere Ansätze zur Anomalieerkennung

Das Vorgehen zur Erweiterung der Anomalieerkennung des Kerndichteschätzers für weitere Anomaliearten nach Abschnitt 4.4 kann auf die anderen Ansätze zur Anomalieerkennung übertragen werden. Die Tabellen A.5-A.8 beschreiben das Vorgehen zum Transfer je Anomalieart auf die einzelnen Ansätze.

Punktanomalien

Die Erkennung von Punktanomalien erfolgt über die bestehenden Grundverfahren.

Schleichende Anomalien

Für die Erkennung von schleichenden Anomalien wird eine Kennzahl eingeführt, welche das Maß der Änderung eines bestimmten Parameters im Grundverfahren abbildet. Die Tabelle A.6 nennt je Ansatz der Anomalieerkennung geeignete Parameter.

Kollektive Anomalien

Für die Erkennung von kollektiven Anomalien ist für das vorgestellte Vorgehen mittels Kerndichteschätzer im ersten Schritt eine Zyklenerkennung erforderlich. Im zweiten Schritt wird die Einordnung der Datenpunkte als kodierte Sequenz abgebildet und mit dem Vorgänger-Zyklus oder einem Musterzyklus verglichen. Die Datenpunkte werden in einem Raster eingeordnet. Die Rasterung soll dabei in Gebieten mit vielen Datenpunkten feiner sein als in Gebieten mit geringer Dichte. Tabelle A.7 stellt eine Übersicht der Parameter, von denen der Grad der Rasterung abhängig gemacht wird, dar.

Kontextanomalien

Die Anomalieerkennung erfolgt mittels mathematischer Verfahren. Eine Punktanomalie stellt eine Anomalie im eindimensionalen Raum dar. Kontextanomalien sind als Anomalien im mehrdimensionalen Raum zu interpretieren, weshalb eine rein mathematische Anpassung des Verfahrens in den mehrdimensionalen Raum vorgenommen wird. Tabelle A.8 zeigt je Ansatz, an welcher Stelle die Dimensionen erweitert wird sowie teilweise konkrete Beispiele für die Umsetzung.

Tabelle A.5: Erkennung von Punktanomalien in Anlehnung an die Beschreibung der Verfahren in Abschnitt 2.5

Ansatz	**Transfer zur Ermittlung von Punktanomalien**
Statistische Verfahren	Erkennung mittels Auftrittswahrscheinlichkeit. Die Wahrscheinlichkeiten sind in einem statistischen Modell hinterlegt. Datenpunkte mit sehr geringen Wahrscheinlichkeiten werden als Anomalie eingestuft.
Clusterbasierte Verfahren	Erkennung mittels Zuordnung zu Clustern. Ist keine Zuordnung zu einem Cluster möglich, das Clusterzentrum zu weit entfernt oder das Cluster sehr klein liegt eine Anomalie vor.
Nächste Nachbarn Verfahren	Erkennung über die Distanz zur den benachbarten Datenpunkten. Ist die Distanz überdurchschnittlich groß, liegt eine Anomalie vor.
Klassifikation	Erkennung über die Möglichkeit Datenpunkte Klassen zuzuordnen. Der (vortrainierte) Klassifizierer ordnet Datenpunkte in Klassen ein. Ist keine Zuordnung möglich, liegt eine Anomalie vor.
Isolationsverfahren	Erkennung durch die Einfachheit der Isolation. Anomalien lassen sich schneller Isolieren als normale Daten.
Unterraum-Verfahren	Vereinfachte Erkennung in einer anderen Darstellung. Transfer in Unterraum verdeutlicht Ausreißer.
Informationstheoretische Verfahren	Erkennung durch Informationsgehalt. Ein überdurchschnittlich hoher Informationsgehalt lässt auf eine Anomalie schließen.

Tabelle A.6: Erkennung von schleichenden Anomalien mittels Einführung von Driftkennzahlen zur Verfolgung langfristiger Bewegung

Ansatz	**Transfer zur Ermittlung von Schleichenden Anomalien**
Statistische Verfahren	Position des Mittelpunkts der Wahrscheinlichkeitsverteilungen. Nachverfolgung der Änderung der Positionsparameter der statistischen Verteilung.
Clusterbasierte Verfahren	Position des Clusterzentrums. Nachverfolgung der Änderung der Position der Clusterzentren.
Nächste Nachbarn Verfahren	Durchschnittliche Distanz. Nachverfolgung der Änderung der durchschnittlichen Distanz zu den nächsten Nachbarn.
Klassifikation	Fehlerdifferenz. Nachverfolgung der Abweichung zwischen Eingabe und Ausgabe bei gleichen Datenpunkten.
Isolationsverfahren	Mittlere Pfadlänge. Nachverfolgung der Änderung der mittleren Pfadlänge.
Unterraum-Verfahren	Bei statischem Modell: zunehmende durchschnittliche Distanz zu Hauptkomponenten.
	Bei dynamischem Modell: langfristige Positionsänderung der Hauptkomponenten.
Informationstheoretische Verfahren	Bei statischem Modell: zunehmender durchschnittlicher Informationsgehalt.

Tabelle A.7: Erkennung von kollektiven Anomalien durch Zyklenerkennung, Einordnung der Datenpunkte und Kodierung

Ansatz	**Transfer zur Ermittlung von Kollektiven Anomalien**
Statistische Verfahren	Flächeninhalt unter der Dichteverteilung.
Clusterbasierte Verfahren	Clusterdichte. Die Clusterdichte ergibt sich aus Größe (z.B. Radius) und Anzahl der Punkte je Cluster.
Nächste Nachbarn Verfahren	Mittlere Distanz. Rasterung entsprechend mittlerer Distanz zu Nachbarn.
Klassifikation	Wertezuordnungsbereich. Rasterung anhand Wertezuordnungsbereich (Klassen)
Isolationsverfahren	Pfadlänge. Die Pfadlänge gibt an, wie häufig ein Gebiet unterteilt werden muss, um die Isolation eines Datenpunktes zu erreichen. Sie ist somit ein direktes Maß über die Dichte in diesem Gebiet.
Unterraum-Verfahren	./.
Informationstheoretische Verfahren	./.

Tabelle A.8: Erkennung von Kontextanomalien durch Anpassung der mathematischen Beschreibung und Übertragung

Ansatz	**Transfer zur Ermittlung von Kontextanomalien**
Statistische Verfahren	Erstellung einer mehrdimensionale Dichteverteilung. Beispiel Kerndichteschätzer: multivariate statt univariate Kerne.
Clusterbasierte Verfahren	Cluster mit mehrdimensionalen Werten.
Nächste Nachbarn Verfahren	Distanz zwischen mehrdimensionalen Daten. Distanz als Vektor im mehrdimensionalen Raum.
Klassifikation	Ein-Klassen-Verfahren: Über Fehlervektorvergleich (ähnlich mehrdimensionalem Clustering). Mehr-Klassen-Verfahren: Abbildung über neue Klassenzuordnung.
Isolationsverfahren	Isolieren der Anomalien im mehrdimensionalen Raum. Die Isolation erfolgt durch das Legen von Ebenen im mehrdimensionalen Raum anstelle von Linien im eindimensionalen Raum. Das Vorgehen zur Erstellung des Isolationsbaums unter Angabe der Pfadlänge bleibt identisch.
Unterraum-Verfahren	Messung des Abstandsvektors zu den Hauptkomponenten im mehrdimensionalen Raum
Informationstheoretische Verfahren	Analog zu Punktanomalien

A.5 Hyperparameterwahl für Multi-Agenten-System

Die Wahl geeigneter Hyperparameterwerte hat großen Einfluss auf die Qualität von ML Modellen. Die folgenden Inhalte zur Wahl der Hyperparameter des MAS1 sind angelehnt an die Veröffentlichung in der Schriftreihe der Technischen Informatik Köln (Urlaub und Rosenberger 2022). Die wesentlichen Ergebnisse werden im Folgenden zusammengefasst.

Sofern nicht anders angegeben, gelten für die durchgeführten Experimente die nachfolgenden Randbedingungen.

- SB3-MlpPolicy
- Gesamtanzahl der Trainingsschritte: 1 000 000
- Trainingsschritte pro Episode: 100
- Lernrate: 0,0007 (A2C) oder 0,0003 (PPO)
- Anzahl der Agenten: 2

1. Experiment: Untersuchung der Belohnungsstrategie

In einem ersten Experiment werden verschiedene Parameterwerte nach Tabelle A.9 für die Belohnungsstratgie entsprechend Gleichung A.1 verglichen.

$$R(s,a) = \begin{cases} \text{Bestrafung}, & \text{wenn } a = 1 \text{ und } s_{t+1} > Schwellwert \\ \text{Bestrafung}, & \text{wenn } a = 0 \text{ und } s_{t+1} < Schwellwert \\ \text{Belohnung}, & \text{sonst} \end{cases} \tag{A.1}$$

Tabelle A.9: Werte der Belohnungsfunktionen

	$R_1(s,a)$	$R_2(s,a)$	$R_3(s,a)$
$a = 1$ & $s > 0,8$	-10	-10	-1
$a = 0$ & $s < 0,8$	-1	-10	-1
sonst	1	1	1

Abbildung A.2 zeigt die Ergebnisse der drei unterschiedlich parametrierten Funktionen. Es werden jeweils die CPU-Auslastungen von zwei durch Agenten verwaltete Maschinen (abstrahierte Endgeräte in der Trainingsumgebung) am Ende jeder Episode dargestellt. Bei jeder der drei Parametrierungen werden die Agenten zur Ausführung von Rechenaufträgen motiviert. Bei keiner der gewählten Parametrierungen wird der vorgegebene Grenzwert der Auslastung durchgehend eingehalten. Während das Endgerät des ersten Agenten häufig überlastet ist, schwankt die Auslastung des Endgeräts von Agent zwei um den 80 % Wert und erreicht seltener die 100 % Auslastung. Mit der Belohnungsfunktion R_1 schwankt die Auslastung sehr stark und die Agenten weisen entweder keine Rechenaufträge zu oder überlasten die

Maschinen. Die Belohnungsfunktionen R_2 und R_3 erzielen bessere Ergebnisse und führen nur selten dazu, dass keine Rechenaufträge ausgeführt werden. zu einem ähnlichen Verhalten. Da R_2 etwas bessere Resultate als R_3 erzielt, wird für die weiteren Experimente die Belohnungsfunktion R_2 festgelegt.

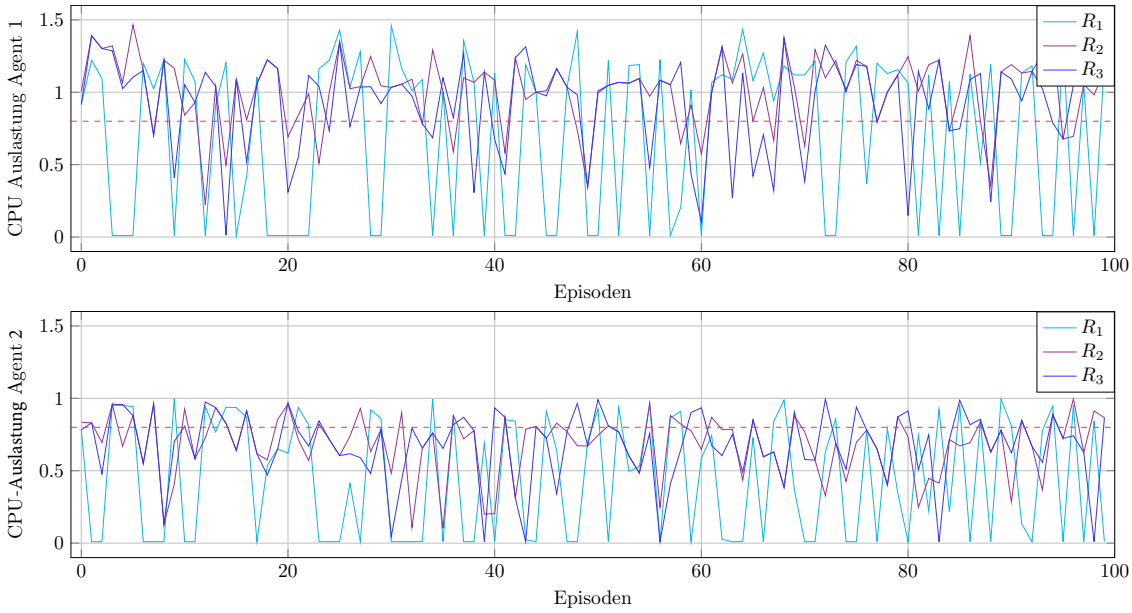


Abbildung A.2: Endauslastungen Episode für verschiedene Belohnungsfunktionen (Urlaub und Rosenberger 2022)

2. Experiment: Untersuchung des Lernalgorithmus

Das zweite Experiment dient dem Vergleich von zwei verschiedenen Lernalgorithmen, PPO und A2C. Er wird mit der Belohnungsfunktion R_2 des ersten Experiments trainiert. Abbildung A.3 zeigt die Ergebnisverläufe, d.h. die CPU-Auslastungen am Ende jeder Episode, jeweils für PPO bzw. A2C.

