BERGISCHE
UNIVERSITÄT
WUPPERTAL

Fakultät für Maschinenbau und Sicherheitstechnik

Masterthesis

im Studiengang **Qualitätsingenieurwesen**
beim Fachgebiet für **Verkehrssicherheit und Zuverlässigkeit**

zur Erreichung des akademischen Grades
Master of Science

Thema:	**Zuverlässigkeitstechnik in der Industrie 4.0 unter Anwendung von RNNs mit Lebesgue-Sampling zur RUL-Prognose**
Autor:	**Zikai Zhang**
MatNr:	1942461
Bearbeitungszeitraum:	**25.04 2024 bis 23.09.2024**
Betreuer*in:	Jun.-Prof. Dr. Antoine Tordeux
Erstprüfer*in:	Jun.-Prof. Dr. Antoine Tordeux
Zweitprüfer*in:	Univ.-Prof. Dr.-Ing. Bracke, Stefan

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die von mir eingereichte Abschlussarbeit (Bachelor-/Master-Thesis) selbstständig verfasst und keine andere als die angegebene Quellen und Hilfsmittel benutzt sowie Stellen der Abschlussarbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen wurden, in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich bin damit einverstanden, dass die Arbeit durch Dritte eingesehen und unter Wahrung urheberrechtlicher Grundsätze zitiert werden darf.

Ort und Datum: _____

Unterschrift: _____

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Vollständige Bezeichnung (Deutsch)	Vollständige Bezeichnung (Englisch)
CPS	Cyber-Physische Systeme	Cyber-Physical Systems
IIoT	Industrielles Internet der Dinge	Industrial Internet of Things
IoT	Internet der Dinge	Internet of Things
LS	Lebesgue-Sampling	Lebesgue Sampling
RNNs	Rekurrente Neuronale Netzwerke	Recurrent Neural Networks
LSTM	Langfristiges Kurzzeitgedächtnis-Netzwerk	Long Short-Term Memory Networks
BiLSTM	Bidirektionales Langfristiges Kurzzeitgedächtnis-Netzwerk	Bidirectional Long Short-Term Memory Networks
RUL	Verbleibende Nutzungsdauer	Remaining Useful Life
FD	Fehlerdiagnose	Fault Diagnosis
RS	Riemann-Sampling	Riemann Sampling
RMSE	Wurzel des mittleren quadratischen Fehlers	Root Mean Square Error
MSE	Mittlerer quadratischer Fehler	Mean Squared Error
NASA	NASA-Bewertungssystem	NASA Scoring System
VIF	Varianzinflationsfaktor	Variance Inflation Factor
FFT	Schnelle Fourier-Transformation	Fast Fourier Transform
PHM	Prognostik und Gesundheitsmanagement	Prognostics and Health Management
TTF	Zeit bis zum Ausfall	Time to Failure
SW	Gleitendes Fenster	Sliding Window
MES	Mittlerer quadratischer Fehler (Summe)	Mean Squared Error (Sum)
SD	Standardabweichung	Standard Deviation
CNN	Convolutional Neural Networks (Konvolutionale Neuronale Netzwerke)	Convolutional Neural Networks
SVM	Support Vector Machines (Unterstützungsvektormaschinen)	Support Vector Machines
PHM	Prognostics and Health Management	Prognostics and Health Management

Tabellenverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Diese Studie analysiert zunächst die grundlegenden Konzepte von Industrie 4.0, einschließlich der Voraussetzungen von Cyber-Physical Systems (CPS), Big Data und dem industriellen Internet der Dinge (IIoT). Anschließend wird der Einfluss von Industrie 4.0 auf die Zuverlässigkeitstechnik eingehend untersucht, Wartungsstrategien analysiert sowie die Schlüsselkomponenten von Lagern und gängige Fehlertypen beschrieben. Im nächsten Schritt wird die Anwendung von maschinellem Lernen in der Lagerfehldiagnose und der RUL-Prognose untersucht, wobei besonderes Augenmerk auf rekurrente neuronale Netze (RNNs), Long Short-Term Memory Netzwerke (LSTM) und bidirektionale LSTM (BiLSTM) gelegt wird.

Diese Studie vergleicht die Parameterauswahl und zeigt, dass die vorgeschlagenen Methoden in ihrer Leistung deutlich besser abschneiden als die in der Batterieforschung verwendeten Methoden. Konkret erreicht die auf Lebesgue-Sampling (LS) basierende RUL-Prognosemethode nicht nur eine höhere Prognosegenauigkeit, sondern erfordert auch weniger Rechenaufwand, was eine höhere Effizienz zeigt. Darüber hinaus verbessert die Anwendung von Echtzeit-Sampling die Skalierbarkeit der Methode weiter und unterstreicht die entscheidende Rolle der Lagerdiagnose für die Betriebssicherheit. In Bezug auf die wirtschaftlichen Effekte ermöglicht die LS-Methode eine relativ genaue prädiktive Instandhaltung durch die Reduzierung der Anforderungen an die Datenverarbeitung und der damit verbundenen Kosten.

Zusammenfassend zeigt diese Studie zukünftige Forschungsrichtungen auf, wie die Verbesserung der Datenvorverarbeitung durch Lebesgue-Sampling, die Optimierung unüberwachter Algorithmen mittels Boosting-Ensemble-Lernmethoden sowie die Weiterentwicklung integrierter Fehlerdiagnosemodule. Diese Methoden haben das Potenzial, die Effizienz und Genauigkeit von prädiktiven Instandhaltungssystemen weiter zu steigern und somit einen signifikanten Mehrwert für Industrie 4.0 zu schaffen.

Summary

This study first analyzes the fundamental concepts of Industry 4.0, including the prerequisites of Cyber-Physical Systems (CPS), Big Data, and the Industrial Internet of Things (IIoT). Subsequently, it delves into the impact of Industry 4.0 on reliability engineering, analyzes maintenance strategies, and describes the key components of bearings and common failure types. Next, the application of machine learning in bearing fault diagnosis and RUL prediction is investigated, with a particular focus on Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTM), and Bidirectional LSTM (BiLSTM).

This study compares parameter selection and demonstrates that the proposed methods significantly outperform those used in battery research in terms of performance. Specifically, the RUL prediction method based on Lebesgue Sampling (LS) not only achieves higher prediction accuracy but also requires less computational effort, demonstrating greater efficiency. Additionally, by employing real-time sampling, the scalability of the method is further enhanced, fully highlighting the critical role of bearing diagnosis in operational safety. From an economic perspective, the LS method enables relatively accurate predictive maintenance by reducing data processing requirements and associated costs.

In summary, this study outlines future research directions, such as improving data preprocessing through Lebesgue Sampling, optimizing unsupervised algorithms using boosting ensemble learning methods, and further developing integrated fault diagnosis modules. These approaches have the potential to further enhance the efficiency and accuracy of predictive maintenance systems, thereby providing significant value to Industry 4.0.

1. Einleitung

1.1 Problemstellung

Lager sind entscheidende Komponenten zur Bewertung des Gesundheitszustands rotierender Maschinen. Die Zuverlässigkeit von Wälzlagern steht in direktem Zusammenhang mit der Zuverlässigkeit rotierender Geräte. Der Betriebszustand der Lager beeinflusst die Leistung der gesamten Maschinerie. Ein Ausfall der Lager kann nicht nur die Produktivität beeinträchtigen, sondern auch zu ernsthaften mechanischen Schäden führen, die die persönliche Sicherheit gefährden und erhebliche Verluste für das betroffene Personal verursachen können.

Daher ist es von großer praktischer Bedeutung für die Instandhaltung von Maschinen und Ausrüstungen, eine genaue und effektive Prognose der verbleibenden Nutzungsdauer (Remaining Useful Life, RUL) der Lager durchzuführen. Eine Überwachung des Zustands der Lager und eine RUL-Analyse sind daher äußerst notwendig. Darüber hinaus erfordert die Erfassung von Lagerdaten im Rahmen des Konzepts von Industrie 4.0 auch die Nutzung von Sensordaten, um eine umfassende Überwachung und Analyse zu ermöglichen. Indem wir den Zustand der Lager kontinuierlich überwachen und ihre RUL genau Prognosen, können wir Ausfälle verhindern, die Effizienz steigern und sowohl die Sicherheit der Mitarbeiter als auch die Vermögenswerte des Unternehmens schützen.

1.2 Zielsetzung

Das Ziel dieser Arbeit besteht darin, die Anwendung von maschinellem Lernen in der prädiktiven Instandhaltung zu untersuchen und verschiedene Instandhaltungsstrategien, insbesondere im Zusammenhang mit der Lagerinstandhaltung, zu analysieren.

Des Weiteren soll eine Ausfallanalyse basierend auf Maschinendaten durchgeführt werden, um die RUL zu prognose und zu verdeutlichen, wie diese Methoden funktionieren. Dabei werden nicht nur die Prognose der RUL angestrebt, sondern auch ein Vergleich der verwendeten Methoden durchgeführt.

Durch das Erreichen dieser Ziele wird das Verständnis für die Bedeutung der Arbeitssicherheit in der Instandhaltung gestärkt. Zudem werden Potenziale des maschinellen Lernens für die Verbesserung der Instandhaltungsstrategien in der Lagerindustrie aufgezeigt.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist wie folgt strukturiert:

Kapitel 2: Stand der Technik
Dieser Abschnitt analysiert den aktuellen Stand der Technik. Es werden die grundlegenden Konzepte von Industrie 4.0 erläutert, einschließlich Cyber-Physical Systems (CPS), Big Data und dem Industrial Internet of Things (IIoT). Besonderes Augenmerk liegt auf der Rolle von Sensoren und IIoT Edge Devices. Zudem wird die Bedeutung von Prognostics and Health Management (PHM) und der Zuverlässigkeitstechnik im Kontext von Industrie 4.0 diskutiert. Eine detaillierte Betrachtung von Wälzlagern umfasst deren verschiedene Komponenten sowie die Arten von Lagerausfällen.

Kapitel 3: Prozess der Datenanalyse
Dieser Abschnitt beschreibt den gesamten Datenanalyseprozess, der die Schritte Datenerfassung und -selektion, Datenvorverarbeitung, Datenanalyse sowie Ergebnisdarstellung und -interpretation umfasst. Es werden Methoden zur Datenbereinigung, -reduktion und -transformation vorgestellt, die zur Vorbereitung der Daten für die anschließende Analyse notwendig sind.

Kapitel 4: Modul für Fehlerdiagnose und RUL-Prognose
Hier werden die entwickelten Module für die Fehlerdiagnose und die RUL-Prognose vorgestellt. Der Abschnitt umfasst die Auswahl des Datensatzes, die Implementierung der Diagnose- und Prognosemodule sowie die Beschreibung der verwendeten Methoden und Modelle, einschließlich RNN, LSTM und BiLSTM.

