JOHANNES KEPLER
UNIVERSITÄT LINZ

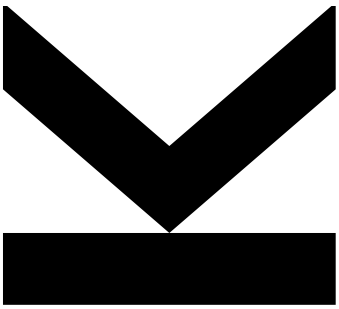
Eingereicht von
Mara Vukadinovic, BSc

Angefertigt am
Institut für Wirtschaftsinformatik - Data & Knowledge Engineering

Betreuer
Assoz.-Prof. Mag. Dr. Christoph Schütz

November 2024

Daten-Monitoring für Roboteranwendungen: Vergleich regelbasierter Methoden mit Methoden des maschinellen Lernens für die Anomalieerkennung



Masterarbeit

zur Erlangung des akademischen Grades

Master of Science

im Masterstudium

Wirtschaftsinformatik

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at

Danksagung

Diese Arbeit wurde vom österreichischen Bundesministerium für Klimapolitik, Umwelt, Energie, Mobilität, Innovation und Technologie im Rahmen des Projekts „RunRoc“ unterstützt.

Kurzfassung

Der Einsatz von Robotersystemen hat in den letzten Jahren in verschiedenen Anwendungsbereichen zunehmend an Bedeutung gewonnen. Dies erfordert jedoch eine genaue Überwachung und Speicherung der von den Robotern erzeugten Daten. Um diesem Bedarf gerecht zu werden, wird ein Monitoringsystem vorgeschlagen, das abnormale Situationen bei Robotern identifiziert und meldet. In der Forschung wird zwischen regelbasierten und maschinellen Lernmethoden unterschieden. Regelbasierte Ansätze verwenden vordefinierte Regeln, um Abweichungen vom erwarteten Verhalten zu erkennen. Im Gegensatz dazu erwerben maschinelle Lernalgorithmen die Fähigkeit, selbstständig Muster zu erlernen. In dieser Arbeit werden beide Ansätze untersucht, um festzustellen, ob regelbasierte Ansätze durch maschinelles Lernen ersetzt werden können. Die Bewertung der beiden Ansätze erfolgt anhand von drei herbeigeführten Anomalien. Die Auswertungen zeigen, dass der regelbasierte Ansatz deutlich bessere Ergebnisse erzielt als die maschinellen Lernmodelle. Die Studie zeigt aber auch, dass eine Kombination beider Ansätze die Schwächen des jeweils anderen überwinden könnte.

Abstract

In recent years, the use of robotic applications has become increasingly important in various industries. However, this development requires precise monitoring and storage of the data generated by the robotic systems. To address this need, a monitoring system is proposed that detects and reports abnormal situations in robots. In this context, the researchers differentiate between rule-based and machine-learning methods. Rule-based approaches use predefined rules to detect deviations from expected behavior. In contrast, machine-learning algorithms acquire the ability to learn patterns on their own. In this thesis, both approaches are evaluated to determine whether rule-based methods can be replaced by machine-learning algorithms. The evaluation of the two methods is based on three induced anomalies. The results show that the rule-based approach achieves significantly better results compared to the machine learning models. However, the study also reveals that a combination of both approaches could overcome the weaknesses of the other.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1

Einleitung

Die Verwendung von Robotersystemen hat in diversen Anwendungsbereichen wie der Industrie, Landwirtschaft, Infrastruktur und im alltäglichen Leben in den letzten Jahren zunehmend an Bedeutung gewonnen (Matthias, 2017). In der Industrie werden Roboter eingesetzt, um Produktionsprozesse zu optimieren und die Qualität der hergestellten Produkte zu verbessern. Roboter übernehmen dabei oft routinemäßige und wiederkehrende Tätigkeiten wie Schweißen, Lackieren, Kommissionieren und Platzieren von Gütern (De Backer et al., 2018). In der Landwirtschaft unterstützen Roboter bei der Aussaat, Ernte und Pflege von Pflanzen, wodurch Kosten eingespart und Ernteverluste reduziert werden können (Mahmud et al., 2020). Zudem werden Robotersysteme in der Infrastrukturinspektion eingesetzt, um Gebäude, Brücken, Straßen und Tunnel zu inspizieren und potenzielle Probleme frühzeitig zu erkennen (Halder & Afsari, 2023).

Die zunehmende Flexibilität von Robotersystemen ermöglicht es diesen, in einer Vielzahl von Umgebungen und Situationen eingesetzt zu werden. Dies erfordert jedoch eine präzise Überwachung und Speicherung der von den Robotern generierten Daten. In diesem Zusammenhang spielt das Datenmonitoring eine entscheidende Rolle (Graabæk et al., 2023). Das Datenmonitoring umfasst das Sammeln und Analysieren von Daten aus verschiedenen Quellen. Die aus der Datenanalyse gewonnenen Erkenntnisse helfen dabei, relevante Muster, Trends und Chancen zu erfassen. Darüber hinaus ermöglicht die Datenüberwachung eine frühzeitige Erkennung von potenziellen Problemen oder Optimierungsbedarfe (Mudliar, 2023).

Das kontinuierliche Protokollieren von Daten birgt jedoch potenzielle Probleme. Mit der zunehmenden Anzahl aufgezeichneter Daten steigt der Bedarf an Speicherplatz und Ressourcen für die Datenverarbeitung. Dies kann zu höheren Kosten und einer verminderten Systemleistung führen (Khan et al., 2014). Des Weiteren sind nicht alle generierten Daten für die Analyse von Wert oder Notwendigkeit. Eine vollständige Protokollierung könnte dazu führen, dass irrelevante Informationen aufgezeichnet werden, die später nicht verwendet werden können. Darüber hinaus kann eine große Datenmenge zu längeren Analysezeiten und Schwierigkeiten bei der Identifikation wesentlicher Informationen führen (Sagiroglu & Sinanc, 2013).

Auf dem Gebiet des Datenmonitorings von Robotersystemen haben Forscher verschiedene Methoden untersucht, um dieses Problem zu lösen. Ein Schwerpunkt liegt auf regelbasierten Ansätzen, bei denen vordefinierte Regeln verwendet werden, um Abweichungen vom erwarteten Verhalten zu detektieren. Diese Regeln sind typischerweise strukturiert, jedoch nicht ausschließlich auf „Wenn-

dann“-Anweisungen beschränkt. Vielmehr lösen spezifische Bedingungen entsprechende Aktionen aus (Liu et al., 2014).

Regelbasierte Ansätze sind jedoch darauf begrenzt, nur diejenigen Abweichungen zu erfassen, die innerhalb des Regelsatzes vordefiniert wurden. Zudem ist die Umsetzung regelbasierter Methoden in der Praxis oft schwierig, da sie detailliertes vorausgehendes Wissen über mögliche Fehler und Störungen erfordern (Graabæk et al., 2023). Eine mögliche Lösung in diesem Zusammenhang sind Ansätze, die maschinelles Lernen verwenden. Durch das Training mit historischen Daten erlangen diese Algorithmen die Fähigkeit, selbstständig Muster zu erkennen. Dies ermöglicht die Erkennung von Anomalien auf der Grundlage des bisherigen Verhaltens des Systems (Bonaccorso, 2017).

Ziel dieser Arbeit ist es, sowohl regelbasierte Ansätze als auch maschinelles Lernen im Kontext des Datenmonitorings von Roboteranwendungen zu untersuchen. In dieser Hinsicht soll bewertet werden, ob maschinelle Lerntechniken die Einschränkungen regelbasierter Systeme überwinden und diese möglicherweise ablösen können. Dementsprechend lautet die Forschungsfrage wie folgt:

Inwieweit können Machine-Learning-basierte Ansätze im Datenmonitoring für Roboterapplikationen als Alternative zu regelbasierten Ansätzen eingesetzt werden?

Zur Beantwortung der Forschungsfrage wird Design Science Research (DSR) als methodischer Rahmen verwendet. Dieser Ansatz ermöglicht es, innovative Lösungen systematisch zu entwickeln und deren praktische Anwendbarkeit zu evaluieren (Dresch et al., 2015).

Im ersten Teil dieser Arbeit wird der Hintergrund erläutert, indem das Datenmonitoring für Roboteranwendungen sowie die Erkennung von Anomalien vorgestellt werden. Dies beinhaltet eine Definition und Kategorisierung von Anomalien sowie eine Übersicht über die Techniken zur Anomalieerkennung. Anschließend wird der aktuelle Stand der Forschung in Bezug auf regelbasierte und maschinelle Ansätze beleuchtet. Im zweiten Teil der Arbeit wird der Anwendungsfall beschrieben. Hierbei wird näher auf den verwendeten Roboter, die generierten Daten sowie die herbeigeführten Anomalien eingegangen. In den zwei darauffolgenden Kapiteln werden der regelbasierte und der Machine-Learning-Ansatz detailliert beschrieben, wobei auf die eingesetzten Technologien, die Implementierung und die Evaluierung der Ansätze eingegangen wird. Schließlich werden in der Diskussion die Ergebnisse interpretiert und mit der vorhandenen Literatur verglichen. Die Arbeit schließt mit einem Fazit, welches die wichtigsten Ergebnisse zusammenfasst, die Forschungsfrage beantwortet und einen Ausblick auf zukünftige Forschungsmöglichkeiten gibt.

2

Hintergrund

In diesem Kapitel werden die grundlegenden Prinzipien und Konzepte dargelegt, die für das weitere Verständnis dieser Arbeit von Bedeutung sind. Zunächst wird der Begriff „Roboter“ definiert, um ein einheitliches Bild zu schaffen. Anschließend wird näher auf das Datenmonitoring von Roboteranwendungen eingegangen. Des Weiteren wird auf die Anomalie eingegangen, was Definitionen, Arten von Anomalien und gängige Techniken zur Erkennung von Abweichungen beinhaltet. Abschließend wird der aktuelle Forschungsstand dargelegt, wobei sowohl regelbasierte als auch maschinelle Ansätze zur Anomalieerkennung betrachtet werden.

2.1 Definition eines Roboters

Der Begriff Roboter stammt vom tschechischen Wort „robota“, was so viel wie „Zwangsarbeit“ oder „Leibeigener“ bedeutet. Dieser Begriff wurde im Jahr 1921 von dem tschechischen Dramatiker Karel Čapek eingeführt. Dieser schrieb das dazugehörige Stück „Rossum's Universal Robots“ (R.U.R.). Laut Čapek ist ein Roboter eine Maschine, die dem Menschen in ihrem äußeren Erscheinungsbild ähnelt (Hunt, 2012).

Heutzutage gibt es unterschiedliche Definitionen für den Begriff Roboter. Laut dem Oxford English Dictionary ist ein Roboter eine Maschine, die in der Lage ist, eine komplexe Reihe von Handlungen automatisch auszuführen, insbesondere eine, die von einem Computer programmierbar ist, und somit eine automatische Maschine darstellt (Press, 1999). Nach der Definition der Internationalen Organisation für Normung (ISO) ist ein Roboter ein programmierter, betätigter Mechanismus mit einem gewissen Grad an Autonomie zur Durchführung von Fortbewegung, Manipulation oder Positionierung (IFR, 2021).

2.2 Datenmonitoring für Roboteranwendungen

Das Datenmonitoring für Roboteranwendungen stellt ein komplexes, jedoch etabliertes Problem in der Softwaretechnik dar und umfasst eine Vielzahl von Ansätzen mit unterschiedlichen Schwerpunkten und Merkmalen (Lotz et al., 2011). Das Ziel ist es, den Zustand und das Verhalten von Robotersystemen zu überwachen, um eine hohe Leistung, Sicherheit und Zuverlässigkeit zu gewährleisten. Einige der zentralsten Ansätze und Technologien für die Datenüberwachung in der Robotik werden im Folgenden beschrieben.

2.2.1 Logging

Logging ist eine Methode, bei der Daten im Voraus gesammelt und später analysiert werden. Dies ermöglicht eine detaillierte Nachverfolgung von Ereignissen und Fehlern. Diese Methode hat jedoch einige Nachteile. Zum einen erfordert sie die frühzeitige Erfassung aller potenziell relevanten Daten, was zu einem hohen Daten-Overhead und negativen Auswirkungen auf das Laufzeitverhalten des Systems führen kann. Zum anderen erfolgt die Datenanalyse ausschließlich im Offline-Modus, was den Entwicklungs- und Testzyklus verlangsamt und die Identifizierung von Problemen in Echtzeit erschwert (Lotz et al., 2011).

2.2.2 Runtime Monitoring

Im Gegensatz zum Logging konzentriert sich das Runtime Monitoring auf die Überwachung und Verarbeitung von Daten in Echtzeit. Dieser Ansatz ermöglicht eine sofortige Reaktion auf Anomalien oder Fehler, was für die Betriebssicherheit und Leistungsoptimierung von Robotersystemen entscheidend ist. Ein Beispiel hierfür ist das Component Based Software Engineering (CBSE). Das CBSE bietet eine strukturierte Herangehensweise an das Monitoring von Softwarekomponenten. Das System ermöglicht es zudem, einzelne Komponenten unabhängig zu überwachen und zu diagnostizieren, was die Flexibilität und Reaktionsgeschwindigkeit erhöht (Lotz et al., 2011).

2.2.3 Robotik-Middleware-Systeme

Mehrere Robotik-Middleware-Systeme wie ROS (Robot Operating System) (»rospy - ROS Wiki«, 2017), Orocos (Bruyninckx, 2001) und R'I-Middleware (Ando et al., 2006) bieten verschiedene Lösungen für das Datenmonitoring von Robotersystemen an. ROS stellt nützliche Werkzeuge wie RViz und rostopic bereit. Diese sind jedoch auf das Publish/Subscribe-Kommunikationsmodell beschränkt. Orocos hingegen dient der Erstellung von Echtzeit-Robotikanwendungen mit modularen, laufzeitkonfigurierbaren Softwarekomponenten. Die R'I-Middleware verfügt über ein klares Komponentenmodell, das jedoch hauptsächlich für Entwicklungszwecke gedacht ist.

2.2.4 Webbasierte Tools

Microsoft Robotics Studio ist ein Beispiel für ein System, das die Analyse von übermittelten Daten über einen Webbrowser ermöglicht. Diese Tools stellen semantische Informationen über SOAP bereit, was die Laufzeitanalyse erleichtert. Ihre Nutzung ist jedoch auf .NET-Anwendungen beschränkt. Zudem kann der hohe Kommunikationsaufwand von SOAP in ressourcenbeschränkten Robotikanwendungen problematisch werden (Jackson, 2007).

Das Datenmonitoring in Roboteranwendungen ist ein vielschichtiges Feld, das verschiedene Technologien und Ansätze umfasst. Während Logging detaillierte historische Analysen ermöglicht, bietet das Runtime Monitoring Echtzeitüberwachung an. Auch die Middleware-Systeme und webbasierte Tools sind für bestimmte Aspekte der Überwachung nützlich.

2.3 Anomalien

Eine Anomalie bezeichnet eine Abweichung oder ein Muster, das sich wesentlich von den erwarteten oder als normal betrachteten Mustern unterscheidet (Chandola et al., 2009). Hawkins beschreibt in

diesem Zusammenhang eine Anomalie als eine Beobachtung, die sich derart deutlich von anderen Beobachtungen unterscheidet, dass die Vermutung nahe liegt, sie könnte von einem alternativen Mechanismus erzeugt worden sein (Hawkins, 1980). Alternativ definierten Barnett und Lewis eine Anomalie als „Ein Ausreißer ist eine Beobachtung (oder eine Teilmenge von Beobachtungen), die scheinbar nicht mit dem Rest dieser Datensammlung übereinstimmt."(Barnett, Lewis et al., 1994). Die Identifikation solcher abweichenden Muster in Daten, die nicht mit dem erwarteten Verhalten übereinstimmen, ist ein zentraler Aspekt der Anomalieerkennung. Die Anomalieerkennung, auch als Ausreißererkennung oder Ereigniserkennung bezeichnet, ist der Prozess, ungewöhnliche oder neuartige Beobachtungen innerhalb der erfassten Daten zu identifizieren (Chandola et al., 2007). Die Erkennung von Anomalien findet in einer breiten Palette von Anwendungen ihren Einsatz, etwa bei der Aufdeckung von Betrugsfällen im Bereich von Kreditkarten und Versicherungen oder im Gesundheitswesen. Ebenso dient sie der Erkennung von Eindringlingen im Bereich der Cybersicherheit, der Fehleridentifikation in sicherheitskritischen Systemen sowie der Überwachung feindlicher Aktivitäten im militärischen Kontext (Chandola et al., 2009).

2.3.1 Arten von Anomalien

Die genaue Bestimmung der Art der Anomalie ist entscheidend, um anschließend den am besten geeigneten Algorithmus für ihre Erkennung auszuwählen. Anomalien können in die folgenden drei Kategorien eingeteilt werden (Chandola et al., 2009).

Punktuelle Anomalie

Wenn eine bestimmte Dateninstanz vom normalen Muster des Datensatzes abweicht, kann sie als eine punktuelle Anomalie betrachtet werden (Ahmed et al., 2016). Diese Form der Anomalie ist die grundlegendste und bildet den Fokus vieler Forschungsarbeiten zur Anomalieerkennung. Ein praktisches Anwendungsbeispiel hierfür ist die Erkennung von Kreditkartenbetrug. Dabei repräsentiert der Datensatz die Kreditkartentransaktionen einer Person, wobei davon ausgegangen wird, dass die Daten durch ein einziges Merkmal, den Betrag der Transaktionen, definiert werden. Eine Transaktion, bei der der Betrag im Vergleich zum normalen Ausgabenbereich dieser Person sehr hoch ist, stellt eine Punktanomalie dar (Chandola et al., 2009).

Visuell wird diese Art von Anomalie durch einen einzigen Punkt in Abbildung 2.1 dargestellt. Dieser Punkt weicht erheblich von der üblichen Datenverteilung im zweidimensionalen Raum ab.

Kontextuelle Anomalie

Wenn das Verhalten einer Dateninstanz nur in einem bestimmten Kontext als abweichend betrachtet wird, nicht aber in anderen Kontexten, spricht man von einer kontextuellen Anomalie (Chandola et al., 2009). So sind beispielsweise die Ausgaben während der Festtage wie Weihnachten oder Silvester in der Regel höher als während des restlichen Jahres. Auch wenn die Ausgaben in dieser Zeit hoch sind, können sie nicht als anomal angesehen werden, da sie in diesem speziellen Kontext als normal gelten. Im Gegensatz dazu könnte eine genauso hohe Ausgabe in einem Nicht-Festtagsmonat als kontextuelle Anomalie betrachtet werden (Ahmed et al., 2016).

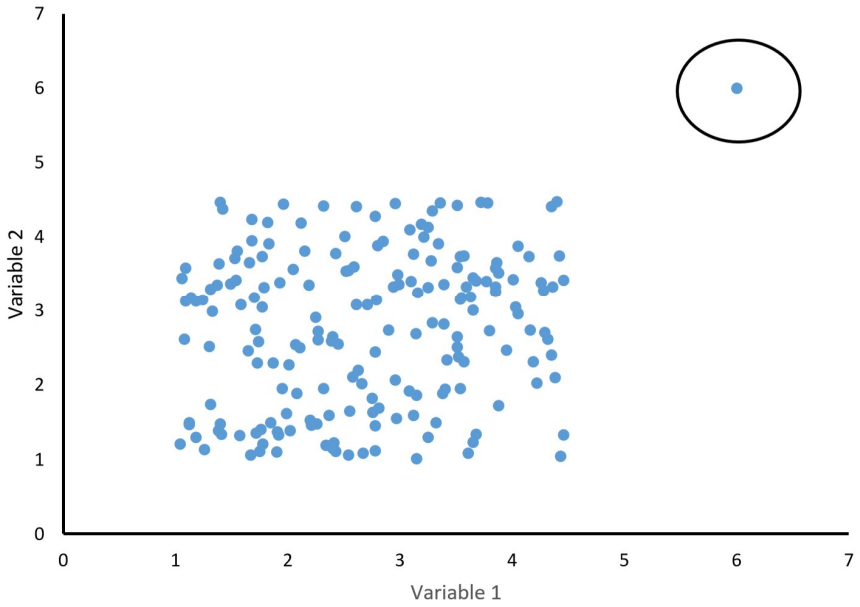


Abbildung 2.1: Punktuelle Anomalie im zweidimensionalen Raum

Abbildung 2.2 veranschaulicht die durchschnittlichen Temperaturdaten der einzelnen Monate. Die Werte t1 und t2 weisen eine ähnliche Ausprägung auf. Allerdings wurde die außergewöhnlich niedrige Temperatur t2 im Sommer gemessen, was im Kontext der zu erwartenden wärmeren Temperaturen in dieser Jahreszeit ungewöhnlich ist.

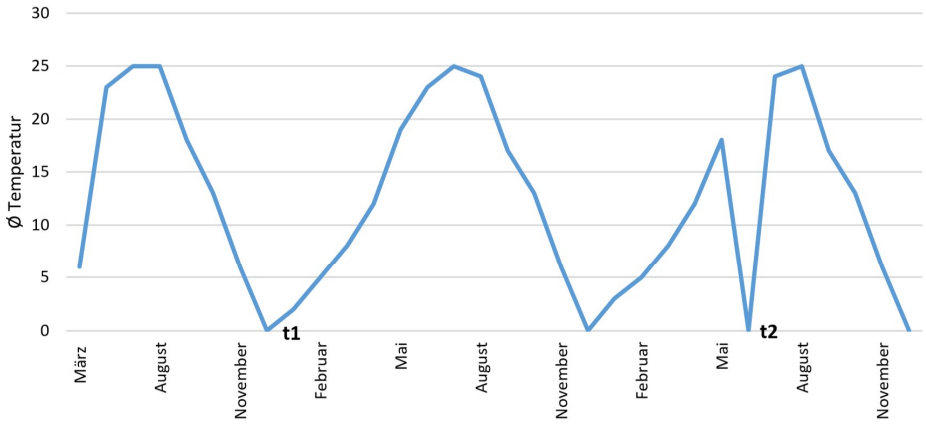


Abbildung 2.2: Kontextuelle Anomalie der Temperatur im Sommer

Kollektive Anomalie

Wenn eine Sammlung von zusammenhängenden Dateninstanzen im Vergleich zum gesamten Datensatz abweicht, wird sie als kollektive Anomalie bezeichnet. Dabei müssen die einzelnen Dateninstanzen in einer kollektiven Anomalie für sich genommen keine Anomalien darstellen, aber ihr gemeinsames Auftreten als Sammlung wird als anomal betrachtet (Chandola et al., 2009). Zum Beispiel kann das kontinuierliche Vorhandensein niedriger Werte in einem menschlichen Elektrokardiogramm (EKG) auf ein zugrunde liegendes Phänomen hindeuten, das einer abnormalen vorzeitigen Kontraktion entspricht (Lin et al., 2005). Allerdings wird ein einzelner niedriger Wert für sich allein genommen nicht als Anomalie betrachtet (Ahmed et al., 2016).

Die nachfolgende Darstellung illustriert diese Beispiel. In Abbildung 2.3 ist zu erkennen, dass das Signal über einen ungewöhnlich langen Zeitraum stabil bleibt (etwa zwischen t=1100 und t=1400). Um diese Anomalie zu identifizieren, ist es erforderlich, alle Werte dieses Intervalls zu berücksichtigen. Weder der Wert noch der Kontext der einzelnen Werte sind ungewöhnlich, sondern die Wiederholung innerhalb dieses Zeitraums (Chandola et al., 2009).

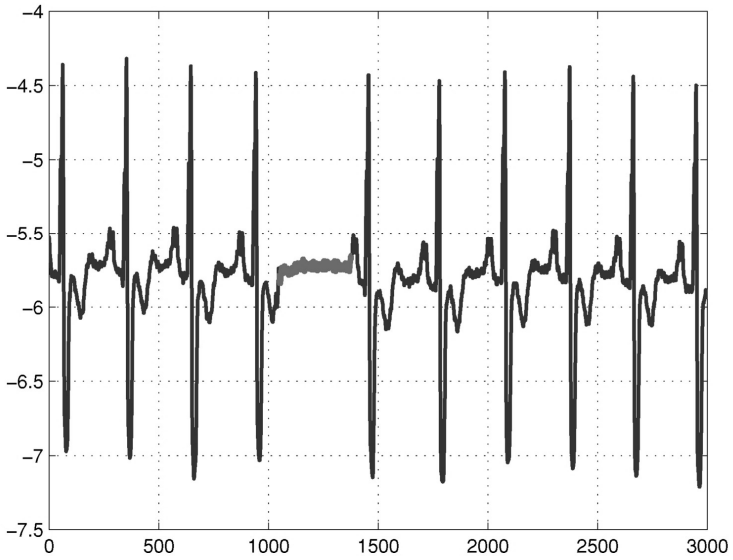


Abbildung 2.3: Kollektive Anomalie am Beispiel eines Elektrokardiogramms (Chandola et al., 2009)

2.3.2 Techniken zur Erkennung von Anomalien

Die Techniken zur Erkennung von Anomalien lassen sich je nach der Verfügbarkeit von Labels unterteilen. Das Label, das einer Dateninstanz zugeordnet ist, bestimmt, ob diese Instanz als normal oder anomal klassifiziert wird. Die Beschaffung präzise gelabelter Daten, die alle relevanten Verhaltensweisen umfassend abdecken, ist häufig sehr kostenintensiv. Der Prozess des Labelns erfolgt meist manuell durch Experten, was sowohl zeitaufwendig als auch ressourcenintensiv ist. Diese Anforderungen machen die Erstellung eines vollständigen und repräsentativen Trainingsdatensatzes zu einer erheblichen Herausforderung (Chandola et al., 2009).