Es ist deutlich ersichtlich, dass der mit A2C trainierte Agent 1 regelmäßig seine Maschinen überlastet und die Maschinen von Agent 2 oft unterlastet bleiben, d. h. er werden weniger Rechenaufträge als möglich ausgeführt. Mit dem Lernalgorithmus PPO dagegen, schwankt die Endauslastung sowohl bei Agent 1 als auch Agent 2 um den Zielwert und ist somit der bevorzugte Algorithmus für das MAS. Die geringeren Schwankungen bei PPO sind auf das zugrundeliegende Gradientenverfahren zurückzuführen.

3. Experiment: Untersuchung der Episodenlänge

Neben dem Lernalgorithmus hat auch die Episodenlänge Einfluss auf die Trainingsergebnisse. Abbildung A.4 zeigt die Ergebnisse für das Training mit Episodenlängen von 10, 100 bzw. 200 Trainingsschritten je Episode.

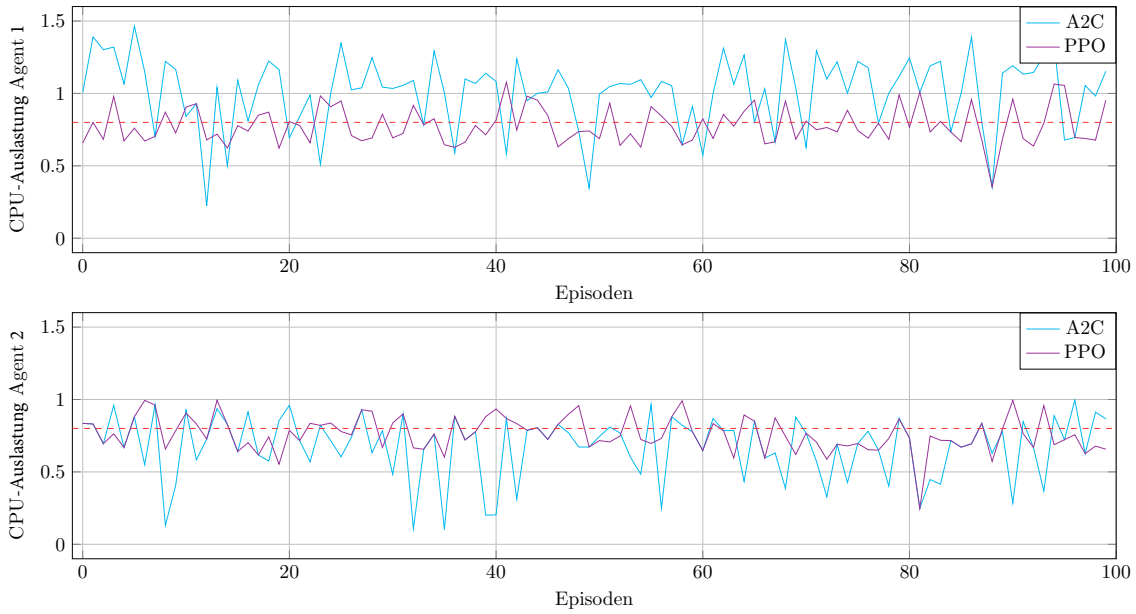


Abbildung A.3: Endauslastungen jeder Episode für verschiedene Lernalgorithmen (Urlaub und Rosenberger 2022)

Das Training mit Episoden von 10 Trainingsschritten führt zu starker Überlastung (Agent 1) bzw. häufigerer Unterlastung (Agent 2). Die Anzahl der Trainingsschritte ist nicht ausreichend, um das gewünschte Verhalten zu erlernen. Die Ergebnisse für 100 bzw. 200 Trainingsschritte je Episode führen dagegen zu Endauslastungen, die sich um den Grenzwert einpendeln. Da die Ergebnisse für 100 bzw. 200 Trainingsschritte ähnlich positiv ausfallen, wird aus Effizienzgründen für die weiteren Experimente eine Episodenlänge von 100 genutzt.

4. Experiment: Untersuchung der Trainingslänge

Nach der Festlegung einer geeigneten Episodenlänge wird in dem vierten Experiment die gesamte Trainingslänge untersucht.

$$\text{Trainingslänge} = \text{Anzahl der Episoden} \cdot \text{Episodenlänge}$$

In dem Experiment wird das Training mit 1 Millionen, 2 Millionen und 4 Millionen Trainingsschritten durchgeführt und anschließend über 100 Episoden evaluiert.

Wie in Abbildung A.5 ersichtlich, liegen die Ergebnisse für die unterschiedlichen Trainingslängen nahe beieinander. Das Training mit 1 Millionen Trainingsschritten führt in der Evaluation zu etwas größeren Schwankungen und in zwei Fällen auch zur Überlastung. Die Ergebnisse für das Training mit 2 bzw. 4 Millionen Trainingsschritten sind sich sehr ähnlich, weshalb im Folgenden das Training mit 2 Millionen Schritten durchgeführt wird.

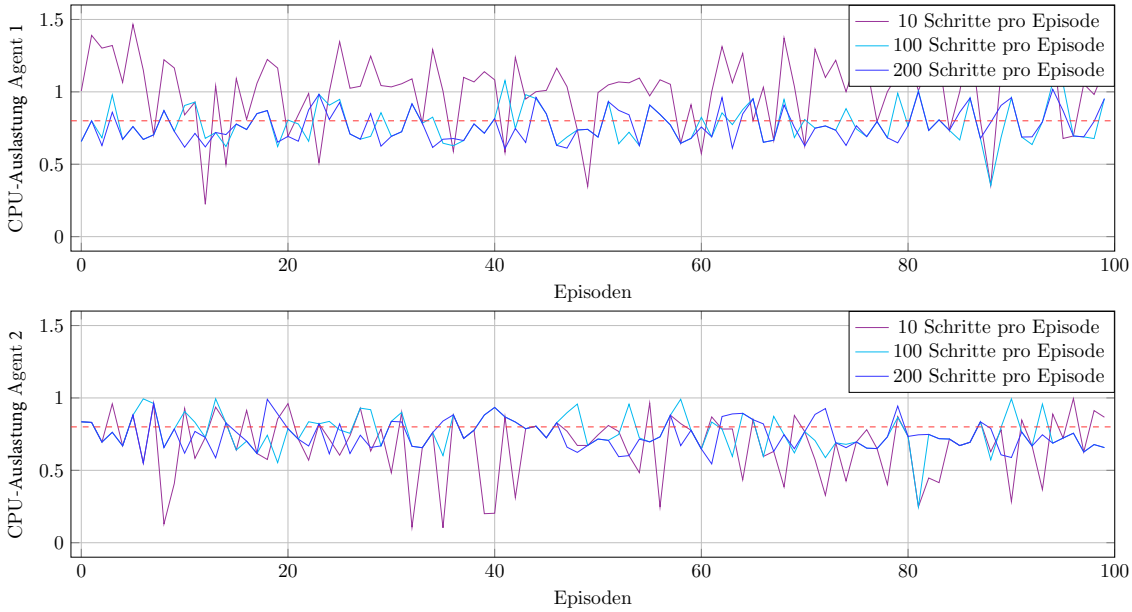


Abbildung A.4: Endauslastungen jeder Episode für verschiedene Episodenlängen (Urlaub und Rosenberger 2022)

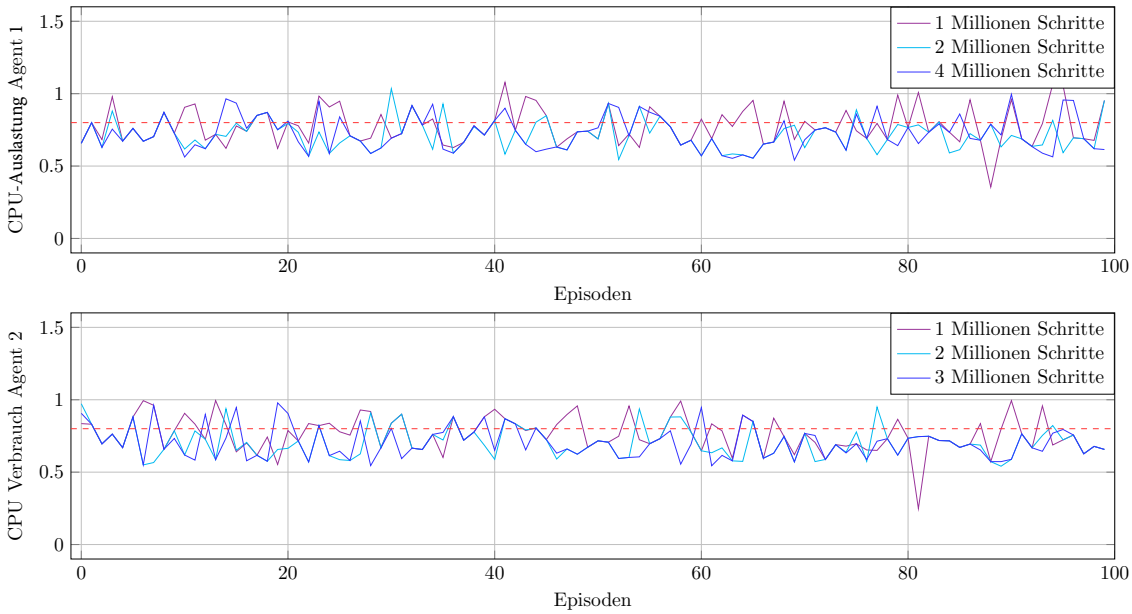


Abbildung A.5: Endauslastungen jeder Episode für verschiedene Trainingslängen (Urlaub und Rosenberger 2022)

5. Experiment: Untersuchung der Lernraten

In den vorherigen Experimenten wurde die in SB3 voreingestellte Lernrate für PPO genutzt. In dem fünften Experiment werden drei unterschiedliche Lernraten (0,003, 0,0003 und 0,00003) miteinander verglichen. Das Training erfolgt entsprechend der vorangegangenen Experimente mit 2 Millionen Trainingschritten und einer Episodenlänge von 100 Schritten sowie dem PPO-Lernalgorithmus.

Die Ergebnisse in Abbildung A.6 zeigen eindeutig, dass die Lernrate von 0,003 nicht zielführend ist, da die Agenten gar keine Rechenaufträge ausführen. Die Lernrate von 0,00003 führt insgesamt zu höheren Auslastungen als eine Lernrate von 0,0003. Bei Agent 1 ist das Endgerät dadurch regelmäßig überlastet. Mit der Lernrate von 0,0003 können die Agenten im gleichen Trainingszeitraum die beste Strategie erlernen. Sie führt zu einer durchschnittlichen Auslastung des Endgeräts knapp unter dem 80 % Grenzwert.

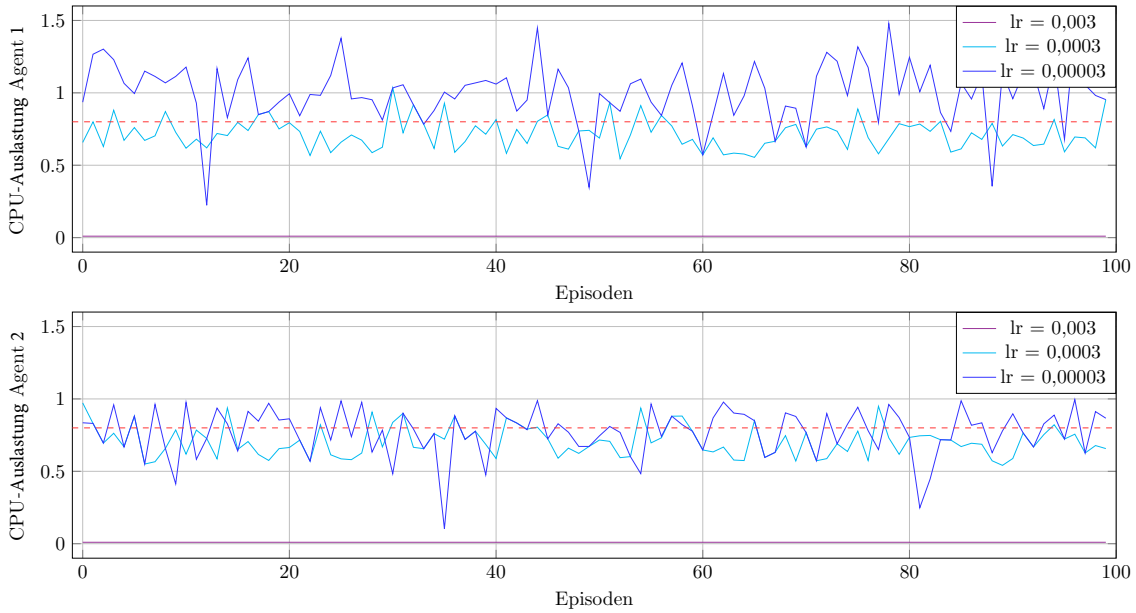


Abbildung A.6: Endauslastungen jeder Episode für verschiedene Lernraten (Urlaub und Rosenberger 2022)

Nach den fünf beschriebenen Experimenten zur Hyperparameter-Wahl zeigt Abbildung A.7 das erlernte Verhalten der Agenten. Die Auslastung am Ende der Episoden (oben) ist durchgehend in der Nähe des vorgegebenen Grenzwerts und nur selten im Bereich der maximalen Auslastung. Der untere Plot zeigt den Verlauf einer einzelnen Episode über 100 Trainingsschritte und bestätigt die erlernte Strategie der Agenten. Es erfolgt die zügige Allokation von freien Ressourcen für Rechenaufträge, die bei Erreichen des vorgegebenen Schwellwerts endet.