Kapitel 5: Anwendungsbeispiele und Ergebnisse
Dieses Kapitel präsentiert praktische Anwendungsbeispiele der entwickelten Module und die erzielten Ergebnisse. Es werden die Modelle anhand verschiedener

Evaluierungsparameter wie RMSE, R^2 und NASA-Scores bewertet und die Leistungsfähigkeit der unterschiedlichen Methoden verglichen.

Kapitel 6: Fazit

Im abschließenden Kapitel werden die wichtigsten Erkenntnisse der Arbeit zusammengefasst. Zudem wird ein Ausblick auf zukünftige Forschungsrichtungen gegeben, die sich aus den gewonnenen Ergebnissen ableiten lassen.

Im vierten Abschnitt wird ein Fazit gezogen, das die wichtigsten Erkenntnisse der Arbeit zusammenfasst und einen Ausblick auf zukünftige Forschungsrichtungen gibt.

2. Stand der Technik

2.1 Industrie 4.0

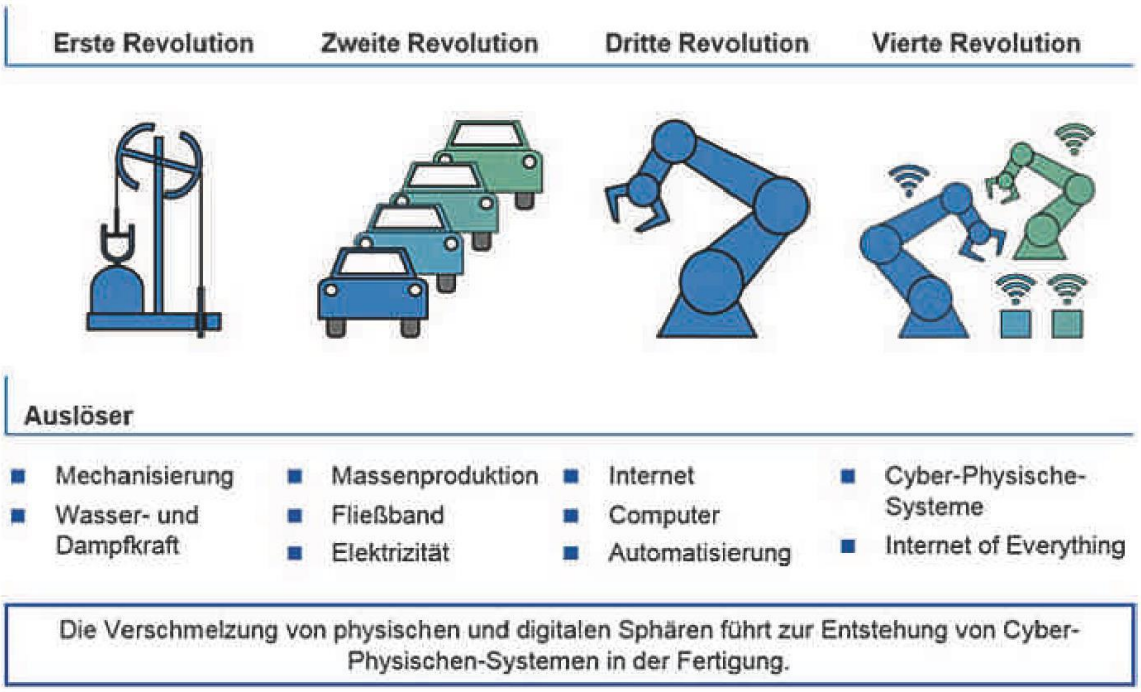


Abbildung 1: Der Ablauf der industriellen Revolution [5]

Abbildung 1 [5] zeigt die vier Hauptphasen der Industrialisierung seit der ersten industriellen Revolution. Diese Reihe von Revolutionen verkörpert eine Entwicklung von einfachen Maschinen zu hochintelligenten Systemen, wobei jede Stufe wesentlich zur Steigerung der Produktionskapazität und zu Veränderungen im gesellschaftlichen Leben beiträgt. Die Schlüsseltechnologie für Industrie 4.0 wächst mit der Zeit. Abbildung 2 veranschaulicht die vier Kategorien von Schlüsseltechnologien und die drei Infrastrukturthemen von Industrie 4.0 [40]. Das Internet der Dinge ermöglicht die Vernetzung von Geräten, die die Geräteebene und die Entscheidungsebene überbrücken. In diesem dichten Netz des Informationsaustauschs werden auch große Mengen an Daten erzeugt und gesammelt. Die Big Data Analytics ist der Wissenschaftszweig, der diesem Trend folgt. Andere Technologien wie Augmented und Virtual Reality, Cloud Computing, additive Fertigung oder dreidimensionaler Druck und neu entwickelte Robotik definieren die Industrie 4.0 neu [49].

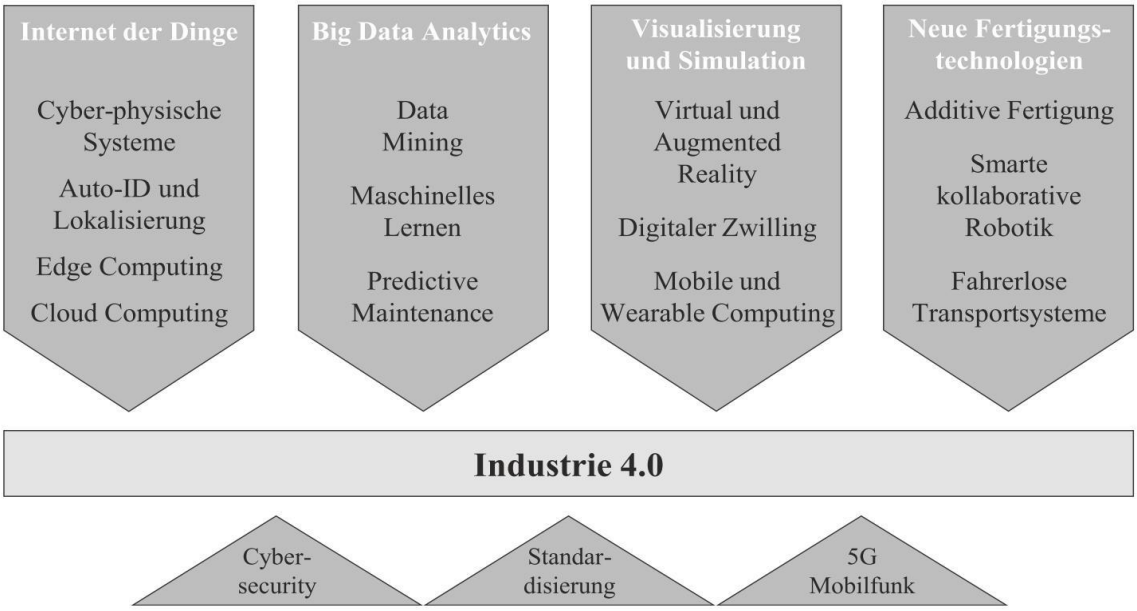


Abbildung 2: Schlüsseltechnologien und Infrastruktur in der Industrie 4.0 [40]

2.2 Cyber- Physical Systems (Cyber-Physische Systeme)

CPS(Cyber-Physische Systeme), laut einer Studie der deutschen Akademie der Technikwissenschaften, sind eingebettete Systeme, die physikalische Daten durch Sensoren erfassen und reale Prozesse über Aktoren beeinflussen. [40] Die wichtigsten Bestandteile und Funktionen des CPS sind nachstehend aufgeführt:

- Eingebettete Systeme: Hierunter fallen Geräte wie Gebäude, Verkehrsmittel und medizinische Geräte. Außerdem gehören Prozesse in den Bereichen Logistik, Koordination und Management sowie Internet-Dienstleistungen dazu.
- Datenerfassung und -reaktion: Durch Sensoren erfolgt die direkte Erfassung physikalischer Daten, während Aktoren in physikalische Vorgänge eingreifen.
- Datenverarbeitung: Daten werden für die aktive oder passive Interaktion mit der physischen und virtuellen Welt analysiert und gespeichert.
- Vernetzung: Durch digitale Netze besteht eine Verbindung, sowohl drahtlos als auch drahtgebunden, auf lokaler sowie globaler Ebene.
- Datennutzung: Es erfolgt ein Zugriff auf weltweit zugängliche Daten und Dienste.

– Mensch-Maschine-Interaktion: Mit multimodalen Mensch-Maschine-Schnittstellen werden differenzierte Möglichkeiten für Kommunikation und Steuerung geboten, wie z.B. die Verwendung von Sprache und Gesten [57].

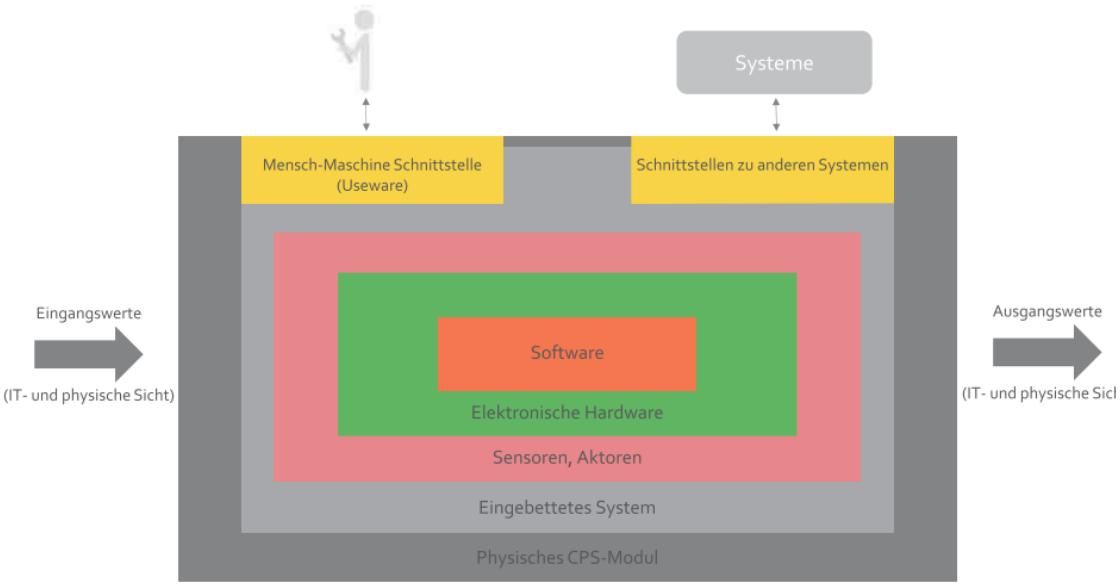


Abbildung 3: Struktur eines CPS-Moduls [59]

Die Struktur eines cyber-physischen Systems kann durch ein "Zwiebelschalen"-Modell dargestellt werden, wie in Abbildung 3 gezeigt. In vielerlei Hinsicht stellen diese Systeme ein sogenanntes System von Systemen dar, bei dem geschaffene Systeme wiederholt mit anderen Systemen kombiniert werden, um übergeordnete Systeme zu bilden. [7] Das Modell des Schemas ist ein CPS des zentralen Teils, der die Optimierung des Fertigungsablaufs als Ganzes darstellt. Der zentrale Punkt besteht aus Software, die durch elektronische Hardware sowie Sensoren und Aktoren ergänzt wird, wodurch ein integriertes System entsteht. Die Mensch-Maschine-Schnittstelle ihrerseits ermöglicht die Interaktion mit diesem System, da hier eingehende Eingaben sowohl aus dem IT- als auch aus dem physischen Bereich angenommen und verarbeitet werden. Umgekehrt erscheinen die Schnittstellen zu anderen Systemen als Verbindung zu internen und externen Ressourcen, die von anderen Fertigungssystemen über Werkzeuge, Lagersysteme, Informationstechnologiesysteme wie ERP, MES, PLM und

Supportsysteme reichen. Die Integration mit externen Lieferanten ist möglich, einschließlich bereitgestellter IT-Systeme und anderer Fertigungsanlagen[59].