Überwachte Anomalieerkennung

Die überwachte Anomalieerkennung (Supervised Anomaly Detection) ist eine Technik, bei der ein Modell auf der Grundlage eines vollständig beschrifteten Trainingsdatensatzes trainiert wird. Dieser Datensatz enthält sowohl normale als auch anomale Instanzen. Das Modell lernt, typische Merkmale der beiden Klassen zu erkennen und kann anschließend neue, ungesehene Dateninstanzen klassifizieren. Diese Methode setzt voraus, dass Anomalien präzise dokumentiert und in ausreichender Menge vorhanden sind (Chandola et al., 2009).

Semi-überwachte Anomalieerkennung

In der semi-überwachten Anomalieerkennung wird davon ausgegangen, dass der Trainingsdatensatz teilweise gelabelt ist und somit sowohl gelabelte als auch ungelabelte Daten enthält. Solche Algorithmen sind besonders wertvoll, wenn die vollständige Annotation des Datensatzes unverhältnismäßig teuer ist. Einerseits erfordert die Beschriftung der Daten oft spezielle Fachkenntnisse. Andererseits

kann die Erkennung von Anomalien in bestimmten Bereichen wie der Industrie oder der Biomedizin kostspielig sein (Ghamry et al., 2024).

Unüberwachte Anomalieerkennung

Unüberwachte Anomalieerkennungstechniken benötigen keine gelabelten Trainingsdaten und sind daher besonders vielseitig einsetzbar. Diese Methoden basieren auf der impliziten Annahme, dass normale Instanzen in den Testdaten erheblich häufiger vorkommen als Anomalien. Wenn diese Annahme nicht zutrifft, kann es zu einer hohen Fehlalarmrate kommen (Chandola et al., 2009).

2.4 Aktueller Stand der Forschung

Auf dem Gebiet des Datenmonitorings von Robotersystemen haben Forscher verschiedene Methoden untersucht, um zuverlässige und effektive Ansätze zur Anomalieerkennung zu entwickeln (Khalastchi & Kalech, 2018). Ein Schwerpunkt liegt auf regelbasierten Ansätzen, die gleichzeitig als wissensbasierte Systeme betrachtet werden. Diese Ansätze verwenden laut Akerkar et. al. (2009) vordefinierte Regeln, um Abweichungen vom erwarteten Systemverhalten zu erkennen. Hierbei ahmt der regelbasierte Ansatz das Verhalten menschlicher Experten nach, um die Symptome des Systems rasch mit Diagnosen verknüpfen zu können. Expertensysteme, die diesen regelbasierten Prinzipien folgen, verwenden typischerweise WENN-DANN-Regeln, um die Roboteranwendung abzubilden. Diese Regeln werden entweder aus grundlegenden Prinzipien oder der strukturellen Beschreibung des Robotersystems abgeleitet. Ein wesentlicher Nachteil dieses Ansatzes ist die Schwierigkeit, unbekannte Anomalien zu erkennen. Dies ist darauf zurückzuführen, dass es herausfordernd ist, neue Diagnosen aus einem festen Satz von Regeln abzuleiten (Akerkar & Sajja, 2009). Ein bemerkenswertes Beispiel ist das von Fantuzzi et al. (2003) entwickelte Expertensystem. Dieses System identifiziert und isoliert bekannte Fehler durch die Anwendung einfacher logischer Regeln. Anstatt auf ein komplexes Roboter-Dynamikmodell zurückzugreifen, konzentriert sich die Methode auf die Analyse potenzieller Fehlerquellen in den elektrischen und mechanischen Komponenten des Roboters sowie in seinem Steuerungssystem (Fantuzzi et al., 2003).

Des Weiteren kommen beim Datenmonitoring modellbasierte Ansätze zum Einsatz. Diese verwenden mathematische Modelle oder logische Systeme zur Beschreibung des Verhaltens von Systemen und deren Komponenten. Ein Beispiel hierfür ist von De Luca und Mattone (2005) entwickelte Methode. Sie verwenden ein dynamisches Modell des Roboters zusammen mit einem linearen Zustandsschätzer, um unbekannte Aktuatorfehler zu erkennen. Hierbei wird die Abweichung zwischen den Modellvorhersagen und den tatsächlichen Messwerten, den sogenannten Residuen, analysiert. Durch den Einsatz eines Tiefpassfilters zur Reduzierung von Datenrauschen und einer niedrigen Erkennungsschwelle wird eine hohe Sensitivität erreicht und Fehlalarme minimiert (De Luca & Mattone, 2005). Ein weiterer Ansatz, vorgestellt von Stavrou et al. (2016), bezieht sich auf die modellbasierte Erkennung und Identifizierung von Aktuatorfehlern bei mobilen Robotern mit Differentialantrieb in Innenräumen. Das Verfahren berechnet eine Fehlergrenze zwischen dem geschätzten und dem gemessenen Roboterzustand, die kontinuierlich auf Basis des aktuellen Zustands und der Eingangssignale angepasst wird. Ein Fehler wird erkannt, wenn der Schätzungsfehler diese Grenze überschreitet (Stavrou et al., 2016).

Ein wesentliches Problem bei wissensbasierten und modellbasierten Ansätzen besteht darin, dass

sie auf im Voraus definiertem Wissen oder einem ausreichend genauen analytischen Modell basieren müssen. Im Gegensatz dazu sind datengetriebene bzw. maschinelle Ansätze modellfrei. Sie ermöglichen es, unbekannte Fehler zu erkennen, da das Systemmodell durch Training erlernt und kontinuierlich angepasst wird. Diese datengetriebenen Methoden nutzen gesammelte Daten, um wertvolle Informationen für die Anomalieerkennung und -diagnose zu gewinnen. Obwohl die Trainingsphase oft rechenintensiv ist, gewinnen datengetriebene Ansätze aufgrund der sinkenden Kosten für Rechenleistung und der kontinuierlichen Fortschritte im Bereich des maschinellen Lernens zunehmend an Beliebtheit (Khalastchi & Kalech, 2018). Zu diesen Ansätzen gehören dichtebasierte Methoden wie K-Nearest Neighbors (KNN) (Angiulli & Pizzuti, 2002) und distanzbasierte Methoden wie die Mahalanobis-Distanz (Khalastchi et al., 2015). Diese können zwar effektiv sein, aber sind bei der Verarbeitung hochdimensionaler Daten durch Zeit- und Rechenaufwand eingeschränkt. Auch rekonstruktionsbasierte Ansätze wie Autoencoder und Variational Autoencoder, die latente Merkmale normaler Daten erlernen und rekonstruieren, werden häufig zur Anomalieerkennung eingesetzt (Hinton & Salakhutdinov, 2006).

In einer aktuellen Studie führten Graabæk et. al. (2023) einen umfassenden experimentellen Vergleich praktischer Methoden zur Anomalieerkennung durch. Sie evaluierten 15 Anomalieerkennungsmethoden in einer typischen Pick-and-Place-Anwendung für einen UR5e Roboterarm. Die Anomalien umfassen das Drücken des Roboters an den Gelenken, das Anwenden eines Wrenches auf das TCP, das Einführen eines fremden Objekts bei der Ablage, das Sampling außerhalb des gewohnten 20 cm $\times$ 20 cm Quadrats, das Ersetzen des Zylinders mit unterschiedlichem Gewicht, das Fallen lassen des Zylinders, das Reduzieren der Geschwindigkeit um 50% sowie das Verlängern der Wartezeit auf bis zu 20 Sekunden. Die Untersuchung umfasste 600 Durchläufe im Normalbetrieb sowie 80 Durchläufe mit exogenen Störungen. Dabei wurden Daten zu den Gelenken des Roboters, den Kraft-Drehmoment-Messungen und der Zeit, die in jedem Programmknoten verbracht wurde, erfasst. Auf Basis der Daten aus dem Normalbetrieb wurden Anomalieerkennungsverfahren entwickelt, die für ressourcenbeschränkte Roboterhardware optimiert sind. Diese Methoden wurden anschließend mit den Daten aus den gestörten Durchläufen evaluiert, wobei mehrere Verfahren eine hohe Anomalieerkennungsleistung zeigten. Der Vergleich der Zeitreihenmethoden offenbarte, dass die LSTM-Autoencoder-Struktur die höchste Bewertung erzielt und bei der Hälfte der bewerteten Anomalien unter den besten drei liegt. Zudem wurde festgestellt, dass die Berücksichtigung des Programmbaums des Roboters die Leistung bei bestimmten Anomalietypen verbessern kann. Gleichzeitig zeigte sich, dass die Leistung generell abnahm, wenn die Anwendung selten besuchte Programmzweige, physischen Kontakt mit der Umgebung oder stochastische Trajektorien beinhaltete (Graabæk et al., 2023).

Park et al. (2018) setzten einen PR2-Humanoidenroboter für eine roboterassistierte Fütterungsaufgabe ein. Die Autoren entwickelten einen Echtzeit-Anomaliedetektor, der in der Lage ist, 12 verschiedene Arten exogener Anomalien zu erkennen. Diese beinhalten die Berührung durch den Benutzer, aggressives Essen, Kollision von Utensilien durch den Benutzer, Geräusche vom Benutzer, Gesichtsverdeckung, Fehlen von Utensilien durch den Benutzer, unerreichbarer Ort, Kollision mit der Umgebung, Umgebungsgeräusche, Fehlen von Utensilien durch Systemfehler, Kollision von Utensilien durch Systemfehler und Einfrieren des Systems. Die Autoren stellten dazu einen auf Long Short-Term Memory basierenden Variational Autoencoder (LSTM-VAE) vor, der Signale kombiniert und deren erwartete Verteilung mithilfe einer fortschrittsbasierten variierenden Priorverteilung

rekonstruiert. Der LSTM-VAE-basierte Detektor meldet eine Anomalie, wenn ein auf Rekonstruktion basierender Anomalie-Score einen Schwellenwert überschreitet. In Evaluierungen mit 1555 roboterassistierten Fütterungsvorgängen, einschließlich der 12 repräsentativen Anomalietypen, zeigte der Detektor eine höhere Fläche unter der Receiver Operating Characteristic-Kurve (0.8710) im Vergleich zu fünf anderen Basisdetektoren aus der Literatur (Park et al., 2018).

Im Großen und Ganzen scheinen die Ergebnisse des maschinellen Lernansatzes vielversprechend zu sein. Es wurden zahlreiche Methoden zur Erkennung von Anomalien bei der Roboterapplikation vorgeschlagen, aber ihre Anwendbarkeit und Leistung in realen Szenarien ist oft nicht bewiesen.

3

Anwendungsfall

Dieses Kapitel beschreibt den Anwendungsfall, welcher mithilfe des Datenmonitorings überwacht werden soll. Es umfasst die Beschreibung des verwendeten Roboters und Montageprozesses sowie die Datenaufnahme über die Real-Time Data Exchange (RTDE)-Schnittstelle (»Real-Time Data Exchange (RTDE) Guide«, 2019). Darüber hinaus werden die herbeigeführten Anomalien erläutert. Diese beinhalten zusätzliches Gewicht, das Fallen lassen eines Objekts und die Reduzierung der Robotergeschwindigkeit.

3.1 Beschreibung

Der zu überwachende Anwendungsfall umfasst die Montage einer Seilführungsscheibe. Diese besteht aus folgenden drei Komponenten: zwei Platten und einem zentralen Kunststoffring. Nachstehende Abbildung veranschaulicht die Anordnung dieser drei Komponenten.

Abbildung 3.1: Zusammensetzung der Seilführungsscheibe

Die Konstruktion der Seilführungsscheibe erfolgt mit einem UR10e-Manipulator (»universal-robots.com«, 2024) von Universal Robots. Dieser Roboter ist in Abbildung 3.2 ersichtlich. Der UR10e verfügt über sechs Gelenke mit einem Bewegungsbereich von $\pm 360°$ und einer maximalen Gelenkgeschwindigkeit von $120°/s$. Mit einer Reichweite von 1300 mm und einer Positionswiederholgenauigkeit von $\pm 0{,}05$ mm ist dieser Roboter für zahlreiche hochpräzise und schwere Aufgaben geeignet (»universal-robots.com«, 2024).

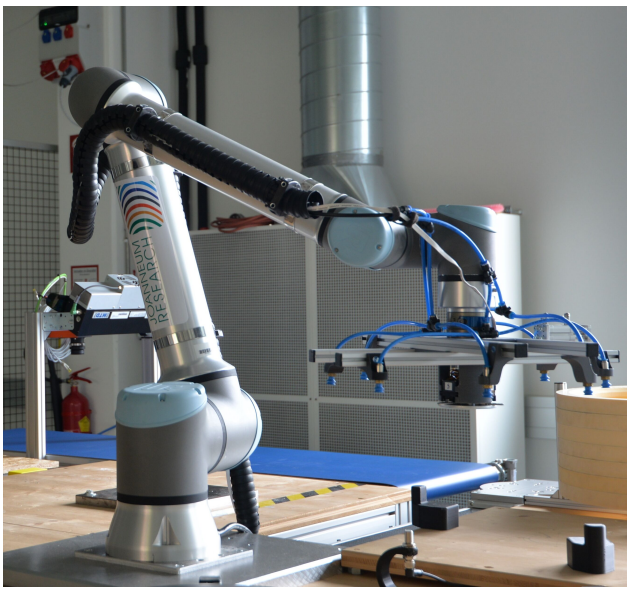


Abbildung 3.2: UR10e Roboter von Universal Robots mit multifunktionalem Endeffektor (»universal-robots.com«, 2024)

Für den Montageprozess wurde ein speziell entwickelter, multifunktionaler Endeffektor am Roboterarm angebracht. Der Greifer setzt sich aus drei Hauptkomponenten zusammen: einem Vakuumgreifer mit acht Vakuum-Saugkappen, einem zylindrischen Innengreifer und einem Leimverteiler. Der Greifarm dient dazu, die Einzelteile zuerst anzuheben und anschließend an die entsprechende Position zu bringen. Dabei werden die drei zusammenzusetzenden Einzelkomponenten auf der Arbeitsfläche so angeordnet, dass sie für den Roboterarm vollständig erreichbar sind. Sie werden durch kleine 3-D-gedruckte Stützen zentriert und in einer vordefinierten Position fixiert. Diese Anordnung gewährleistet die Wiederholgenauigkeit des Montagevorgangs und minimiert das Risiko einer Kollision des Greifers mit der Arbeitsfläche. Abbildung 3.3 zeigen das Greifwerkzeug und den Montagearbeitsplatz des Roboters.

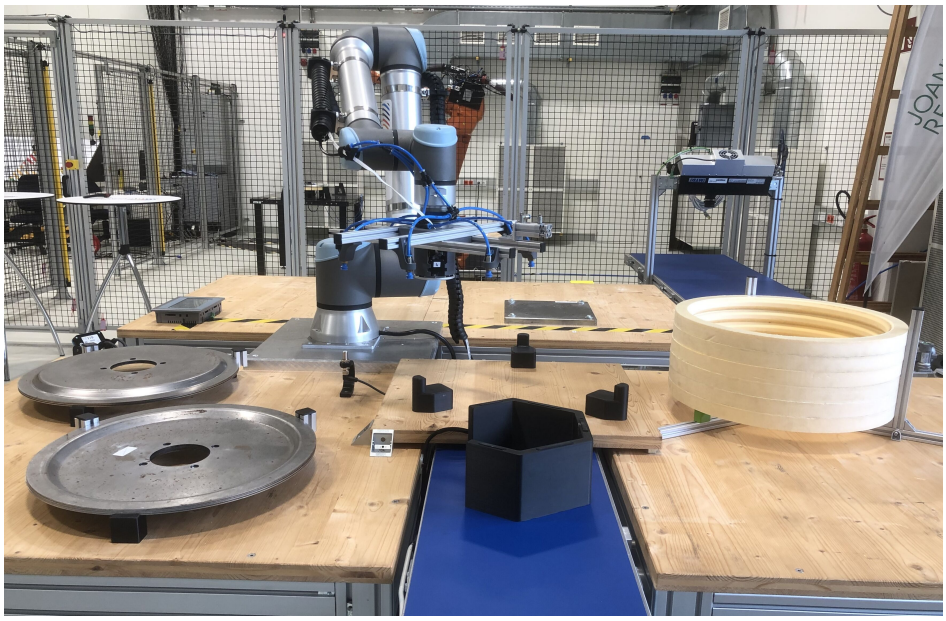


Abbildung 3.3: Montagearbeitsplatz des Roboters

Der Zusammenbau der Seilführungsscheibe erfolgt in acht Schritten. Zuerst hebt der Manipulator die erste Platte an (1) und transportiert sie zum induktiven Näherungssensor, der zur präzisen Ausrichtung und Erkennung der Bohrungen in der Platte dient. Dabei muss sichergestellt werden, dass die anfänglich in unbekannter Rotation vorliegenden Platten kongruent zueinander ausgerichtet werden. Sobald ein Bohrloch detektiert worden ist (2), speichert der Manipulator die Position und befördert die Metallplatte zur zentralen Montagestation (3). Anschließend holt sich der Roboterarm den Kunststoffring (4) und befestigt diesen mithilfe einer Klebung an der ersten Platte (5). Darauffolgend greift der Manipulator die zweite Metallplatte (6). Auch hier wird zunächst das Bohrloch detektiert, um eine präzise Positionierung sicherzustellen. Sobald die Position des Bohrlochs erfasst wurde (7), wird die Platte zur Montagestation transportiert, wo sie mit der bereits vorbereiteten Platte und dem befestigten Kunststoffring verbunden wird. Zu guter Letzt werden die zusammengesetzten Komponenten angehoben und zum Förderband transportiert (8).

Die Implementierung dieses Montageprozesses wurde mittels des Teach-Pendants und der Software PolyScope durchgeführt. Diese Software bietet eine Drag-and-Drop-Programmierschnittstelle, die einen Programmbaum aus verschiedenen Roboteroperationen erstellt. Abbildung 3.4 zeigt das Teach-Pendant sowie einen Ausschnitt des angefertigten Programmbaums mit den zugehörigen Programmknoten.

Abbildung 3.4: Teach-Pendant samt Ausschnitt des Programmbaums und Programmknoten

3.2 Real-Time Data Exchange (RTDE)

Nachdem der Anwendungsfall konstruiert wurde, kommt die RTDE-Schnittstelle (Real-Time Data Exchange) zum Einsatz. Diese ermöglicht es, die benötigten Rohdaten vom Robotersystem effizient und in Echtzeit zu erfassen. Die RTDE-Schnittstelle ermöglicht eine nahtlose Synchronisation externer Anwendungen mit der Universal Robots (UR)-Steuerung über eine Standard-TCP/IP-Verbindung, ohne dabei die Echtzeiteigenschaften zu beeinträchtigen. Diese Synchronisationsmöglichkeit steht automatisch zur Verfügung, sobald die UR-Steuerung in Betrieb genommen wurde (»Real-Time Data Exchange (RTDE) Guide«, 2019).

RTDE bietet dabei die Möglichkeit, gezielt Variablen wie den Roboterstatus, die Gelenkpositionen,

die Werkzeugdaten und den Sicherheitsstatus auszulesen sowie Steuerungsbefehle und Eingabewerte wie digitale und analoge Ein- und Ausgänge an die Robotersteuerung zu übermitteln. Die Kommunikation erfolgt dabei über sogenannte Rezepte, welche die Struktur der zu übertragenden Daten definieren und somit eine gezielte Konfiguration und Synchronisation der Datenpakete ermöglichen (»Real-Time Data Exchange (RTDE) Guide«, 2019).

Listing 3.1 zeigt ein Beispiel für ein Ausgangsrezept, das die zu synchronisierenden Variablen, ihre Datentypen und die Frequenz, mit der die Daten erfasst werden sollen, spezifiziert.

Listing 3.1: Ausgangsrezept für den Kommunikationsaufbau mit RTDE 1

```
configuration:
  ur:
    robot_ip: '143.224.224.3'
    variables: [timestamp, runtime_state, actual_q, actual_qd,
        actual_TCP_pose, actual_TCP_speed, joint_temperatures,
        script_control_line, tcp_force_scalar, elbow_velocity]
    types: [DOUBLE, UINT32, VECTOR6D, VECTOR6D, VECTOR6D, VECTOR6D,
        VECTOR6D, UINT32, DOUBLE, VECTOR3D]
    frequency: 20
```

Um auf diese Variablen zuzugreifen, muss zunächst eine Verbindung zur RTDE-Schnittstelle eines UR hergestellt und die Kommunikation entsprechend konfiguriert werden. Listing 3.2 zeigt dies exemplarisch.

Listing 3.2: Ausgangsrezept für den Kommunikationsaufbau mit RTDE 2

```
import yaml
from rtde import *

        with open(config, 'r') as yaml_file:
            configuration = yaml.safe_load(yaml_file)

        self.robot_ip = configuration['configuration']['ur']['robot_ip']
        self.variables =
            configuration['configuration']['ur']['variables']
        self.types = configuration['configuration']['ur']['types']
        self.frequency =
            configuration['configuration']['ur']['frequency']

        self.connect()

    def connect(self):
        try:
            self.rtde_if = RTDE(self.robot_ip)
            self.rtde_if.connect()
            self.rtde_if.negotiate_protocol_version()
            self.rtde_if.send_output_setup(variables=self.variables,
```

```
20                                          types=self.types,
                                                frequency=self.frequency)
21              self.rtde_if.send_start()
22          except:
23              pass
```

In diesem Beispiel wird die Konfiguration aus einer YAML-Datei geladen, welche die IP-Adresse des UR-Roboters, die zu übertragenden Variablen, ihre Typen und die Frequenz der Datenübertragung beinhaltet. Anschließend wird versucht, eine Verbindung zur RTDE-Schnittstelle des angegebenen Roboters hergestellt. Nachdem diese erfolgreich aufgebaut worden ist, werden die Informationen zur Ausgabe gesendet und die Datensynchronisierung gestartet. Diese ermöglicht es, in Echtzeit präzise Daten zu erfassen und den aktuellen Zustand des Roboters kontinuierlich zu verfolgen.

Folgende Tabelle 3.1 zeigt einen Ausschnitt der verfügbaren Ausgabedaten, die von der RTDE-Schnittstelle bereitgestellt werden (»Real-Time Data Exchange (RTDE) Guide«, 2019).

Name	Typ	Bemerkung
timestamp	DOUBLE	Zeit, die seit dem Start des Controllers verstrichen ist [s]
actual_q	VECTOR6D	Tatsächliche Gelenkpositionen
actual_qd	VECTOR6D	Tatsächliche Gelenkgeschwindigkeiten
actual_current	VECTOR6D	Tatsächlicher Strom je Gelenk
actual_current_window	VECTOR6D	Tatsächlicher Strom je Gelenk im aktuellen Zeitfenster
joint_control_output	VECTOR6D	Aktueller Steuerungsstrom je Gelenk
actual_TCP_pose	VECTOR6D	Tatsächliche kartesische Koordinaten des Werkzeugs: (x,y,z,rx,ry,rz), wobei rx, ry und rz die Darstellung eines Rotationsvektors der Werkzeugausrichtung ist
actual_TCP_speed	VECTOR6D	Die tatsächliche Geschwindigkeit des Werkzeugs wird in kartesischen Koordinaten angegeben. Die Geschwindigkeit wird in [m/s] angegeben und der Rotationsanteil der TCP-Geschwindigkeit (rx, ry, rz) ist die Winkelgeschwindigkeit, die in [rad/s] angegeben wird.
actual_TCP_force	VECTOR6D	Verallgemeinerte Kräfte im TCP. Er kompensiert die Messung für Kräfte und Momente, die durch die Nutzlast erzeugt werden
...	...	...

Tabelle 3.1: Auszug RTDE Ausgäng der Robotersteuerung (»Real-Time Data Exchange (RTDE) Guide«, 2019)

3.3 Herbeigeführte Anomalien

Um den Anwendungsfall sowohl für das regelbasierte Datenmonitoring als auch für den maschinellen Ansatz evaluieren zu können, wurden selbst herbeigeführte Anomalien erzeugt. Diese umfassen

zusätzliches Gewicht sowie die reduzierte Geschwindigkeiten für die kontextuelle Anomalie und das Fallen lassen von Objekten bei der punktuellen Anomalie. Jede dieser Anomalien wurde gezielt herbeigeführt, um spezifische Szenarien zu simulieren und die Leistung der Monitoringsysteme in verschiedenen Testbedingungen zu bewerten.

Die Anomalie des zusätzlichen Gewichts wurde realisiert, indem die zweite Platte der Seilführungsscheibe mit einem ergänzenden Gewicht von 814 g belastet wurde. Ein höheres Gewicht war in diesem Fall nicht möglich, da der Vakuumgreifer sonst nicht in der Lage gewesen wäre, die Last zu heben. Diese Belastung wurde bei jedem zweiten Durchlauf innerhalb von 15 Zyklen durchgeführt. Folgende Abbildungen veranschaulichen den Prozesses mit zusätzlichem Gewicht und zeigen deutlich, dass nicht nur die zweite Platte, sondern auch die gesamte konstruierte Seilführungsscheibe an Gewicht zugenommen hat. In Abbildung 3.5 ist das zusätzliche Gewicht auf der zweiten Platte deutlich erkennbar. Dieses wird durch die beiden dreieckigen Gewichte dargestellt.