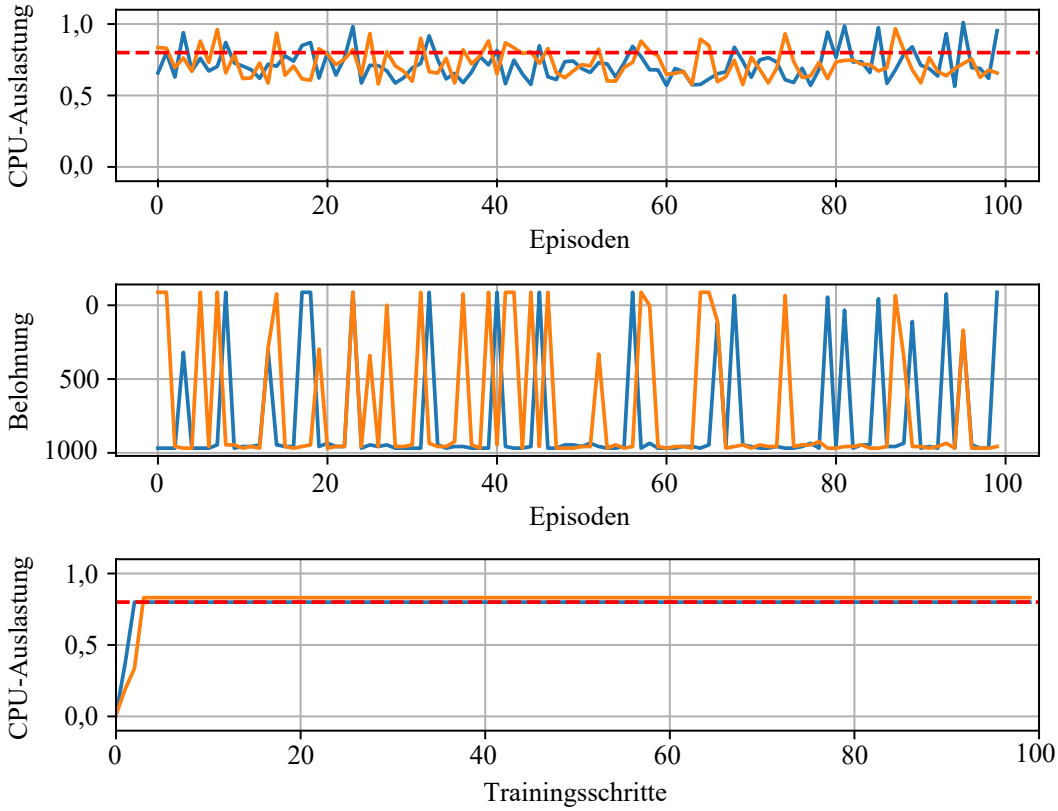


Abbildung A.7: Trainingsergebnis nach manueller Hyperparameteroptimierung (Urlaub und Rosenberger 2022)

A.6 Evaluation des Gesamtsystems

Tabelle A.10 gibt eine Übersicht über die Datenflüsse, die für die Evaluation der Datenflussoptimierungen ausgewählt wurden. Die dazugehörigen Steuerungsskripte, die Konfigurationsdateien für die Kommunikation zwischen Simulationsmodell und Steuereinheiten sowie das Simulationsmodell selbst sind der GitHub Ablage unter `https://github.com/JuliRosenberger/iPhysicsEvaluation` zu entnehmen. Die Inhalte basieren auf dem bereits veröffentlichten Artikel (Rosenberger, Selig, Ristic u. a. 2023). Die Abbildungen A.8-A.11 zeigen Ergebnisse der Experimente.

Tabelle A.10: Ausgewählte, in der Simulationsumgebung erzeugte Datenflüsse (Rosenberger, Selig, Ristic u. a. 2023)

Datenflussvariable	Beschreibung	Datentyp	Bereich
Ausgewählte Datenflüsse der realen Steuereinheit 1 (192.168.8.102)			
Pusher_AXIS	Position des Schiebers	float32	0 bis 0.25
Ausgewählte Datenflüsse der realen Steuereinheit 2 (192.168.8.103)			
Gleitschiene_Links_Axis	Position der Gleitschiene (links)	float32	0 bis 0.025
EndOfLaufwagen_ACT_SPD	Aktuelle Geschwindigkeit des Laufwagens	float32	-0.8 bis 0.8
EndOfLaufwagen_Axis	Position des Laufwagens	float32	0 bis 1.5
EndOfLaufwagen_DLT_POS	Delta zwischen Ist- und Sollwert der Laufwagenposition	float32	0 bis 1.5
Ausgewählte Datenflüsse der realen Steuereinheit 3 (192.168.8.104)			
Drehschieber_Achse_101_AXIS	Position des Drehschiebers	float32	0 bis 0.13
Ausgewählte Datenflüsse der virtualisierten Steuereinheit 4 (192.168.8.83)			
EndoflHubgeruest01_ACT_SPD	Aktuelle Geschwindigkeit des Portals	float32	max. 1.4
EndoflHubgeruest01_AXIS	Position des Portals	float32	-2 bis 0
EndoflHubgeruest01_DLT_POS	Delta zwischen Ist- und Sollwert der Portalposition	float32	-2 bis 0
EndoflGreifarm01_ACT_SPD	Aktuelle Geschwindigkeit des Greifarms	float32	max. 0.6
EndoflGreifarm01_AXIS	Position des Greifarms	float32	-0.3 bis 5
EndoflGreifarm01_DLT_POS	Delta zwischen Ist- und Sollwert der Greifarmposition	float32	-0.3 bis 5

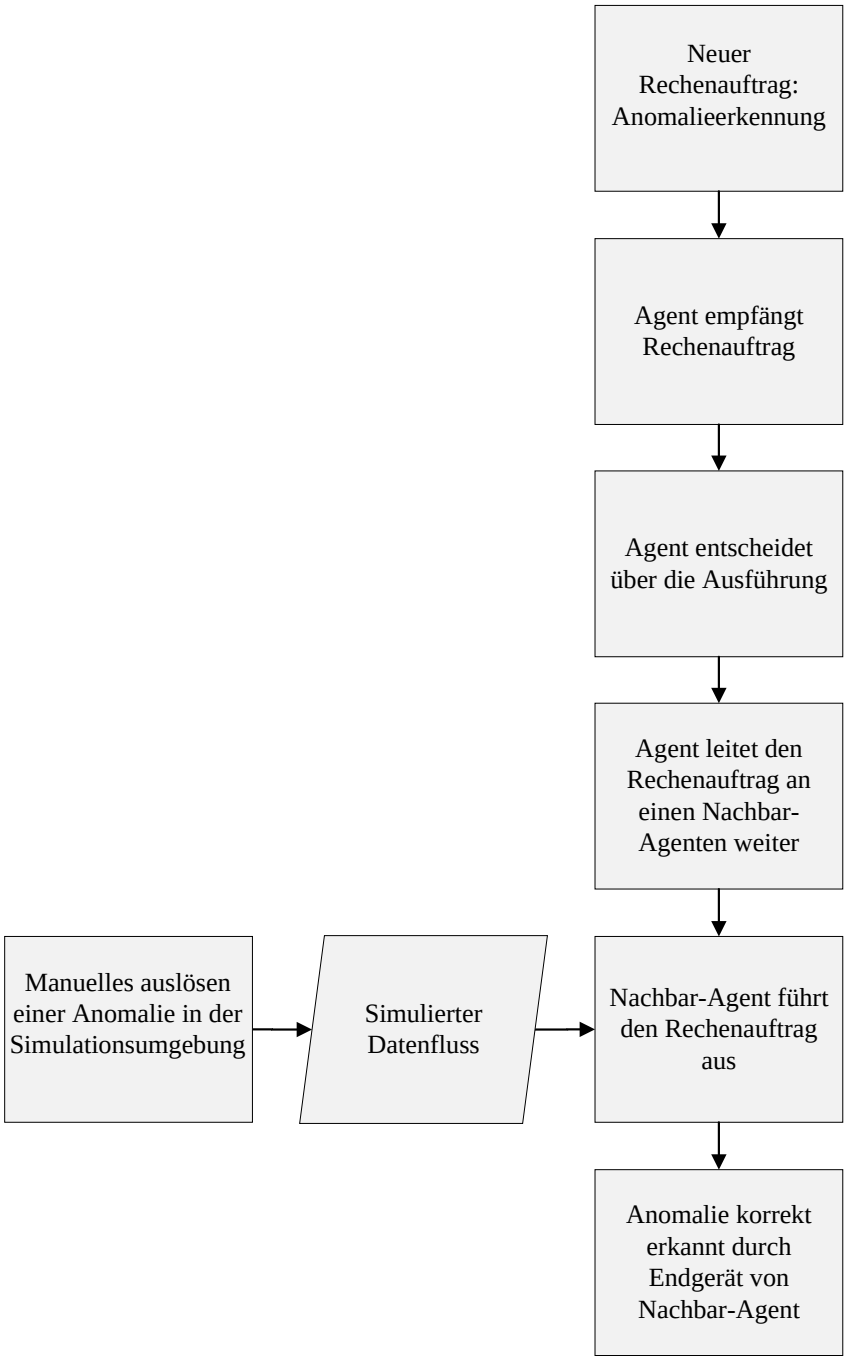


Abbildung A.8: Veranschaulichung des Ablaufs der Anomalieerkennung für einen simulierten Datenfluss mit manuell initiierter Anomalie

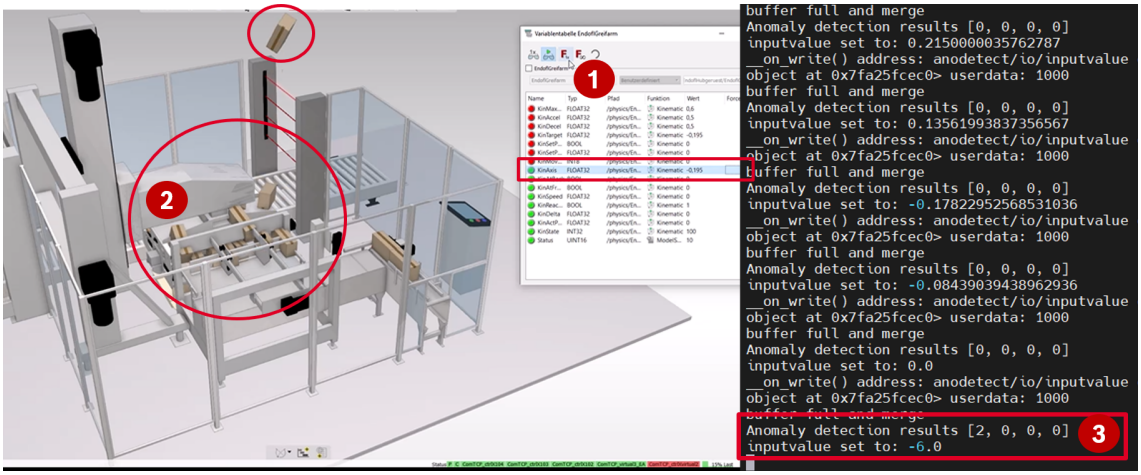


Abbildung A.9: Veranschaulichung der manuell initiierten Anomalie in der Simulationsumgebung (Rosenberger, Selig, Ristic u. a. 2023). (1) Schreiben eines fehlerhaften Werts, (2) Visualisierung des Fehlverhaltens in der Anlage, (3) Erkennung und Anzeige der Anomalie durch das Endgerät