CPS bietet eine gute Infrastruktur für den Prozess der prädiktiven Instandhaltung. Zum Beispiel kann der Prozess der Fehlerdiagnose einer zu überwachenden Komponente oder Anlage, die Verarbeitung der Daten und die endgültige Ausgabe der Zustandsparameter mithilfe der CPS-Struktur realisiert werden.

2.3 Big Data

Big Data-Konzepte haben im Laufe der Jahre eine Evolution erfahren. Im Jahr 2001 leistete Doug Laney, damals Experte bei der META Group, einen wichtigen Beitrag zum Verständnis von Big Data. Zwar wurde der Begriff Big Data in dieser Arbeit nicht direkt erwähnt. Die Arbeit begann jedoch mit der Identifizierung von drei Schlüsselaspekten von Big Data, die gemeinhin als die "3 V's" bezeichnet werden [28]. "Volume" steht für die Tatsache, dass mit der Produktion und Erfassung von Daten, die Datenmenge immer weiter anwächst; "Velocity" repräsentiert die Aktualität von Big Data, was bedeutet, dass die Datensammlung und -analyse schnell und im Voraus erfolgen muss, um den potenziellen kommerziellen Wert aus Big Data zu erschließen; "Variety" weist auf die Vielfalt der Daten hin, die von halbstrukturierten und unstrukturierten Daten (Audio, Video, Webseiten, Text) bis zu herkömmlichen strukturierten Daten reicht [9].

Im Rahmen dieser Arbeit ist " Industrielle Big Data " ein würdigeres Thema für die Diskussion. Im Unterschied zu Big Data im Internetbereich sind industrielle Big Data strukturierter, relevanter und aktueller [30] und eignen sich besser für die Speicherung in großem Maßstab. Industrielle Big Data ist ein allgemeiner Begriff für alle Datenkategorien, die über den gesamten Produktlebenszyklus in der Industrie generiert werden, von Aufträgen, Forschung und Entwicklung, Planung, Technologie, Herstellung und Vertrieb. Industrielle Big Data stammen in der Regel von verschiedenen Einheiten im gesamten Fertigungsbereich, z. B. von Sensoren, intelligenten Geräten, Logistik-, Elektro- und Werkstattpersonal. Im Allgemeinen können diese verteilten Daten in verschiedene Quellen unterteilt werden:

- **Geräte-/Sensordaten**: Prozessdaten, Betriebsdaten, Umgebungsdaten usw. von Geräten im Kontext des IoT (z. B. Sensoren, Aktoren, RFID-Geräte und SCADA-Daten usw.).

- **Unternehmensbetriebsdaten**: umfassen vor allem Unternehmensführung, Marketing, Beschaffung, Inventar, Geschäftsplan, etc. Die Analyse von Beschaffungs-, Lager-, Verkaufs- und anderen Daten hilft dabei, die Bestände zu verringern und eine schlanke Produktion zu erreichen.
- **Produktionskettendaten**: umfassen Lieferkettenmanagement, Kunden-beziehungsmanagement, Umweltmanagementsystem, Produktmarkt und makroökonomische Daten. Die Analyse von Produktionskettendaten kann Unternehmen dabei behilflich sein, neue Wachstums- oder Verbesserungspotenziale zu finden und neue Perspektiven für die Entwicklung der Unternehmensstrategie zu eröffnen [26].

2.4 Industrielles Internet of Things

Ein Industrielles Internet of Things (IIoT) ist ein vernetztes System zwischen der physischen Umgebung und den Datenverarbeitungsfunktionen von Verwaltungsgeräten. Durch die Integration von Datenverarbeitungsfunktionen und der physischen Umgebung verbindet das IIoT auf intelligente Weise unabhängige Geräte, um eine selbstanpassende Interaktion zu erreichen, sodass die Teilsysteme voneinander wissen und eine wirksame Koordinierung erreichen können, wodurch die Betriebslogik automatisch entsprechend den Aufgabenanforderungen angepasst und konfiguriert wird. Allgemein gesagt, sind die beiden Hauptfunktionen des IIoT:

- Effiziente Informationsverbindung auf der physikalischen Ebene, d.h. Sicherstellung der Datenerfassung in Echtzeit in der physikalischen Umgebung und der Informationsrückmeldung im Cyberspace;
- Intelligente Datenverwaltung, Analyse- und Berechnungsmöglichkeiten im Cyberspace [26].

Um diese Funktionen zu erfüllen, müssen IIoT-Systeme aus intelligenten Systemanwendungen, Mikrocontrollern, drahtlosen Sensoren und intelligenten Sicherheitsmechanismen sowie einer netzweiten Hochgeschwindigkeits-Datenkommunikationsinfrastruktur einschließlich Cloud und Edge Computing bestehen. Zu den Vorteilen des IIoT gehören erhöhte Sicherheit, verbesserte Zuverlässigkeit, intelligente Verbrauchsmessung, Bestandsverwaltung, Geräteverfolgung und intelligente Anlagen [33].

2.4.1 Sensor

Sensoren werden für die Messung quantitativ und qualitativ von physikalischen, chemischen, klimatischen, biologischen sowie medizinischen Größen eingesetzt. Ein typischen Sensoren ist eine Sensorelement-Auswerteelektronik. Beim Sensorelement wird die Messung der unelektrischen Eingangsgrößen, unter Ausnutzung naturwissenschaftlicher Gesetzmäßigkeiten, z. B. piezoelektrischer oder magnetoresistiver Effekt usw., durch Einwirkung der unelektrischen Eingangsgrößen in elektrische Ausgangssignale umgewandelt. In der Auswerteelektronik werden die Ausgangssignale von der Schaltungselektronik oder von Softwareprogrammen zu Sensorausgangssignalen verarbeitet, die für Steuerungs- oder Auswertezwecke verwendet werden können [18].

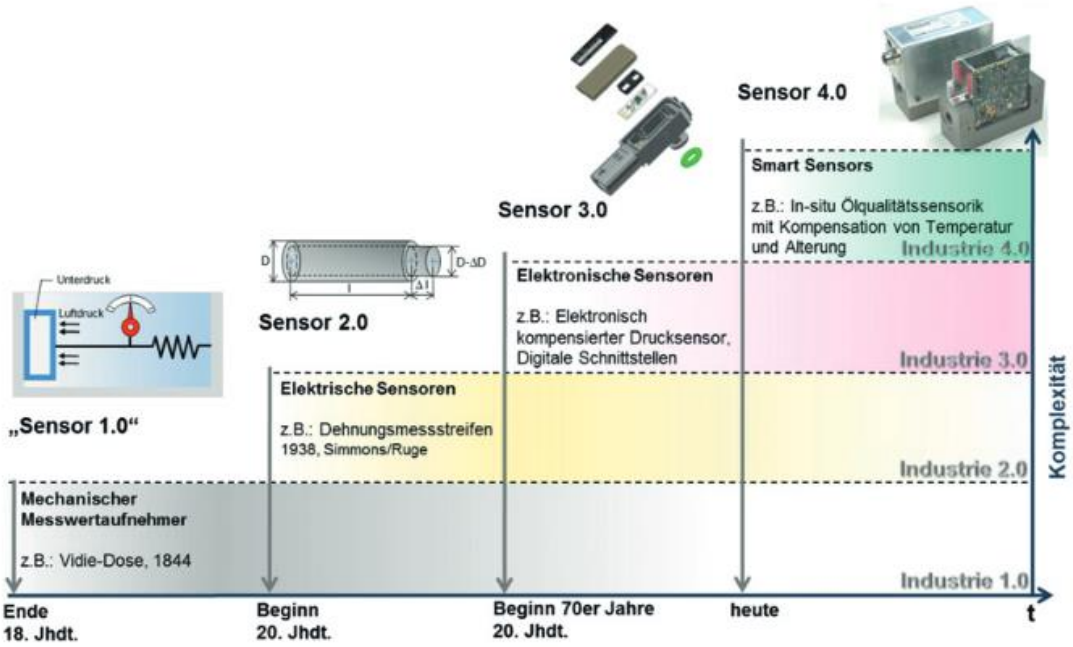


Abbildung 4: Historische Entwicklung von Sensoren [46]

Rein mechanische Sensoren, wie beispielsweise der in der Mitte des 19. Jahrhunderts eingeführte Verdi-Kanister zur Druckmessung, markieren den Beginn der Sensorik als Sensor 1.0. Elektrische Sensoren, darunter der klassische Dehnungsmessstreifen aus dem Jahr 1938, repräsentieren Sensor 2.0. Die Entwicklung setzte sich fort mit elektronischen Sensoren, wie den in den 1970er Jahren eingeführten temperaturkompensierten Drucksensoren, als Sensor 3.0. Bis heute ist die Schaltung und Akzeptierung weit verbreiteterer elektronischer Sensoren, auch Smart Sensors oder

intelligente Sensoren, als Sensor 4.0 etabliert. Diese unterscheiden sich von herkömmlichen elektronischen Sensoren durch umfangreiche interne Funktionen, zum Beispiel die Erfassung und Verknüpfung mehrerer Parameter und die Selbstdiagnose. Außerdem bieten sie umfangreiche Kommunikationsmöglichkeiten, nicht nur für die Ausgabe von Messwerten, sondern auch für die Parametrierung des Messsystems. Abbildung 4 liefert einen historischen Überblick über diese Entwicklung und veranschault, wie eng die Evolution der Sensorik mit dem Fortschritt der Industrialisierung verknüpft ist [46].

