Carl von Ossietzky Universität Oldenburg
Abteilung für Digitalisierte Energiesysteme

Vergleich lernender Ansätze für die verteilte Anomalieerkennung in agentenbasierten cyber-physischen Energiesystemen

Masterarbeit

vorgelegt von

Julia Catharina Heiken

Geboren am 25.08.1996 in Oldenburg

Erstprüferin: Prof. Dr.-Ing. Astrid Nieße
Zweitprüferin: M.Sc. Emilie Frost

Oldenburg, den 27. Oktober 2022

Abstract

In modern energy systems, the individual components are intelligently networked by information and communication technologies. Due to the increasing use of renewable energies and new technologies, more distributed, intelligent subsystems (smart grids) are being created, which are controlled and coordinated in a decentralized manner. This can be realized by multi-agent systems, in which the components of the energy system are assigned to individual agents, which can find cooperative solutions for different use cases through communication. As interconnectivity in smart grids increases, so does complexity, as well as potential for threats such as cyber-attacks. The power grid is a critical infrastructure, so a secure energy system is essential for our society. To ensure reliable operation, faults of all kinds must be detected. Anomaly detection with methods from machine learning is suitable for this.

In this work, an observer is developed for distributed anomaly detection in the communication of an agent-based cyber-physical power system. For this purpose, different machine learning methods are investigated, comparing "classical" and graph-based approaches. The "classical" methods used are the Autoencoder, the Isolation Forest and the One-Class Support Vector Machine (OCSVM). The graph-based approach used is the Graph Deviation Network (GDN).

Anomalies in the values of exchanged messages, in communication behavior, and in communication topology are considered. These are examined individually, furthermore a combination of value and topology anomalies is considered.

At least one "classical" approach is suitable for detecting anomalies in values, communication behavior, and one type of topology anomaly. The graph-based approach is suitable for detecting another type of topology anomaly that the "classical" approaches cannot detect. No model can outperform the others for all anomaly types.

Zusammenfassung

In modernen Energiesystemen werden die einzelnen Komponenten durch Informations- und Kommunikationstechnologien (IKT) intelligent vernetzt. Durch die zunehmende Nutzung erneuerbarer Energien und neuer Technologien entstehen immer mehr verteilte, intelligente Subsysteme (Smart Grids), welche dezentral gesteuert und koordiniert werden. Dies kann durch Multi-Agentensysteme (MAS) realisiert werden, indem die Komponenten des Energiesystems einzelnen Agenten zugewiesen werden, welche durch Kommunikation kooperative Lösungen für verschiedene Anwendungsfälle finden können. Mit steigender Vernetzung in Smart Grids wachsen auch die Komplexität sowie das Potential für Bedrohungen wie Cyber-Angriffe. Das Energienetz ist eine Kritische Infrastruktur (KRITIS), dessen Sicherheit für die Gesellschaft essentiell ist. Um einen zuverlässigen Betrieb zu gewährleisten müssen Störungen aller Art erkannt werden. Dafür eignet sich Anomalieerkennung mit Verfahren aus dem Machine Learning (ML).
In dieser Arbeit wird ein Observer zur verteilten Anomalieerkennung in der Kommunikation eines agentenbasierten cyber-physischen Energiesystem entwickelt. Dafür werden verschiedene ML-Verfahren untersucht und dabei "klassische" und graphbasierte Ansätze verglichen. Die verwendeten "klassischen" Verfahren sind der Autoencoder, der Isolation Forest und die OCSVM. Als graphbasierter Ansatz wird das GDN eingesetzt.
Es werden Anomalien in den Werten von ausgetauschten Nachrichten, im Kommunikationsverhalten und in der Kommunikationstopologie betrachtet. Diese werden einzeln untersucht, weiterhin wird eine Kombination von Werte- und Topologieanomalien berücksichtigt.
Mindestens ein "klassisches" Verfahren eignet sich für die Erkennung von Anomalien in den Werten, im Kommunikationsverhalten sowie für eine Art von Topologieanomalie. Der graphbasierte Ansatz ist geeignet für die Erkennung einer anderen Art von Topologieanomalie, welche die "klassischen" Ansätze nicht erkennen. Es kann allerdings kein Modell die anderen für alle Anomaliearten übertreffen.

Inhaltsverzeichnis

Abkürzungen

IKT	Informations- und Kommunikationstechnologie
CPS	Cyber-physisches System
KRITIS	Kritische Infrastruktur
MAS	Multi-Agentensystem
SCADA	Supervisory Control and Data Aquisition
OC	Organic Computing
ML	Machine Learning
OCSVM	One-Class Support Vector Machine
KNN	Künstliches Neuronales Netz
FCN	Fully Connected Network
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
GDN	Graph Deviation Network
TP	True Positives
TN	True Negatives
FN	False Negatives
FP	False Positives
TPR	True Positive Rate
FPR	False Positive Rate
TNR	True Negative Rate

ROC	Receiver Operating Characteristics
AUC	Area Under Curve
RBF	Radial Basis Function
GNN	Graph Neural Network
MSE	Mean Squared Error
SuOC	System under Observation/Control
PCA	Principal Component Analysis
ARIMA	Auto Regressive Integrated Moving Average
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
CAE	Convolutional Autoencoder
LOF	Local Outlier Factor
k-NN	k Nearest Neighbors
TANN	Triangle Area based Nearest Neighbors
SA-Isolation Forest	Simulated Annealing Isolation Forest
RF	Random Forest
LAN	Local Area Network
FB	Feature Bagging
VAE	Variational Autoencoder
MAD-GAN	Multivariate Anomaly Detection with Generative Adversarial Network
DAGM	Deep Autoencoding Gaussian Model
DL	Deep Learning
MIRAGE	Multi-Purpose Battery Storage Swarm

Formelzeichen

Machine Learning

a	Aktivierung eines Neurons
b	Bias
o	Output eines Neurons
w	Gewicht, mit dem der Input eines Neurons multipliziert wird
x	Input eines Neurons
$\vec{x}$	Feature-Vektor
y	Tatsächlicher, gewünschter Output (Label)
z	Gewichteter Input eines Neurons
$\mathcal{L}$	Kostenfunktion eines Künstliches Neuronalen Netzes
δ	Fehler eines Neurons
ϵ	Reconstruction Error (Autoencoder)
η	Lernrate des Backpropagation-Algorithmus
ν	Obere Grenze für Anteil an Anomalien und untere Grenze für Anteil der Support Vektoren (One-Class Support Vector Machine)
σ	Aktivierungsfunktion eines Neurons
θ	Entscheidungsgrenze zum Feststellen von Anomalien (Autoencoder)

1 Einleitung

Die Einleitung in diese Arbeit beginnt mit der Motivation des Themas. Anschließend wird die Problemstellung behandelt, um daraus die Zielsetzung der Arbeit abzuleiten und zu formulieren. Weiterhin wird der Aufbau der Arbeit beschrieben.

1.1 Motivation

Um den Klimawandel zu verlangsamen soll durch den Kohleausstieg bis 2038 der CO_2- Ausstoß deutlich verringert werden [1]. Das gesamte Energieversorgungssystem durchlebt derzeit einen Wandel, in dem die großen, umweltbelastenden Kraftwerke durch effizientere, dezentrale erneuerbare Energiequellen wie beispielsweise Wind-, Wasser- und Solarkraftanlagen ersetzt werden. Die Energiegewinnung bei erneuerbaren Quellen ist hauptsächlich vom Wetter abhängig. Da Sonne und Wind allerdings nicht kontinuierlich verfügbar sind, entstehen Schwankungen und Abweichungen zwischen Erzeugung und Verbrauch. Mit der Wende von der fossilen zur regenerativen Energieerzeugung wächst die Notwendigkeit, die entstehenden vielen kleinen Systeme zu koordinieren und die schwankende Verfügbarkeit der Energie auszugleichen. Um dabei ein stabiles Energienetz sicherzustellen, werden neue Speicherkonzepte und intelligente Lösungen für das Energiemanagement benötigt [1, 2].

Die derzeit angestrebte Lösung sind intelligente Stromnetze (Smart Grids), in denen die einzelnen Komponenten durch Systeme für Energiemanagement mithilfe von Informations- und Kommunikationstechnologien (IKT) koordiniert werden. Somit werden zusätzlich zur Energieübertragung auch Informationen darüber ausgetauscht. Durch das Kommunikationsnetz können Netzbetreiber Informationen erhalten, die notwendig sind, um eine automatisierte Koordination von Transport und Verteilung der Energie zu bewerkstelligen. Durch die Kommunikation der dezentralen Systeme über die Erzeugung, Speicherung und den Verbrauch können intelligente Lösungen für eine möglichst effiziente Energienutzung gefunden werden. Auch die wetterabhängigen Leistungsschwankungen können ausgeglichen werden, indem durch die gewonnenen Informationen eine entsprechende Umverteilung der Energie veranlasst wird [3, 4]. Somit können die entstehenden Energiesysteme als Cyber-physische Systeme (CPS) betrachtet werden: Elektro-mechanische Komponenten mit IT-Schnittstellen und IT-Funktionalitäten, die über ein Kommunikationssystem miteinander

verbunden sind. CPS bilden eine neue Forschungsgrundlage für große, verteilte, komplexe Systeme [5].

Eine Möglichkeit Smart Grids zu koordinieren sind Multi-Agentensysteme (MAS). Ein Agent ist ein Computersystem, das sich in einer Umgebung befindet und in dieser autonom agieren kann, um ein bestimmtes definiertes Ziel zu erreichen. Der Input sind Wahrnehmungen durch Sensoren und der Output ist eine gewählte Aktion des Agenten, um seine Umgebung zu verändern [6]. Die Eigenschaften von Agenten werden in Abschnitt 2.2 erläutert. Ein MAS ist ein System aus mehreren Agenten, die miteinander kommunizieren und interagieren können [6]. Zur Realisierung von Smart Grids werden einzelne Komponenten des Energienetzes einzelnen Agenten zugewiesen, die sich dann zu einem MAS zusammenschließen und beispielsweise über die Energieverteilung verhandeln und kooperative Lösungen finden können.

Um neue Erkenntnisse in dem Forschungsfeld von Smart Grids zu erhalten bietet sich Simulation an. Die Komplexität ist so hoch, dass die internen Abhängigkeiten nicht mehr analytisch in angemessener Zeit und an realen physischen Systemen analysiert werden können [7]. Ein weiterer Grund für Simulation ist, dass Komunikationsnetze Teil des modernen Energiesystems werden und gut in Simulationen untersucht werden können [8]. In einem simulierten Smart Grid können agentenbasierte Kontrollsysteme mit verschiedenen Algorithmen, in denen Agenten des MAS kontrollierend auf die simulierte physikalische Umwelt einwirken, hinsichtlich unterschiedlicher Forschungsfragen untersucht werden [6].

1.2 Problemstellung

Ein wichtiger Bereich der Entwicklung von Smart Grids ist Cyber-Resilienz. Da das Energieversorgungssystem zu den Kritischen Infrastrukturen (KRITIS) zählt, sind Ausfallsicherheit und eine zuverlässige Versorgung von höchster Relevanz [9]. Da in einem Smart Grid nicht nur Energie, sondern auch Informationen ausgetauscht werden, ist für ein intaktes, robustes Energienetz ebenso ein zuverlässiges und sicheres Kommunikationsnetz notwendig [10].

Die wachsende Verwendung neuer Technologien wie beispielsweise Energiespeichern oder Elektroautos, welche in das Smart Grid integriert werden, sorgt für eine steigende Komplexität und schafft somit immer mehr Angriffsfläche für Bedrohungen, welche die Sicherheit des Energienetzes gefährden [11]. Neben natürlichen, physischen Störungen wie Naturkatastrophen oder Defekte an physischen Komponenten können Cyber-Angriffe die Aufrechterhaltung des intakten Energienetzes bedrohen. Während eines solchen Angriffs können beispielsweise Daten gestohlen oder falsche Daten injiziert werden, welche ein unerwünschtes Verhalten auslösen. Dieses fehlerhafte Verhalten kann zu Betriebsausfällen und schließlich zu einer (Teil-) Unterbrechung der Energieversorgung führen. Es ist also

von größter Bedeutung, jegliche Art von Gefahr zu erkennen und abzufangen [12].

Cyber-Angriffe bewirken Störungen im Energiesystem, welche sich als Anomalien in den physischen und den Kommunikationsdaten auswirken. Anomalien sind Muster in Daten, die von definiertem Normalverhalten abweichen [13]. Um Angriffe zu erkennen, ihnen entgegenzuwirken und das CPS zu schützen, ist eine Erkennung solcher Abweichungen von normalem Verhalten notwendig. Dafür kann Anomalieerkennung eingesetzt werden [11].

Die dezentrale Verortung der einzelnen Komponenten des Smart Grids sowie die Verknüpfung der physischen mit den Kommunikationsdaten erschweren die gleichzeitige Überwachung des Gesamtsystems. Durch die verteilte Steuerung und Kontrolle werden viele Angriffspunkte geschaffen, an denen unbefugt eingegriffen werden kann. Somit müssen die verschiedenen Komponenten einzeln geschützt werden, um die Sicherheit für das gesamte Smart Grid zu gewährleisten [14].

1.3 Ziel

Diese Arbeit beschäftigt sich mit der Entwickung einer erweiternden Komponente für einzelne Agenten eines MAS, welche zur Erkennung von Anomalien in den Kommunikationsdaten dient. Anomalieerkennung hat zum Ziel Unregelmäßigkeiten und Ausreißer in Daten sowie Abweichungen von Normalverhalten zu finden [13]. In der Kommunikation eines MAS können verschiedene Arten von Anomalien auftreten. Beispielsweise können in ausgetauschten Nachrichten Ausreißer in den Energiewerten auftreten. Weiterhin könnte ein Agent dahingehend manipuliert werden, ein fehlerhaftes Kommunikationsverhalten zu zeigen, wie beispielsweise das Versenden vieler Nachrichten in kurzer Zeit bei einem Denial-of-Service-Angriff. Darüber hinaus ist eine Manipulation denkbar, durch welche ungewünschte Änderungen in der Kommunikationstopologie auftreten. Ein manipulierter Agent könnte beispielsweise bestimmte Nachbarn ignorieren und keine Nachrichten mehr an diese versenden.

In dem betrachteten MAS soll jeder Agent einen bestimmten Anwendungsfall erfüllen, zu dessen Einhaltung er Verhandlungen mit anderen Agenten anstoßen kann. Diese übernehmen dann die vom ersten Agenten nicht erfüllbaren Verpflichtungen, sodass global gesehen jeder Agent durch kooperative Lösungen seinen Anwendungsfall erfüllt. Bei der Simulation der Verhandlungen im betrachteten Energiesystem kommt das Kommunikationsprotokoll Winzent zum Einsatz. Darin verhandeln Agenten über den Austausch von Energie nach definierten Kommunikationsregeln. Mithilfe von Kurzzeitvorhersagen über den eigenen Energiebedarf wird von jedem Agenten berechnet, ob er für seine zuständige Netzkomponente Energie benötigt oder abgeben kann. Dazu werden entsprechende Nachrichten ausgetauscht. Daraus entsteht ein selbstorganisiertes Netz aus kooperierenden Agenten, in

dem die Energie effizient genutzt wird [15].

Zur Umsetzung der ständigen Überwachung und Kontrolle des Energiesystems wird eine dezentrale Observer/Controller-Architektur im Kontext von Organic Computing (OC) angestrebt, bei der jede Komponente einen Agenten erhält, der jeweils von einem Observer zur Anomalieerkennung aus dezentraler Sicht und einem Controller zur Steuerung und Reaktion auf Unregelmäßigkeiten überwacht wird. In dieser Arbeit wird ein Observer entwickelt, für den verschiedene Verfahren zur Anomalieerkennung eingesetzt und verglichen werden. Der zugehörige Controller ist allerdings nicht Teil der Arbeit.

Für Anomalieerkenung gibt es eine Vielzahl an möglichen Algorithmen, die auf verschiedenen mathematischen Methoden basieren. Die in der Arbeit als klassisch bezeichneten Algorithmen nutzen Vorhersagen über das normale Verhalten anhand der bisherigen erlernten Muster oder verwenden statistische Ansätze um anomales Verhalten zu erkennen [16]. Es gibt des Weiteren Methoden, die die vorliegenden Daten als Graphen interpretieren, was der Natur von Komunikationsnetzen nah ist [17]. Daher sind graphbasierte Verfahren vielversprechend für die Erkennung von Anomalien in Kommunikationsverhalten und -topologien. Es gibt außerdem weitere Möglichkeiten, Daten eines Energiesystems als Graphen zu repräsentieren. In [18] werden beispielsweise Messpunkte von Sensoren als Knoten und die Korrelationen der Sensormessungen als gewichtete Kanten eines Graphen interpretiert. Anhand der gelernten Abhängigkeiten können durch einen Vergleich von berechneten Vorhersagen mit den tatsächlichen Werten Abweichungen als Anomalien erkannt werden.

Das Ziel der Masterarbeit ist ein Vergleich von klassischen und graphbasierten Ansätzen zur verteilten Anomalieerkennung in MAS für verschiedene Arten von Anomalien.

Grundlage der Untersuchungen bilden Ergebnisse aus Vorarbeiten mit synthetischen Daten sowie eine vorausgegangene Masterarbeit ([19]), in welcher ebenfalls ein Observer zur Anomalieerkennung in einem MAS entwickelt wurde, jedoch aus zentraler Sicht.

Die folgenden Forschungsfragen sollen mit den Ergebnissen dieser Arbeit beantwortet werden:

- Wie gut eignen sich klassische Machine Learning (ML) Verfahren für die Erkennung von verschiedenen Arten von Anomalien in der Kommunikation eines Winzent-basierten MAS für die Koordination von dezentralen Energiespeichern?
- Eignen sich graphbasierte ML Verfahren besser für die Erkennung bestimmter Anomalien in der Kommunikation eines Winzent-basierten MAS für die Koordination von dezentralen Energiespeichern, die von den klassischen Ansätzen weniger gut erkannt werden?

Die Datengrundlage der Arbeit bilden das Masterprojekt von Emilie Frost [19] sowie das Projekt Multi-Purpose Battery Storage Swarm (MIRAGE) vom OFFIS [20]. Dort

wird ein MAS betrachtet, welches für die Verwaltung und Steuerung eines Systems aus vernetzten Energiespeichern eingesetzt wird. Jeder Agent steuert einen Energiespeicher und verfolgt dabei das Ziel einer Multi-Purpose-Nutzung, wobei der Ausgleich von kurzfristigen Leistungsschwankungen im Vordergrund steht. Dies wird durch Verhandlungen über einen Energieaustausch ermöglicht. Darüber hinaus zur Verfügung stehende Freiheitsgrade werden an Flexibilitätsmärkten vermarktet, um das Speichersystem wirtschaftlich und technisch optimal zu nutzen [20]. Die ausgetauschten Verhandlungsnachrichten werden als Daten zur Verfügung gestellt und enthalten sowohl Informationen über die Kommunikation (Zeit, Sender, Empfänger, Nachrichtentyp und -größe) als auch relevante physikalische Größen über den Batteriezustand. Für die Injizierung von Anomalien in die Daten werden die Agenten so manipuliert, dass sie ein Fehlverhalten aufweisen. Es ist möglich verschiedene Arten von Anomalien zu erzeugen, die dann in der Anomalieerkennung analysiert werden können.

Das grobe Vorgehen der Arbeit ist wie folgt: Es werden Anomalien in den Werten der Nachrichten sowie in der Kommunikationstopologie und im Kommunikationsverhalten des Agentensystems aus dem in [19] definierten Szenario betrachtet. Diese sind sowohl einzeln als auch in Kombination zu untersuchen. Dafür sind aus den vorhandenen Normaldaten, Daten mit Anomalien in den Werten und dem Kommunikationsverhalten geeignete Datensätze auszuwählen und für das Setting einer verteilten Anomalieerkennung anzupassen. Datensätze mit Anomalien in der Kommunikationstopologie sowie einer Kombination aus mehreren Anomaliearten müssen für diese Arbeit erzeugt werden. Für den Vergleich werden drei klassische Algorithmen und mindestens ein graphbasierter Ansatz verwendet. Die Verfahren müssen jeweils für jede Art von Anomalie und für die Kombination implementiert werden. Da mit Simulationsdaten gearbeitet wird, bei denen das Ergebnis bekannt ist, können die gängigen Metriken zur Bewertung von Machine Learning Verfahren genutzt werden. Es handelt sich um eine Klassifizierungsaufgabe mit zwei Klassen (Anomalie/normaler Datenpunkt). Somit soll die Bewertung anhand von *Confusion Matrizen* stattfinden, aus denen sich weitere Kennzahlen wie Genauigkeit (Accuracy), Präzision (Precision) und Sensitivität (Sensitivity) berechnen lassen [21].

1.4 Aufbau der Arbeit

Die Masterarbeit beginnt nach dieser Einleitung mit der Präsentation aller relevanten Grundlagen in Kapitel 2. Anschließend werden in Kapitel 3 die geleisteten Vorarbeiten beschrieben sowie die Ergebnisse präsentiert und diskutiert. In dem darauf folgenden Kapitel 4 wird das Konzept entwickelt und vorgestellt. Daran anknüpfend wird in Kapitel 5 die Umsetzung beschrieben. In Kapitel 6 werden die Ergebnisse präsentiert, welche

dann in Kapitel 7 evaluiert werden. Abschließend werden Fazit und Ausblick in Kapitel 8 beschrieben.

2 Theoretische Grundlagen

In dem folgenden Kapitel werden die relevanten Grundlagen zum Verständnis von Anomalieerkennung in agentenbasierten cyber-physischen Systemen (CPS) präsentiert. Dazu wird zunächst erläutert, was ein Cyber-physisches System (CPS) ist und wie Multi-Agentensysteme (MAS) dort zum Einsatz kommen können. Anschließend wird das Prinzip des Organic Computing vorgestellt. Weiterhin wird in die Themen Anomalien, Machine Learning und Anomalieerkennung in Zeitreihen eingeführt, wobei die unterschiedlichen lernenden Ansätze erklärt werden. Abschließend werden mit dem Masterarbeitsthema verwandte Arbeiten aufgeführt, sowie relevante Entwicklungsumgebungen und Bibliotheken für Anomalieerkennung mit Machine Learning (ML).

2.1 Cyber-physische Systeme

In gängigen technischen Systemen gibt es häufig eine Trennung zwischen dem Kontrollsystem und Hardware/Software-Implementierungsdetails. Das Kontrollsystem wird durch aufwendige Simulation verifiziert und gegen Unsicherheiten und zufällige Störungen getestet. Die Integration neuer Subsysteme ist dabei zeitaufwendig und teuer. In modernen Systemen werden die Komponenten immer komplexer und die hochentwickelten Technologien für Sensorik und Aktorik sowie drahtlose Kommunikation und höhere Rechenleistung stellen eine Herausforderung dar. Um unabhängig entwickelte Subsysteme kosteneffizient integrieren zu können, müssen die verschiedenen Bereiche der Computertechnik (Vernetzung, Steuerung, Software, menschliche Interaktion, Lerntheorie) und des Ingenieurwesens (Elektrotechnik, Mechanik, Chemie, Biomedizin, Materialwissenschaften etc.) verbunden werden. Hier setzt die Forschung über CPS an [22].

Ein physisches System aus Sensoren, Aktoren und weiterer Hardware sammelt Daten und interagiert mit dem Cyber-System. Dieses verarbeitet und analysiert die Daten, führt Berechnungen durch und generiert Aktionssignale, die es über ein Kommunikationsnetzwerk an das physische System weiterleitet. Dabei ist das Kommunikationsnetz das Herzstück des Systems, welches das physische mit dem Cyber-System verbindet und für den Datenaustausch zuständig ist. Es muss einen sicheren, geschützten, zuverlässigen und ununterbrochenen Service bieten, um das System durchgehend lauffähig zu erhalten [23].

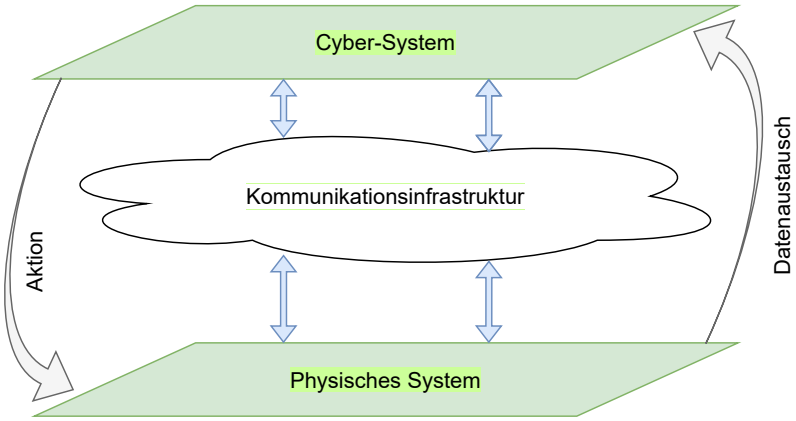


Abbildung 2.1: Cyber-Physisches System. Eigene Abbildung nach [23].

Abbildung 2.1 zeigt schematisch, wie die Kommunikationsinfrastruktur das physische und das Cyber-System verbindet. Aus der kontinuierlichen Kommunikation und Interaktion ergibt sich ein intelligenter Regelkreis, der sich anpassen kann, autonom ist (d.h. ohne menschliches Eingreifen läuft) und die Effizienz in seinem Aufgabenbereich verbessert [24].

Beispiele für CPS sind, neben Smart Grids, industrielle Kontrollsysteme (Industrie 4.0: enge Integration von Informationssystemen, Unternehmensdaten und computergestützten Produktionssystemen), computergesteuerte Fahr- und Flugzeugsysteme oder medizinische Geräte wie moderne Herzschrittmacher. CPS wie das Energieversorgungssystem zählen zu den modernen Kritischen Infrastrukturen (KRITIS) und stellen daher einen wichtigen Forschungszweig dar [24].

Das Ziel der Smart-Grid-Entwicklung ist eine intelligente Integration des Verhaltens und der Aktionen aller Beteiligten in der Energieversorgungskette, um effizient und zuverlässig nachhaltige und wirtschaftliche Energie zu liefern. Durch Nutzung von CPS-Technologien kann dies erreicht werden: Eine effektive und effiziente Interaktion und Integration von Cyber- und physischen Systemen. Dabei bilden Informationserfassung und -verarbeitung, Intelligenz und Kontrolle die Cybersysteme und die Stromnetzinfrastrukturen die physischen Systeme [25]. Die Überbrückung zwischen Cyberwelt und physischer Welt wird durch eine Vielzahl von Sensor- und Aktornetzwerken, die unter einem intelligenten Entscheidungssystem integriert sind, umgesetzt [26]. Diese Netzwerke können als MAS realisiert werden, welche im folgenden Abschnitt (2.2) vorgestellt werden.

2.2 Multi-Agentensysteme

MAS gehören zum Feld der verteilten künstlichen Intelligenz und bestehen aus Agenten, die miteinander kommunizieren und interagieren. Ein Agent ist eine Hardware- oder Softwarebasierte Einheit, die ihre Umgebung durch Sensoren wahrnimmt und durch Aktoren mit dieser Umwelt, die ebenfalls weitere Agenten beinhalten kann, interagiert (siehe Abbildung 2.2). Dabei werden anhand der Sensor-Inputs rational und autonom Aktionen vorgenommen, um den gewünschten Einfluss auf die Umgebung zu erzielen. Das bedeutet, dass der Agent durch seine Aktionen versucht ein bestimmtes Ziel möglichst gut zu erfüllen (Rationalität) und ohne äußeren Einfluss agieren kann (Autonomie) [27].

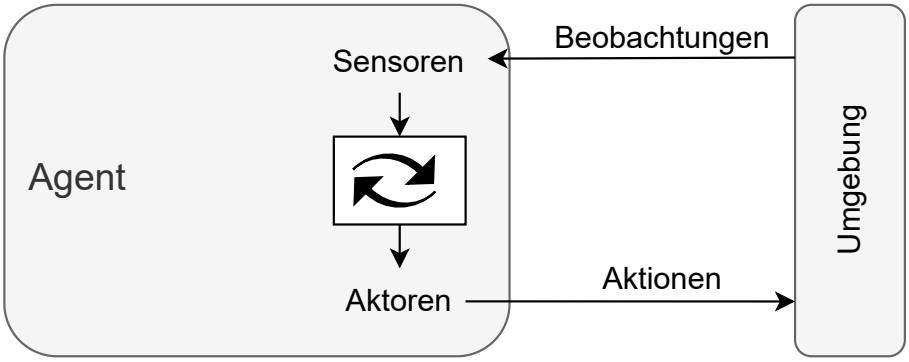


Abbildung 2.2: Ein Agent mit Sensoren und Aktoren in Interaktion mit seiner Umgebung. Eigene Abbildung nach [27].

Ein Agent hat maximal teilweise Kontrolle über seine Umgebung, in dessen Rahmen er sie beeinflussen kann. Die Umgebung ist allerdings im Allgemeinen nicht-deterministisch, d.h. die gleiche Aktion unter gleichen Umständen kann einen völlig anderen Effekt haben. Das Kernproblem besteht für den Agenten darin, aus seinem definierten Repertoire an Aktionen diejenige auszuwählen, die ihn seinem Ziel am meisten näherbringt [6].

Es kann in intelligenten Systemen verschiedene Arten von Agenten geben, welche die folgenden Eigenschaften besitzen [28]:

- Reflexive Agenten:
  - Autonomie: Agieren ohne direkten äußeren Einfluss, Kontrolle über Aktionen und internen Zustand
  - Soziale Fähigkeiten: Kommunikation über eine definierte Sprache
  - Reaktivität: Der Agent nimmt seine Umgebung wahr und reagiert in einem zeitlichen Zusammenhang auf auftretende Veränderungen.

- Deliberative Agenten: Reflexive Agenten mit den zusätzlichen Eigenschaften
  - Kognitive Fähigkeiten: Der Agent kann Wissen über die Umgebung intern abbilden.
  - Proaktivität: Der Agent kann ohne äußeren Trigger Aktionen vornehmen.

MAS bestehen aus mehreren reflexiven oder deliberativen Agenten, die miteinander kommunizieren und interagieren. Jeder Agent hat dabei Einfluss auf den ihm zugewiesenen Teil der Umgebung und kann die Handlungen der anderen Agenten, welche die Umgebung beeinflussen, zum Erreichen seines Ziel einbeziehen. Die gesamte Veränderung der Umwelt ergibt sich aus den einzelnen Aktionen aller Agenten und verfolgt beispielsweise in kooperierenden Systemen zudem ein übergeordnetes Ziel [6].

In Smart Grids können MAS eingesetzt werden, um die vielen dezentralen Energieanlagen zu kontrollieren und zusammenzuarbeiten, damit die beste und sicherste Konfiguration des Stromnetzes erreicht wird. Durch die kleinformatige Stromerzeugung und -verteilung werden die Endnutzer mit ihren lokalen Managementsystemen zu aktiven Beteiligten am freien Energiemarkt. Agenten können dort eingesetzt werden und in Echtzeit Aufgaben wie Optimierung der Ressourcen, aufwendige Berechnungen und Entscheidungsfindung übernehmen. Sie agieren intelligent und selbstorganisiert in einer integrierten Energiesystemumgebung, die Stromerzeuger, Umspannwerke, Transformatoren und (Hochspannungs-) Stromleitungen umfasst. Somit können sie die Supervisory Control and Data Aquisition (SCADA) übernehmen und gleichzeitig autonome und intelligente Eigenschaften einbringen, um einen zuverlässigen Betrieb sicherzustellen und gleichzeitig die Interessen ihres zugeordneten Bereichs zu erfüllen [29].

Für konkrete Szenarien einer agentenbasierten Steuerung von Smart Grids kann das in [8] vorgestellte Konzept verwendet werden. Dort wird jede Netzkomponente von einem intelligenten Agenten gesteuert, der die zukünftige Leistung (und somit Gleichgewicht oder Ungleichgewicht des Energienetzes an lokaler Stelle) mithilfe historischer Daten vorhersagt. Anhand der Vorhersage kann jeder Agent bei einem Ungleichgewicht den anderen Agenten kommunizieren, wie viel Energie er selbst von anderen benötigt oder abgeben kann. Durch Zusammenarbeit soll das Agentensystem für einen Ausgleich von Angebot und Nachfrage sowie eine effiziente Nutzung der erneuerbaren Energiequellen sorgen. Für die Kommunikation verwenden die Agenten das Protokoll Winzent, welches nachfolgend vorgestellt wird.

2.2.1 Winzent

Winzent ist ein Kommunikationsprotokoll für Verhandlungen über den Austausch von Energie zwischen Agenten in Smart Grids. Jeder Agent besitzt eine Vorhersagekomponente für seinen zukünftigen Energiebedarf. Wird ein Ungleichgewicht vorhergesagt, kann der Agent eine Verhandlung über einen Ausgleich starten. Die Agenten stellen die Knoten im Netz dar und tauschen über bidirektionale Kommunikationskanäle Daten aus. Die Verbindungen sind Ende-zu-Ende, also existiert je ein solcher Kanal zwischen genau zwei Knoten (Endpunkten). Das Kommunikationsprotokoll erzeugt eine Kommunikationsstruktur, die dem tatsächlichen Stromnetz ähnelt, aber nicht exakt passen muss [15].

Die Kommunikation erfolgt in Form von Nachrichten, die ausgetauscht werden. Eine Verhandlung wird gestartet, indem ein Agent durch eine *Demand Notification* eine Anfrage nach der benötigten Energiemenge und -art (Wirkleistung oder Blindleistung in Kilowatt bzw. KiloVar) an seine Nachbarn sendet. Die anderen Agenten können diese mit einer *Offer Notification* beantworten, wenn sie in der Lage sind zur Lösung des Problems beizutragen. Es ist auch möglich, dass ein Agent einen Energieüberschuss vorhersagt und daher mit einer *Offer Notification* startet, worauf er *Demand Notifications* als Antwort erhält. Nachdem der Agent Antworten auf seine Anfrage erhalten hat, berechnet er, ob und wie er mit der angebotenen Energie sein Defizit oder seinen Überschuss ausgleichen kann. Hat er eine Lösung gefunden, muss er diejenigen Agenten, die er in seine Lösung einbezieht, mit einer *Acceptance Notification* benachrichtigen. Diese müssen wiederum dem ersten Agenten durch eine *Acceptance Acknowledgement Notification* bestätigen, dass sie das Angebot einhalten. Dieser *Four-Way-Handshake* (siehe Abbildung 2.3) stellt einen kurzzeitigen Vertrag zwischen den beteiligten Agenten für einen festgelegten Zeitraum dar [8].

Nachrichten können auch weitergeleitet werden, damit auch Knoten zur Lösung beitragen können, die kein direkter Nachbar des anfragenden Knotens sind. Die Nachricht wird dann neu versendet an alle Nachbarn (außer dem ursprünglichen Absender) mit etwaigen Modifizierungen wie beispielsweise einer angepassten Energiemenge, wenn der weiterleitende Knoten einen Teil der angefragten Energie abgeben kann. Die Nachrichten beinhalten die nötigen Informationen um die Nachricht entsprechend zu verarbeiten. Die folgenden Felder sind für alle Arten von Nachrichten anzugeben [15]:

- Nachrichten ID (ID): Jede Nachricht muss eine eindeutige Identifizierungsnummer haben.
- Nachrichtentyp (type): Die Art der Nachricht, beispielsweise *Demand Notification*.
- Sender ID (sender): Der Originalsender der Nachricht.

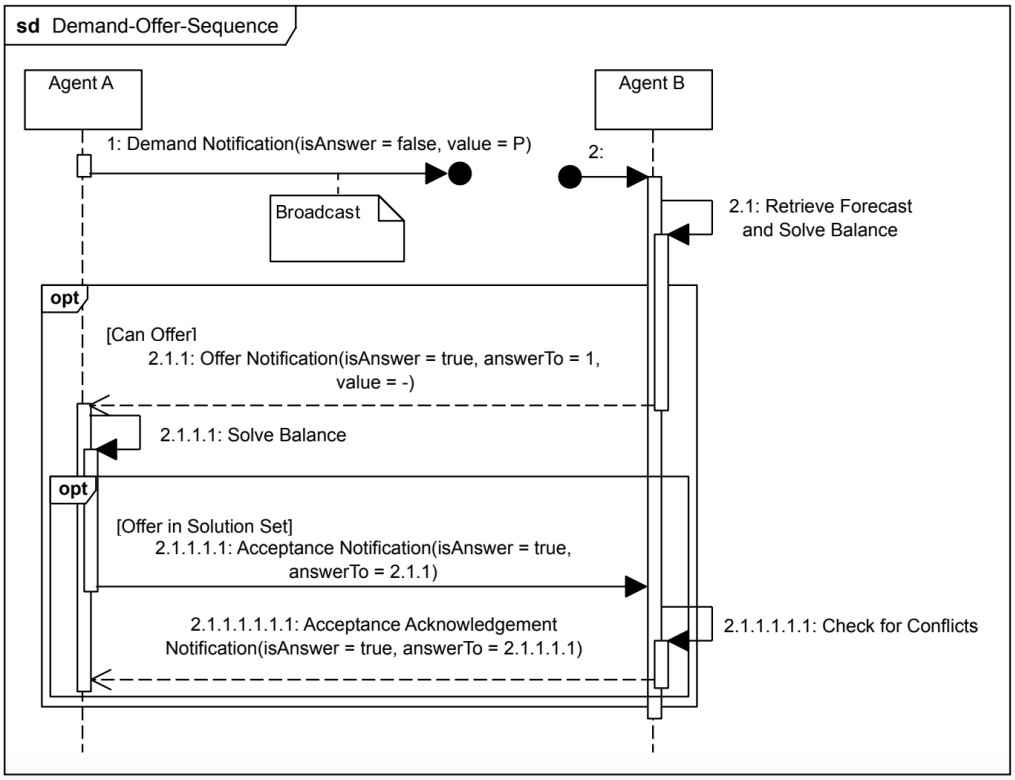


Abbildung 2.3: Four-Way Handshake einer Demand-Offer-Sequenz [8].

- Receiver ID (receiver): Der Empfänger der Nachricht.
- Zeitstempel (sent): Der Zeitpunkt, an dem die Nachricht versendet wurde.
- Time To Live (TTL): Damit eine Nachricht nicht endlos weitergeleitet wird, startet die TTL bei einer Zahl größer 0 und wird mit jedem Weiterleiten oder Senden um 1 heruntergezählt, bis sie bei 0 verworfen wird.
- Hop Count (hops): Das Gegenteil der TTL; startet bei 0 und wird mit jedem Weiterleiten hochgezählt. Der Hop Count wird genutzt um die Distanz zwischen zwei Knoten in Bezug auf Knoten dazwischen zu bestimmen.
- isResponse: Ein Boolean, der angibt, ob es sich bei der Nachricht um eine Antwort handelt. Dies ist wichtig zur Unterscheidung der ursprünglichen Anfrage von den Antworten.
- Wenn isResponse = true gesetzt ist, müssen zusätzlich replyTo (die Originalnachricht) und received (der Zeitstempel, an dem die Nachricht erhalten wurde) übergeben werden.

Abhängig vom Nachrichtentyp gibt es weitere Werte, die in der Nachricht angegeben werden müssen. Die verschiedenen Arten von Nachrichten sind [15]:

- *Null Message*: Beinhaltet nur die Basisfelder und kann als eine Art Herzschlaginformation genutzt werden, welche in einem regelmäßigen Zeitabstand gesendet wird.
- *Echo Request Message*: Kann gesendet werden um zu prüfen, ob der Endpunkt erreichbar ist.
- *Echo Reply Message*: Die Antwort auf eine *Echo Request Message*.
- *Online Notification*: Ein Knoten kann dadurch seine Nachbarn darüber informieren, dass er online ist oder zu einem bestimmten Zeitpunkt in der Zukunft online sein wird. Hier müssen zusätzlich die Felder validFrom und validUntil angegeben werden. Die Nachricht kann, muss aber nicht weitergeleitet werden.
- *Offline Notification*: Wird gesendet, wenn ein Knoten offline geht oder gehen wird und benötigt die gleichen Felder wie die *Online Notification*. Diese Art von Nachricht muss weitergleitet werden, da es Einfluss auf die Berechnungen der anderen Knoten hat, wenn ein Knoten offline geht.
- *Demand Notification*: Diese Nachricht gibt an, dass der Senderknoten ein Energiedefizit hat und Hilfe anfragt. Neben den validFrom und validUntil Feldern wird die benötigte Energiemenge in Watt (power) sowie die Art der Energie (power type) übergeben. Außerdem muss ein Zeitintervall (timespan) angegeben werden, für welches die Verhandlung gültig ist. Die Nachricht muss weitergeleitet werden, wenn ein Knoten die Anfrage nicht oder nur teilweise erfüllen kann.
- *Offer Notification*: Diese Nachricht zeigt an, dass der Senderknoten dem Netz Energie anbieten kann. Sie enthält die gleichen Felder wie eine *Demand Notification*, wobei power die angebotene Energiemenge ist.
- *Offer Accepted Notification*: Nachdem der anfragende Knoten die Antworten erhalten hat, berechnet er eine Lösung und sendet allen Nachbarn diese Nachricht, die diejenigen IDs enthält, deren Angebot der Knoten annehmen möchte.
- *Offer Acceptance Acknowledgement*: Jeder anbietende Knoten sendet dies als Antwort auf eine *Offer Accepted Notification*, um anzuzeigen, dass das Angebot noch gültig ist.
- *Offer Withdrawal Notification*: Falls ein Knoten eine bestimmte Energiemenge angeboten hat, dies aber durch Änderungen in der Zwischenzeit nicht mehr einhalten

kann, muss er mit dieser Nachricht sein Angebot zurückziehen. Es muss die ID des ursprünglichen Angebots in dem ReplyTo-Feld angegeben werden.

Darüber hinaus gibt es noch weitere Kommunikationsregeln. Nachrichten dürfen nicht ignoriert werden (”no ignores”-Regel). Alle Nachrichten außer *Null Message*, *Echo Request Message* und *Echo Response Message* müssen weitergeleitet/teilweise beantwortet und weitergeleitet/beantwortet werden (”match or forward”-Regel). Jeder Knoten muss die erhaltenen Nachrichten speichern, sodass Duplikate erkannt werden und nicht erneut reagiert wird (”no duplicates”-Regel) [15].