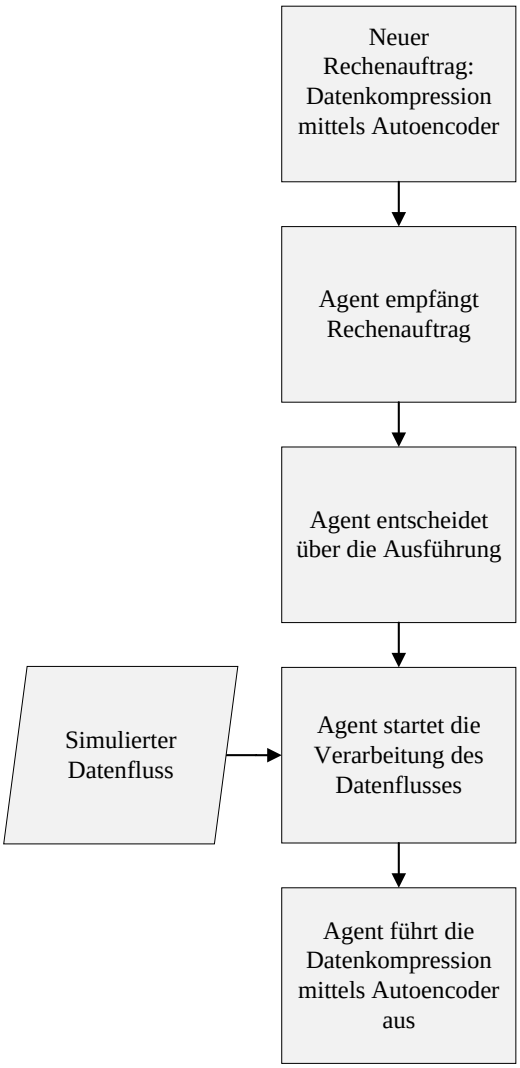


Abbildung A.10: Veranschaulichung des Ablaufs der Datenkompression für einen simulierten Datenfluss

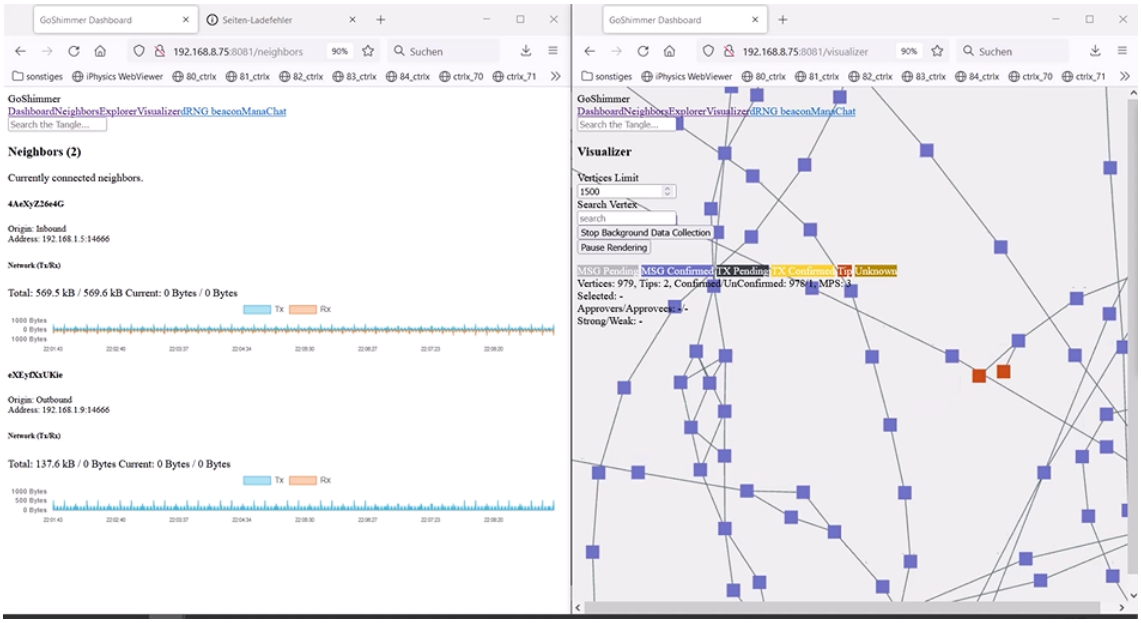


Abbildung A.11: Visualisierung des Tangles von dem durch die Endgeräte erstellten privaten IOTA Netzwerks (rechts) und GoShimmer Dashboard eines IOTA Knotens vernetzt mit zwei weiteren Konten (links)

Literaturverzeichnis

Agovic, Amrudin, Arindam Banerjee, Auroop Ratan Ganguly und Vladimir A. Protopopescu (2008). „Anomaly detection in transportation corridors using manifold embedding“. In: *Knowledge Discovery from Sensor Data*, S. 81–105.

Ahlfeld, Martin, Till Barleben, Mathias Cellarius, Alexander Duisberg, Jürgen Ensthaler, Bernhard Fischer, Florian Hilbert, Thomas Kriesel, Thomas Schauf, Martin Schweinoch, Daniel van Geerenstein und Wolfgang Zeiler (2022). *Fokusthema: Daten im Kontext von Industrie 4.0*. URL: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/daten-im-kontext-von-i40.html (besucht am 26. 04. 2022).

Ahmad, Subutai, Alexander Lavin, Scott Purdy und Zuha Agha (2017). „Unsupervised real-time anomaly detection for streaming data“. In: *Neurocomputing* 262. Online Real-Time Learning Strategies for Data Streams, S. 134–147. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.04.070.

Ahmad, Subutai und Scott Purdy (2016). „Real-Time Anomaly Detection for Streaming Analytics“. In: *CoRR* abs/1607.02480. DOI: 10.48550/arXiv.1607.02480.

Alagha, Hirad Emami (2019). „Communicating Intention in Decentralized Multi-Agent Multi-Objective Reinforcement Learning Systems“. Magisterarb. University of Groningen.

Aldweesh, Amjad, Maher Alharby, Ellis Solaiman und Aad van Moorsel (2018). „Performance Benchmarking of Smart Contracts to Assess Miner Incentives in Ethereum“. In: *2018 14th European Dependable Computing Conference (EDCC)*, S. 144–149. DOI: 10.1109/EDCC.2018.00034.

Amarasinghe, Gayashan, Marcos D. de Assunção, Aaron Harwood und Shanika Karunasekera (2018). „A Data Stream Processing Optimisation Framework for Edge Computing Applications“. In: *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, S. 91–98. DOI: 10.1109/ISORC.2018.00020.

Bakakeu, Jupiter, Dominik Kisskalt, Joerg Franke, Shirin Baer, Hans-Henning Klos und Joern Peschke (2020). „Multi-Agent Reinforcement Learning for the Energy Optimization of Cyber-Physical Production Systems“. In: *2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, S. 1–8. DOI: 10.1109/CCECE47787.2020.9255795.

Bellman, Richard (1954). „The theory of dynamic programming“. In: *Bulletin of the American Mathematical Society* 60.6, S. 503–515.

Ben Said, Ahmed, Amr Mohamed, Tarek Elfouly, Khaled Harras und Z. Jane Wang (2017). „Multimodal Deep Learning Approach for Joint EEG-EMG Data Compression and Classification“. In: *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, S. 1–6. DOI: 10.1109/WCNC.2017.7925709.

BITKOM e. V., VDMA e. V. und ZVEI e. V., Hrsg. (2015). *Umsetzungsstrategie Industrie 4.0. Result report of the platform Industrie 4.0.* URL: https://www.bitkom.org/Bitkom/Publikationen/Umsetzungsstrategie-Industrie-40.html (besucht am 09. 07. 2021).

Boyes, Hugh, Bil Hallaq, Joe Cunningham und Tim Watson (2018). „The industrial internet of things (IIoT): An analysis framework“. In: *Computers in Industry* 101, S. 1–12. ISSN: 0166-3615. DOI: 10.1016/j.compind.2018.04.015.

Burkov, Andriy (2019). *Machine Learning kompakt: Alles, was Sie wissen müssen.* MITP-Verlags GmbH & Co. KG.

Canese, Lorenzo, Gian Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re und Sergio Spanò (2021). „Multi-Agent Reinforcement Learning: A Review of Challenges and Applications“. In: *Applied Sciences* 11.11, S. 4948. DOI: 10.3390/app11114948.

Cao, Zilong, Pan Zhou, Ruixuan Li, Siqi Huang und Dapeng Wu (2020). „Multiagent Deep Reinforcement Learning for Joint Multichannel Access and Task Offloading of Mobile-Edge Computing in Industry 4.0“. In: *IEEE Internet of Things Journal* 7.7, S. 6201–6213. DOI: 10.1109/JIOT.2020.2968951.

Chandola, Varun, Arindam Banerjee und Vipin Kumar (2009). „Anomaly Detection: A Survey“. In: *ACM Comput. Surv.* 41.3. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

Chapman, Pete, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, Rudiger Wirth u. a. (2000). „CRISP-DM 1.0: Step-by-step data mining guide“. In: *SPSS inc* 9, S. 13.

Chatterjee, Baibhab, Ningyuan Cao, Arijit Raychowdhury und Shreyas Sen (2019). „Context-Aware Intelligence in Resource-Constrained IoT Nodes: Opportunities and Challenges“. In: *IEEE Design Test* 36.2, S. 7–40. DOI: 10.1109/MDAT.2019.2899334.

Chen, Baotong, Jiafu Wan, Lei Shu, Peng Li, Mithun Mukherjee und Boxing Yin (2018). „Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges“. In: *IEEE Access* 6, S. 6505–6519. DOI: 10.1109/ACCESS.2017.2783682.

Chen, Dan, Yi-Jen Chiang, Nasir Memon und Xiaolin Wu (2003). „Optimal alphabet partitioning for semi-adaptive coding of sources of unknown sparse distributions“.

In: *Data Compression Conference, 2003. Proceedings. DCC 2003.* IEEE, S. 372–381.

Chen, Wuhui, Xiaoyu Qiu, Ting Cai, Hong-Ning Dai, Zibin Zheng und Yan Zhang (2021). „Deep Reinforcement Learning for Internet of Things: A Comprehensive Survey". In: *IEEE Communications Surveys Tutorials*, S. 1–1. DOI: 10.1109/COMST.2021.3073036.

Chen, Xing, Junqin Hu, Zheyi Chen, Bing Lin, Naixue Xiong und Geyong Min (2022). „A Reinforcement Learning-Empowered Feedback Control System for Industrial Internet of Things". In: *IEEE Transactions on Industrial Informatics* 18.4, S. 2724–2733. DOI: 10.1109/TII.2021.3076393.

Chen, Ying, Zhiyong Liu, Yongchao Zhang, Yuan Wu, Xin Chen und Lian Zhao (2021). „Deep Reinforcement Learning-Based Dynamic Resource Management for Mobile Edge Computing in Industrial Internet of Things". In: *IEEE Transactions on Industrial Informatics* 17.7, S. 4925–4934. DOI: 10.1109/TII.2020.3028963.

Chenaghlou, Milad, Masud Moshtaghi, Christopher Leckie und Mahsa Salehi (2018). „Online Clustering for Evolving Data Streams with Online Anomaly Detection". In: *Advances in Knowledge Discovery and Data Mining*. Hrsg. von Dinh Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji und Lida Rashidi. Springer International Publishing, S. 508–521. ISBN: 978-3-319-93037-4.

Chiarot, Giacomo und Claudio Silvestri (2023). „Time Series Compression Survey". In: *ACM Comput. Surv.* 55.10. ISSN: 0360-0300. DOI: 10.1145/3560814.

Dautov, Rustem und Salvatore Distefano (2020). „Stream Processing on Clustered Edge Devices". In: *IEEE Transactions on Cloud Computing*, S. 1–1. DOI: 10.1109/TCC.2020.2983402.

Dautov, Rustem, Salvatore Distefano, Dario Bruneo, Francesco Longo, Giovani Merlino und Antonio Puliafito (2017). „Pushing Intelligence to the Edge with a Stream Processing Architecture". In: *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, S. 792–799. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.121.

De Blasi, Stefano und Elmar Engels (2020). „Next generation control units simplifying industrial machine learning". In: *IEEE 29th International Symposium on Industrial Electronics (ISIE)*.

Desforges, M. J., P. J. Jacob und J. E. Cooper (1998). „Applications of probability density estimation to the detection of abnormal conditions in engineering". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 212.8, S. 687–703. DOI: 10.1243/0954406981521448.

Dias de Assunção, Marcos, Alexandre da Silva Veith und Rajkumar Buyya (2018). „Distributed data stream processing and edge computing: A survey on resource elasticity and future directions". In: *Journal of Network and Computer Applications* 103, S. 1–17. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2017.12.001.

Diedrich, Christian und Matthias Riedl (2016). „Integration von Automatisierungsgeräten in Industrie-4.0-Komponenten". In: *Handbuch Industrie 4.0: Produktion,*

Automatisierung und Logistik. Hrsg. von Birgit Vogel-Heuser, Thomas Bauernhansl und Michael ten Hompel. Springer Berlin Heidelberg, S. 1–14. ISBN: 978-3-662-45537-1. DOI: 10.1007/978-3-662-45537-1_63-1.