Sensoren sind auch die zentrale Komponente der Waren und Lösungen und können die wichtigsten Prozessparameter überwachen, die sich auf die Leistung, die Wirtschaftlichkeit und den Schutz auswirken. Die Einführung einer völlig neuen Sensorklasse umfasst Firmware-Prozesse für den Netzwerk-Stack und die Sensoranwendungen. Die Vernetzung von drahtlosen Sensoren ist heute eine Tatsache, und die Möglichkeiten für Fortschritte in diesem Bereich werden weiter zunehmen. Gerätenetzwerke müssen die großen Datenmengen, die von den zahlreichen Systemsensoren erfasst werden, austauschen und koordinieren, um die Vorteile des IIoT zu nutzen. Es kann automatische Steuerungsfunktionen und Geräte wie Motoren einfach überwachen, kalibrieren und steuern und eine vorausschauende Planung und vorausschauende Wartung ermöglichen [24].

2.4.2 IIoT Edge Device

In der realen Welt erzeugen die oben genannten Sensoren und IIoT-Geräte ständig große Datenmengen. Diese Daten sind für viele moderne Technologieanwendungen wie vorausschauende Wartung entscheidend. Entsprechende Algorithmen für maschinelles Lernen oder Deep Learning können Experten dabei helfen, Gerätezustände aus diesen Daten zu extrahieren und Entscheidungen zu treffen. In realen industriellen Szenarien haben die Geräte jedoch oft eine begrenzte Rechenleistung. Dies schränkt die Implementierung von Algorithmen für maschinelles Lernen auf diesen Geräten ein. Daher werden die Daten in der Regel zur Berechnung in die Cloud übertragen. Dieser Ansatz der Übertragung von Rohdaten zu Cloud-Servern bringt jedoch einige Nachteile mit sich, wie z. B. höhere Kommunikationskosten, Verzögerungen bei der Systemreaktion und die Anfälligkeit privater Daten für Lecks. Um diese Probleme zu lösen, hat sich das Edge Computing etabliert [8, 35].

Gartner definiert Edge Computing als "Teil einer verteilten Computertopologie, bei der die Informationsverarbeitung in der Nähe des Randes angesiedelt ist - dort, wo Dinge und Menschen diese Informationen produzieren oder konsumieren" [13]. Die direkte Übertragung dieser Daten an die Cloud führt oft zu steigenden Betriebskosten. Da Edge Computing als wichtiges Paradigma für IoT-basierte Systeme entstanden ist, wird versucht, Geräte am Netzwerkrand für so viele Berechnungen wie möglich zu nutzen, anstatt ausschließlich die Cloud zur Datenverarbeitung zu verwenden [35].

Als Beispiel untersuchte Floyer [35, 51] die Datenverwaltungs- und Verarbeitungskosten eines entfernten Windparks mit einem reinen Cloud-System im Vergleich zu einem kombinierten Edge-Cloud-System. Der Windpark bestand aus mehreren datenerzeugenden Sensoren und Geräten wie Überwachungskameras, Sicherheitssensoren, Zugangssensoren für alle Mitarbeiter und Sensoren an Windturbinen. Das Edge-Cloud-System erwies sich als 36 % kostengünstiger und kostete nur 28.927 $ im Vergleich zu dem reinen Cloud-System, das 80.531 $ kostete. Außerdem wurde beobachtet, dass das Volumen der zu übertragenden Daten um 96 % geringer war als bei dem reinen Cloud-System. Ähnliche Beispiele umfassen ein Videoanalyse-System für autonome Drohnen, bei dem ein Edge-Computing-System verwendet wird, um Bandbreite zu sparen, was die Notwendigkeit, Videodaten in die Cloud zu übertragen, um bis zu dem Zehnfachen reduziert.

Die Vorteile des Edge Computing sind zwar vielfältig, doch die Rechenleistung des verteilten Rechnens unterliegt der Realität praktischer und wirtschaftlicher Faktoren. Um Rechenressourcen im Edge-System zu sparen, müssen neue Ansätze in der Phase der Datenerfassung oder -verarbeitung gefunden werden. In [53] schlagen Yan et al. einen Datenverarbeitungsalgorithmus vor, der auf Lebesgue-Sampling basiert. Dieser Algorithmus führt den Algorithmus häufiger aus, wenn die Fehler schneller wachsen, und weniger häufig, wenn die Fehler langsamer wachsen. Dadurch werden Rechenressourcen gespart. Das Lebesgue-Sampling als vielversprechender Algorithmus trägt dazu bei, den weit verbreiteten Einsatz von Edge Computing zu fördern.

2.5 Zuverlässigkeitstechnik in der Industrie 4.0

In der technischen Praxis werden viele Systeme für bestimmte Aufgaben konzipiert und müssen während ihrer gesamten Lebensdauer sicher funktionieren. Die Qualität und Leistung dieser Systeme werden durch Alterung, unterschiedliche Belastungen und Betriebsumgebungen allmählich beeinträchtigt, was letztendlich zu ihrem endgültigen Ausfall führen kann [38]. Daher hat sich die Zuverlässigkeitstechnik entwickelt. Der traditionelle technische Ansatz in der Zuverlässigkeitstechnik basiert hauptsächlich auf statistischer und mathematischer Modellierung und qualitativen Methoden [6]. Hier bietet der modellbasierte Ansatz der Zuverlässigkeitstechnik einige entscheidende Vorteile. Wenn die Ausfallparameter genau geschätzt werden, können die Modelle eine gute Prognosegenauigkeit aufweisen. Allerdings sind parametrische Darstellungen von Ausfallprozessen für immer komplexere mechanische Systeme oft schwer herzuleiten. Änderungen des Systemzustands im Laufe der Zeit können zu einem Anstieg der Berechnungskomplexität führen, insbesondere bei Systemen mit einer hohen Zustandsraumdimension. Außerdem ist für die Parameterschätzung eine große Menge an Daten erforderlich [38].

Sensoren und IoT-Geräte verbreiten sich immer weiter und ermöglichen die Sammlung einer großen Menge an Daten über Prozesse und Geräte während des Maschinenbetriebs [19]. Dies hat zu einem neuen Schritt in der Entwicklung der Zuverlässigkeitstechnik geführt, wobei die Entwicklung des PHM einen wesentlichen Aspekt darstellt.

Abbildung 5 skizziert den PHM-Prozess. Während des Betriebs des Geräts bewertet das PHM-System zunächst den Gesundheitszustand des Geräts anhand der vom industriellen Datenerfassungssystem gelieferten Daten. Anschließend wird der ausgegebene Zustandswert mit einer vordefinierten Zustandsschwelle verglichen (die in der Regel von einem erfahrenen Zuverlässigkeitsingenieur festgelegt wird). Falls der Zustandswert unter dem Standardwert liegt, wird die verbleibende Nutzungsdauer der kritischen Maschinenkomponenten (in der Regel bewegliche Teile wie Spindeln, Servomotoren, Getriebe usw.) vorhergesagt und die Fehlerdiagnose durchgeführt. Die Diagnosen werden an das Modul Instandhaltungsvorschläge Unterstützung weitergeleitet. Daraufhin werden auf der Basis des aktuellen Zustands der Anlage geeignete Instandhaltungsempfehlungen ausgegeben [26]. Als wesentlicher Bestandteil

der Zuverlässigkeitstechnik und des PHM werden im Folgenden Instandhaltungsstrategien und prädiktive Instandhaltung beschrieben.

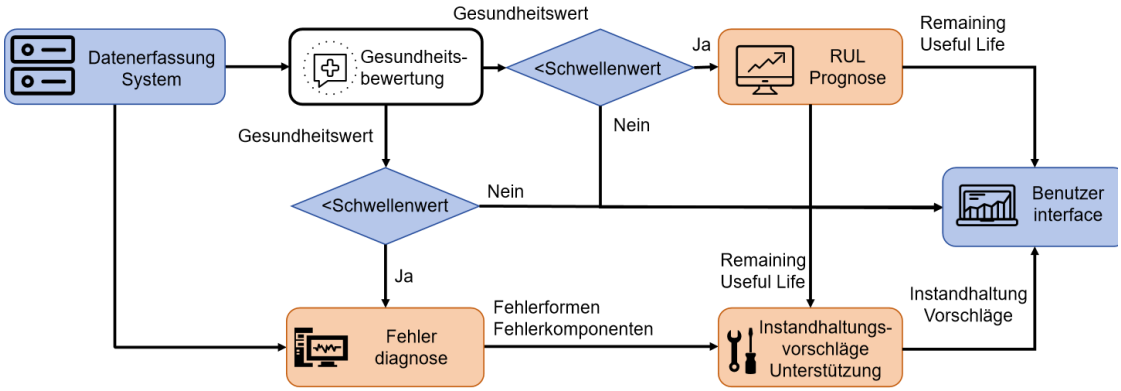


Abbildung 5: Schlüsselkomponenten im PHM[26]

2.5.1 Instandhaltungsstrategie

In jüngsten Studien wurde jedoch festgestellt, dass es in der Industrie keine einheitliche Definition der Instandhaltungsstrategie gibt. Um dies zu klären, werden in diesem Beitrag Instandhaltungsstrategien und geeignete Systeme in Anlehnung an die Literatur in Tabelle 1 kategorisiert [34, 39]. Um schwerwiegende Folgen wie Maschinenstillstände, die eine Fortsetzung der Produktion verhindern, zu vermeiden, wurde die präventive Instandhaltung entwickelt. Diese basiert auf der Zusammenfassung von Ausfallintervallen aus historischen Ausfallraten oder auf der Durchführung von Instandhaltungsmaßnahmen gemäß den von den Herstellern empfohlenen Wartungszyklen [23]. Dies kann jedoch dazu führen, dass noch funktionsfähige Teile ersetzt werden, was die Instandhaltungskosten erhöht. Eine zu frühe Intervention kann aufgrund des Austauschs von Teilen mit einer noch langen Restnutzungsdauer (RUL) zu einer Verschwendung von Ressourcen führen, während eine zu späte Intervention katastrophale Ausfälle verursachen kann [34]. Wie bestimmt man also den genauen Zeitpunkt für die Instandhaltung? Die prädiktive Instandhaltung bietet eine Lösung, indem sie durch planmäßige Überwachung der Betriebsbedingungen und Trendanalyse der Komponentenverhalten den optimalen Zeitpunkt für Instandhaltungs-arbeiten bestimmt. Auf diese Weise minimiert die Betriebskosten durch Vermeidung vorzeitiger oder katastrophaler Ausfälle und maximiert die Zeit zwischen den Reparaturen [23].