2.3 Organic Computing

Eine Möglichkeit zur ständigen Überwachung und Steuerung der Komponenten des Energiesystems ist die Verwendung von Organic Computing (OC). Das Ziel von OC ist die Entwicklung von Lösungen zukünftiger hochkomplexer Informations- und Kommunikationssysteme und Konzepten zu deren Beherrschung. Hohe Komplexität entsteht durch die zunehmende Anzahl von Komponenten und Subsystemen sowie wechselseitige Abhängigkeit durch stärkere Vernetzung [30]. In diesen Systemen interagieren viele Komponenten und das Verhalten der einzelnen Teile sagt nur wenig über das Gesamtverhalten aus [31]. Daher wird es prinzipiell unmöglich, exakte Vorhersagen über alle künftigen Ziele und potentielle Störungen zu treffen [30]. Die Komplexität kann durch Selbstorganisation beherrscht werden, was allerdings zu neuem und unbekanntem Verhalten führen kann. Es gibt also auch die Notwendigkeit von Kontrollaktionen, die das System in definierten Grenzen halten [31].

Hierfür beinhaltet das Konzept von OC eine generische Observer/Controller-Architektur (siehe Abbildung 2.4) zur gesteuerten Selbstorganisation technischer Systeme. ML-Verfahren spielen hier häufig eine Rolle, da sie die autonome Anpassung des Systems an dynamische Umgebungen übernehmen [30]. Die Observer/Controller-Architektur erlaubt Selbstorganisation und gleichzeitig die Reaktion und Kontrolle auf bzw. von neu auftretendem globalen Verhalten. Das System under Observation/Control (SuOC) ist nicht mehr ein zentrales, sondern dezentrales System mit vielen interagierenden Subsystemen. Den Kern der Architektur stellen Sensoren und Aktoren dar. Das System durchläuft eine Kontrollschleife: Der Observer beobachtet das Systemverhalten durch Sensoren und vergleicht das Ergebnis mit dem erwarteten Verhalten anhand eines vom Controller gewählten Observationsmodells. Der Controller erhält das Ergebnis, entscheidet welche Aktion notwendig ist und kontrolliert das SuOC nach seiner Zielfunktion mit der besten bekannten Aktion durch die Aktoren [31].

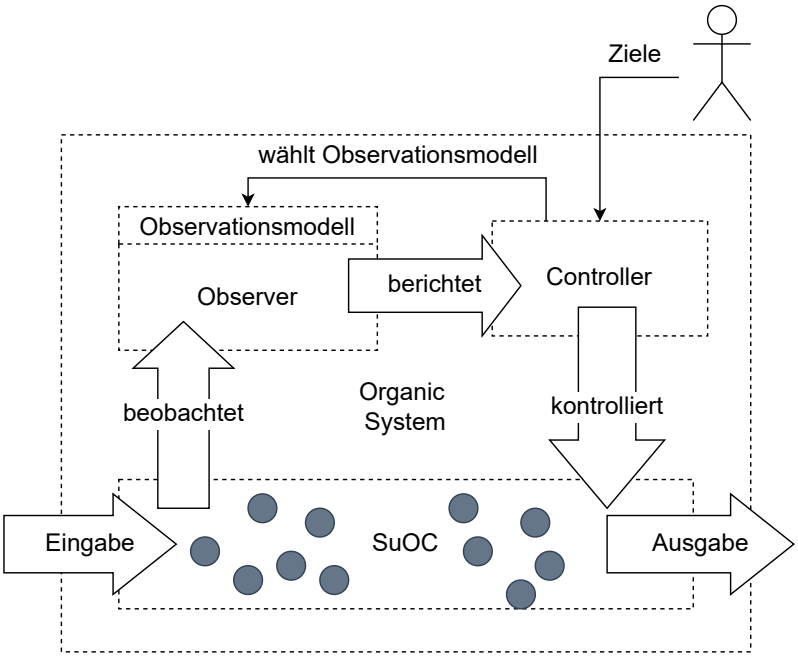


Abbildung 2.4: Observer/Controller Architektur. Eigene Abbildung nach [31].

2.4 Anomalien

Da in agentenbasierten cyber-physischen Energiesystemen sowohl die physischen Daten wie beispielsweise Spannung, Stromstärke und Temperaturwerte als auch Kommunikationsdaten erfasst werden, liegt eine große, heterogene Informationsmenge vor [23]. Störungen wirken sich als Anomalien in Daten des Energiesystems aus.

Anomalien sind Muster in Daten, die von definiertem Normalverhalten abweichen. Solche Datenpunkte sind von besonderem Interesse bei Analysen, da sie wichtige und kritische verwertbare Informationen über das System beinhalten. In Abbildung 2.5 ist beispielhaft dargestellt, wie sich Anomalien in einer Datenmenge mit zwei Attributen x und y zeigen können. Die meisten Datenpunkte liegen in den Normalregionen N_1 und N_2. Die Punkte o_1 und o_2 sowie die Region O_3 liegen so weit von den als normal definierten Punkten entfernt, dass sie als Anomalien betrachtet werden [13]. Anomalien können verschiedene Ursachen zugrunde liegen, wie beispielsweise Cyber-Attacken oder physische Störungen, die auch in CPS denkbar sind. Die Erkennung solcher Anomalien ist für ein zuverlässiges und sicheres System essentiell. In Unterabschnitt 2.5.6 wird auf verschiedene Methoden zur Anomalieerkennung eingegangen.

Grundsätzlich können drei Arten von Anomalien unterschieden werden [32]:

- *Punktanomalien* Punktanomalien sind einzelne Datenpunkte, die sich stark von den

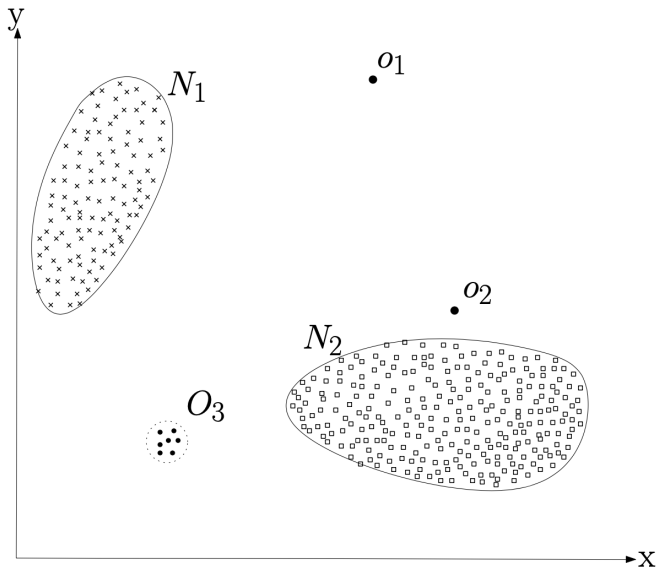


Abbildung 2.5: Ein Beispiel für Anomalien in einem zwei-attributigen Datenset [13].

anderen unterscheiden. Beispiele dafür sind o_1 und o_2 in Abbildung 2.5.

- *Kollektive Anomalien* Kollektive Anomalien sind eine zusammenhängende Menge von Datenpunkten, die insgesamt von den normalen Datenpunkten abweicht. Ein Beispiel dafür ist die Region O_3 in Abbildung 2.5.

- *Kontextuelle Anomalien* Kontextuelle Anomalien treten in einem bestimmten inhaltlichen Kontext des Anwendungsfalls auf. Beispielsweise kann es anomal sein, wenn zwei Agenten, die über eine lange Zeit regelmäßig kommuniziert haben, aufhören sich Nachrichten zu schicken.

2.4.1 Anomalien in Zeitreihen

In CPS ist es sinnvoll, verfügbare Messwerte als Zeitreihen zu erfassen, um eine ständige Überwachung sicherzustellen. Zeitreihen sind Sequenzen von Werten, die durch Messungen eines zugrunde liegenden Prozesses in gleichmäßigen zeitlichen Abständen erhalten werden. Wird nur eine Variable betrachtet, ist die Zeitreihe univariat. Werden mehrere Variablen gleichzeitig gemessen, spricht man von einer multivariaten Zeitreihe.

Eine *univariate Zeitreihe* $X = \{x_t\}_{t \in T}$ ist eine geordnete Sequenz bestehend aus reellwertigen Observationen, welche jeweils zu einem bestimmten Zeitpunkt $t \in T \subseteq \mathbb{Z}^+$ aufgenommen werden. Somit ist x_t ein Punkt (oder eine Beobachtung) zur Zeit t und $S = x_p, x_{p+1}, ..., x_{p+n-1}$ ist eine Teilsequenz der Länge $n \leq |T|$, angefangen an Position p

der Zeitreihe X, wobei $p, t \in T$ und $p \leq |T| - n + 1$ gilt. Es wird angenommen, dass jedes x_t ein realisierter Wert einer Zufallsvariablen X_t ist [33].

Eine *multivariate Zeitreihe* $\mathbf{X} = \{\mathbf{x}_t\}_{t \in T}$ ist eine geordnete Menge von k- dimensionalen Vektoren, die jeweils zu einem bestimmten Zeitpunkt $t \in T \subseteq \mathbb{Z}^+$ aufgenommen werden und aus k reellwertigen Observationen $\mathbf{x}_t = (x_{1t}, ..., x_{kt})$ bestehen. Für jede Dimension $j \in \{1, ..., k\}$ ist $X_j = \{x_{jt}\}_{t \in T}$ eine univariate Zeitreihe und jeder Punkt x_{jt} im Vektor $\mathbf{x_t}$ ist ein realisierter Wert einer zeitabhängigen Zufallsvariablen X_{jt} in $\mathbf{X}_t = (X_{1t}, ..., X_{kt})$. Hier ist jede Variable nicht nur von ihren vorigen Werten abhängig, sondern auch von den anderen zeitabhängigen Variablen [33].

Arten von Anomalien in Zeitreihen

Punktanomalien sind einzelne Datenpunkte, die sich im Vergleich zu ihren Nachbarpunkten (lokale Punktanomalie) oder den anderen Datenpunkten in der Zeitreihe (globale Punktanomalie) ungewöhnlich verhalten. Die Anomalien können uni- oder multivariat sein, je nachdem, ob sie eine oder mehrere zeitabhängige Variablen betreffen (siehe Abbildung 2.6 und 2.7) [33].

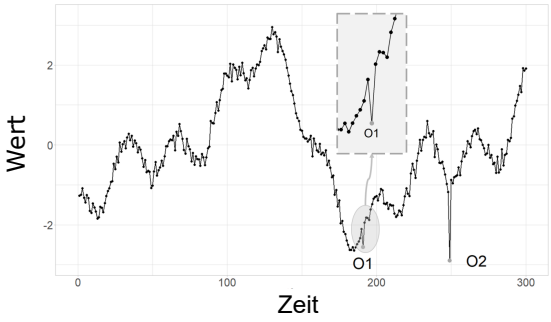


Abbildung 2.6: Punktanomalie in einer univariaten Zeitreihe [33].

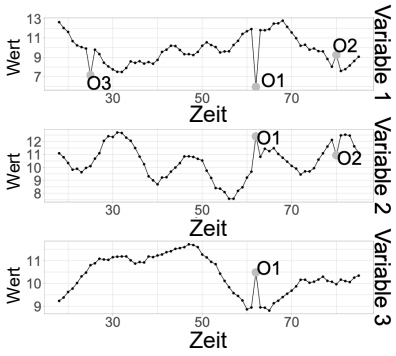


Abbildung 2.7: Punktanomalie in einer multivariaten Zeitreihe [33].

Unter Teilsequenz- Anomalien (oder kollektiven Anomalien) versteht man mehrere aufeinanderfolgende Datenpunkte, die sich insgesamt ungewöhnlich verhalten, wobei jeder einzelne Punkt nicht zwangsläufig eine Punktanomalie sein muss. Auch Teilsequenz- Anomalien können lokal oder global sein und entweder eine (univariate Teilsequenz- Anomalie) oder mehrere (multivariate Teilsequenz- Anomalie) zeitabhängige Variablen betreffen [33]. Dies ist in Abbildung 2.8 und 2.9 dargestellt.

Außerdem gibt es ganze anomale Zeitreihen, die aber nur bei multivariaten Zeitreihen erkannt werden können (siehe Abbildung 2.10) [33].

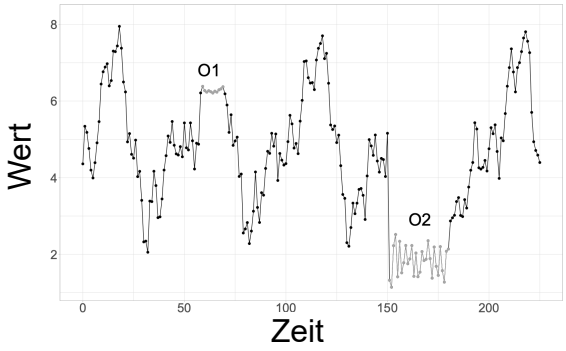


Abbildung 2.8: Kollektive Anomalien O1 und O2 in einer univariaten Zeitreihe [33].

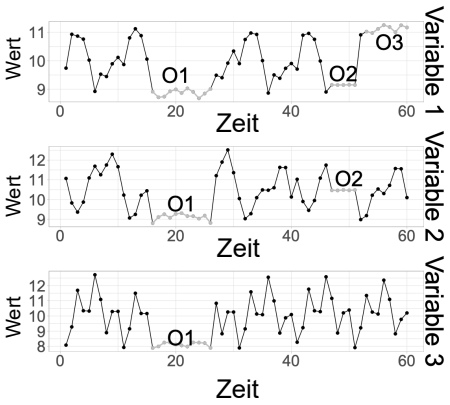


Abbildung 2.9: Kollektive Anomalien O1, O2 und O3 in einer multivariaten Zeitreihe [33].

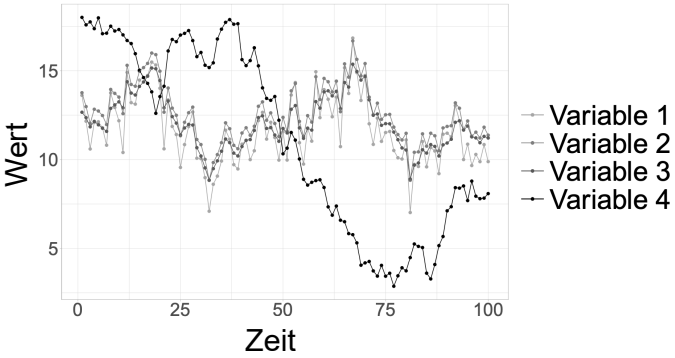


Abbildung 2.10: Variable 4 (schwarz) ist eine anomale Zeitreihe [33].

2.5 Machine Learning

ML ist ein Teilbereich der künstlichen Intelligenz, bei dem Modelle selbstständig Zusammenhänge und Muster aus großen Datenmengen erlernen. Der entscheidende Unterschied zu klassischen Programmen ist, dass ML-Modelle trainiert anstatt explizit programmiert werden [34]. Ein klassisches Programm erhält Regeln und Daten als Eingabe und gibt eine Antwort aus. Ein ML-Algorithmus erhält viele Beispieldaten und ggf., je nach Art des Lernens, die korrekte Antwort und findet statistische Strukturen in diesen Daten um selbst Regeln zu finden, die zu einer korrekten Antwort für neue Daten führen. Viele statistische Analysemethoden selbst werden als ML-Algorithmus deklariert [35]. Die vier Arten von ML sind Supervised (überwachtes), Unsupervised (unüberwachtes), Semi-supervised (teil-

überwachtes) und Reinforcement (verstärkendes) Learning [34]. Diese werden im Folgenden kurz vorgestellt.

2.5.1 Supervised Learning

Beim Supervised Learning wird eine Funktion generiert, die den Input auf den gewünschten Output abbildet und anschließend für ungesehene Daten den richtigen Output geben kann [36]. Dazu ist ein Trainingsprozess mit gelabelten Daten notwendig. Ein Label y ist die gewünschte Vorhersage einer Instanz $\vec{x}$ (Feature-Vektor). Die Trainingsdaten mit Größe n bestehen aus Paaren der Instanz und des zugehörigen Labels: $\{\vec{x}_i, y_i\}_{i=1}^n$. Die Label y können, je nach Aufgabe des Verfahrens, diskrete Werte einer endlichen Menge an unterschiedlichen Klassen sein (Klassifizierungsproblem) oder kontinuierliche Werte in $\mathbb{R}$ annehmen (Regressionsproblem). Dem Lernprozess liegt die Annahme zugrunde, dass die Trainingsdaten mit einer unbekannten, aber festen gemeinsamen Wahrscheinlichkeitsverteilungsfunktion $P(\vec{x}, y)$ generiert werden. Das Ziel ist es, eine Funktion $f : X \mapsto Y$ zu finden, die die Beziehung aus $P(\vec{x}, y)$ zwischen $\vec{x}$ und y modelliert, sodass $f(\vec{x})$ das tatsächliche Label von zukünftigen Daten vorhersagt. Dabei ist X der Feature-Raum (d-dimensional mit $d =$ Anzahl Features) und Y der Definitionsbereich von y (bei Klassifizierungsproblemen ist $dim Y =$ Anzahl der Klassen) [36, 37].

Im Training kann mithilfe der Labels bestimmt werden, wie nah die Vorhersage $f(\vec{x})$ am gewünschten Ouptut y liegt. Dies wird durch eine Kostenfunktion (auf Englisch häufig als loss function betitelt) $\mathcal{L} : Y \times Y \mapsto \mathbb{R}^+$ evaluiert. Es gibt verschiedene Kostenfunktionen $\mathcal{L}(f(\vec{x}, y))$, deren Wahl vom vorliegenden Problem abhängt [36]. Grundsätzlich sollte das ML-Modell ein f finden, das den Fehler minimiert. Wenn das Modell allerdings anhand der Trainingsdaten zusätzlich zum statistischen Zusammenhang zwischen X und Y auch noch statistisches Rauschen lernt, wird dies *Overfitting* genannt. Die Performance auf den Trainingsdaten ist dann zwar sehr gut, doch das Modell ist so überangepasst, dass es die Label ungesehener Daten fehlerhaft vorhersagt. Daher sollte ein f angestrebt werden, das den Trainingsfehler nur näherungsweise minimiert und nicht zu komplex ist. Durch den Einsatz gelabelter Testdaten, die das Modell während des Trainings nicht gesehen hat, kann die Performanz auf zukünftigen Daten eingeschätzt werden [37].

2.5.2 Unsupervised Learning

Beim Unsupervised Learning gibt es nur n Instanzen $\{\vec{x}_i\}_{i=1}^n$ ohne Label. Es gibt also keine Möglichkeit, die Performanz des Modells zu überwachen, da der gewünschte Output nicht bekannt ist. Das Ziel ist es stattdessen, bestimmte Strukturen in den vorliegenden Daten zu finden. Die typischen Aufgaben des Unsupervised Learning sind [37]:

- Clustering: $\{\vec{x}_i\}_{i=1}^{n}$ in k Cluster aufteilen, sodass Instanzen im selben Cluster ähnlich und Instanzen aus unterschiedlichen Clustern unähnlich sind.
- Erkennung von Neuartigkeiten: Die wenigen Instanzen identifizieren, die sich sehr stark von der Mehrheit der Datenpunkte unterscheiden.
- Dimensionsreduktion: Jede Instanz durch einen niederdimensionalen Featurevektor repräsentieren, während die Hauptcharakteristika erhalten bleiben.

2.5.3 Semi-supervised Learning

Semi-supervised Learning ist eine Mischung aus Supervised und Unsupervised Learning und arbeitet mit sowohl gelabelten als auch ungelabelten Daten. Die Idee ist, für einen Supervised oder Unsupervised Algorithmus zusätzliche Informationen des jeweils anderen Lernens einzubeziehen, um die Performanz zu verbessern. Beispiele dafür sind [37]:

- Semi-supervised Klassifizierung: Klassifizierung mit l gelabelten Instanzen $\{\vec{x}_i, y_i\}_{i=1}^{l}$ und u ungelabelten Instanzen $\{\vec{x}_i\}_{i=l+1}^{l+u}$, wobei $l \ll u$, also sehr viel weniger gelabelte als ungelabelte Daten vorhanden sind. Es wird dann f auf allen Daten trainiert, sodass f besser wird als bei Training auf nur gelabelten Daten.
- Begrenztes Clustering: Eine Erweiterung von Unsupervised Clustering, bei der es erweiterte Informationen über die Cluster gibt, beispielsweise welche Instanzen im selben oder in verschiedenen Clustern sein müssen.
- Regression mit gelabelten und ungelabelten Daten
- Dimensionsreduktion mit gelabelten Instanzen, deren reduzierter Featurevektor bekannt ist

2.5.4 Reinforcement Learning

Reinforcement Learning findet durch einen Agenten statt, der mit seiner Umgebung interagiert und durch Versuch und Irrtum eine optimale Strategie für Entscheidungsprobleme findet. Dabei ist die Entscheidungsfindung sequentiell und es gibt nach einer Entscheidung bewertende Rückmeldungen anstatt gelabelter Daten. Beim Supervised und Unsupervised Learning gibt es eine einmalige Antwort sowie kurzfristige, sofortige Rückmeldungen in Form von beispielsweise der Kostenfunktion. Im Gegensatz dazu arbeitet Reinforcement Learning weitblickend und mit langfristigen, akkumulierten Belohnungen [38].

Der Agent interagiert über die Zeit mit der Umgebung. Zu jedem Zeitpunkt t erhält der Agent einen Zustand s_t aus dem Zustandsraum $\mathcal{S}$ und wählt eine Aktion a_t aus dem

Aktionsraum $\mathcal{A}$. Dabei folgt er einer Strategie $\pi(a_t|s_t)$, welche das Verhalten des Agenten (also eine Abbildung des Zustands auf die Aktion) definiert. Daraufhin erhält der Agent eine skalare Belohnung r_t nach einer Belohnungsfunktion $\mathcal{R}(s, a)$ und geht in den nächsten Zustand s_{t+1} über mit einer Übergangswahrscheinlichkeit $\mathcal{P}(s_{t+1}|s_t, a_t)$. Dies geht so lange, bis der Agent einen Endzustand erreicht und erneut startet. Die akkumulierte Belohnung mit Diskontfaktor $\gamma \in (0, 1]$ ist [38]:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.1}$$

Der Agent zielt darauf ab, R_t aus jedem Zustand heraus langfristig zu optimieren [38].

2.5.5 Künstliche Neuronale Netze und Deep Learning

Künstliche Neuronale Netze (KNN) sind Modelle, die aus Daten schichtweise Darstellungen lernen. Diese Schichten enthalten zunehmend aussagekräftige Repräsentationen und die Anzahl der aufeinanderfolgenden Schichten wird als Tiefe des KNN bezeichnet. Im Allgemeinen bestehen KNN aus einer Eingabeschicht (Input Layer), in der Merkmale durch eine Reihe von Neuronen (auch Knoten genannt) dargestellt werden. Danach gibt es eine Anzahl an versteckten Schichten (Hidden Layer) mit unterschiedlicher Anzahl an Knoten. Die Ausgabeschicht (Output Layer) enthält eine je nach Aufgabe definierte Anzahl an Knoten und gibt die Lösung des Modells aus [35].

Jedes Neuron erhält eine Anzahl an Eingaben (Features) x_j und produziert eine Ausgabe (Output), der definiert ist als die gewichtete Summe $\sum_j w_j x_j$ größer oder kleiner als ein gegebener Grenzwert (threshold). Die Features des Input Layers sind mit der spezifischen Aufgabe verknüpft und können beispielsweise physische Daten wie Spannung, Leistung oder Temperatur einer Netzkomponente im Smart Grid sein. Bringt man den Grenzwert auf die andere Seite der Ungleichung, erhält man $-threshold = b$ und b wird Bias genannt. Der Bias ist analog zu einer Konstanten, die zu einer linearen Funktion addiert wird und verschiebt den Output um einen konstanten Wert. Das Ziel ist, das Netz alle Gewichte und Bias lernen zu lassen, sodass der Output des Netzes die richtige Lösung für das gegebene Problem liefert [39].

In vollständig verbundenen Netzen (FCNs) ist jedes Neuron mit allen Neuronen der benachbarten Schichten verbunden. Ein Beispiel für ein FCN ist in Abbildung 2.11 gezeigt. Der Output eines Neurons in einer Schicht mit N Inputs $\{x_1, ..., x_N\}$ und n Gewichten

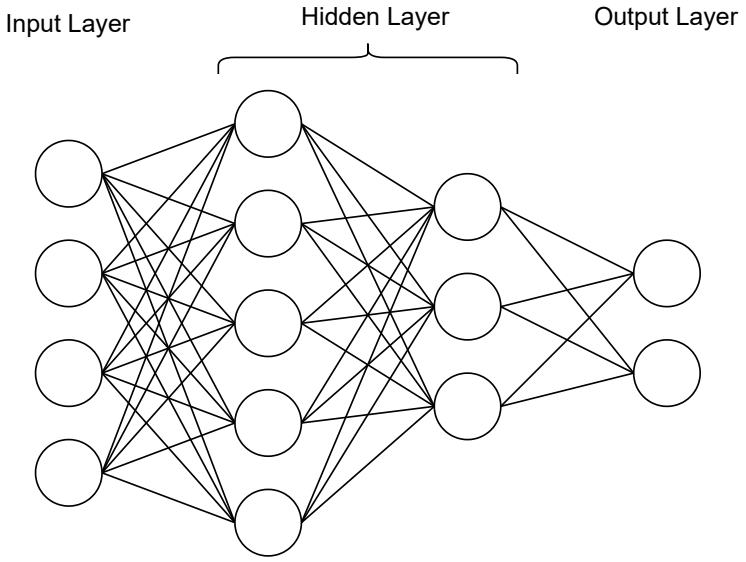


Abbildung 2.11: Ein Fully Connected Network (FCN) mit zwei versteckten Schichten. Eigene Abbildung nach [39].

$\{w_1, ..., w_N\}$ wird berechnet als

$$o = \sum_j w_j * x_j + b, \tag{2.2}$$

wobei x, w und b als Vektoren mit Länge gleich der Anzahl an Neuronen in der entsprechenden Schicht interpretiert werden. Dies ist zunächst linear, doch damit das Netz komplexe Funktionen modellieren kann, wird eine nichtlineare Aktivierungsfunktion σ auf die Outputs angewendet. Häufig verwendet wird die *Rectified Linear Unit (ReLU)* [39]:

$$g(z) = max\{0, z\} \tag{2.3}$$

Die *Sigmoid*-Funktion [39] wird ebenfalls oft benutzt:

$$\sigma(z) = \frac{1}{1 + exp(-z)} \tag{2.4}$$

Eine weitere Funktion ist *Leaky ReLU* [40]:

$$h(z) = max\{\frac{z}{r}, z\}, \tag{2.5}$$

wobei r ein positiver reeller Faktor ist.

Somit ergibt sich die Aktivierung a^l des $j-$ten Neurons in der $l-$ten Schicht aus den Aktivierungen in der $(l-1)-$ten Schicht als Summe über alle k Neuronen in der $(l-1)-$ten Schicht:

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \tag{2.6}$$

oder in vektorisierter Form:

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{2.7}$$

Die Aktivierung eines Neurons repräsentiert, wie groß der Einfluss des entsprechenden Features, welches das Neuron lernt, auf das gesamte Netz ist. Mit dieser grundlegenden Struktur kann das Netz komplexe Funktionen modellieren [39].

Lernprozess

Nachdem die Aktivierungen für die Neuronen des Input Layers initialisiert sind, wird der Output jeder Schicht zum Input für die nächste Schicht. Der gewichtete Input der Neuronen in Schicht l kann also geschrieben werden als (in vektorisierter Form)

$$z^l = w^l a^{l-1} + b^l. \tag{2.8}$$

Das Berechnen der Inputs und Aktivierungen für alle Schichten bis zur Ausgabeschicht wird *feedforward* genannt. Wenn dies geschehen ist, produziert das KNN einen finalen Output. Da der Trainingsprozess überwacht (supervised) ist, also der gewünschte Output bekannt ist, lässt sich sagen wie richtig oder falsch die Lösung des KNN ist. Dafür wird eine Kostenfunktion berechnet. Beispiele für Kostenfunktionen sind mittlerer quadratischer Fehler oder Kreuzentropie. Das Ziel des Trainings ist die Kostenfunktion zu minimieren [39, 41].

Um die Kostenfunktion durch Verändern der Argumente (was auf die Gewichte und Bias zurückführt) zu minimieren wird eine gradientenbasierte Optimierung angewendet. Die Ableitung einer Funktion gibt ihre Steigung an und spezifiziert, wie eine kleine Änderung ϵ im Input den Output verändert. Formal ist also

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x). \tag{2.9}$$

Da

$$f(x - \epsilon \operatorname{sign}(f'(x))) < f(x) \tag{2.10}$$

für ein ausreichend kleines ϵ, kann $f(x)$ verkleinert werden, indem x in kleinen Schritten mit umgekehrtem Vorzeichen der Ableitung bewegt wird. Dafür werden die Gradienten für jeden Input x berechnet und der Mittelwert gebildet, der dann als Ableitung der Funktion verwendet wird. Dieser Algorithmus heißt *gradient descent*. Die Größe der Schritte ist die Lernrate. Diese muss sorgfältig gewählt werden, da das Minimum bei zu großen Schritten übergangen würde, aber der Prozess bei zu kleinen Schritten sehr langsam wird. In der Praxis werden dafür *Optimizer* verwendet, die die Lernrate für jedes Gewicht individuell festlegen. Wenn $f'(x) = 0$ hält der Algorithmus, da die Funktion einen stationären Punkt erreicht hat. Möglicherweise hat die Funktion mehrere lokale Minima, die global gesehen nicht optimal sind. Daher ist die Optimierung eine komplexe Aufgabe, vor allem, wenn der Input multidimensional ist. Der Kompromiss ist es, einen ausreichend kleinen Wert für die Kostenfunktion zu finden, der für eine gute Performanz des Modells sorgt und gleichzeitig nicht zu große Rechenressourcen beansprucht. Nach jedem Schritt werden die Gewichte und Bias aktualisiert (für jede Schicht) [39, 41]:

$$w_k \to w'_k = w_k - \eta \frac{\partial \mathcal{L}}{\partial w_k} \tag{2.11}$$

$$b_j \to b'_j = b_j - \eta \frac{\partial \mathcal{L}}{\partial b_j} \tag{2.12}$$

Dabei ist η die Lernrate und $\mathcal{L}$ die Kostenfunktion. Da dies für große Inputmengen sehr rechenaufwendig ist, wird üblicherweise der Trainingsdatensatz in zufällige *Batches* mit einer definierten Anzahl an Inputs aufgeteilt. Die Gewichte und Bias werden dann nur nach Ende jedes Batches aktualisiert [41].

Für die Berechnung des Gradienten der Kostenfunktion wird der *Backpropagation* Algorithmus verwendet. Dabei werden die Informationen rückwärts vom Output Layer wieder zum Input Layer getragen um die einzelnen Gradienten zu berechnen. Der Fehler von Neuron j in Schicht l ist

$$\delta^l_j = \frac{\partial \mathcal{L}}{\partial z^l_j}, \tag{2.13}$$

wobei $\mathcal{L}$ die Kostenfunktion ist und z^l_j in Gleichung 2.8 definiert ist. Um die Fehler für alle Schichten zu berechnen startet der Algorithmus beim Output Layer, dessen Fehler

folgendermaßen berechnet wird:

$$\delta_j^L = \frac{\partial \mathcal{L}}{\partial a_j^L} \sigma'(z_j^L) \tag{2.14}$$

Dabei ist σ' die Ableitung der Aktivierungsfunktion. Die linke partielle Ableitung gibt an, wie schnell sich die Kosten ändern und die rechte Ableitung bestimmt, wie stark sich die Aktivierungsfunktion an diesem Input z_j^L ändert [39, 41].

Der Fehler der $l-$ten Schicht kann also als Fehler der nachfolgenden Schicht ausgedrückt werden:

$$\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l), \tag{2.15}$$

wobei $\odot$ das Hadamard-Produkt ist, also das elementweise Produkt der zwei Vektoren. So kann der Fehler jeder Schicht rückwärts durch das gesamte Netz berechnet werden. Daraus ergeben sich dann die Gradienten der Kostenfunktion in Bezug auf die Gewichte und Bias [39, 41]:

$$\frac{\partial \mathcal{L}}{\partial w_j^l} = a_k^{l-1} \delta_j^l \tag{2.16}$$

$$\frac{\partial \mathcal{L}}{\partial b_j^l} = \delta_j^l \tag{2.17}$$

Performanzkriterien

Während des Lernprozesses werden die Trainingsdaten wiederholt in das Netz gegeben und die o.g. Schritte durchlaufen. Ein vollständiger Durchlauf des Trainingsdatensatzes wird Epoche genannt. Um zu überprüfen, ob das Netz lernt, gibt es einige Techniken zur Überwachung des Trainingsprozesses. Das Modell sollte in der Lage sein, auch bei zuvor unbekannten Inputs gute Ergebnisse zu erzielen. Daher wird ein Validierungsdatensatz mit Daten, die der Trainingsalgorithmus zuvor nicht gesehen hat, von den Trainingsdaten getrennt und zur Schätzung des allgemeinen Fehlers während und nach dem Training verwendet. Durch die Bewertung der Leistung anhand dieser Validierungsdaten können die Hyperparameter des Netzes, wie beispielsweise die Batchgröße, die Lernrate oder die Anzahl der Epochen, anschließend aktualisiert werden. Üblicherweise werden 80 % der Daten für das Training und 20 % für die Validierung verwendet. Nachdem der Trainingsprozess abgeschlossen ist, kann das Modell auf zuvor unbekannten Testdaten ausgeführt werden [41].

Zwei Maße für die Performanz eines ML-Modells sind die Fähigkeit, den Trainingsfehler sowie den Abstand zwischen Trainings- und Validierungsfehler klein zu halten. Die Probleme, die auftreten können, sind *Underfitting* oder *Overfitting*. *Underfitting* bedeutet, dass das Modell nicht in der Lage ist, einen ausreichend niedrigen Fehler für die Trainingsdaten zu erzielen. *Overfitting* liegt vor, wenn der Abstand zwischen Trainings- und Validierungsfehler zu groß wird und das Modell auf die Details der Trainingsdaten überangepasst ist. Es performt dann schlecht auf zuvor ungesehenen Daten, wird also schlecht im Generalisieren [41].

Eine Möglichkeit, *Overfitting* zu verhindern, ist die Anwendung von *Dropout*-Schichten. Sie entfernen vorübergehend einen bestimmten Anteil von zufälligen Knoten. Der Input und Output Layer bleiben davon unberührt. Ein KNN mit n Knoten hat 2^n mögliche ausgedünnte Netze, die sich alle Gewichte teilen. Eine bestimmte Anzahl dieser ausgedünnten Netze wird über jeweils ein Batch trainiert, und dieser Vorgang wird mehrmals wiederholt. Da auf diese Weise viele verschiedene Netze trainiert werden, werden sie unterschiedlich *overfitten*. Schlussendlich werden die Effekte aller Netze gemittelt und führen dazu, dass das ursprüngliche Netz weniger stark überangepasst ist. Durch diesen Prozess lernt das Netz robustere Merkmale [42].

2.5.6 Anomalieerkennung mit Ansätzen aus dem Machine Learning

Bei der Anomalieerkennung sollen in Daten diejenigen Instanzen gefunden werden, die im Kontext des Anwendungsfalls als anomal einzuordnen sind. Dafür gibt es verschiedene Strategien aus dem ML.

Supervised Anomalieerkennung

Bei überwachten Ansätzen müssen die Trainingsdaten gelabelte normale und anomale Instanzen enthalten. Die Schwierigkeit besteht hier darin, genaue Label und Daten zu finden, die alle möglichen Normal- und Fehlerfälle abdecken. Außerdem gibt es in den Daten oft ein statistisches Rauschen, das zu häufigeren Fehlalarmen führt. Beispiele für Supervised Learning Methoden zur Anomalieerkennung sind Supervised KNN, Support Vector Machines (SVMs) und Entscheidungsbäume [43].

Unsupervised Anomalieerkennung

Für unüberwachte Methoden sind keine Trainingsdaten und keine Label notwendig. Unsupervised Learning kann zur Anomalieerkennung genutzt werden, wenn zwei grundlegende Annahmen eingehalten werden: es gibt in den Daten nur wenige anomale Instanzen und

diese weisen statistische Unterschiede zu den normalen Daten auf. Datengruppen aus ähnlichen und häufig auftretenden Instanzen werden dann als normal und seltene Instanzen, die statistisch abweichen, als anomal eingeordnet. Dafür können zum Beispiel verschiedene Cluster-Techniken, Isolation Forests oder One-Class Support Vector Machines (OCSVMs) (siehe Unterabschnitt 2.6.2 und 2.6.3) verwendet werden [43].

Semi-supervised Anomalieerkennung

Mit teilweise gelabelten Daten können teilüberwachte Ansätze zur Anomalieerkennung genutzt werden. Hierbei werden beispielsweise nur normale Daten als solche gelabelt und das Modell lernt, was normal ist. Anomale Datenpunkte können dann als abweichende Instanzen von den normalen Daten differenziert werden. Ein Beispiel für ein solches Verfahren sind Autoencoder (siehe Unterabschnitt 2.6.1) [44].

2.5.7 Bewertung eines Machine-Learning-Modells zur Anomalieerkennung

Für die Evaluation eines ML-Modells gibt es verschiedene Metriken, die verwendet werden können, wenn die Daten gelabelt sind. Bei Supervised Ansätzen ist dies bereits gegeben. Auch Methoden, die Unsupervised oder Semi-supervised sind, können bewertet werden, wenn sie mit Daten ausgeführt werden, deren Label bekannt sind. Im Kontext von CPS könnten beispielsweise historische Daten durch Analysen gelabelt und daran die Performanz von Modellen bewertet werden, die später mit aktuelleren Daten arbeiten, für die es noch kein Label gibt.

Mit den Labels der Daten und den Ergebnissen des Modells lässt sich eine *Confusion Matrix* erstellen. Da es bei der Anomalieerkennung zwei Klassen (normal/anomal) gibt und das Modell die Instanzen entweder richtig oder falsch einordnen kann, bilden sich vier Möglichkeiten. Die True Positives (TP) sind die Instanzen, die das Modell richtig als Anomalien erkannt hat. Die Instanzen, die das Modell fälschlicherweise als anomal klassifiziert, sind die False Positives (FP). Entsprechend sind die True Negatives (TN) und die False Negatives (FN) die richtig bzw. falsch erkannten Normalinstanzen. Abbildung 2.12 zeigt eine Confusion Matrix mit den ensprechenden Kennzahlen, die sich aus einem Modell zur Anomalieerkennung ergeben.

Mit diesen Kennzahlen lassen sich die verschiedenen Metriken zur Bewertung definieren. Die *Accuracy* gibt den Anteil korrekt klassifizierter Instanzen an und wird wie folgt berechnet [46]:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.18}$$

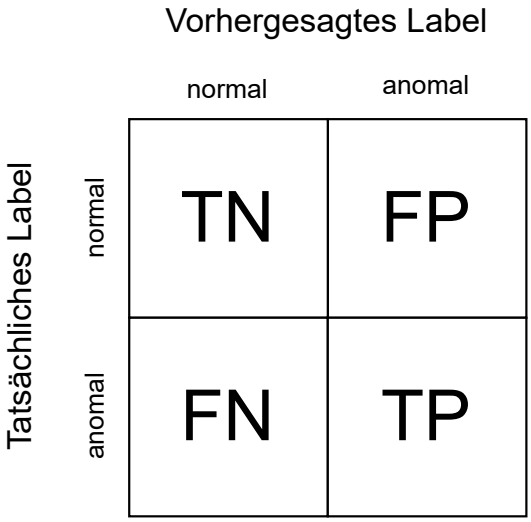


Abbildung 2.12: Confusion Matrix für zwei Klassen normal und anomal mit den entsprechenden Kennzahlen. Eigene Abbildung nach [45].

Der *Recall* (auch True Positive Rate (TPR) oder Sensitivität [47]) lässt sich interpretieren als die Wahrscheinlichkeit, dass das Modell die Anomalien in den Daten als solche erkennt und wird berechnet als

$$r = \frac{TP}{TP + FN}. \tag{2.19}$$

Die *Precision* gibt den Anteil der gefundenen Anomalien an, der auch tatsächlich anomal ist:

$$p = \frac{TP}{TP + FP} \tag{2.20}$$

Das harmonische (gewichtete) Mittel aus Precision und Recall ist der *F-Score*:

$$F_\beta = (1 + \beta^2)\frac{p * r}{r + \beta^2 * p} \tag{2.21}$$

Häufig wird der F_1-Score (mit $\beta = 1$) verwendet [48]. Der Anteil richtig erkannter Normaldaten ist die True Negative Rate (TNR) (auch *Spezifität* [47]) und wird berechnet als

$$TNR = \frac{TN}{TN + FP}. \tag{2.22}$$

Der Anteil der Normalinstanzen, die als Anomalien missinterpretiert wurden ist die False

Positive Rate (FPR) [49]:

$$FPR = \frac{FP}{FP + TN} \tag{2.23}$$

Ein weiteres Performanzmaß ist die Area Under Curve (AUC) der Receiver Operating Characteristics (ROC)-Kurve. Die ROC-Kurve erhält man durch die grafische Darstellung der FPR auf der x-Achse und der TPR auf der y-Achse. Die AUC ist die Fläche unter dieser Kurve. Je näher die AUC an 1 ist, desto besser ist das Modell. Eine AUC von 0,5 würde bedeuten, dass das Modell nicht besser als zufälliges Raten ist [47].

2.6 Machine-Learning-Algorithmen zur Anomalieerkennung in Zeitreihen

In dieser Arbeit werden ausgewählte Semi-supervised und Unsupervised Algorithmen zur Anomalieerkennung in Zeitreihen verwendet. Diese basieren auf unterschiedlichen Vorgehensweisen, welche im Folgenden erklärt werden.