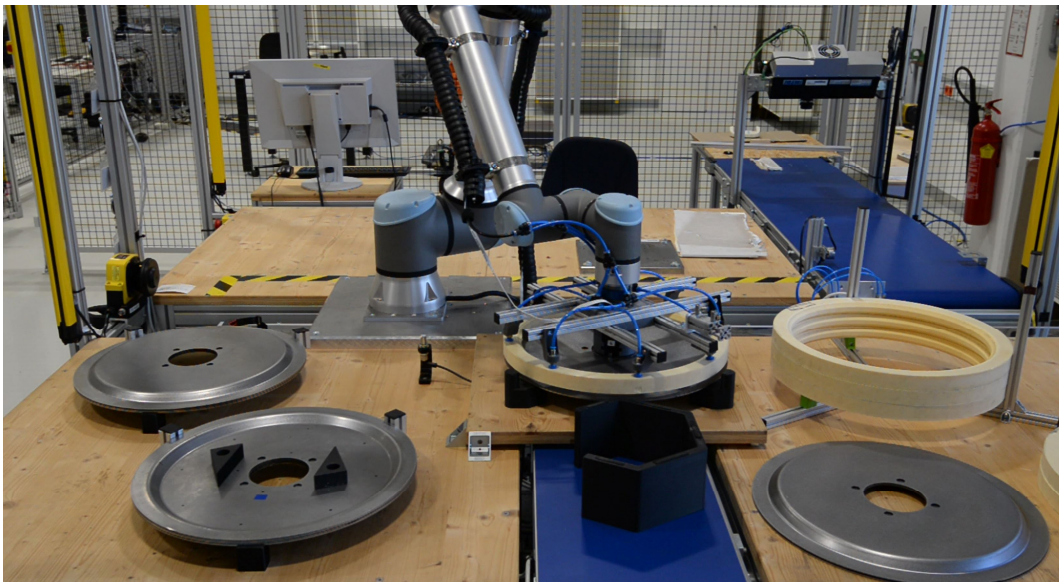


Abbildung 3.5: Anomalie des zusätzlichen Gewichts 1

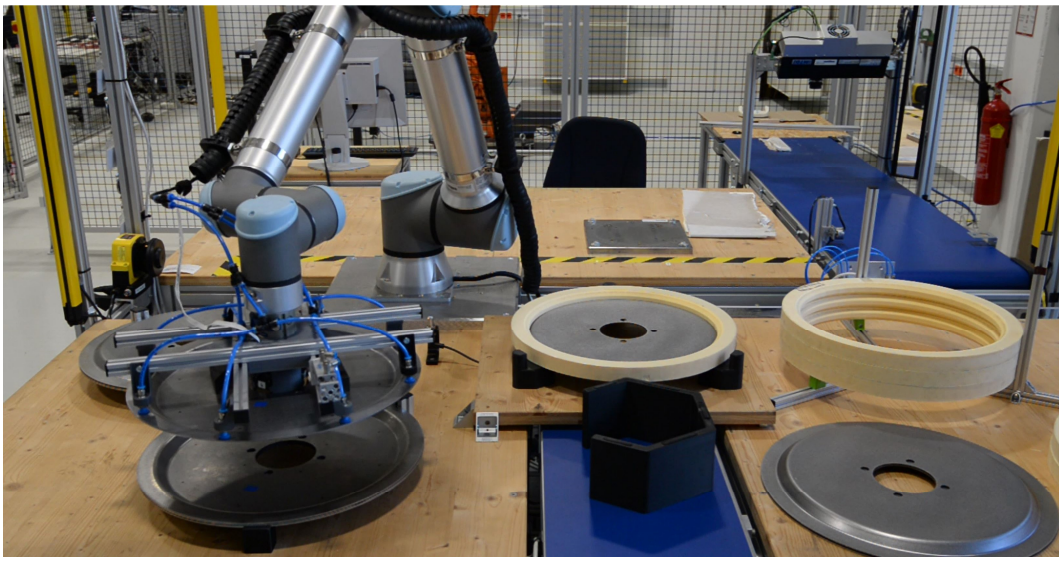


Abbildung 3.6: Anomalie des zusätzlichen Gewichts 2

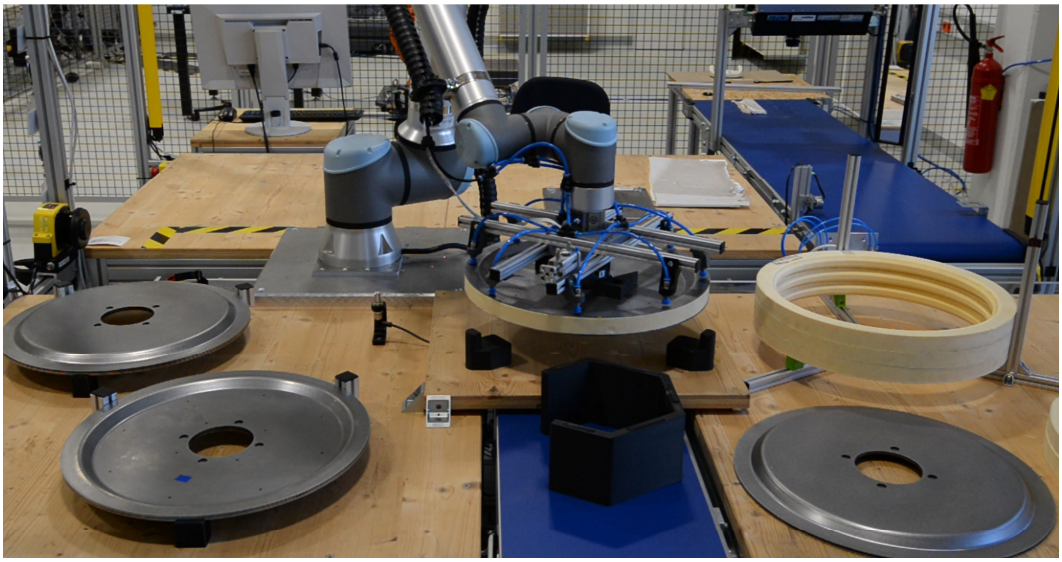


Abbildung 3.7: Anomalie des zusätzlichen Gewichts 3

Die Anomalie des Fallens eines Objektes wurde durch gezieltes Herbeiführen des Falls der zweiten Platte der Seilführungsscheibe simuliert. Dies erfolgte, indem das Luftdruckventil kurz nach dem Anheben den Platte abgeschaltet wurde, wodurch das Vakuum im Greifer aufgehoben und die Haltekraft verloren ging. Dieser Prozess wurde regelmäßig bei jedem zweiten Durchlauf innerhalb von 15 Zyklen wiederholt. In Abbildung 3.8 wird der Moment dargestellt, in dem die Platte fallengelassen wird. Abbildung 3.9 zeigt hingegen, dass die zuvor fallengelassene Platte in der Baugruppe nicht mehr vorhanden ist.

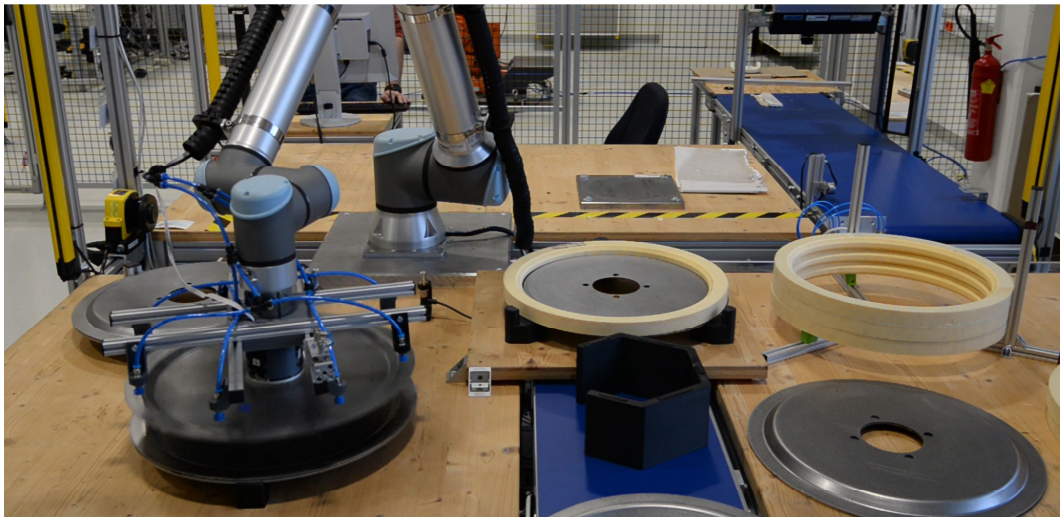


Abbildung 3.8: Anomalie des Fallen lassen eines Objektes 1

Abbildung 3.9: Anomalie des Fallen lassen eines Objektes 2

Die Reduzierung der Geschwindigkeit wurde mithilfe des Geschwindigkeitsreglers der UR-Steuerung realisiert. In jedem zweiten Durchlauf wurde die Geschwindigkeit zwischen dem Aufheben des Kunststoffrings und dem Montieren um 50% verringert. Abbildungen 3.10 und 3.11 illustrieren die Phase, in der der Roboter mit reduzierter Geschwindigkeit arbeitet.

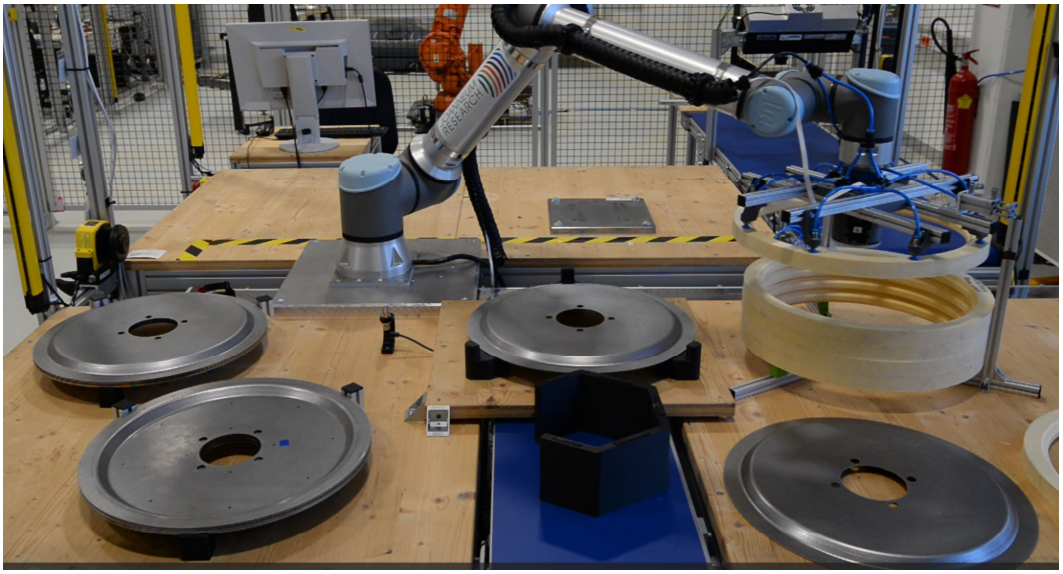


Abbildung 3.10: Anomalie der reduzierten Geschwindigkeit 1

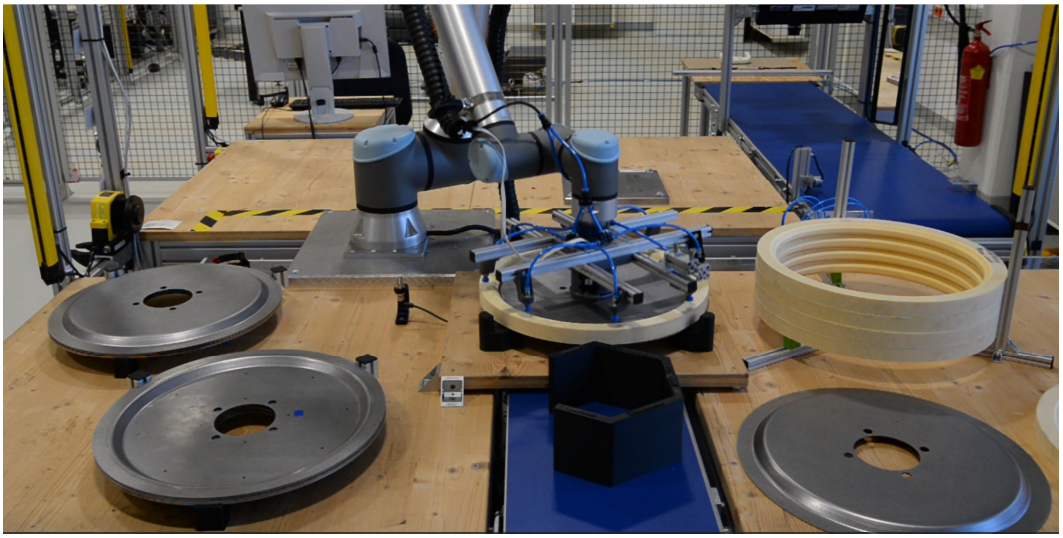


Abbildung 3.11: Anomalie der reduzierten Geschwindigkeit 2

Bei einem UR10e wäre die maximale Frequenz der Datenerfassung 500 Hz. Für die vorliegende Studie wurde eine ausreichende Frequenz von 125 Hz gewählt. Tabelle 3.2 gibt eine Übersicht über die Datensätze.

3.4 Kennzahlen zur Evaluierung

Um die Leistung des regelbasierten und maschinellen Ansatzes anhand der herbeigeführten Anomalien zu bewerten, werden standardisierte Metriken zur Leistungsbewertung herangezogen, darunter die Genauigkeit, Präzision, Recall und das F-Maß. Diese Kennzahlen können mithilfe der Konfusionsmatrix einfach abgebildet werden (Room, 2019). Eine beispielhafte Konfusionsmatrix ist in Abbildung 3.12 dargestellt.

Datensatzname	Gesamtproben	Anomalieproben	Anteil Anomalien (%)
Normal	854608	0	0.00
Zusätzliches Gewicht	85769	15544	18.12
Objekt fallen lassen	125949	14132	11.22
Reduzierte Geschwindigkeit	134208	2154	1.60

Tabelle 3.2: Zusammenfassung der im Normalbetrieb und bei herbeigeführten Anomalien erfassten Datensätze

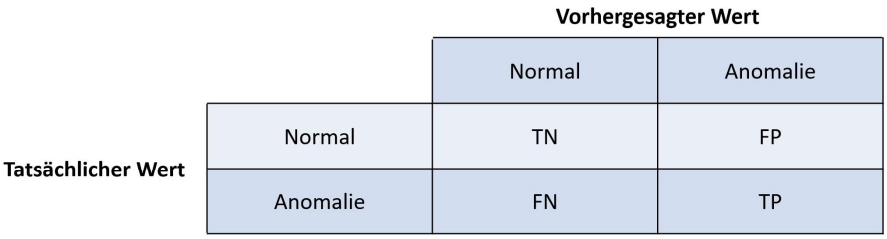


Abbildung 3.12: Konfusionsmatrix

Die Definitionen der Metriken TP, TN, FP und FN lauten wie folgt (Room, 2019):

- True Positive (TP): Datenpunkte, die Anomalien korrekt als Anomalien identifiziert haben.
- True Negative (TN): Datenpunkte, die normale Zustände korrekt als normal identifiziert haben.
- False Positive (FP): Datenpunkte, die normale Zustände fälschlicherweise als Anomalien klassifiziert haben.
- False Negative (FN): Datenpunkte, die Anomalien fälschlicherweise als normal klassifiziert haben.

Basierend auf den Definitionen von TP, TN, FP und FN können die angegebenen Bewertungsmetriken wie folgt berechnet werden.

Ein häufig verwendetes Maß für die Qualität der gesamten Klassifizierung ist die Genauigkeit, auch Accuracy genannt, die den Anteil der korrekt zugeordneten Fälle angibt (Foody, 2023).

$$\text{Genauigkeit} = \frac{TP+TN}{TP+TN+FP+FN}$$

Der Recall, auch als Sensitivität bekannt, misst, wie gut ein Klassifikator positive Fälle erkennt, die tatsächlich positiv sind. Diese Metrik ist weit verbreitet, insbesondere in der Informatik und im maschinellen Lernen (Lalkhen & McCluskey, 2008). Trotz ihrer Nützlichkeit hat sie Einschränkungen, da sie keine Informationen über falsch-positive (FP) Fälle liefert und somit nicht die vollständige Genauigkeit der Klassifikation in Bezug auf positive Fälle erfasst. Der Recall drückt demnach die Wahrscheinlichkeit aus, dass ein positiver Fall korrekt vorhergesagt wird (Baldi et al., 2000).

$$\text{Recall} = \frac{TP}{TP+FN}$$

Alternativ dazu interessiert sich die Präzision für die Wahrscheinlichkeit, dass eine positive Vorhersage tatsächlich richtig ist (Baldi et al., 2000).

$$\text{Präzision} = \frac{TP}{TP+FP}$$

Eine weitere weitverbreitete Metrik, die die Informationen von zwei grundlegenden Metriken kombiniert, ist das F-Maß. Das F-Maß ist das harmonische Mittel von Recall und Präzision und kann wie folgt berechnet werden:

$$\text{F-Maß} = 2 \cdot \frac{\text{Präzision} \cdot \text{Recall}}{\text{Präzision} + \text{Recall}}$$

Das F-Maß variiert zwischen 0 und 1. Der kleinstmögliche Wert 0 wird erreicht, wenn Recall und/oder Präzision null sind. Der größtmögliche Wert 1 wird erreicht, wenn sowohl Recall als auch Präzision eine perfekte Klassifizierung anzeigen. Obwohl das F-Maß häufig als Genauigkeitsmaß verwendet wird, ist er für unausgewogene Datensätze ungeeignet. Seine Größe kann durch die Prävalenz der Klassen beeinflusst werden, was zu irreführenden Ergebnissen führen kann. Zudem nutzt das F-Maß nicht alle in der Konfusionsmatrix enthaltenen Informationen (Chicco & Jurman, 2020).

Während die Genauigkeit und das F-Maß nützliche Metriken sein können, sind sie bei unausgewogenen Klassen oft unzureichend. Ein Hauptproblem besteht darin, dass diese Metriken stark von unausgewogenen Klassen beeinflusst werden und die Mehrheitsklasse bevorzugen, was sie oft uninformativ macht. Zum Beispiel würde eine einfache Strategie, die alle Datenpunkte als normal klassifiziert, ebenfalls zu einer scheinbar hohen Genauigkeit führen, obwohl sie die eigentliche Aufgabe der Anomalieerkennung nicht erfüllt (Chicco & Jurman, 2020).

Eine gängige Kennzahl zur Bewältigung der Herausforderungen unausgewogener Klassendaten ist die Receiver Operating Characteristic (ROC)-Kurve (Aggarwal, 2024). Die ROC-Kurve veranschaulicht den Kompromiss zwischen der True Positive Rate (TPR), auch als Recall bekannt, die den Anteil der korrekt erkannten Anomalien darstellt, und der False Positive Rate (FPR), die den Anteil der normalen Instanzen angibt, die fälschlicherweise als anomal klassifiziert wurden. Sie wird durch das Darstellen der TPR gegen die FPR erstellt (Fan et al., 2006).

$$\text{TPR} = \frac{TP}{TP+FN}$$

$$\text{FPR} = \frac{FP}{FP+TN}$$

Die ROC-Kurve bietet den Vorteil, dass ihre Fläche unter der Kurve (Area Under Curve, AUC) als Maß für die Trennschärfe des Detektors interpretiert werden kann (Fan et al., 2006). Die ROC-AUC kann als die Wahrscheinlichkeit verstanden werden, dass der Detektor einer Anomalie eine höhere Bewertung zuweist, wie z. B. einen größeren Abstand oder einen höheren Rekonstruktionsfehler im Vergleich zu einem normalen Datenfall (Aggarwal, 2024). Eine rein zufällige Klassifizierung wird durch eine Diagonale von links unten nach rechts oben repräsentiert. Die zugehörige AUC wäre in diesem Fall 0,5. Im optimalen Szenario erreicht die AUC den Wert eins.

Abbildung 3.13 zeigt beispielhafte einen Verlauf der ROC-Kurve mit dazugehöriger AUC.

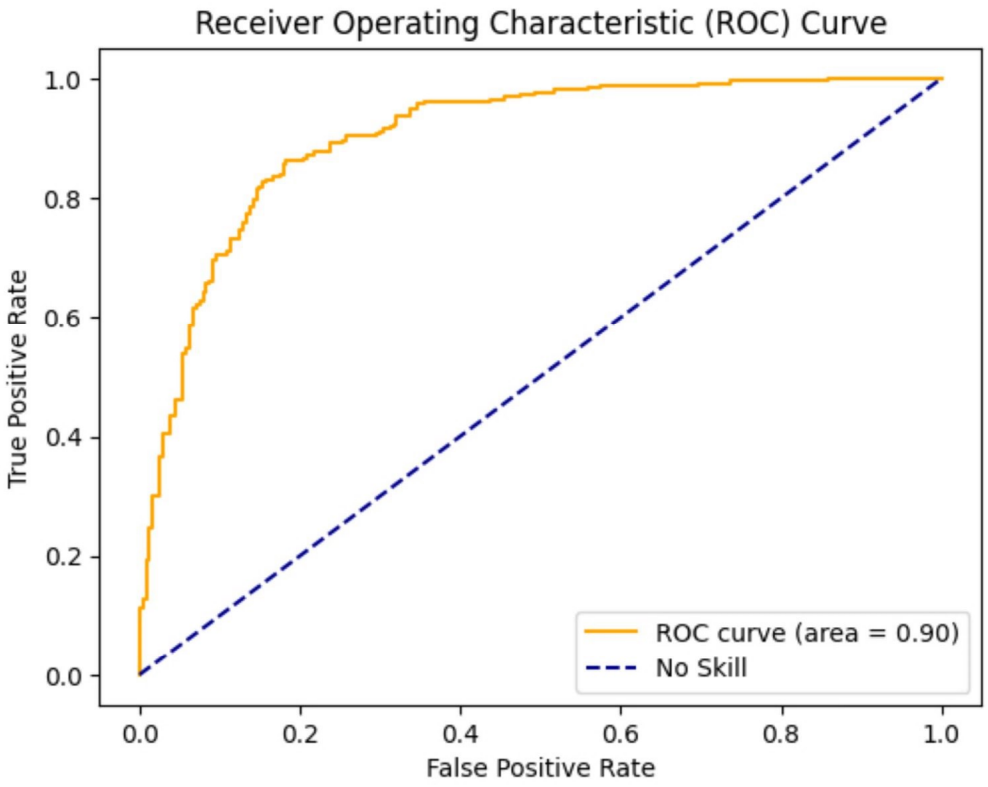


Abbildung 3.13: Beispiel einer ROC

Wie in der Grafik ersichtlich, repräsentiert die x-Achse die FPR und die y-Achse die TPR. Die daraus resultierende Fläche unter der Kurve, der AUC-Wert, beträgt 0.90. Eine AUC von 0.90 zeigt, dass das Modell eine sehr gute Trennleistung besitzt. Das bedeutet, dass das Modell in 90% der Fälle einen zufälligen positiven Fall korrekt von einem zufälligen negativen Fall unterscheiden kann.

4

Regelbasierter Ansatz

Dieses Kapitel thematisiert den regelbasierten Ansatz für das Datenmonitoring von Roboteranwendungen. Zunächst wird das zugrunde liegende Konzept dieses Ansatzes erläutert. Daraufhin folgt die Implementierung, die anhand eines exemplarischen Regelwerks veranschaulicht wird. Abschließend wird eine Evaluierung durchgeführt, in der die wesentlichen Kennzahlen dargelegt werden, um die Leistungsfähigkeit des regelbasierten Ansatzes zu bewerten.

4.1 Konzept

Das Konzept des regelbasierten Datenmonitorings beschreibt die Implementierung des Überwachungsmechanismus, welcher vordefinierte Regeln zur kontinuierlichen Analyse und Bewertung von Echtzeitdaten verwendet. Abbildung 4.1 zeigt den grundlegenden Rahmen dieses Verfahrens.

In der dargestellten Abbildung 4.1 umfasst der erste Schritt die Übertragung von Rohdaten zum Roboterstatus vom Robotersystem. Die Robotersteuerung überträgt diese Daten entweder über eine direkte Verbindung wie Real-Time Data Exchange (RTDE) im Falle eines UR-Roboters oder über Kommunikationsprotokolle wie das Robot Operating System (ROS) (Quigley et al., 2009). Anschließend verarbeitet das vorgeschlagene System die erfassten Daten, um Herausforderungen wie Rauschunterdrückung, Kalibrierung oder die Extraktion relevanter Merkmale vor der Anwendung der Regeln zu bewältigen. Die Regelüberprüfung bewertet aktiv die vorverarbeiteten Daten und entscheidet anhand der festgelegten Parameter, ob diese die Regeln auslösen. Dadurch wird sichergestellt, dass das System die vordefinierten Bedingungen kontinuierlich einhält. Abhängig von der Aktivierung einer Regel wählt das System eine spezifisch definierte Menge von Datenpunkten aus und zeichnet sie auf. In diesem Sinne werden zusätzlich historische Daten aufgezeichnet, die sowohl den Zeitraum vor als auch nach der Aktivierung von Regeln umfassen. Die historischen Daten werden genutzt, um die Herkunft des auslösenden Ereignisses miteinzubeziehen. Auf der Grundlage der gesammelten Daten können Fachleute, die über eine hohe Kompetenz in der Verifikation von Robotersystemen verfügen, die Regeln iterativ anpassen und verbessern und neue Regeln zum bestehenden Regelwerk hinzufügen.