Tabelle 1: Klassifizierung von Instandhaltungsstrategien und geeigneten Systemen in Anlehnung an [34, 39].

Instandhaltungsstrategie		Geeignete Systeme	Merkmale
Korrektive Instandhaltung		Nicht sicherheitskritische Systeme	- Wird nach Fehlererkennung ausgeführt - Beschränkt sich auf Reparatur- oder Austauschprozesse - Keine schwerwiegenden Folgen bei Ausfällen
Präventive Instandhaltung	Planmäßige Instandhaltung	Geeignet bei gravierenden Sicherheitskonsequenzen oder erheblichen Produktionsverlusten	- Regelmäßige oder sporadische Maßnahmen basierend auf statistischen Daten (Ausfallzeiten, Dauer der Instandhaltungsmaßnahmen) - Herausfordernd durch komplexe Degradationsmechanismen und Unsicherheiten - Vorteilhaft, wenn Ersatzteile schwer verfügbar sind
	Opportunistische Instandhaltung	Geeignet für Anlage mit langen, geplanten Ausfallzeiten (z.B. Kernkraftwerke) und Anlagen, bei denen teure Ausrüstungen für Wartungsarbeiten angemietet werden müssen (z.B. Kräne, Schiffe)	- Gleichzeitige Instandhaltungsmaßnahmen an mehreren Geräteelementen oder Subsystemen - Kombination von Instandhaltungen für Komponenten mit ähnlichen Ausfallraten und Betriebsbedingungen - Nutzung geplanter Stillstände oder unerwarteter Ausfälle zur gleichzeitigen Instandhaltung mehrerer Komponenten
Prädiktive Instandhaltung	Zustandsbezogene Instandhaltung	Systeme, bei denen ungeplante Stillstände vermieden werden sollen	- Basierend auf direkten Daten von der Ausrüstung zur Bewertung ihres Zustands - Einsatz von Überwachungssystemen und Techniken zur Fehlererkennung und diagnose - Geeignet für Systeme, bei denen die Vorteile der Vermeidung ungeplanter Stillstände die Kosten für Überwachungssysteme überwiegen
	Prescriptive Instandhaltung	Systeme, die operative Optimierung benötigen	- Nutzt Fehlerprognose und betriebliche Daten zur Optimierung des Instandhaltungseinflusses - Empfehlungen können Instandhaltungs- und Betriebsmaßnahmen umfassen

Die Instandhaltungsstrategien sollten auch für verschiedene Ausfallarten differenziert werden. Im Zusammenhang mit der Zuverlässigkeitstechnik beschreibt die Kurve der Ausfallrate über die Zeit (auch als Badewannenkurve bekannt) die Art des Ausfalls eines

Produkts in verschiedenen Zeitabschnitten. Abbildung 6 beschreibt die Ausfallarten von Anlagen in verschiedenen Lebensphasen [58]. Frühe Ausfälle passieren in den Anfangsphasen der Lebensdauer einer Komponente.

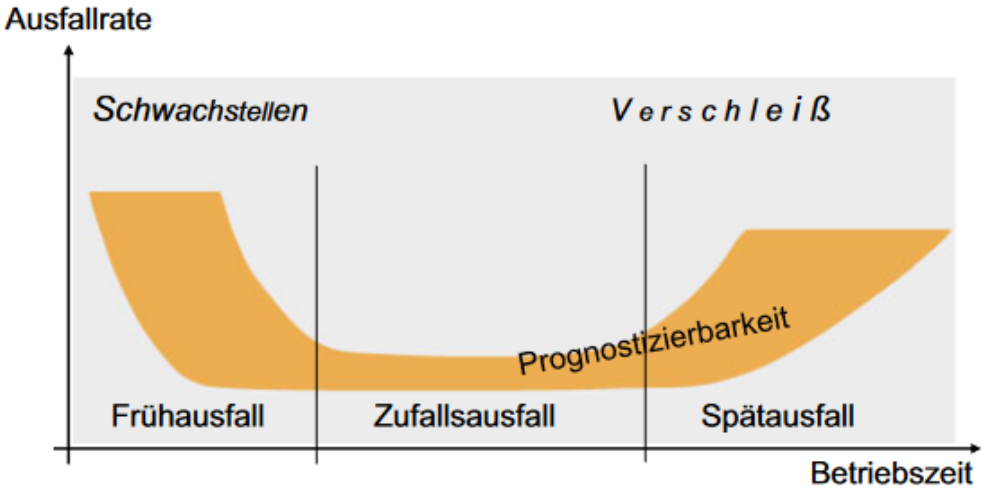


Abbildung 6: Ausfallverhalten über die Betriebszeit [58]

Aufgrund von Produktionsmängel sowie mangelhafte Gestaltung oder Fehlern bei der Montage führten zu Frühausfällen. Frühausfälle sind schwer prognostizierbar und können durch falsche Bedienung durch unqualifiziertes Personal verschlimmert werden. Bei Frühausfällen wird in der Regel eine korrigierende Instandhaltung durchgeführt. Sobald sich der Prozess stabilisiert hat, beginnt die Phase der zufälligen Ausfälle. In dieser Phase liegt die Ausfallrate in einem gleichmäßigen Bereich und die entsprechende Instandhaltungsstrategie ist die präventive Instandhaltung. In den späteren Phasen der altersbedingten Ausfälle fallen wichtige Komponenten durch planmäßige technische Abnutzung aus. Je genauer diese Ausfälle prognostiziert werden können, desto größer ist der Nutzen einer prädiktiven Instandhaltung. Einerseits kann der Bestand an Ersatzteilen optimiert und vermieden werden, dass teuer eingekaufte Ersatzteile lange im Lager liegen. Zum anderen lässt sich das Risiko eines Anlagenausfalls quantifizieren. Die prädiktive Instandhaltung verlängert die Lebensdauer der Anlagen so weit wie möglich und hält dabei die Kosten unter Kontrolle [58]. Zusammenfassend könnte die prädiktive Instandhaltung die profitabelste prädiktive Strategie im Kontext von Industrie 4.0 sein.

2.5.2 Prädiktive Instandhaltung

Im Rahmen der prädiktiven Instandhaltung sind datengesteuerte Ansätze die vielversprechendsten Forschungsrichtungen. Die Studie von T. Zonta et al. zeigt, dass ANNs und maschinelles Lernen, dargestellt durch Zufallswälder, viel Aufmerksamkeit erhalten haben. Die Verwendung von neuronalen Netzen und Deep Learning nimmt in der aktuellen Forschung zu [56]. Der wesentlichste Grund für die Entwicklung datengesteuerter Methoden liegt in den erheblichen Vorteilen der Prädiktiven Instandhaltung. Nach Correa et al. [11] können die Vorteile prädiktiver Instandhaltungssysteme wie folgt zusammengefasst werden:

- Niedrigere Instandhaltungskosten
- Weniger Ausfälle
- Kürzere Reparaturintervalle
- weniger Ersatzteillager
- Verlängerte Lebensdauer der Maschine
- Erhöhte Produktivität der Anlage
- Verbesserte Betriebssicherheit
- Überprüfung des Betriebsstatus neuer Ausrüstung
- Verbesserung der Gesamtrentabilität

Die prädiktive Instandhaltung gilt jedoch nicht für alle Anlagen und kann herkömmliche Instandhaltungssysteme nicht ersetzen, da ein Großteil der Anlagen nicht kritisch ist und schon lange im Einsatz ist und die Kosten für ihre Digitalisierung hoch sind, während der Nutzen gering ist [29]. Die Entscheidung für eine prädiktive Instandhaltung muss vor der Betrachtung des tatsächlichen Input-Output-Verhältnisses getroffen werden.

2.6 Wälzlager

Wälzlager sind ein Schlüsselelement in den meisten rotierenden Maschinenteilen und für etwa 40 % bis 50 % aller Ausfälle verantwortlich [2, 12]. Rotierende Maschinen kommen in fast allen Industriezweigen zum Einsatz, z. B. im Maschinenbau, in der Luft- und Raumfahrt und in der Energieerzeugung. Ob die Wälzlager nun Teil der Produktionsanlage oder Teil des Produkts sind, ihre Zuverlässigkeit ist für den normalen Betrieb der Maschine unabdingbar. Bevor die Zuverlässigkeit einer ganzen Maschine berücksichtigt wird, ist ein grundlegendes Verständnis der Wälzlager notwendig. Im Folgenden werden die grundlegenden Komponenten von Lagern und die häufigsten Ausfallarten beschrieben.

2.6.1 Komponenten von Wälzlager

In Abbildung 7 [2] sind die vier Bestandteile eines klassischen Wälzlagers dargestellt.: den Wälzkörpern, dem Käfig sowie den Innen- und Außenringen [48, 61]. Je nach Einsatzbedingungen können die Lager ganz oder teilweise aus Stahl, Kunststoff (PA66) oder Keramik (Si3N4) bestehen [45]. Wälzkörper können unterschiedliche Formen haben, wie z. B. Kugeln, Zylinderrollen, Nadel, Kegelrollen und Tonnenrolle. Wälzkörper werden in der Regel mit Käfigen zu Wälzkörperkränzen kombiniert und halten so einen gleichmäßigen Abstand ein [61].