2.6.1 Autoencoder

Ein Autoencoder ist ein Unsupervised KNN, welches lernt, die Input-Vektoren als Output zu reproduzieren. Die Anzahl der Neuronen des Input Layers entspricht also der Zahl der Neuronen des Output Layers. Dazwischen liegen eine oder mehr Hidden Layer mit kleinerer Neuronenanzahl. Der Autoencoder kann somit als zweiteiliges KNN verstanden werden, bestehend aus einem *Encoder* und einem *Decoder*. Dies ist in Abbildung 2.13 dargestellt. Im Encoder werden die Input-Vektoren $\{\vec{x}_1, ..., \vec{x}_n\}$, wobei $\vec{x}_i \in \mathbb{R}^d$, auf m $(m < d)$ Neuronen des Hidden Layers komprimiert. Bei mehr als einem Hidden Layer wird die Zahl der Neuronen weiter bis zum kleinsten Hidden Layer reduziert. Der Decoder dekodiert die reduzierte Repräsentation wieder in den originalen Input-Raum $\mathbb{R}^d$. Bei mehr als einem Hidden Layer geschieht dies durch entsprechende Zwischenlayer mit steigender Neuronenzahl in Richtung Output Layer. Die Abweichung zwischen Input und reproduziertem Output $\{\vec{x'}_1, ..., \vec{x'}_n\}$ ist der *Reconstruction Error* [50]:

$$\epsilon(\vec{x}_i, \vec{x'}_i) = \sum_{j=1}^{d} (\vec{x}_i - \vec{x'}_i)^2 \tag{2.24}$$

Der Autoencoder minimiert diesen während des Trainings.

Zur Anomalieerkennung kann ein Autoencoder eingesetzt werden, indem er mit ausschließlich normalen Daten trainiert wird. Das Verfahren ist also Semi-supervised, da ein

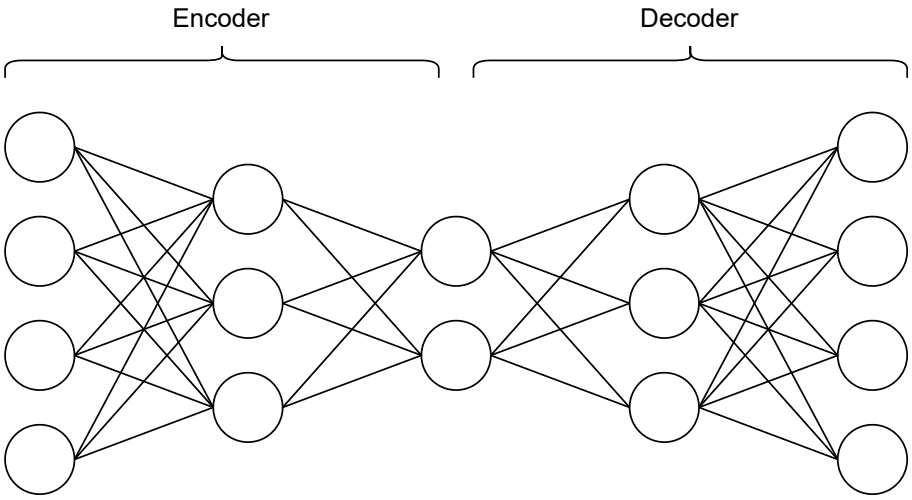


Abbildung 2.13: Beispiel eines Autoencoders (vgl. [50]).

Labeling von normalen Daten notwendig ist. Während des Trainings lernt der Autoencoder die normalen Daten gut zu reproduzieren. Werden anschließend Daten eingegeben, die Anomalien enthalten, wird der Reconstruction Error bei den Anomalien höher sein. Durch Festlegen eines Grenzwerts θ kann den Daten eine Klasse c zugeordnet werden [50]:

$$c(\vec{x}_i) = \begin{cases} normal & \epsilon_i < \theta \\ anomal & \epsilon_i > \theta \end{cases} \tag{2.25}$$

2.6.2 Isolation Forest

Die Idee bei einem Isolation Forest ist, anomale Daten anhand abweichender Charakteristika von normalen Daten zu separieren. Dabei gelten die Annahmen, dass Anomalien im Gesamtdatensatz einen geringen Anteil ausmachen und es einen signifikanten Unterschied zwischen anomalen Instanzen und den Features normaler Daten gibt [51].

Ein Isolation Forest erstellt Isolationsbäume. Ein Isolationsbaum ist ein binärer Baum, bei dem jeder Knoten A entweder ein externer Knoten ohne Kindknoten oder ein interner Knoten ist, der genau zwei Kindknoten A_l, A_r hat. Jeder interne Knoten hat einen *Test* mit einem Attribut q und einem Splitwert p, sodass *Test* $q < p$ die Daten in A_l und A_r aufteilt. Ein Datensatz $X = \vec{x}_1, ..., \vec{x}_n$ aus n Instanzen mit d Features wird rekursiv anhand eines zufällig gewählten Attributs q mit einem zufälligen Splitwert p geteilt, bis

(i) der Baum die maximale Höhe erreicht,

(ii) $|X| = 1$, oder

(iii) alle Daten in X denselben Wert haben.

Angenommen, X besteht nur aus Instanzen mit unterschiedlichen Werten, dann ist jede Instanz zu einem externen Knoten isoliert, wenn der Isolationsbaum voll ausgewachsen ist. Die Anzahl externer Knoten ist dann n, die Anzahl der internen Knoten ist $n-1$ und die Gesamtzahl an Knoten ist $2n-1$. Der Speicherbedarf des Algorithmus zum Erstellen der Isolationsbäume wächst also linear mit n [52].

Anomale Datenpunkte liegen nun näher an der Wurzel des Isolationsbaums, da sie selten vorkommen und durch weniger charakteristische Bedingungen von normalen Datenpunkten abgegrenzt werden können. Normale Datenpunkte liegen weiter von der Wurzel entfernt [51]. Die Pfadlänge $h(x)$ ist die Zahl der Kanten im Isolationsbaum von x als externem Knoten bis zum Wurzelknoten [52]. In Abbildung 2.14 und 2.15 ist angedeutet, dass sich anomale Datenpunkte schneller isolieren lassen als normale Punkte und somit eine kürzere Pfadlänge im Isolationsbaum besitzen.

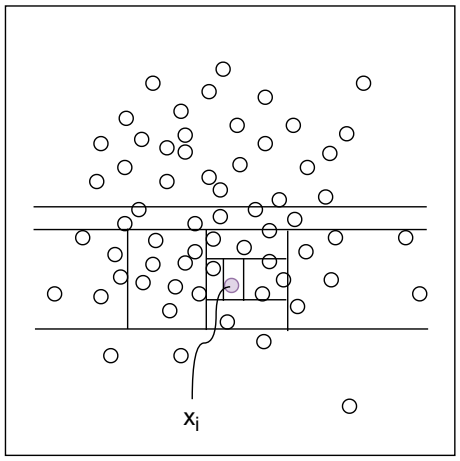


Abbildung 2.14: Isolation von einem normalen Datenpunkt x_i. Es werden viele Splits benötigt, um den Punkt zu isolieren. Eigene Darstellung nach [52].

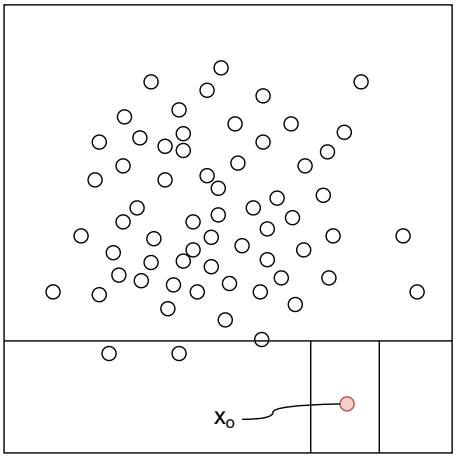


Abbildung 2.15: Isolation eines anomalen Datenpunkts x_o. Es werden deutlich weniger Splits benötigt, um den Punkt zu isolieren. Eigene Darstellung nach [52].

Ein Isolation Forest besteht aus einer definierten Zahl von zufälligen Isolationsbäumen. Über die mittlere Pfadlänge einer Instanz x kann bestimmt werden, ob es sich um eine Anomalie oder einen normalen Datenpunkt handelt. Dazu werden zunächst die erwartete mittlere Pfadlänge der Instanz $E(h(x))$ und die mittlere Pfadlänge im Isolationsbaum $c(n)$

bei n Instanzen berechnet. Damit kann ein Anomalie-Score berechnet werden [52]:

$$S(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2.26}$$

Zur Einordnung der Instanz wird wie folgt vorgegangen:

- $E(h(x)) \to c(n) \Rightarrow S \to 0,5$
- $E(h(x)) \to 0 \Rightarrow S \to 1$
- $E(h(x)) \to n-1 \Rightarrow S \to 0$

Wenn $S \approx 0,5$, weist der gesamte Datensatz keine klar abzugrenzende Anomalie auf. Bei $S \approx 1$ ist die Instanz eine Anomalie und wenn $S < 0,5$, dann ist die Instanz normal [52].

2.6.3 One-Class Support Vector Machine

Bei einer SVM wird zunächst jeder eingegebene Datenpunkt als Feature-Vektor umgewandelt und in einem Vektorraum repräsentiert. Es wird eine Hyperebene bestimmt, welche die Daten in zwei Klassen trennt. Das Ziel ist, die optimale Hyperebene zu finden. Diese wird durch die Vektoren, die ihr am nächsten sind (sogenannte Support Vektoren), definiert, wobei die Support Vektoren den größtmöglichen Abstand der beiden Klassen bilden. Anhand der optimalen Hyperebene wird eine Entscheidungsfunktion definiert, die den Daten eine Klasse zuordnet [53].

Eine OCSVM ist eine Variante einer SVM und wird im Folgenden genauer definiert. Seien $\{\vec{x}_1, ..., \vec{x}_n\}, \vec{x}_i \in X \subseteq \mathbb{R}^d$ die Input-Vektoren, wobei X der Input-Raum ist. Eine nichtlineare Funktion $\Phi(\vec{x}$ bildet einen Vektor $\vec{x}$ aus dem Input-Raum auf einen hochdimensionalen (oder sogar unendlichen) Feature-Raum F ab. Dann wird eine Hyperebene in F bestimmt als

$$f(\vec{x}) = \vec{W}^T \Phi(\vec{x}) - \rho \tag{2.27}$$

sodass möglichst viele der abgebildeten Vektoren $\{\Phi(\vec{x}_i), i = 1, ...n\}$ mit maximalem Abstand vom Ursprung in F separiert werden [54]. Dabei sind $\vec{W}$ der senkrechte Vektor zur Entscheidungsgrenze (definiert durch die Hyperebene) und ρ ein Verzerrungseffekt, welche sich ergeben aus dem Optimierungsproblem

$$min_{\vec{W},\xi,\rho} \frac{1}{2}\vec{W}^T\vec{W} - \rho + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \quad \text{sodass} \quad \vec{W}^T\Phi(\vec{x}_i) \geq \rho - \xi_i, \xi_i \geq 1, \tag{2.28}$$

wobei ξ_i die Variable für Punkt i ist, die ihm erlaubt auf der anderen Seite der Entscheidungsgrenze zu liegen (dargestellt in Abbildung 2.16) [55]. ν kontrolliert den Kompromiss zwischen Maximieren der Distanz zum Ursprung und Beinhalten der meisten Daten in der Region, die durch die Hyperebene erhalten wird [56], und ist eine obere Grenze für den Anteil an Anomalien in den Daten sowie eine untere Grenze für die Anzahl der Support Vektoren [55].

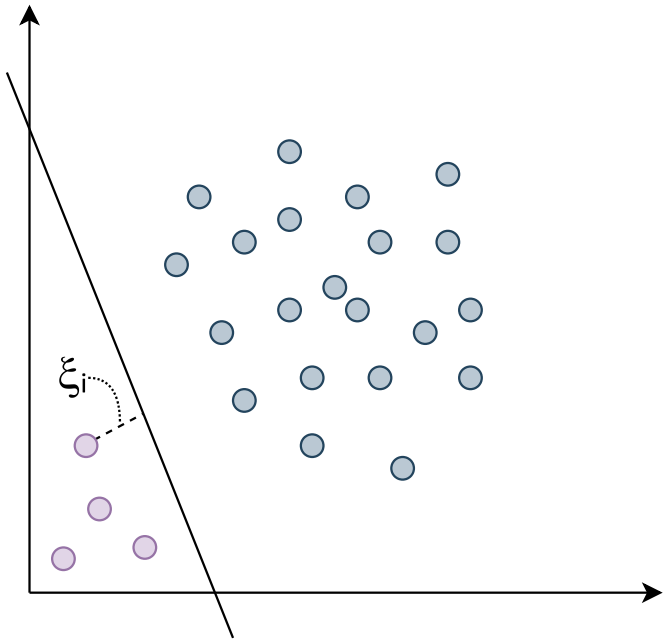


Abbildung 2.16: Zweidimensionales Beispiel der Entscheidungsgrenze einer OCSVM im Feature-Raum. Eigene Darstellung nach [55].

Mithilfe der Lagrange-Multiplikatoren α_i für jeden Vektor $\vec{x}_i$ kann das Optimierungsproblem aus Gleichung 2.28 in ein duales Problem umgewandelt werden, dessen Lösung auf

$$\vec{W} = \sum_{i=1}^{n} \alpha_i \Phi(\vec{x}_i), \tag{2.29}$$

wobei $0 \leq \alpha_i \leq \frac{1}{\nu n}$, führt. Durch Anwenden der Kernel-Methode wird das innere Produkt der zweier Vektoren im Feature-Raum F durch die Kernel-Funktion im Input-Raum X ersetzt. Dadurch kann die Hyperebene in F als nichtlineare Funktion in X dargestellt werden, um dort die Daten linear trennbar zu machen: Setzt man Gleichung 2.29 in 2.27

ein, ergibt sich

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i \Phi(\vec{x}_i)^T \Phi(\vec{x}) - \rho. \tag{2.30}$$

Mit einer Kernel-Funktion $K(\vec{x}_i, \vec{x}) = \Phi(\vec{x}_i)^T \Phi(\vec{x})$ im Input-Raum X lässt sich f schreiben als

$$f(\vec{x}) = \sum_{i=1}^{n} \alpha_i K(\vec{x}_i, \vec{x}) - \rho. \tag{2.31}$$

Es existieren verschiedene Kernel-Funktionen. Eine häufig genutzte Funktion ist die Radial Basis Function (RBF) [54]:

$$K(\vec{x}_i, \vec{x}) = exp(-\gamma ||\vec{x}_i - \vec{x}||^2) \tag{2.32}$$

mit $\gamma = \frac{1}{2\sigma^2}, \gamma > 0$ [57].

Anhand des Vorzeichens von $f(\vec{x})$ wird die Entscheidung getroffen, ob ein Vektor $\vec{x}$ eine Anomalie ist. Es gilt [54]:

- $f(\vec{x}) < 0 \Rightarrow \vec{x}$ ist Anomalie
- $f(\vec{x}) > 0 \Rightarrow \vec{x}$ ist keine Anomalie

2.6.4 Graph Deviation Network

Das Graph Deviation Network (GDN) ist ein aufmerksamkeitsbasiertes Graph Neural Network (GNN) für die Anomalieerkennung in multivariaten Zeitreihen von Sensormessdaten von CPS. Typischerweise bekommen GNNs Graphen als Input. Ein Graph ist eine Datenstruktur aus Knoten und Kanten, welche Knoten verbinden. Das GDN kennt die Kanten zu Beginn nicht, sondern lernt diese in einem Trainingsprozess. Der Algorithmus ist also Semi-supervised, da im Training nur normale Daten verwendet werden und anschließend in Testdaten Abweichungen als Anomalien erkannt werden. Die Knoten des Graphen repräsentieren Sensoren in einem CPS und die Kanten stellen die komplexen Abhängigkeiten zwischen den Sensoren dar. Die Trainingsdaten bestehen aus multivariaten Zeitreihen von Messdaten aus N Sensoren über T_{Train} Zeitstempel:

$$\vec{s}_{Train} = \{\vec{s}_{Train}^{(1)}, ..., \vec{s}_{Train}^{(T_{Train})}\} \tag{2.33}$$

Zu jedem Zeitpunkt t formen die Sensordaten $\vec{s}_{Train}^{(t)} \in \mathbb{R}^N$ einen N-dimensionalen Vektor. Der Algorithmus besteht aus vier Schritten, die im Folgenden näher erläutert werden. Eine

schematische Übersicht der Schritte ist in Abbildung 2.17 zu finden [18].

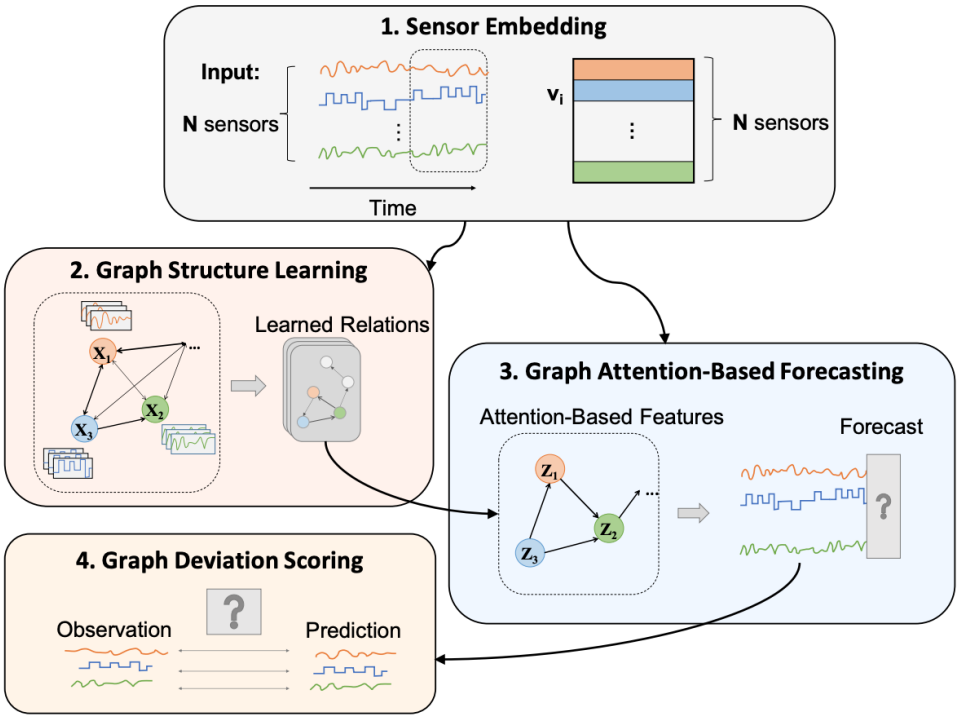


Abbildung 2.17: Überblick über die Funktionsweise des GDN. 1. Sensoreinbettung 2. Lernen der Graphstruktur 3. Graphbasierte aufmerksamkeitsbasierte Vorhersagen 4. Bewertung der Abweichung [18].

Sensoreinbettung

Da die Sensoren in einem CPS verschiedene physikalische Größen messen, denen ein physischer Gesamtprozess zugrunde liegt, existieren zwischen den verschiedenen Sensoren mit unterschiedlichen Eigenschaften komplexe Beziehungen. Um jeden Sensor flexibel mit den Faktoren, die sein Verhalten bestimmen, zu repräsentieren und die Beziehungen der Sensoren zu erfassen wird ein multidimensionaler Einbettungsvektor für jeden Sensor i definiert:

$$\vec{v}_i \in \mathbb{R}^d, i = 1, ..., N \tag{2.34}$$

Die Ähnlichkeit zwischen zwei Einbettungsvektoren $\vec{v}_i$ und $\vec{v}_j$ gibt an, wie stark zwei Sensoren i und j in Beziehung stehen [18].

Lernen der Graphstruktur

Die Beziehungen zwischen den Sensoren sollen nun in Form eines Graphen gelernt werden. Dafür wird ein gerichteter Graph mit den Sensoren als Knoten und den Abhängigkeitsbeziehungen als Kanten genutzt. Eine Kante von Sensor i zu Sensor j bedeutet also, dass i zum Modellieren des Verhaltens von j benutzt werden kann. Um den Graphen zu repräsentieren, wird die Adjazenzmatrix A verwendet. Ein Eintrag A_{ij} steht dabei für eine gerichtete Kante von i zu j. Falls vorab bekannt ist, welche Kanten sinnvoll sind, wird dies durch eine Menge aus *Candidate Relations* C_i für jeden Sensor i ausgedrückt, bestehend aus den Sensoren, von denen i abhängig sein könnte. Ansonsten beinhaltet C_i alle Sensoren außer i. Die Berechnung der Ähnlichkeit zwischen dem Einbettungsvektor von Knoten i und den Einbettungen seiner Kandidaten $j \in C_i$ führt zurück auf die Berechnung des entsprechenden Eintrags A_{ij} in der Adjazenzmatrix:

$$e_{ij} = \frac{\vec{v}_i^T \vec{v}_j}{||\vec{v}_i||\ ||\vec{v}_j||} \text{ für } j \in C_i \tag{2.35}$$

$$A_{ij} = \mathbb{1}\{j \in TopK(\{e_{ki} : k \in C_i\})\} \tag{2.36}$$

wobei $TopK$ die Indizes der k höchsten Werte der normalisierten Vektorprodukte e_{ij} sind. Die Zahl k kann zu Beginn festgelegt werden. Da im Training nur normale Daten verwendet werden, lernt der Algorithmus hier die normalerweise bestehenden Abhängigkeiten [18].

Graphbasierte aufmerksamkeitsbasierte Vorhersagen

Im nächsten Schritt soll das erwartete Verhalten jedes Sensors auf der Vergangenheit basierend vorhergesagt und mit dem beobachteten Verhalten verglichen werden. Somit kann man anschließend die Sensoren identifizieren, die von der Erwartung abweichen. Für die Vorhersagen wird ein ML-Modell genutzt. Der Input für das Modell zur Zeit t ist $\vec{x}^{(t)} \in \mathbb{R}^{N \times w}$ mit einem verschiebbaren Zeitfenster der Größe w über den historischen Zeitreihendaten:

$$\vec{x}^{(t)} := [\vec{s}^{(t-w)}, ..., \vec{s}^{(t-1)}] \tag{2.37}$$

Das Ziel des Modells ist es, $\vec{s}^{(t)}$ vorherzusagen [18].

Mithilfe eines Feature-Extraktors werden Informationen über einen Graphknoten basierend auf der gelernten Graphstruktur mit seinen Nachbarn verknüpft. Dazu werden die Einbettungsvektoren $\vec{v}_i$ der Sensoren einbezogen. Die aggregierte Repräsentation $\vec{z}_i$ von

Knoten (Sensor) i ist

$$\vec{z}_i^{(t)} = ReLU(\alpha_{i,i} W \vec{x}_i^{(t)} + \sum_{j \in \mathcal{N}(\rangle)} \alpha_{i,j} W \vec{x}_j^{(t)}), \tag{2.38}$$

wobei $W \in \mathbb{R}^{d \times w}$ eine trainierbare Gewichtematrix, $\vec{x}_i^{(t)} \in \mathbb{R}^w$ das Input-Feature und $\mathcal{N}(\rangle) = \{j : A_{ij} > 0\}$ die Menge der Nachbarn sind. Die Aufmerksamkeitskoeffizienten α werden berechnet als:

$$\vec{g}_i^{(t)} = \vec{v}_i \oplus W \vec{x}_i^{(t)} \tag{2.39}$$

$$\pi(i,j) = LeakyReLU(\vec{a}^{\,T}(\vec{g}_i^{(t)} \oplus \vec{g}_j^{(t)})) \tag{2.40}$$

$$\alpha_{i,j} = \frac{exp(\pi(i,j))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} exp(\pi(i,j))} \tag{2.41}$$

Dabei bedeutet $\oplus$ eine Konkatenation und $\vec{a}$ ist ein Vektor aus gelernten Koeffizienten für den Aufmerksamkeitsmechanismus [18]. Die Aktivierungsfunktionen ReLU und LeakyReLU sind in Unterabschnitt 2.5.5 definiert.

Diese Berechnungen werden für alle Knoten durchgeführt, sodass $\{\vec{z}_1^{(t)}, ..., \vec{z}_N^{(t)}\}$ vorliegt. Dann wird jedes $\vec{z}_i^{(t)}$ elementweise mit $\vec{v}_i$ multipliziert und das Ergebnis als Input in ein FCN gegeben. Der Output des Modells ist die Vorhersage für den Vektor der Sensorwerte zum Zeitpunkt t:

$$\vec{s'}^{\,t} = f_\theta([\vec{v}_1 \odot \vec{z}_1^{(t)}, ..., \vec{v}_N \odot \vec{z}_N^{(t)}]) \tag{2.42}$$

Als Kostenfunktion, die während des Trainings mit normalen Daten optimiert wird, wird der Mean Squared Error (MSE) zwischen Vorhersage $\vec{s'}^{\,t}$ und Beobachtung $\vec{s}^{\,t}$ verwendet [18]:

$$\mathcal{L}_{MSE} = \frac{1}{T_{Train} - w} \sum_{t=w+1}^{T_{Train}} ||\vec{s'}^{\,(t)} - \vec{s}^{\,(t)}||_2^2 \tag{2.43}$$

Bewertung der Abweichung

Nachdem der typische Trainingsprozess mit normalen Trainings- und Validierungsdaten abgeschlossen ist, können Testdaten verwendet werden, die Anomalien enthalten. Für jeden Sensor i wird für den Zeitpunkt t ein Fehlerwert berechnet:

$$Err_i(t) = |\vec{s}_i^{(t)} - \vec{s'}_i^{(t)}| \tag{2.44}$$

Dieser wird normalisiert zu

$$a_i(t) = \frac{Err_i(t) - \tilde{\mu}_i}{\tilde{\sigma}_i} \tag{2.45}$$

mit dem Median $\tilde{\mu}_i$ und dem Interquartilsabstand $\tilde{\sigma}_i$ von $Err_i(t)$. Der Gesamt-Anomalie-Score $A(t) = max(a_i(t))$ wird geglättet mit dem Simple Moving Average zu $A_s(t)$. Der maximale Fehler $A_s(t)$, der in den Validierungsdaten auftritt, wird als Grenzwert festgelegt. Wenn nun für einen Zeitstempel aus dem Testdatensatz $A_s(t)$ größer als dieser Grenzwert ist, erkennt das GDN diesen als Anomalie [18].

2.7 Verwandte Arbeiten

In netzwerkbasierten Szenarien gibt es viele unterschiedliche Ansätze zur Anomalieerkennung. Dazu gehören statistische, Regel-basierte, Vergleichs-basierte, Aktivitäts-basierte und andere Ansätze, wie beispielsweise Modell- oder Graph-basierte ML-Methoden [32]. Einige davon werden nachfolgend kurz vorgestellt, um die Relevanz der in dieser Arbeit verwendeten Algorithmen aufzuzeigen.

Einige Ansätze nutzen die Methode des Auto Regressive Integrated Moving Average (ARIMA). Dort wird mithilfe von Vergangenheitswerten eine Vorhersage über den nächsten Punkt gemacht, wobei eine Zufallsvariable (weißes Rauschen) hinzugefügt wird, um Schätz- und Vorhersagefehler mit einzubeziehen. Für jeden Datenpunkt wird ein neues Modell erstellt. Die Anomalieerkennung erfolgt durch einen Vergleich eines Modells mit Ausreißer-Punkten mit einem normal angepassten Modell. Wenn das Ausreißer-Modell besser passt, liegt eine Anomalie vor [32]. ARIMA wird in [58] für ein Frühwarnsystem in Kommunikationssoftware und Netzwerken verwendet.

Ein weiterer Ansatz ist Long Short-Term Memory (LSTM). Dabei handelt es sich um eine Art Recurrent Neural Network (RNN), das Reihenfolge-Abhängigkeiten lernen kann. Durch Selbstschleifen bleiben die Daten lange im Modell, ohne dass sie verloren gehen. Statt den Vorwärts-Verbindungen eines FCN hat das Modell Feedback-Verbindungen. LSTM konnte in [59] auf Smart Grid Daten gute Ergebnisse erzielen, jedoch die Vergleichsmethode

(K-means) insgesamt nicht übertreffen.

In [60] wird ein agentenbasierter Ansatz vorgestellt. Dazu wurde ein MAS entwickelt, in dem Agenten in verschiedenen Aufgabenfeldern eingesetzt werden. Diese sind Administration, Datenabstraktion, Datenverarbeitung sowie Analyse und Überwachung (erfassen, überwachen, Anomalien definieren). Die Anomalieerkennung ist keine explizite Aufgabe, sondern ergibt sich aus der Interaktion und Kommunikation der individuellen, verteilten, intelligenten Agenten. Anhand eines Prototypen für ein Energienetz konnte gezeigt werden, dass das MAS die Aufgben erfüllt und Anomalien erkennt.

2.7.1 Weitere Machine Learning Verfahren für Anomalieerkennung

Es existieren einige Methoden zur Anomalieerkennung in Netzwerken, die aus dem ML-Bereich kommen. ML-Verfahren erlauben Anomalieerkennungsalgorithmen, die sich an unterschiedliche Charakteristiken von Normalverhalten anpassen können und auf viele Anwendungen übertragbar sind [61].

Autoencoder

Autoencoder werden in [62] zur Anomalieerkennung durch Dimensionsreduktion eingesetzt. Es wird angenommen, dass die Variablen in den Daten miteinander korrelieren und in einem niederdimensionalen Subraum abgebildet werden können, in dem sich dann normale und anomale Daten signifikant unterscheiden. Die Funktionsweise eines Autoencoders zur Anomalieerkennung ist in Unterabschnitt 2.6.1 beschrieben. In [62] werden lineare Principal Component Analysis (PCA), Kernel PCA, ein Autoencoder und ein rauschunterdrückender Autoencoder zur Dimensionsreduktion eingesetzt und verglichen. PCA (Hauptkomponentenanalyse) ist eine Methode der multivariaten Statistik zur Dimensionsreduktion [63]. Auf synthetischen Daten, die mithilfe des Lorenz-Systems erzeugt wurden, konnten die Autoencoder die Aufgabe der Anomalieerkennung gut erfüllen, im Gegensatz zur linearen PCA. Dabei übertraf zudem der rauschunterdrückende Autoencoder den normalen. Es wurden außerdem zwei Real-Datensätze aus dem Bereich der Raumfahrzeug-Telemetrie getestet, bei denen die Autoencoder gleich gut oder besser als die Kernel PCA performten.

In [50] werden ein Autoencoder und ein Convolutional Autoencoder (CAE) mit weiteren Ansätzen zur Anomalieerkennung verglichen. Ein CAE nutzt statt Fully Connected Layern einen Faltungs-Layer im Encoder und einen Entfaltungs-Layer im Decoder. Die Testdaten stammen aus dem NSL-KDD Datenset (ein öffentlich verfügbares Benchmark-Datenset zum Testen für Anomalieerkennungsmethoden, bestehend aus Internet-Verkehrsdaten). Im Vergleich erreichen der normale Autoencoder und der CAE eine höhere AUC als PCA, k Nearest Neighbors (k-NN) und SVMs. Im Vergleich mit Triangle Area based Nearest

Neighbors (TANN) ist der normale Autoencoder nicht ganz so gut, aber der CAE erreicht die gleiche AUC.

Isolation Forest

Da beim Isolation Forest Algorithmus auf Distanz- oder Dichtemethoden verzichtet wird, benötigt er wenig Rechenaufwand bei gleichzeitig guten Ergebnissen in der Anomalieerkennung. In [51] wird eine Erweiterung des Algorithmus, ein Simulated Annealing Isolation Forest (SA-Isolation Forest), vorgestellt. Simulated Annealing (simulierte Abkühlung) ist ein heuristisches Approximationsverfahren, das die durch stochastische Faktoren auftretenden redundanten Bäume im Isolation Forest reduziert. Der SA-Isolation Forest wird mit einem normalen Isolation Forest und der Local Outlier Factor (LOF)-Methode (dichtebasiert) anhand verschiedener Realdatensets (medizinischer Hintergrund, Internet-Verkehrsdaten, Raumfahrt-Daten) verglichen. Der Isolation Forest erzielt etwas bessere Ergebnisse als LOF und der SA-Isolation Forest ist deutlich besser als die anderen Ansätze.

Ähnliche Untersuchungen werden in [52] gemacht. Dort wird die Performanz eines Isolation Forest, ORCA (eine k-NN-basierte Methode), LOF und eines Random Forest (RF) (ebenfalls Nutzen von Bäumen) verglichen. Dazu wird sowohl auf synthetischen als auch Realdaten getestet. Die Realdaten stammen aus verschiedenen Kontexten, darunter militärische Netzwerkumgebungen, Satellitendaten oder Daten mit medizinischem Hintergrund (Breast Cancer Wisconsin Datenset). Ab einer Größe von mehr als 1000 Datenpunkten im Datensatz übertrifft der Isolation Forest ORCA. Da LOF eine hohe Rechenzeit und RF einen hohen Speicherbedarf hat, werden diese nur für kleinere Datensätze verwendet. Der Isolation Forest ist besser als der RF und in acht von neun Fällen auch besser als LOF. Die Ergebnisse weiterer Experimente sind eine schnelle und gute Performanz des Isolation Forest für hochdimensionale Probleme mithilfe von Attributselektion. Ebenfalls wird eine gute Performanz erreicht, wenn nur auf normalen Daten trainiert wird, sowie eine hohe Genauigkeit und Effizienz für große Datensätze.

One-Class Support Vector Machine

In einer frühen Arbeit wurde die erste Idee einer OCSVM so erweitert, dass nicht nur der Ursprung des Feature-Raums, sondern alle Punkte ”ausreichend nah” am Ursprung in die Klasse ”Anomalie” eingeordnet werden (vgl. Unterabschnitt 2.6.3) [64]. In einem Test auf Simulationsdaten eines US Air Force Local Area Network (LAN) wurde diese OCSVM mit einer Clustermethode, der Naive Bayes Methode, der k-NN-Methode und einer Standard-SVM verglichen, wobei die OCSVM alle anderen Ansätze übertraf.

Auch in aktuellen Arbeiten finden OCSVMs Anwendung. In [65] wird eine OCSVM

zur Anomalieerkennung kombiniert mit mit dem K-Means-Algorithmus, um die erkannten Anomalien anhand eines Ähnlichkeitsindex in Arten von Angriffen zu unterscheiden. Im Test mit Bitcoin-Transaktionsdaten konnte die OCSVM hier sehr gute Ergebnisse erzielen.

Graph Deviation Network

Der in Unterabschnitt 2.6.4 vorgestellte graphbasierte Ansatz wurde in der gleichen Arbeit auf Daten von zwei realen Wasserversorgungs- und aufbereitungsanlagen getestet. Das GDN wurde mit den Methoden PCA, k-NN, Feature Bagging (FB), Autoencoder, Deep Autoencoding Gaussian Model (DAGM), LSTM-Variational Autoencoder (VAE) und Multivariate Anomaly Detection with Generative Adversarial Network (MAD-GAN) verglichen. Für beide Datensätze erzielte das GDN die höchsten Werte für *Recall*, *Precision* und *F1-Score*. Darüber hinaus lassen sich mithilfe des GDN die anomalen Knoten und damit der Ursprung jeder Anomalie lokalisieren [18].

2.8 Entwicklungsumgebungen und Bibliotheken für Machine Learning

Python ist eine beliebte und weit verbreitete Programmiersprache, da sie die allgemeine und einfache Nutzbarkeit einer Programmier-Hochsprache mit einer Skriptsprache vereint. Es gibt für Python viele Bibliotheken für Daten, Visualisierung, ML und vieles Weitere. Sie bietet hohe Funktionalität für allgemeine aber auch spezielle Einsatzgebiete. Darüber hinaus gibt es die Möglichkeit, in der Konsole oder anderen Umgebungen direkt mit dem Code zu interagieren [66].

Nachfolgend werden einige relevante Entwicklungsumgebungen und Bibliotheken für ML mit Python vorgestellt.

Jupyter-Lab

Jupyter-Lab ist eine browserbasierte, interaktive Entwicklungsumgebung mit einer flexiblen Oberfläche für verschiedene Arbeiten in den Bereichen Data Science, Wisschenschaftliches Rechnen und ML. Durch das modulare Design gibt es viele Funktionalitäten und Erweiterungen. Neben Python werden alle gängigen Programmiersprachen unterstützt [67].

Scikit-Learn

Scikit-Learn ist eine Open Source Bibliothek für ML und stellt viele hochentwickelte Algorithmen zur Verfügung. Die Bibliothek wird in vielen Bereichen der Wirtschaft und

Forschung eingesetzt [66].

NumPy

NumPy ist das grundlegende Paket in Python für wissenschaftliches Rechnen sowie die Standard-Datenstruktur in Scikit-Learn. Es bietet Funktionalitäten für mehrdimensionale Arrays, lineare Algebra, Fourier-Transformationen und Pseudo-Zufallszahlen [66].

Pandas

Pandas ist eine Bibliothek zur Datenaufbereitung und -analyse. Die Standard-Datenstruktur DataFrame basiert auf Tabellen und bietet viele Methoden zum Modifizieren und Verarbeiten der Tabellen. Dazu gehören auch Suchanfragen und Verbindungsoperationen. Dabei dürfen in einem DataFrame in jeder Spalte unterschiedliche Typen enthalten sein. Pandas unterstützt außerdem das Einlesen und Exportieren vieler Dateiformate [66].

TensorFlow

TensorFlow ist eine Open Source Plattform für ML. Sie verfügt über viele Tools und Bibliotheken. Für Python gibt es Keras, eine Deep Learning (DL) API, die eine schnelle, einfache Ausführung und Evaluation von DL-Algorithmen ermöglicht [68].

PyOD

PyOD ist eine Open-Source Bibliothek in Python für die Erkennung von Anomalien in multivariaten Daten. Eine Vielzahl von Algorithmen sowie nützliche Funktionen zur Evaluation und Visualisierung der Ergebnisse werden bereitgestellt, wobei die Implementierungen auf Scikit-Learn oder Tensorflow basieren. Somit kann durch wenige Anpassungen die gewünschte Analyse durchgeführt werden [69].

3 Vorarbeiten

In Vorbereitung auf diese Abschlussarbeit werden im Rahmen einer Beschäftigung als Wissenschaftliche Hilfskraft einige Vorarbeiten geleistet, in denen ”klassische” und graphbasierte Anomalieerkennung anhand synthetischer Daten verglichen werden.

In diesem Kapitel werden zunächst das Szenario, das Vorgehen und das Ziel der Vorarbeit beschrieben. Anschließend werden der Datengenerator und die ”klassischen” sowie graphbasierten Ansätze im Detail vorgestellt. Zum Schluss folgt eine Präsentation der Ergebnisse, wobei der Fokus auf dem Vergleich der verschiedenen Ansätze liegt.

3.1 Szenario und Vorgehen

Für die Untersuchungen soll das Szenario möglichst simpel gehalten werden, da der Fokus auf der Erkennung der verschiedenen Arten von Anomalien liegt. Es wird die Annahme getroffen, dass eine Zahl von Agenten in einem Energiesystem nach einer zu Beginn festgelegten Topologie miteinander kommuniziert, indem jeder Agent seinen zugewiesenen Nachbarn Nachrichten schickt. Dabei sind keinerlei weitere Eigenschaften der Agenten bekannt und die Nachrichten bestehen aus reellen Zahlenwerten ohne weitere Informationen. Die Nachrichten werden in gleichbleibenden zeitlichen Abständen gesendet, sodass Daten als Zeitreihen vorliegen. Der genaue Aufbau der Daten wird in Abschnitt 3.2 vorgestellt. In die Nachrichten werden dann verschiedene Arten von Anomalien eingefügt, welche von der Anomalieerkennung identifiziert werden sollen. Dies sind Anomalien in den gesendeten Werten und in der Kommunikationstopologie, wobei Punkt- und Teilsequenz-Anomalien vorkommen. Die Details über die Anomalien werden in Unterabschnitt 3.2.1 beschrieben. Dafür wird ein Datengenerator (siehe Abschnitt 3.2) in Python geschrieben, der die benötigten Daten in einem Tabellenformat (CSV und xml) ausgibt. Da die Erkennung der verschiedenen Anomaliearten verglichen werden soll, sind mehrere Datensätze nötig. Es gibt einen Datensatz ohne Anomalien für das Training von Semi-supervised Ansätzen. Die Daten mit Anomalien teilen sich auf in einen Datensatz mit Werteanomalien, einen mit Topologieanomalien und einen mit beiden Anomaliearten. Zusätzlich soll verglichen werden, ob die Anomalieerkennung aus globaler Sicht des Systems mit allen Agenten (network-based) oder aus Sicht eines Agenten (host-based) bessere Ergebnisse liefert. Somit gibt

es jeden Datensatz einmal mit allen Nachrichten und einmal mit nur einem Agenten als Sender.

Für die ”klassische” Anomalieerkennung werden die Algorithmen Autoencoder (siehe Unterabschnitt 2.6.1), Isolation Forest (siehe Unterabschnitt 2.6.2) und One-Class Support Vector Machine (OCSVM) (siehe Unterabschnitt 2.6.3) gewählt. Diese werden häufig in verschiedenen Anwendungsbereichen eingesetzt und erzielen dort gute Ergebnisse in der Anomalieerkennung (siehe Unterabschnitt 2.7.1). Als Programmierumgebung wird JupyterLab genutzt, da es eine einfache und flexible Programmierung in Python erlaubt. Für die ”klassischen” Algorithmen wird die Python-Bibliothek PyOD verwendet. Mithilfe einer automatischen Suche (*Grid Search* von *Scikit-Learn*) werden die Parameter der Algorithmen optimiert. Es wird mit jedem Algorithmus je ein Modell erstellt mit jeweils network-based- und host-based-Daten für jede Anomalieart sowie für die Kombination aus beiden Anomaliearten.

Eine vollständige Implementierung in Python des in Unterabschnitt 2.6.4 vorgestellten graphbasierten Ansatzes wird von den Autoren öffentlich auf Github zur Verfügung gestellt, welche mit einigen Anpassungen genutzt wird. Auch hier werden für den network-based- und den host-based-Ansatz je drei individuelle Modelle trainiert und evaluiert für Werteanomalien, Topologieanomalien und deren Kombination.

Ziel

Das Ziel der Vorarbeiten ist, mithilfe von synthetischen Daten in einem agentenbasierten Szenario zu untersuchen, ob gängige Machine Learning (ML)-Algorithmen zur Erkennung von verschiedenen Anomaliearten gleich gut geeignet sind und ob graphbasierte Ansätze diese Algorithmen für bestimmte Anomaliearten oder sogar insgesamt übertreffen.

Der Vergleich der Ansätze erfolgt anhand der in Unterabschnitt 2.5.7 eingeführten Metriken Area Under Curve (AUC) und *Precision* sowie visuell durch *Confusion Matrizen*. Dabei werden folgende Vergleiche gezogen:

- Wie gut erkennt jeder Algorithmus die verschiedenen Anomaliearten?
- Welche Anomalieart erkennen die klassischen Ansätze besser?
- Welche Anomalieart erkennt der graphbasierte Ansatz besser?
- Werden im network-based oder im host-based Ansatz Anomalien besser erkannt?
- Performen die klassischen Ansätze oder der graphbasierte Ansatz insgesamt besser?