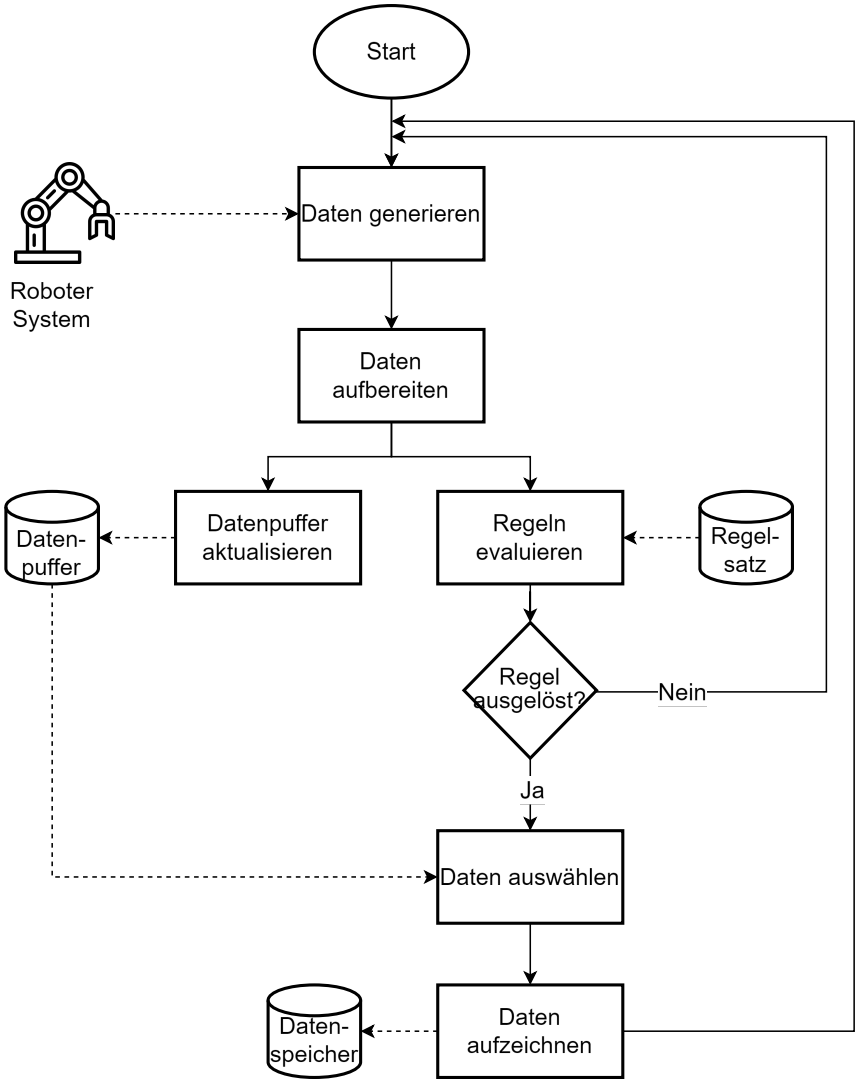


Abbildung 4.1: Konzeptueller Rahmen des regelbasierten Ansatzes

4.2 Implementierung

Die Implementierung des regelbasierten Datenmonitorings wurde mittels der Programmiersprache Python realisiert. Hierbei wurde ein modulares Design angewendet, welches die Regelverwaltung, die Dateneingabe und die Regelüberprüfung in klar abgegrenzte Module unterteilt. Die UML Diagramme 4.2 und 4.3 veranschaulichen die Struktur des entworfenen Monitoringsystems. Dabei werden die Klassen, deren Beziehungen, Attribute und Methoden übersichtlich dargestellt.

4.2.1 Dateneingabemodul

Die Klassen DataReceiver und DataSource stellen die Grundlage für das Empfangen von Daten dar. Sie dienen gemeinsam dem Zweck, Daten von externen Quellen zu empfangen und sie an das System weiterzuleiten. Quellen können dabei Real-Time Data Exchange (RTDE) Daten, ROS-Topics oder tabellenbasierte Eingaben für Offline-Tests umfassen. Hierbei ist es ebenfalls mittels der unten beschriebenen Klasse SerialMergingSource möglich, Eingaben von einer beliebigen Anzahl von Quellen dieser Typen zu kombinieren.

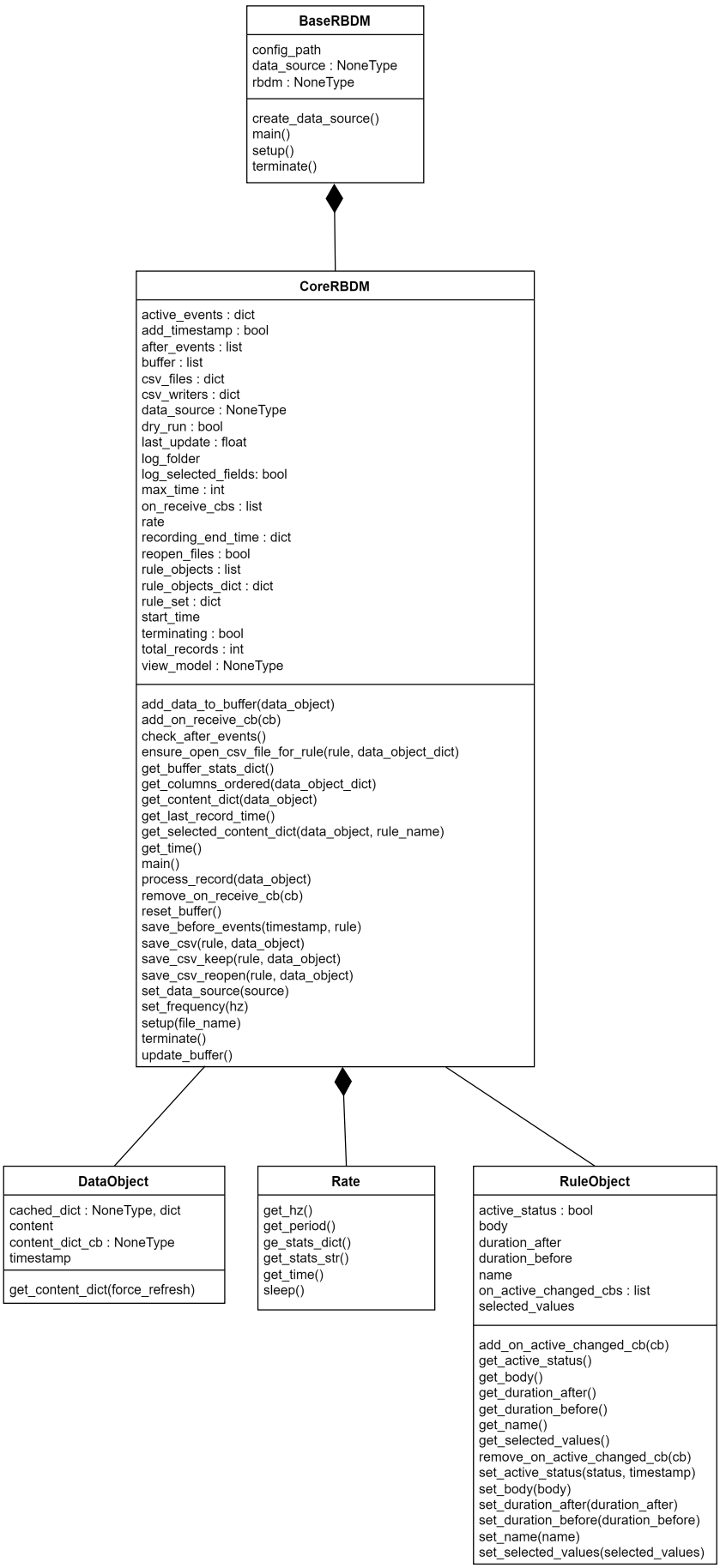


Abbildung 4.2: UML Diagramm regelbasiertes Monitoring 1

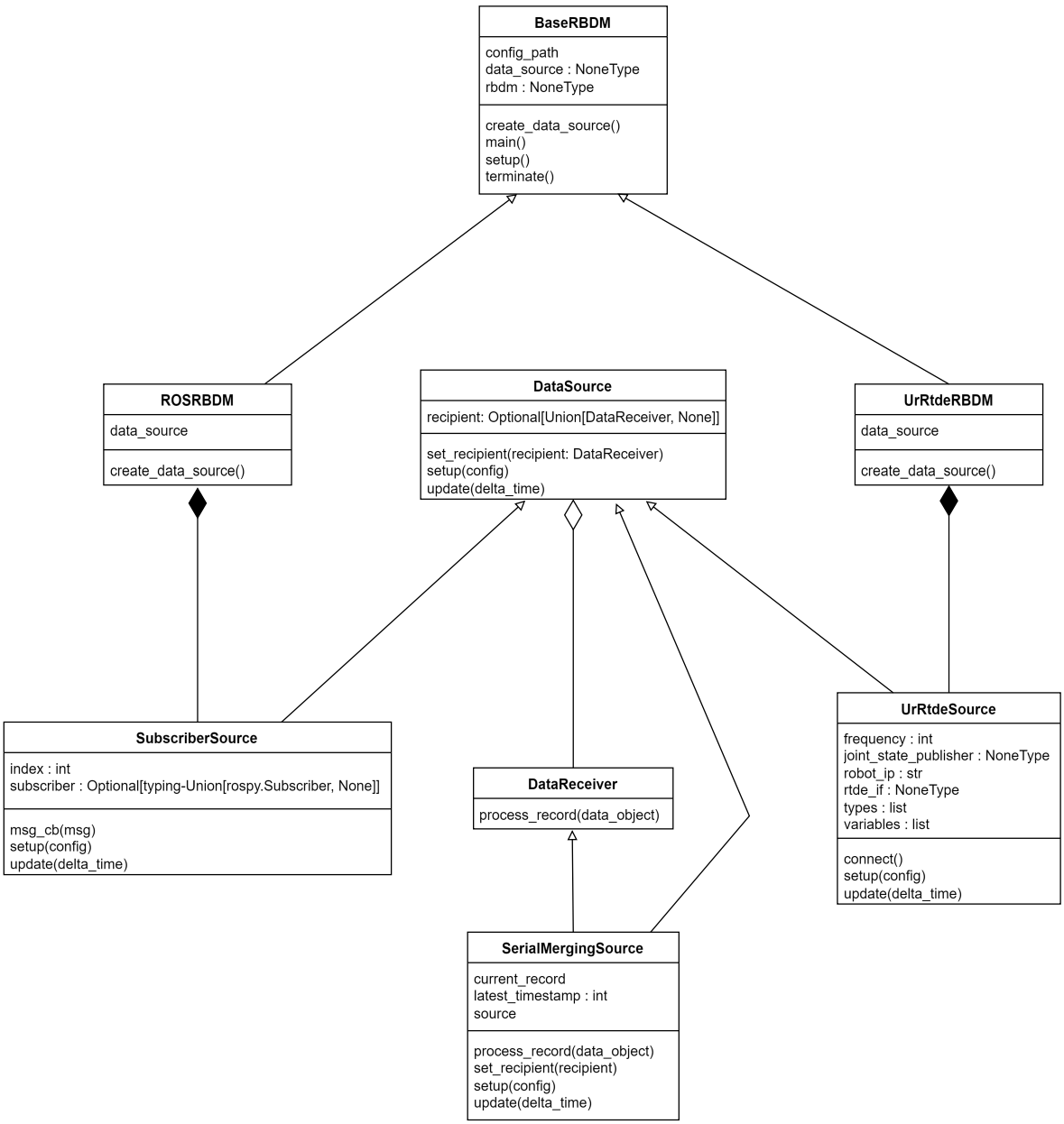


Abbildung 4.3: UML Diagramm regelbasiertes Monitoring 2

Real-Time Data Exchange (RTDE)

Die UrRtdeSource-Klasse erbt von der abstrakten Klasse DataSource und implementiert die spezifischen Details für den Empfang von Daten über das RTDE-Protokoll. Hierzu verwendet sie die RTDE-Bibliothek (Milkowski & Kautilya, 2022). Die Klasse initialisiert und verwaltet die RTDE-Verbindung, stellt eine Möglichkeit zur Datenaktualisierung bereit und sendet die empfangenen Datenobjekte an den registrierten DataReceiver.

Robot Operating System (ROS)

Die SubscriberSource-Klasse erbt von der abstrakten Klasse DataSource und implementiert die spezifischen Details für den Empfang von ROS-Nachrichten über einen ROS-Abonnenten. Hierzu verwendet sie die ROS-Bibliothek (»rospy - ROS Wiki«, 2017). Die Klasse initialisiert und verwaltet den ROS-Abonnenten, stellt eine Möglichkeit zur Datenaktualisierung bereit und sendet die empfangenen Datenobjekte an den registrierten DataReceiver.

Tabellen und Speed and Separation Monitoring (SSM)-Bibliothek

Eine im Projektteam entwickelte Python-Bibliothek wurde verwendet, um Berechnungen gemäß den Anforderungen des Speed and Separation Monitoring (SSM) (International Organization for Standardization, 2016) durchzuführen. Die Bibliothek nutzt ein flexibles Konzept zur Verknüpfung verschiedener Datenquellen und Verarbeitungsschritte. Um die jeweiligen Module nutzen zu können, wurde eine Adapterklasse TableToRTDEAdapter entwickelt, die den Datenaustausch zwischen den Datenquellen des regelbasierten Monitorings und SSM ermöglichen. Die SSM-Bibliothek verfügt zudem über ein Modul, das es ermöglicht, Tabellendateien zu lesen. Auf diese Funktionalität wird im regelbasierten Ansatz zugegriffen. Dies hat sich besonders beim Testen neuer Regeln und Funktionen bewährt, da dieselben Auswertungen auch ohne Echtzeitdaten durchgeführt werden können.

Mehrere verschiedene Datenquellen

Die SerialMergingSource ist eine spezielle Implementierung der DataSource-Klasse. Diese Klasse zeichnet sich durch die Fähigkeit aus, mehrere Datenquellen miteinander zu fusionieren. Das bedeutet, dass sie in der Lage ist, Daten von verschiedenen Quellen zu kombinieren und als kohärente Datensätze an die nächste Verarbeitungsstufe weiterzuleiten.

4.2.2 Regelverwaltung

Die Regelverwaltung erfolgt durch die Klasse RuleObject, die Regeln anhand von Parametern wie Regelkörper, Dauer vor und nach dem Ereignis sowie ausgewählten Werten repräsentiert. Die CoreRBDM-Klasse übernimmt die Verwaltung dieser Regelobjekte. Sie lädt die Regeldefinitionen aus einer YAML-Konfigurationsdatei und konvertiert sie mithilfe der rule-engine-Bibliothek (»rule-engine — pypi.org«, 2024) von Python in ausführbare Regelinstanzen. Diese Bibliothek erleichtert die Filterung verschiedener Python-Objekte durch die Verwendung der definierten Regeln, die als String-Werte ausgedrückt werden. Ein Beispiel für eine definierte Regel lautet $tcp\_force\_scalar > 50$, was eine Bedingung darstellt, bei der die Tool Center Point (TCP)-Kraft den definierten Schwellenwert von 50 N überschreitet. Listing 4.1 zeigt zum einen eine Übersicht der Konfigurationspunkte, hier Verbindung zum UR Roboter über RTDE, und zum anderen die definierten Regeln.

Listing 4.1: Auszug aus Konfigurationsdatei mit RTDE-Anbindung und drei Regeln

```
configuration:
  ur:
    robot_ip: '143.224.224.3'
    variables: [timestamp, runtime_state, actual_q, actual_qd,
        actual_TCP_pose, actual_TCP_speed, joint_temperatures,
        script_control_line, tcp_force_scalar, elbow_velocity]
    types: [DOUBLE, UINT32, VECTOR6D, VECTOR6D, VECTOR6D, VECTOR6D,
        VECTOR6D, UINT32, DOUBLE, VECTOR3D]
    frequency: 20
  core:
    frequency: 20
    frequency_strict: False
```

```
    reopen_files: False
    add_timestamp: False
    dry_run: True
rules:
  - name: 'TCP bounding box'
    body: 'not(actual_TCP_pose[0] >= 0.252 and actual_TCP_pose[0] <=
       0.877 and actual_TCP_pose[1] >= -0.402 and actual_TCP_pose[1] <=
       0.969 and actual_TCP_pose[2] >= -0.001 and actual_TCP_pose[2] <=
       0.445)'
    duration_before: 2
    duration_after: 0.2
    selected_values: [
       'actual_TCP_pose[0]','actual_TCP_pose[1]','actual_TCP_pose[2]' ]
    plot_threshold_lower: [ 0.252, -0.402, -0.001 ]
    plot_threshold_upper: [ 0.877, 0.969, 0.445 ]
    plot_axis_y_max: [ 1, 1, 1 ]
    plot_axis_y_min: [ -1, -1, -1 ]
    plot_theme: ['theme_plot_x', 'theme_plot_y', 'theme_plot_z']
    description: 'Triggered if TCP exceeds bounding box'
  - name: 'TCP force'
    body: 'tcp_force_scalar > 50'
    duration_before: 2
    duration_after: 0.4
    selected_values: [ 'tcp_force_scalar' ]
    plot_threshold_upper: [ 50 ]
    plot_axis_y_max: [ 100 ]
    plot_axis_y_min: [ 0 ]
    description: 'maximum on TCP force'
  - name: 'TCP speed'
    body: 'actual_TCP_speed[0] * actual_TCP_speed[0] +
       actual_TCP_speed[1] * actual_TCP_speed[1] + actual_TCP_speed[2] *
       actual_TCP_speed[2] > 0.1'
    duration_before: 2
    duration_after: 0.2
    selected_values: [
       'actual_TCP_speed[0]','actual_TCP_speed[1]','actual_TCP_speed[2]'
       ]
    plot_axis_y_max: [ .5, .5, .5 ]
    plot_axis_y_min: [ -.5, -.5, -.5 ]
    plot_theme: ['theme_plot_x', 'theme_plot_y', 'theme_plot_z']
    description: 'Triggered if TCP speed (square sum) exceeds a limit'
```

Der erste Abschnitt aus Listing 4.1 beschreibt die Konfigurationsdaten des Roboters, einschließlich seiner IP-Adresse, der erfassten Variablen, deren Datentypen und der Frequenz, mit der diese Daten aufgezeichnet werden. Zusätzlich werden Parameter für das Logging festgelegt, wie das Öffnen neuer Dateien oder das Hinzufügen von Zeitstempeln. Die nachfolgenden Regeln bestehen jeweils aus einem Namen, einem Regelkörper, einer Zeitspanne vor und nach dem Auslösen der Regel sowie Parametern zur grafischen Visualisierung. Die erste Regel überprüft, ob der TCP (Tool Center Point) den definierten Arbeitsbereich verlässt, indem sie die x-, y- und z-Koordinaten

auf bestimmte Grenzwerte hin untersucht. Die zweite Regel überwacht, ob die auf den TCP wirkende Kraft einen Schwellenwert von 50 N überschreitet. In der dritten Regel wird untersucht, ob die Geschwindigkeit des TCP eine bestimmte Grenze überschreitet, indem die quadrierten Komponenten der Geschwindigkeitssumme ausgewertet werden.

4.2.3 Datenverwaltung

Die Datenverwaltung erfolgt durch die CoreRBDM-Klasse, die einen Puffer für empfangene Datenobjekte bereitstellt. Der Puffer wird in Echtzeit aktualisiert und bereinigt, um die Effizienz der Datenverarbeitung sicherzustellen. Die Datenobjekte, repräsentiert durch die Klasse DataObject, enthalten dabei sowohl den Zeitstempel als auch den Inhalt der übermittelten Daten. Dieser Puffer ermöglicht es dem System, einen klaren Überblick über die zeitliche Abfolge von Datenpunkten zu behalten und effektiv auf Veränderungen und Ereignisse zu reagieren.

4.2.4 Datenlogging

Die CSV-Protokollierung ermöglicht eine persistente Aufzeichnung der Daten. Bei Aktivierung der Regel wird der entsprechende Datenpunkt, der die Regel auslöst, in einer CSV-Datei gespeichert. Zusätzlich zu den Zeitstempeln bei Regelaktivierungen werden auch Zeitpunkte vor und nach der Aktivierung einer Regel entsprechend der Regeldefinition mitgeloggt. Dies ermöglicht es, den Zustand des Systems vor und nach dem Auftreten eines Ereignisses zu analysieren und potenzielle Ursachen besser zu verstehen. Die Struktur der CSV-Dateien orientiert sich an den definierten Datenobjekten und enthält Zeitstempel sowie ausgewählte Datenpunkte gemäß den konfigurierten Regeln. Die Implementierung berücksichtigt zudem verschiedene Optionen, wie das Hinzufügen von Zeitstempeln, die selektive Protokollierung ausgewählter Felder und die Möglichkeit, CSV-Dateien entweder kontinuierlich zu erweitern oder bei jeder Aktivierung einer Regel neu zu erstellen.

4.2.5 Systemsteuerung

Die Klasse “CoreRBDM” bildet das Herzstück des regelbasierten Datenmonitorings. Sie beschäftigt sich mit der Regelüberprüfung und dem Logging der entsprechenden Datenpunkte. Hierbei wird mithilfe der rule-engine Bibliothek (»rule-engine — pypi.org«, 2024) von Python überprüft, ob das in den Regeln definierte Systemverhalten mit dem während der Laufzeit übereinstimmt. Während der Laufzeit wird in der Methode $process\_record$ überprüft, ob ein vom Dateneingabemodul erhaltener Datensatz eine Regel auslöst. Falls dies der Fall ist, dann wird diese Regel auf aktiv gesetzt. Zudem werden die in der Regel definierten Datenpunkte sowie Protokollzeiten ausfindig gemacht und entsprechend geloggt. Anhand des Feldes $duration\_before$ und $duration\_after$ der Regel werden Daten aus dem Puffer für den entsprechenden Zeitraum vor und nach der Aktivierung der Regel beim Loggen inkludiert.

4.3 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche (GUI) für das regelbasierte Datenmonitoring wurde unter Verwendung von Dear PyGui (»dearpygui — pypi.org«, 2022) implementiert. Die GUI enthält verschiedene Elemente wie Fenster, Registerkarten, Schaltflächen, Textfelder und Diagramme. Die Anzeige visualisiert Regelverletzungen in Echtzeit und ermöglicht es Benutzern, zwischen verschiedenen Regeln zu navigieren und Details zu überwachen. Abbildung 4.4 zeigt einen Ausschnitt aus der implementierten Benutzeroberfläche.

Abbildung 4.4: Auszug aus der GUI des regelbasierten Datenmonitorings

Der obere Abschnitt der Abbildung 4.4 zeigt einen Überblick über die definierten Regeln und ihre jeweiligen Auslösepunkte. Die visuelle Darstellung umfasst zeitliche Muster der Regelaktivierung und Zeitstempel der Auslöser, was ein chronologisches Verständnis ereignisbasierter Regeln ermöglicht. Zu illustrativen Zwecken werden die Schwellenwerte für die Regeln absichtlich niedriger gesetzt, um ihre Aktivierung auszulösen und die Reaktionsfähigkeit des Systems zu zeigen. Der Abschnitt „Rule Details “bietet eine detaillierte Beschreibung der definierten Regeln. Benutzer können zwischen den einzelnen Regeln mithilfe der verfügbaren Registerkarten navigieren. Jede Regel wird mit ihren jeweiligen Bedingungen und ihrer zeitlichen Entwicklung repräsentiert. Das Liniendiagramm visualisiert sowohl Grenz- als auch Laufzeitwerte. In dieser speziellen Illustration hat die TCP-Kraft einen Grenzwert von 50 N, der zum Zeitpunkt 7131 überschritten wird.

4.4 Evaluierung

Die Evaluierung umfasst zunächst die Erstellung der Regeln zur Identifizierung der herbeigeführten Anomalien. Anschließend erfolgt die Auswertung dieser Regeln anhand der in Kapitel 3.4 festgelegten Kennzahlen.

4.4.1 Regeldefinition

Wie im Kapitel 3.3 beschrieben, wurden für die Evaluierung des regelbasierten und maschinellen Lernansatzes drei herbeigeführte Anomalien aufgezeichnet. Um diese Anomalien auszuwerten, wurde für den regelbasierten Ansatz folgender Regelsatz definiert.

Der Regelsatz zur Erfassung eines zusätzlichen Gewichts, speziell auf der zweiten Platte, ist im nachfolgenden Codeausschnitt 4.2 veranschaulicht. Jede Regel spezifiziert dabei eine Bedingung, die erfüllt sein muss, damit ein erhöhtes Gewicht erkannt werden kann. In diesem Zusammenhang werden Parameter wie Registerwerte, Nutzlast, Moment und Rohkraft überwacht, um festzustellen, ob ein zusätzliches Gewicht auf der zweiten Platte vorhanden ist.

Listing 4.2: Auszug aus dem Regelwerk zur Detektion zusätzlicher Gewichte

```
rules:
  - weight_phase_5:
    name: 'weight_phase_5'
    body: 'output_int_register_0 == 5 and payload == 2.5 and
       actual_momentum > 0 and ft_raw_wrench_2 > 36680'
    duration_before: 0
    duration_after: 0
    selected_values: [ 'output_int_register_0', 'payload',
       'actual_momentum', 'ft_raw_wrench_2' ]
  - weight_phase_6:
    name: 'weight_phase_6'
    body: 'output_int_register_0 == 6 and ft_raw_wrench_2 > 36681'
    duration_before: 0
    duration_after: 0
    selected_values: [ 'output_int_register_0', 'ft_raw_wrench_2' ]
  - weight_phase_7:
    name: 'weight_phase_7'
    body: 'output_int_register_0 == 7 and payload == 2.5 and
       ft_raw_wrench_2 > 36680'
    duration_before: 0
    duration_after: 0
    selected_values: [ 'output_int_register_0', 'payload',
       'ft_raw_wrench_2' ]
  - weight_phase_8:
    name: 'weight_phase_8'
    body: 'output_int_register_0 == 8 and payload == 8.252 and
       actual_momentum > 0 and ft_raw_wrench_2 > 36700'
    duration_before: 0
    duration_after: 0
```

```
25      selected_values: [ 'output_int_register_0', 'payload',
            'actual_momentum', 'ft_raw_wrench_2' ]
```

Des Weiteren wurde ein Regelsatz zur Erkennung des Fallens der zweiten Platte angefertigt. Im Codeausschnitt 4.3 ist das Regelwerk dazu dargestellt. Die Regeln sind so konzipiert, dass sie bestimmte Bedingungen für den Roboterstatus und die Sensordaten überwachen. Diese Regeln berücksichtigen Faktoren wie die Phase des Roboterprogramms, die Nutzlast, den aktuellen Schwung, die Kraft des TCP (Tool Center Point) und den maximalen Wert in der jeweiligen Phase. Wenn diese Bedingungen erfüllt sind, wird ein Fallen lassen des Objektes erkannt. Die genauen Schwellenwerte und Parameter variieren je nach spezifischer Phase, in der sich der Roboter befindet.