Din, Muhammad Faisal und Sameer Qazi (2018). „A compressed framework for monitoring and anomaly detection in cloud networks". In: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, S. 1–7. DOI: 10.1109/ICOMET.2018.8346394.

DIN EN 62264 (2014). *Enterprise-control system integration*. DOI: 10.31030/2156368.

DIN SPEC 91345 (2016). *91345:2016-04 Reference Architecture Model Industrie 4.0 (RAMI4.0)*.

Ding, Ruijin, Yuwen Yang, Jun Liu, Hongyan Li und Feifei Gao (2020). „Packet Routing Against Network Congestion: A Deep Multi-agent Reinforcement Learning Approach". In: *2020 International Conference on Computing, Networking and Communications (ICNC)*, S. 932–937. DOI: 10.1109/ICNC47757.2020.9049759.

Ding, Zhiguo und Minrui Fei (2013). „An Anomaly Detection Approach Based on Isolation Forest Algorithm for Streaming Data using Sliding Window". In: *IFAC Proceedings Volumes* 46.20. 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013, S. 12–17. ISSN: 1474-6670. DOI: 10.3182/20130902-3-CN-3020.00044.

Domingos, Pedro und Geo Hulten (2001). „Catching up with the data: Research issues in mining data streams". In: *In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

Dönicke, Nicole, Wolfgang Fritsche, Thomas Gamer, Tobias Heer, Lutz Jänicke, Michael Jochem, Wolfgang Klasen, Thomas Lantermann, Lukas Linke, Jens Mehrfeld, Tobias Pfeiffer und Andreas Teuscher (2017). *Security der Verwaltungsschale*. URL: https://www.zvei.org/presse-medien/publikationen/diskussionspapier-security-der-verwaltungsschale/ (besucht am 01. 04. 2022).

Eichinger, Frank, Pavel Efros, Stamatis Karnouskos und Klemens Böhm (2015). „A time-series compression technique and its application to the smart grid". In: *The VLDB Journal* 24.2, S. 193–218.

ElMamy, Sidi Boubacar, Hichem Mrabet, Hassen Gharbi, Abderrazak Jemai und Damien Trentesaux (2020). „A Survey on the Usage of Blockchain Technology for Cyber-Threats in the Context of Industry 4.0". In: *Sustainability* 12.21. ISSN: 2071-1050. DOI: 10.3390/su12219179.

Ertel, Wolfgang (2021). *Grundkurs Künstliche Intelligenz*. 5. Aufl. Springer Vieweg.

Eskin, Eleazar (2000). „Anomaly Detection over Noisy Data using Learned Probability Distributions". In: *In Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, S. 255–262.

Europäische Wirtschaftsforschung GmbH, Zentrum für (2015). *Industrie 4.0: Digitale (R)Evolution der Wirtschaft*. URL: https://www.zew.de/publikationen/2015 (besucht am 18. 12. 2019).

Fahim, Muhammad und Alberto Sillitti (2019). „Anomaly Detection, Analysis and Prediction Techniques in IoT Environment: A Systematic Literature Review". In: *IEEE Access* 7, S. 81664–81681. DOI: 10.1109/ACCESS.2019.2921912.

Fan, Caixiang, Sara Ghaemi, Hamzeh Khazaei und Petr Musilek (2020). „Performance Evaluation of Blockchain Systems: A Systematic Survey“. In: *IEEE Access* 8, S. 126927–126950. DOI: 10.1109/ACCESS.2020.3006078.

Fan, Caixiang, Hamzeh Khazaei, Yuxiang Chen und Petr Musilek (2019). „Towards A Scalable DAG-based Distributed Ledger for Smart Communities“. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, S. 177–182. DOI: 10.1109/WF-IoT.2019.8767342.

Farahani, Bahar, Farshad Firouzi und Markus Luecking (2021). „The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions“. In: *Journal of Network and Computer Applications* 177, S. 102936. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2020.102936.

Felser, Max, Markus Rentschler und Oliver Kleineberg (2019). „Coexistence Standardization of Operation Technology and Information Technology“. In: *Proceedings of the IEEE* 107.6, S. 962–976. DOI: 10.1109/JPROC.2019.2901314.

Firouzi, Farshad, Krishnendu Chakrabarty und Sani Nassif (2020). *Intelligent internet of things: From device to fog and cloud*. Springer.

Fisch, Alexander T. M., Idris A. Eckley und Paul Fearnhead (2019). *A linear time method for the detection of point and collective anomalies*. DOI: 10.48550/arXiv.1806.01947.

Foundation, IOTA, Hrsg. (2022). *A Permanent IOTA Message Storage Solution*. URL: https://github.com/iotaledger/chronicle.rs (besucht am 13. 08. 2022).

Fragkoulis, Marios, Paris Carbone, Vasiliki Kalavri und Asterios Katsifodimos (2020). „A Survey on the Evolution of Stream Processing Systems“. In: *ArXiv* abs/2008.00842.

Frochte, Jörg (2018). *Maschinelles Lernen*. Carl Hanser Verlag.

Garg, Sahil, Kuljeet Kaur, Neeraj Kumar, Georges Kaddoum, Albert Y. Zomaya und Rajiv Ranjan (2019). „A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks“. In: *IEEE Transactions on Network and Service Management* 16.3, S. 924–935. DOI: 10.1109/TNSM.2019.2927886.

Giusto, Daniel, Antonio Iera, Giacomo Morabito und Luigi Atzori (2010). *The internet of things: 20th Tyrrhenian workshop on digital communications*. Springer Science & Business Media.

Goldstein, Markus und S. Uchida (2016). „A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data“. In: *PLoS ONE* 11.

Gomes, Heitor Murilo, Jesse Read, Albert Bifet, Jean Paul Barddal und João Gama (2019). „Machine Learning for Streaming Data: State of the Art, Challenges, and Opportunities“. In: *SIGKDD Explor. Newsl.* 21.2, S. 6–22. ISSN: 1931-0145. DOI: 10.1145/3373464.3373470.

Gong, Yongkang, Haipeng Yao, Jingjing Wang, Liang Jiang und F. Richard Yu (2021). „Multi-Agent Driven Resource Allocation and Interference Management for Deep Edge Networks“. In: *IEEE Transactions on Vehicular Technology*, S. 1–1. DOI: 10.1109/TVT.2021.3134467.

Gronauer, Sven und Klaus Diepold (2021). „Multi-agent deep reinforcement learning: a survey“. In: *Artificial Intelligence Review*, S. 1–49.

Gupta, Jayesh K., Maxim Egorov und Mykel Kochenderfer (2017). „Cooperative Multi-agent Control Using Deep Reinforcement Learning“. In: *Autonomous Agents and Multiagent Systems*. Hrsg. von Gita Sukthankar und Juan A. Rodriguez-Aguilar. Springer International Publishing, S. 66–83. ISBN: 978-3-319-71682-4.

Gusev, Marjan, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Schahram Dustdar, Ognjen Scekic, Thomas Rausch, Stefan Nastic, Sasko Ristov und Thomas Fahringer (2019). „A Deviceless Edge Computing Approach for Streaming IoT Applications“. In: *IEEE Internet Computing* 23.1, S. 37–45. DOI: 10.1109/MIC.2019.2892219.

Guttormsson, Sigurour E, RJ Marks, MA El-Sharkawi und I Kerszenbaum (1999). „Elliptical novelty grouping for on-line short-turn detection of excited running rotors“. In: *IEEE Transactions on Energy Conversion* 14.1, S. 16–22.

Habak, Karim, Mostafa Ammar, Khaled A. Harras und Ellen Zegura (2015). „Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge“. In: *2015 IEEE 8th International Conference on Cloud Computing*, S. 9–16. DOI: 10.1109/CLOUD.2015.12.

Han, Runchao, Gary Shapiro, Vincent Gramoli und Xiwei Xu (2020). „On the performance of distributed ledgers for Internet of Things“. In: *Internet of Things* 10. Special Issue of the Elsevier IoT Journal on Blockchain Applications in IoT Environments, S. 100087. ISSN: 2542-6605. DOI: 10.1016/j.iot.2019.100087.

Hasan, Ragib, Md. Mahmud Hossain und Rasib Khan (2015). „Aura: An IoT Based Cloud Infrastructure for Localized Mobile Computation Outsourcing“. In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, S. 183–188. DOI: 10.1109/MobileCloud.2015.37.

Hawkins, S. Edward und Edward Hugo Darlington (2012). „Algorithm for Compressing Time-Series Data“. In: URL: %5Curl%7Bhttps://ntrs:nasa:gov/search:jsp?R=20120010460%7D (besucht am 04. 05. 2022).

Hermann, Mario, Tobias Pentek und Boris Otto (2016). „Design Principles for Industrie 4.0 Scenarios“. In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*, S. 3928–3937. DOI: 10.1109/HICSS.2016.488.

Hesse, Guenter, Werner Sinzig, Christoph Matthies und Matthias Uflacker (2019). „Application of Data Stream Processing Technologies in Industry 4.0: What is Missing?“ In: *Proceedings of the 8th International Conference on Data Science, Technology and Applications - DATA*. INSTICC. SciTePress, S. 304–310. DOI: 10.5220/0007950203040310.

Hilbrich, Hans-Christian (2008). „Produktionsdaten ganzheitlich integrieren“. In: *wissensmanagement*.

Hirzel, Martin, Robert Soulé, Scott Schneider, Buğra Gedik und Robert Grimm (2014). „A Catalog of Stream Processing Optimizations“. In: *ACM Comput. Surv.* 46.4. ISSN: 0360-0300. DOI: 10.1145/2528412.

Hochreiter, Sepp und Jürgen Schmidhuber (1997). „Long Short-Term Memory“. In: *Neural Computation* 9.8, S. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Hoffmeister, Michael, Birgit Boss, Andreas Orzelski und Johanna Wagner (2021). „Die Verwaltungsschale: Zentrum der digitalen Vernetzung in Fabirken (Teil 1)“. In: *atp magazin*.

Hofmockel, Julia (2019). „Anomalieerkennung in Kommunikationsdaten zur Datenselektion im Fahrzeug“. Diss. Karlsruher Institut für Technologie (KIT).

Hsu, Daniel (2017). „Time series compression based on adaptive piecewise recurrent autoencoder“. In: DOI: 10.48550/arXiv.1707.07961.

Hu, Weiming, Jun Gao, Bing Li, Ou Wu, Junping Du und Stephen Maybank (2020). „Anomaly Detection Using Local Kernel Density Estimation and Context-Based Regression“. In: *IEEE Transactions on Knowledge and Data Engineering* 32.2, S. 218–233. DOI: 10.1109/TKDE.2018.2882404.

Huang, Changchi und Yuwen Chen (2017). „Lossless Compression Algorithm for Multi-source Sensor Data Research“. In: *Proceedings of the 7th International Conference on Education, Management, Information and Mechanical Engineering (EMIM 2017)*. Atlantis Press, S. 1324–1331. ISBN: 978-94-6252-356-2. DOI: 10.2991/emim-17.2017.267.

Huber, Daniel und Thomas Kaiser (2017). „Wie das Internet der Dinge neue Geschäftsmodelle ermöglicht“. In: *Industrie 4.0: Herausforderungen, Konzepte und Praxisbeispiele*. Hrsg. von Stefan Reinheimer. Springer Fachmedien Wiesbaden, S. 17–27. ISBN: 978-3-658-18165-9. DOI: 10.1007/978-3-658-18165-9_2.

Huber, Walter (2016). *Industrie 4.0 in der Automobilproduktion. Ein Praxisbuch*. Springer Vieweg.

Huffman, David A. (1952). „A method for the construction of minimum-redundancy codes“. In: *Proceedings of the IRE* 40.9, S. 1098–1101.

Hundman, Kyle, Valentino Constantinou, Christopher Laporte, Ian Colwell und Tom Soderstrom (2018). „Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding“. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, S. 387–395. ISBN: 9781450355520. DOI: 10.1145/3219819.3219845.

Ibarria, Lawrence, Peter Lindstrom, Jarek Rossignac und Andrzej Szymczak (2003). „Out-of-core compression and decompression of large n-dimensional scalar fields“. In: *Computer Graphics Forum* 22, S. 343–348.

Idé, Tsuyoshi und Hisashi Kashima (2004). „Eigenspace-based anomaly detection in computer systems“. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, S. 440–449.

IEC 63278-1 (2021). *ED1 - Asset administration shell (AAS) for industrial applications - Part 1: Asset Administration shell structure*.

IEEE 754 (1985). „Standard for Binary Floating-Point Arithmetic“. In: *ANSI/IEEE Std 754-1985*, S. 1–20. DOI: 10.1109/IEEESTD.1985.82928.