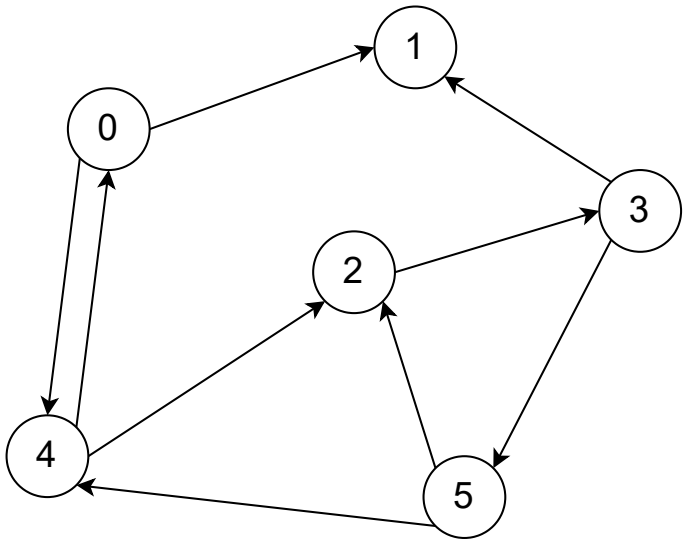


Abbildung 3.1: Topologie des erzeugten Kommunikationsgraphen. Die Kommunikation findet in Pfeilrichtung statt. Eigene Darstellung.

3.2 Datengenerator

Der Datengenerator erstellt synthetische Daten einer selbst festgelegten Graphstruktur, in der Agenten Nachrichten an bestimmte andere Agenten anhand der definierten Topologie senden. In der Umsetzung wurden sechs Graphknoten erzeugt, die wie in Abbildung 3.1 dargestellt an ihre Nachbarknoten Nachrichten senden. Dafür wird eine Adjazenzmatrix erstellt, in welcher die Kanten repräsentiert sind. Es wurde ein gerichteter Graph gewählt, also kann z.B. Knoten 0 an Knoten 1 senden, aber nicht umgekehrt. In einem realistischen System, welches beispielsweise auf Winzent basiert, sind die Kommunikationskanäle jedoch bidirektional.

Jedem erstellten Knoten wird zufällig eine periodische Funktion (Sinus, Kosinus oder Sinus∗Kosinus) zugewiesen, deren Funktionswert in Abhängigkeit vom Wert des Zeitstempels mit einer Zufallszahl zwischen $0,5$ und $1,5$ multipliziert und als Nachrichtenwert verwendet wird. Dazu wird ein Pandas DataFrame erzeugt mit einer Spalte für den Zeitstempel, einer Spalte für den Sender und je einer Spalte für jeden existierenden Knoten, in deren entsprechende Zeilen der Nachrichtenwert eingetragen wird. Für denselben Zeitstempel gibt es also eine Zeile pro Knoten (Sender), in der der Wert bei den Empfängerknoten eingetragen wird. Alle anderen Einträge werden auf 0 gesetzt. So ergibt sich die Topologie des Kommunikationsgraphen aus den Einträgen, die $\neq 0$ sind. Das DataFrame wird in einer Datei in Tabellenformat (CSV und xml) ausgegeben.

3.2.1 Einfügen von Anomalien

Werteanomalien

Punktuelle Werteanomalien werden erzeugt, indem zufällige Werte mit 1000 multipliziert werden. Dies ergibt deutliche Ausreißer in den Daten. Für das Einfügen von kollektiven Werteanomalien werden die Werte einiger Zeiträume mit einer zufälligen Zahl zwischen 100 und 200 multipliziert.

Topologieanomalien

Um punktuelle und kollektive Topologieanomalien zu erzeugen, wird für verschiedene Zeitpunkte bzw. Zeiträume die Adjazenzmatrix manipuliert, sodass die Einträge in den Daten sich entsprechend ändern. Hierbei kommen Kanten hinzu oder fallen weg, wobei immer nur ein Fall pro Zeitraum eintritt.

3.3 "Klassische" Ansätze

Im Folgenden wird die Implementierung der klassischen Ansätze in PyOD vorgestellt.

3.3.1 Autoencoder

PyOD verwendet für den Autoencoder Keras von TensorFlow. Um die bestmöglichen Ergebnisse zu erzielen, werden die veränderlichen Parameter des Modells optimiert. Dafür wird für jede Parameter-Kombination ein Modell erstellt, welches jeweils auf Daten mit Werteanomalien, Topologieanomalien und der Kombination aus beiden Anomaliearten getestet wird. Dies wird mithilfe der *Grid Search* von *Scikit-Learn* durchgeführt.

Network-based Ansatz

Die Größe des Datensatzes für das Training beträgt 17280 Zeilen. Die Features sind der Zeitstempel, Sender und jeder Knoten als Empfänger. In dem in Abbildung 3.1 dargestellten Setting mit sechs Knoten ergeben sich also acht Features. Daher gibt es acht Neuronen im Input- und Output-Layer des Autoencoders. Die Anzahl der versteckten Schichten wurde variiert und die besten Ergebnisse lassen sich mit drei versteckten Schichten erzielen. Die Anzahl der Neuronen in den Schichten ist $\{8, 6, 4, 6, 8\}$. Die Aktivierungsfunktionen für die versteckten Schichten und die Output-Schicht werden auf den Standardbelegungen Rectified Linear Unit (ReLU) bzw. Sigmoid belassen, da diese in der Literatur häufig verwendet werden und zu guten Ergebnissen führen. Ebenso werden die Kostenfunktion Mean Squared Error (MSE) und der *Optimizer* Adam verwendet. Die *Dropout*-Rate und

die *Batch*-Größe wurden variiert und die besten Ergebnisse lieferten eine *Dropout*-Rate von $0,1$ und eine *Batch*-Größe von 16.

Anschließend liegt ein Modell vor, welches auf die normalen Daten trainiert ist. Es wird dann auf drei Datensätzen, die Werteanomalien, Topologieanomalien und beide Arten enthalten, ausgeführt und die Ergebnisse werden verglichen.

Host-based Ansatz

Für den host-based Ansatz wird der Datensatz jeweils nach einem Knoten gefiltert, sodass insgesamt sechs Datensätze mit jeweils einer Größe von 2880 Zeilen entstehen. Die Anzahl der Features bleibt gleich. Für jeden Datensatz wird ein Modell erstellt, wobei nicht alle Modelle in den Ergebnissen evaluiert werden. Da in den Daten nicht bei allen Knoten ausreichend Anomalien enthalten sind, wird pro Anomalieart, bzw. Kombination, ein Modell gewählt, welches für einen Knoten trainiert ist, der die entsprechende Anomalieart in den Testdaten aufweist. Die Modellparameter werden vom network-based Ansatz übernommen.

3.3.2 Isolation Forest

Da der Isolation Forest direkt mit Daten, die Anomalien enthalten, trainiert werden kann, wird für jede Art von Anomalie und die Kombination der beiden Anomaliearten ein Modell erstellt und die Parameter werden individuell optimiert. Dies wird für den network-based und den host-based Ansatz durchgeführt, sodass insgesamt sechs Modelle evaluiert werden.

Network-based Ansatz

Beim Erstellen des Modells wird vor dem Training die *contamination* festgelegt, die den Anteil anomaler Instanzen in den Daten angibt. Diese wird hier mit $0,1$, also 10% Anomalien, belegt. Der Parameter *n_estimators* gibt die Anzahl der Isolationsbäume an, die erstellt werden. Die besten Ergebnisse werden mit 150 Isolationsbäumen erreicht. Weiterhin wird die Anzahl der Features untersucht, die für die Isolationsbäume betrachtet werden. Der optimale Wert liegt hier bei 8, also werden alle Features einbezogen. Die restlichen Parameter werden auf ihren Standardbelegungen belassen.

Host-based Ansatz

Wie auch für den Autoencoder wird der Datensatz aufgeteilt, sodass je ein Modell pro Knoten trainiert wird. Dabei werden die Parameter aus dem network-based Ansatz übernommen. Für die Evaluation werden die gleichen Knoten wie beim Autoencoder je Anomalieart evaluiert.

3.3.3 One-Class Support Vector Machine

Die OCSVM wird ebenfalls direkt auf Daten ausgeführt, die Anomalien enthalten.

Network-based Ansatz

Für den network-based Ansatz wird für jeden anomalen Datensatz ein Modell erstellt, dessen Parameter optimiert werden. Der *kernel* ist die Kernel-Funktion (siehe Unterabschnitt 2.6.3) und die besten Ergebnisse werden mit der Radial Basis Function (RBF) erzielt. Der Parameter *nu* ist das ν aus Unterabschnitt 2.6.3 und gibt eine obere Grenze für den Anteil an Anomalien sowie eine untere Grenze für den Anteil an Support-Vektoren an. Die besten Ergebnisse liefert $nu = 0,2$. Die *contamination* wird auf $0,2$ gesetzt. Die restlichen Parameter werden mit ihren Standardbelegungen verwendet.

Host-based Ansatz

Für die Untersuchungen im host-based Szenario werden auch hier die Daten nach Knoten aufgeteilt und es wird ein Modell je Knoten erstellt. Dabei werden die Parameter aus dem network-based Ansatz übernommen. Zur Evaluation wird für jede Anomalieart ein Knoten gewählt, der die entsprechende Anomalie am meisten aufweist.

3.4 Graphbasierter Ansatz

Die Implementierung des Graph Deviation Network (GDN) (siehe Unterabschnitt 2.6.4) wird auf GitHub von den Entwicklern öffentlich zur Verfügung gestellt. Diese wird mit kleineren Anpassungen verwendet. Das Training findet auf Normaldaten ohne Anomalien statt. In einem Ordner für die Daten müssen diese als CSV-Datei sowie eine Text-Datei mit den Features bereitgestellt werden. In die Testdaten, welche Anomalien enthalten, muss eine Spalte mit der Bezeichung ”attack” eingefügt werden, damit bekannt ist, bei welchen Daten Anomalien vorliegen und die Ergebnisse evaluiert werden können.

Es wurden kleinere Code-Teile eingefügt, um sich eine Confusion Matrix als Bild und Kennzahlen zur Performanz als CSV-Datei ausgeben zu lassen.

Um die bestmöglichen Ergebnisse zu erhalten wurden verschiedene Werte für die Parameter *Batch_Size*, *sliding_window* (das Zeitfenster, das für die Berechnung des erwarteten Werts betrachtet wird) und *topk* (die Anzahl der nächsten Nachbarn, die betrachtet werden) ausprobiert. Die besten Ergebnisse lassen sich mit *Batch_Size*$= 32$, *sliding_window*$= 10$ und *topk*$= 5$ erzielen.

3.5 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der verschiedenen Algorithmen zur Anomalieerkennung für den network-based und den host-based Ansatz mit jeweils Anomalien in den Werten, in der Topologie und der Kombination aus beiden Arten präsentiert. Alle zugehörigen Confusion Matrizen befinden sich im Anhang unter Abschnitt A.2.

3.5.1 Autoencoder

Da der Autoencoder mit normalen Daten trainiert wird, gibt es je ein Modell für den network-based und den host-based Ansatz, welches auf den drei anomalen Datensätzen ausgeführt wird.

Network-based Ansatz

Tabelle 3.1 zeigt die AUC und Precision je nach Anomalieart für den network-based Ansatz.

Anomalieart	AUC	Precision
Werteanomalien	0,87	0,79
Topologieanomalien	0,53	0,14
Alle	0,67	0,18

Tabelle 3.1: Performanz des Autoencoders je Anomalieart im network-based Ansatz.

Host-based Ansatz

In Tabelle 3.2 ist die Performanz des Autoencoders für den host-based Ansatz dargestellt. Für das Testen auf Daten mit Werteanomalien wird Knoten 0 gewählt, für Topologieanomalien Knoten 1 und für beide Arten Knoten 5, da dort jeweils die meisten Anomalien der jeweiligen Art bzw. nahezu gleich viele Anomalien jeder Art vorliegen.

Anomalieart	AUC	Precision
Werteanomalien	0,98	0,96
Topologieanomalien	0,72	0,03
Alle	0,64	0,19

Tabelle 3.2: Performanz des Autoencoders je Anomalieart im host-based Ansatz.

3.5.2 Isolation Forest

Im Folgenden werden die Ergebnisse der Isolation Forest Modelle für den network-based und den host-based Ansatz dargestellt.

Network-based Ansatz

Tabelle 3.3 enthält die Ergebnisse des Isolation Forest für die unterschiedlichen Anomaliearten im network-based Ansatz.

Anomalieart	AUC	Precision
Werteanomalien	0,80	0,63
Topologieanomalien	0,37	0,26
Alle	0,63	0,40

Tabelle 3.3: Performanz des Isolation Forest je Anomalieart im network-based Ansatz.

Host-based Ansatz

In Tabelle 3.4 ist die Performanz der Isolation Forest Modelle des host-based Ansatzes zu sehen.

Anomalieart	AUC	Precision
Werteanomalien	0,99	0,89
Topologieanomalien	0,98	0,50
Alle	0,57	0,33

Tabelle 3.4: Performanz des Isolation Forest je Anomalieart im host-based Ansatz.

3.5.3 One-Class Support Vector Machine

Nachfolgend werden die Ergebnisse der OCSVM präsentiert.

Network-based Ansatz

In Tabelle 3.5 ist die Performanz der OCSVM Modelle für die verschiedenen anomalen Datensätze im network-based Szenario gezeigt.

Anomalieart	AUC	Precision
Werteanomalien	0,95	0,78
Topologieanomalien	0,61	0,004
Alle	0,69	0,20

Tabelle 3.5: Performanz der OCSVM je Anomalieart im network-based Ansatz.

Host-based Ansatz

Tabelle 3.6 zeigt die Ergebnisse der OCSVM Modelle für den host-based Ansatz.

Anomalieart	AUC	Precision
Werteanomalien	0,91	0,87
Topologieanomalien	0,55	0,55
Alle	0,93	0,45

Tabelle 3.6: Performanz der OCSVM je Anomalieart im host-based Ansatz.

3.5.4 Graph Deviation Network

Das GDN wurde für den network-based Ansatz mit normalen Daten trainiert und mit den anomalen Datensätzen getestet. In allen Fällen wurden mehr als die Hälfte der Datenpunkte als Anomalie klassifiziert, weshalb dieser Ansatz nicht weiter verfolgt wurde.

Die Ergebnisse mit den host-based Daten sind in Tabelle 3.7 dargestellt.

Anomalieart	AUC	Precision
Werteanomalien	0,92	0,39
Topologieanomalien	0,97	0,71
Alle	0,83	0,71

Tabelle 3.7: Performanz des GDN je Anomalieart im host-based Ansatz.

3.6 Evaluation

In diesem Abschnitt werden die Ergebnissse diskutiert und die Vergleichsfragen aus Abschnitt 3.1 beantwortet.

Wie gut erkennt jeder Algorithmus die verschiedenen Anomaliearten?

Der Autoencoder erkennt im network-based Ansatz die Werteanomalien recht gut. Bei den Datensätzen mit Topologieanomalien und beiden Anomaliearten werden die Anomalien nicht gut erkannt. Im host-based Szenario werden die Topologieanomalien und beide Arten nicht gut erkannt (Precision von 0,03 bzw. 0,19), während Werteanomalien sehr gut erkannt werden (Precision von 0,96).

Die Ergebnisse des Isolation Forest sind denen des Autoencoders ähnlich. Im network-based Ansatz werden Werteanomalien mäßig gut (Precision von 0,63) und die anderen Anomalien schlecht (Precision von 0,26 bzw. 0,40) erkannt. Bei den Werteanomalien liegt die Performanz unter der des Autoencoders. Die Ergebnisse für den host-based Ansatz sind ebenfalls mit dem Ergebnis des Autoencoders vergleichbar (Precision von 0,89 für Werteanomalien und 0,50 für Topologieanomalien). Allerdings ist die Performanz für die Daten

mit beiden Anomaliearten schlechter als für Daten mit Topologieanomalien (Precision von $0,33$).

Die OCSVM erkennt Werteanomalien im network-based und im host-based Szenario ähnlich gut (Precision von $0,78$ bzw. $0,87$ bei einer AUC von $0,95$ bzw. $0,91$), wobei die Performanz leicht unter der der anderen klassischen Algorithmen im host-based Ansatz liegt. Topologieanomalien und beide Anomaliearten werden im network-based Ansatz nicht gut erkannt (Precision von $0,004$ bzw. $0,20$). Im host-based Ansatz werden diese allerdings im Vergleich zum Autoencoder und Isolation Forest am besten erkannt (Precision von $0,55$ für Topologieanomalien und $0,45$ für beide Anomaliearten).

Die niedrigen Kennzahlen für die Kombination aus zwei Anomaliearten zeigen, dass die klassischen Modelle schlecht an verschiedene Arten von Anomalien angepasst werden können.

Das GDN wird nur auf den host-based Daten evaluiert. Hier werden Topologieanomalien besser (Precision von $0,71$) erkannt als Werteanomalien (Precision von $0,39$) und die Kombination aus beiden Arten wird etwas schlechter erkannt als Werteanomalien (niedrigere AUC von $0,83$). Das GDN liefert von allen Algorithmen die beste Performanz für Topologieanomalien und die schlechteste Performanz für Werteanomalien, wobei diese insgesamt noch als akzeptabel gesehen werden kann.

Welche Anomalieart erkennen die klassischen Ansätze besser?

In Abbildung 3.2 ist die erreichte Precision der verschiedenen Ansätze für Daten mit Wertenanomalien dargestellt. Abbildung 3.3 zeigt die Precision der Verfahren für Daten mit Topologieanomlien. Es ist aus dem Vergleich der Abbildungen erkennbar, dass alle drei klassischen Ansätze deutlich bessere Ergebnisse für Werteanomalien als für Topologieanomalien liefern, wobei sie auch den graphbasierten Ansatz für Wertenanomalien übertreffen.

Welche Anomalieart erkennt der graphbasierte Ansatz besser?

Der graphbasierte Ansatz erkennt Topologieanomalien besser als Werteanomalien, wie aus Abbildung 3.2 und 3.3 zu entnehmen ist. Dabei liefert er die besten Ergebnisse für Topologieanomalien im Vergleich zu den klassischen Ansätzen. In Abbildung 3.4 ist zu sehen, dass das GDN auch bei der Erkennung der Kombination aus beiden Anomaliearten alle klassischen Ansätze übertrifft.

Werden im network-based oder im host-based Ansatz Anomalien besser erkannt?

Die Ergebnisse zeigen (mit Ausnahme des Autoencoders für Topologieanomalien) eine höhere Performanz der klassischen Modelle mit Daten aus dem host-based Szenario für jeweils

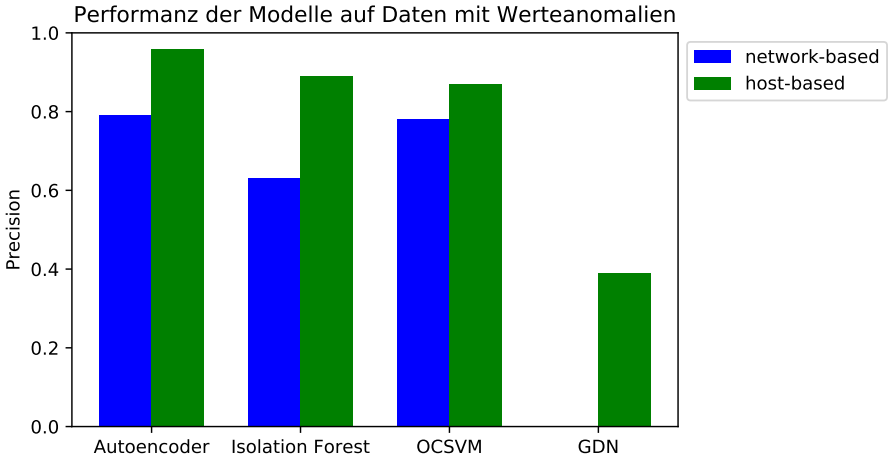


Abbildung 3.2: Performanz der Modelle für Werteanomalien im Vergleich.

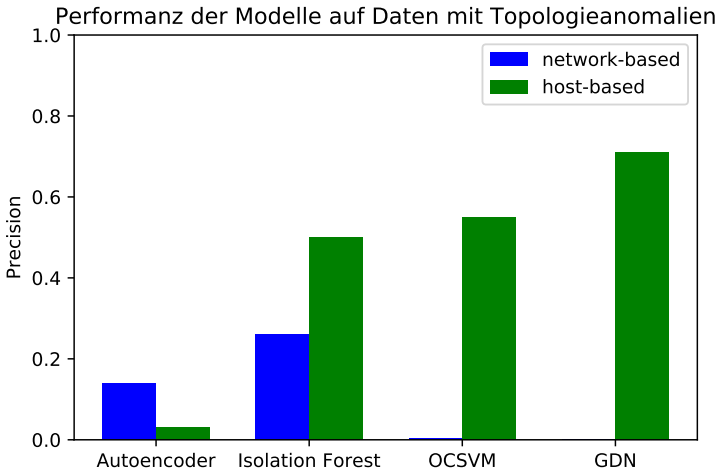


Abbildung 3.3: Performanz der Modelle für Topologieanomalien im Vergleich.

nur Werte- und Topologieanomalien (siehe Abbildung 3.2 und 3.3). Für die Kombination aus beiden Anomaliearten sind die Ergebnisse der klassischen Modelle im network-based Ansatz teilweise etwas besser als im host-based Ansatz, allerdings sind sie insgesamt nicht gut (siehe Abbildung 3.4). Das graphbasierte Verfahren ist für den network-based Ansatz nicht verwendbar.

Anhand dieser Ergebnisse wurde für die Masterarbeit die Entscheidung getroffen, mit host-based Daten zu arbeiten.

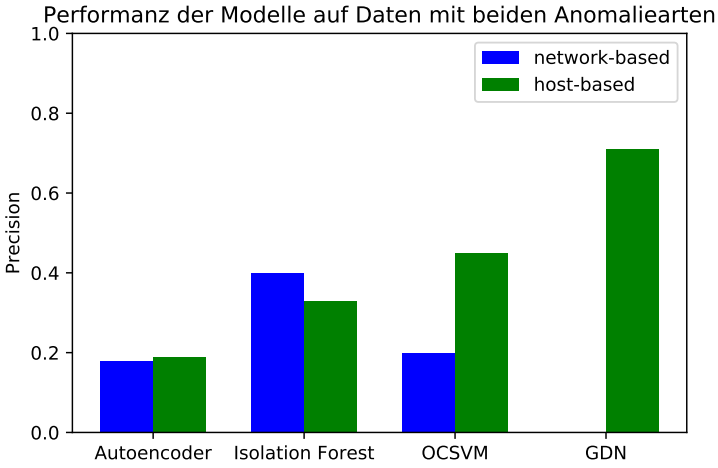


Abbildung 3.4: Performanz der Modelle für beiden Anomaliearten im Vergleich.

Performen die klassischen Ansätze oder der graphbasierte Ansatz insgesamt besser?

Die klassischen Ansätze liefern die besten Ergebnisse für Werteanomalien (siehe Abbildung 3.2). Das GDN zeigt die beste Performanz auf Topologieanomalien und der Kombination aus zwei Anomaliearten, wobei es die klassischen Ansätze übertrifft. Insgesamt kann jedoch kein Algorithmus alle anderen für jede Anomalieart übertreffen.

3.7 Fazit und Ausblick

In den Vorarbeiten wurden für ein stark vereinfachtes Agentensystem synthetische Kommunikationsdaten erstellt, welche auf eingefügte Anomalien untersucht wurden. Dabei wurden Werteanomalien und Topologieanomalien einzeln und in Kombination betrachtet. Um das bestmögliche Setting und die bestmögliche Art von Algorithmus für die Anomalieerkennung zu finden, wurden ein network-based und ein host-based Szenario betrachtet und drei klassische sowie ein graphbasierter Ansatz gewählt.

Die Ergebnisse zeigen, dass ein host-based Szenario besser geeignet ist und klassische Ansätze Ausreißer in den Werten gut erkennen können, sowie dass ein graphbasierter Ansatz vielversprechend für die Erkennung von Anomalien in der Topologie und von verschiedenen Anomaliearten gleichzeitig ist. Dabei ist anzumerken, dass mit kleinen synthetischen Datensätzen (< 20.000 Zeilen) gearbeitet wurde und diese Ergebnisse nur eine erste Einschätzung der Eignung der Verfahren zulassen. Diese soll in der eigentlichen Masterarbeit anhand von realistischen Simulationsdaten verifiziert werden.

4 Konzept

In diesem Kapitel wird das Konzept entwickelt. Dazu werden zunächst die Forschungsfragen analysiert, um daraus Anforderungen an die Anomalieerkennung und die Daten abzuleiten. Zudem wird festgelegt, wie die verwendeten Machine Learning (ML)-Ansätze evaluiert werden. Abschließend wird das Vorgehen zur Umsetzung der Arbeit beschrieben.

4.1 Analyse der Forschungsfragen

> ***Forschungsfrage 1:*** *Wie gut eignen sich klassische ML-Verfahren für die Erkennung verschiedener Arten von Anomalien in der Kommunikation eines Winzent-basierten Multi-Agentensystem (MAS) für die Koordination von dezentralen Energiespeichern?*

Zur Untersuchung dieser Frage müssen repräsentative klassische ML-Verfahren ausgewählt werden, wobei sich an den verwandten Arbeiten aus Unterabschnitt 2.7.1 orientiert werden soll. Da die Ergebnisse aus Kapitel 3 gezeigt haben, dass ein host-basierter Ansatz bei synthetischen Daten die besseren Ergebnisse liefert, soll die Anomalieerkennung aus Sicht eines Knoten in einem simulierten, Winzent-basierten MAS zur Steuerung von Energiespeichern erfolgen. Dieses realistische System soll konstant überwacht werden, was durch das Prinzip des Organic Computing (OC) umgesetzt werden kann. Daher soll die Anomalieerkennung als Observer einer Observer/Controller-Architektur implementiert werden, welcher das MAS als System under Observation/Control (SuOC) beobachtet und dem Controller die Ergebnisse der Anomalieerkennung berichtet. Bei dieser Architektur wird jedem Agenten des MAS ein Observer/Controller-Paar zugeordnet. Durch die Erkennung von Anomalien in den versendeten Nachrichten jedes Agenten wird eine verteilte Überwachung des gesamten Systems ermöglicht. In dieser Arbeit wird nur die Anomalieerkennung umgesetzt.

Da das System in einem realistischen Szenario mit Nachrichten arbeitet, bei denen nicht bekannt ist, ob eine Anomalie vorliegt, werden Verfahren des Semi-supervised und Unsupervised Learning betrachtet. Für eine Bewertung und einen Vergleich der verwendeten Verfahren werden jedoch zunächst gelabelte Daten benötigt. Die Datengrundlage bilden

die vorhandenen Simulationsdaten, die im Umfang der Arbeit in [19] erzeugt wurden. Diese Daten werden in der Umsetzung auf das Szenario angepasst. In den Daten sollen Werteanomalien, Topologieanomalien und Anomalien im Kommunikationsverhalten vorkommen. Für die Topologieanomalien werden entsprechende neue Datensätze erzeugt. Für einen detaillierten Vergleich sind für jeden gewählten Ansatz die Anomaliearten einzeln zu untersuchen. Da es in der Realität vorkommen kann, dass mehrere Arten von Anomalien gleichzeitig auftreten, soll jedes Verfahren zusätzlich mit Daten, die zwei Anomaliearten enthalten, ausgeführt werden.

> ***Forschungsfrage 2:*** *Eignen sich graphbasierte ML Verfahren besser für die Erkennung bestimmter Anomalien in der Kommunikation eines Winzent-basierten MAS für die Koordination von dezentralen Energiespeichern, die von den klassischen Ansätzen weniger gut erkannt werden?*

Diese Forschungsfrage wird nach Abschluss der Untersuchungen zur ersten Frage erforscht. Dafür muss mindestens ein geeigneter graphbasierter Ansatz gewählt werden. Für einen aussagekräftigen Vergleich des graphbasierten Ansatzes mit den klassischen Ansätzen sollen dieselben Daten für die Anomaleerkennung und Metriken zur Evaluation verwendet werden.

4.2 Anforderungen an die Anomalieerkennung

Aus der Analyse der Forschungsfragen ergeben sich folgende Anforderungen an den Observer zur Anomalieerkennung:

- Die Anomalieerkennung soll verteilt aus Sicht eines Agenten stattfinden.
- Es sind verschiedene Verfahren zur Anomalieerkennung zu implementieren, wobei zwischen klassisch und graphbasiert unterschieden wird.
- Anhand festgelegter Metriken sollen die Verfahren bewertet werden, um diese untereinander zu vergleichen. Dabei sollen insbesondere die klassischen Ansätze mit dem graphbasierten Ansatz verglichen werden.
- Es soll für jede Art von Anomalie ein Modell je Verfahren implementiert werden, um die Performanz der Verfahren für verschiedene Anomaliearten vergleichen zu können.
- Auch die Performanz der Verfahren für eine Kombination aus zwei Anomaliearten soll untersucht und in den Vergleich einbezogen werden.
- Der Observer soll in einem Kontext anwendbar sein, in dem nicht bekannt ist, ob eine betrachtete Nachricht eine Anomalie darstellt oder nicht.

- Folglich muss die Anomalieerkennung mit nicht gelabelten Daten arbeiten können und daher Verfahren des Semi-supervised und Unsupervised Learning nutzen.
- Es müssen dem Observer Datensätze mit den versendeten Nachrichten eines Agenten bereitgestellt werden.

4.3 Anforderungen an das Gesamtsystem

Wie in der Zielsetzung in Abschnitt 1.3 beschrieben, sollen verschiedene Ansätze zur verteilten Anomalieerkennung in einem Winzent-basierten MAS verglichen werden. Da der host-basierte Ansatz in Abschnitt 3.5 die besten Ergebnisse liefert, soll jeder einzelne Agent des Systems eine Komponente zur Anomalieerkennung erhalten. Diese Komponente soll zur Überwachung der einzelnen Agenten dienen und wird als Observer im Kontext von OC umgesetzt. Der Agent, der überwacht wird, ist das SuOC.

Ziel des in Abschnitt 2.3 vorgestellten Konzepts von OC ist die kontrollierte Selbstorganisation technischer Systeme durch die Observer/Controller-Architektur. Damit sich das betrachtete MAS selbstständig an Änderungen anpassen kann, ist es notwendig auf Abweichungen von normalem Verhalten reagieren zu können. Mithilfe eines Observers kann ein einzelner Agent überwacht und anomales Verhalten erkannt werden. Das Zielsetting wäre, jedem Agenten des MAS einen Observer zuzuordnen und damit eine Überwachung des gesamten Systems sicherzustellen. Zur Reaktion auf unerwartetes Verhalten könnte für jeden Agenten ein Controller implementiert werden, der mit dem Observer zusammenarbeitet und die Observer/Controller-Architektur mit dem Agenten als SuOC vervollständigt. Da der Fokus dieser Arbeit auf der Anomalieerkennung liegt, wird nur der Observer entwickelt.

Zunächst wird ein Agentensystem benötigt, in dem Agenten nach dem Kommunikationsprotokoll Winzent (siehe Unterabschnitt 2.2.1) Nachrichten austauschen. Die Kommunikation soll in einem sinnvollen Kontext eines cyber-physischen Systems (CPS) stattfinden, in dem die Agenten bestimmte Use Cases umsetzen und dabei definierte Ziele verfolgen. Beispielsweise kann es verschiedene vernetzte Energieanlagen geben, denen je ein Agent zur Steuerung zugeordnet ist. Der Observer soll die Agentenkommunikation überwachen und dort verschiedene Arten von Anomalien erkennen. Dafür müssen dem Observer die Kommunikationsdaten bereitgestellt werden. Die Daten enthalten detaillierte Informationen über die versendeten Nachrichten des dem Observer zugeordneten Agenten.

Für die Untersuchungen in dieser Arbeit müssen die Daten künstlich erzeugte Anomalien enthalten. Die aus [19] vorliegenden Daten enthalten bereits Anomalien, werden aber um eine weitere Art von Anomalie ergänzt. Es sollen verschiedene Arten von Anomalien betrachtet werden, da in [19] gezeigt wurde, dass die Performanz der Anomalieerkennung

abhängig von der Anomalieart ist. Die verschiedenen Ansätze zur Anomalieerkennung sollen für die unterschiedlichen Arten von Anomalien verglichen werden. Insbesondere soll untersucht werden, ob für bestimmte Anomaliearten graphbasierte Ansätze eine bessere Performanz zeigen als klassische Ansätze.

Nach der Definition aus Abschnitt 2.4 sind Anomalien Instanzen, die von einem definierten Normalverhalten abweichen. In einem Setting eines MAS, in dem Agenten miteinander kommunizieren, sind bezüglich der versendeten Nachrichten drei Arten von Anomalien möglich: in den Werten der Nachrichten selbst, im Sendeverhalten und in der Kommunikationstopologie. Für Werteanomalien in den Nachrichten werden bestimmte Felder so manipuliert, dass sie nicht in den physischen Kontext der Anlage des Agenten passen. Eine Anomalie im Kommunikationsverhalten liegt vor, wenn der Agent deutlich mehr Nachrichten verschickt, als er es im Normalfall tut. In einem MAS wird durch die Kommunikationstopologie für jeden Agenten eine Nachbarschaft aus einer Anzahl anderer Agenten festgelegt, mit denen er direkt kommunizieren kann. Es gibt Arbeiten, wie beispielsweise [70], die sich mit dynamischen Topologien beschäftigen. In dem Szenario, das in dieser Arbeit betrachtet wird, ist die Topologie jedoch fest. Eine Topologie-Anomalie liegt vor, wenn ein Agent an einen oder mehrere bestimmte Nachbarn keine Nachrichten mehr versendet oder versucht, eine Nachricht an einen Agenten zu senden, der nicht in seiner Nachbarschaft enthalten ist. Diese Art von Anomalie ist aus Sicht der host-basierten Anomalieerkennung relevant, damit der Observer feststellen kann, dass sich der zugeordnete Agent nicht korrekt verhält. Der Observer kennt die Topologie nicht zwangsläufig, da er nur den eigenen Agenten kennt, sodass beide Fälle zu einer Abweichung vom Normalverhalten beim Versenden der Nachrichten bzgl. der Nachbarn führen, welche als Anomalien erkannt werden sollen. Entsprechende Datensätze müssen neu erzeugt werden.

Zusammengefasst sind folgende Anomalien zu untersuchen:

Anomalien in den Werten der Nachrichten
Bei dieser Art von Anomalie weichen Inhalte der versendeten Nachrichten von den Normalwerten ab.

Anomalien im Kommunikationsverhalten
Hier weicht ein Agent von seinem üblichen Kommunikationsverhalten ab, indem er deutlich mehr Nachrichten verschickt, die nicht in den physischen Kontext passen.

Anomalien in der Kommunikationstopologie In diesem Fall weicht ein Agent von der festgelegten Topologie ab, indem er an einen oder mehrere Nachbarn keine Nachrichten über einen längeren Zeitraum versendet oder versucht Nachrichten an Agenten zu schicken, die nicht seine Nachbarn sind.

Daraus ergeben sich die folgenden Anforderungen an das Gesamtsystem:

- Es soll eine Komponente entwickelt werden, die das Verhalten eines Agenten des MAS überwacht und Anomalien erkennt.
- Die Überwachung des einzelnen Agenten soll durch einen Observer im Kontext von OC stattfinden.
- Der Observer muss verschiedene ML-Verfahren zur Anomalieerkennung einsetzen.
- Der Observer muss Informationen über die versendeten Nachrichten jeweils eines Agenten des MAS erhalten.
- Die Verfahren zur Anomalieerkennung sollen anhand von gelabelten Datensätzen mit bekannten Anomalien evaluiert werden.
- Es sollen Anomalien in den Werten der versendeten Nachrichten untersucht werden.
- Es sollen Anomalien im Kommunikationsverhalten des betrachteten Agenten untersucht werden.
- Es sollen Anomalien in der Kommunikationstopologie untersucht werden.
- Es soll eine Kombination von zwei Anomaliearten untersucht werden.

4.4 Anforderungen an die Daten

Die versendeten Nachrichten der Agenten liegen als Datensätze vor, auf die der Observer zugreifen können muss. Es werden die Datensätze aus [19] verwendet. Um die Daten auf das in dieser Arbeit untersuchte Setting anzupassen, sind folgende Schritte notwendig: die Datensätze sollen nur die versendeten Nachrichten eines (manipulierten) Agenten beinhalten und die Daten sollen als Zeitreihen vorliegen, da einige Ansätze zur Anomalieerkennung Vorhersagen anhand historischer Werte nutzen und somit nur auf Zeitreihen arbeiten. Also müssen entprechende Zeilen eingefügt werden für Zeiträume, in denen keine Nachrichten versendet werden. Dabei werden verschiedene Frequenzen (beispielsweise sekündlich/minütlich) untersucht. Es soll herausgefunden werden, ob sich der Zeitreihencharakter auch für Ansätze, die keine Zeitreihen benötigen, positiv auf die Performanz der Erkennung der Verhaltens- und Topologieanomalien auswirkt. Für das Lernen des Normalverhaltens der Agenten wird ein Datensatz mit Nachrichten ohne Anomalien verwendet. Um die Performanz der Modelle für die verschiedenen Arten von Anomalien vergleichen zu können, wird ein Datensatz pro Anomalieart benötigt. Es liegen Datensätze mit Anomalien in den Werten und mit Anomalien im Kommunikationsverhalten vor, welche auf das Szenario

angepasst werden sollen. Für die Erzeugung von Anomalien in der Kommunikationstopologie sollen vorhandene Normaldatensätze entsprechend verändert werden. Zudem soll ein Datensatz untersucht werden, in dem zwei Arten von Anomalien vorkommen. Für diesen Fall muss ein entsprechender Datensatz durch die Simulation des MAS erzeugt werden, der beide Anomaliearten enthält. Wie auch bei den vorhandenen Datensätzen muss für jede Nachricht bekannt sein, ob es sich um eine Anomalie handelt und der Eintrag muss ein entsprechendes Label erhalten. Außerdem muss untersucht werden, welche Nachrichtenfelder für die Anomalieerkennung weniger oder nicht relevant sind, um die Komplexität der Modelle so gering wie möglich zu halten.

Daraus ergeben sich folgende Anforderungen an die Daten:

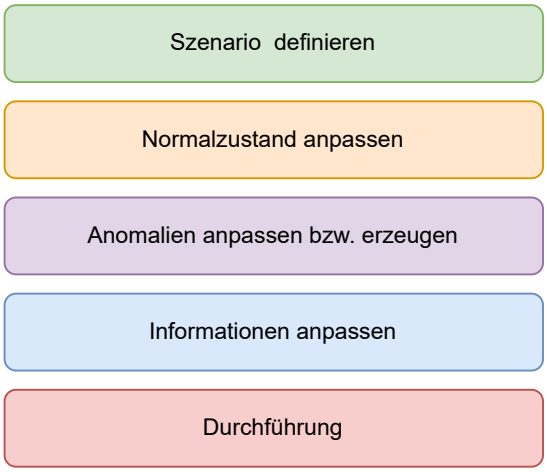
- Die Datensätze müssen dem Observer zur Verfügung stehen.
- Die Datensätze müssen die versendeten Nachrichten eines Agenten beinhalten.
- Die Datensätze müssen die für die Anomalieerkennungsmodelle relevanten Informationen in den versendeten Nachrichten enthalten.
- Jeder Datensatz soll als Zeitreihe vorliegen. Dazu sollen die Zeiträume, in denen keine Nachrichten versendet werden, aufgefüllt werden.
- Es muss ein Datensatz ohne Anomalien vorhanden sein, der ein Normalverhalten abbildet.
- Es muss für jede Art von Anomalie einen Datensatz geben, der diese Anomalie enthält.
- Es muss einen Datensatz geben, der zwei Arten von Anomalien enthält.
- Für alle Datensätze muss bekannt sein, welche Nachrichten anomale Instanzen sind.

4.5 Evaluation

Zur Bewertung der entwickelten Modelle werden die in Unterabschnitt 2.5.7 vorgestellten Metriken verwendet. Diese sind anwendbar, da die Daten gelabelt sind. Die Gesamtperformanz lässt sich anhand der Accuracy einschätzen, welche den Anteil korrekt eingeordneter Instanzen angibt. In Bezug auf die gefundenen Anomalien ist die Precision interessant, da sie den Anteil der Anomalien angibt, der auch tatsächlich anomal ist. Die Sensitivität und die Spezifität beschreiben den Anteil an Anomalien bzw. Normalinstanzen, die korrekt identifiziert wurden. Ebenfalls zur Bewertung hinzugezogen werden der F_1-Score und die Area Under Curve (AUC).

4.6 Vorgehen

Das Vorgehen in der Arbeit lässt sich aus den vorausgegangenen Abschnitten 4.1 - 4.5 ableiten.



Szenario

Das genaue Szenario, in dem Anomalien in einem MAS erkannt werden sollen, muss definiert werden.

Anpassung Normalzustand

Die vorhandenen Daten, die den Normalzustand des Systems erfassen, müssen angepasst werden.

Anpassung und Erzeugung Anomalien

Die vorhandenen Daten, die Anomalien enthalten, müssen angepasst werden. Es müssen bestehende anomale Datensätze ausgewählt werden, die verwendet werden sollen. Weitere zu betrachtende Anomalien müssen in die Daten eingefügt werden. Für die Betrachtung der Kombination aus mehreren Anomaliearten müssen weitere Datensätze erzeugt werden.

Anpassung Informationen

Es muss untersucht werden, ob weniger Informationen für die Anomalieerkennung ausreichen, als in den Daten vorhanden sind.

Durchführung

Nach Anpassung aller Daten können die Analysen zur Anomalieerkennung mit den verschiedenen Ansätzen durchgeführt, ausgewertet und verglichen werden.

4.7 Verwendete Programmiersprache, Bibliotheken und Umgebung

In Abschnitt 2.8 wurde die Relevanz der Programmiersprache Python begründet. Da außerdem das in der Arbeit verwendete Szenario größenteils in Python geschrieben ist, wird diese Sprache für die Entwicklung des Observers verwendet.

Als Entwicklungsumgebung wird Jupyter-Lab gewählt. Dort werden die Daten verarbeitet, die Modelle entwickelt und umfassend ausgewertet.