Listing 4.3: Auszug aus dem Regelwerk zur Detektion eines herabfallenden Objektes

```
1  rules:
2    - drop_phase_5:
3      name: 'drop_phase_5'
4      body: 'output_int_register_0 == 5 and payload == 2.5 and
           actual_momentum > 0 and tcp_force_scalar < max_value_in_phase *
           (2/3)'
5      duration_before: 0
6      duration_after: 0
7      selected_values: [ 'output_int_register_0', 'payload',
           'actual_momentum', 'tcp_force_scalar', 'max_value_in_phase']
8    - drop_phase_6:
9      name: 'drop_phase_6'
10     body: 'output_int_register_0 == 6 and tcp_force_scalar <
           max_value_in_phase * (2/3)'
11     duration_before: 0
12     duration_after: 0
13     selected_values: [ 'output_int_register_0', 'tcp_force_scalar',
           'max_value_in_phase']
14   - drop_phase_7:
15     name: 'drop_phase_7'
16     body: 'output_int_register_0 == 7 and payload == 2.5 and
           tcp_force_scalar < max_value_in_phase * (2/3)'
17     duration_before: 0
18     duration_after: 0
19     selected_values: [ 'output_int_register_0', 'payload',
           'tcp_force_scalar', 'max_value_in_phase' ]
20   - drop_phase_7:
21       name: 'drop_phase_8'
22       body: 'output_int_register_0 == 8 and payload == 8.252 and
             actual_momentum > 0 and tcp_force_scalar < 36'
23       duration_before: 0
24       duration_after: 0
25       selected_values: [ 'output_int_register_0', 'payload',
             'actual_momentum', 'tcp_force_scalar' ]
```

Die letzte Anomalie bezieht sich auf die Reduktion der Geschwindigkeit. Die Detektion dieser Anomalie lässt sich durch eine einzige Regel ableiten. Diese Regel wird aktiviert, wenn der Skalierungsfaktor der Geschwindigkeit unter 0.9 fällt. Dieses Verhalten weist auf eine Reduzierung der Geschwindigkeit hin.

Listing 4.4: Auszug aus dem Regelwerk zur Detektion reduzierter Geschwindigkeiten

```
rules:
  - reduced_speed:
    name: 'reduced_speed'
    body: 'speed_scaling < 0.9'
    duration_before: 0
    duration_after: 0
    selected_values: [ 'speed_scaling']
```

4.4.2 Ergebnisse

Nachdem der Regelsatz zur Identifikation der herbeigeführten Anomalien definiert wurde, erfolgte die Überprüfung anhand der tabellenbasierten Eingabe des entwickelten Frameworks, wie in Kapitel 4.2.1 beschrieben. In der nachstehenden Tabelle 4.1 sind die Metriken Genauigkeit, Präzision, Recall und das F-Maß für die jeweiligen Anomalien aufgelistet.

	Zusätzliches Gewicht	**Objekt fallen lassen**	**Reduzierte Geschwindigkeit**
Genauigkeit	0.9318	0.9607	1.0
Präzision	0.7414	0.7408	1.0
Recall	0.9581	0.9998	1.0
F-Maß	0.8359	0.8510	1.0

Tabelle 4.1: Analyse der Genauigkeit, Präzision, Recall und F-Maß bei Anomalieerkennung durch regelbasierten Ansatz

Wie der Tabelle zu entnehmen ist, zeigen die Kennzahlen unterschiedlich hohe Leistungsmerkmale für die drei Arten von Anomalien auf.

Die Anomalie des zusätzlichen Gewichts wird mit dem regelbasierten Ansatz mit einer Genauigkeit von 0.9318 erkannt. Demnach wird der Großteil der Anomalien korrekt klassifiziert. Die Präzision deutet darauf hin, dass von allen als positiv klassifizierten Ereignissen 74.14% tatsächlich diese Anomalie aufweisen. Der hohe Recall-Wert von 0.9581 zeigt, dass fast alle tatsächlichen Fälle von zusätzlichem Gewicht erkannt werden. Das F-Maß von 0.8359 bietet eine ausgewogene Bewertung und zeigt, dass das System gut darin ist, ein Gleichgewicht zwischen Präzision und Recall zu halten. Diese Auswertung lässt auf eine gute Gesamtleistung bei der Erkennung von zusätzlichem Gewicht schließen. Dies lässt sich ebenfalls in der nachstehenden Abbildung 4.5 der ROC-Kurve ablesen. Eine AUC von 0.94 weißt auf eine hervorragende Unterscheidungsfähigkeit des Modells zwischen positiven und negativen Klassen hin.

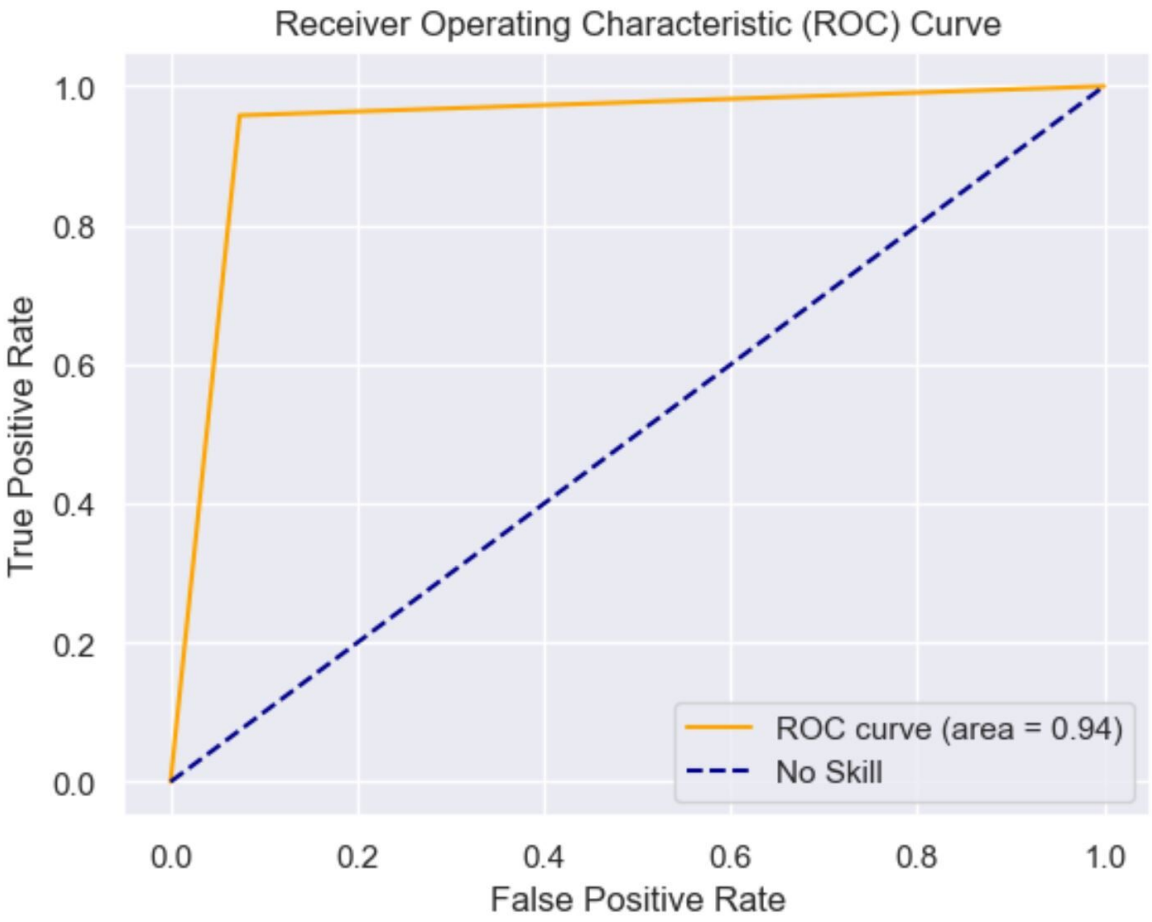


Abbildung 4.5: ROC-Kurve des regelbasierten Ansatzes zur Erkennung der Anomalie des zusätzlichen Gewichtes

Bei der Erkennung von Objekten, die fallen gelassen wurden, zeigt der regelbasierte Ansatz eine sehr hohe Genauigkeit von 0.9607 auf. Dies deutet darauf hin, dass in den meisten Fällen genau erkannt wird, wann ein Objekt fallen gelassen wurde. Die Präzision liegt bei 0.7408, ähnlich wie beim zusätzlichen Gewicht. Dies deutet ebenfalls auf eine moderate Anzahl von Fehlalarmen hin. Mit einem nahezu perfekten Recall von 0.9998 erkennt das System fast alle tatsächlichen Fälle, in denen die zweite Platte fallen gelassen worden ist. Dementsprechend zeigt das F-Maß mit einem Wert von 0.8510 eine sehr gute Gesamtleistung an. Die Auswertung deutet darauf hin, dass das System effektiv in der Erkennung dieser Anomalie arbeitet. Auch hier lässt sich dieses Ergebnis an der ROC-Kurve in Abbildung 4.6 ablesen. Eine AUC von 0.98 zeigt, dass das Modell eine exzellente Fähigkeit zur Unterscheidung zwischen normalen Datensätzen und Anomalien besitzt und damit äußerst zuverlässig und präzise in der Anomalieerkennung ist.

Die Ergebnisse für die Anomalie der reduzierten Geschwindigkeit zeigen eine perfekte Leistung des regelbasierten Ansatzes, da die Genauigkeit, Präzision, Recall und das F-Maß jeweils bei 100% liegen. Das bedeutet, dass alle Klassifizierungen korrekt sind und eine fehlerfreie Erkennung dieser Anomalie gewährleistet wird. Dies kann man ebenfalls an der perfekten AUC in Abbildung 4.7 erkennen.

Zusammenfassend verdeutlicht die Auswertung dieser Kennzahlen, dass der regelbasierte Ansatz sehr effektiv bei der Erkennung verschiedener Anomalien ist, wobei die Leistung je nach Komplexität

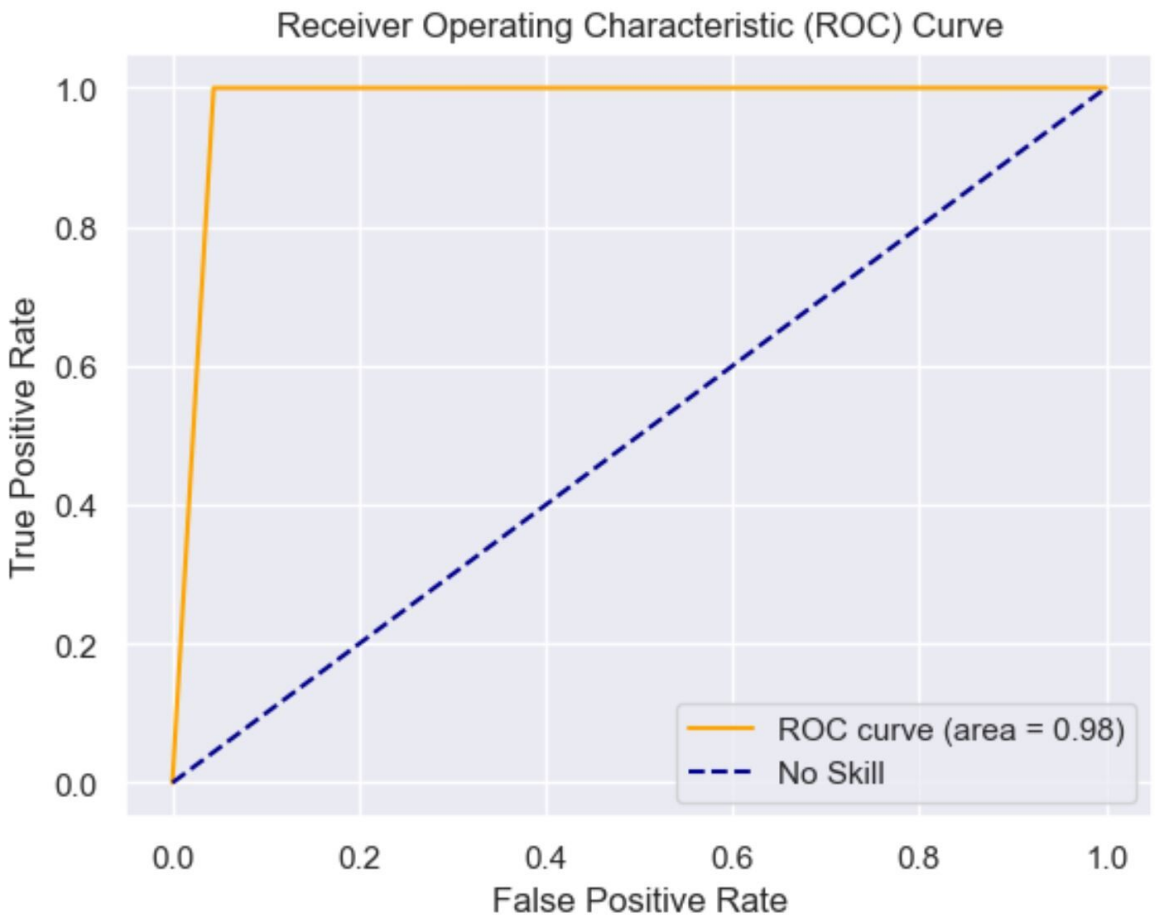


Abbildung 4.6: ROC-Kurve des regelbasierten Ansatzes zur Erkennung der Anomalie Objekt fallen lassen

der Anomalie variiert. Die Anomalien des zusätzlichen Gewichts und des fallen lassen eines Objektes zeigen eine hohe Genauigkeit und einen sehr hohen Recall. Jedoch erzielen sie niedrigere Präzisionswerte, was auf eine höhere Anzahl von Fehlalarmen hinweist. Im Gegensatz dazu zeigt die Anomalie der reduzierten Geschwindigkeit perfekte Werte in den Metriken Genauigkeit, Präzision, Recall und F-Maß, was auf eine herausragende Erkennungsleistung hinweist. Dies ist darauf zurückzuführen, dass für die Erkennung der Anomalie nur eine einfache Regel notwendig war. Im Großen und Ganzen zeigt die Analyse, dass der regelbasierte Ansatz besonders gut bei der Erkennung von reduzierter Geschwindigkeit ist, während es bei den anderen beiden Anomalien einige Fehlalarme gibt.

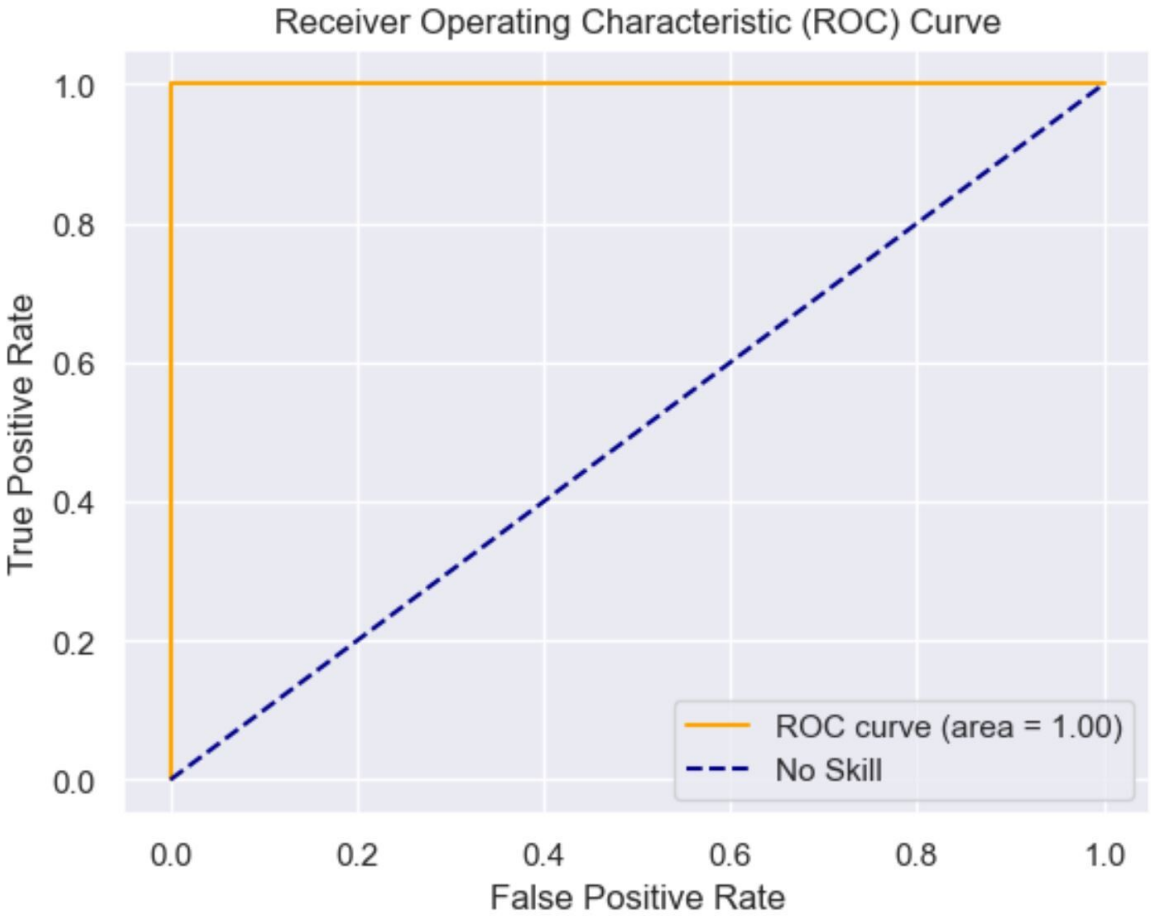


Abbildung 4.7: ROC-Kurve des regelbasierten Ansatzes zur Erkennung reduzierter Geschwindigkeiten

5

Machine-Learning-Ansatz

Dieses Kapitel behandelt den maschinellen Lernansatz für das Datenmonitoring von Roboteranwendungen. Zunächst wird das grundlegende Konzept dieses Ansatzes erläutert, wobei besonderes Augenmerk auf die Funktionsweise von Autoencodern gelegt wird. Daraufhin folgt die Implementierung, bei der die eingesetzten Technologien und die Datenaufbereitung detailliert beschrieben werden. Abschließend wird eine Evaluierung durchgeführt, um die Leistungsfähigkeit des maschinellen Lernansatzes zu bewerten.

5.1 Konzept

Das Konzept des maschinellen Lernansatzes beschreibt die Implementierung eines Überwachungsmechanismus, der mithilfe von Autoencodern komplexe Datenmuster analysiert und bewertet. Abbildung 5.1 veranschaulicht den grundlegenden Rahmen dieses Ansatzes.

Wie in Abbildung 5.1 dargestellt, beginnt der Prozess mit der Übertragung von Rohdaten, die den Status des Roboters beschreiben. Die Robotersteuerung leitet diese Daten entweder über eine direkte Verbindung wie Real-Time Data Exchange (RTDE) bei UR-Robotern oder über Kommunikationsprotokolle wie das Robot Operating System (ROS) weiter. Hier ist eine zusätzliche Vorverarbeitung erforderlich, um die Daten für die Ausführung durch das maschinelle Lernmodell kompatibel zu machen. Die Vorverarbeitungsschritte umfassen die Normalisierung, Datenumformung und Merkmalsextraktion der Eingabedaten. In der nachfolgenden Phase hat ein Autoencoder das Ziel, die Eingabedaten möglichst genau zu rekonstruieren, indem er seine gelernte Repräsentation zur Nachbildung der ursprünglichen Datenverteilung verwendet. Durch das Erlernen der Rekonstruktion normaler Muster in den Eingabedaten kann das Modell Anomalien erkennen, indem es Abweichungen zwischen den Originaldaten und den rekonstruierten Darstellungen identifiziert (Michelucci, 2022). Überschreitet diese Abweichung einen vordefinierten Schwellenwert, wird der betroffene Datensatz zurückgehalten und einer Expertenanalyse zur detaillierten Überprüfung unterzogen. Auf Basis der gesammelten Daten können Fachleute mit hoher Expertise in der Verifikation von Robotersystemen die Leistung des Robotersystems analysieren und bewerten.

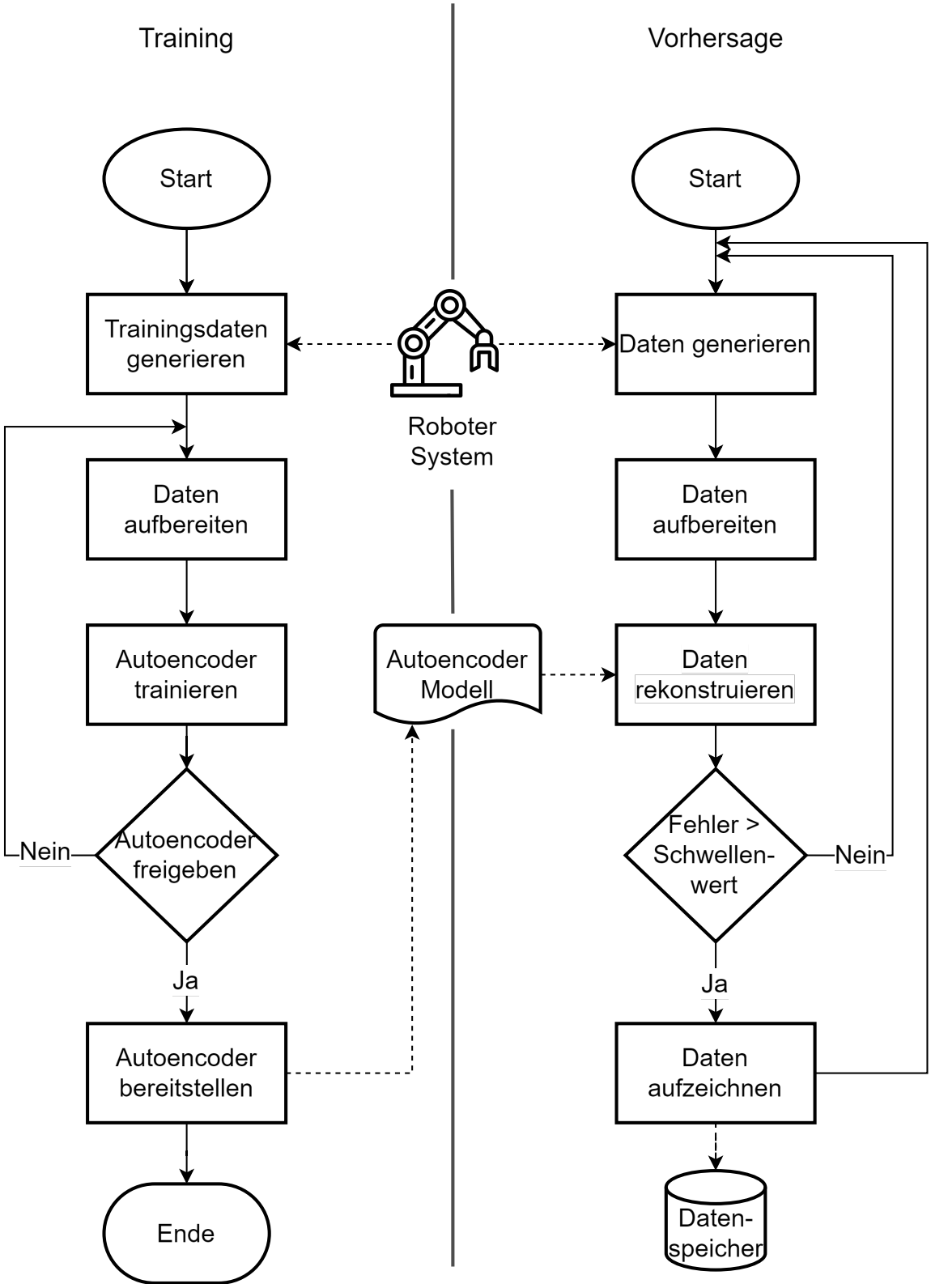


Abbildung 5.1: Konzeptueller Rahmen des maschinellen Lernansatzes

5.2 Autoencoder

Wie im vorhergehenden Konzept erläutert, kommt ein Autoencoder bei der Implementierung des maschinellen Ansatzes zum Einsatz. Ein Autoencoder (AE) ist ein neuronales Netzwerk, das darauf ausgelegt ist, Eingabedaten zu komprimieren und anschließend zu rekonstruieren. Er bestehen aus drei zentralen Komponenten: einem Encoder, einem latenten Raum und einem Decoder. Die

Eingabedaten werden im Encoder verarbeitet, der die wesentlichen Merkmale eines Prozesses lernt, wobei diese Merkmale in einer reduzierten Dimension dargestellt werden. Dieser latente Raum enthält demnach die komprimierte Wissensdarstellung der Eingabedaten. Der Decoder wiederum nutzt diese Merkmale, um die ursprünglichen Daten zu rekonstruieren (Mobtahej et al., 2022).

Der Rekonstruktionsfehler dient der Anomalieerkennung, indem die Differenz zwischen der ursprünglichen Eingabe und der rekonstruierten Ausgabe gemessen wird. Nachdem das Modell ausschließlich mit normalen Daten trainiert wird, resultiert die Anwendung des Modells auf anomalen oder neuen Daten in erheblichen Rekonstruktionsfehlern, da diese Daten während des Trainingsprozesses nicht berücksichtigt wurden (Chen et al., 2018).