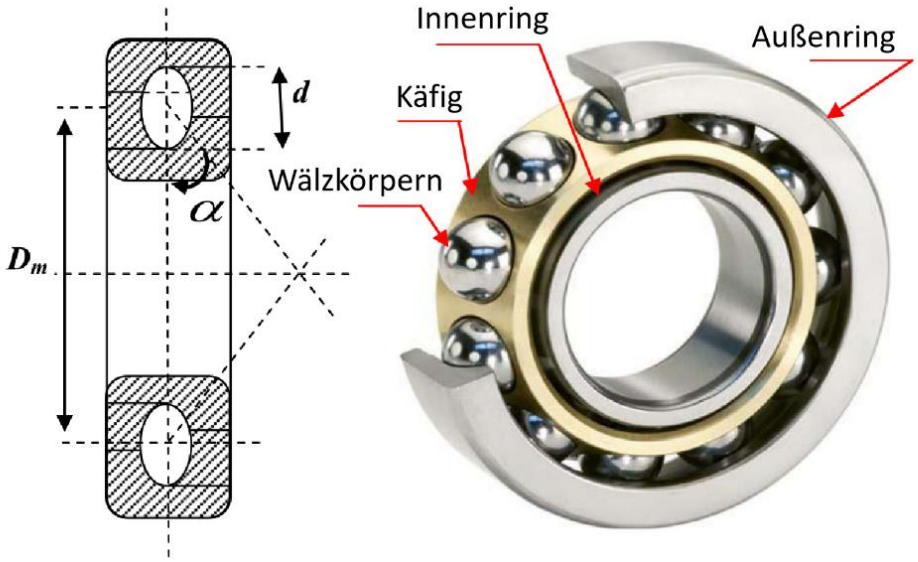


Abbildung 7: Aufbau und Bestandteile eines Wälzlagers [2]

2.6.2 Arten von Wälzlagerausfällen

Ein Wälzlagerausfall kann auf einen Fehler an jedem der vier Lagerbestandteile zurückzuführen sein, aber es wird geschätzt, dass 90 % aller Fehler am Innen- und Außenring auftreten [48]. Dies kann darauf zurückzuführen sein, dass die Ringe ständig unter Spannung stehen, während die Kugeln im Gegensatz dazu rotieren und ihre Kontaktfläche ständig ändert und ein Käfig kaum eine Last halten kann [45, 48]. Jeder Ausfalltyp in Kugellagern hat eine andere fehlerspezifische Frequenz (Gleichungen (1) - (4)), die durch das Abrollen der Kugeln auf der Oberfläche der Ringe verursacht wird. Ein periodischer Impuls tritt immer dann auf, wenn in einem der Bestandteile eine Abnormalität vorhanden ist, die von der Drehzahl der Achse f_r, den Geometrieparametern Anzahl der Kugeln (N_{Kugeln}) Teilkreisdurchmesser (D_m), Kugeldurchmesser (d) und Kontaktwinkel (α) abhängt (siehe Abbildung 7) [48]. Der

Berührungswinkel beschreibt den Nennwinkel zwischen der Radialebene und der Drucklinie [61]. Der Außenring, der Innenring, die Kugel und der Käfig haben alle unterschiedliche Ausfallfrequenzen [2, 45, 48].

$$f_{\text{Käfig Ausfall}} = \frac{1}{2} * f_r * \left(1 - \frac{d}{D_m} * \cos\alpha\right) \tag{2.1}$$

$$f_{\text{AußenringAusfall}} = \frac{1}{2} * f_r * N_{\text{Kugel}} \left(1 - \frac{d}{D_m} * \cos\alpha\right) \tag{2.2}$$

$$f_{\text{InnerringAusfall}} = \frac{1}{2} * f_r * N_{\text{Kugel}} \left(1 + \frac{d}{D_m} * \cos\alpha\right) \tag{2.3}$$

$$f_{\text{KugelAusfall}} = \frac{1}{2} * f_r * \frac{D_m}{d} * \left(1 - \left(\frac{d}{D_m} * \cos\alpha\right)^2\right) \tag{2.4}$$

Die Schwingungsanalyse kann zur Überwachung des Betriebszustands von Lagern eingesetzt werden. Da sie zerstörungsfrei ist, wird sie in der Industrie immer beliebter und anerkannt. Die Amplitude des Vibrationssignals kann auf die Schwere des Problems hinweisen, während die Frequenz die Quelle des Defekts anzeigen kann. Abnormale Schwingungen sind in der Regel der erste Hinweis auf einen möglichen Maschinenausfall. Zu den Bedingungen, die zu diesen Schwingungen führen, gehören Unwucht, Ausrichtungsfehler, lose Komponenten, Ausfall von Lagerkomponenten und vieles mehr. Schwingungsanalysewerkzeuge und -systeme können viele dieser Probleme in einem frühen Stadium erkennen, so dass das Personal in der Lage ist, rechtzeitig Reparaturarbeiten durchzuführen [15, 27].

2.7 Machine Learning Methoden

Eine umfangreiche Sammlung historischer Daten, die während des Betriebs und der Wartung von Systemen erfasst wurden, stellt eine wesentliche Ressource zur Bewertung des Systemgesundheitszustands dar. Als Kernkomponente der prädiktiven Wartung zeigen datengesteuerte Methoden des maschinellen Lernens (ML) signifikante Vorteile im Bereich der Fehlerdiagnose und -prognose (FDP). Diese Methoden sind in der Lage, Gesundheitsmerkmale des Systems aus historischen Daten zu identifizieren oder zu erlernen und verborgene Informationen in den Daten zu extrahieren. Selbst bei unzureichendem Verständnis des physikalischen Vorwärtsmodells können sie das zukünftige Verhalten des Systems präzise analysieren und Prognosen. Insgesamt erfordern ML-Methoden keine Annahmen über die Datenverteilung, verfügen über

stärkere FDP-Fähigkeiten und benötigen weniger Verarbeitungsschritte sowie manuelle Eingriffe. Dies macht den FDP-Prozess reibungsloser und intelligenter und reduziert die Anforderungen an Vorwissen bei der Modellierung komplexer Komponenten oder Systeme [41].

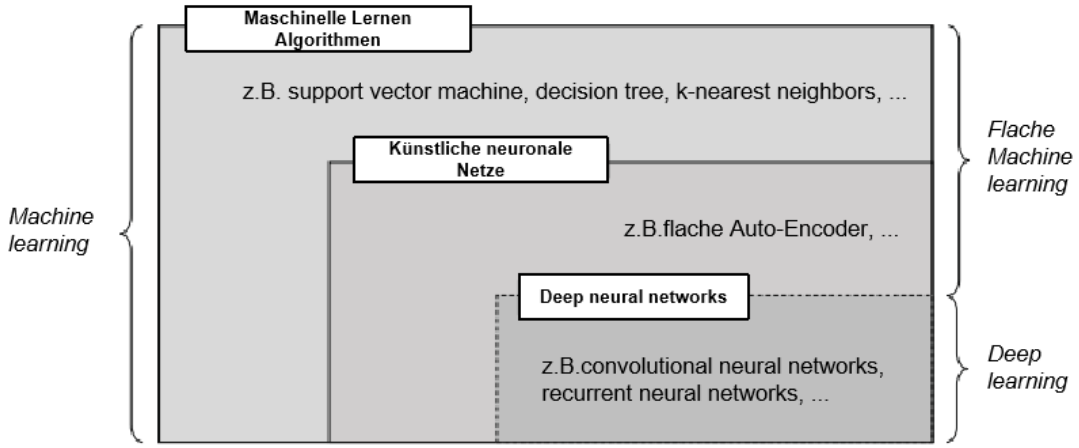


Abbildung 8: Beziehungsdiagramm von Machine Learning, künstlichen neuronalen Netzen und Deep Learning [22]

Abbildung 8 zeigt die Beziehung zwischen maschinellem Lernen, künstlichen neuronalen Netzwerken und Deep Learning. Die äußerste Ebene besteht aus maschinellen Lernalgorithmen, darunter klassische Algorithmen wie Support Vector Machines, Entscheidungsbäume und K-Nearest Neighbors. Diese Ebene kann als flaches maschinelles Lernen betrachtet werden. Die nächste innere Ebene bildet das künstliche neuronale Netzwerk, das relativ einfache neuronale Netzwerkmodelle wie flache Autoencoder umfasst. Die kerne Ebene sind tiefe neuronale Netzwerke, einschließlich Convolutional Neural Networks (CNN) und Recurrent Neural Networks (RNN), die den Hauptinhalt des Deep Learning darstellen. Deep Learning ist tatsächlich ein Teilbereich des maschinellen Lernens, dessen Merkmal der Einsatz komplexer neuronaler Netzwerke zur Lösung anspruchsvollerer Probleme ist.

Überwachtes Lernen, unüberwachtes Lernen und verstärkendes Lernen sind die drei Haupttypen des maschinellen Lernens. Überwachtes Lernen basiert auf Datensätzen, die sowohl Eingaben als auch gelabelte Ausgaben enthalten, und nutzt das Training von Modellen zur Prognose der Zielvariablen. Es wird in Regressions- und Klassifikationsprobleme unterteilt, die auch die am häufigsten in der prädiktiven Wartung eingesetzten Typen des maschinellen Lernens darstellen. Regressionsaufgaben korrespondieren zu unüberwachtem Lernen, das in unmarkierten Daten potenzielle

Muster wie Clusterbildung (Gruppierung von Daten) und Dimensionsreduktion (Vereinfachung der Datenrepräsentation) entdeckt, die häufig in Anwendungen wie der Kundensegmentierung eingesetzt werden. Verstärkendes Lernen definiert hingegen den Systemzustand, das Ziel und die zulässigen Aktionen und lässt das Modell durch Versuch und Irrtum lernen, um die Belohnung zu maximieren. Es findet breite Anwendung in Spielen und Multi-Agenten-Systemen.

2.8 Machine Learning Methoden zur Fehlerdiagnose und RUL-Prognose

Fehlerdiagnose und -prognose sind im Rahmen des datengesteuerten modernen PHM und der Zuverlässigkeit zwei der wichtigsten Säulen. Fehlerdiagnose und RUL-Prognose sind die Auswüchse dieser beiden. Dieser Abschnitt bietet einen umfassenden Überblick über die aktuelle Literatur im Bereich der RUL-Prognose und der Fehlerdiagnose von Wälzlager auf der Grundlage der Arbeiten [42, 47]. Durch die Klassifizierung und Strukturierung der relevanten Studien werden Forschungsrichtungen, Datenquellen und Algorithmen Typen in einem Sankey-Diagramm zusammengefasst. Dies trägt dazu bei, den aktuellen Entwicklungsstand der RUL-Prognose und Fehlerdiagnose von Wälzlagern zu klären und bietet wertvolle Einblicke für zukünftige Forschungen.

2.8.1 Fehlerdiagnose

Abbildung 9 zeigt die Korrelation zwischen Forschern, Jahren, Algorithmen und Datensätzen bei der Fehlerdiagnose von Wälzlagern. Wie in Abbildung 9 zu sehen ist, entfallen auf CNNs, SVMs und baumbasierte Algorithmen mehr als 80 % der gesamten Forschung. Beispielsweise erzielten Zhao et al. [55] kombinierten beispielsweise 1D-CNN mit Batch-Normalisierung und erzielten damit gute Klassifizierungsergebnisse für Lagersignale in zwei Datensätzen. Lei et al. [25]entwickelten ein neuronales Wavelet-Netz zur Erkennung von Fehlerstellen und Fehlerstärke. Liang et al. verwendeten eine unbeaufsichtigte Deep-Transfer-Learning-Methode mit dem Isolated-Forest-Algorithmus, um die Proben zunächst automatisch zu klassifizieren und zu kennzeichnen, und nutzten diese Kennzeichnungen dann für das Deep-Learning-Training, um ein hohes Maß an Genauigkeit und Allgemeinheit zu erreichen [32].