In der Arbeit werden die in Abschnitt 2.8 vorgestellten Bibliotheken verwendet. Die Anpassung und Analyse der Daten werden durch Pandas und NumPy ermöglicht. Für die klassischen Ansätze wird PyOD verwendet, welches sich wiederum der Backends von Scikit-Learn und Tensorflow bedient. Ein weiterer Ansatz basiert auf Scikit-Learn und PyTorch.

4.8 Nachvollziehbarkeit

Um die Ergebnisse dieser Arbeit nachvollziehbar zu machen, müssen die Daten und entwickelter Code zur Verfügung gestellt werden. Die Erstellung der Daten sowie die Manipulation der Agenten für die Erzeugung von Anomalien sind in [19] nachvollziehbar. Der Code zur Anpassung der Daten, die fertig angepassten Daten sowie die Implementierungen der Modelle werden zur Verfügung gestellt, um die ausgewerteten Ergebnisse nachvollziehbar zu machen. Die trainierten Modelle und die Skripte zur Auswertung werden ebenfalls gespeichert und zur Verfügung gestellt.

5 Umsetzung

Im folgenden Kapitel wird die Umsetzung beschrieben. Zunächst wird das Szenario vorgestellt, aus welchem die verwendeten Daten stammen. Anschließend werden diese Daten genau beschrieben. Dabei werden der Normalzustand und die betrachteten Anomalien erklärt. Danach folgen die Auswahl und Anpassung der vorhandenen Daten und die Erzeugung weiterer Datensätze. Nach der begründeten Auswahl der Verfahren zur Anomalieerkennung wird abschließend die Implementierung der Ansätze vorgestellt.

5.1 Szenario

Das betrachtete Szenario wird aus der Masterarbeit von Emilie Frost ([19]) übernommen. Es stammt aus dem Multi-Purpose Battery Storage Swarm (MIRAGE)-Projekt, in dem ein Multi-Agentensystem (MAS) für die Verwaltung und Steuerung eines Systems aus vernetzten Energiespeichern eingesetzt wird. Jeder Agent des Systems steuert einen Energiespeicher und verfolgt dabei als Ziel die Erfüllung eines bestimmten Anwendungsfalls. Dieses Ziel ist eine Multi-Purpose-Nutzung, wobei der Ausgleich von kurzfristigen Leistungsschwankungen im Vordergrund steht. Ist das primäre Ziel erfüllt, werden die übrigen Freiheitsgrade an Flexibilitätsmärkten vermarktet, um das Speichersystem wirtschaftlich und technisch optimal zu nutzen [20]. Abbildung 5.1 zeigt das Agentensystem, welches aus fünf Agenten besteht, mit einer beispielhaften Kommunikationstopologie (Kommunikationsverbindungen durch Pfeile dargestellt).

In [19] wird ein Setting aus fünf Agenten und Energiespeichern simuliert. Im Rahmen des MIRAGE-Projekts wurde das System bereits in einem Feldtest umgesetzt, für die Generierung der Daten wird jedoch eine Simulation verwendet.

Jeder Agent des Systems besitzt eine Reihe von Modulen, die für unterschiedliche Aufgaben zuständig sind. Die für die Arbeit relevanten Module sind das Forecast-Modul, das Reaktive Modul, das Flexibilitätsmodul und der zu entwickelnde Observer, der als weiteres Modul des Agenten dient. Das Forecast-Modul erstellt mit Prognosen über Energieerzeugung und -verbrauch anhand historischer Daten einen Fahrplan, in welchem Leistungswerte festgehalten werden, die der Speicher für bestimmte Zeitintervalle erfüllen muss. Kommt

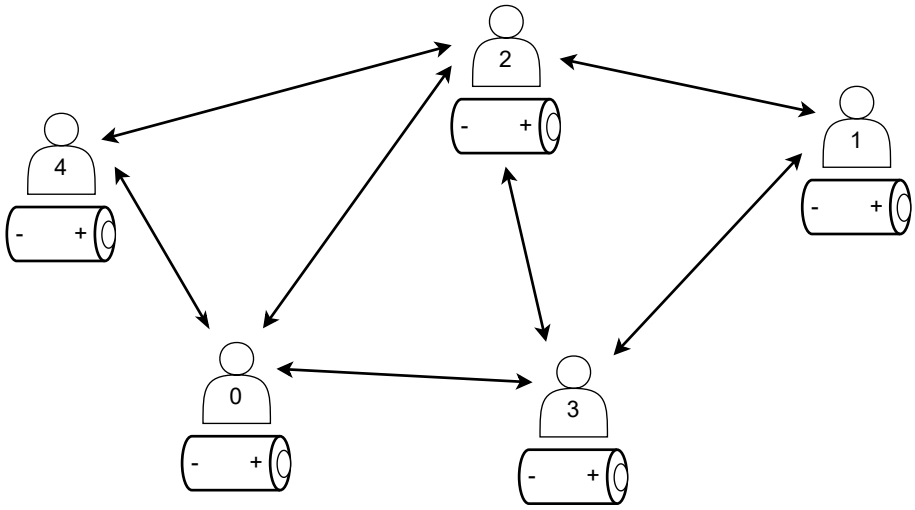


Abbildung 5.1: MAS für die Multi-Purpose-Nutzung vernetzter Energiespeicher.

es durch beispielsweise eine Ungenauigkeit in der Prognose zu einer Abweichung vom Fahrplan, stößt der Agent, der seinen zu erfüllenden Leistungswert nicht einhalten kann, eine Verhandlung an. Er schickt eine Nachricht mit z.B. der fehlenden Energiemenge an die anderen Agenten, welche dann versuchen einen Ausgleich anzubieten. Die Durchführung der Verhandlungen ist Aufgabe des Reaktiven Moduls.

Das Flexibilitätsmodul übernimmt die Berechnung der Flexibilität, also der noch zur Verfügung stehenden Energiemenge nach Erfüllen des primären Ziels, sowie die Überprüfung des Fahrplans und die Planung des Einsatzes des Speichers. Die verfügbare Flexibilität wird unter Berücksichtigung bestehender Verpflichtungen, Lastprognosen und des aktuellen Zustands des Speichers ermittelt. Stellt das Flexibilitätsmodul bei der Planung ein Problem fest, stößt es eine Verhandlung an.

Als Kommunikationsprotokoll für die Verhandlungen wird Winzent verwendet. Es werden nur die für dieses Szenario relevanten Nachrichtentypen genutzt. Diese sind:

- Demand Notification
- Offer Notification
- Acceptance Notification
- Acceptance Acknowledgement Notification
- Withdrawal Notification

Außerdem werden nicht alle Nachrichtenfelder betrachtet, da beispielsweise immer über die Wirkleistung verhandelt wird und das Feld ”Leistungstyp” somit irrelevant ist. Die verwendeten Nachrichtenfelder sind ”Nachrichtentyp”, ”Zeitintervall”, ”Leistungswert”, ”Sender” und ”Empfänger”. Die in der Simulation versendeten Verhandlungsnachrichten jedes Agenten werden in einem Nachrichtenlog gespeichert und dem Observer zur Verfügung gestellt.

5.2 Daten

Die verwendeten Daten stammen aus Simulationsläufen, die im Rahmen der Masterarbeit von Emilie Frost ([19]) durchgeführt wurden. Dort wird das Agentensystem für einige Zeiträume des Jahres 2019 simuliert und die einzelnen Nachrichtenlogs werden gespeichert, zusammengeführt und aufbereitet, sodass Datensätze mit den gesendeten Nachrichten aller Agenten zur Verfügung stehen.

Da in dieser Arbeit ein Observer implementiert wird, der jeweils nur einen Agenten betrachtet, werden für jede Art von Anomalie die Datensätze nach dem manipulierten Agenten gefiltert. So wird eine host-basierte Anomalieerkennung ermöglicht.

5.2.1 Normalzustand

Da einige Machine Learning (ML)-Modelle auf Normaldaten trainiert werden, werden Datensätze ohne Anomalien benötigt. Diese liegen aus [19] vor. Das Flexibilitätsmodul startet Verhandlungen, wenn festgestellt wurde, dass eine Verpflichtung nicht eingehalten werden kann. Dabei wird in der Implementierung des Moduls nicht überprüft, ob dieser Wert innerhalb der Grenzen des Energiespeichers liegt. Es kann also bei normalen Verhandlungen bereits zu Grenzwertverletzungen bei Verhandlungsstarts kommen. Ausreißer in den Leistungswerten, die außerhalb der physikalisch möglichen Grenzen des Batteriespeichers liegen und durch Fehler in anderen Modulen entstehen, werden auf die maximal möglichen Werte gerundet. Somit kommen in den Normaldaten keine Grenzwertverletzungen vor, um das gewünschte Normalverhalten abzubilden.

In den Datensätzen werden Zeiträume von durchschnittlich $2,5$ Wochen simuliert, wobei insgesamt in allen Daten ca. $1,2$ Millionen Nachrichten vorhanden sind.

5.2.2 Anomalien

Für die Anomalieerkennung werden verschiedene Arten von Anomalien untersucht. Dafür werden in der Simulation aus [19] bestimmte Agenten so manipuliert, dass sie ein fehlerhaftes Verhalten beim Versenden der Nachrichten zeigen. Dieses fehlerhafte Verhalten könnte

beispielsweise durch einen Cyber-Angriff hervorgerufen werden. Die betrachteten Anomalien sind Anomalien in den Leistungswerten und im Kommunikationsverhalten. Darüber hinaus werden im Rahmen dieser Masterarbeit auch Anomalien in der Kommunikationstopologie sowie gleichzeitig in der Topologie und in Leistungswerten bestimmter Nachrichten untersucht, wofür entsprechende Simulationen und Datenmanipulationen durchgeführt werden.

Anomalien in den Leistungswerten

Für diese Art von Anomalie werden Ausreißer in den Werten der Nachrichten erzeugt. Der Leistungswert, um den verhandelt wird, ist entscheidend für den weiteren Verlauf der Energieversorgung durch das System. Ist dieser Wert fehlerhaft, kann dies weitere Fehler im System zur Folge haben. Wird beispielsweise ein Agent durch einen Cyber-Angreifer derart manipuliert, dass er einem anderen Agenten eine höhere Energiemenge anbietet, als der Speicher tatsächlich leisten kann, würde der Agent die Verpflichtung nicht einhalten und es müsste neu verhandelt werden. Daher sind Ausreißer in den Leistungswerten für die Anomalieerkennung relevant.

Es werden hier zunächst nur die Leistungswerte manipuliert, die ein Agent als Antwort auf eine Verhandlungsanfrage schickt und dem anfragenden Agenten zusichert. Wenn der Agent dann mit einer höheren Energiemenge plant, als er letztendlich von dem Agenten erhält, und Energieangebote von anderen Agenten ablehnt, da die benötigte Menge von dem manipulierten Agenten zugesichert wurde, kann es dazu kommen, dass der Fahrplan trotz Verhandlung nicht eingehalten werden kann.

In den aus [19] vorliegenden Daten wird dazu ein Agent gewählt, welcher normalerweise geringe Energiemengen aufgrund seiner Speicherkapazität zur Verfügung stellt und dabei recht häufig an der Lösung einer Verhandlung beteiligt ist. Dabei handelt es sich um *Agent_2*. Das anomale Verhalten in Form von nun hohen Energiemengen, welche häufig in Verhandlungslösungen einbezogen werden, bewirkt eine deutliche Abweichung von dem normalen Systemverhalten des gesamten Speicherschwarms.

Weiterhin ist ein Cyber-Angriff-Szenario denkbar, in dem ein Agent Verhandlungen mit besonders hohen Leistungswerten anfragt, um die anderen Agenten dazu zu bringen, viel Energie abzugeben, die der manipulierte Agent eigentlich nicht benötigt. Auch dies kann unerwünschte Auswirkungen auf den weiteren Verlauf des Systems haben. Dieser Fall wird betrachtet für die Untersuchungen mit Daten, die zwei Anomaliearten beinhalten. Dazu wird die Simulation für diese Arbeit erneut gestartet, wobei der manipulierte Agent bei allen Verhandlungen, die er startet, einen zu hohen Leistungswert angibt.

Die Ausprägung der Anomalien sowie die Auswahl des Datensatzes bzw. des manipulier-

ten Agenten werden in Unterabschnitt 5.3.2 bzw. 5.4.2 diskutiert.

Anomalien im Kommunikationsverhalten

Für diese Art von Anomalie werden in [19] Nachrichten betrachtet, mit denen Verhandlungen gestartet werden. Diese sind unabhängig von vorherigen Nachrichten, lösen aber eine Kette an Reaktionen auf die angefragte Energiemenge aus, sodass ein anomales Verhalten zu Beginn einer Verhandlung Einfluss auf das gesamte System nimmt. Es wird ein Agent (*Agent_1*) gewählt, der zusätzlich zu normalen Verhandlungen weitere Verhandlungen anstößt, die nicht notwendig sind. Der gewählte Agent beginnt normalerweise im Vergleich zu den anderen Agenten wenige Verhandlungen, wodurch ein Verhalten mit vielen zufälligen Verhandlungsstarts deutlich anomal ist.

Der Leistungswert in den anomalen Verhandlungsstarts wird auf 50% der Energiespeicherkapazität gesetzt, damit die Grenzwerte nicht verletzt werden und keine Werteanomalien vorkommen. In den vorliegenden Daten gibt es unterschiedliche Ausprägungen des Zeitintervalls, über welches verhandelt wird, sowie der Häufigkeit der angestoßenen Verhandlungen. Die Auswahl der in dieser Arbeit verwendeten Datensätze und die Details zu den Ausprägungen werden in Unterabschnitt 5.3.3 erläutert.

Anomalien in der Kommunikationstopologie

Eine weitere Art in das System einzugreifen und unerwünschtes Verhalten zu erzeugen ist die Manipulation der Kommunikationstopologie. Jeder Agent des Systems hat eine bestimmte Nachbarschaft aus Agenten, an die er direkt Nachrichten senden kann. Wird diese eigenmächtig von einem manipulierten Agenten geändert, hat dies Auswirkungen auf das gesamte System. Es gibt Anwendungsfälle, wie in [70], in denen die Topologie sich während der Laufzeit ändern kann, dies trifft jedoch in dieser Arbeit nicht zu.

Der erste Fall ist das Versenden von Nachrichten an Agenten, die eigentlich nicht in der Nachbarschaft enthalten sind. Dafür wird ein Agent gewählt, der normalerweise die wenigsten Nachbarn hat. Alle versendeten Nachrichten werden dann jedoch auch an alle anderen Nachbarn geschickt. Es gibt einige Settings, in denen diese Nachrichten durch das System abgefangen werden und die Agenten, die nicht Nachbarn sind, nicht erreichen. Dies ist jedoch nicht garantiert und einem potentiellen Angreifer ist dies nicht unbedingt bekannt. Zudem betrachtet der Observer nur den manipulierten Agenten und kennt nicht zwangsläufig die Topologie, sodass er anhand dieses Verhaltens feststellen kann, dass der Agent fehlerhaft ist.

Der zweite Fall wurde in [19] in ähnlicher Weise untersucht. Dort hat ein Agent ab einem bestimmten Zeitpunkt aufgehört auf Nachrichten zu reagieren. Dies ist bei gleichzeitiger

Betrachtung aller Agenten als Anomalie erkennbar. In einem host-basierten Szenario ist dies nur erschwert möglich. Betrachtet man den ausgefallenen Agenten, liegen für den Ausfallzeitraum schlicht keine Daten vor. Betrachtet man einen Agenten, der versucht mit dem ausgefallenen Agenten zu kommunizieren, ist dies nur daran erkennbar, dass der betrachtete Agent keine Antwortnachrichten an den ausgefallenen Agenten schickt, da er von diesem keine Nachrichten erhält, auf die er antworten könnte. Da dieses Szenario zu komplexe Logik enthält, die ein ML-Modell nur schwer abbilden kann, und eher einem anomalen Kommunikationsverhalten zugeordnet werden kann, wird stattdessen eine andere Manipulation der Daten vorgenommen. Ein interessanter Fall ist ein teilweiser Ausfall der Kommunikation. Es wird der Agent mit den meisten Nachbarn gewählt und alle Nachrichten an einen Teil der Nachbarn werden aus den Daten gelöscht. Somit sendet der Agent ab einem bestimmten Zeitpunkt keine Nachrichten mehr an bestimmte Nachbarn, wodurch diese nur noch höchstens indirekt über andere Agenten mit dem Agenten verhandeln können. Ist der manipulierte Agent jedoch beispielsweise der einzige Nachbar eines bestimmten Agenten, zu welchem die Kommunikation eingestellt wird, kann dieser Agent an keinen Verhandlungen mehr teilnehmen. Die veränderte Topologie wirkt sich so auf die gesamte Energieverteilung im System aus.

Die Details zum gewählten Agenten und der Erstellung der Datensätze sind in Unterabschnitt 5.4.1 beschrieben.

Anomalien in den Leistungswerten und der Kommunikationstopologie

Dieser Fall soll untersucht werden um herauszufinden, ob es ein Modell geben kann, welches gleichzeitig auf die Erkennung mehrerer Arten von Anomalien angepasst ist. In der Realität kann es bei beispielsweise einem Cyber-Angriff dazu kommen, dass das Verhalten des Agenten gleichzeitig auf mehrere Arten manipuliert wird. Daher ist dieser Fall relevant.

Dafür wird mit der in [19] verwendeten Simulation ein neuer Datensatz erzeugt, in welchem ein Agent mit wenigen Nachbarn Anomalien in den Leistungswerten von Verhandlungsstarts aufweist. Er fragt also bei anderen Agenten einen höheren Wert an, als er eigentlich benötigt. Wenn diese darauf entsprechend reagieren und diesem Agenten mehr Energie als nötig zur Verfügung stellen, kann dies Auswirkungen auf nachfolgende Verhandlungen haben, bei denen die anderen Agenten entsprechend weniger Flexibilität haben.

Zusätzlich wird dieser Datensatz so manipuliert, dass nur für Verhandlungsstarts der erste Fall der Topologieanomalien auftritt, diese Nachrichten also an weitere Agenten gesendet werden, die nicht Nachbarn des Agenten sind. Die Anomalien werden auf diese Nachrichtentypen beschränkt, da nur in diesen auch Werteanomalien vorkommen und

die Annahme getroffen wird, dass ein potentieller Angreifer beide Manipulationen für die gleichen Arten von Nachrichten vornimmt. Es wird untersucht, ob der eigene Observer des Agenten dieses Fehlverhalten feststellen kann.

Somit liegen in den Daten Topologieanomalien für bestimmte Nachrichtentypen sowie Werteanomalien für diese Nachrichten vor.

Die Details zur Auswahl des Agenten und der Datenerzeugung sind in Unterabschnitt 5.4.2 zu finden.

5.3 Auswahl und Anpassung der Daten

In den folgenden Unterabschnitten wird die Auswahl der Datensätze aus den vorhandenen Daten aus [19] begründet. Diese Daten müssen anschließend für diese Arbeit angepasst werden und es wird anhand der Ergebnisse aus [19] der verwendete Informationsgehalt festgelegt.

5.3.1 Datensätze mit Normaldaten

Es liegen mehrere Datensätze mit Normaldaten aus [19] vor. Eine Übersicht der Datensätze mit dazugehörigem Zeitraum sowie der Anzahl an gesendeten Nachrichten aller Agenten ist in Tabelle 5.1 zu sehen.

Datensatz	Zeitraum	Anzahl Nachrichten
Datensatz_1	29.04. – 12.05.2019	98.116
Datensatz_2	20.04. – 04.05.2019	101.008
Datensatz_3	22.04. – 30.04.2019	60.400
Datensatz_4	14.04. – 24.04.2019	113.757
Datensatz_5	19.04. – 10.05.2019	175.503
Datensatz_6	05.04. – 19.04.2019	83.320
Datensatz_7	05.04. – 14.04.2019	97.656
Datensatz_8	05.04. – 14.04.2019	35.513
Datensatz_9	05.04. – 23.04.2019	141.169

Tabelle 5.1: Übersicht der Normaldaten mit Zeitraum und Gesamtzahl Nachrichten aller Agenten.

Da sich die meisten Datensätze in ihren Zeiträumen überschneiden, ist eine Zusammenführung und anschließende Sortierung nach dem Zeitstempel nicht sinnvoll, da sich dadurch verschiedene Verhandlungen aus verschiedenen Simulationsläufen vermischen würden. Eine zeitliche Sortierung ist jedoch für die Modelle notwendig, weshalb nur ein Datensatz verwendet wird.

Für die Auswahl kommen die Datensätze Datensatz_1, Datensatz_2, Datensatz_4 und Datensatz_7 in Betracht, da die anderen Datensätze einen längeren Zeitraum oder andererseits weniger Nachrichten als die genannten Datensätze beinhalten. Es wurde außerdem untersucht, wie viele Nachrichten jeweils in den Datensätzen von den später manipulierten Agenten vorhanden sind. Für zwei der drei manipulierten Agenten enthält der Datensatz Datensatz_1 die meisten Nachrichten, weshalb dieser für die weitere Vorbereitung ausgewählt wird.

5.3.2 Datensatz mit Anomalien in den Leistungswerten

Es liegen Datensätze mit verschiedenen Ausprägungen der Werteanomalien aus [19] vor, in denen der Leistungswert mit verschiedenen Faktoren multipliziert wird. In Voruntersuchungen wurde festgestellt, dass bei einer Abweichung von 900% vom ursprünglichen Wert bereits in allen Fällen die maximale Kapazität des Energiespeichers überschritten wird. Dies war für ML-Modelle mit einer Performanz von 1 für alle Metriken zu erkennen. Da jedoch auch Ausreißer erkannt werden sollen, die nicht über der Batteriegrenze liegen, wurden in den Untersuchungen Abweichungen von $100\%, 200\%, 300\%, 400\%$ und 500% gewählt. Es konnte festgestellt werden, dass eine Aberichung von unter 400% nicht ausreichend ist für eine zuverlässige Erkennung, da die Ausprägung nicht signifikant genug ist. Die aussagekräftigsten Ergebnisse wurden mit einer Abweichung von 500% erzielt. Daher wird dieser Datensatz verwendet. Diese Ausprägung ist deutlich genug, um als Ausreißer erkannt zu werden und dennoch gering genug, sodass die maximale Kapazität der Batterie nicht immer überschritten wird.

5.3.3 Datensätze mit Anomalien im Kommunikationsverhalten

In [19] werden verschiedene Ausprägungen des Intervalls und der Häufigkeit untersucht. Wenn ein Agent eine Verhandlung startet, wird das Intervall, für welches verhandelt wird, angegeben. Damit wird bestimmt, wie weit die Verhandlung im Voraus gestartet wird. Es werden der minimale (58 Sekunden), durchschnittliche (2012 Sekunden) und maximale (8996 Sekunden) Wert aus den Normaldaten für *Agent_1* gewählt. Es liegt für jede Ausprägung des Intervalls ein Datensatz für eine Häufigkeit der Verhandlungsstarts von $1, 5, 15, 30$ Minuten vor. Das bedeutet, dass *Agent_1* jeweils in dieser Frequenz Nachrichten für den Start einer Verhandlung an seine Nachbarn schickt.

Von den vorliegenden Datensätzen kommen nur zwei für die Untersuchungen in dieser Arbeit in Frage. Dies sind der Datensatz 1m_8996_50p und 15m_2012_50p. Die restlichen Datensätze enthalten entweder zu viele anomale Nachrichten von *Agent_1* ($50 - 100\%$ Anomalien) oder zu wenige (unter 5%). Da die hier zu untersuchenden Daten nur die

Nachrichten von *Agent_1* enthalten und zudem aufgefüllt werden, kommt es bei zu vielen oder zu wenigen anomalen Nachrichten zu Problemen. In Untersuchungen mit einigen Datensätzen, die 100% anomale Nachrichten enthalten, wurde Folgendes festgestellt: Die Modelle erkennen zwar alle Anomalien und ”normale” Nachrichten richtig, aber es werden alle Nachrichten als Anomalie und alle aufgefüllten Datenzeilen als normale Datenpunkte interpretiert. Dies ist nicht Sinn der Untersuchungen. Liegen bei den unaufgefüllten Daten bereits weniger als 5% Anomalien vor, wird der Anteil nach Auffüllen der Daten zu gering (deutlich unter 1%), um ein Modell auf die Erkennung dieser Anomalien anzupassen. Datensatz 1m_8996_50p enthält 25% anomale Nachrichten und Datensatz 15m_2012_50p enthält 12%, sodass nach dem Auffüllen der Anteil an Anomalien bei $8,8\%$ bzw. $2,1\%$ liegt.

In den Ergebnissen aus [19] ist zu sehen, dass die Modelle dort für die Häufigkeiten 15 und 30 Minuten die Anomalien nicht mehr gut erkennen können, da diese Ausprägung nicht auffällig genug ist. Es werden für diese Arbeit aufgrund der vorher genannten Aspekte jedoch beide Datensätze 1m_8996_50p und 15m_2012_50p verwendet.

5.3.4 Auffüllen der Daten

Die Daten aus [19] beinhalten die versendeten Nachrichten. Diese werden in unregelmäßigen zeitlichen Abständen gesendet, da ein Agent beispielsweise eine Demand Notification nach und nach an mehrere Nachbarn schickt, wodurch sich ein Abstand von wenigen Millisekunden ergibt, während er dann möglicherweise für längere Zeiträume von einigen Sekunden oder Minuten keine Nachrichten versendet, da er beispielsweise auf Antworten wartet oder keine Verhandlungen notwendig sind.

Da der verwendete graphbasierte Ansatz jedoch auf zeitabhängigen Vorhersagen basiert und somit Zeitreihen benötigt (siehe Unterabschnitt 5.5.2), müssen die Daten aufgefüllt werden. Da während Verhandlungen Nachrichten im Sekundenbereich verschickt werden, werden die Daten sekündlich aufgefüllt.

Dazu wird für jede volle Sekunde, in der keine Nachricht gesendet wird, eine Zeile erzeugt, die außer des entsprechenden Zeitstempels den Wert -1 in allen anderen Nachrichtenfeldern bis auf das Feld für den Leistungswert aufweist. Der Leistungswert wird auf 0 gesetzt. Der Wert -1 wurde für die anderen Felder anstatt 0 gewählt, da der Wert 0 in Feldern wie beispielsweise Sender eine semantische Bedeutung hat.

Da die verwendeten Datensätze Zeiträume von mehreren Tagen bis Wochen abdecken, werden die aufgefüllten Datensätze sehr groß (einige mit > 1000000 Zeilen). Teilweise ist der Anteil an Anomalien dadurch unter 1%. Für diese Fälle wird untersucht, ob ein Auffüllen mit einer höheren Häufigkeit von 5 oder 30 Sekunden zu einer Verbesserung der Ergebnisse

führt. Im Allgemeinen ist jedoch das Auffüllen mit einer niedrigeren Frequenz problematisch, da viele Verhandlungen innerhalb weniger Sekunden ablaufen und pro Sekunde häufig viele Nachrichten gesendet werden, sodass es zu stärkeren Unregelmäßigkeiten in den Zeitabständen zwischen den Datenpunkten kommt.

5.4 Erzeugung weiterer Datensätze

In dieser Arbeit werden weitere Arten von Anomalien untersucht, wofür entsprechende Daten erzeugt werden müssen. Für die Topologieanomalien wurde eine Manipulation der vorhandenen Normaldaten vorgenommen, während für die Erstellung der Datensätze mit einer Kombination aus zwei Anomaliearten neue Daten erzeugt und anschließend manipuliert werden.

5.4.1 Datensätze mit Anomalien in der Kommunikationstopologie

Da Datensatz_1 ausgewählt wurde, um das Normalverhalten zu repräsentieren, wird dieser Datensatz verwendet, um dort Anomalien einzufügen.

Um festzulegen, welcher Agent für die Manipulation geeignet ist, werden zunächst die Nachbarn jedes Agenten für diesen Datensatz bestimmt. *Agent_1* und *Agent_4* haben jeweils nur zwei Nachbarn (*Agent_2* und *Agent_3* bzw. Agent_0 und *Agent_2*) und kommen daher für den Fall in Frage, dass Nachrichten an Agenten geschickt werden, die nicht in der Nachbarschaft sind. Da für *Agent_1* zum Zeitpunkt dieser Untersuchung bereits aufgefüllte Normaldaten vorliegen, wird dieser Agent gewählt. *Agent_2* hat alle anderen Agenten des Systems (Agent_0, *Agent_1*, *Agent_3*, *Agent_4*) in seiner Nachbarschaft und wird daher für den Fall verwendet, dass an bestimmte Nachbarn keine Nachrichten mehr gesendet werden.

Für den Fall des Versendens an Agenten, die nicht Nachbarn sind, wird der Datensatz also nach *Agent_1* gefiltert und folgendermaßen ergänzt:

- Für Nachrichten, mit denen Verhandlungen gestartet werden und die daher bereits an die Nachbarn *Agent_2* und *Agent_3* gesendet werden, wird jeweils eine Zeile hinzugefügt, in der der Nachbar Agent_0 und *Agent_4* ist. Der Zeitstempel der kopierten Zeilen wird um die Differenz, bzw. die doppelte Differenz, zwischen den beiden Originalnachrichten erhöht, sodass keine Nachrichten exakt gleichzeitig gesendet werden.

- Alle anderen Nachrichtentypen, welche jeweils nur an einen Nachbarn geschickt werden, werden in gleicher Weise kopiert mit Nachbar Agent_0 und *Agent_4*.

Es wurde zunächst der zweite Punkt weggelassen, jedoch ist dann der Anteil anomaler Datenpunkte zu gering.

Für den Fall, dass Kommunikationskanten wegfallen, wird der Datensatz nach *Agent_2* gefiltert. Er soll keine Nachrichten mehr an Agent_0 und *Agent_1* schicken. Dazu werden alle Nachrichtenfelder auf den Wert -1 (wie beim Auffüllen der Daten) gesetzt für alle Zeilen, in denen der Nachbar Agent_0 oder *Agent_1* ist. Dies wurde zunächst nur für Nachrichten durchgeführt, mit denen Verhandlungen gestartet werden, allerdings ist dann der Anteil an Anomalien zu gering. Daher wurden alle Arten von Nachrichten an Agent_0 und *Agent_1* entfernt.

5.4.2 Datensätze mit Anomalien in den Leistungswerten und der Kommunikationstopologie

Für die Untersuchung, ob ein Modell gleichzeitig an verschiedene Arten von Anomalien angepasst werden kann, sind entsprechende Datensätze notwendig. Es sollen Anomalien in den Leistungswerten sowie in der Topologie vorliegen. Dazu wird die Simulation des Agentensystems mit einem manipulierten Agenten durchlaufen und daraus ein neuer Datensatz erstellt.

Es soll eine neue Art von Werteanomalie erzeugt werden, indem diesmal die Nachrichten für Verhandlungsstarts manipuliert werden. Für die Auswahl des zu manipulierenden Agenten werden die Untersuchungen aus [19] einbezogen. *Agent_3* und *Agent_4* starten die meisten Verhandlungen, verwalten jedoch Speicher mit einer sehr hohen Kapazität, wodurch diese häufig Verhandlungen über hohe Leistungswerte starten. Multipliziert man diese Werte zusätzlich mit einem Faktor, sind die Werteanomalien zu auffällig, da sie dann sehr häufig über den Batteriegrenzen liegen. Diese Agenten kommen also nicht in Frage. Daher wird ein Agent gewählt, der einen kleineren Energiespeicher verwaltet und dennoch genügend Verhandlungen startet. Dies ist bei Agent_0 der Fall. Als Topologieanomalie soll der Fall betrachtet werden, dass Nachrichten zu Verhandlungsstarts an weitere Agenten geschickt werden, die nicht in der Nachbarschaft enthalten sind. Dies ist bei Agent_0 möglich, da *Agent_1* kein Nachbar ist und somit entsprechende Zeilen hinzugefügt werden können.

Für die Simulation wird Agent_0 so manipuliert, dass er für einen zufälligen Anteil von 60% seiner gestarteten Verhandlungen einen Leistungswert mit einer Abweichung von 500% übergibt. Diese Ausprägung wird gewählt, damit sowohl Verhandlungsstarts mit normalen als auch mit anomalen Leistungswerten vorliegen und die Werteanomalien die gleiche prozentuale Abweichung von den Normaldaten wie in den anderen Untersuchungen haben.

Nach der Simulation liegen die versendeten Nachrichten von Agent_0 mit allen weiteren Datenfeldern vor. Anschließend wird für jede Nachricht, mit der eine Verhandlung gestartet wird, eine Zeile hinzugefügt, in welcher *Agent_1* der Nachbar ist. Der Zeitstempel dieser Zeile ist der ursprüngliche Zeitstempel, der um die Zeitdifferenz zwischen den Originalnachrichten erhöht wird. Alle anderen Nachrichtenfelder bleiben unverändert.

5.4.3 Auffüllen der Daten

Auch die neu erzeugten Datensätze müssen aufgefüllt werden, um Zeitreihen zu erhalten. Dazu wird wie in Unterabschnitt 5.3.4 beschrieben vorgegangen.

Der Datensatz mit der Kombination aus Werte- und Topologieanomalien enthält nach dem Auffüllen nur ungefähr $0,07\%$ Anomalien. Um diesen Anteil zu erhöhen wird für die Untersuchungen ein weiterer Datensatz verwendet, in dem jede 5. Sekunde, in welcher keine Verhandlung stattfindet, aufgefüllt wird. In diesem Datensatz liegt der Anteil an Anomalien bei ungefähr $0,36\%$. Es wird untersucht, ob sich die Ergebnisse dadurch verbessern.

5.5 Verfahren zur Anomalieerkennung

Für die Untersuchungen sollen Verfahren aus dem supervised und semi-supervised ML eingesetzt werden. Da in der Anwendung des Observers in der Realität keine gelabelten Datensätze vorliegen, können keine supervised Verfahren verwendet werden. Es sollen die gleichen Verfahren wie in den Vorarbeiten (siehe Kapitel 3) genutzt werden, da die gewählten Verfahren auf den synthetischen Daten vielversprechende Ergebnisse liefern und diese mit realistischen Simulationsdaten überprüft werden sollen. Dabei wird die gleiche Unterscheidung in klassische und graphbasierte Ansätze vorgenommen. Die Auswahl der Algorithmen wird im Folgenden erneut kurz dargestellt.

5.5.1 Klassische Verfahren

Zu den klassischen Verfahren zur Anomalieerkennung werden hier der Autoencoder, der Isolation Forest und die One-Class Support Vector Machine (OCSVM) gezählt. Diese unterscheiden sich prinzipiell in ihrer algorithmischen Funktionsweise, werden jedoch alle erfolgreich in verschiedenen Anwendungsbereichen zur Anomalieerkennung eingesetzt (vgl. Unterabschnitt 2.7.1).

5.5.2 Graphbasiertes Verfahren

Als graphbasierter Ansatz wird das Graph Deviation Network (GDN) gewählt. Die Funktionsweise unterscheidet sich fundamental von den anderen Verfahren, da auf Graphstruk-

turen gearbeitet wird. Außerdem wird eine Zeitabhängigkeit eingebracht, da auf Basis historischer Werte Vorhersagen getroffen werden. Dies ist bei keinem der klassischen Ansätze der Fall. Das GDN erzielt in einem Anwendungskontext mit einem Sensornetzwerk (siehe Unterabschnitt 2.7.1) sowie in den Vorarbeiten gute Ergebnisse und wird daher als Vergleichsmodell zu den klassischen Verfahren verwendet.

5.6 Implementierung

Nachfolgend wird ausführlich beschrieben, wie die einzelnen Modelle implementiert werden. Dafür sind zunächst einige Schritte zur Vorbereitung der Daten für alle Verfahren notwendig.

In den originalen Daten sind die normalen Instanzen mit 1 und die Anomalien mit -1 gekennzeichnet. Da für die Auswertung der Modelle das *sklearn.metrics*-Modul von Scikit-Learn genutzt wird, müssen die Labels angepasst werden. Normale Instanzen werden mit 0, anomale Instanzen mit 1 gekennzeichnet.

Der Autoencoder und das GDN werden auf normalen Daten trainiert und erst im Testing kommen Anomalien vor. Daher finden Training und Testing auf unterschiedlichen Datensätzen statt und die Performanz wird anhand der Testdaten bestimmt. Die unsupervised Ansätze Isolation Forest und OCSVM werden beim Training direkt auf Daten ausgeführt, die Anomalien enthalten. Da überprüft werden soll, ob die trainierten Modelle jeweils weitere Datensätze mit der gleichen Performanz klassifizieren können, werden bei diesen Ansätzen die jeweils verwendeten Datensätze in Trainings- und Testdaten aufgeteilt. Eine in der Literatur häufig verwendete Aufteilung für Künstliche Neuronale Netze (KNN) ist 70% der Daten für das Training, 20% der Daten für Validation und 10% der Daten für das Testing zu verwenden [41]. Da der Isolation Forest und die OCSVM keine KNN sind und keine Validierung durchgeführt wird, werden 70% der Daten für das Training und die restlichen 30% für das Testing verwendet. Diese Aufteilung wird für den Autoencoder und das GDN ebenfalls vorgenommen, um für den Vergleich Testergebnisse auf denselben Daten zu erhalten. Diese beiden Verfahren nehmen selbstständig eine Aufteilung der ihnen zur Verfügung stehenden Daten in Trainings- und Validierungsdaten vor.

Weiterhin werden die Daten für die OCSVM und das GDN normalisiert, da dies zu deutlich besseren Ergebnissen führt. Dazu werden die Daten auf den Wertebereich $(-1, 1)$ skaliert. Negative Werte bleiben so für eine bessere Lesbarkeit erhalten.

Da die Topologieanomalien in den entsprechenden anomalen Datensätzen für alle Nachrichten eingefügt wurden, ist es für den Isolation Forest und die OCSVM nicht möglich, gute Ergebnisse zu liefern, wenn nur diese Daten betrachtet werden. Daher werden für diese Modelle für die Topologieanomalien für die Trainingsdaten jeweils 70% des Datensatzes

mit Anomalien und 30% der normalen Daten sowie für die Testdaten die anderen 30% des anomalen Datensatzes und 10% der Normaldaten zusammengeführt, um auch normales Verhalten in den Daten abzubilden.

In Tabelle 5.2 ist eine Übersicht über die aufgefüllten Datensätze zu finden. Sie zeigt für jede Art von Anomalie den Zeitraum, die Größe (Anzahl der Zeilen) sowie den Anteil anomaler Instanzen des jeweils verwendeten Datensatzes.

Datensatz	Zeitraum	Größe	Anteil Anomalien
Werteanomalien	05.04. – 09.04.2019	361.308	4,8%
Anomalien im Kommunikationsverhalten (1m)	05.04. – 10.04.2019	462.459	8,8%
Anomalien im Kommunikationsverhalten (15m)	05.04. – 10.04.2019	493.323	2,1%
Topologieanomalien (Nachbarn kommen hinzu)	29.04. – 12.05.2019	1.108.002	1,3%
Topologieanomalien (Nachbarn fallen weg)	29.04. – 12.05.2019	1.113.858	1,4%
Kombination (jede Sekunde aufgefüllt)	05.04. – 16.04.2019	978.836	0,07%
Kombination (alle 5 Sekunden aufgefüllt)	05.04. – 16.04.2019	198.664	0,36%

Tabelle 5.2: Übersicht über Zeitraum, Größe und Anteil an Anomalien der aufgefüllten Daten.

5.6.1 Autoencoder

Für die Implementierung des Autoencoders wird die Bibliothek PyOD[1] verwendet. Diese wiederum nutzt für die Implementierung Keras von Tensorflow. Bei der Erstellung der Modelle in PyOD gibt es verschiedene Hyperparameter, welche vom Nutzer übergeben werden können. Für verschiedene Belegungen kann sich die Performanz des Modells verändern. Daher werden unterschiedliche Werte für einige Parameter untersucht, um das Modell mit der bestmöglichen Performanz zu erhalten. Nachfolgend sind die Parameter erklärt, welche variiert wurden.

hidden_neurons: Hier wird in einer Liste die Anzahl der Hidden Layer sowie die Anzahl der Neuronen in jedem Layer festgelegt, womit der Aufbau des Autoencoders bestimmt wird. Die Anzahl der Neuronen im Input- und Output-Layer wird automatisch auf die Anzahl der Features gesetzt. Daher hängt der Aufbau stark von der Anzahl verwendeter Features ab.

[1] https://pyod.readthedocs.io/en/latest/

batch_size: Dieser Wert gibt an, wie viele Datenpunkte betrachtet werden, bevor die Gradienten aktualisiert werden. Eine detaillierte Beschreibung ist in Unterabschnitt 2.5.5 zu finden.

dropout_rate: Dieser Parameter gibt die Dropout-Rate an, die in allen Layern benutzt wird. Dropout wird in Unterabschnitt 2.5.5 erklärt.

Weitere Parameter, die nicht mit der Standardbelegung verwendet werden, sind:

contamination: Die contamination wird jeweils auf den Anteil an anomaler Datenpunkte im Testdatensatz gesetzt, da dieser bei den gelabelten Daten vorab berechnet werden kann. Von der contamination ist der *threshold* abhängig, welcher für die Klassifizierung in normal/anomal genutzt wird. Den *threshold* bilden die *Anzahl Samples* $*$ *contamination* anomalsten Samples in den *decision_scores*. Die *decision_scores* sind höher für Ausreißer und hängen somit mit dem Reconstruction Error zusammen.

epochs: Die Anzahl an Trainingsepochen wird auf 10 gesetzt, da sich der Trainingsfehler bei keinem der Modelle nach weiteren Epochen verbessert.

Die Standardbelegungen für die Aktivierungsfunktionen sind *Rectified Linear Unit (ReLU)* für die Hidden Layer und *Sigmoid* für den Output Layer. Diese werden verwendet, da in [71] gezeigt werden konnte, dass diese Kombination andere Modelle, in denen nur eine der beiden Aktivierungsfunktionen genutzt wird, übertrifft. Eine vollständige Liste mit allen Parametern und ihren Standardbelegungen kann der Dokumentation des Autoencoders[2] in PyOD entnommen werden.

In Tabelle 5.3 sind die besten gefundenen Parameterbelegungen der Modelle nach Anomalieart aufgelistet. Die unterschiedlichen betrachteten Werte können im Git den Jupyter Notebooks entnommen werden.

Es wird außerdem untersucht, welche Features je Anomalieart die besten Ergebnisse liefern. Dazu wird sich zunächst an der Feature-Auswahl in [19] orientiert und untersucht, ob weniger Features ausreichend sind. Dafür wurden die Modelle jeweils mit den gleichen Parametern ausgeführt, um die Ergebnisse vergleichen zu können. Die besten gefundenen Belegungen der untersuchten Parameter sind in Tabelle 5.4 aufgeführt. Dabei sind Minimum und Maximum die Grenzwerte des Energiespeichers.