Abbildung 5.2 visualisiert die Struktur eines Autoencoders mit vier Schichten im Encoder und Decoder.

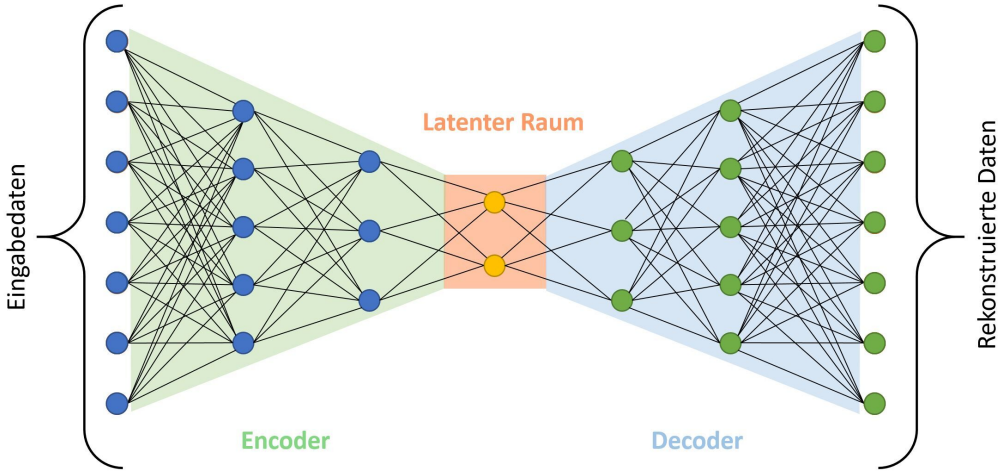


Abbildung 5.2: Struktur eines Autoencoders

Basierend auf der Dimensionalität der Kodierungsschicht lassen sich vier verschiedene Typen von Autoencodern unterscheiden. Unvollständige Autoencoder weisen eine Kodierungsschicht auf, die eine geringere Dimensionalität als die Eingabedaten besitzt. Vollständige Autoencoder hingegen haben eine Kodierungsschicht, die die gleiche oder eine höhere Dimensionalität als die Eingabe aufweist. Flache Autoencoder bestehen lediglich aus drei Schichten: Eingabe-, Kodierungs- und Ausgabeschicht, während tiefe Autoencoder mehr als eine versteckte Schicht enthalten (Charte et al., 2018). Die Abbildungen 5.3 und 5.4 illustrieren diese vier Typen von Autoencodern.

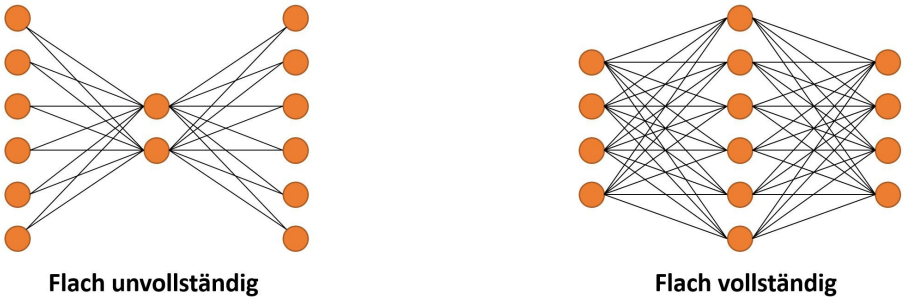


Abbildung 5.3: Flache Autoencoder Struktur (Charte et al., 2018)

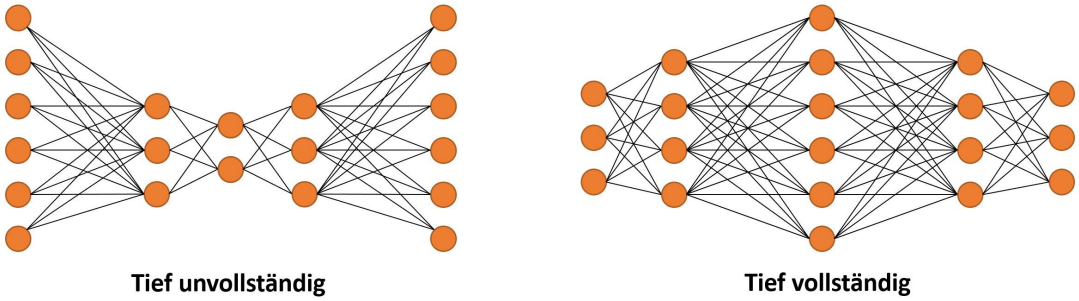


Abbildung 5.4: Tiefe Autoencoder Struktur (Charte et al., 2018)

5.3 Implementierung

Die Implementierung des Machine-Learning Ansatzes, erfolgt in der Programmiersprache Python. Diese Entscheidung wurde aufgrund der Vielseitigkeit und Leistungsfähigkeit dieser Programmiersprache in maschinellen Lernaufgaben getroffen. Für die konkrete Umsetzung wurde TensorFlow ausgewählt. TensorFlow ist eine der führende Bibliothek für maschinelles Lernen und neuronale Netzwerke und bietet eine umfangreiche Sammlung von Werkzeugen und Funktionen für das Trainieren von Modellen (Abadi et al., 2016).

5.3.1 Datenvorbereitung

Datensammlung

Die Datensammlung ist ein grundlegender und entscheidender Schritt bei der Implementierung eines Autoencoders. In diesem Prozess werden die notwendigen Daten aufgezeichnet, die anschließend für das Training und die Evaluierung des Modells verwendet werden. Die Datenerfassung erfolgt direkt vom Robotersystem unter Verwendung des Real-Time Data Exchange (RTDE) Protokolls. Hierbei werden die Daten durch die kontinuierliche Ausführung des Anwendungsfalls in Echtzeit übertragen und mittels RTDE in eine CSV-Datei gespeichert.

Die gesammelten Messwerte umfassen verschiedene Arten von Zeitreihendaten. Zu den wesentlichen erfassten Daten gehören:

- Positionsdaten: Gelenkpositionen des Roboters
- Geschwindigkeitsdaten: Gelenkgeschwindigkeiten und Gelenkströme
- Kraft- und Momentdaten: Kräfte und Momente, die auf den Roboterarm wirken
- Systemstatus: Informationen über den aktuellen Betriebszustand des Roboters

Diese Daten werden mit einer Frequenz von 125 Hz erfasst, was eine hohe zeitliche Auflösung und Genauigkeit gewährleistet. Eine detaillierte Auflistung der über RTDE empfangenen Daten befindet sich im Anhang dieser Arbeit. Insgesamt wurden 137 Merkmale über 100 Durchläufe hinweg aufgezeichnet.

Datenvorverarbeitung

Nachdem die Daten aufgezeichnet wurden, erfolgt im nächsten Schritt die Vorverarbeitung dieser. Die Vorverarbeitung ist entscheidend, um die Rohdaten für die Modellierung geeignet zu machen. Dieser Prozess umfasst mehrere Schritte, darunter die Datenbereinigung und die Normalisierung.

Die *Datenbereinigung* umfasst die Identifikation und Beseitigung ungenauer oder irrelevanter Datenpunkte. Dazu gehört die Entfernung von Ausreißern und fehlerhaften Daten, wobei Datenpunkte identifiziert und entfernt werden, die außerhalb des erwarteten Bereichs liegen oder offensichtliche Fehler aufweisen. Diese Maßnahmen sind entscheidend, um Verzerrungen im Modell zu vermeiden (Ilyas & Chu, 2019). In dieser Arbeit wurde zunächst überprüft, ob fehlende Werte in den Datenaufzeichnungen vorhanden sind. Anschließend erfolgte eine gründliche Analyse auf Ausreißer unter Verwendung statistischer Methoden, wie dem Box-Plot. Zudem wurde die Daten auf Konsistenz überprüft, um sicherzustellen, dass sie den erwarteten Formaten und Bereichen entsprechen. Durch diese systematische Bereinigung wurde eine solide Grundlage für die weitere Analyse und Modellierung geschaffen.

Anschließend wurden die Daten *normalisiert*, um zu gewährleisten, dass alle Merkmale auf einer einheitlichen Skala liegen. Sollten Merkmale mit stark unterschiedlichen Skalen in das Modell eingespeist werden, könnte dies dazu führen, dass das neuronale Netz die Merkmale ungleich gewichtet. Dies wiederum kann eine falsche Priorisierung von Merkmalen mit höheren Werten gegenüber anderen verursachen und den Lernprozess beeinträchtigen (D. Singh & Singh, 2020). Um dies zu verhindern, wurde ein Min-Max-Skalierer verwendet, der alle Merkmale auf einen Bereich von null bis eins skaliert. Dies wird erreicht, indem für jedes Merkmal der minimale und maximale Wert ermittelt wird und die Datenpunkte anschließend entsprechend skaliert werden.

Der Min-Max-Skalierer funktioniert nach der Formel (Thara et al., 2019):

$$x_{skaliert} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Dabei ist x der ursprüngliche Wert, min(x) der minimale Wert des Merkmals und max(x) der maximale Wert. Durch diese Transformation wird sichergestellt, dass alle Datenpunkte im gleichen

Wertebereich liegen, was insbesondere für Algorithmen des maschinellen Lernens von großer Bedeutung ist, da viele Modelle empfindlich auf die Skala der Eingabedaten reagieren (Thara et al., 2019).

Zeitreihen-Formatierung

Die Erfassung von Roboterdaten über RTDE erfolgte mit einer Frequenz von 125 Hz, was bedeutet, dass die Daten alle 8 Millisekunden erfasst wurden. Dadurch entstanden Zeitreihendaten, die für die Analyse und Modellierung von erheblicher Bedeutung sind. Um diese Daten jedoch für das Training und die Evaluierung des Modells nutzbar zu machen, müssen sie in ein geeignetes Format umgewandelt werden. Diese Umformung ist entscheidend, um sicherzustellen, dass das Modell zeitliche Abhängigkeiten und Muster effektiv erfassen kann.

Der Einsatz eines LSTM-Autoencoders bietet sich hierbei besonders an, da dieses Modell in der Lage ist, zeitliche Abhängigkeiten zu erkennen und komplexe Muster innerhalb der Daten zu verarbeiten. Dabei werden die Fähigkeit des LSTM, Muster über längere Sequenzen zu erfassen, sowie die Eigenschaften des Autoencoders, um aus normalen Sequenzen zu lernen und diese zu rekonstruieren, genutzt. Für vertiefte Informationen zu LSTM-Modellen und deren Funktionsweise sei auf die einschlägige Fachliteratur verwiesen (Hochreiter & Schmidhuber, 1997).

Die Eingabe für den LSTM-Autoencoder erfolgt in Form einer dreidimensionalen Matrix, die die Stapelgröße, die Fenstergröße und die Attributgröße umfasst. Die Stapelgröße definiert, wie viele Teilsequenzen in einem Batch verwendet werden, während die Fenstergröße angibt, wie viele aufeinanderfolgende Datensätze innerhalb jeder Teilsequenz berücksichtigt werden. Die Attributgröße bestimmt schließlich die Anzahl der Merkmale pro Datensatz in der Teilsequenz (Homayouni et al., 2020).

Auswahl der Merkmale

Ein wesentlicher Aspekt beim maschinellen Lernen ist die Auswahl der Merkmale. Eine sorgfältige Merkmalsauswahl trägt dazu bei, die Modellleistung zu optimieren und die Komplexität zu reduzieren. Durch die Identifizierung relevanter Merkmale wird sichergestellt, dass nur die signifikantesten Informationen in den Lernprozess einfließen, was zu robusteren und effizienteren Modellen führt. (Jović et al., 2015).

In dieser Arbeit wurden drei Methoden zur Merkmalsauswahl eingesetzt: Varianzschwellenwertverfahren, Hauptkomponenten-Analyse und regelbasierte Merkmalsauswahl.

Das *Varianzschwellenwertverfahren* entfernt Merkmale mit geringer Varianz, die wenig zur Differenzierung der Daten beitragen. In der Statistik wird die Varianz genutzt, um die Streuung eines Datensatzes zu messen und den Varianzwert eines Merkmals zu bestimmen (Lane et al., 2017).

$$\sigma^2 = \frac{\Sigma(x-\mu)^2}{N}$$

Der Varianzwert eines Merkmals wird wie in Formel 5.3.1 berechnet, wobei μ den Mittelwert und N die Anzahl der Datenzeilen darstellen. Merkmale, deren Varianzwert den festgelegten Schwellenwert nicht erreicht, werden verworfen (Lane et al., 2017). Diese Daten sind typischerweise über den

gesamten Datensatz hinweg konstant oder nahezu konstant und bieten daher wenig wertvolle Informationen für Modellierungsaufgaben (DataScience-ProF, 2024).

Die *Hauptkomponenten-Analyse* oder auch Principal Component Analysis (PCA) genannt, ist eine Methode zur Umwandlung eines Satzes von korrelierten Variablen in eine kleinere Menge von unkorrelierten Variablen, bekannt als Hauptkomponenten (PCs). Jeder PC ist eine Linearkombination der ursprünglichen Variablen, wobei die Gewichtung der Variablen durch ihre Bedeutung bestimmt wird. Dies ermöglicht es, die ursprünglichen Variablen in einem neuen Koordinatensystem darzustellen, das die wesentlichen Strukturen und Muster der Daten aufzeigt. Die Anzahl der Hauptkomponenten ist typischerweise kleiner als die Anzahl der ursprünglichen Variablen, was zu einer Reduktion der Dimensionalität führt und dabei hilft, die bedeutendsten Informationen der Daten zu bewahren (Boutsidis et al., 2008).

Die *regelbasierte Merkmalsauswahl* zielt darauf ab, relevante Merkmale zu identifizieren und irrelevante zu eliminieren, indem sie spezifische Parameter aus den aufgestellten Regeln verwendet. Hierbei wird das Domänenwissen genutzt, das bereits zur Bildung der Regeln in Kapitel 4.4.1 verwendet wurde. Dieser Ansatz ermöglicht es, gezielt diejenigen Merkmale auszuwählen, die signifikant zur Erklärung der Datenvariation beitragen und somit die Modellgenauigkeit verbessern können.

5.3.2 Modellarchitekturen

Es wurden vier verschiedene Modellarchitekturen zur Anomalieerkennung in multivariaten Zeitreihendaten evaluiert. Zu diesem Zweck wurden LSTM-basierte Autoencoder mit unterschiedlicher Schichtenzahl im Encoder und Decoder konzipiert. Konkret handelt es sich um folgende Architekturen:

- Modellarchitektur A: Eine LSTM-Schicht im Encoder und Decoder.
- Modellarchitektur B: Zwei LSTM-Schichten im Encoder und Decoder.
- Modellarchitektur C: Drei LSTM-Schichten im Encoder und Decoder.
- Modellarchitektur D: *n* LSTM-Schichten.

Die Modellarchitektur D besteht aus einer variablen Anzahl von LSTM-Schichten, wobei *n* durch Hyperparametertraining optimiert wird.

Zusätzlich zu den Modellarchitekturen wurden weitere Modellparameter festgelegt, um die Leistungsfähigkeit der Autoencoder zu optimieren:

Der mittlere quadratische Fehler (Mean Squared Error, MSE) wurde als Verlustfunktion festgelegt, da Autoencoder in der Regel ein Regressionsproblem lösen. Der MSE berechnet sich wie folgt:

$$L_{MSE} = MSE = \frac{1}{M}\sum_{i=1}^{M}(x_i - \tilde{x}_i)^2$$

wobei M die Anzahl der Proben in jeder Sequenz, x_i die Eingabe des Encoders und $\tilde{x}_i$ die Ausgabe des Decoders darstellt (Michelucci, 2022).

Zudem wurden in den Modellen der Adam-Optimierer eingesetzt, um die Effizienz des Algorithmus zu steigern und die Zeitkomplexität zu reduzieren. Der Adam-Optimierungsalgorithmus stellt eine fortschrittliche Alternative zur traditionellen stochastischen Gradientenabstiegsmethode dar und ermöglicht eine effektive Aktualisierung der Gewichte des Trainingsnetzwerks (Kingma & Ba, 2014).

5.3.3 Modelltraining

Aufteilung des Datensatzes in Trainings- und Testsets

Das Training des Modells beginnt mit der Aufteilung des Datensatzes in Trainings- und Testsets. Der Trainingsdatensatz umfasst 80 Prozent der Gesamtmenge und hat eine Dimension von 686346. Der Testdatensatz, der die verbleibenden 20 Prozent ausmacht, weist eine Dimension von 168262 auf.

Hyperparameterauswahl

Die Leistung neuronaler Netzwerke hängt stark von der Wahl der Hyperparameter ab (Claesen & De Moor, 2015). Daher ist es entscheidend, eine geeignete Strategie zur Hyperparameterauswahl bei der Entwicklung von Autoencodern zu verwenden. In dieser Arbeit kommt Hyperband (Li et al., 2018) aus dem Keras-Tunermodul (O'Malley et al., 2019) zum Einsatz, um Netzwerkparameter wie die Anzahl der Neuronen pro Schicht, Aktivierungsfunktion und Lernrate zu optimieren. Hyperband wurde gewählt, weil es Konfigurationen schnell testet und nicht vielversprechende Konfigurationen zügig eliminiert. Es ist wichtig zu beachten, dass die optimalen Hyperparameter nicht für alle Datensätze oder Aufgaben gleich sind und daher von Fall zu Fall bestimmt werden müssen (K. Singh et al., 2023).

Folgende Hyperparameter wurden für die beschriebenen Modellarchitekturen ermittelt. Die Tabellen geben Aufschluss über die Schichten, deren Größe und die verwendeten Aktivierungsfunktionen.

Tabelle 5.1 zeigt die Parameter der Modellarchitektur A unter Verwendung der Merkmale der Varianzschwellenwertmethode.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 80)	
lstm_2 (LSTM)	(64)	reLu
repeat_vector_1 (RepeatVector)	(10, 64)	-
lstm_3 (LSTM)	(10, 64)	reLu
time_distributed_1 (TimeDistributed)	(10, 80)	linear

Tabelle 5.1: Implementierte LSTM-Netzstruktur von Modellarchitektur A und Varianzschwellenwertverfahren

Tabelle 5.2 gibt einen Überblick über die Parameter der Modellarchitektur A unter Verwendung der Merkmale der PCA-Methode.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 26)	
lstm_2 (LSTM)	(64)	reLu
repeat_vector_1 (RepeatVector)	(10, 64)	-
lstm_3 (LSTM)	(10, 64)	reLu
time_distributed_1 (TimeDistributed)	(10, 26)	linear

Tabelle 5.2: Implementierte LSTM-Netzstruktur von Modellarchitektur A und PCA

In Tabelle 5.3 sind die Parameter der Modellarchitektur A aufgeführt, das die Merkmale der regelbasierten Methode verwendet.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 4)	
lstm_2 (LSTM)	(56)	reLu
repeat_vector_1 (RepeatVector)	(10, 56)	-
lstm_3 (LSTM)	(10, 56)	reLu
time_distributed_1 (TimeDistributed)	(10, 4)	linear

Tabelle 5.3: Implementierte LSTM-Netzstruktur von Modellarchitektur A und regelbasierter Merkmalsauswahl

Die Parameter von Modellarchitektur B werden für die Merkmalsauswahl der Varianzschwellenwertmethode in Tabelle 5.4 dargestellt.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 80)	
lstm_2 (LSTM)	(10, 64)	tanh
lstm_3 (LSTM)	(64)	reLu
repeat_vector_1 (RepeatVector)	(10, 64)	-
lstm_4 (LSTM)	(10, 64)	reLu
lstm_5 (LSTM)	(10, 80)	sigmoid
dense_1 (Dense)	(10, 80)	linear

Tabelle 5.4: Implementierte LSTM-Netzstruktur von Modellarchitektur B und Varianzschwellenwertverfahren

In Tabelle 5.5 sind die Parameter von Modellarchitektur B aufgeführt, die bei der Merkmalsauswahl mittels der PCA zum Einsatz kommen.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 26)	
lstm_2 (LSTM)	(10, 48)	sigmoid
lstm_3 (LSTM)	(48)	tanh
repeat_vector_1 (RepeatVector)	(10, 48)	-
lstm_4 (LSTM)	(10, 48)	sigmoid
lstm_5 (LSTM)	(10, 26)	tanh
dense_1 (Dense)	(10, 26)	linear

Tabelle 5.5: Implementierte LSTM-Netzstruktur von Modellarchitektur B und PCA

Modellarchitektur B und seine Parameter, die für die Merkmalsauswahl mittels der regelbasierten Methode verwendet wurden, sind in Tabelle 5.6 dargestellt.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 4)	
lstm_2 (LSTM)	(10, 40)	reLu
lstm_3 (LSTM)	(40)	reLu
repeat_vector_1 (RepeatVector)	(10, 40)	-
lstm_4 (LSTM)	(10, 40)	tanh
lstm_5 (LSTM)	(10, 4)	tanh
dense_1 (Dense)	(10, 4)	linear

Tabelle 5.6: Implementierte LSTM-Netzstruktur von Modellarchitektur B und regelbasierter Merkmalsauswahl

Die Parameter von Modellarchitektur C, die bei der Merkmalsauswahl durch die Varianzschwellenwertmethode verwendet werden, sind in Tabelle 5.7 dargestellt.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 80)	
lstm_2 (LSTM)	(10, 48)	reLu
lstm_3 (LSTM)	(10, 48)	reLu
lstm_4 (LSTM)	(48)	reLu
repeat_vector_1 (RepeatVector)	(10, 48)	-
lstm_5 (LSTM)	(10, 48)	reLu
lstm_6 (LSTM)	(10, 48)	reLu
lstm_7 (LSTM)	(10, 80)	reLu
dense_1 (Dense)	(10, 80)	reLu

Tabelle 5.7: Implementierte LSTM-Netzstruktur von Modellarchitektur C und Varianzschwellenwertverfahren

In Tabelle 5.8 sind die Parameter von Modellarchitektur C aufgeführt, die auf der Merkmalsauswahl durch die PCA-Methode basieren.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 26)	
lstm_2 (LSTM)	(10, 56)	reLu
lstm_3 (LSTM)	(10, 56)	reLu
lstm_4 (LSTM)	(56)	reLu
repeat_vector_1 (RepeatVector)	(10, 56)	-
lstm_5 (LSTM)	(10, 56)	reLu
lstm_6 (LSTM)	(10, 56)	reLu
lstm_7 (LSTM)	(10, 26)	reLu
dense_1 (Dense)	(10, 26)	reLu

Tabelle 5.8: Implementierte LSTM-Netzstruktur von Modellarchitektur C und PCA

Tabelle 5.9 zeigt die Parameter von Modellarchitektur C unter Verwendung der regelbasierten Merkmalsauswahl.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 4)	
lstm_2 (LSTM)	(10, 56)	reLu
lstm_3 (LSTM)	(10, 56)	reLu
lstm_4 (LSTM)	(56)	reLu
repeat_vector_1 (RepeatVector)	(10, 56)	-
lstm_5 (LSTM)	(10, 56)	reLu
lstm_6 (LSTM)	(10, 56)	reLu
lstm_7 (LSTM)	(10, 4)	reLu
dense_1 (Dense)	(10, 4)	reLu

Tabelle 5.9: Implementierte LSTM-Netzstruktur von Modellarchitektur C und regelbasierter Merkmalsauswahl

In Tabelle 5.10 werden die Parameter der n-Schichten Architektur unter Verwendung der Varianzschwellenwertmethode dargestellt.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 80)	
lstm_2 (LSTM)	(10, 512)	tanh
lstm_3 (LSTM)	(10, 320)	tanh
lstm_4 (LSTM)	(10, 512)	tanh
dropout_1 (Dropout)	(10, 512)	-
dense_1 (Dense)	(10, 80)	reLu

Tabelle 5.10: Implementierte LSTM-Netzstruktur von Modellarchitektur D und Varianzschwellenwertverfahren

Tabelle 5.11 zeigt die Parameter der n-schichtigen Architektur, die mithilfe der PCA-Methode ermittelt wurden.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 26)	
lstm_2 (LSTM)	(10, 128)	tanh
lstm_3 (LSTM)	(10, 448)	tanh
lstm_4 (LSTM)	(10, 352)	tanh
lstm_5 (LSTM)	(10, 224)	tanh
lstm_6 (LSTM)	(10, 224)	tanh
lstm_7 (LSTM)	(10, 288)	tanh
dropout_1 (Dropout)	(10, 288)	-
dense_1 (Dense)	(10, 26)	reLu

Tabelle 5.11: Implementierte LSTM-Netzstruktur von Modellarchitektur D und PCA

Die Parameter von Modellarchitektur D unter Anwendung der regelbasierten Merkmalsauswahl werden in Tabelle 5.12 abgebildet.

Schicht	Schichtgröße	Aktivierungsfunktion
input_layer_1 (InputLayer)	(10, 4)	
lstm_2 (LSTM)	(10, 256)	tanh
lstm_3 (LSTM)	(10, 320)	tanh
lstm_4 (LSTM)	(10, 480)	tanh
dropout_1 (Dropout)	(10, 480)	-
dense_1 (Dense)	(10, 4)	reLu

Tabelle 5.12: Implementierte LSTM-Netzstruktur von Modellarchitektur D und regelbasierter Merkmalsauswahl

Training

Nachdem die optimalen Hyperparameter ermittelt wurden, muss das Training durchgeführt werden. Alle Varianten des LSTM-Autoencoders wurden für 50 Epochen mit einer Batchgröße von 128 trainiert. Durch die Anwendung der Early-Stopping-Methode wurde sichergestellt, dass das Modell nicht zu lange trainiert wurde, um Überanpassung und eine schlechte Generalisierung auf neue Daten zu vermeiden. Early-Stopping beendet das Training, sobald sich die Leistung auf einem Validierungsdatensatz nicht weiter verbessert (Prechelt, 2002).