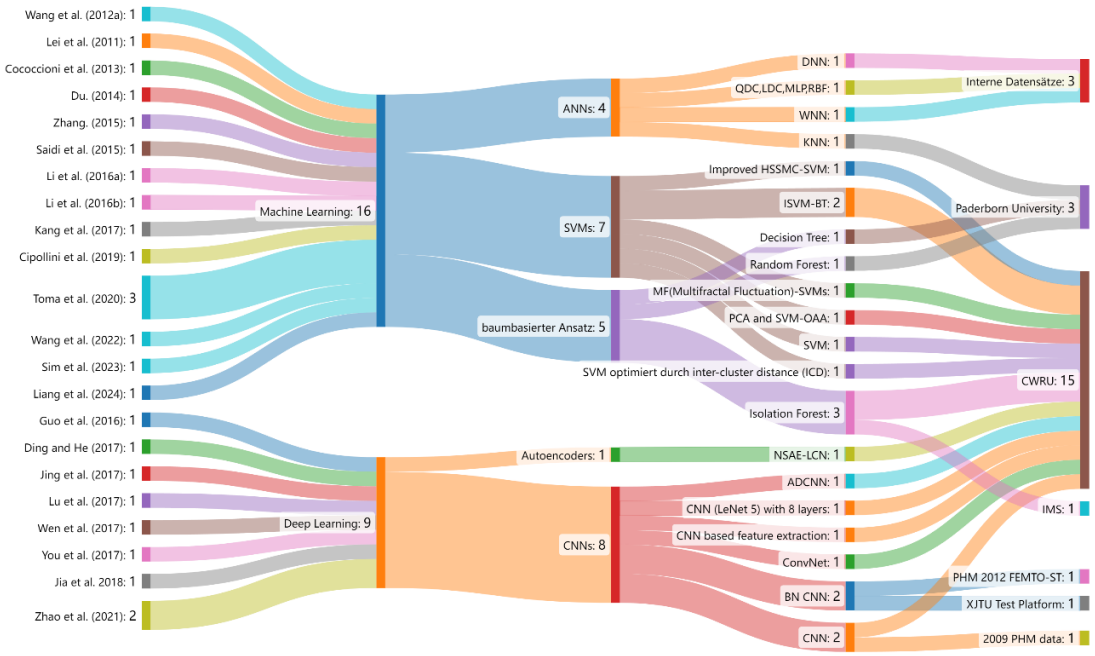


Abbildung 9: Korrelationsdiagramm in der Fehlerdiagnose von Wälzlagern in Anlehnung an [42, 47]

2.8.2 RUL-Prognose

Abbildung 10 zeigt die Korrelation zwischen Forschern, Jahren, Algorithmen und Datensätzen in der RUL-Prognose von Wälzlager. Die Zahl der Prognoseliteratur für RUL ist etwas geringer als die für die Fehlerklassifizierung. Mögliche Gründe dafür sind, dass die RUL-Prognose von Natur aus schwieriger ist als die Fehlerklassifikation und dass für die RUL-Prognose in der Regel die vollständigen Lebenszyklusdaten des Lagers benötigt werden und klassifizierte Datensätze wie CWRU nicht direkt für die RUL-Prognose verwendet werden können. Die Algorithmen der RNN-Familie können leicht an Zeitreihendaten angepasst werden, was als vielversprechendes Forschungsgebiet für die RUL-Prognose angesehen wird.

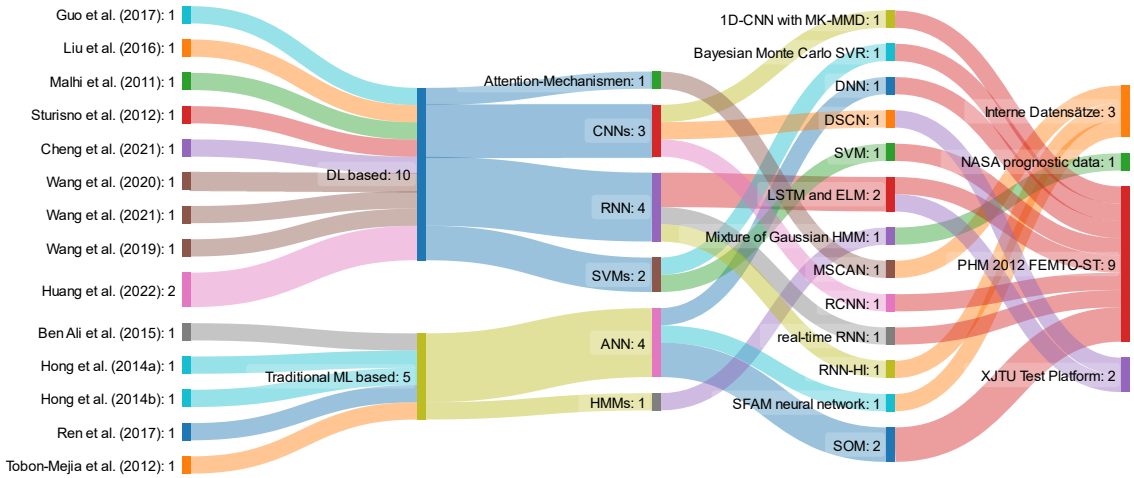


Abbildung 10: Korrelationsdiagramm in der RUL-Prognose von Wälzlager in Anlehnung an [42, 47]

2.8.3 Die Familie der rekurrenten neuronalen Netze (RNNs)

Beim Schätzen der RUL von Lagern ist es notwendig, zeitbezogene Prozesse zu berücksichtigen. Rekurrente Neuronale Netzwerke (RNNs) wurden von Anfang an entwickelt, um Zeitreihendaten zu modellieren [47]. Daher wählt diese Studie RNNs als RUL-Prognosemodell für Lager aus. Um jedoch das Problem der Neigung zu Gradienten Explosionen bei RNN auszugleichen, wurden LSTM-Netzwerke eingeführt, die theoretisch eine bessere Prognoseleistung bieten. BILSTM, eine Variante des LSTM, verwendet zwei LSTM-Modelle für die Prognose. Im Folgenden werden die konzeptionellen Komponenten und Eigenschaften dieser drei Modelle kurz vorgestellt.

2.8.3.1 Recurrent Neural Network (RNN)

Verglichen mit Feedforward-Netzwerken liegt der Vorteil von RNN darin, dass durch die Einführung von Zeitschritten das Konzept der Zeit integriert wird. In RNN-Modellen werden neben den üblichen Eingaben und Ausgaben auch Kanten hinzugefügt, die benachbarte Schritte verbinden, sogenannte Rekurrenz Kanten. Diese Kanten ermöglichen den Neuronen, Selbstverbindungen zu bilden. Dadurch kann ein RNN bei jedem Zeitschritt t zusätzliche Eingaben vom vorherigen Zeitschritt t−1 über die rekurrenten Neuronen erhalten. Daher hängt die Ausgabe zum aktuellen Zeitpunkt t nicht nur von den aktuellen Eingabedaten x_t ab, sondern auch von den Informationen des vorherigen Zeitschritts t−1, h_{t-1} [16].

.

$$h_t = f(w_{hx} x_t + w_{hh} h_{t-1} + b_h) \tag{2.5}$$

$$\hat{y}_t = f(w_{yx} x_t + b_y) \tag{2.6}$$

Dabei ist $f(\cdot)$ eine Aktivierungsfunktion, w_{hx} die Gewichtungsmatrix von der Eingabeschicht zur verborgenen Schicht, w_{hh} die Gewichtungsmatrix zwischen den Zeitstufen innerhalb der verborgenen Schicht, und b_h sowie b_y sind Bias-Terme. Durch diesen Mechanismus kann ein RNN frühere Zustände speichern und diese Informationen nutzen, um zukünftige Ausgaben zu beeinflussen [16].

2.8.3.2 Long Short-Term Memory-Netzwerke (LSTM)

Um die Probleme des Gradientenverschwinden und -explodierens in RNNs zu überwinden, wurden Long Short-Term Memory-Netzwerke (LSTM) eingeführt. LSTM kann durch die Einführung von Gedächtniseinheiten langfristige Informationen speichern. In einem LSTM gibt es einen Knoten, der als „Zustand“ bezeichnet wird und durch feste Gewichtungen mit sich selbst verbunden ist, um vergangene Informationen zu bewahren. Zusätzlich wurden Eingangs-, Vergessens- und Ausgangstore hinzugefügt, wodurch das Netzwerk Informationen selektiv speichern oder vergessen kann. Dies löst das Problem des Gradientenverschwinden oder -explodierens und stellt sicher, dass RNNs langfristige Erinnerungen behalten können [16].

$$g_t = \phi(w_{gx} x_t + w_{gh} h_{t-1} + b_g) \tag{2.7}$$

$$i_t = \sigma\left(w_{ix} x_t + w_{ih} h_{t-1} + b_i\right) \tag{2.8}$$

$$f_t = \sigma\left(w_{fx} x_t + w_{fh} h_{t-1} + b_f\right) \tag{2.9}$$

$$o_t = \sigma\left(w_{ox} x_t + w_{oh} h_{t-1} + b_o\right) \tag{2.10}$$

$$s_t = g_t \otimes i_t + s_{t-1} \otimes f_t \tag{2.11}$$

$$h_t = \phi(s_t) \otimes o_t \tag{2.12}$$

In LSTM-Netzwerken umfassen die beteiligten Parameter mehrere Komponenten: Zunächst gibt es die Gewichtungsmatrizen und Bias-Terme der einzelnen Tore. Für das Eingangstor sind die Gewichtungsmatrizen w_{gx} und w_{gh}, die jeweils auf die aktuelle Eingabe x_t und den verborgenen Zustand des vorherigen Zeitschritts h_{t-1} wirken, sowie der Bias-Term b_g. Für den Eingangsaktualisierungsbereich sind die

Gewichtungsmatrizen w_{ix} und w_{ih}, die ebenfalls auf x_t und h_{t-1} wirken, sowie der Bias-Term b_i. Für das Vergessenstor sind die Gewichtungsmatrizen w_{fx} und w_{fh}, die jeweils auf die aktuelle Eingabe und den verborgenen Zustand des vorherigen Zeitschritts wirken, sowie der Bias-Term b_f. Für das Ausgangstor sind die Gewichtungsmatrizen w_{ox} und w_{oh}, die entsprechenden Bias-Terme sind b_o. Abbildung x zeigt die Struktur einer einzelnen LSTM-Speicherzelle[16].