In Abbildung 5.2 ist beispielhaft die Architektur des Modells für Anomalien in der Kommunikationstopologie dargestellt. Die Anzahl der Nodes im Input und Output Layer entspricht der Anzahl der betrachteten Features, die Hidden Layer wurden variiert und zeigen in der Abbildung die Variante, die zu den besten Ergebnissen führt.

[2] https://pyod.readthedocs.io/en/latest/_modules/pyod/models/auto_encoder.html

Parameter	Anomalieart	Parameterbelegung
hidden_neurons	Werteanomalien	[4, 2, 4]
	Anomalien Kommunikationsverhalten (1m und 15m)	[3, 2, 3]
	Topologieanomalien (”dazu” und ”weg”)	[3, 2, 3]
	Kombination	[4, 2, 4]
batch_size	Werteanomalien	64
	Anomalien Kommunikationsverhalten (1m und 15m)	64
	Topologieanomalien (”dazu” und ”weg”)	64
	Kombination	64
dropout_rate	Werteanomalien	0, 1
	Anomalien Kommunikationsverhalten (1m und 15m)	0, 1
	Topologieanomalien (”dazu” und ”weg”)	0, 2
	Kombination	0, 1

Tabelle 5.3: Die besten gefundenen Belegungen der Parameter der Autoencoder-Modelle nach Anomalieart.

Anomalieart	Features
Werteanomalien	Leistungswert, Minimum, Maximum, Sender
Anomalien im Kommunikationsverhalten (1m und 15m)	Zeitstempel, Nachrichtentyp, Sender, Nachbar, Empfänger
Topologieanomalien (”dazu” und ”weg”)	Zeitstempel, Nachrichtentyp, Sender, Nachbar
Kombination	Zeitstempel, Leistungswert, Nachrichtentyp, Sender, Nachbar, Empfänger

Tabelle 5.4: Verwendete Features für die Autoencoder-Modelle je Anomalieart.

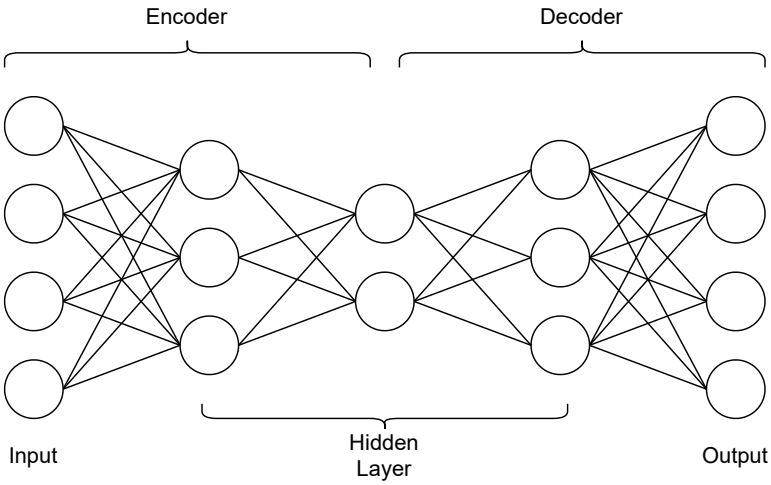


Abbildung 5.2: Architektur des Autoencoders für Daten mit Anomalien in der Kommunikationstopologie.

5.6.2 Isolation Forest

Der Isolation Forest wird ebenfalls in PyOD implementiert. Hier nutzt PyOD für die Implementierung die Bibliothek Scikit-Learn. Die vollständige Auflistung aller Parameter kann der Dokumentation von Pyod zum Isolation Forest[3] entnommen werden. Die nachfolgend aufgelisteten Parameter werden variiert, während die restlichen Parameter mit ihren Standardbelegungen verwendet werden, da diese für die Ergebnisse unerheblich sind (beispielsweise kann die Sichtbarkeit des Trainingsfortschritts oder die Anzahl paralleler Ausführungen eingestellt werden).

n_estimators: Dieser Wert gibt die Anzahl der erstellten Isolationsbäume in einem Ensemble an.

contamination: Mit Angabe dieses Wertes wird der Anteil anomaler Datenpunkte im Datensatz bestimmt, also wird dem Modell mitgegeben, wie viele Anomalien es erkennen soll. Dies lässt sich bei gelabelten Daten vorab bestimmen, aber es können hier teilweise bei höher angesetzten Werten bessere Ergebnisse erzielt werden.

max_features: Dieser Parameter gibt die Anzahl der Features an, die jeweils für die einzelnen Isolationsbäume betrachtet werden.

max_samples: Dieser Wert gibt den Anteil der Datenpunkte an, die jeweils für das Training der einzelnen Isolationsbäume verwendet werden. Es werden nicht alle Daten für alle Isolationsbäume betrachtet, da die Datensätze sehr groß sind und es für gute Ergebnisse ausreichend ist, nur einen Teil der Daten pro Isolationsbaum zu betrachten.

Für die Parameteroptimierung wird die von Scikit-Learn bereitgestellte *GridSearchCV* verwendet, die automatisch für jede Kombination der angegeben Werte für die Parameter ein Modell erstellt, es trainiert und auf Testdaten evaluiert. Zurückgegeben wird dann die Parameterkombination, mit der die besten Ergebnisse erzielt werden. Nachfolgend sind in Tabelle 5.5 die besten gefundenen Parameterbelegungen der Modelle nach Anomalieart dargestellt. Die verschiedenen untersuchten Werte können im Git den Jupyter Notebooks entnommen werden.

Auch hier wird untersucht, welche Features je Anomalieart die besten Ergebnisse liefern. In Tabelle 5.6 sind die Features je Anomalieart aufgeführt, die für die Anomalieerkennung mit dem Isolation Forest zu den besten Ergebnissen führen. Für die Erkennung der Werteanomalien ist die Hinzunahme des Zeitstempels im Gegensatz zu dem Autoencoder hier notwendig, während die Betrachtung des Senders nicht hilfreich ist. Bei den Topologieanomalien, bei denen Nachbarn wegfallen, führt die Hinzunahme des Empfängers zu besseren Ergebnissen. Für die Kombination führen die Betrachtung des Nachrichtentyps, Senders und Empfängers zu keiner Verbesserung.

[3] https://pyod.readthedocs.io/en/latest/_modules/pyod/models/iforest.html

Parameter	Anomalieart	Parameterbelegung
n_estimators	Werteanomalien	150
	Anomalien im Kommunikationsverhalten (1m)	50
	Anomalien im Kommunikationsverhalten (15m)	150
	Topologieanomalien "dazu"	100
	Topologieanomalien "weg"	50
	Kombination	50
contamination	Werteanomalien	0,05
	Anomalien im Kommunikationsverhalten (1m)	0,09
	Anomalien im Kommunikationsverhalten (15m)	0,02
	Topologieanomalien "dazu"	0,008
	Topologieanomalien "weg"	0,01
	Kombination	0,001
max_features	Werteanomalien	1
	Anomalien im Kommunikationsverhalten (1m)	1
	Anomalien im Kommunikationsverhalten (15m)	3
	Topologieanomalien "dazu"	1
	Topologieanomalien "weg"	3
	Kombination	3
max_samples	Werteanomalien	0,8
	Anomalien im Kommunikationsverhalten (1m und 15m)	0,8
	Topologieanomalien ("dazu" und "weg")	0,6
	Kombination	0,6

Tabelle 5.5: Die besten gefundenen Belegungen der Parameter der Isolation-Forest-Modelle nach Anomalieart.

Anomalieart	Features
Werteanomalien	Zeitstempel, Leistungswert, Minimum, Maximum
Anomalien im Kommunikationsverhalten (1m und 15m)	Zeitstempel, Nachrichtentyp, Sender, Nachbar, Empfänger
Topologieanomalien "dazu"	Zeitstempel, Nachrichtentyp, Sender, Nachbar
Topologieanomalien "weg"	Zeitstempel, Nachrichtentyp, Sender, Nachbar, Empfänger
Kombination	Zeitstempel, Leistungswert, Nachbar

Tabelle 5.6: Verwendete Features für die Isolation-Forest-Modelle je Anomalieart.

5.6.3 One-Class Support Vector Machine

Die Implementierung der OCSVM wird ebenfalls in PyOD vorgenommen, wobei auch hier die Implementierung von Scikit-Learn verwendet wird. Aufgrund der Komplexität des Verfahrens und der damit verbundenen langen Laufzeit von bis zu 8 Stunden pro Training wird nur ein Parameter mithilfe der *GridSearchCV* variiert, wobei die untersuchten Werte den Jupyter Notebooks im Git entnommen werden können.

nu: Dieser Parameter stellt eine obere Grenze für den Anteil an Anomalien sowie eine untere Grenze für den Anteil an Support-Vektoren dar. Er gibt an, wie hoch der Anteil der Trainingsdaten höchstens sein darf, der auf der anderen Seite der Entscheidungsgrenze liegt und somit als anomal interpretiert werden kann.

Als **kernel**-Funktion wird standardmäßig die Radial Basis Function (RBF) genutzt, da sie in der Literatur (beispielsweise in [54]) häufig verwendet wird und gute Ergebnisse erzielt. Die

contamination wird auf den Anteil an Anomalien in den Trainings- und Testdaten gesetzt, welcher bei beiden Datensätzen ungefähr gleich ist.

Der **gamma**-Wert gibt den Kernel-Koeffizienten an und wird standardmäßig auf 1/Anzahl der Features belassen.

Die anderen Parameter werden ebenfalls mit ihren Standardbelegungen verwendet. Eine vollständige Liste dazu ist in der Dokumentation der OCSVM[4] von PyOD zu finden. In Tabelle 5.7 sind die besten gefundenen Parameterbelegungen der Modelle nach Anomalieart dargestellt.

Parameter	Anomalieart	Parameterbelegung
nu	Werteanomalien	0,2
	Anomalien Kommunikationsverhalten (1m und 15m)	0,5
	Topologieanomalien (”dazu” und ”weg”)	0,3
	Kombination	0,4

Tabelle 5.7: Die besten gefundenen Belegungen der Parameter der OCSVM-Modelle nach Anomalieart.

Für die OCSVM-Modelle wird ebenfalls untersucht, welche Features je Anomalieart die besten Ergebnisse liefern. Diese sind in Tabelle 5.8 aufgelistet. Dabei sind Minimum und Maximum die Grenzwerte des Energiespeichers. Hier wird im Vergleich zu dem Autoencoder der Zeitstempel für die Erkennung von Werteanomalien benötigt. Bei Anomalien im Kommunikationsverhalten ist die Betrachtung des Empfängers nicht hilfreich, während sie für Anomalien in der Topologie zu besseren Ergebnissen führt. Für die Kombination

[4] https://pyod.readthedocs.io/en/latest/_modules/pyod/models/ocsvm.html

verbessert die Hinzunahme von Sender und Empfänger die Ergebnisse nicht.

Anomalieart	Features
Werteanomalien	Zeitstempel, Leistungswert, Minimum, Maximum, Sender
Anomalien im Kommunikationsverhalten (1m und 15m)	Zeitstempel, Nachrichtentyp, Sender, Nachbar
Topologieanomalien (”dazu” und ”weg”)	Zeitstempel, Nachrichtentyp, Sender, Nachbar, Empfänger
Kombination	Zeitstempel, Nachrichtentyp, Leistungswert, Nachbar

Tabelle 5.8: Verwendete Features für die OCSVM-Modelle je Anomalieart.

5.6.4 Graph Deviation Network

Für Untersuchungen mit dem GDN wird die auf Github öffentlich verfügbare Implementierung in Python verwendet. Es werden kleinere Anpassungen vorgenommen, um die für die Evaluation notwendigen Metriken der Modelle sowie die vorhergesagten Labels der Daten in Tabellenform abzuspeichern.

In einer Shellscript-Datei lassen sich dem Modell einige Parameter übergeben. Die nachfolgend aufgelisteten Parameter wurden variiert, um die bestmöglichen Ergebnisse zu erzielen. Eine vollständige Auflistung der Parameter und deren Standardbelegungen kann dem originalen Repository[5] oder dem zugehörigen Paper [18] entnommen werden.

batch_size: Mit diesem Wert wird die Batchgröße für das Modell festgelegt, welches die Vorhersagen über das erwartete Verhalten trifft. Die verwendeten Werte sind {128,256,512}, da in dieser Größenordung ein guter Kompromiss aus Genauigkeit und Laufzeit (welche aufgrund der großen Datensätze hoch ist) gefunden wird.

slide_win: Dieser Parameter gibt die Größe des verschiebbaren Zeitfensters an, also die Anzahl an vergangenen Werten, die für die Vorhersage des nächsten Datenpunktes einbezogen werden. Dieser Wert muss kleiner als die Batchgröße sein. Die betrachteten Werte sind {50,100,150}.

topk: Dieser Wert legt die Anzahl der betrachteten ähnlichsten Nachbarknoten beim Lernen der Graphstruktur fest, welche dann für die Vorhersagen berücksichtigt werden. Es kann festgehalten werden, dass in dieser Arbeit die besten Ergebnisse erzielt werden, wenn dieser Wert auf die Anzahl der Features gesetzt wird und somit alle Features einbezogen werden.

[5] https://github.com/d-ailin/GDN

In der folgenden Tabelle 5.9 sind je Anomalieart die Parameterbelegungen aufgelistet, die zu dem besten Ergebnis führen.

Parameter	Anomalieart	Parameterbelegung
batch_size	Werteanomalien	512
	Anomalien Kommunikationsverhalten (1m und 15m)	512
	Topologieanomalien (”dazu” und ”weg”)	512
	Kombination	512
slide_win	Werteanomalien	100
	Anomalien Kommunikationsverhalten (1m und 15m)	50
	Topologieanomalien (”dazu” und ”weg”)	100
	Kombination	100
topk	Werteanomalien	5
	Anomalien Kommunikationsverhalten (1m und 15m)	4
	Topologieanomalien (”dazu” und ”weg”)	4
	Kombination	5

Tabelle 5.9: Die besten gefundenen Belegungen der Parameter der GDN-Modelle nach Anomalieart.

Die verwendeten Features werden variiert, um die bestmöglichen Ergebnisse zu erhalten. In Tabelle 5.10 sind die Features je nach Anomalieart aufgeführt, mit denen die besten Ergebnisse erzielt werden. Auch hier ist die Hinzunahme des Zeitstempels im Vergleich zu dem Autoencoder für die Erkennung von Werteanomalien notwendig. Bei Anomalien im Kommunikationsverhalten ist die Betrachtung des Empfängers nicht hilfreich, während sie für Anomalien in der Topologie zu besseren Ergebnissen führt. Zur Erkennung der Topologieanomalien, bei denen Nachbarn wegfallen ist zudem die Hinzunahme des Leistungswertes hilfreich. Für die Kombination führt die Betrachtung des Empfängers zu keinen besseren Ergebnissen.

Anomalieart	Features
Werteanomalien	Zeitstempel, Leistungswert, Minimum, Maximum, Sender
Anomalien im Kommunikationsverhalten (1m und 15m)	Zeitstempel, Nachrichtentyp, Sender, Nachbar
Topologieanomalien "dazu"	Zeitstempel, Nachrichtentyp, Sender, Nachbar, Empfänger
Topologieanomalien "weg"	Zeitstempel, Leistungswert, Nachrichtentyp, Sender, Nachbar, Empfänger
Kombination	Zeitstempel, Leistungswert, Nachrichtentyp, Sender, Nachbar

Tabelle 5.10: Verwendete Features für die GDN-Modelle je Anomalieart.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse präsentiert. Zunächst wird auf den Informationsgehalt und die Auffüllfrequenz der Daten eingegangen. Anschließend wird für jedes Modell die Performanz für die verschiedenen Arten von Anomalien aufgezeigt.

6.1 Informationsgehalt

Der Informationsgehalt, welcher dem Observer zur Verfügung gestellt wird, wird in [19] variiert. Die grundlegende Information über Agent und Zeitstempel ist dort in dem network-basierten Szenario ausreichend für die Erkennung von Anomalien im Kommunikationsverhalten. In dieser Arbeit werden allerdings bestimmte Informationen aus dem Inhalt der Nachrichten über beispielsweise den Nachrichtentyp, Sender, Empfänger oder Nachbar hinzugezogen, da nur ein Agent betrachtet wird. Die gleichen Informationen sind für die Erkennung von Topologieanomalien notwendig. Für die Erkennung von Werteanomalien ist in [19] die Einbeziehung der Grenzwerte des Energiespeichers hilfreich. In den Daten liegen der Minimal- und Maximalwert jedes Energiespeichers vor. Daher werden in dieser Arbeit für die Erkennung von Werteanomalien diese Informationen zusätzlich zum Leistungwert genutzt.

Zudem wird in [19] untersucht, ob Informationen über Intervalle, in denen Verpflichtungen vorliegen, zu verbesserten Ergebnissen führen. Da dies nicht der Fall ist, werden diese Informationen hier nicht betrachtet.

Die Untersuchungen werden also zunächst mit den oben genannten Informationen durchgeführt. Für jedes Modell wird geprüft, ob weniger Informationen zu gleichen oder sogar besseren Ergebnissen führen. Der angepasste Informationsgehalt wird in den später folgenden Abschnitten für jedes Modell in Tabellenform festgehalten.

6.2 Auffüllfrequenz der Daten

Um zu überprüfen, ob sich die Performanz der klassischen Modelle ändert, wenn die Originaldaten nur nach dem manipulierten Agenten gefiltert und anschließend nicht aufgefüllt

werden, wurde dies für alle klassischen Verfahren exemplarisch an dem Datensatz mit Werteanomalien und für den Isolation Forest mit dem Datensatz mit der Kombination aus Werte- und Topologieanomalien untersucht.

Der Datensatz mit Werteanomalien enthält 62% Anomalien, wenn nur der manipulierte Agent betrachtet wird. Dies macht gute Ergebnisse bereits im Vorfeld unwahrscheinlich, da Anomalien selten vorkommen sollten. Der Autoencoder erreicht hier eine Precision von $0,72$, allerdings nur eine Accuracy von $0,54$ und eine Area Under Curve (AUC) von $0,43$. Der Isolation Forest kann im Training nur eine AUC von $0,07$ erreichen, im Testing wird der gesamte Datensatz als anomal klassifiziert. Auch die One-Class Support Vector Machine (OCSVM) erzielt im Training eine schlechte AUC von $0,29$ und klassifiziert den gesamten Testdatensatz als anomal. Anhand der schlechten AUC-Werte wird deutlich, dass die Modelle nicht an die Erkennung der Anomalien in diesen Daten angepasst werden können. Alle Kennzahlen zur Performanz auf den Testdaten sind in Tabelle 6.1 aufgelistet und die zugehörigen Confusion Matrizen sind im Git den Jupyternotebooks mit der Bezeichnung "_original" am Ende des Dateinamen zu entnehmen.

Modell	Accuracy	Precision	F1-Score	Recall	AUC
Autoencoder Werteanomalien	$0,54$	$0,60$	$0,66$	$0,73$	$0,43$
Isolation Forest Werteanomalien	$0,61$	$0,61$	$0,76$	$1,0$	$0,03$
OCSVM Topologieanomalien	$0,61$	$0,61$	$0,76$	$1,0$	$0,30$
Isolation Forest Kombination	$0,56$	$0,32$	$0,45$	$0,74$	$0,72$

Tabelle 6.1: Die Ergebnisse der Modelle mit den unaufgefüllten Daten. Alle klassischen Modelle wurden mit Werteanomalien untersucht, die Kombination aus Werte- und Topologieanomalien wurde nur von dem Isolation Forest betrachtet.

Auch in dem Datensatz mit der Kombination aus Werte- und Topologieanomalien liegt mit 21% ein sehr großer Anteil an Anomalien vor. Der Isolation Forest erzielt im Training eine Accuracy von $0,73$ und eine AUC von $0,67$. Im Testing wird mehr als die Hälfte des Datensets als anomal klassifiziert, wodurch nur eine Precision von $0,57$ erreicht wird. Die weiteren Kennzahlen sind der letzten Zeile von Tabelle 6.1 zu entnehmen. Die Confusion Matrizen befinden sich im Git in dem Jupyternotebook "IForest_Kombi". Aufgrund dieser Ergebnisse und des hohen Anomalieanteils werden die unaufgefüllten Daten nicht für weitere Untersuchungen verwendet.

In Unterabschnitt 5.3.4 wurde außerdem das Auffüllen mit geringerer Häufigkeit diskutiert. Während das Auffüllen jeder fünften vollen Sekunde für einige Verfahren zu besseren oder zumindest nicht schlechteren Ergebnissen führt, werden diese in den nachfolgenden Abschnitten in die Präsentation der Ergebnisse einbezogen. Es wird zudem ein Datensatz alle 30 Sekunden aufgefüllt. Dabei handelt es sich um den Datensatz 15m_2012_50p

mit Anomalien im Kommunikationsverhalten, bei denen alle 15 Minuten eine anomale Verhandlungsanfrage geschickt wird. Dieser Datensatz bietet sich für das Auffüllen mit geringerer Häufigkeit an, da die Anomalien in großen zeitlichen Abständen auftreten und somit schwieriger zu erkennen sind als beispielsweise minütlich gesendete Verhandlungsstarts (siehe nachfolgende Abschnitte). Der Datensatz wird für alle 30 Sekunden aufgefüllt und exemplarisch mit dem Isolation Forest untersucht. Es wird im Testing eine Accuracy von $0,17$, eine Precision von $0,04$, ein Recall von $0,23$, ein F1-Score von $0,07$ und eine AUC von $0,13$ erreicht. Mehr als zwei Drittel der Datenpunkte werden als Anomalie klassifiziert. Die Confusion Matrizen für Training und Testing befinden sich im Git, mit dem Zusatz "_30s" im Dateinamen. Aufgrund dieser schlechten Ergebnisse werden keine weiteren Untersuchungen mit Daten, die alle 30 Sekunden aufgefüllt werden, durchgeführt.

6.3 Autoencoder

In den nachfolgenden Unterabschnitten werden die Ergebnisse der Autoencoder-Modelle auf den Testdaten je Anomalieart gezeigt. Da der Autoencoder mit normalen Daten trainiert wird, wird er nur auf den Testdaten, welche Anomalien enthalten, evaluiert. Zu allen Modellen befinden sich die Confusion Matrizen im Git.

6.3.1 Anomalien in den Leistungswerten

Der Autoencoder erkennt die Werteanomalien mit einer Accuracy von $0,97$ und einer Precision von $0,68$. Der F1-Score liegt bei $0,77$ und der Recall bei $0,89$. Die dennoch hohe AUC von $0,99$ sowie die hohe Accuracy kommen durch den relativ geringen Anteil von $4,8\%$ Anomalien zustande. Diese Werte sind in Abbildung 6.1 dargestellt.

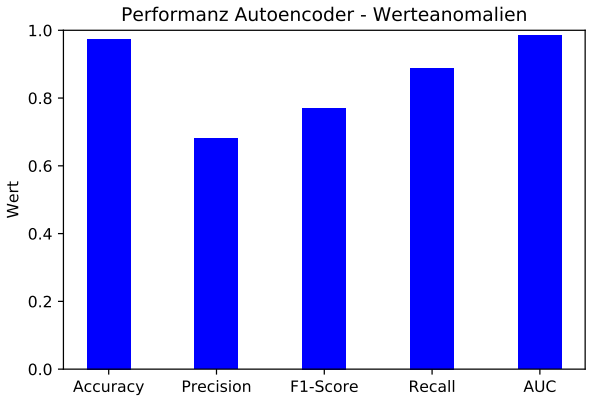


Abbildung 6.1: Ergebnisse des Autoencoders für Daten mit Anomalien in den Leistungswerten

6.3.2 Anomalien im Kommunikationsverhalten

Die Ergebnisse des Autoencoders auf dem Datensatz 1m_8996_50p mit Anomalien im Kommunikationsverhalten durch minütliches Senden nicht notwendiger Verhandlungsstarts sind in Abbildung 6.2 dargestellt. Die Accuracy liegt bei $0,87$, die Precision bei $0,36$, der F1-Score bei $0,45$ und der Recall bei $0,58$. Es wird eine AUC von $0,82$ erreicht.

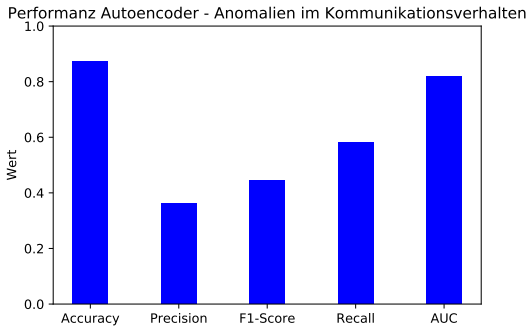


Abbildung 6.2: Ergebnisse des Autoencoders für Datensatz 1m_8996_50p

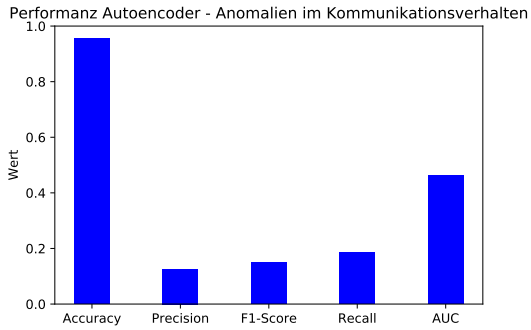


Abbildung 6.3: Ergebnisse des Autoencoders für Datensatz 15m_2012_50p

Für den Datensatz, in dem die Häufigkeit der anomalen Verhandlungsstarts bei 15 Minuten liegt, fallen die Ergebnisse deutlich schlechter aus. Die Performanz ist in Abbildung 6.3 dargestellt. Es werden nur eine Precision von $0,13$, ein F1-Score von $0,15$ und ein Recall von $0,19$ erreicht. Die Accuracy und die AUC liegen bei $0,95$ bzw. $0,46$. In dem Datensatz liegt der Anteil anomaler Instanzen nur bei 2%.

6.3.3 Anomalien in der Kommunikationstopologie

Wie in Abbildung 6.4 zu sehen ist, erkennt der Autoencoder die Anomalien in der Kommunikationstopologie, für den Fall, dass Nachbarn dazu kommen, sehr gut mit einem Recall von $1,0$. Allerdings liegt die Precision nur bei $0,57$. Es ergibt sich ein F1-Score von $0,0.72$. Aufgrund des sehr geringen Anteils an Anomalien liegen die Accuracy und die AUC bei $0,99$.

Für den Fall, dass Nachbarn wegfallen, können die Anomalien von dem Autoencoder nicht erkannt werden. Die Precision ist $0,02$, der Recall und der F1-Score $0,01$. Die AUC liegt bei $0,43$ und die Accuracy ist aufgrund des geringen anomalen Anteils dennoch $0,98$.

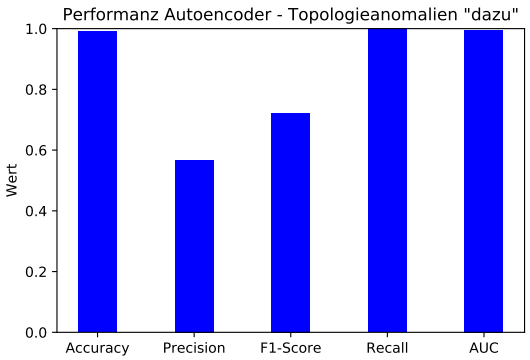


Abbildung 6.4: Ergebnisse des Autoencoders für Topologieanomalien (Nachbarn kommen dazu)

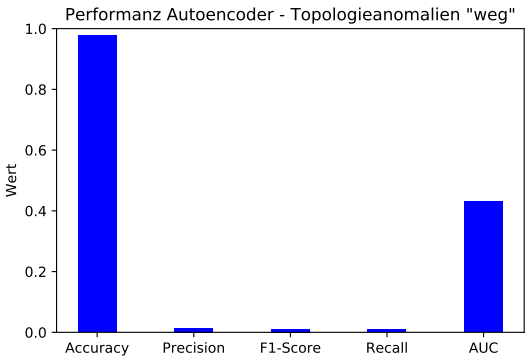


Abbildung 6.5: Ergebnisse des Autoencoders für Topologieanomalien (Nachbarn fallen weg)

6.3.4 Anomalien in den Leistungswerten und der Kommunikationstopologie

Der Autoencoder erzielt etwas bessere Ergebnisse für die Daten mit einer Kombination aus Werte- und Topologieanomalien, wenn diese nur alle 5 Sekunden aufgefüllt werden. Insgesamt können die Anomalien nicht sehr gut erkannt werden. Die Accuracy und die AUC liegen jeweils bei $0,99$. Die Precision ist $0,22$, der F1-Score $0,29$ und der Recall $0,45$. Diese sind in Abbildung 6.6 abgebildet.

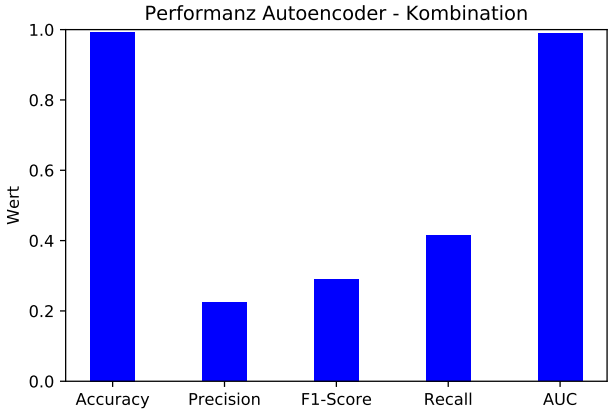


Abbildung 6.6: Ergebnisse des Autoencoders für Daten mit Anomalien in den Leistungswerten und der Topologie (alle 5 Sekunden aufgefüllt)

Die Ergebnisse für die sekündlich aufgefüllten Daten befinden sich im Anhang unter Tabelle A.1.

6.4 Isolation Forest

Nachfolgend werden die Ergebnisse der Isolation-Forest-Modelle für die verschiedenen Arten von Anomalien präsentiert. Die Modelle werden auf den Trainings- und Testdaten evaluiert. Die Confusion Matrizen zu allen Modellen sind im Git zu finden.

6.4.1 Anomalien in den Leistungswerten

Der Isolation Forest erreicht für die Trainingsdaten mit Werteanomalien eine Accuracy von $0,96$, eine Precision von $0,55$, einen F1-Score von $0,59$ und einen Recall von $0,64$. Die AUC liegt bei $0,98$. Auf dem Testdatensatz verbessert sich die Ergebnisse auf eine Accuracy von $0,97$, eine Precision von $0,64$, einen F1-Score von $0,78$ und einen Recall von $1,0$. Die AUC ist $0,97$. Es können also in den Testdaten alle Anomalien als solche erkannt werden, allerdings gibt es einige False Positives. Die genannten Werte sind in Abbildung 6.7 dargestellt.

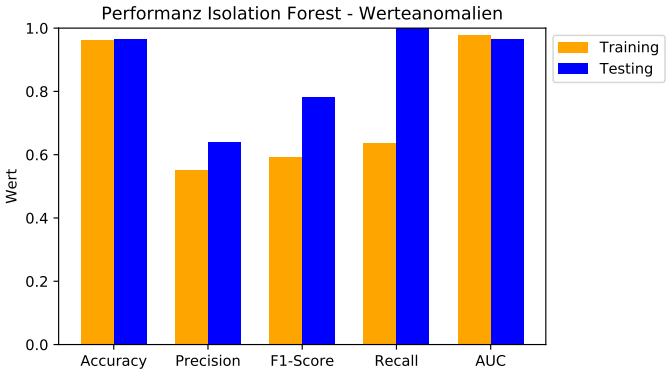


Abbildung 6.7: Ergebnisse des Isolation Forest für Daten mit Anomalien in den Leistungswerten

6.4.2 Anomalien im Kommunikationsverhalten

Der Isolation Forest wird zunächst für den Datensatz evaluiert, welcher anomale Verhandlungsstarts für jede Minute enthält. Die Anomalien werden gut erkannt. Die Accuracy liegt bei $0,97$, die Precision bei $0,79$, der F1-Score bei $0,83$ und der Recall bei $0,87$. Es wird eine AUC von $0,98$ erreicht. Auf den Testdaten verbessern sich der F1-Score auf $0,86$ und der Recall auf $1,0$. Die Accuracy und die AUC bleiben nahezu gleich. Die Precision liegt für die Testdaten bei $0,76$. Die Werte sind in Abbildung 6.8 zu sehen.

Für den Datensatz, in dem alle 15 Minuten eine anomale Verhandlung gestartet wird, werden die Anomalien in den Trainingsdaten nicht gut erkannt. Die Accuracy ist $0,97$, die

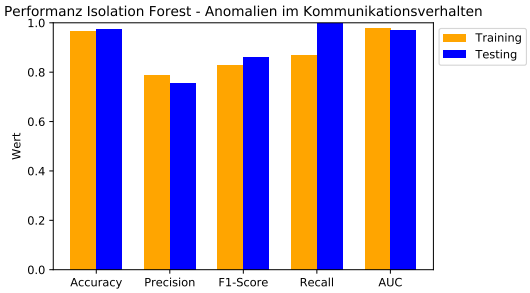


Abbildung 6.8: Ergebnisse des Isolation Forest für Datensatz 1m_8996_50p

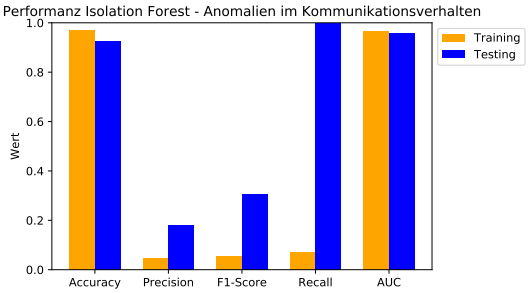


Abbildung 6.9: Ergebnisse des Isolation Forest für Datensatz 15m_2012_50p

Precision $0,05$, der F1-Score $0,06$ und der Recall $0,07$. Die AUC liegt bei $0,97$. Für die Testdaten werden alle Anomalien als solche erkannt (Recall von $1,0$), allerdings ist die Zahl der False Positives sehr hoch. Die Accuracy liegt hier bei $0,92$, die Precision bei $0,18$, der F1-Score bei $0,31$ und die AUC bei $0,96$.

6.4.3 Anomalien in der Kommunikationstopologie

Wie in Abbildung 6.10 zu sehen ist, erkennt der Isolation Forest Anomalien in der Topologie, bei denen Nachbarn dazu kommen, mäßig zuverlässig. Im Training werden eine Precision und ein F1-Score von $0,52$ und ein Recall von $0,53$ erreicht. Für die Testdaten verbessern sich diese auf eine Precision von $0,59$, einen Recall von $0,62$ und einen F1-Score von $0,60$. Die Accuracy und die AUC ist in beiden Fällen aufgrund des niedrigen Anomalie-Anteils bei $0,99$.

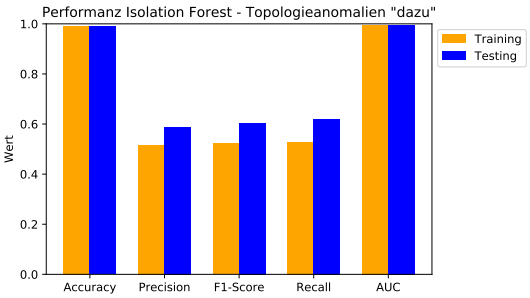


Abbildung 6.10: Ergebnisse des Isolation Forest für Topologieanomalien (Nachbarn kommen dazu)

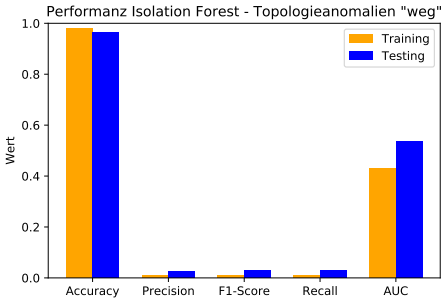


Abbildung 6.11: Ergebnisse des Isolation Forest für Topologieanomalien (Nachbarn fallen weg)

Der Fall, bei dem Nachbarn wegfallen, kann von dem Isolation Forest nicht erkannt

werden. Für die Trainingsdaten liegen die Precision, der Recall und der F1-Score jeweils nur bei $0,01$. Die AUC ist $0,43$. Für die Testdaten verbessert sich die Performanz minimal auf $0,03$ für Precision, Recall und F1-Score und auf $0,54$ für die AUC. Die Accuracy liegt aufgrund des geringen Anteils an Anomalien bei $0,98$ im Training und bei $0,97$ beim Testing. Diese Ergebnisse sind in Abbildung 6.11 dargestellt.

6.4.4 Anomalien in den Leistungswerten und der Kommunikationstopologie

Die Kombination aus Anomalien in den Leistungswerten und der Topologie für Verhandlungsanfragen kann von dem Isolation Forest für die Trainingsdaten nicht gut erkannt werden. Die Ergebnisse mit sekündlich aufgefüllten Daten sind in Abbildung 6.12 zu sehen. Die Ergebnisse mit Daten, die alle 5 Sekunden aufgefüllt sind, erreichen etwas geringere Zahlen und befinden sich im Anhang unter Tabelle A.1. Die Precision liegt im Training bei $0,22$, der Recall bei $0,35$ und der F1-Score bei $0,27$. In den Testdaten können allerdings alle Anomalien als solche erkannt werden (Recall von $1,0$), jedoch wird nur eine Precision von $0,25$ erreicht, sodass sich ein F1-Score von $0,40$ ergibt. Durch den sehr geringen Anteil an Anomalien liegen die Accuracy sowie die AUC im Training und im Testing bei $0,99$.

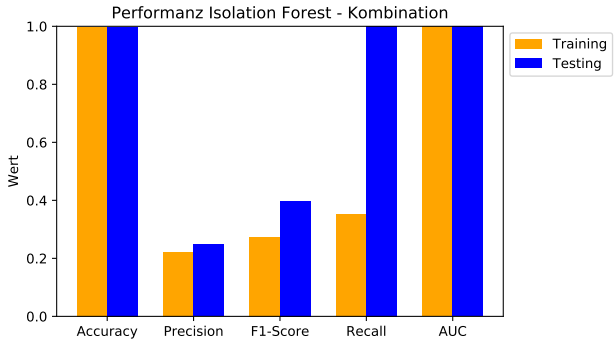


Abbildung 6.12: Ergebnisse des Isolation Forest für Daten mit Anomalien in den Werten und der Topologie

6.5 One-Class Support Vector Machine

Die nachfolgenden Unterabschnitte zeigen die Ergebnisse der OCSVM für die verschiedenen Arten von Anomalien. Da dieses Verfahren wie der Isolation Forest unsupervised ist, werden die Modelle sowohl auf Trainings- als auch Testdaten ausgewertet. Die Confusion Matrizen zu jedem Modell befinden sich im Git.

6.5.1 Anomalien in den Leistungswerten

Anomalien in den Leistungswerten können von der OCSVM recht gut erkannt werden. Für die Trainingsdaten wird eine Precision von $0,69$, ein Recall von $0,95$ und ein F1-Score von $0,80$ erreicht. In den Testdaten können alle Anomalien erkannt werden (Recall von $1,0$), allerdings sind die Precision mit $0,62$ und der F1-Score mit $0,77$ hier etwas geringer als im Training. Die Accuracy beträgt im Training $0,98$ und im Testing $0,96$. Die AUC ist im Training $0,99$ und im Testing $0,98$. Diese Ergebnisse sind in Abbildung 6.13 dargestellt.

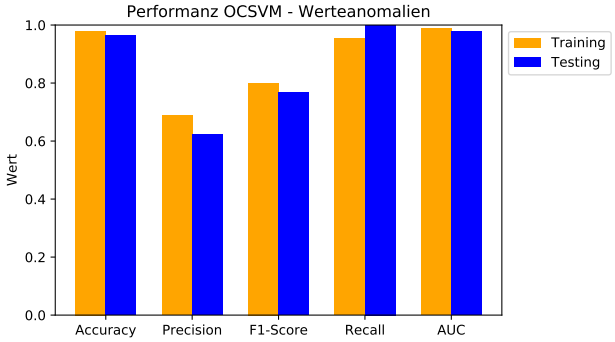


Abbildung 6.13: Ergebnisse der OCSVM für Daten mit Anomalien in den Leistungswerten

6.5.2 Anomalien im Kommunikationsverhalten

Die Ergebnisse der OCSVM bezüglich der Anomalien im Kommunikationsverhalten sind in Abbildung 6.14 für die Häufigkeit von einer Minute und in Abbildung 6.15 für die Häufigkeit von 15 Minuten abgebildet.

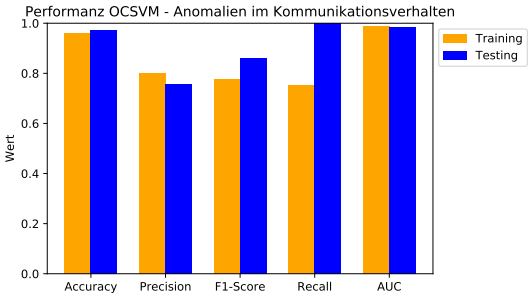


Abbildung 6.14: Ergebnisse der OCSVM für Datensatz 1m_8996_50p

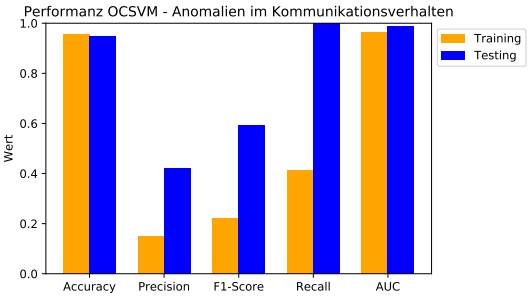


Abbildung 6.15: Ergebnisse der OCSVM für Datensatz 15m_2012_50p

Die minütlich gesendeten Verhandlungsstarts können von der OCSVM gut als Anomalien erkannt werden. Während im Training die Precision mit $0,80$ etwas höher liegt als der

Recall mit 0, 75, kann für die Testdaten ein Recall von 1, 0 erreicht werden, wobei dann die Precision etwas geringer bei 0, 76 liegt. Der F1-Score beträgt im Training 0, 78 und im Testing 0, 76. Die Accuracy ist 0, 96 im Training und 0, 97 im Testing. Die AUC beträgt sowohl im Training als auch im Testing 0, 99.