Schwellenwert definieren

Zur Erkennung von Anomalien ist es erforderlich, einen Schwellenwert festzulegen, der als Referenzpunkt für den rekonstruierten Fehler dient. Durch den Vergleich des rekonstruierten Fehlers mit diesem Schwellenwert können diejenigen Beobachtungen identifiziert werden, deren rekonstruierter Fehler den Schwellenwert übersteigt (Nortey et al., 2021). Folgende Gleichung veranschaulicht diesen Sachverhalt.

$$X' = \begin{cases} X_i & \text{wenn der rekonstruierte Fehler } > \text{Schwellenwert (Anomalie)} \\ X_i & \text{wenn der rekonstruierte Fehler } \leq \text{Schwellenwert (Normal)} \end{cases} \tag{5.1}$$

Die rekonstruierten Daten werden als X' und die Eingabedaten als X_i bezeichnet. Das wesentliche Problem besteht darin, den optimalen Schwellenwert zu bestimmen, um falsch-positive Ergebnisse zu minimieren und gleichzeitig die Mehrheit der Anomalien zu erfassen. Zur Lösung dieses Problems ist es notwendig, die Entscheidungsgrenze zwischen normalen Daten und Anomalien zu schätzen. In diesem Zusammenhang wird eine Quantilmethode auf die rekonstruierten Fehler angewandt, um einen festen Vertrauensbereich für Anomalien festzulegen (Nortey et al., 2021). Das 90%-Quantil wird hierbei als Grenze verwendet. Dieses Quantil ermöglicht es, einen spezifischen Anteil der Anomalien basierend auf einem festgelegten Wert zu identifizieren. Übersteigt demnach der rekonstruierte Fehler den Schwellenwert, wird der Datenpunkt als anomal klassifiziert. Liegt der Fehler dagegen unterhalb des Schwellenwerts, wird der Datenpunkt als normal betrachtet.

5.4 Evaluierung

Die Modelle wurden sowohl in Bezug auf ihre Architektur als auch auf die Auswahl der Merkmale analysiert. Im folgenden Abschnitt werden die Kennzahlenauswertung für jedes Modell und jede Anomalie übersichtlich in Tabellenform dargestellt.

Modellarchitektur A, das aus jeweils einer Schicht im Encoder und Decoder besteht, erzielt für die Anomalie des zusätzlichen Gewichts die in Tabelle 5.13 angeführten Kennzahlenwerte.

	Varianzschwellenwert	**PCA**	**Regelbasierte Merkmale**
Genauigkeit	0.7469	0.7504	0.7456
Präzision	0.2066	0.1602	0.2085
Recall	0.1395	0.0888	0.1442
F-Maß	0.1665	0.1143	0.1705
ROC-AUC	0.5104	0.4929	0.5115

Tabelle 5.13: Kennzahlenauswertung von Modellarchitektur A für die Anomalie des zusätzlichen Gewichts mit unterschiedlicher Merkmalsauswahl

Die Genauigkeit der Klassifikation zeigt sich für Modellarchitektur A nahezu gleichbleibend über alle Methoden hinweg. In Bezug auf die Präzision zeigt die regelbasierte Merkmalsauswahl mit einem Wert von 0.2085 die höchste Leistung, während PCA mit einem Wert von 0.1602 die niedrigste Präzision aufweist. Im Bereich Recall und F-Maß erzielt ebenfalls die regelbasierte Methode die höchsten Werte von 0.1442 und 0.1705. Hierbei bestätigt der niedrige Wert des F-Maßes die unausgewogene Leistung zwischen Präzision und Recall. Für die ROC-AUC-Metrik, welche die Fähigkeit des Modells zur Trennung von Klassen misst, erreicht ebenfalls die regelbasierte Methode den höchsten Wert mit 0.5115. Diese Leistung deutet jedoch darauf hin, dass das Modell nur wenig besser als die zufällige Klassifikation abschneidet.

Modellarchitektur A, das über je eine Schicht im Encoder und Decoder verfügt, liefert für die Anomalie des Fallens von Objekten die in Tabelle 5.14 dargestellten Kennzahlen.

	Varianzschwellenwert	**PCA**	**Regelbasierte Merkmale**
Genauigkeit	0.7884	0.8216	0.8127
Präzision	0.1525	0.2232	0.1996
Recall	0.1945	0.2377	0.2222
F-Maß	0.1710	0.2302	0.2103
ROC-AUC	0.5290	0.5666	0.5548

Tabelle 5.14: Kennzahlenauswertung von Modellarchitektur A für die Anomalie des Fallens von Objekten mit unterschiedlicher Merkmalsauswahl

Die Genauigkeit der Modellarchitektur A bei der Erkennung von fallen gelassenen Objekten ist relativ hoch. Die höchste Genauigkeit erreicht PCA mit 0.8216, gefolgt von der regelbasierten Merkmalsauswahl und dem Varianzschwellenwertverfahren. In Bezug auf die Präzision zeigt PCA mit 0.2232 ebenfalls die beste Leistung. Auch beim Recall schneidet die PCA-Methode mit 0.2377 besser als die anderen beiden Methoden ab. Die regelbasierte Merkmalsauswahl erreicht hierbei einen Wert von 0.2222, während der Varianzschwellenwert mit 0.1945 den niedrigsten Recall aufweist. Das F-Maß ist bei PCA mit 0.2302 am stärksten ausgeprägt. Zudem erzielt PCA mit 0.5666 die höchste AUC, gefolgt von den regelbasierten Merkmalen. Zusammengefasst zeigen die Ergebnisse jedoch, dass das Modell insbesondere in Bezug auf Präzision und Recall Schwächen aufweist.

Modellarchitektur A zeigt für die Anomalie der reduzierten Geschwindigkeit die Kennzahlen, die in Tabelle 5.15 zusammengefasst sind. In diesem Fall wurde keine regelbasierte Merkmalsauswahl durchgeführt, da diese Methode nur eine einzelne Variable betreffen würde, was in diesem Kontext nicht relevant ist.

	Varianzschwellenwert	**PCA**
Genauigkeit	0.0466	0.0455
Präzision	0.0157	0.0155
Recall	0.9443	0.9355
F-Maß	0.0308	0.0305
ROC-AUC	0.4881	0.4832

Tabelle 5.15: Kennzahlenauswertung von Modellarchitektur A für die Anomalie der reduzierten Geschwindigkeit mit unterschiedlicher Merkmalsauswahl

Die Kennzahlenauswertung von Modellarchitektur A für die Anomalie der reduzierten Geschwindigkeit zeigen insgesamt geringe Werte in der Auswertung der Kennzahlen. Die Genauigkeit des Modells ist mit einem Wert von 0.0466 für die Varianzschwellenwertmethode und 0.0455 für die PCA-Methode sehr niedrig. Ähnlich niedrig sind die Präzisionswerte, die bei 0.0157 und

0.0155 liegen. Im Gegensatz dazu weist das Modell jedoch einen sehr hohen Recall auf. Dieser beträgt bei der Verwendung des Varianzschwellenwerts 0.9443 und 0.9355 bei PCA. Dies deutet darauf hin, dass das Modell dazu in der Lage ist, die meisten tatsächlichen Anomalien korrekt zu erkennen. Das F-Maß ist hingegen gering mit 0.0308 für den Varianzschwellenwert und 0.0305 für PCA. Die ROC-AUC-Werte liegen bei 0.4881 und 0.4832, was auf eine begrenzte Fähigkeit zur Klassentrennung hinweist.

Modellarchitektur B, das aus jeweils zwei Schichten im Encoder und Decoder besteht, erreicht für die Anomalie des zusätzlichen Gewichts die in Tabelle 5.16 angeführten Kennzahlen.

	Varianzschwellenwert	**PCA**	**Regelbasierte Merkmale**
Genauigkeit	0.6647	0.7674	0.7723
Präzision	0.1118	0.1282	0.4250
Recall	0.1223	0.0488	0.7261
F-Maß	0.1168	0.0707	0.5361
ROC-AUC	0.4536	0.4877	0.7543

Tabelle 5.16: Kennzahlenauswertung von Modellarchitektur B für die Anomalie des zusätzlichen Gewichts mit unterschiedlicher Merkmalsauswahl

Die Auswertung der Kennzahlen von Modellarchitektur B zur Anomalie des Fallens von Objekten belegen, dass die regelbasierte Merkmalsauswahl insgesamt die beste Leistung erzielt. Mit einer Genauigkeit von 0.7723 erreicht sie den höchsten Wert, gefolgt von PCA mit 0.7674 und dem Varianzschwellenwert mit 0.6647. Die Präzision ist ebenfalls bei der regelbasierten Methode mit 0.4250 höher als bei den anderen beiden Methoden. Auch beim Recall erzielt die regelbasierte Auswahl mit 0.7261 den besten Wert. Ebenso ist das F-Maß bei der regelbasierten Methode mit 0.5361 am höchsten. Schließlich zeigt auch die Auswertung der AUC, dass die regelbasierte Methode mit 0.7543 am effektivsten ist. Die anderen beiden Methoden schneiden deutlich schlechter ab. Zusammengefasst deutet dies darauf hin, dass die regelbasierte Merkmalsauswahl die beste Leistung bei dieser Art von Anomalie liefert, während PCA und der Varianzschwellenwert insgesamt schwächere Resultate erzielen.

Modellarchitektur B, ausgestattet mit jeweils zwei Schichten im Encoder und Decoder, zeigt die Auswertung der Kennzahlen der Anomalie des Fallens eines Objektes in Tabelle 5.17.

Die Kennzahlenwerte von Modellarchitektur B zur Anomalie des Fallens von Objekten zeigen, dass auch hier die regelbasierte Merkmalsauswahl die beste Gesamtleistung erzielt. Mit einer Genauigkeit von 0.8504, einer Präzision von 0.3974 und einem Recall von 0.6456 erzielt diese Methode die höchsten Werte in diesen Kennzahlen. Das F-Maß erreicht bei der regelbasierten Methoden einen Wert von 0.4920, was auf eine gute Balance zwischen Präzision und Recall hinweist. Zudem zeigt die AUC von 0.7609, dass die regelbasierte Merkmalsauswahl eine gute Fähigkeit zur Unterscheidung zwischen Anomalien und normalen Datensätzen besitzt. Im Vergleich dazu schneiden PCA und der Varianzschwellenwert schlechter ab, insbesondere in Bezug auf Präzision und Recall.

	Varianzschwellenwert	PCA	Regelbasierte Merkmale
Genauigkeit	0.7559	0.8132	0.8504
Präzision	0.2432	0.0788	0.3974
Recall	0.5564	0.0622	0.6456
F-Maß	0.3385	0.0695	0.4920
ROC-AUC	0.6688	0.4851	0.7609

Tabelle 5.17: Kennzahlenauswertung von Modellarchitektur B für die Anomalie des Fallens von Objekten mit unterschiedlicher Merkmalsauswahl

Modellarchitektur B erzielt für die Anomalie der reduzierten Geschwindigkeit folgende Kennzahlenwerte.

	Varianzschwellenwert	PCA
Genauigkeit	0.0454	0.9465
Präzision	0.0165	0.2307
Recall	1.0	1.0
F-Maß	0.0325	0.3750
ROC-AUC	0.5149	0.9728

Tabelle 5.18: Kennzahlenauswertung von Modellarchitektur B für die Anomalie der reduzierten Geschwindigkeit mit unterschiedlicher Merkmalsauswahl

Die Analyse der Kennzahlen von Modellarchitektur B für die Anomalie der reduzierten Geschwindigkeit verdeutlicht, dass PCA in nahezu allen Metriken erheblich überlegen ist im Vergleich zum Varianzschwellenwertverfahren. PCA erreicht eine Genauigkeit von 0.9465, während der Varianzschwellenwert lediglich einen Wert von 0.0454 erzielt. Bei der Präzision weist PCA mit 0.2307 ebenfalls einen deutlich höheren Wert auf. Beide Methoden erzielen jedoch einen identischen Recall-Wert von 1.0. Dies deutet darauf hin, dass beide Ansätze sämtliche tatsächlichen Anomalien korrekt klassifiziert haben. Das F-Maß ist bei PCA mit 0.3750 signifikant höher als der Wert von 0.0325 für den Varianzschwellenwert. Darüber hinaus erreicht die PCA-Methode eine AUC von 0.9728, was auf eine herausragende Fähigkeit zur Trennung zwischen Anomalien und normalen Datensätzen hinweist. Zusammenfassend demonstrieren die Ergebnisse, dass PCA in Kombination mit Modellarchitektur B eine gute Methode zur Erkennung der Anomalie der reduzierten Geschwindigkeit darstellt.

Für die Anomalie des zusätzlichen Gewichts zeigt die Auswertung von Modellarchitektur C, das über jeweils drei Schichten im Encoder und Decoder verfügt, die in Tabelle 5.19 dargestellten Kennzahlen.

	Varianzschwellenwert	PCA	Regelbasierte Merkmale
Genauigkeit	0.6514	0.7379	0.7956
Präzision	0.1295	0.1354	0.4105
Recall	0.1613	0.0829	0.2926
F-Maß	0.1437	0.1028	0.3417
ROC-AUC	0.4606	0.4829	0.5998

Tabelle 5.19: Kennzahlenauswertung von Modellarchitektur C für die Anomalie des zusätzlichen Gewichts mit unterschiedlicher Merkmalsauswahl

Die Auswertung der Kennzahlen von Modellarchitektur C zur Anomalie des zusätzlichen Gewichts zeigt deutliche Unterschiede zwischen den Merkmalsauswahlmethoden. Die regelbasierte Merkmalsauswahl erzielt die besten Werte in fast allen Metriken. Sie erreicht eine Genauigkeit von 0.7956, eine Präzision von 0.4105 und einen Recall von 0.2926. Auch das F-Maß ist mit 0.3417 bei der regelbasierten Methode höher als bei den anderen beiden Methoden. Die ROC-AUC, die die Trennfähigkeit des Modells zwischen Anomalien und normalen Datensätzen misst, ist bei der regelbasierten Merkmalsauswahl ebenfalls höher, gefolgt von PCA und dem Varianzschwellenwertverfahren.

Bei der Anomalie des Fallens von Objekten liefert die Auswertung der Modellarchitektur C die in Tabelle 5.20 angeführten Kennzahlen.

	Varianzschwellenwert	PCA	Regelbasierte Merkmale
Genauigkeit	0.7418	0.7830	0.8306
Präzision	0.2517	0.0857	0.1279
Recall	0.6596	0.0966	0.0875
F-Maß	0.3644	0.0908	0.1039
ROC-AUC	0.7059	0.4831	0.5060

Tabelle 5.20: Kennzahlenauswertung von Modellarchitektur C für die Anomalie des Fallens von Objekten mit unterschiedlicher Merkmalsauswahl

Die Genauigkeit erreicht bei der regelbasierten Merkmalsauswahl den höchsten Wert von 0.8306, was auf eine gute Gesamtklassifikationsfähigkeit des Modells hinweist. Bei der Präzision erzielt die Varianzschwellenwertmethode den höchsten Wert mit 0.2517, während die anderen beiden Methoden geringere Werte aufweisen. Auch beim Recall erzielt die Varianzschwellenwertmethode mit einem Wert von 0.6596 die beste Leistung. Dies deutet darauf hin, dass sie einen größeren Anteil der tatsächlichen Anomalien korrekt erkennt. Das F-Maß ist mit einem Wert von 0.3644 beim Varianzschwellenwertverfahren am höchsten. Diese Methode schneidet mit einem Wert von 0.7059 auch beim AUC am besten ab. Insgesamt zeigen die Ergebnisse, dass Modellarchitektur C mit der Varianzschwellenwertmethode die stärkste Gesamtleistung für die Anomalie des Fallens von Objekten bietet, insbesondere in Bezug auf Recall und ROC-AUC.

Die Auswertung der Anomalie der reduzierten Geschwindigkeit zeigt bei Modellarchitektur C, die in Tabelle 5.21 zusammengefassten Kennzahlen.

	Varianzschwellenwert	PCA
Genauigkeit	0.0469	0.8999
Präzision	0.0164	0.1381
Recall	0.9893	0.9995
F-Maß	0.0322	0.2426
ROC-AUC	0.5104	0.9489

Tabelle 5.21: Kennzahlenauswertung von Modellarchitektur C für die Anomalie der reduzierten Geschwindigkeit mit unterschiedlicher Merkmalsauswahl

Die Analyse der Anomalie der reduzierten Geschwindigkeit für Modellarchitektur C zeigt deutliche Unterschiede zwischen den beiden getesteten Methoden. Die Genauigkeit erreicht bei PCA einen Wert von 0.8999 und beim Varianzschwellenwertverfahren nur einen Wert von 0.0469. Ähnlich ist die Präzision bei PCA mit 0.1381 höher als bei der Varianzschwellenwertmethode. Beim Recall sind mit einem nahezu perfekten Ergebnis beide Methoden gleichwertig. Das F-Maß ist bei PCA mit 0.2426 signifikant höher als bei der Varianzschwellenwertmethode, die ein F-Maß von nur 0.0322 aufweist. PCA erreicht beim AUC einen Wert von 0.9489, was auf eine exzellente Fähigkeit zur Klassentrennung hinweist. Dagegen erreicht der Varianzschwellenwert nur einen Wert von 0.5104.

Bei Modellarchitektur D wurde die Anzahl der Schichten mithilfe der Hyperparameterauswahl von Hyperband festgelegt. Folgende Tabelle 5.22 zeigt die Auswertung anhand der Anomalie des zusätzlichen Gewichts.

	Varianzschwellenwert	PCA	Regelbasierte Merkmale
Genauigkeit	0.6702	0.7320	0.7303
Präzision	0.1651	0.1615	0.3439
Recall	0.2020	0.1141	0.5374
F-Maß	0.1817	0.1337	0.4194
ROC-AUC	0.4879	0.4915	0.6552

Tabelle 5.22: Kennzahlenauswertung von Modellarchitektur D für die Anomalie des zusätzlichen Gewichts mit unterschiedlicher Merkmalsauswahl

Die Kennzahlenauswertung von Modellarchitektur D für die Anomalie des zusätzlichen Gewichts verdeutlichen, dass die regelbasierte Merkmalsauswahl in allen Metriken besser abschneidet als die anderen beiden Ansätze. In erster Linie liegt die Genauigkeit bei 0.7303, die Präzision bei 0.3439 und der Recall bei 0.5374. Diese Auswertung führt zu einem F-Maß von 0.4194. Die ROC-AUC für die regelbasierte Merkmalsauswahl weist mit einem Wert von 0.6552 auf eine solide Leistungsfähigkeit hin. Im Vergleich dazu erzielen die Varianzschwellenwert- und PCA-Ansätze niedrigere Werte in allen Metriken.

Die Auswertung der Anomalie des Fallens eines Objektes zeigt bei Modellarchitektur D, die in Tabelle 5.23 zusammengefassten Kennzahlen.

	Varianzschwellenwert	PCA	Regelbasierte Merkmale
Genauigkeit	0.7923	0.7967	0.8134
Präzision	0.3080	0.0662	0.1914
Recall	0.6827	0.0620	0.2056
F-Maß	0.4245	0.0640	0.1982
ROC-AUC	0.7444	0.4758	0.5479

Tabelle 5.23: Kennzahlenauswertung von Modellarchitektur D für die Anomalie des Fallens von Objekten mit unterschiedlicher Merkmalsauswahl

Die Auswertung der Kennzahlen der Modellarchitektur D für die Anomalie des Fallens eines Objektes signalisiert, wie unterschiedlich Resultate sein können. Die Varianzschwellenwertmethode schneidet grundsätzlich über alle Kennzahlen hinweg am besten ab. Sie erreicht eine Präzision von 0.3080, einen Recall von 0.6827 sowie ein F-Maß von 0.4245. Darüber hinaus weist sie mit einer ROC-AUC von 0.7444 eine gute Trennfähigkeit auf. Im Vergleich dazu erzielen PCA und die regelbasierte Merkmalsauswahl niedrigere Ergebnisse.

Die Analyse der Anomalie der reduzierten Geschwindigkeit für Modellarchitektur D zeigt die in Tabelle 5.24 angeführten Kennzahlen.

	Varianzschwellenwert	PCA
Genauigkeit	0.0468	0.0324
Präzision	0.0166	0.0
Recall	1.0	0.0
F-Maß	0.0326	0.0
ROC-AUC	0.5156	0.0165

Tabelle 5.24: Kennzahlenauswertung von Modellarchitektur D für die Anomalie der reduzierten Geschwindigkeit mit unterschiedlicher Merkmalsauswahl

Die Kennzahlenauswertung der Modellarchitektur D für die Anomalie der reduzierten Geschwindigkeit zeigen erhebliche Unterschiede zwischen der Varianzschwellenwertmethode und dem PCA-Ansatz. Die Varianzschwellenwertmethode erreicht eine extrem niedrige Genauigkeit von 0.0468, aber zeigt wiederum einen vollständigen Recall von 1.0 an. Das bedeutet, dass die Methode zwar alle tatsächlichen Anomalien erkennt, aber auch viele False Positives erzeugt. Das F-Maß von 0.0326 und die ROC-AUC von 0.5156 signalisieren, dass die Methode nur geringfügig besser ist als ein zufälliger Klassifikator. Im Vergleich dazu schneidet die PCA-Methode viel schlechter ab. Sie erzielt eine Genauigkeit von 0.0324, eine Präzision von 0.0 und einen Recall von 0.0. Demnach hat die PCA-Methode keine Anomalien korrekt identifiziert.

Die Analyse der verschiedenen Modellarchitekturen in Kombination mit den Merkmalsauswahlverfahren zeigt deutliche Unterschiede bei der Erkennung von Anomalien. Einerseits erzielt die PCA-Methode eine sehr gute Gesamtleistung für die Anomalie der reduzierten Geschwindigkeit. Andererseits zeigt sich die regelbasierte Merkmalsauswahl oft als die effektivste Methode in Bezug

auf die Kennzahlen AUC und F-Maß. Die Varianzschwellenmethode erzielt hingegen nur geringe Leistungen bei der Erkennung von Anomalie des zusätzlichen Gewichts sowie der reduzierten Geschwindigkeit. Im Gesamten deutet die Auswertung darauf hin, dass die Wahl der Merkmale einen wesentlichen Einfluss auf die Leistungsfähigkeit der Modelle hat. PCA und regelbasierte Ansätze sind in der Regel besser aufgestellt, da sie weniger Merkmale zum Trainieren des Modells verwenden. Die Modellarchitektur spielt ebenfalls eine wichtige Rolle bei der Bestimmung der Modellleistung. Hierbei erwies sich die Architektur bestehend aus jeweils zwei Schichten im Encoder und Decoder als besonders vorteilhaft.

6

Diskussion

In diesem Kapitel werden die Ergebnisse der Auswertung zum Datenmonitoring für Roboteranwendungen diskutiert. Dabei wird ein Vergleich zwischen regelbasierten und maschinellen Lernmethoden durchgeführt. Zunächst erfolgt eine Interpretation und Diskussion der erhobenen Daten und Ergebnisse. Anschließend werden diese Ergebnisse anhand der unter Kapitel 2.4 aufgelisteten Literatur verglichen, um ihre Bedeutung und Relevanz im Kontext der bisherigen Forschung zu bewerten. Schließlich werden neue Erkenntnisse hervorgehoben und mögliche Implikationen der Forschungsergebnisse aufgezeigt.

6.1 Interpretation und Diskussion der Ergebnisse

Die Ergebnisse der regelbasierten und maschinellen Methoden zur Anomalieerkennung zeigen interessante Einblicke in ihre jeweiligen Stärken und Schwächen.

Der regelbasierte Ansatz zeigt starke Leistungen bei der Anomalieerkennung in den drei Kategorien „Zusätzliches Gewicht“, „Objekt fallen lassen“ und „Reduzierte Geschwindigkeit“. Die Genauigkeit, Präzision, Recall und das F-Maß sind über alle Anomalien hinweg relativ hoch. Hervorzuheben ist dabei die Erkennung der Anomalie der reduzierten Geschwindigkeit. Die regelbasierte Methode erzielt in diesem Kontext in allen Metriken perfekte Werte von 1.0. Dies deutet darauf hin, dass der regelbasierte Ansatz besonders zuverlässig in der Erkennung dieser spezifischen Anomalie ist. Ein möglicher Grund dafür könnte die Abhängigkeit der Regel von einer einzigen Variablen sein. Dadurch wird das Problem linear separierbar, was die Identifikation der Anomalie vereinfacht.

Für die Anomalien des zusätzlichen Gewichts sowie des Fallens eines Objektes zeigt der regelbasierte Ansatz ebenfalls eine hohe Genauigkeit von 0.9318 und 0.9607 sowie einen hohen Recall von 0.9581 und 0.9998 auf. Die Präzision liegt bei diesen beiden Anomalien jedoch etwas niedriger, nämlich bei 0.7414 und 0.7408. Dies lässt auf eine gewisse Anzahl von False Positives schließen. Im Großen und Ganzen ist das F-Maß für beide Kategorien solide, was auf ein gutes Gleichgewicht zwischen Präzision und Recall hinweist. Die AUC-Werte sind ebenfalls hoch und belegen die Effektivität des regelbasierten Ansatzes bei der Anomalieerkennung in diesen spezifischen Szenarien.

Im Gegensatz dazu variieren die Ergebnisse des maschinellen Ansatzes. Abhängig von der Anzahl der Schichten im Encoder und Decoder sowie der verwendeten Methoden zur Merkmalsauswahl, erzielt der Autoencoder unterschiedliche Resultate.