Isah, Haruna, Tariq Abughofa, Sazia Mahfuz, Dharmitha Ajerla, Farhana Zulkernine und Shahzad Khan (2019). „A Survey of Distributed Data Stream Processing Frameworks“. In: *IEEE Access* 7, S. 154300–154316. DOI: 10.1109/ACCESS.2019.2946884.

JavaTpoint (2018). *JavaTpoint - K-Means Clustering Algorithm.* URL: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning (besucht am 14. 10. 2021).

Jiang, Yiming, Chenxu Wang, Ye Huang, Siyu Long und Yilun Huo (2018). „A Cross-Chain Solution to Integration of IoT Tangle for Data Access Management". In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, S. 1035–1041. DOI: 10.1109/Cybermatics_2018.2018.00192.

Jin, Tiankai, Zhiduo Ji, Shanying Zhu und Cailian Chen (2021). „Learning-based Co-Design of Distributed Edge Sensing and Transmission for Industrial Cyber-Physical Systems". In: *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, S. 1–6. DOI: 10.1109/INDIN45523.2021.9557472.

Jogunola, Olamide, Mohammad Hammoudeh, Kelvin Anoh und Bamidele Adebisi (2020). „Distributed Ledger Technologies for Peer-to-Peer Energy Trading". In: *2020 IEEE Electric Power and Energy Conference (EPEC)*, S. 1–6. DOI: 10.1109/EPEC48502.2020.9320061.

Keogh, Eamonn, Stefano Lonardi und Chotirat Ann Ratanamahatana (2004). „Towards parameter-free data mining". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, S. 206–215.

Kingma, Diederik P. und Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization.* DOI: 10.48550/arXiv.1412.6980.

Köhne, Frank (2019). *Anomalien mit H2O.AI - Isolation Forests finden und erklären.* URL: https://blog.viadee.de/isolation-forests-anomalien-mit-h2o-ai (besucht am 14. 10. 2021).

Kusmierz, Bartosz, William Sanders, Andreas Penzkofer, Angelo Capossele und Alon Gal (2019). „Properties of the Tangle for Uniform Random and Random Walk Tip Selection". In: *2019 IEEE International Conference on Blockchain (Blockchain).* DOI: 10.1109/blockchain.2019.00037.

Lavin, Alexander und Subutai Ahmad (2015). „Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark". In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, S. 38–44. DOI: 10.1109/ICMLA.2015.141.

Lee, Edward A. (2008). „Cyber physical systems: Design challenges". In: *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC).* IEEE, S. 363–369.

Lee, Wenke und Dong Xiang (2001). „Information-theoretic measures for anomaly detection". In: *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, S. 130–143. DOI: 10.1109/SECPRI.2001.924294.

Lemke, Claudia und Walter Brenner (2015). „Einführung in das digitale Zeitalter". In: *Einführung in die Wirtschaftsinformatik: Band 1: Verstehen des digitalen Zeitalters.* Springer Berlin Heidelberg, S. 11–51. ISBN: 978-3-662-44065-0. DOI: 10.1007/978-3-662-44065-0_2.

Lexa, Carsten (2021). „Digitale Revolution“. In: *Fit für die digitale Zukunft: Trends der digitalen Revolution und welche Kompetenzen Sie dafür brauchen*. Springer Fachmedien Wiesbaden, S. 15–17. ISBN: 978-3-658-33073-6. DOI: 10.1007/978-3-658-33073-6_5.

Li, Di, Hao Tang, Shiyong Wang und Chengliang Liu (2017). „A big data enabled load-balancing control for smart manufacturing of Industry 4.0“. In: *Cluster Computing* 20 (2), S. 1855–1864. ISSN: 1573-7543. DOI: 10.1007/s10586-017-0852-1.

Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh und Ameet Talwalkar (2018). „Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization“. In: *Journal of Machine Learning Research* 18.185, S. 1–52.

Li, Teng, Zhiyuan Xu, Jian Tang und Yanzhi Wang (2018). „Model-Free Control for Distributed Stream Data Processing Using Deep Reinforcement Learning“. In: 11.6, S. 705–718. ISSN: 2150-8097. DOI: 10.14778/3199517.3199521.

Li, Zheng und Caili Guo (2020). „Multi-Agent Deep Reinforcement Learning Based Spectrum Allocation for D2D Underlay Communications“. In: *IEEE Transactions on Vehicular Technology* 69.2, S. 1828–1840. DOI: 10.1109/TVT.2019.2961405.

Lindstrom, Peter und Martin Isenburg (2006). „Fast and Efficient Compression of Floating-Point Data“. In: *IEEE Transactions on Visualization and Computer Graphics* 12.5, S. 1245–1250. DOI: 10.1109/TVCG.2006.143.

Liu, Bo, Mohamed Mohandes, Hilal Nuha, Mohamed Deriche und Faramarz Fekri (2018). „A Distributed Principal Component Analysis Compression for Smart Seismic Acquisition Networks“. In: *IEEE Transactions on Geoscience and Remote Sensing* 56.6, S. 3020–3029. DOI: 10.1109/TGRS.2018.2789354.

Liu, Fei Tony, Kai Ming Ting und Zhi-Hua Zhou (2008). „Isolation Forest“. In: *2008 Eighth IEEE International Conference on Data Mining*, S. 413–422.

Liu, Jianlin, Fenxiong Chen und Dianhong Wang (2018). „Data Compression Based on Stacked RBM-AE Model for Wireless Sensor Networks“. In: *Sensors* 18, S. 4273. DOI: 10.3390/s18124273.

Liu, Jianlin, Fenxiong Chen, Jun Yan und Dianhong Wang (2019). „CBN-VAE: A Data Compression Model with Efficient Convolutional Structure for Wireless Sensor Networks“. In: *Sensors* 19.16. ISSN: 1424-8220. DOI: 10.3390/s19163445.

Liu, Xiaolan, Jiadong Yu, Zhiyong Feng und Yue Gao (2020). „Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing“. In: *China Communications* 17.9, S. 220–236. DOI: 10.23919/JCC.2020.09.017.

Liu, Xunyun und Rajkumar Buyya (2020). „Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and Future Directions“. In: *ACM Comput. Surv.* 53.3. ISSN: 0360-0300. DOI: 10.1145/3355399.

Liu, Yong, Don Towsley, Jing Weng und Dennis Goeckel (2005). „An information theoretic approach to network trace compression“. In: *University of Massachusetts, Amherst, Tech. Rep. CS TR05-03*.

Lu, Yang (2017). „Industry 4.0: A survey on technologies, applications and open research issues“. In: *Journal of Industrial Information Integration* 6, S. 1–10. ISSN: 2452-414X. DOI: 10.1016/j.jii.2017.04.005.

Luo, Shu, Linxuan Zhang und Yushun Fan (2021). „Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning“. In: *IEEE Transactions on Automation Science and Engineering*, S. 1–19. DOI: 10.1109/TASE.2021.3104716.

Luong, Nguyen Cong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang und Dong In Kim (2019). „Applications of Deep Reinforcement Learning in Communications and Networking: A Survey“. In: *IEEE Communications Surveys Tutorials* 21.4, S. 3133–3174. DOI: 10.1109/COMST.2019.2916583.

Lupascu, Cristian, Alexandru Lupascu und Ion Bica (2020). „DLT Based Authentication Framework for Industrial IoT Devices“. In: *Sensors* 20.9. ISSN: 1424-8220. DOI: 10.3390/s20092621.

machineering GmbH & Co. KG, Hrsg. (2022). *Virtual commissioning VIBN with iPhysics*. URL: https://www.machineering.com/en/ (besucht am 20. 10. 2022).

Mao, Hangyu, Zhibo Gong und Zhen Xiao (2020). *Reward Design in Cooperative Multi-agent Reinforcement Learning for Packet Routing*. DOI: 10.48550/arXiv.2003.03433.

Mao, Yuyi, Jun Zhang und Khaled B. Letaief (2016). „Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices“. In: *IEEE Journal on Selected Areas in Communications* 34.12, S. 3590–3605. DOI: 10.1109/JSAC.2016.2611964.

Marteau, Pierre-Francois (2021). „Random Partitioning Forest for Point-Wise and Collective Anomaly Detection—Application to Network Intrusion Detection“. In: *IEEE Transactions on Information Forensics and Security* 16, S. 2157–2172. DOI: 10.1109/TIFS.2021.3050605.

Martin, G Nigel N (1979). „Range encoding: an algorithm for removing redundancy from a digitised message“. In: *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*. Bd. 2.

Meudt, Tobias, Malte Pohl und Joachim Metternich (2017). *Die Automatisierungspyramide - Ein Literaturüberblick*.

Meyer, Martin (2019). *Kommunikationstechnik - Konzepte der modernen Nachrichtenübertragung*. Springer.

Mierswa, Ingo (2017). *K-Nearest Neighbors: A Simple Machine Learning Algorithm*. URL: https://rapidminer.com/blog/k-nearest-neighbors-laziest-machine-learning-technique (besucht am 14. 10. 2021).

Milisavljevic-Syed, Jelena, Lane Thames und Dirk Schaefer (2020). „The Digitization of Design and Manufacturing: A State-of-the-Art Report on the Transition from Strategic Vision to Implementation in Industry“. In: *Procedia CIRP* 93. 53rd CIRP Conference on Manufacturing Systems 2020, S. 575–580. ISSN: 2212-8271. DOI: 10.1016/j.procir.2020.03.088.

Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill. ISBN: 9780071154673.

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver und Koray Kavukcuoglu (2016). „Asynchronous Methods for Deep Reinforcement Learning“. In: *Proceedings of the 33rd*

International Conference on International Conference on Machine Learning - Volume 48. ICML'16. New York, NY, USA: JMLR.org, S. 1928–1937.

Monahan, George E. (1982). „State of the Art—A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms". In: *Management Science* 28.1, S. 1–16. DOI: 10.1287/mnsc.28.1.1.

Mtibaa, Abderrahmen, Khaled A. Harras und Afnan Fahim (2013). „Towards Computational Offloading in Mobile Device Clouds". In: *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. Bd. 1, S. 331–338. DOI: 10.1109/CloudCom.2013.50.

Murudkar, Chetana V. und Richard D. Gitlin (2019). „Optimal-Capacity, Shortest Path Routing in Self-Organizing 5G Networks using Machine Learning". In: *2019 IEEE 20th Wireless and Microwave Technology Conference (WAMICON)*, S. 1–5. DOI: 10.1109/WAMICON.2019.8765434.

Nagl, Anna und Karlheinz Bozem (2018). *Geschäftsmodelle 4.0. Business Model Building mit Checklisten und Fallbeispielen*. Springer Gabler.

Nasiri, Hamid, Saeed Nasehi und Maziar Goudarzi (2019). „Evaluation of distributed stream processing frameworks for IoT applications in Smart Cities". In: *Journal of Big Data* 6.1, S. 1–24.

Nastic, Stefan, Thomas Rausch, Ognjen Scekic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov und Radu Prodan (2017). „A Serverless Real-Time Data Analytics Platform for Edge Computing". In: *IEEE Internet Computing* 21.4, S. 64–71. DOI: 10.1109/MIC.2017.2911430.

Nedden, Max (2012). *Neue Methoden zur Charakterisierung der QSAR-Anwendungsdomäne: modifizierte Kerndichteschätzung der Vorhersagegüte empirisch abgeleiteter Modelle in der Chemie*. Springer-Verlag.

Ni, FuTao, Jian Zhang und Mohammad N. Noori (2019). „Deep learning for data anomaly detection and data compression of a long-span suspension bridge". In: *Computer-Aided Civil and Infrastructure Engineering* 35.7, S. 685–700. DOI: 10.1111/mice.12528.

Nibali, Aiden und Zhen He (2015). „Trajic: An Effective Compression System for Trajectory Data". In: *IEEE Transactions on Knowledge and Data Engineering* 27.11, S. 3138–3151. DOI: 10.1109/TKDE.2015.2436932.

Nikoui, Tina Samizadeh, Amir Masoud Rahmani und Hooman Tabarsaied (2019). „Data Management in Fog Computing". In: *Fog and Edge Computing*. John Wiley & Sons, Ltd. Kap. 8, S. 171–190. ISBN: 9781119525080. DOI: 10.1002/9781119525080.ch8.

Park, Junmin, Hyunjae Park und Young-June Choi (2018). „Data compression and prediction using machine learning for industrial IoT". In: *2018 International Conference on Information Networking (ICOIN)*, S. 818–820. DOI: 10.1109/ICOIN.2018.8343232.

Parra, Lucas, Gustavo Deco und Stefan Miesbach (1996). „Statistical Independence and Novelty Detection with Information Preserving Nonlinear Maps". In: *Neural Comput.* 8.2, S. 260–269. ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.2.260.

Peña-Cabrera, Mario, Victor Lomas und Gastón Lefranc (2019). „Fourth industrial revolution and its impact on society“. In: *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, S. 1–6. DOI: 10.1109/CHILECON47746.2019.8988083.