Die anomalen Verhandlungsstarts, die mit einer Häufigkeit von 15 Minuten auftreten, können von der OCSVM weniger gut erkannt werden. Für die Trainingsdaten liegt die Precision bei 0, 15, der Recall bei 0, 41 und der F1-Score bei 0, 22. Im Testing kann zwar ein Recall von 1, 0 erreicht werden, allerdings liegt die Precision bei 0, 42, es gibt also viele False Positives. Daher liegt der F1-Score im Testing bei 0, 59. Die Accuracy beträgt im Training 0, 96 und im Testing 0, 95. Die AUC ist im Training 0, 99 und im Testing 0, 99.

6.5.3 Anomalien in der Kommunikationstopologie

In Abbildung 6.16 werden die Ergebnisse der OCSVM für Anomalien in der Topologie, bei denen Nachbarn dazu kommen, gezeigt. Die Anomalien können mäßig gut erkannt werden. Im Training liegt die Precision bei 0, 44, der Recall bei 0, 57 und der F1-Score bei 0, 50. Für die Testdaten werden eine Precision von 0, 56, ein Recall von 0, 68 und ein F1-Score von 0, 62 erreicht. Die Accuracy und die AUC liegen jeweils bei 0, 99 für Training und Testing.

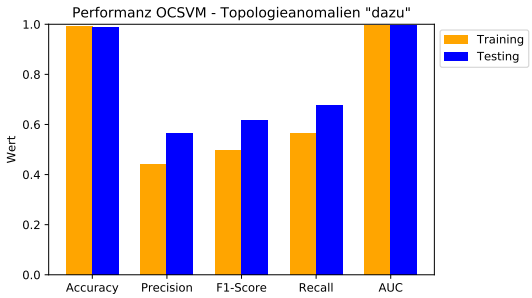


Abbildung 6.16: Ergebnisse der OCSVM für Topologieanomalien (Nachbarn kommen dazu)

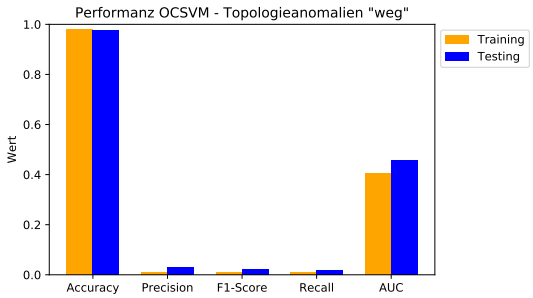


Abbildung 6.17: Ergebnisse der OCSVM für Topologieanomalien (Nachbarn fallen weg)

Abbildung 6.17 zeigt die Ergebnisse der OCSVM für Anomalien in der Topologie, bei denen Nachbarn wegfallen. Die Anomalien werden sehr schlecht erkannt. Für die Trainingsdaten liegen die Precision, der Recall und der F1-Score bei 0, 01. Die Accuracy ist aufgrund des geringen Anteils an Anomalien bei 0, 98, die AUC liegt jedoch nur bei 0, 40. Im Testing sind die Werte für Precision bei 0, 03, für Recall und F1-Score jeweils bei 0, 02, die Accuracy liegt bei 0, 97 und die AUC bei 0, 46.

6.5.4 Anomalien in den Leistungswerten und der Kommunikationstopologie

Die Kombination von Anomalien in den Leistungswerten und in der Topologie kann von der OCSVM nicht gut erkannt werden. Hier werden in Abbildung 6.18 die Ergebnisse für die Daten gezeigt, die alle fünf Sekunden aufgefüllt wurden, da diese minimal besser ausfallen. Die Ergebnisse für die sekündlich aufgefüllten Daten befinden sich im Anhang unter Tabelle A.1. Für die Trainingsdaten beträgt die Precision 0,04, der Recall 0,05 und der F1-Score 0,04. Im Testing wird eine Precision von 0,06, ein Recall von 0,17 und ein F1-Score von 0,09 erreicht. Da in den Daten nur 0,7% Anomalien vorkommen, liegt die Accuracy für Training und Testing bei 0,99. Die AUC ist im Training 0,99 und im Testing 0,5.

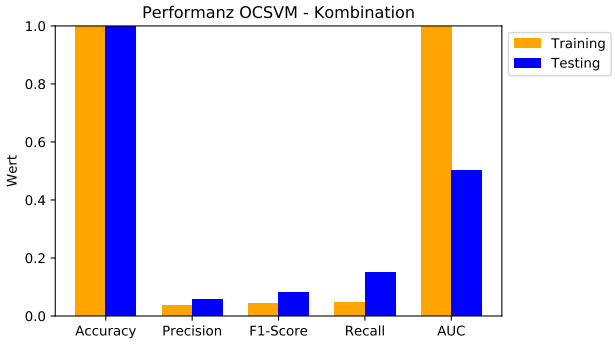


Abbildung 6.18: Ergebnisse der OCSVM für Daten mit Anomalien in den Werten und der Topologie

6.6 Graph Deviation Network

Nachfolgend werden die Ergebnisse des Graph Deviation Network (GDN) präsentiert. Die Modelle werden mit normalen Daten trainiert und auf Daten, die Anomalien enthalten, evaluiert. Die zugehörigen Confusion Matrizen zu jedem Modell befinden sich im Git.

6.6.1 Anomalien in den Leistungswerten

Die Anomalien in den Leistungswerten können von dem GDN recht gut erkannt werden. Der Recall liegt bei 0,99, allerdings ist die Precision nur 0,51. Der F1-Score liegt bei 0,67 und es werden eine Accuracy von 0,95 sowie eine AUC von 0,98 erreicht. Die Ergebnisse sind in Abbildung 6.19 dargestellt.

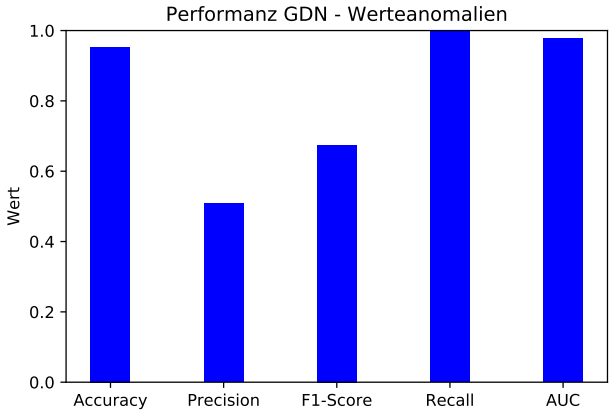


Abbildung 6.19: Ergebnisse des GDN für Daten mit Anomalien in den Leistungswerten

6.6.2 Anomalien im Kommunikationsverhalten

Die Anomalien im Kommunikationsverhalten können mäßig gut erkannt werden (siehe Abbildung 6.20 und 6.21). Für die Häufigkeit von einer Minute der anomalen Verhandlungsstarts werden eine Precision von $0,57$, ein Recall von $0,64$ und ein F1-Score von $0,60$ erreicht. Die Accuracy liegt bei $0,93$ und die AUC bei $0,95$.

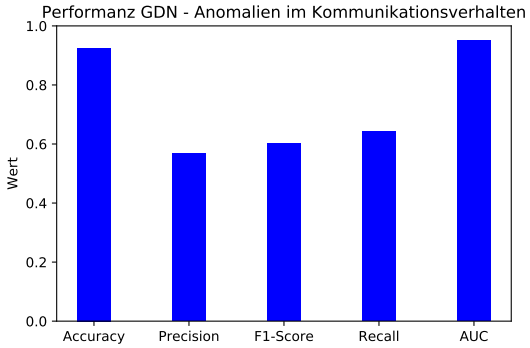


Abbildung 6.20: Ergebnisse des GDN für Datensatz 1m_8996_50p

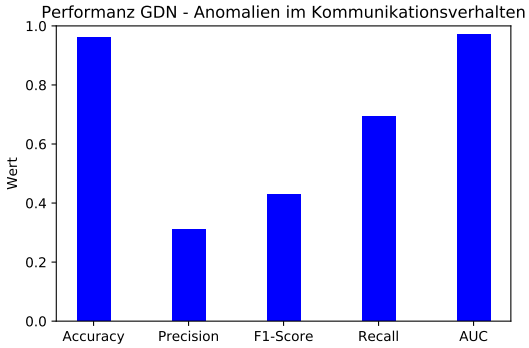


Abbildung 6.21: Ergebnisse des GDN für Datensatz 15m_2012_50p

Die alle 15 Minuten gesendeten anomalen Verhandlungsstarts werden mit einer Precision von $0,31$, einem Recall von $0,69$ und einem F1-Score von $0,43$ erkannt. Die Accuracy liegt hier bei $0,96$ und die AUC bei $0,97$.

6.6.3 Anomalien in der Kommunikationstopologie

Die Anomalien in der Topologie, bei denen Nachbarn dazu kommen, können von dem GDN relativ gut erkannt werden. Die Ergebnisse sind in Abbildung 6.22 zu sehen. Der Recall liegt bei $0,85$, die Precision dagegen bei $0,54$. Es werden ein F1-Score von $0,66$ sowie eine Accuracy und eine AUC von $0,99$ erreicht.

Der Fall, bei dem Nachbarn wegfallen, kann etwas schlechter, aber dennoch relativ gut erkannt werden mit einem Recall von $0,82$. Die Precision liegt hier bei $0,34$, der F1-Score bei $0,48$. Die Accuracy ist $0,97$ und die AUC ist $0,96$.

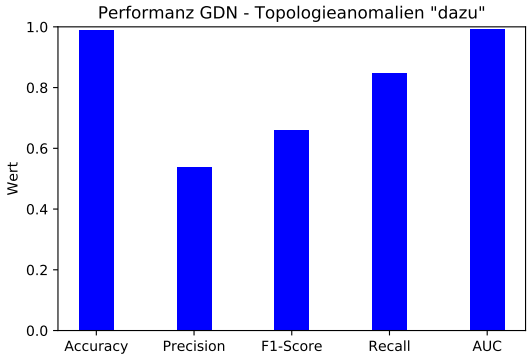


Abbildung 6.22: Ergebnisse des GDN für Topologieanomalien (Nachbarn kommen hinzu)

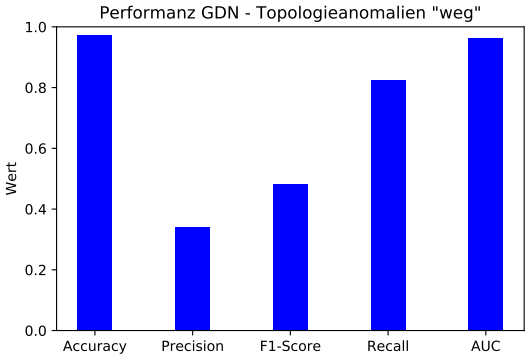


Abbildung 6.23: Ergebnisse des GDN für Topologieanomalien (Nachbarn fallen weg)

6.6.4 Anomalien in den Leistungswerten und der Kommunikationstopologie

Das GDN erkennt die Kombination aus Anomalien in den Leistungswerten und der Topologie für die alle 5 Sekunden aufgefüllten Daten mit hohem Recall (0.96), aber niedriger Precision (0.14). Der F1-Score liegt bei 0.24. Aufgrund des sehr niedrigen Anteils an Anomalien ist die Accuracy 0.98 und die AUC 0.99. Die Ergebnisse sind in Abbildung 6.24 dargestellt. Im Anhang unter Tabelle A.1 befinden sich die Ergebnisse für die sekündlich aufgefüllten Daten, da die Precision und der Recall dort etwas geringer sind.

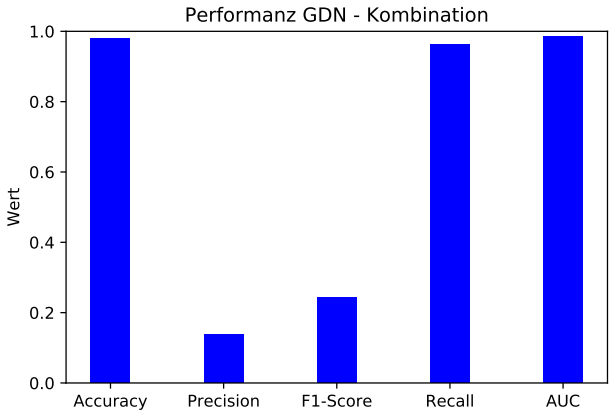


Abbildung 6.24: Ergebnisse des GDN für Daten mit Anomalien in den Werten und der Topologie

7 Evaluation

In diesem Kapitel werden die Ergebnisse evaluiert, um die Forschungsfragen zu beantworten.

7.1 Evaluation Forschungsfrage 1

> *Wie gut eignen sich klassische Machine Learning (ML) Verfahren für die Erkennung von verschiedenen Arten von Anomalien in der Kommunikation eines Winzent-basierten Multi-Agentensystem (MAS) für die Koordination von dezentralen Energiespeichern?*

Um die erste Forschungsfrage zu beantworten, werden nachfolgend die Ergebnisse der klassischen Modelle für die verschiedenen Arten von Anomalien evaluiert. Dafür wird die Performanz der jeweiligen Modelle auf den Testdaten betrachtet. Für den Isolation Forest und die One-Class Support Vector Machine (OCSVM) wird die Performanz auf den Trainingsdaten hier nicht betrachtet, da der Observer mit trainierten Modellen die Anomalieerkennung ebenfalls nur auf vorher unbekannten Testdaten ausführt und diese Ergebnisse daher von höherer Bedeutung sind.

In den folgenden Plots sind für die Vollständigkeit alle betrachteten Kennzahlen für die Performanz abgebildet. Die Accuracy und die Area Under Curve (AUC) geben eine Einschätzung der Gesamtperformanz (siehe Unterabschnitt 2.5.7), sind aber sehr abhängig von dem Anteil der Anomalien in den Daten. In den Ergebnissen wurde bereits gezeigt, dass diese Werte sehr hoch sein können, wenn sehr wenige Anomalien vorliegen, obwohl diese nicht gut erkannt werden. Zur Bewertung der Eignung der Modelle für die jeweilige Anomalieart werden daher hauptsächlich der Recall und die Precision betrachtet. Der Recall gibt an, wie groß der Anteil der vorhandenen Anomalien ist, die das Modell als solche erkennt (siehe Unterabschnitt 2.5.7). Ein Recall von $1,0$ bedeutet also, dass alle Anomalien erfasst werden. Die Precision gibt den Anteil der vom Modell gefundenen Anomalien an, der auch tatsächlich anomal ist (siehe Unterabschnitt 2.5.7). Eine Precision von $1,0$ bedeutet, dass keine False Positives (”Fehlalarme”) vorliegen. Diese beiden Größen sind daher aussagekräftig zur Bewertung der Ergebnisse.

In Abbildung 7.1 ist die Performanz der klassischen Modelle auf Daten mit Anomalien in den Leistungswerten dargestellt. Der Isolation Forest und die OCSVM erreichen einen Recall von $1,0$, sie erkennen also alle Werteanomalien. Allerdings ist die Precision mit $0,64$ bei dem Isolation Forest und $0,62$ bei der OCSVM niedriger als bei dem Autoencoder, für den sie $0,68$ beträgt. Der Autoencoder erreicht einen Recall von $0,89$. Insgesamt erkennen die klassischen Modelle die Werteanomalien als solche, allerdings mit einigen False Positives. Der Isolation Forest kann hier die besten Ergebnisse erzielen.

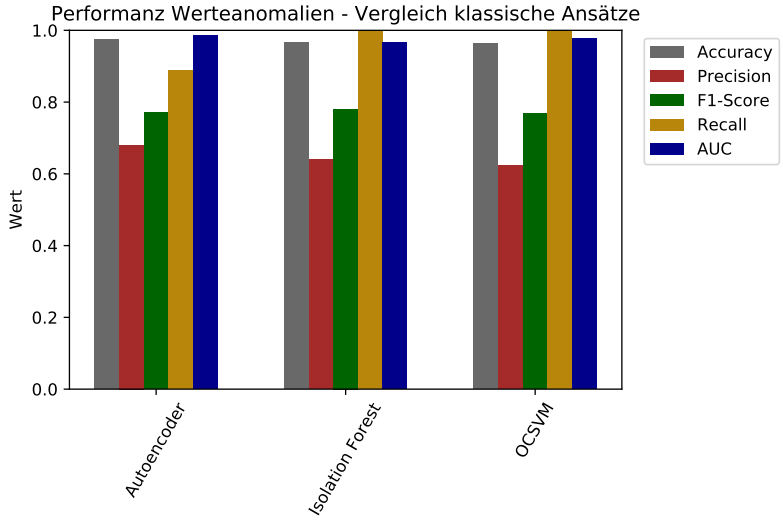


Abbildung 7.1: Die Ergebnisse der klassischen Modelle im Vergleich für Anomalien in den Leistungswerten.

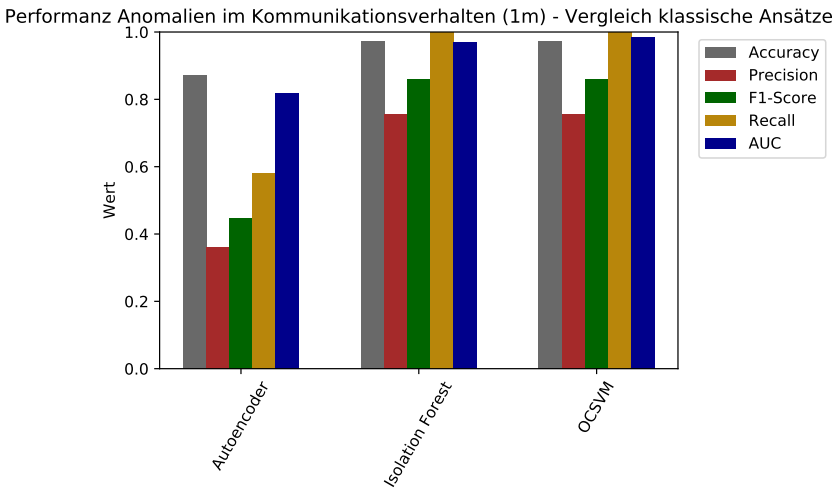


Abbildung 7.2: Die Ergebnisse der klassischen Modelle im Vergleich für Anomalien im Kommunikationsverhalten (1m).

Abbildung 7.2 zeigt die Ergebnisse der klassischen Modelle für Anomalien im Kommunikationsverhalten für Häufigkeit der anomalen Verhandlungsstarts von einer Minute. Der Isolation Forest und die OCSVM sind geeignet für die Erkennung dieser Anomalieart, wenn auch die Zahl der False Positives verbesserungswürdig ist. Sie erreichen einen Recall von $1,0$ und erkennen somit alle anomalen Verhandlungsstarts. Dabei liegt die Precision für den Isolation Forest und für die OCSVM bei $0,76$, es gibt also weniger False Positives als bei den Modellen für die Werteanomalien. Der Autoencoder ist mit einem Recall von $0,58$ und einer Precision von $0,36$ weniger gut geeignet.

Bei den Anomalien im Kommunikationsverhalten für den Fall, dass alle 15 Minuten eine anomale Verhandlung gestartet wird, ist die Performanz der klassischen Modelle in Abbildung 7.3 dargestellt. Die Modelle erkennen diese Anomalien mit geringerer Performanz als bei der Häufigkeit von einer Minute. Der Isolation Forest und die OCSVM erkennen zwar alle Anomalien (Recall von $1,0$), jedoch ist die Zahl der False Positives bei dem Isolation Forest sehr hoch (Precision von $0,20$) und bei der OCSVM hoch (Precision von $0,42$). Der Autoencoder kann die Anomalien sehr schlecht erkennen (Recall von $0,19$ und Precision von $0,13$). Insgesamt ist die OCSVM am besten für die Erkennung dieser Anomalieart für diese Häufigkeit geeignet, wenn eine niedrige Precision in Kauf genommen wird. Eine Häufigkeit von 15 Minuten für anomale Verhandlungsstarts ist für die Modelle schwer zu erkennen, da normalerweise mehrere Nachrichten pro Sekunde gesendet werden und ebenfalls in unregelmäßigen Zeitintervallen von einigen Sekunden keine Nachrichten gesendet werden. Die Regelmäßigkeit mit so großen Abständen von 15 Minuten lässt sich daher von den Modellen schwer erfassen.

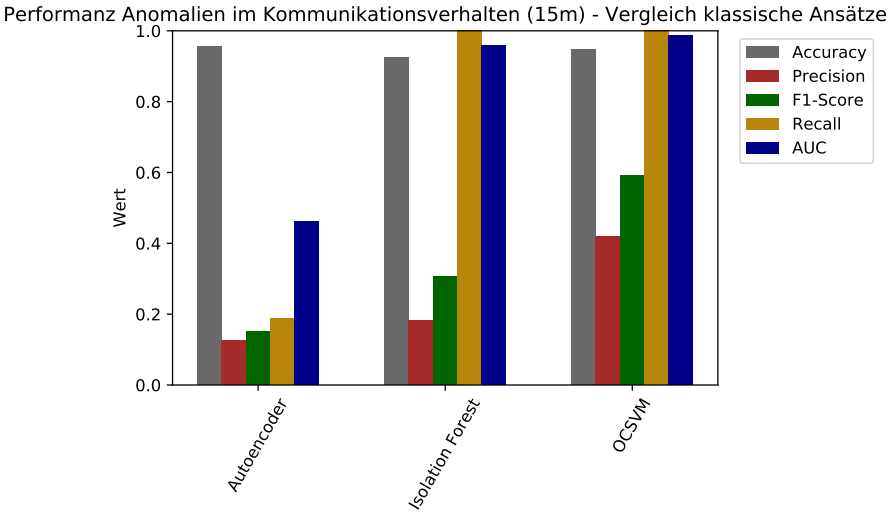


Abbildung 7.3: Die Ergebnisse der klassischen Modelle im Vergleich für Anomalien im Kommunikationsverhalten (15m).

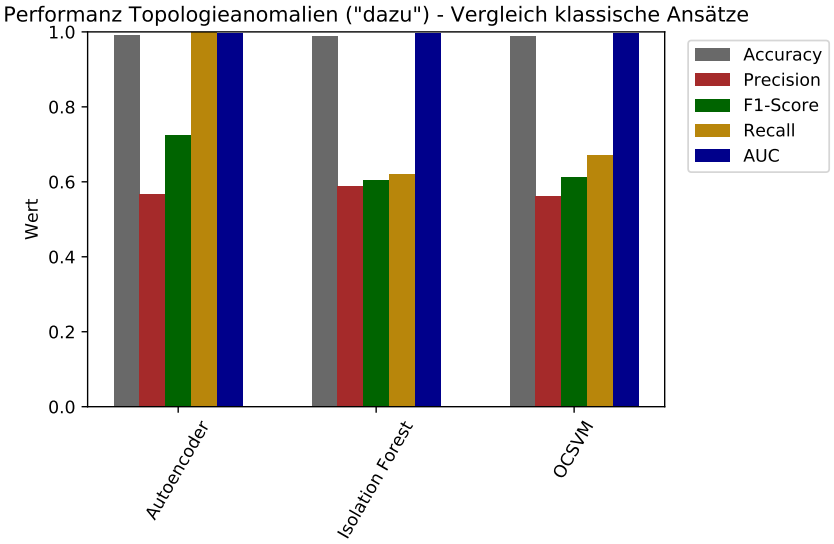


Abbildung 7.4: Die Ergebnisse der klassischen Modelle im Vergleich für Anomalien in der Kommunikationstopologie (Nachbarn kommen dazu).

In Abbildung 7.4 sind die Ergebnisse der klassischen Modelle für Topologieanomalien, bei denen Nachbarn dazu kommen, abgebildet. Der Autoencoder erkennt alle Anomalien (Recall von $1,0$), allerdings liegt die Precision nur bei $0,57$. Dennoch ist der Autoencoder geeignet um zu erkennen, ob der betrachtete Agent Nachrichten an weitere Agenten schickt, die nicht in seiner Nachbarschaft enthalten sind. Der Isolation Forest und die OCSVM

können die Anomalien weniger gut erkennen (Recall von $0,62$ bzw. $0,68$ und Precision von $0,59$ bzw. $0,56$).

Abbildung 7.5 zeigt die Performanz der klassischen Modelle für Anomalien in der Topologie, bei denen an bestimmte Nachbarn keine Nachrichten mehr gesendet werden. Keiner der klassischen Ansätze kann diese Anomalieart erkennen. Die Precision und der Recall sind für alle Modelle unter $0,03$. Grund dafür kann das Szenario sein, in dem normalerweise jeder Nachrichtentyp prinzipiell an alle oder nur an bestimmte Nachbarn gesendet wird. Nachrichten wie beispielsweise *Demand Notifications* können als Broadcast-Nachricht an alle Nachbarn oder aber auch als Antwort auf eine *Offer Notification* an einen bestimmten Nachbarn gesendet werden. Es kann also sehr unterschiedlich sein, an wie viele und an welche Nachbarn der Agent eine Nachricht schickt. Die Modelle erkennen offenbar kein Muster darin, dass im Fall der manipulierten Daten bestimmte Nachbarn wegfallen. Ein weiterer möglicher Grund für die schlechte Performanz ist die Art, die Anomalien einzufügen. Es wurden in den entsprechenden Zeilen in den Normaldaten, bei denen ursprünglich eine Nachricht gesendet wurde, der Zeitstempel unverändert behalten und alle weiteren Datenfelder auf den Wert -1, wie beim Auffüllen der Daten, gesetzt. In weiterführenden Arbeiten könnte untersucht werden, ob sich die Ergebnisse verbessern, wenn die gesamten Zeilen entfernt und anschließend der Datensatz wieder für jede volle Sekunde aufgefüllt wird.

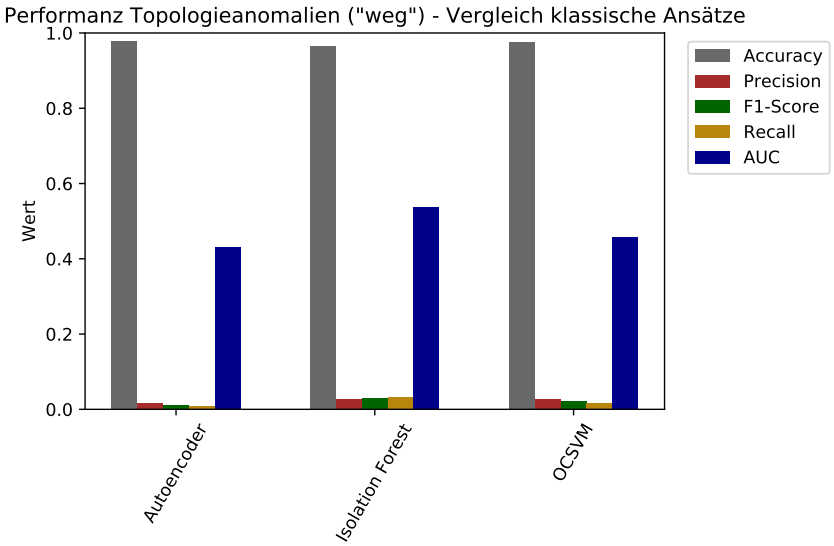


Abbildung 7.5: Die Ergebnisse der klassischen Modelle im Vergleich für Anomalien in der Kommunikationstopologie (Nachbarn fallen weg).

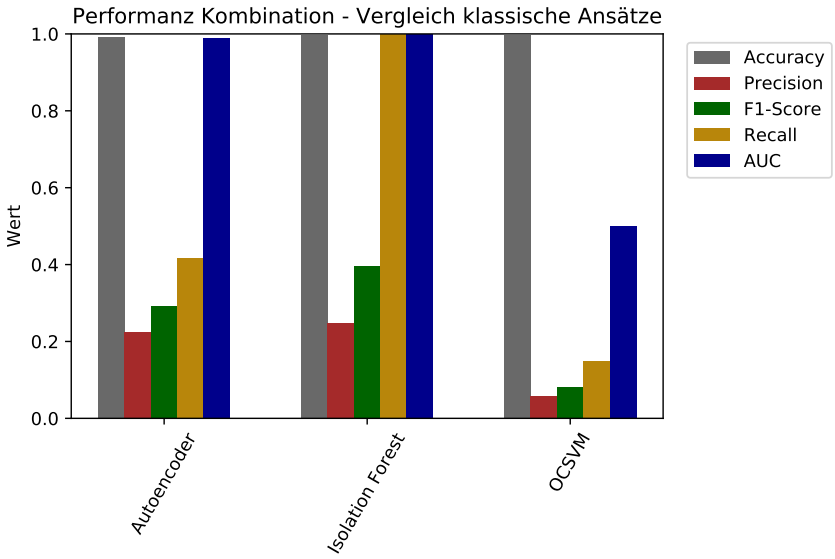


Abbildung 7.6: Die Ergebnisse der klassischen Modelle im Vergleich für Anomalien in den Leistungswerten und der Kommunikationstopologie.

In Abbildung 7.6 sind die Ergebnisse der klassischen Modelle für die Kombination aus Anomalien in den Leistungswerten und der Topologie zu sehen. Die OCSVM ist nicht geeignet für die Erkennung dieser Anomalien, wie an der niedrigen Precision von $0,06$ und dem geringen Recall von $0,15$ zu erkennen ist. Auch der Autoencoder kann die Anomalien mit einer Precision von $0,22$ und einem Recall von $0,42$ nicht gut erkennen. Der Isolation Forest ist jedoch geeignet, wenn eine niedrige Precision in Kauf genommen wird. Es können bei dem verwendeten Datensatz alle Anomalien erkannt werden (Recall von $1,0$), wobei die Precision dann nur bei $0,25$ liegt.

Zusammenfassend kann festgehalten werden, dass für die Erkennung von Anomalien in den Leistungswerten der Isolation Forest am besten geeignet ist, wobei die OCSVM ähnliche Ergebnisse erzielt.

Für die Erkennung von Anomalien im Kommunikationsverhalten eignen sich der Isolation Forest und die OCSVM gleich gut bei einer Häufigkeit der anomalen Nachrichten von einer Minute. Hier können bessere Ergebnisse in der Precision erzielt werden, wenn die Häufigkeit der anomalen Verhandlungsstarts höher ist. Die Regelmäßigkeit der anomalen Nachrichten kann also von den Modellen besser erfasst werden, wenn die zeitlichen Abstände kleiner sind.

Anomalien in der Topologie, bei denen Nachbarn dazu kommen, können von dem Autoencoder am besten erkannt werden. Da dieser auf Normaldaten trainiert wird, in denen die anomalen Nachbarn nicht vorkommen, können in den Testdaten diese entsprechenden

Instanzen erkannt werden. Allerdings ist der Autoencoder aufgrund der Precision von $0,57$ nicht nur auf das Erkennen dieser Anomalien angepasst, sondern es werden viele weitere Datenpunkte als anomal klassifiziert.

Der Fall der Topologieanomalien, bei dem Nachbarn wegfallen, kann von keinem der klassischen Modelle erkannt werden.

Für die Kombination aus Anomalien in den Leistungswerten und der Topologie eignet sich der Isolation Forest von allen klassischen Modellen am besten, wobei allerdings nur eine geringe Precision erreicht werden kann. Es gibt also eine hohe Zahl an False Positives und das Modell ist insgesamt nicht gut an die Daten angepasst.

Mindestens ein klassisches Modell eignet sich für die Erkennung von Anomalien in den Leistungswerten, im Kommunikationsverhalten und in der Topologie, wenn Nachbarn dazu kommen. Allerdings kann keiner der Ansätze eine ausreichend hohe Precision erreichen, sodass die fehlerhafte Klassifizierung normaler Instanzen als Anomalien nicht vernachlässigbar ist.

Insgesamt kann kein klassischer Ansatz die anderen für alle Anomaliearten übertreffen, allerdings erzielt der Isolation Forest für Werteanomalien und die Kombination zweier Anomaliearten die besten Ergebnisse im Vergleich zu den anderen Ansätzen.

Die klassischen Verfahren eignen sich nicht für die Erkennung von Anomalien in der Topologie, wenn Nachbarn dazu kommen. Für eine zuverlässige Erkennung der Kombination aus Anomalien in den Werten und der Topologie ist die Performanz ebenfalls nicht ausreichend.

7.2 Evaluation Forschungsfrage 2

Eignen sich graphbasierte Machine Learning Verfahren besser für die Erkennung bestimmter Anomalien in der Kommunikation eines Winzent-basierten MAS für die Koordination von dezentralen Energiespeichern, die von den klassischen Ansätzen weniger gut erkannt werden?

Die klassischen Modelle können die Topologieanomalien, bei denen Nachbarn wegfallen, nicht erkennen. Abbildung 7.7 zeigt die Performanz aller Ansätze für diese Anomalieart. Es ist zu sehen, dass das Graph Deviation Network (GDN) die klassischen Ansätze hier deutlich übertrifft. Es kann ein Großteil der Anomalien erkannt werden (Recall von $0,82$). Die Precision liegt nur bei $0,34$, dennoch kann das GDN als geeignet für die Erkennung der Topologieanomalien, bei denen Nachbarn wegfallen, eingestuft werden. Die Abhängigkeiten zwischen Features werden von dem GDN als Kanten eines Graphen interpretiert und während des Trainings gelernt. Das Verfahren kann offenbar während des Trainings lernen, dass bestimmte Nachrichtentypen normalerweise an bestimmte Nachbarn gesendet werden. Fehlen diese Nachbarn in den Testdaten, kann dies für einen hohen Anteil der Fälle erkannt werden. Die niedrige Precision lässt sich damit erklären, dass das GDN viele weitere Abhängigkeiten lernt, welche sich in den Testdaten verändern können und als Anomalien erkannt werden, ohne dass diese tatsächlich vorliegen.

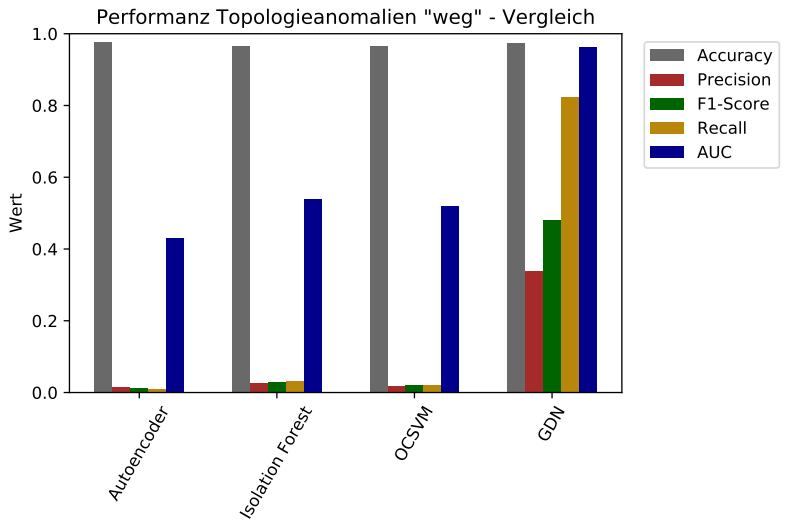


Abbildung 7.7: Vergleich der Performanz aller Modelle für Anomalien in der Kommunikationstopologie (Nachbarn fallen weg).

In Abbildung 7.8 ist ein Überblick über die Performanz des GDN für die verschiedenen Arten von Anomalien gegeben. Es ist zu sehen, dass das GDN die besten Ergebnisse für

Anomalien in den Werten erzielt. Beide Arten von Topologieanomalien können mit ordentlichem Recall, aber mittlerer bis geringer Precision erkannt werden. Für die Kombination aus Werte- und Topologieanomalien erzielt das GDN ähnliche Ergebnisse wie der Isolation Forest: sehr hoher Recall aber sehr geringe Precision.

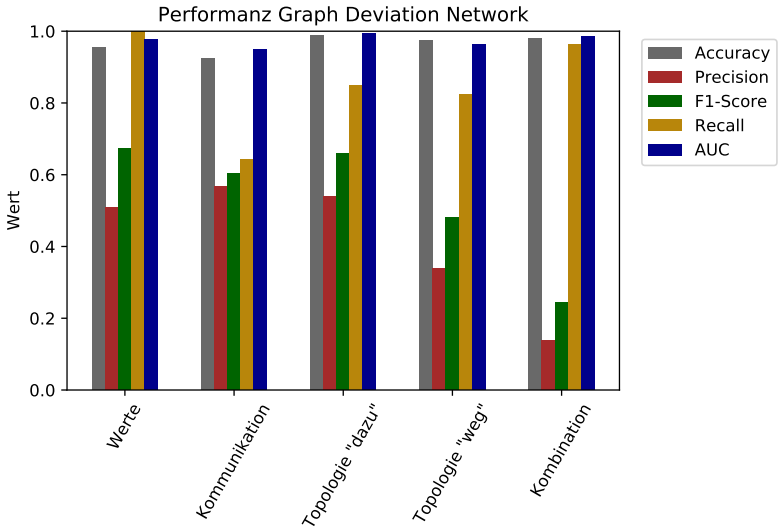


Abbildung 7.8: Performanz des GDN je Anomalieart.

Insgesamt lässt sich feststellen, dass das GDN besser geeignet ist für die Erkennung einer bestimmten Anomalieart, die von den klassischen Modellen nicht erkannt werden kann. Es stellt ein mögliches Verfahren dar zur Erkennung von Anomalien in der Topologie, bei denen Nachbarn wegfallen. Auch für Werteanomalien und Topologieanomalien, bei denen Nachbarn dazu kommen, zeigt das GDN eine ordentliche Performanz für den Recall. Wie bei den klassischen Modellen auch ist jedoch die Precision nicht zufriedenstellend.

8 Fazit und Ausblick

In diesem abschließenden Kapitel wird nach einer kurzen Zusammenfassung der Untersuchungen ein Fazit gezogen, in dem auch auf die Vorarbeiten und die verwandte Masterarbeit ([19]) Bezug genommen wird. Schließlich wird in einem Ausblick festgehalten, welche weiteren Untersuchungen an diese Arbeit anschließen können.

8.1 Fazit

In dieser Arbeit wurden Untersuchungen zur verteilten Anomalieerkennung in der Kommunikation eines Winzent-basierten Multi-Agentensystem (MAS) für die Koordination von dezentralen Energiespeichern durchgeführt. Dafür wurde ein Observer entwickelt, welcher einen einzelnen Agenten überwacht und verschiedene Arten von Anomalien erkennt. Der Observer wurde für die Betrachtung jeder Aomalieart jeweils für einen manipulierten Agenten entwickelt und analysiert. Die verwendeten Verfahren und Daten sowie die Entscheidung für eine host-based Sicht werden anhand von eigenen Vorarbeiten sowie einer vorangegangenen Masterarbeit ([19]) ausgewählt. Das Szenario besteht aus einem MAS, in welchem fünf Agenten je einen Energiespeicher verwalten und im Sinne einer Multi-Purpose-Nutzung der Energiespeicher durch Kooperation eine effiziente Energienutzung ermöglichen. Dafür können die Agenten bei Prognosefehlern für die Last mit anderen Agenten über einen Ausgleich des Ungleichgewichts verhandeln. Für die Verhandlungen werden Nachrichten nach dem Kommunikationsprotokoll Winzent ausgetauscht.

Es wurden verschiedene Verfahren zur Anomalieerkennung eingesetzt, um diese zu vergleichen und einen möglichst zuverlässigen Observer zu entwickeln. Dabei wurde zwischen klassischen und graphbasierten Ansätzen unterschieden. Die klassischen Ansätze sind in der Literatur sehr verbreitet zur Anomalieerkennung in verschiedenen Anwendungsbereichen, unterscheiden sich dennoch algorithmisch in ihrer Funktionsweise. Es wurden ein Autoencoder, ein Isolation Forest und eine One-Class Support Vector Machine (OCSVM) ausgewählt. Diese wurden sowohl in den Vorarbeiten als auch in [19] (mit dem Unterschied, dass dort eine Support Vector Machine (SVM) statt einer OCSVM verwendet wird, Unterschied siehe Unterabschnitt 2.6.3) untersucht, weshalb sie für die Untersuchungen dieser Arbeit gewählt wurden. Der graphbasierte Ansatz Graph Deviation Network (GDN) wurde

für den Vergleich ausgewählt, da dieser sich algorithmisch grundlegend von den klassischen Ansätzen unterscheidet und eine Implementierung öffentlich verfügbar und somit schnell zugänglich ist.

Bei dem anomalen Verhalten kann es sich um verschiedene Arten von Anomalien handeln. Es werden Anomalien in den Leistungswerten, die als Antwort auf Verhandlungsfragen zugesichert werden, untersucht. Weiterhin werden Anomalien im Kommunikationsverhalten betrachtet. Dabei schickt ein Agent in regelmäßigen Zeitabständen nicht notwendige Verhandlungsanfragen an seine Nachbarn. Hier wurden verschiedene Häufigkeiten von einer Minute und 15 Minuten betrachtet. Die dritte Art von Anomalien sind Anomalien in der Topologie, wobei zwei Fälle untersucht wurden. Im ersten Fall schickt ein Agent Nachrichten an weitere Agenten, die eigentlich nicht in seiner Nachbarschaft enthalten sind und mit denen der Agent normalerweise nur indirekt über andere Agenten verhandeln kann. Im zweiten Fall schickt ein Agent an bestimmte Nachbarn keine Nachrichten mehr, wodurch keine Verhandlungen mehr zwischen dem Agenten und diesen Nachbarn möglich ist. Um zu untersuchen, welche Anomaliearten von welchen Verfahren am besten erkannt werden können, wurde jedes Verfahren einzeln für jede Anomalieart entwickelt und ausgewertet. Da in der Realität jedoch verschiedene Arten von Anomalien gleichzeitig auftreten können, wird jeder Ansatz zusätzlich mit einer Kombination aus zwei Anomaliearten gleichzeitig untersucht. Dabei handelt es sich um Anomalien in den Leistungswerten von Verhandlungsstarts sowie Anomalien in der Topologie, bei denen Verhandlungsanfragen an weitere Agenten geschickt werden, die nicht Nachbarn des sendenden Agenten sind.

Insgesamt konnte in den Untersuchungen kein Ansatz die anderen für alle Anomaliearten übertreffen. Es gibt jedoch für jede Art von Anomalie mindestens einen Ansatz, der für die Erkennung dieser Anomalien geeignet ist. Zur Veranschlaulichung befinden sich die Ergebnisse aller Ansätze je Anomalieart im Anhang unter Abschnitt A.3. Für die Erkennung von Werteanomalien sind alle Ansätze gut geeignet. Zur Erkennung von Anomalien im Kommunikationsverhalten eignen sich der Isolation Forest und die OCSVM. Anomalien in der Topologie für den Fall, dass Nachbarn dazu kommen, können von dem Autoencoder am besten erkannt werden. Der Fall, dass Nachbarn wegfallen, wird vom GDN erkannt. Die Kombination aus zwei Anomaliearten ist für alle Ansätze schwierig zu erkennen und nur mit einer niedrigen Precision. Es ist offenbar schwierig, ein Modell an verschiedene Arten von Anomalien gleichzeitig anzupassen. Daher ist es sinnvoll, einen Observer einzusetzen, der eine Kombination aus den verwendeten Verfahren nutzt, um jede Art von Anomalie einzeln abzufangen und in Summe alle auftretenden Anomalien zuverlässig zu erkennen. Darüber hinaus können Erkenntnisse aus [19] einbezogen werden über den Informationsgehalt je Anomalieart und Verfahren. Dort konnte gezeigt werden, dass abhängig von den zur Verfügung stehenden Informationen unterschiedliche Verfahren eingesetzt werden sollten.