Modellarchitektur A, das eine Schicht im Encoder und Decoder verwendet, zeigt konstant niedrige Werte im Vergleich zum regelbasierten Ansatz. Beispielsweise beträgt die Genauigkeit für die Anomalie des zusätzlichen Gewichts bei Anwendung der regelbasierten Merkmalsauswahl lediglich 0.7456. Auch die Präzision ist mit einem Wert von 0.2085 deutlich unter dem Ergebnis des Regelsatzes. Für die Anomalie des Fallens eines Objekts zeigt das Modellarchitektur A mit PCA eine Genauigkeit von 0.8216, eine Präzision von 0.2232 und einen Recall von 0.2377, was zu einem F-Maß von 0.2302 führt. Bei der Anomalie der reduzierten Geschwindigkeit mit Varianzschwellenwertverfahren liegt die Genauigkeit bei 0.0466, die Präzision bei 0.0157 und der Recall bei 0.9443. Auch die AUC-Werte sind bei allen Auswertungen dieser Modellarchitektur sehr gering. Dies deutet darauf hin, dass diese Architektur Schwierigkeiten hat, diese Arten von Anomalien effektiv zu erkennen.

Modellarchitektur B, das zwei Schichten im Encoder und Decoder verwendet, zeigt signifikante Verbesserungen gegenüber Modellarchitektur A. Für die Anomalie des zusätzlichen Gewichts erreicht Modellarchitektur B mit der regelbasierten Merkmalsauswahl die besten Ergebnisse. Die erzielte Genauigkeit liegt bei 0.7723, die Präzision bei 0.4250 und der Recall bei 0.7261. Das F-Maß von 0.5361 und die ROC-AUC von 0.7543 bestätigen die überlegene Leistungsfähigkeit der regelbasierten Merkmalsauswahl für diese spezifische Anomalie. Für die Anomalie des Fallens von Objekten erzielt Modellarchitektur B ebenfalls die besten Ergebnisse mit der regelbasierten Merkmalsauswahl. Die Genauigkeit beträgt hier 0.8504, die Präzision liegt bei 0.3974 und der Recall bei 0.6456. Das F-Maß von 0.4920 sowie die ROC-AUC von 0.7609 unterstreicht die gute Leistung dieser Methode im Vergleich zu PCA und dem Varianzschwellenwert. Bei der Anomalie der reduzierten Geschwindigkeit hingegen zeigt sich, dass PCA überlegen ist. Mit einer Genauigkeit von 0.9465, einer Präzision von 0.2307 und einer ROC-AUC von 0.9728 erzielt PCA deutlich bessere Ergebnisse als der Varianzschwellenwert. Die beobachteten Verbesserungen von Modellarchitektur B im Vergleich zu Modellarchitektur A sind zwar beachtlich, erreichen jedoch nicht die Signifikanz, um mit den Ergebnissen des regelbasierten Ansatzes gleichzuziehen.

Modellarchitektur C, das mit jeweils drei Schichten im Encoder und Decoder ausgestattet ist, zeigt in der Analyse der Anomalien keine Leistungsverbesserungen im Vergleich zu Modellarchitektur B. Für die Anomalie des zusätzlichen Gewichts zeigt die Kennzahlenauswertung von Modellarchitektur C, dass die regelbasierte Merkmalsauswahl weiterhin die besten Ergebnisse liefert. Das F-Maß von 0.3417 und die ROC-AUC von 0.5998 bestätigen zwar die Überlegenheit der regelbasierten Methode, jedoch sind die Ergebnisse nicht signifikant genug, um als Fortschritt gegenüber Modellarchitektur B betrachtet zu werden. Bei der Anomalie des Fallens von Objekten erreicht Modellarchitektur C eine Genauigkeit von 0.8306. Dieser hohe Wert wird jedoch durch eine niedrige Präzision von 0.1279 relativiert. Auch das F-Maß und die ROC-AUC liegen hinter den Ergebnissen von Modellarchitektur B. Im Großen und Ganzen zeigt sich, dass die zusätzliche Schichtanzahl in Modellarchitektur C nicht notwendigerweise zu besseren Ergebnissen führt, sondern in einigen Fällen sogar die Leistung im Vergleich zu den einfacheren Architekturen verringert.

Modellarchitektur D, bei dem die Anzahl der Schichten mittels Hyperband bestimmt wurde, zeigt ebenfalls keine Leistungsverbesserungen im Vergleich zu Modellarchitektur B. Bei der Anomalie des zusätzlichen Gewichts schneidet ebenfalls die regelbasierte Merkmalsauswahl besser als die anderen beiden Methoden ab. Sie erzielt eine Genauigkeit von 0.7303, einer Präzision von 0.3439, einen Recall von 0.5374, einem F-Maß von 0.4194 und einer ROC-AUC von 0.6552. Für die

Anomalie des Fallens von Objekten erreicht die Varianzschwellenwertmethode die höchsten Werte mit einer Präzision von 0.3080, einem Recall von 0.6827 und einer ROC-AUC von 0.7444. Bei der Anomalie der reduzierten Geschwindigkeit erzielt ebenfalls die Varianzschwellenwertmethode bessere Ergebnisse als PCA. Sie erreicht einen vollständigen Recall von 1.0, jedoch auf Kosten einer extrem niedrigen Präzision von 0.0166 und eines F-Maßes von 0.0326.

Zusammenfassend lässt sich sagen, dass der regelbasierte Ansatz in diesem speziellen Anwendungsfall eine robustere und zuverlässigere Leistung bei der Anomalieerkennung zeigt als die maschinellen Lernmodelle. Die maschinellen Modelle sind theoretisch flexibler und anpassungsfähiger, scheinen aber in dieser spezifischen Konfiguration und Merkmalsauswahl nicht die gleiche Effizienz zu erreichen. Weitere Forschung und Optimierung wären notwendig, um die Leistung der maschinellen Lernmodelle zu verbessern und sie möglicherweise an den regelbasierten Ansatz heranzuführen oder zu übertreffen.

6.2 Vergleich der Ergebnisse mit der vorhandenen Literatur

Die Ergebnisse des maschinellen Ansatzes zur Anomalieerkennung zeigen eine Variation in der Leistung, abhängig von der Konfiguration und der verwendeten Merkmalsauswahl. Im Vergleich zur Literatur sind einige interessante Gemeinsamkeiten und Unterschiede zu beobachten.

In Kapitel 2.4 wird der aktuelle Stand der Literatur dargelegt. Dabei wird ein LSTM-Autoencoder-Modell zur Anomalieerkennung erwähnt. Dieses Modell erreicht für die Auswertung der Anomalien einer Pick-and-Place Aufgabe folgende ROC-AUC-Werte (Graabæk et al., 2023).

- Für die Anomalie des zusätzlichen Gewichts: 0.8697
- Für die Anomalie des Fallen lassen eines Objekts: 0.9524
- Für die Anomalie der reduzierten Geschwindigkeit: 0.9163

Ein Vergleich mit den Ergebnissen dieser Arbeit zeigt, dass insbesondere die Modellarchitekturen A und C im Allgemeinen eine geringere Leistungsfähigkeit aufweisen als das LSTM-Autoencoder-Modell aus der Literatur. Modellarchitektur B hingegen zeigt unter Verwendung der regelbasierten Merkmalsauswahl in einigen Fällen vergleichbare Ergebnisse. Besonders hervorzuheben ist die Erkennung der reduzierten Geschwindigkeit, bei der dieser Ansatz eine bemerkenswert hohe AUC von 0.9728 erzielt.

Die Erkennung des zusätzlichen Gewichts sowie des Fallens eines Objektes schneiden hingegen etwas schlechter ab. Sie erzielen mit Modellarchitektur B eine ROC-AUC von 0.7543 und 0.7609. Dies könnte darauf zurückzuführen sein, dass die Montage einer Seilführungsscheibe eine höhere Komplexität aufweist als eine Pick-and-Place-Aufgabe, wie sie in der Literatur beschrieben wird.

6.3 Erkenntnisse und mögliche Implikationen

Nachdem regelbasierte Methoden auf expliziten, vordefinierten Regeln basieren, sind sie besonders effektiv bei der Handhabung klar definierter und einfacher Aufgaben. Die Regeln bieten eine verlässliche Grundlage für die Fehlererkennung in stabilen und gut verstandenen Umgebungen. Auf der

anderen Seite ermöglichen maschinelle Lernverfahren die Bewältigung komplexer und dynamischer Szenarien. Maschinelle Verfahren können subtile Muster oder Anomalien identifizieren, die durch vordefinierte Regeln möglicherweise nicht erfasst werden. Der Einsatz von maschinellem Lernen kann insbesondere in Umgebungen von Vorteil sein, in denen Anomalien weniger offensichtlich sind.

Des Weiteren wurde in dieser Arbeit eine Kombination aus regelbasierte und maschinellen Lernansätzen untersucht. Dabei wurde festgestellt, dass das Zusammenspiel aus beiden Ansätzen zu einer vielversprechenden Möglichkeit der Verbesserung der Leistungsfähigkeit führt. In den Untersuchungen wurde eine Anomalie als solche klassifiziert, wenn mindestens einer der beiden Ansätze, entweder der regelbasierte oder der maschinelle Lernansatz, den Datensatz als anomal identifizierte. Die Auswertung der Ergebnisse zeigt eine Steigerung des Recalls. Bei der Anomalie des zusätzlichen Gewichts erzielt der regelbasierte Ansatz einen Recall-Wert von 0.9581. Durch die Kombination mit maschinellem Lernen konnte dieser auf bemerkenswerte 0.9967 gesteigert werden.

Demnach könnte die Kombination dieser beiden Ansätze den Gesamt-Recall der Anomalieerkennung erheblich verbessern. Während regelbasierte Methoden eine solide Basis bieten, könnten maschinelle Lerntechniken zusätzliche Tiefe und Flexibilität zur Erkennung komplexer Anomalien beitragen. Daher liegt eine wesentliche Implikation in der potenziellen Synergie beider Ansätze. Die Integration von regelbasierten Methoden mit fortschrittlichen maschinellen Lernverfahren könnte eine umfassendere und robustere Lösung zur Anomalieerkennung bieten. Diese Lösung könnte sowohl einfache als auch komplexe Anomalien effektiv adressieren.

7

Fazit

In den letzten Jahren hat die Anwendung von Robotersystemen in der Industrie, Landwirtschaft und Infrastruktur erheblich zugenommen (Matthias, 2017). Roboter optimieren die Produktionsprozesse, unterstützen bei landwirtschaftlichen Tätigkeiten und führen Infrastrukturinspektionen durch (De Backer et al., 2018; Halder & Afsari, 2023; Mahmud et al., 2020). Diese Entwicklungen erfordern ein präzises Datenmonitoring. Mithilfe des kontinuierlichen Sammelns und Analysierens von Daten können frühzeitige Probleme erkannt und Systeme optimiert werden (Mudliar, 2023). Nichts desto trotz kann die kontinuierliche Datenprotokollierung auch zu Herausforderungen wie erhöhtem Speicherbedarf, höheren Kosten und längeren Analysezeiten führen (Khan et al., 2014; Sagiroglu & Sinanc, 2013).

In diesem Kontext wurden in der Forschung zum Datenmonitoring von Robotersystemen verschiedene Ansätze untersucht, um den Herausforderungen der Datenverarbeitung entgegenzuwirken. Regelbasierte Methoden nutzen vordefinierte Regeln zur Identifikation von Abweichungen und erfordern dabei detailliertes Vorwissen (Graabæk et al., 2023; Liu et al., 2014). Im Gegensatz dazu bieten maschinelle Lerntechniken die Möglichkeit, selbstständig Muster zu erkennen und Anomalien basierend auf historischen Daten zu identifizieren (Bonaccorso, 2017).

Ziel dieser Arbeit war es, die Leistung regelbasierter Methoden und maschineller Lernansätze im Datenmonitoring von Robotersystemen zu untersuchen. Dabei wurde bewertet, ob maschinelles Lernen als eigenständige Alternative oder als ergänzende Technik zu den traditionellen regelbasierten Ansätzen dienen kann. Um diese Forschungsfrage zu beantworten, wurden beide Ansätze in der Programmiersprache Python implementiert und anhand von drei gezielt eingeführten Anomalien evaluiert. Untersucht wurden dabei die Erkennung von zusätzlichem Gewicht, das Fallen lassen von Objekten und die Reduzierung der Geschwindigkeit.

Die Ergebnisse der Evaluierung zeigen, dass der regelbasierte Ansatz bei der Anomalieerkennung durchgehend hohe Werte in der Genauigkeit, Recall, F-Maß und ROC-AUC erreicht. Besonders bei linear separierbaren Anomalien wie der reduzierter Geschwindigkeit erzielt der regelbasierte Ansatz perfekte Werte in allen Kennzahlen. Allerdings ist die Präzision bei bestimmten Anomalien wie zusätzlichem Gewicht und dem Fallen lassen von Objekten niedriger, was auf eine gewisse Anzahl von False Positives hinweist. Insgesamt zeichnet sich dieser Ansatz durch seine leichte Interpretierbarkeit und die einfache Anwendbarkeit aus.

Im Gegensatz dazu zeigen maschinelle Lernmodelle je nach Anzahl der Schichten und der ausgewählten Merkmale unterschiedliche Ergebnisse. In dieser Arbeit erwies sich die aus zwei Schichten

bestehende Architektur in Kombination mit der regelbasierten Merkmalsauswahl als besonders leistungsfähig. Allerdings wird damit nicht die Gesamtleistung des regelbasierten Ansatzes erreicht.

Zusammengefasst deutet dies darauf hin, dass maschinelles Lernen in der aktuellen Konfiguration nicht als eigenständige Alternative zu regelbasierten Ansätzen dienen kann. Stattdessen kann das maschinelle Lernen als ergänzende Technik angesehen werden, um die Erkennungsleistung zu verbessern. Eine kombinierte Methode, die beide Ansätze integriert, könnte eine umfassendere und robustere Lösung zur Anomalieerkennung bieten. Aus diesem Grund sollten in der Zukunft weitere Studien zum Testen unter verschiedenen Anomalieszenarien sowie Anwendungsfällen durchgeführt werden.

Literatur

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). TensorFlow: a system for Large-Scale machine learning. *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283.

Aggarwal, C. C. (2024). *Probability and Statistics for Machine Learning: A Textbook*. Springer Nature.

Ahmed, M., Mahmood, A. N., & Islam, M. R. (2016). A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, *55*, 278–288.

Akerkar, R., & Sajja, P. (2009). *Knowledge-based systems*. Jones & Bartlett Publishers.

Ando, N., Suehiro, T., Kitagaki, K., & Kotoku, T. (2006). Rt (robot technology)-component and its standardization-towards component based networked robot systems development. *2006 SICE-ICASE International Joint Conference*, 2633–2638.

Angiulli, F., & Pizzuti, C. (2002). Fast outlier detection in high dimensional spaces. *European conference on principles of data mining and knowledge discovery*, 15–27.

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C. A., & Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, *16*(5), 412–424.

Barnett, V., Lewis, T., et al. (1994). *Outliers in statistical data* (Bd. 3). Wiley New York.

Bonaccorso, G. (2017). *Machine learning algorithms*. Packt Publishing Ltd.

Boutsidis, C., Mahoney, M. W., & Drineas, P. (2008). Unsupervised feature selection for principal components analysis. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 61–69.

Bruyninckx, H. (2001). Open robot control software: the OROCOS project. *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No. 01CH37164)*, *3*, 2523–2528.

Chandola, V., Banerjee, A., & Kumar, V. (2007). Outlier detection: A survey. *ACM Computing Surveys*, *14*, 15.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 1–58.

Charte, D., Charte, F., Garcia, S., del Jesus, M. J., & Herrera, F. (2018). A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion*, *44*, 78–96.

Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T. (2018). Autoencoder-based network anomaly detection. *2018 Wireless telecommunications symposium (WTS)*, 1–5.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, *21*, 1–13.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*.

DataScience-ProF. (2024). A Comprehensive Guide to Feature Selection using Variance Threshold in Scikit-Learn — medium.com [[Accessed 07-07-2024]].

De Backer, K., DeStefano, T., Menon, C., & Suh, J. R. (2018). Industrial robotics and the global organisation of production.

De Luca, A., & Mattone, R. (2005). An identification scheme for robot actuator faults. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1127–1131.

dearpygui — pypi.org [[Accessed 28-06-2024]]. (2022).

Dresch, A., Lacerda, D. P., Antunes Jr, J. A. V., Dresch, A., Lacerda, D. P., & Antunes, J. A. V. (2015). *Design science research*. Springer.

Fan, J., Upadhye, S., & Worster, A. (2006). Understanding receiver operating characteristic (ROC) curves. *Canadian Journal of Emergency Medicine*, *8*(1), 19–20.

Fantuzzi, C., Secchi, C., & Visioli, A. (2003). On the fault detection and isolation of industrial robot manipulators. *IFAC Proceedings Volumes*, *36*(17), 399–404.

Foody, G. M. (2023). Challenges in the real world use of classification accuracy metrics: From recall and precision to the Matthews correlation coefficient. *Plos one*, *18*(10), e0291908.

Ghamry, F. M., El-Banby, G. M., El-Fishawy, A. S., El-Samie, F. E. A., & Dessouky, M. I. (2024). A survey of anomaly detection techniques. *Journal of Optics*, *53*(2), 756–774.

Graabæk, S. G., Ancker, E. V., Fugl, A. R., & Christensen, A. L. (2023). An Experimental Comparison of Anomaly Detection Methods for Collaborative Robot Manipulators. *IEEE Access*.

Halder, S., & Afsari, K. (2023). Robots in inspection and monitoring of buildings and infrastructure: A systematic review. *Applied Sciences*, *13*(4), 2304.

Hawkins, D. M. (1980). *Identification of outliers* (Bd. 11). Springer.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Homayouni, H., Ghosh, S., Ray, I., Gondalia, S., Duggan, J., & Kahn, M. G. (2020). An autocorrelation-based LSTM-autoencoder for anomaly detection on time-series data. *2020 IEEE international conference on big data (big data)*, 5068–5077.

Hunt, V. D. (2012). *Understanding robotics*. Elsevier.

IFR, I. F. o. R. (2021). ISO 8373 [Accessed: 2024-07-21]. https://ifr.org/standardisation

Ilyas, I. F., & Chu, X. (2019). *Data cleaning*. Morgan & Claypool.

International Organization for Standardization. (2016). *ISO/TS 15066:2016, Robots and robotic devices – Collaborative robots* (Techn. Ber.) [Technical Specification]. International Organization for Standardization.

Jackson, J. (2007). Microsoft robotics studio: A technical introduction. *IEEE robotics & automation magazine*, *14*(4), 82–87.

Jović, A., Brkić, K., & Bogunović, N. (2015). A review of feature selection methods with applications. *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, 1200–1205.

Khalastchi, E., & Kalech, M. (2018). On fault detection and diagnosis in robotic systems. *ACM Computing Surveys (CSUR)*, *51*(1), 1–24.

Khalastchi, E., Kalech, M., Kaminka, G. A., & Lin, R. (2015). Online data-driven anomaly detection in autonomous robots. *Knowledge and Information Systems*, *43*, 657–688.

Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Mahmoud Ali, W. K., Alam, M., Shiraz, M., Gani, A., et al. (2014). Big data: survey, technologies, opportunities, and challenges. *The scientific world journal*, *2014*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lalkhen, A. G., & McCluskey, A. (2008). Clinical tests: sensitivity and specificity. *Continuing education in anaesthesia, critical care & pain*, *8*(6), 221–223.

Lane, D. M., Scott, D., Hebl, M., Guerra, R., Osherson, D., & Zimmer, H. (2017). Introduction to statistics, online edition. *Rice University, University of Houston Clear Lake, and Tufts University*.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*(185), 1–52.

Lin, J., Keogh, E., Fu, A., & Van Herle, H. (2005). Approximations to magic: Finding unusual medical time series. *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, 329–334.

Liu, H., Gegov, A., & Stahl, F. (2014). Categorization and construction of rule based systems. *Engineering Applications of Neural Networks: 15th International Conference, EANN 2014, Sofia, Bulgaria, September 5-7, 2014. Proceedings 15*, 183–194.

Lotz, A., Steck, A., & Schlegel, C. (2011). Runtime monitoring of robotics software components: Increasing robustness of service robotic systems. *2011 15th International Conference on Advanced Robotics (ICAR)*, 285–290.

Mahmud, M. S. A., Abidin, M. S. Z., Emmanuel, A. A., & Hasan, H. S. (2020). Robotics and automation in agriculture: present and future applications. *Applications of Modelling and Simulation*, *4*, 130–140.

Matthias, B. (2017). Human robot collaboration–industrial applications and open challenges. *Dagstuhl seminar on"computer-assisted engineering for robotics and autonomous systems". Dagstuhl, Germany: ABB*.

Michelucci, U. (2022). An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*.

Milkowski, M., & Kautilya, C. (2022). RTDE Python Client Library [Accessed: 2024-07-19].

Mobtahej, P., Zhang, X., Hamidi, M., & Zhang, J. (2022). An LSTM-Autoencoder Architecture for Anomaly Detection Applied on Compressors Audio Data. *Computational and Mathematical Methods*, *2022*(1), 3622426.

Mudliar, S. (2023). Beginner's Guide to Data Monitoring [[Accessed 25-06-2024]].

Nortey, E. N., Pometsey, R., Asiedu, L., Iddi, S., & Mettle, F. O. (2021). Anomaly detection in health insurance claims using bayesian quantile regression. *International Journal of Mathematics and Mathematical Sciences*, *2021*(1), 6667671.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., de Marmiesse, G., Fu, Y., Podivin, J., Schäfer, F., et al. (2019). Keras Tuner. *Available online: github.com/keras-team/kerastuner*.

Park, D., Hoshi, Y., & Kemp, C. C. (2018). A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, *3*(3), 1544–1551.

Prechelt, L. (2002). Early stopping-but when? In *Neural Networks: Tricks of the trade* (S. 55–69). Springer.

Press, O. U. (1999). *The Oxford American Dictionary of Current English* (Animate).

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. (2009). ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, *3*(3.2), 5.

Real-Time Data Exchange (RTDE) Guide [[Accessed 01-06-2024]]. (2019).

Room, C. (2019). Confusion matrix. *Mach. Learn*, *6*, 27.

rospy - ROS Wiki [[Accessed 19-07-2024]]. (2017).

rule-engine — pypi.org [[Accessed 31-01-2024]]. (2024).

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. *2013 international conference on collaboration technologies and systems (CTS)*, 42–47.

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*, 105524.

Singh, K., Sharma, K., Avinash, G., Kumar, R. R., Ray, M., Ramasubramanian, V., Lama, A., & Lal, S. (2023). LSTM based Stacked Autoencoder Approach for Time Series Forecasting. *J. Indian Soc. Agricultural Statist.*, *77*, 71–78.

Stavrou, D., Eliades, D. G., Panayiotou, C. G., & Polycarpou, M. M. (2016). Fault detection for service mobile robots using model-based method. *Autonomous Robots*, *40*, 383–394.

Thara, D., PremaSudha, B., & Xiong, F. (2019). Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques. *Pattern Recognition Letters*, *128*, 544–550.

universal-robots.com [[Accessed 05-06-2024]]. (2024).

Anhang A

Name	Type	Comment	Introduced in version
timestamp	DOUBLE	Time elapsed since the controller was started [s]	
target_q	VECTOR6D	Target joint positions	
target_qd	VECTOR6D	Target joint velocities	
target_qdd	VECTOR6D	Target joint accelerations	
target_current	VECTOR6D	Target joint currents	
target_moment	VECTOR6D	Target joint moments (torques)	
actual_q	VECTOR6D	Actual joint positions	
actual_qd	VECTOR6D	Actual joint velocities	
actual_current	VECTOR6D	Actual joint currents	
joint_control_output	VECTOR6D	Joint control currents	
actual_TCP_pose	VECTOR6D	Actual Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation	
actual_TCP_speed	VECTOR6D	Actual speed of the tool given in Cartesian coordinates. The speed is given in [m/s] and the rotational part of the TCP speed (rx, ry, rz) is the angular velocity given in [rad/s]	
actual_TCP_force	VECTOR6D	Generalized forces in the TCP. It compensates the measurement for forces and torques generated by the payload	

target_TCP_pose	VECTOR6D	Target Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation	
target_TCP_speed	VECTOR6D	Target speed of the tool given in Cartesian coordinates. The speed is given in [m/s] and the rotational part of the TCP speed (rx, ry, rz) is the angular velocity given in [rad/s]	
actual_digital_input_bits	UINT64	Current state of the digital inputs. 0-7: Standard, 8-15: Configurable, 16-17: Tool	
joint_temperatures	VECTOR6D	Temperature of each joint in degrees Celsius	
actual_execution_time	DOUBLE	Controller real-time thread execution time	
robot_mode	INT32	Robot mode	
joint_mode	VECTOR6INT32	Joint control modes	
safety_mode	INT32	Safety mode	
safety_status	INT32	Safety status	3.10.0 / 5.4.0
actual_tool_accelerometer	VECTOR3D	Tool x, y and z accelerometer values	
speed_scaling	DOUBLE	Speed scaling of the trajectory limiter	
target_speed_fraction	DOUBLE	Target speed fraction	
actual_momentum	DOUBLE	Norm of Cartesian linear momentum	
actual_main_voltage	DOUBLE	Safety Control Board: Main voltage	
actual_robot_voltage	DOUBLE	Safety Control Board: Robot voltage (48V)	
actual_robot_current	DOUBLE	Safety Control Board: Robot current	

actual_joint_voltage	VECTOR6D	Actual joint voltages	
actual_digital_output_bits	UINT64	Current state of the digital outputs. 0-7: Standard, 8-15: Configurable, 16-17: Tool	
runtime_state	UINT32	Program state	
elbow_position	VECTOR3D	Position of robot elbow in Cartesian Base Coordinates	3.5.0 / 5.0.0
elbow_velocity	VECTOR3D	Velocity of robot elbow in Cartesian Base Coordinates	3.5.0 / 5.0.0