Pervez, Huma, Muhammad Muneeb, Muhammad Usama Irfan und Irfan Ul Haq (2018). „A Comparative Analysis of DAG-Based Blockchain Architectures“. In: *2018 12th International Conference on Open Source Systems and Technologies (ICOSST)*, S. 27–34. DOI: 10.1109/ICOSST.2018.8632193.

Pinheiro, Pedro, Mário Macedo, Ricardo Barbosa, Ricardo Santos und Paulo Novais (2018). „Multi-agent Systems Approach to Industry 4.0: Enabling Collaboration Considering a Blockchain for Knowledge Representation“. In: *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*. Hrsg. von Javier Bajo, Juan M. Corchado, Elena María Navarro Martínez, Eneko Osaba Icedo, Philippe Mathieu, Patrycja Hoffa-Dabrowska, Elena del Val, Sylvain Giroux, Antonio J.M. Castro, Nayat Sánchez-Pi, Vicente Julián, Ricardo Azambuja Silveira, Alberto Fernández, Rainer Unland und Rubén Fuentes-Fernández. Springer International Publishing, S. 149–160.

Pinjala, Sandeep Kiran und Krishna M. Sivalingam (2019). „DCACI: A Decentralized Lightweight Capability Based Access Control Framework using IOTA for Internet of Things“. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, S. 13–18. DOI: 10.1109/WF-IoT.2019.8767356.

Pisani, Flávia, Jeferson Rech Brunetta, Vanderson Martins Do Rosario und Edson Borin (2017). „Beyond the Fog: Bringing Cross-Platform Code Execution to Constrained IoT Devices“. In: *2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, S. 17–24. DOI: 10.1109/SBAC-PAD.2017.10.

Pokrajac, Dragoljub, Aleksandar Lazarevic und Longin Jan Latecki (2007). „Incremental Local Outlier Detection for Data Streams“. In: *2007 IEEE Symposium on Computational Intelligence and Data Mining*, S. 504–515. DOI: 10.1109/CIDM.2007.368917.

Popov, Serguei (2018). *On the Tangle, White Papers, Proofs, Airplanes, and Local Modifiers*. https://blog.iota.org/on-the-tangle-white-papers-proofs-airplanes-and-local-modifiers-44683aff8fea/ (besucht am 14.10.2021).

Prometheus, Hrsg. (2022). *Introduction - What is Prometheus?* URL: https://prometheus.io/docs/introduction/overview/ (besucht am 13. 08. 2022).

Raschendorfer, Alexander, Benjamin Mörzinger, Eric Steinberger, Patrick Pelzmann, Ralf Oswald, Manuel Stadler und Friedrich Bleicher (2019). „On IOTA as a potential enabler for an M2M economy in manufacturing“. In: *Procedia CIRP* 79. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy, S. 379–384. ISSN: 2212-8271. DOI: 10.1016/j.procir.2019.02.096.

Ren, Yijing, Yaohua Sun und Mugen Peng (2021). „Deep Reinforcement Learning Based Computation Offloading in Fog Enabled Industrial Internet of Things“. In:

IEEE Transactions on Industrial Informatics 17.7, S. 4978–4987. DOI: 10.1109/TII.2020.3021024.

Renart, Eduard Gibert, Javier Diaz-Montes und Manish Parashar (2017). „Data-Driven Stream Processing at the Edge“. In: *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, S. 31–40. DOI: 10.1109/ICFEC.2017.18.

Roesch, Martin, Christian Linder, Christian Bruckdorfer, Andrea Hohmann und Gunther Reinhart (2019). „Industrial Load Management using Multi-Agent Reinforcement Learning for Rescheduling“. In: *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*, S. 99–102. DOI: 10.1109/AI4I46381.2019.00033.

Rosenberger, Julia, Michael Bühren und Dieter Schramm (2021). „Perspective on efficiency enhancements in processing streaming data in industrial IoT networks“. In: *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. DOI: 10.1109/GCAIoT53516.2021.9693073.

Rosenberger, Julia, Alexander Kübel und Fabian Rothfuß (2022). „Comparison and extension of autoencoder models for uni- and multivariate signal compression in IIoT“. In: *2022 Data Compression Conference (DCC)*, S. 481–481. DOI: 10.1109/DCC52660.2022.00092.

Rosenberger, Julia, Kevin Müller, Andreas Selig, Michael Bühren und Dieter Schramm (2022). „Extended kernel density estimation for anomaly detection in streaming data“. In: *Procedia CIRP* 112. 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 14-16 July 2021, S. 156–161. ISSN: 2212-8271. DOI: 10.1016/j.procir.2022.09.065.

Rosenberger, Julia, Felix Rauterberg und Dieter Schramm (2021). „Performance study on IOTA Chrysalis and Coordicide in the Industrial Internet of Things“. In: *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. DOI: 10.1109/GCAIoT53516.2021.9692985.

Rosenberger, Julia, Andreas Selig, Mirjana Ristic, Michael Bühren und Dieter Schramm (2023). „Virtual Commissioning of Distributed Systems in the Industrial Internet of Things“. In: *Sensors* 23.7. ISSN: 1424-8220. DOI: 10.3390/s23073545.

Rosenberger, Julia, Michael Urlaub, Felix Rauterberg, Tina Lutz, Andreas Selig, Michael Bühren und Dieter Schramm (2022). „Deep Reinforcement Learning Multi-Agent System for Resource Allocation in Industrial Internet of Things“. In: *Sensors* 22.11. ISSN: 1424-8220. DOI: 10.3390/s22114099.

Rosenberger, Julia, Michael Urlaub und Dieter Schramm (2021). „Multi-agent reinforcement learning for intelligent resource allocation in IIoT networks“. In: *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. DOI: 10.1109/GCAIoT53516.2021.9692913.

Rouhani, Sara und Ralph Deters (2019). „Security, Performance, and Applications of Smart Contracts: A Systematic Survey“. In: *IEEE Access* 7, S. 50759–50779. DOI: 10.1109/ACCESS.2019.2911031.

Russo, Gabriele R., Matteo Nardelli, V. Cardellini und F. L. Presti (2018). „Multi-Level Elasticity for Wide-Area Data Streaming Systems: A Reinforcement Learning Approach“. In: *Algorithms* 11, S. 134.

Salari, Anna (2018). *Hierarchie in der Industrie 4.0*. URL: https://www.plattform-i40.de/IP/Redaktion/DE/Infografiken/hierarchie-in-der-industrie-4-0.html (besucht am 27. 01. 2022).

Salehi, Mahsa, Christopher Leckie, James C. Bezdek, Tharshan Vaithianathan und Xuyun Zhang (2016). „Fast Memory Efficient Local Outlier Detection in Data Streams“. In: *IEEE Transactions on Knowledge and Data Engineering* 28.12, S. 3246–3260. DOI: 10.1109/TKDE.2016.2597833.

Salehi, Mahsa und Lida Rashidi (2018). „A Survey on Anomaly Detection in Evolving Data: [With Application to Forest Fire Risk Prediction]“. In: *SIGKDD Explor. Newsl.* 20.1, S. 13–23. ISSN: 1931-0145. DOI: 10.1145/3229329.3229332.

Salomon, David und Giovanni Motta (2010). *Handbook of data compression*. Springer.

Sanabria-Russo, Luis, David Pubill, Jordi Serra und Christos Verikoukis (2019). „IoT Data Analytics as a Network Edge Service“. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, S. 969–970. DOI: 10.1109/INFCOMW.2019.8845207.

Satyavolu, Prasad, Badrinath Setlur, Prasanth Thomas und Ganesh Iyer (2014). *Designing for Manufacturing's 'Internet of Things'. Cognizant Report*. URL: https://www.cognizant.com/InsightsWhitepapers/Designing-for-Manufacturings-Internet-of-Things.pdf (besucht am 06. 08. 2021).

Schaller, R.R. (1997). „Moore's law: past, present and future“. In: *IEEE Spectrum* 34.6, S. 52–59. DOI: 10.1109/6.591665.

Schulman, John, Oleg Klimov, Filip Wolski, Prafulla Dhariwal und Alec Radford (2017). *Proximal Policy Optimization*. URL: https://openai.com/blog/openai-baselines-ppo/ (besucht am 01. 04. 2022).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford und Oleg Klimov (2017). „Proximal Policy Optimization Algorithms“. In: *ArXiv* abs/1707.06347.

Schweizer, André, Patricia Knoll, Nils Urbach, Heiko Andreas von der Gracht und Thomas Hardjono (2020). „To What Extent Will Blockchain Drive the Machine Economy? Perspectives From a Prospective Study“. In: *IEEE Transactions on Engineering Management* 67.4, S. 1169–1183. DOI: 10.1109/TEM.2020.2979286.

Shahzad, Khurram und Mattias O'Nils (2018). „Condition Monitoring in Industry 4.0-Design Challenges and Possibilities: A Case Study“. In: *2018 Workshop on Metrology for Industry 4.0 and IoT*, S. 101–106. DOI: 10.1109/METROI4.2018.8428306.

Shapley, L. S. (1953). „Stochastic Games“. In: *Proceedings of the National Academy of Sciences* 39.10, S. 1095–1100. ISSN: 0027-8424. DOI: 10.1073/pnas.39.10.1095.

Singh, Saurabh, Sami Abu-El-Haija, Nick Johnston, Johannes Ballé, Abhinav Shrivastava und George Toderici (2020). „End-to-End Learning of Compressible Fea-

tures“. In: *2020 IEEE International Conference on Image Processing (ICIP)*, S. 3349–3353.

SkyllaDB, Hrsg. (2022). *System Requirements*. URL: https://docs.scylladb.com/stable/getting-started/system-requirements.html (besucht am 13. 08. 2022).

Snaps in Ubuntu Core (2022). URL: https://ubuntu.com/core/docs/snaps-in-ubuntu-core (besucht am 29. 09. 2022).

Song, Hongchao, Zhuqing Jiang, Aidong Men und B. Yang (2017). „A Hybrid Semi-Supervised Anomaly Detection Model for High-Dimensional Data“. In: *Computational Intelligence and Neuroscience* 2017.

Sparka, Hagen, Roman Naumann, Stefan Dietzel und Björn Scheuermann (2017). „Effective Lossless Compression of Sensor Information in Manufacturing Industry“. In: *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, S. 480–488. DOI: 10.1109/LCN.2017.89.

Subbotin, D. (1999). *Carryless rangecoder*. URL: https://metacpan.org/release/SALVA/Compress-PPMd-0.10/source/Coder.hpp (besucht am 31. 03. 2023).

Sun, Penghao, Zehua Guo, Gang Wang, Julong Lan und Yuxiang Hu (2020). „MARVEL: Enabling controller load balancing in software-defined networks with multi-agent reinforcement learning“. In: *Computer Networks* 177, S. 107230. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2020.107230.

TensorFlow Lite (2022). URL: https://www.tensorflow.org/lite/guide (besucht am 08. 06. 2022).

Terry, Justin K., Benjamin Black, Ananth Hari, Luis Santos, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, Caroline Horsch und Praveen Ravi (2020). „PettingZoo: Gym for Multi-Agent Reinforcement Learning“. In: *CoRR* abs/2009.14471. DOI: 10.48550/arXiv.2009.14471.

Terry, Justin K., Nathaniel Grammel, Benjamin Black, Ananth Hari, Caroline Horsch und Luis Santos (2020). „Agent Environment Cycle Games“. In: *CoRR* abs/2009.13051. DOI: 10.48550/arXiv.2009.13051.

Thames, Lane und Dirk Schaefer (2016). „Software-defined Cloud Manufacturing for Industry 4.0“. In: *Procedia CIRP* 52. The 6th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016), S. 12–17. ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.07.041.

Thill, Markus, Wolfgang Konen und Thomas Bäck (2017). „Time series anomaly detection with discrete wavelet transforms and maximum likelihood estimation“. In: *Intern. Conference on Time Series (ITISE)*.

Thompson, Benjamin Berry, Robert J Marks, Jai J Choi, Mohamed A El-Sharkawi, Ming-Yuh Huang und Carl Bunje (2002). „Implicit learning in autoencoder novelty assessment“. In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*. Bd. 3. IEEE, S. 2878–2883.

Urlaub, Michael und Julia Rosenberger (2022). „Multi-Agent Reinforcement Learning for smart Computing Resource Allocation in the Industry 4.0“. In: *Kölner Beiträge zur Technischen Informatik*.

Vinyals, Oriol, Samy Bengio und Manjunath Kudlur (2016). „Order Matters: Sequence to sequence for sets“. In: *CoRR* abs/1511.06391.

Vinyals, Oriol, Meire Fortunato und Navdeep Jaitly (2015). „Pointer Networks“. In: *Advances in Neural Information Processing Systems*. Hrsg. von C. Cortes, N. Lawrence, D. Lee, M. Sugiyama und R. Garnett. Bd. 28. Curran Associates, Inc.