Es konnten weiterhin einige Ergebnisse der Vorarbeiten bestätigt werden. Auf den synthetischen Daten konnten die klassischen Verfahren Werteanomalien besser erkennen als Anomalien in der Topologie. Dies trifft für den Isolation Forest und die OCSVM für die in dieser Arbeit verwendeten Simulationsdaten ebenfalls zu. Für den Autoencoder trifft dies nur für die Topologieanomalien zu, bei denen Nachbarn wegfallen. Im Fall hinzukommender Nachbarn erkennt der Autoencoder diese Anomalien etwas besser als die Werteanomalien. Die Beobachtungen bzgl. des GDN aus den Vorarbeiten bestätigen sich für die Simulationsdaten jedoch nicht. Das GDN erkennt zwar den Fall der Topologieanomalien, bei dem Nachbarn wegfallen, als einziger Ansatz, jedoch kann es für Werteanomalien die höchste Performanz im Vergleich zu den anderen Anomaliearten erzielen. Dennoch zeigen die Ergebnisse, dass das GDN ein gutes Verfahren für die Erkennung von Topologieanomalien ist.

Vergleicht man die Ergebnisse dieser Arbeit, in der ein host-based Observer entwickelt wurde, mit der vorangegangenen Masterarbeit [19], in der ein network-based Observer implementiert wurde, lassen sich die folgenden Erkenntnisse festhalten. Für denselben Datensatz (mit entsprechender Vorverarbeitung mit Filtern und Auffüllen für den host-based Ansatz) mit Anomalien im Kommunikationsverhalten mit einer Häufigkeit von einer Minute der anomalen Verhandlungsstarts erzielt die host-based Anomalieerkennung bessere Ergebnisse als der network-based Ansatz. Im network-based Ansatz konnte der Autoencoder eine Precision von $1,0$ und einen Recall von $0,0$ erreichen, er erkennt also keine der anomalen Instanzen auch als Anomalien. Der Isolation Forest erzielt dort eine Precision von $0,30$ und einen Recall von $0,55$. Bei der SVM lagen die Precision dort bei $1,0$ und der Recall bei $0,08$, es können also auch hier fast keine der Anomalien als solche erkannt werden. In dem host-based Szenario können von dem Isolation Forest und der OCSVM hier deutlich bessere Ergebnisse erzielt werden. Bei beiden Modellen liegt im host-based Ansatz der Recall bei $1,0$ und die Precision bei $0,76$. Der Autoencoder im host-based Szenario erzielt zudem mit $0,58$ einen besseren Recall als der network-based Autoencoder. Es kann also besser erkannt werden, wenn ein Agent in regelmäßigen kurzen Zeitabständen anomale Verhandlungsanfragen versendet, wenn nur dieser Agent von dem Observer überwacht wird.

Weiterhin wurde in [19] untersucht, ob ein Ausfall eines Agenten im System erkannt werden kann. Dazu wurde im network-based-Blick auf die Daten ein Agent für die Erzeugung von zwei Datensätzen dahingehend manipuliert, dass er ab einem bestimmten Zeitpunkt keine Nachrichten mehr sendet. Dies ist dem Fall der hier betrachteten Topologieanomalien, bei denen an bestimmte Nachbarn keine Anomalien mehr gesendet werden, ähnlich. Vergleicht man die Ergebnisse, kann das GDN aus dem host-based Szenario den Autoencoder des network-based Szenarios übertreffen. Der Autoencoder erreicht dort für

einen Datensatz eine Precision von $1,0$, jedoch nur einen Recall von $0,15$ und bei dem anderen Datensatz sind diese Werte $0,0$. Das GDN erzielt hier einen wesentlich höheren Recall von $0,82$, wobei die Precision nur bei $0,34$ liegt. Dennoch ist das GDN wesentlich sensitiver bzgl. der Anomalien.

8.2 Ausblick

Weitere Untersuchungen des aktuellen Settings Um noch aussagekräftigere Ergebnisse zu erhalten, wäre ein höherer Anteil an Anomalien in den Daten mit Topologieanomalien und der Kombination aus zwei Anomaliearten wünschenswert. Wenn mehr Anomalien in den Daten vorliegen, kann mit mehr Zuverlässigkeit bestimmt werden, ob das Modell an die Erkennung dieser Anomalien angepasst ist. Dabei ist anzumerken, dass Anomalien weiterhin wenig und selten vorkommen sollten, allerdings handelt es sich in den genannten Datensätzen um einen Anteil von $1,4\%$ oder weniger, der somit sehr niedrig ist. Da die Anzahl der Nachrichten nicht beeinflusst werden kann, weil die Verhandlungen in der Realität ähnlich wie in der Simulation ablaufen, ist die einzige Möglichkeit das Auffüllen der Daten mit einer niedrigeren Frequenz. Die Untersuchungen mit dem Datensatz, der die Kombination aus Werte- und Topologieanomalien enthält, haben gezeigt, dass ein Auffüllen jeder fünften vollen Sekunde bei einigen Modellen zu etwas besseren Ergebnissen oder zumindest keiner signifikanten Verschlechterung führt. Es könnte untersucht werden, wie die Ergebnisse mit Daten ausfallen, die beispielsweise alle $10,20$ Sekunden aufgefüllt werden. Allerdings muss hier aufgepasst werden, dass durch eine geringere Frequenz die Zeitintervalle zwischen zwei Datenzeilen nicht zu unregelmäßig werden, da häufig mehrere Nachrichten innerhalb einer vollen Sekunde gesendet werden und Verhandlungen innerhalb weniger Sekunden ablaufen. Untersuchungen mit dem Datensatz, welcher Anomalien im Kommunikationsverhalten mit einer Häufigkeit von 15 Minuten beinhaltet und für jede 30. volle Sekunde aufgefüllt wurde, haben zu sehr schlechten Ergebnissen geführt. Daher muss hier ein Mittelweg gefunden werden.

Erweiterte Untersuchungen In dieser Arbeit wurde jeweils nur ein manipulierter Agent des Systems für die Anomalieerkennung betrachtet. Eine interessante Erweiterung wäre, jedem Agenten einen bereits anhand des manipulierten Agenten trainierten Observer zuzuordnen und zu untersuchen, ob die anomalen Nachrichten des manipulierten Agenten auch bei den anderen Agenten zu anomalem Verhalten führen, welches von den jeweiligen Observern erkannt wird, oder ob die anderen Observer beispielsweise hohe False Positives verzeichnen. Dies scheint recht wahrscheinlich, da die anderen Agenten des Systems entsprechend auf die anomalen Nachrichten reagieren und beispielsweise auf häufige Verhandlungsanfragen antworten oder versuchen, einen besonders hohen Leistungswert zur

Verfügung zu stellen, wenn dieser angefragt wurde. Eine Schwierigkeit besteht hier allerdings in der Überprüfung der Ergebnisse, da anomale Reaktionen auf anomale Nachrichten schwierig zu labeln sind.

Weiterhin wurden in dem network-based Setting aus [19] keine Topologieanomalien betrachtet, bei denen Nachbarn dazu kommen. Entsprechende Untersuchungen würden einen Vergleich und eine Einschätzung zulassen, welches Setting für die Erkennung dieser Anomalieart besser geeignet ist.

Weitere Verfahren zur Anomalieerkennung Neben den Verfahren, die in dieser Arbeit verwendet wurden, gibt es weitere Ansätze, die in der Literatur zu guten Ergebnissen in der Anomalieerkennung, häufig auch im Kontext von Energiesystemen, führen. Dazu gehören beispielsweise die in Unterabschnitt 2.7.1 erwähnten Verwandten Arbeiten, in denen Verfahren wie Auto Regressive Integrated Moving Average (ARIMA), Long Short-Term Memory (LSTM) oder agentenbasierte Ansätze genannt werden. Diese könnten ebenfalls für einen Vergleich hinzugezogen werden.

A Anhang

A.1 Ergebnisse Datensatz Kombination

Modell mit Auffüllfrequenz	Accuracy	Precision	F1-Score	Recall	AUC
Autoencoder 1s	0,99	0,26	0,11	0,07	0,99
Isolation Forest 5s	0,98	0,23	0,38	1,0	0,99
OCSVM 5s	0,99	0,06	0,08	0,15	0,5
GDN 1s	0,99	0,13	0,23	0,91	0,99

Tabelle A.1: Die Performanz der Modelle mit der Auffüllfrequenz, die zu weniger guten Ergebnissen führt.

A.2 Confusion Matrizen aus den Vorarbeiten

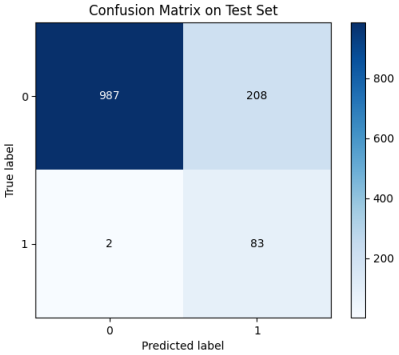


Abbildung A.1: GDN Werteanomalien host-based

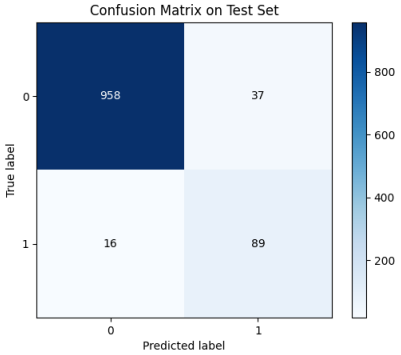


Abbildung A.2: GDN Topologieanomalien host-based

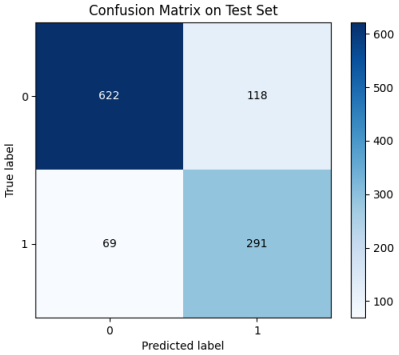


Abbildung A.3: GDN Kombination host-based

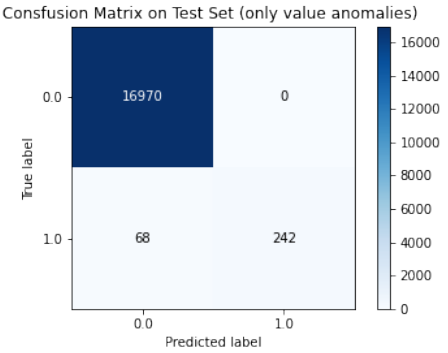


Abbildung A.4: Autoencoder Werteanomalien network-based

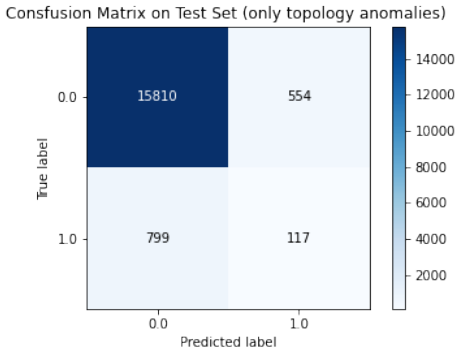


Abbildung A.5: Autoencoder Topologieanomalien network-based

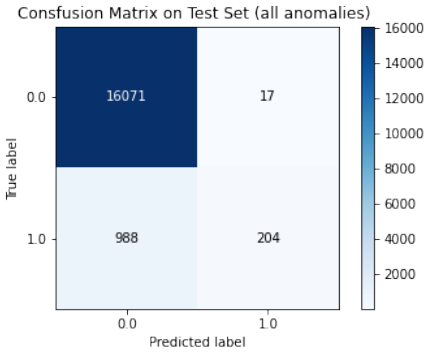


Abbildung A.6: Autoencoder Kombination network-based

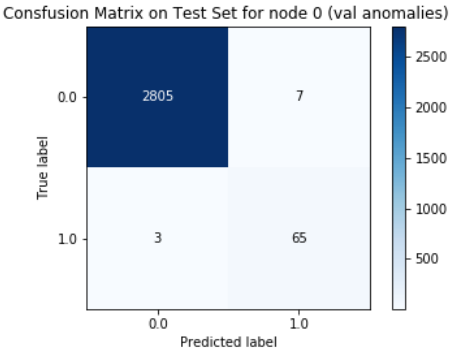


Abbildung A.7: Autoencoder Werteanomalien host-based

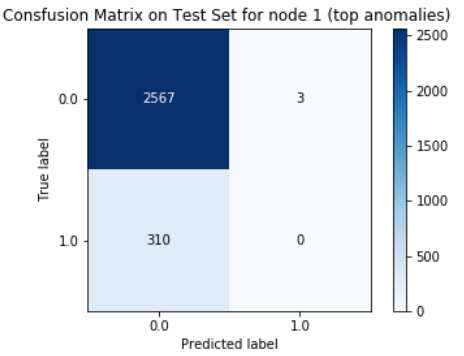


Abbildung A.8: Autoencoder Topologieanomalien host-based

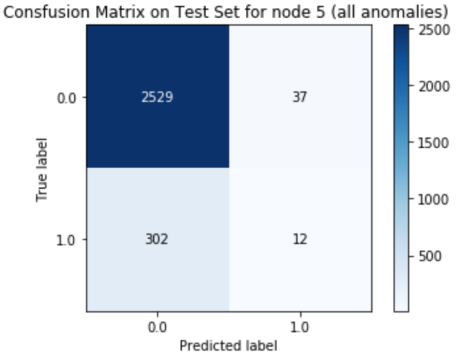


Abbildung A.9: Autoencoder Kombination host-based

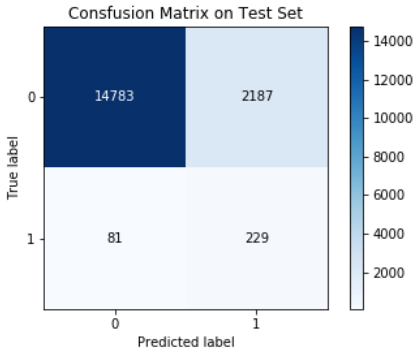


Abbildung A.10: Isolation Forest Werteanomalien network-based

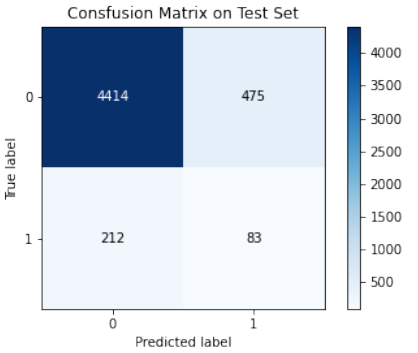


Abbildung A.11: Isolation Forest Topologieanomalien network-based

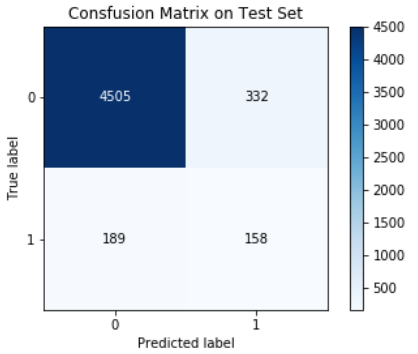


Abbildung A.12: Isolation Forest Kombination network-based

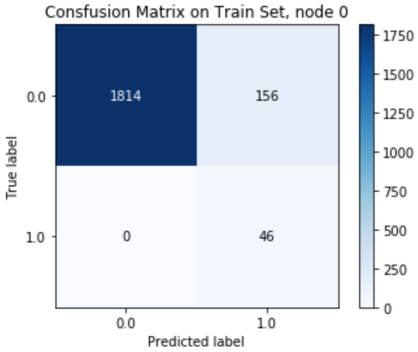


Abbildung A.13: Isolation Forest Werteanomalien host-based

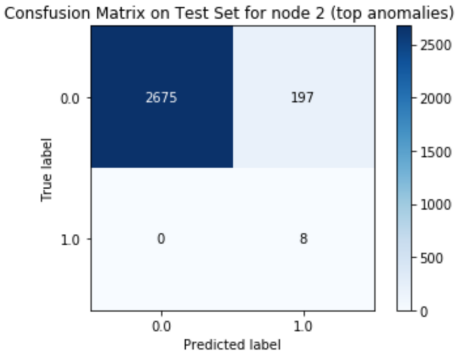


Abbildung A.14: Isolation Forest Topologieanomalien host-based

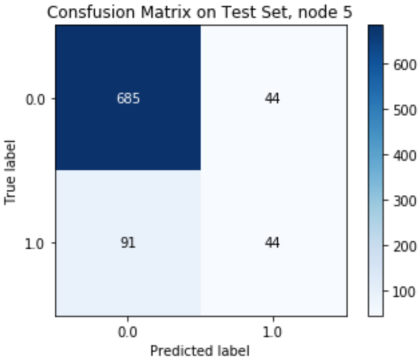


Abbildung A.15: Isolation Forest Kombination host-based

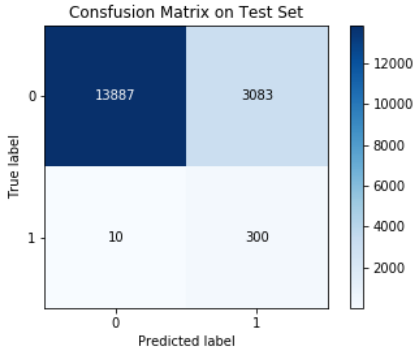


Abbildung A.16: OCSVM Werteanomalien network-based

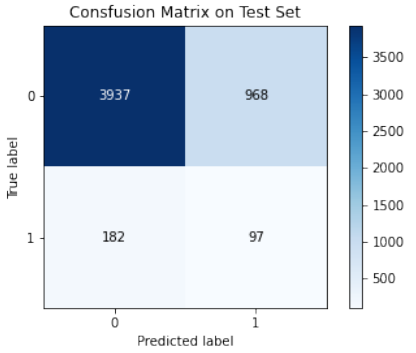


Abbildung A.17: OCSVM Topologieanomalien network-based

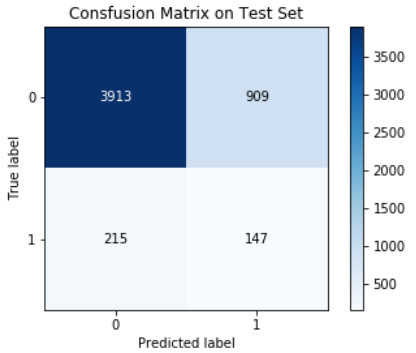


Abbildung A.18: OCSVM Kombination network-based

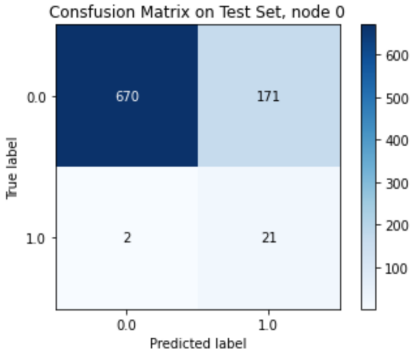


Abbildung A.19: OCSVM Werteanomalien host-based

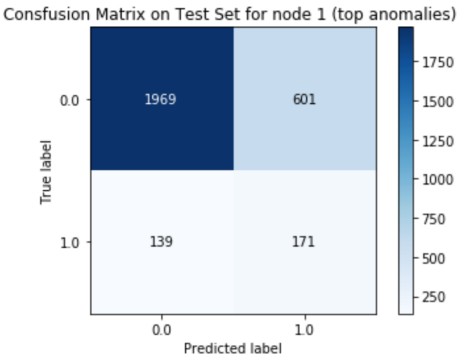


Abbildung A.20: OCSVM Topologieanomalien host-based

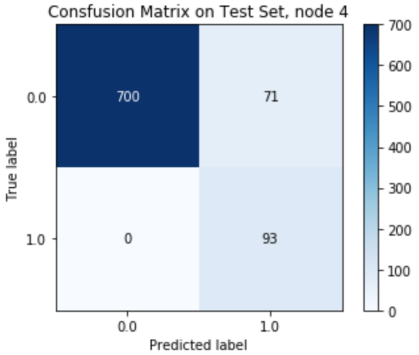


Abbildung A.21: OCSVM Kombination host-based

A.3 Ergebnisse aller Ansätze im Vergleich

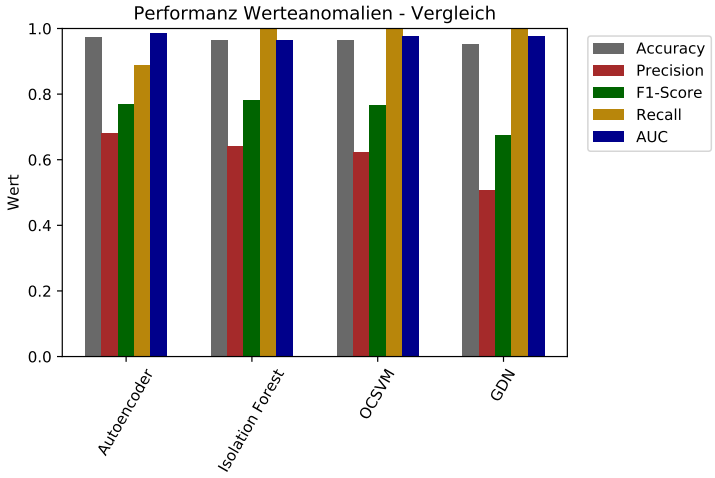


Abbildung A.22: Vergleich der Ergebnisse aller Ansätze für Anomalien in den Leistungswerten

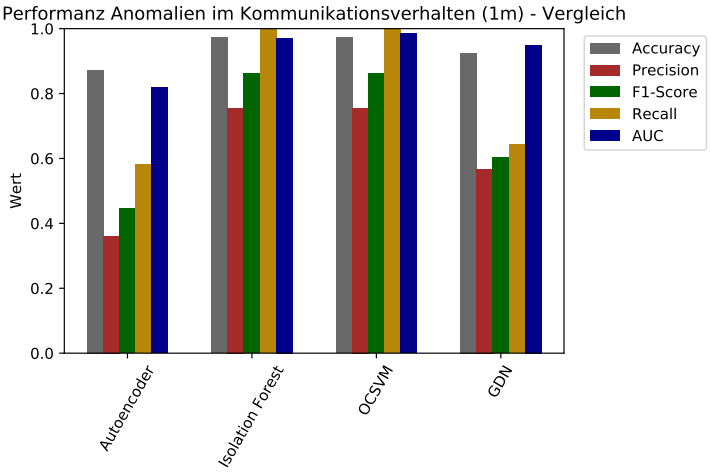


Abbildung A.23: Vergleich der Ergebnisse aller Ansätze für Anomalien im Kommunikationsverhalten (1m)

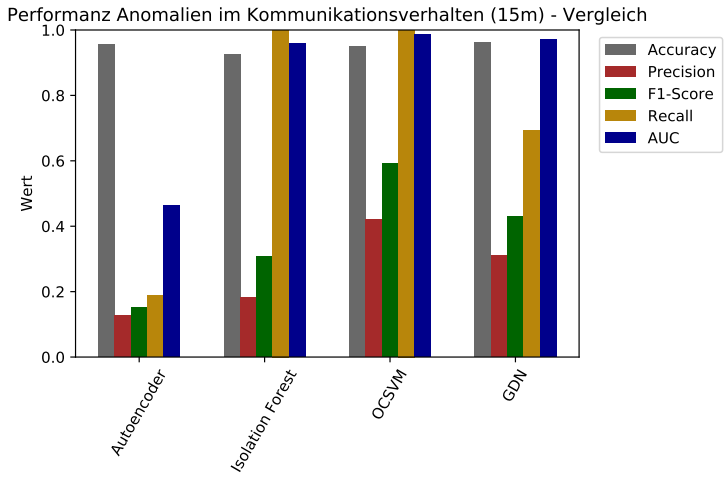


Abbildung A.24: Vergleich der Ergebnisse aller Ansätze für Anomalien im Kommunikationsverhalten (15m)

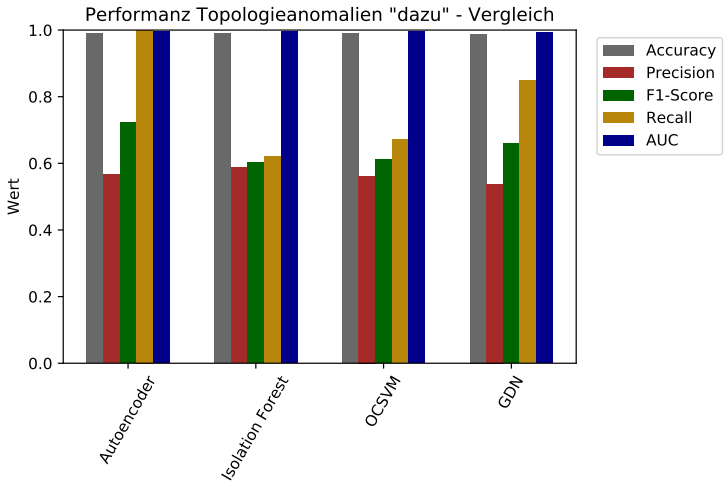


Abbildung A.25: Vergleich der Ergebnisse aller Ansätze für Anomalien in der Kommunikationstopologie (Nachbarn kommen dazu)

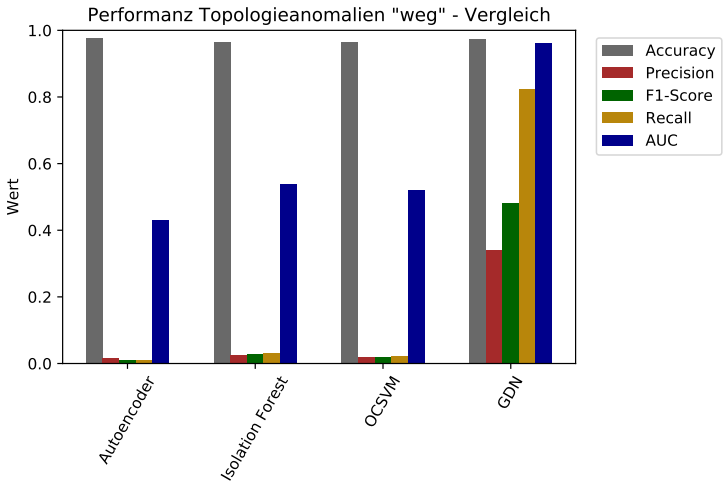


Abbildung A.26: Vergleich der Ergebnisse aller Ansätze für Anomalien in der Kommunikationstopologie (Nachbarn fallen weg)

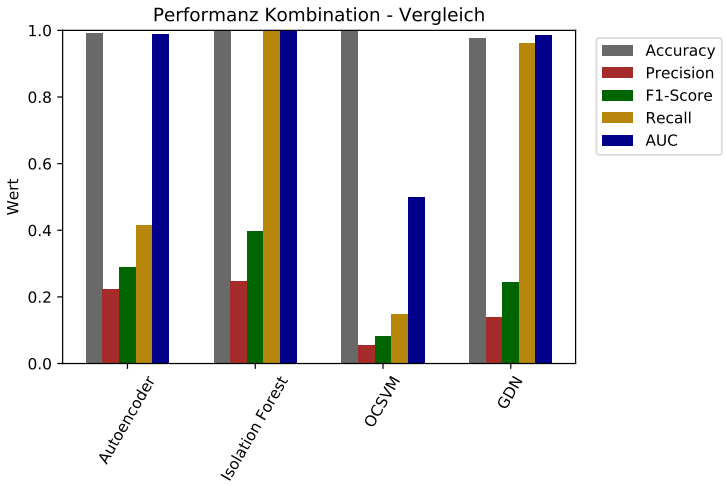


Abbildung A.27: Vergleich der Ergebnisse aller Ansätze für Anomalien in den Leistungswerten und der Kommunikationstopologie

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Adler, B.: Moderne energiesysteme – ein beitrag zur energiewende, 1–1 (2019). doi:10.1007/978-3-662-60688-9_1

[2] Bundesministerium für Bildung und Forschung: Energiewende. https://www.bmbf.de/bmbf/de/forschung/energiewende-und-nachhaltiges-wirtschaften/energiewende/energiewende_node.html

[3] Umweltbundesamt: Was ist ein "Smart Grid"? https://www.umweltbundesamt.de/service/uba-fragen/was-ist-ein-smart-grid (2013)

[4] Bundesministerium für Wirtschaft und Klimaschutz: Intelligente Netze. https://www.bmwk.de/Redaktion/DE/Artikel/Energie/intelligente-netze.html#:~:text=Der%20Begriff%20%22intelligentes%20Stromnetz%22%20(,zum%20Verbrauch%20an%20das%20Energieversorgungsnetz.

[5] Dipl.-Ing. (FH) Stefan Luber: Was ist ein Cyber-physisches System (CPS)? (2017). https://www.bigdata-insider.de/was-ist-ein-cyber-physisches-system-cps-a-668494/

[6] Wooldridge, M.: An Introduction to MultiAgent Systems, pp. 16–17 (2002)

[7] Guo, F., Herrera, L., Murawski, R., Inoa, E., Wang, C.-L., Huang, Y., Ekici, E., Wang, J., Beauchamp, P.: Real time simulation for the study on smart grid, pp. 1013–1018 (2011). doi:10.1109/ECCE.2011.6063883

[8] Veith, E.M.S.P.: Universal smart grid agent for distributed power generation management. PhD thesis, Freiberg University of Mining and Technology, Germany (2017). https://d-nb.info/1138577952

[9] Bundesamt für Bevölkerungsschutz und Katastrophenhilfe: Kritische Infrastrukturen. https://www.bbk.bund.de/DE/Themen/Kritische-Infrastrukturen/kritische-infrastrukturen_node.html

[10] Panajotovic, B., Janković, M., Odadzic, B.: Ict and smart grid (2011). doi:10.1109/TELSKS.2011.6112018

[11] Anwar, A., Mahmood, A.N.: Cyber Security of Smart Grid Infrastructure (2014). 1401.3936. http://arxiv.org/abs/1401.3936

[12] Fischer, L., Memmen, J.-M., Veith, E.M., Tröschel, M.: Adversarial Resilience Learning - Towards Systemic Vulnerability Analysis for Large and Complex Systems. arXiv (2018). doi:10.48550/ARXIV.1811.06447. https://arxiv.org/abs/1811.06447

[13] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41** (2009). doi:10.1145/1541880.1541882

[14] Shadmanov, I., Shadmanova, K.: Summarization of various security aspects and attacks in distributed systems: A review. Advances in Computer Science **5**, 35–39 (2016)

[15] Veith, E., Steinbach, B., Windeln, J.: A lightweight messaging protocol for smart grids. (2013). doi:10.13140/2.1.1219.0086

[16] shaker Ashoor, A., prof. Sharad Gore: Anomaly Detection Algorithm Using Multi-agents (2012). https://www.ijstr.org/final-print/april2012/Anomaly-Detection-Algorithm-Using-Multi-agents.pdf

[17] Prasanna, N.L., Sravanthi, K., Sudhakar, N.V.S.K.: Applications of graph labeling in communication networks. (2014)

[18] Deng, A., Hooi, B.: Graph neural network-based anomaly detection in multivariate time series (2021). doi:10.48550/ARXIV.2106.06947

[19] Frost, E.: Entwicklung eines Observers zur Anomalieerkennung in agentenbasierten cyber-physischen Energiesystemen (2021)

[20] Multi-Purpose Battery Storage Swarm (MIRAGE). https://www.offis.de/offis/projekt/mirage.html

[21] Handelman, G.S., Kok, H.K., Chandra, R.V., Razavi, A.H., Huang, S., Brooks, M., Lee, M.J., Asadi, H.: Peering Into the Black Box of Artificial Intelligence: Evaluation Metrics of Machine Learning Methods. PMID: 30332290 (2019). https://doi.org/10.2214/AJR.18.20224. doi:10.2214/AJR.18.20224. https://doi.org/10.2214/AJR.18.20224

[22] Baheti, R., Gill, H.: Cyber-physical systems. The impact of control technology **12**(1), 161–166 (2011)

[23] Jha, A.V., Appasani, B., Ghazali, A.N., Pattanayak, P., Gurjar, D.S., Kabalci, E., Mohanta, D.K.: Smart grid cyber-physical systems: communication technologies, standards and challenges. Wireless Networks **27**(4), 2595–2613 (2021). doi:10.1007/s11276-021-02579-1

[24] Zanero, S.: Cyber-Physical Systems (2017). doi:10.1109/MC.2017.105

[25] Yu, X., Xue, Y.: Smart grids: A cyber–physical systems perspective. Proceedings of the IEEE **104**(5), 1058–1070 (2016). doi:10.1109/JPROC.2015.2503119

[26] Wu, F.-J., Kao, Y.-F., Tseng, Y.-C.: From wireless sensor networks towards cyber physical systems. Pervasive and Mobile Computing **7**(4), 397–413 (2011). doi:10.1016/j.pmcj.2011.03.003

[27] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach., 3rd edn. PearsonEducation,Inc., Upper Saddle River, New Jersey 07458 (2010)

[28] Badjonski, M., Ivanović, M., Budimac, Z.: Agent Oriented Programming Language LASS. Woodhead Publishing (1999). doi:10.1533/9781782420613.111. https://www.sciencedirect.com/science/article/pii/B9781898563563500122

[29] Al-Agtash, S., Hafez, H.A.: Agents for smart power grids. Energy and Power Engineering **12**, 477–489 (2020). doi:10.4236/epe.2020.128029

[30] Müller-Schloer, C., Schmeck, H., Ungerer, T.: Organic computing. Informatik-Spektrum **35** (2012). doi:10.1007/s00287-012-0599-2

[31] Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., Schmeck, H.: Towards a generic observer/controller architecture for organic computing, pp. 112–119 (2006)

[32] Madhuri, G.S., Rani, D.M.U.: Anomaly detection Techniques (2018). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3167172

[33] Blázquez-García, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. https://arxiv.org/pdf/2002.04236.pdf (2020). 2002.04236

[34] Wuttke, L.: Machine Learning: Definition, Algorithmen, Methoden und Beispiele (2020). https://datasolut.com/was-ist-machine-learning

[35] Chollet, F.: Deep Learning with Python (2017)

[36] Cunningham, P., Cord, M., Delany, S.J.: In: Cord, M., Cunningham, P. (eds.) Supervised Learning, pp. 21–49. Springer, Berlin, Heidelberg (2008). doi:10.1007/978-3-540-75171-7_2

[37] Zhu, X., Goldberg, A.B.: Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning **3**(1), 1–130 (2009)

[38] Li, Y.: Deep Reinforcement Learning (2018). 1810.06339. http://arxiv.org/abs/1810.06339

[39] Nielsen, M.: Neural Networks and Deep Learning (2019)

[40] Xu, B., Wang, N., Chen, T., Li, M.: Empirical Evaluation of Rectified Activations in Convolutional Network. doi:10.48550/ARXIV.1505.00853. https://arxiv.org/abs/1505.00853

[41] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning (2016)

[42] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting (2014). https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[43] Omar, S., Ngadi, A., Jebur, H.H.: Machine learning techniques for anomaly detection: an overview. International Journal of Computer Applications **79**(2) (2013)

[44] Alla, S., Adari, S.K.: Beginning Anomaly Detection Using Python-based Deep Learning. Springer, (2019)

[45] Visa, S., Ramsay, B., Ralescu, A., Knaap, E.: Confusion matrix-based feature selection., vol. 710, pp. 120–127 (2011)

[46] Elmrabit, N., Zhou, F., Li, F., Zhou, H.: Evaluation of machine learning algorithms for anomaly detection. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1–8 (2020). doi:10.1109/CyberSecurity49315.2020.9138871

[47] Gaur, M., Makonin, S., Bajić, I.V., Majumdar, A.: Performance evaluation of techniques for identifying abnormal energy consumption in buildings. IEEE Access **7**, 62721–62733 (2019). doi:10.1109/ACCESS.2019.2915641

[48] Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: Losada, D.E., Fernández-Luna, J.M. (eds.) Advances in Information Retrieval, pp. 345–359. Springer, Berlin, Heidelberg (2005)

[49] Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proceedings of the 23rd International Conference on Machine Learning. ICML

'06, pp. 233–240. Association for Computing Machinery, New York, NY, USA (2006). doi:10.1145/1143844.1143874

[50] Chen, Z., Yeo, C.K., Lee, B.S., Lau, C.T.: Autoencoder-based network anomaly detection. In: 2018 Wireless Telecommunications Symposium (WTS), pp. 1–5 (2018). doi:10.1109/WTS.2018.8363930

[51] Xu, D., Wang, Y., Meng, Y., Zhang, Z.: An improved data anomaly detection method based on isolation forest. In: 2017 10th International Symposium on Computational Intelligence and Design (ISCID), vol. 2, pp. 287–291 (2017). doi:10.1109/ISCID.2017.202

[52] Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422 (2008). doi:10.1109/ICDM.2008.17

[53] Bottou, L., Lin, C.-J.: Support vector machine solvers. Large Scale Kernel Machines, 301–320 (2007)

[54] Ma, J., Perkins, S.: Time-series novelty detection using one-class support vector machines. In: Proceedings of the International Joint Conference on Neural Networks, 2003., vol. 3, pp. 1741–17453 (2003). doi:10.1109/IJCNN.2003.1223670

[55] Amer, M., Goldstein, M., Abdennadher, S.: Enhancing one-class support vector machines for unsupervised anomaly detection. ODD '13, pp. 8–15. Association for Computing Machinery, New York, NY, USA (2013). doi:10.1145/2500853.2500857

[56] Wang, Y., Wong, J., Miner, A.: Anomaly intrusion detection using one class svm. In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004., pp. 358–364 (2004). doi:10.1109/IAW.2004.1437839

[57] Ravale, U., Marathe, N., Padiya, P.: Feature selection based hybrid anomaly intrusion detection system using k means and rbf kernel function. Procedia Computer Science **45**, 428–435 (2015). doi:10.1016/j.procs.2015.03.174. International Conference on Advanced Computing Technologies and Applications (ICACTA)

[58] Yaacob, A.H., Tan, I.K.T., Chien, S.F., Tan, H.K.: Arima based network anomaly detection. In: 2010 Second International Conference on Communication Software and Networks, pp. 205–209 (2010). doi:10.1109/ICCSN.2010.55

[59] Parsai, S., Mahajan, S.: Anomaly detection using long short-term memory. In: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 333–337 (2020). doi:10.1109/ICESC48915.2020.9155897

[60] McArthur, S.D.J., Booth, C.D., McDonald, J.R., McFadyen, I.T.: An agent-based anomaly detection architecture for condition monitoring. IEEE Transactions on Power Systems **20**(4), 1675–1682 (2005). doi:10.1109/TPWRS.2005.857262

[61] Ahmed, T., Oreshkin, B., Coates, M.: Machine learning approaches to network anomaly detection. Proc. SysML (2007)

[62] Sakurada, M., Yairi, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. MLSDA'14, pp. 4–11. Association for Computing Machinery, New York, NY, USA (2014). doi:10.1145/2689746.2689747

[63] Maćkiewicz, A., Ratajczak, W.: Principal components analysis (pca). Computers & Geosciences **19**(3), 303–342 (1993). doi:10.1016/0098-3004(93)90090-R

[64] Li, K.-L., Huang, H.-K., Tian, S.-F., Xu, W.: Improving one-class svm for anomaly detection. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693), vol. 5, pp. 3077–30815 (2003). doi:10.1109/ICMLC.2003.1260106

[65] SAYADI, S., BEN REJEB, S., CHOUKAIR, Z.: Anomaly detection model over blockchain electronic transactions. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 895–900 (2019). doi:10.1109/IWCMC.2019.8766765

[66] Müller, A.C., Guido, S.: Einführung in Machine Learning Mit Python: Praxiswissen Data Science. O'Reilly, (2017)

[67] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C.: Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows. In: Loizides, F., Schmidt, B. (eds.) Positioning and Power in Academic Publishing: Players, Agents and Agendas, pp. 87–90 (2016). IOS Press

[68] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org (2015). https://www.tensorflow.org/

[69] Zhao, Y., Nasrullah, Z., Li, Z.: PyOD: A Python Toolbox for Scalable Outlier Detection (2019). http://jmlr.org/papers/v20/19-011.html

[70] Holly, S., Nieße, A.: Dynamic communication topologies for distributed heuristics in energy system optimization algorithms (2021). 2108.01380. https://arxiv.org/abs/2108.01380

[71] Colonel, J.T., Keene, S.: Conditioning Autoencoder Latent Spaces for Real-Time Timbre Interpolation and Synthesis. arXiv (2020). doi:10.48550/ARXIV.2001.11296. https://arxiv.org/abs/2001.11296

Danksagungen

Die Erstellung einer Masterarbeit erfordert viel Selbstdisziplin und Durchhaltevermögen. Auf diesem Weg haben mich viele liebe Menschen unterstützt, denen ich an dieser Stelle danken möchte.

Als erstes möchte ich Prof. Dr.-Ing. Astrid Nieße für die Beurteilung dieser Masterarbeit als Erstprüferin sowie für die positiven Lehrerfahrungen in den vergangenen Semestern danken.

Besonderen Dank verdient Emilie Frost, da sie mich während der gesamten Zeit hervorragend betreut hat und neben fachlichen Diskussionen immer Zeit für persönliche Worte blieb.

Außerdem möchte ich Torge Wolff für die Betreuung während der Vorarbeiten sowie für die Unterstützung bei der Abschlusspräsentation danken. Ebenso danke ich Martin Tröschel für die Unterstützung und die motivierenden Gespräche.

Meine Familie und meine Freunde haben mich und meine teilweise etwas durch Stress getrübte Launen mit stets aufbauenden Worten über insgesamt sechs Jahre durch zwei Studiengänge getragen, wofür ich jedem von ihnen sehr dankbar bin.

Erklärung

Ich versichere an Eides statt, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichungen, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

J. Heiken

Oldenburg, den 27. Oktober 2022 — Julia Catharina Heiken