Vogel-Heuser, Birgit (2014). „Herausforderungen und Anforderungen aus Sicht der IT und der Automatisierungstechnik“. In: *Industrie 4.0 in Produktion, Automatisierung und Logistik: Anwendung · Technologien · Migration*. Hrsg. von Thomas Bauernhansl, Michael ten Hompel und Birgit Vogel-Heuser. Springer Fachmedien Wiesbaden, S. 37–48.

Wang, Hao-nan, Ning Liu, Yi-yun Zhang, Da-wei Feng, Feng Huang, Dong-sheng Li und Yi-ming Zhang (2020). „Deep reinforcement learning: a survey“. In: *Frontiers of Information Technology & Electronic Engineering* 21.12, S. 1726–1744.

Wang, Huayong und Li-Shiuan Peh (2014). „Mobistreams: A reliable distributed stream processing system for mobile devices“. In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, S. 51–60.

Wang, Jiadai, Lei Zhao, Jiajia Liu und Nei Kato (2019). „Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach“. In: *IEEE Transactions on Emerging Topics in Computing*, S. 1–1. DOI: 10.1109/TETC.2019.2902661.

– (2021). „Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach“. In: *IEEE Transactions on Emerging Topics in Computing* 9.3, S. 1529–1541. DOI: 10.1109/TETC.2019.2902661.

Wang, Libing, Xin Hu, Yin Wang, Sujie Xu, Shijun Ma, Kexin Yang, Zhijun Liu und Weidong Wang (2021). „Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning“. In: *Computer Networks* 190, S. 107969. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2021.107969.

Wang, Wei und Min Zhang (2020). „Tensor Deep Learning Model for Heterogeneous Data Fusion in Internet of Things“. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.1, S. 32–41. DOI: 10.1109/TETCI.2018.2876568.

Wang, X., J. Hu, H. Lin, S. Garg, G. Kaddoum, M. Jalilpiran und M.S. Hossain (2021). „QoS and Privacy-Aware Routing for 5G enabled Industrial Internet of Things: A Federated Reinforcement Learning Approach“. In: *IEEE Transactions on Industrial Informatics*. DOI: 10.1109/TII.2021.3124848.

Watkins, Christopher und Peter Dayan (1992). „Q-learning“. In: *Machine Learning* 8, S. 279–292.

Weiss, Gary M. und Haym Hirsh (1998). „Learning to Predict Rare Events in Event Sequences“. In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. Bd. 98. New York, NY: AAAI Press, S. 359–363.

Weiss, Gerhard (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*. 1st. MIT press. ISBN: 0262232030.

Werner, Martin (2017). „Information und Codierung“. In: *Nachrichtentechnik: Eine Einführung für alle Studiengänge*. Springer Fachmedien, S. 461–543. DOI: 10.1007/978-3-8348-2581-0_8.

Wiering, Marco und Martijn van Otterlo (2012). *Reinforcement Learning. State-of-the-Art*. Springer. DOI: 10.1007/978-3-642-27645-3.

Williams, Ronald J. (1992). „Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3, S. 229–256. DOI: 10.1007/978-1-4615-3618-5_2.

Wong, Timothy und Zhiyuan Luo (2018). „Recurrent Auto-Encoder Model for Large-Scale Industrial Sensor Signal Analysis". In: *Engineering Applications of Neural Networks*. Hrsg. von Elias Pimenidis und Chrisina Jayne. Springer International Publishing, S. 203–216. ISBN: 978-3-319-98204-5. DOI: 10.1007/978-3-319-98204-5_17.

Wu, Chao, Liyi Zhou, Chulin Xie, Yuhang Zheng und Jiawei Yu (2019). „Data Quality Transaction on Different Distributed Ledger Technologies". In: *Big Scientific Data Management*. Hrsg. von Jianhui Li, Xiaofeng Meng, Ying Zhang, Wenjuan Cui und Zhihui Du. Springer International Publishing, S. 301–318. ISBN: 978-3-030-28061-1.

Wu, Yulei, Hong-Ning Dai und Hao Wang (2021). „Convergence of Blockchain and Edge Computing for Secure and Scalable IIoT Critical Infrastructures in Industry 4.0". In: *IEEE Internet of Things Journal* 8.4, S. 2300–2317. DOI: 10.1109/JIOT.2020.3025916.

Xiong, Xiong, Kan Zheng, Lei Lei und Lu Hou (2020). „Resource Allocation Based on Deep Reinforcement Learning in IoT Edge Computing". In: *IEEE Journal on Selected Areas in Communications* 38.6, S. 1133–1146. DOI: 10.1109/JSAC.2020.2986615.

Yang, Helin, Arokiaswami Alphones, Wen-De Zhong, Chen Chen und Xianzhong Xie (2020). „Learning-Based Energy-Efficient Resource Management by Heterogeneous RF/VLC for Ultra-Reliable Low-Latency Industrial IoT Networks". In: *IEEE Transactions on Industrial Informatics* 16.8, S. 5565–5576. DOI: 10.1109/TII.2019.2933867.

Yang, Yaodong und Jun Wang (2020). „An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective". In: *CoRR* abs/2011.00583. DOI: 10.48550/arXiv.2011.00583.

Ye, Hao, Geoffrey Ye Li und Biing-Hwang Fred Juang (2019). „Deep Reinforcement Learning Based Resource Allocation for V2V Communications". In: *IEEE Transactions on Vehicular Technology* 68.4, S. 3163–3173. DOI: 10.1109/TVT.2019.2897134.

Yildirim, Ozal, Ru San Tan und U. Rajendra Acharya (2018). „An efficient compression of ECG signals using deep convolutional autoencoders". In: *Cognitive Systems Research* 52, S. 198–211. ISSN: 1389-0417. DOI: 10.1016/j.cogsys.2018.07.004.

You, Xinyu, Xuanjie Li, Yuedong Xu, Hui Feng und Jin Zhao (2019). „Toward Packet Routing with Fully-distributed Multi-agent Deep Reinforcement Learning". In: *2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*, S. 1–8. DOI: 10.23919/WiOPT47501.2019.9144110.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov und Alexander J Smola (2017). „Deep Sets“. In: *Advances in Neural Information Processing Systems*. Hrsg. von I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett. Bd. 30. Curran Associates, Inc.

Zhang, Kaiqing, Zhuoran Yang und Tamer Basar (2019). „Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms“. In: *CoRR* abs/1911.10635. DOI: 10.48550/arXiv.1911.10635.

Zhang, Wen, Tao Liu, Mimi Xie, Jun Zhang und Chen Pan (2021). „SAC: A Novel Multi-hop Routing Policy in Hybrid Distributed IoT System based on Multi-agent Reinforcement Learning“. In: *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, S. 129–134. DOI: 10.1109/ISQED51717.2021.9424255.

Zhang, Yuchen, Jason Lee, Martin Wainwright und Michael I. Jordan (Apr. 2017). „On the learnability of fully-connected neural networks“. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Hrsg. von Aarti Singh und Jerry Zhu. Bd. 54. Proceedings of Machine Learning Research. PMLR, S. 83–91. URL: https://proceedings.mlr.press/v54/zhang17a.html.

Zheng, Peilin, Zibin Zheng, Xiapu Luo, Xiangping Chen und Xuanzhe Liu (2018). „A Detailed and Real-Time Performance Monitoring Framework for Blockchain Systems“. In: *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, S. 134–143. ISBN: 978-1-4503-5659-6.

Zhou, Aoying, Zhiyuan Cai, Li Wei und Weining Qian (2003). „M-kernel merging: towards density estimation over data streams“. In: *Eighth International Conference on Database Systems for Advanced Applications, 2003. (DASFAA 2003). Proceedings.* S. 285–292. DOI: 10.1109/DASFAA.2003.1192393.

Zivic, Natasa, Christoph Ruland und Jochen Sassmannshausen (2019). „Distributed Ledger Technologies for M2M Communications“. In: *2019 International Conference on Information Networking (ICOIN)*, S. 301–306. DOI: 10.1109/ICOIN.2019.8718115.

Veröffentlichungen des Autors mit Relevanz für die Dissertation

2021

Rosenberger, Julia, Michael Bühren und Dieter Schramm (2021). „Perspective on efficiency enhancements in processing streaming data in industrial IoT networks“. In: *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. DOI: 10.1109/GCAIoT53516.2021.9693073.

Rosenberger, Julia, Kevin Müller, Andreas Selig, Michael Bühren und Dieter Schramm (2022). „Extended kernel density estimation for anomaly detection in streaming data“. In: *Procedia CIRP* 112. 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 14-16 July 2021, S. 156–161. ISSN: 2212-8271. DOI: 10.1016/j.procir.2022.09.065.

Rosenberger, Julia, Felix Rauterberg und Dieter Schramm (2021). „Performance study on IOTA Chrysalis and Coordicide in the Industrial Internet of Things“. In: *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. DOI: 10.1109/GCAIoT53516.2021.9692985.

Rosenberger, Julia, Michael Urlaub und Dieter Schramm (2021). „Multi-agent reinforcement learning for intelligent resource allocation in IIoT networks“. In: *2021 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. DOI: 10.1109/GCAIoT53516.2021.9692913.

2022

Rosenberger, Julia, Alexander Kübel und Fabian Rothfuß (2022). „Comparison and extension of autoencoder models for uni- and multivariate signal compression in IIoT“. In: *2022 Data Compression Conference (DCC)*, S. 481–481. DOI: 10.1109/DCC52660.2022.00092.

Rosenberger, Julia, Michael Urlaub, Felix Rauterberg, Tina Lutz, Andreas Selig, Michael Bühren und Dieter Schramm (2022). „Deep Reinforcement Learning Multi-Agent System for Resource Allocation in Industrial Internet of Things“. In: *Sensors* 22.11. ISSN: 1424-8220. DOI: 10.3390/s22114099.

Urlaub, Michael und Julia Rosenberger (2022). „Multi-Agent Reinforcement Learning for smart Computing Resource Allocation in the Industry 4.0“. In: *Kölner Beiträge zur Technischen Informatik*.

2023

Rosenberger, Julia, Andreas Selig, Mirjana Ristic, Michael Bühren und Dieter Schramm (2023). „Virtual Commissioning of Distributed Systems in the Industrial Internet of Things“. In: *Sensors* 23.7. ISSN: 1424-8220. DOI: 10.3390/s23073545.

Patentanmeldungen

Rosenberger, Julia, Felix Rauterberg und Michael Urlaub (2023). *Verfahren zum Optimieren von Ressourcen in einem Kommunikationsnetzwerk mittels wenigstens zwei Agenten; Aktenzeichen DE102021213898A1.* Patentanmeldung. angemeldet am: 07.12.2021, offengelegt am: 07.06.2023.

Rosenberger, Julia und Andreas Selig (2023). *Verfahren zum Verarbeiten eines Datensatzes durch ein Internet-of-Things-Gerät; Aktenzeichen DE102021213899A1.* Patentanmeldung. angemeldet am: 07.12.2021, offengelegt am: 07.06.2023.

Rosenberger, Julia, Andreas Selig und Mirjana Ristic (2022a). *Verfahren und System zum Verdichten eines Datensatzes und/oder eines Datenstroms; Aktenzeichen DE102021202009A1.* Patentanmeldung. angemeldet am: 03.03.2021, offengelegt am: 08.09.2022.

– (2022b). *Verfahren und System zur Aufbereitung eines Datensatzes in einem verteilten System und Trainingsverfahren; Aktenzeichen DE102021201966A1.* Patentanmeldung. angemeldet am: 02.03.2021, offengelegt am: 08.09.2022.

Betreute studentische Arbeiten

Studienarbeiten

Rauterberg, Felix (2022b). „Performancevergleich von Datenkompressionsalgorithmen auf industriellen Edge-Devices“. Studienarbeit. Technische Hochschule Mittelhessen.

Bachelorarbeiten

Lutz, Tina (2022). „Evaluation von Optimierungen im Einsatz von Deep Reinforcement Learning Systemen in der Industrie 4.0“. Bachelorarbeit. Technische Hochschule Aschaffenburg.

Müller, Kevin (2021). „Bewertung und Entwicklung von Methoden zur Anomalie Erkennung auf Edge-Ebene“. Bachelorarbeit. Technische Hochschule Aschaffenburg.

Rauterberg, Felix (2022a). „Deep Reinforcement Learning zur Verteilung von Datenflüssen in der Industrie 4.0“. Bachelorarbeit. Technische Hochschule Mittelhessen.

Masterarbeiten

Urlaub, Michael (2021). „Multi-Agent Reinforcement Learning für intelligente Ressourcenallokation in der Industrie 4.0“. Masterarbeit. Technische Hochschule Köln.

DOI: 10.17185/duepublico/81315
URN: urn:nbn:de:hbz:465-20231215-093943-0