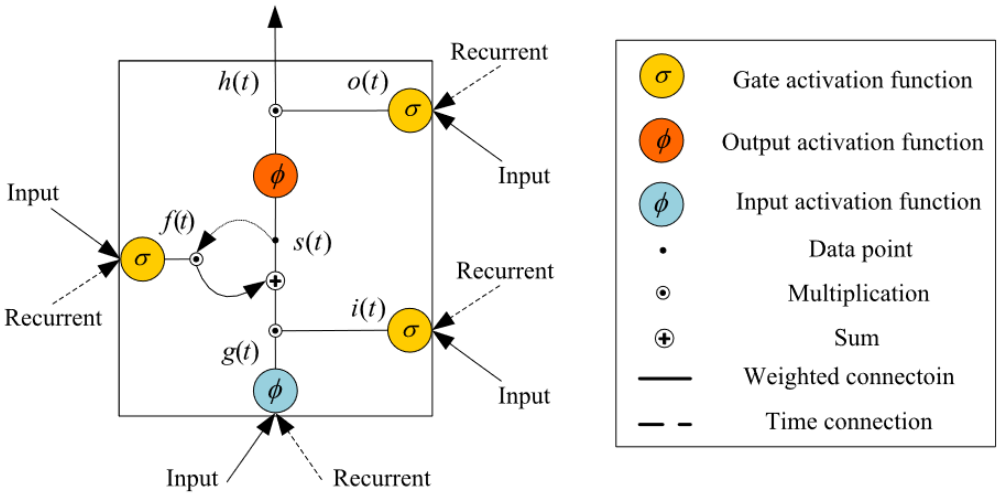


Abbildung 11: Die Architektur einer LSTM- Speicherzelle [16]

2.8.3.3 BILSTM

BILSTM besteht aus zwei unabhängigen LSTM-Netzwerken, von denen eines die Vorwärtszeitreihen-Daten verarbeitet und das andere die Rückwärtszeitreihen-Daten. Seine Ausgabe ist eine Kombination der Ausgabewerte des Vorwärts-LSTM und des Rückwärts-LSTM. Die Ausgabe jedes Zeitschritts besteht aus einer Kombination der versteckten Zustände des Vorwärts-LSTM und des Rückwärts-LSTM:

$$\hat{y}_t = \sigma(h_t^f, h_t^b) \tag{2.13}$$

Dabei ist h_t^f die Ausgabe des Vorwärts-LSTM und h_t^b die Ausgabe des Rückwärts-LSTM. Abbildung 12 zeigt die Architektur des BILSTMs. [17]

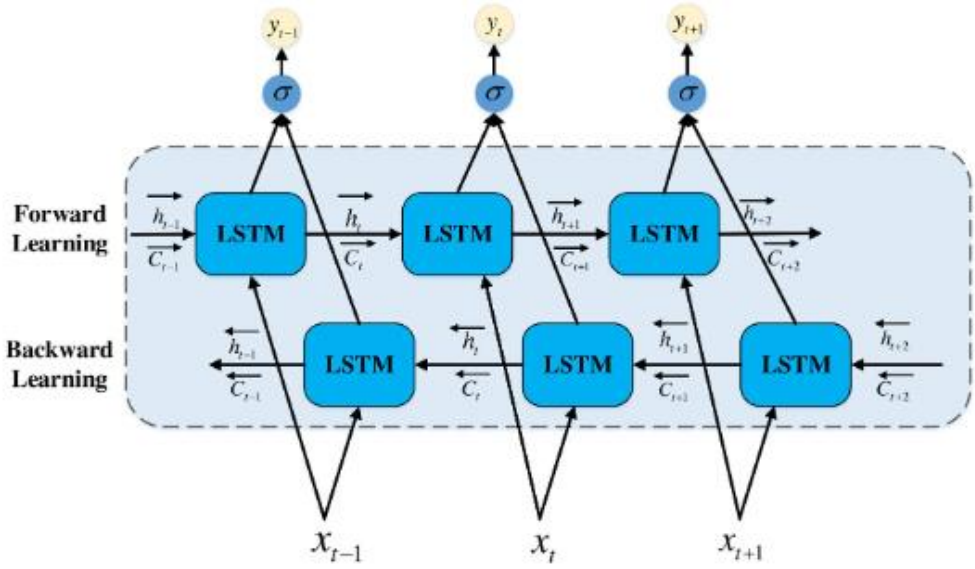


Abbildung 12: Die Architektur des BILSTMs [17]

BILSTM kann gleichzeitig vergangene und zukünftige zeitliche Informationen berücksichtigen und zeigt daher in Aufgaben, die Kontextinformationen erfordern, eine überlegene Leistung [17].

3 Prozess der Datenanalyse

Der allgemeine Datenverarbeitungsablauf in Abbildung 13 wird sowohl bei der traditionellen statistischen Datenanalyse als auch beim maschinellen Lernen eingehalten.

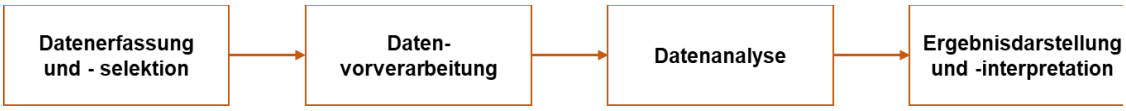


Abbildung 13: Datenverarbeitungsablauf [60]

Der erste Schritt in diesem vier Schritte beinhaltende Abläufe ist die Datenerfassung und -selektion, d.h. das Sammeln bereits vorhandener Daten oder die Erhebung geeigneter Daten. Entsprechend den späteren Analyseanforderungen werden die Daten durch Transformation, Reduktion oder Säuberung leicht analysierbar gestaltet. Der dritte Schritt ist die Datenanalyse, die durch statistische Datenanalyse oder maschinelles Lernen aus den Daten Informationen oder Beispiele extrahiert. Der letzte Schritt der Ergebnispräsentation und -interpretation erklärt mögliche Zusammenhänge durch die Betrachtung der Ergebnisse der visuellen Analysen [60].

3.1 Datenerfassung und –Selektion

Die Datenerfassung ist der erste Schritt der Datenanalyse. Maschinendaten, die im Bereich der Prädiktiven Instandhaltung erfasst werden, werden üblicherweise von maschinennahen Sensoren erfasst und über Schnittstellen an MES oder andere Datenbanken zur Speicherung übertragen. In der Praxis gibt es jedoch häufig Herausforderungen, wie z. B. nicht gängige Datenschnittstellen oder die Unmöglichkeit, vollständige Daten zu erfassen [59]. Neue Kanäle für die Datenerfassung sind daher zu schaffen [3].

Im Bereich der Prädiktiven Instandhaltung sollten unterschiedliche Datensätze für unterschiedliche Bedürfnisse ausgewählt werden. Für die Fehlerdiagnose reicht es beispielsweise aus, Daten in Scheiben auszuwählen, die die normalen Betriebsdaten und die abnormalen Daten der beobachteten Komponente charakterisieren. Für die RUL-Prognosephase werden jedoch die Daten des gesamten Lebenszyklus vom Normalbetrieb bis zum vollständigen Ausfall benötigt.

3.2 Datenvorverarbeitung

Datenvorverarbeitung ist ein entscheidender Schritt im Datenanalyseprozess, dessen Ziel es ist, die Qualität und Konsistenz der Daten zu verbessern. Zu den gängigen Vorverarbeitungsmethoden gehören Datenbereinigung (Korrektur von Fehlern und Ergänzung fehlender Informationen), Datenumwandlung (Anpassung der Darstellungsform der Daten), Datenreduktion (Reduzierung der Datensatzgröße zur Vermeidung von Verzerrungen) sowie Datenintegration (Zusammenführung von Daten aus verschiedenen Quellen zu einem einheitlichen Datensatz). Diese Schritte tragen dazu bei, die Daten im weiteren Analyseprozess zuverlässiger und genauer zu machen. Es wird zudem empfohlen, in diesem Prozess vorab basiertes Fachwissen zu integrieren, um optimale Ergebnisse zu erzielen [3].

Die in dieser Arbeit verwendete Methode zur Datenoptimierung basierend auf Lebesgue-Sampling ist ebenfalls ein Teil der Datenvorverarbeitung. Der Schwerpunkt dieser Arbeit liegt zudem darauf, den Einfluss von Datenvorverarbeitungsmethoden auf die Trainingsleistung von RNN-Modellen zu untersuchen.

3.2.1 Datenbereinigung

Datenbereinigung dient der Verbesserung der Datenqualität durch das Entfernen oder Korrigieren von Fehlern und Inkonsistenzen in den Daten. Sie befasst sich hauptsächlich mit zwei Fehlerkategorien: zufällige Fehler (wie Mess- und Übertragungsfehler, fehlende Werte und Ausreißer) sowie systematische Fehler (wie Sensordrift oder Kalibrierungsfehler).

Gängige Methoden zur Fehlerbehebung umfassen das Entfernen von Rauschen durch Filterung, die Identifizierung und Behandlung von Ausreißern mittels statistischer Methoden (wie der 3-σ-Regel) sowie das Ersetzen oder Löschen fehlender Werte, um die Genauigkeit und Vollständigkeit der Daten zu gewährleisten [3].

3.2.2 Datenreduktion

Im Datenanalyseprozess sind nicht alle Attribute oder Datensätze gleichermaßen wichtig, dennoch verbraucht ihre Verarbeitung Speicherplatz und Rechenressourcen. Um diese unnötigen Aufwendungen zu reduzieren, wird üblicherweise eine Datenreduktion in der Vorverarbeitung durchgeführt. Das Ziel der Datenreduktion besteht darin, durch das Löschen eines Teils der Daten keine signifikanten Auswirkungen auf die

Analyseergebnisse zu haben. Neben der Aggregation durch Datenumwandlung gehören auch Sampling und Dimensionalitätsreduktion zu den gängigen Reduktionstechniken [3].

3.2.2.1 Sliding Window

Sliding Window (SW) [4, 31] ist eine Methode, die üblicherweise verwendet wird, um wichtige Merkmale (z. B. Mittelwert oder Extremwert) aus einem groß angelegten Rohdatensatz (z. B. Strom- oder Vibrationssignal) zu extrahieren und die Datenmenge auf ein verarbeitbares Volumen zu reduzieren. Das Konzept des SW ist dem gleitenden Durchschnitt ähnlich. Im SW können die folgenden Parameter für verschiedene Datensätze angepasst werden:

- Fenstergröße: ein Vektor oder eine Liste, die als Offset relativ zur aktuellen Zeit dient.
- Schrittgröße: Verschiebung um eine festgelegte Anzahl von Datenpunkten anstelle einer Bewegung von Punkt zu Punkt.
- Gewählte Statistik innerhalb des Fensters: Statistiken wie Mittelwert, Maximum, Minimum können innerhalb des Fensters berechnet werden.

Im Allgemeinen gibt es zwei gängige Arten von SW: nicht überlappende Fenster und überlappende Fenster. Im ersten Fall ist die Fenstergröße gleich der Schrittweite, was bedeutet, dass nicht alle Daten wiederholt abgetastet werden. Im zweiten Fall ist die Fenstergröße größer als die Schrittweite und die Daten werden erneut abgetastet. In dieser Arbeit wird das überlappende Fenster verwendet, um die Verzerrung der Daten durch das Abtasten zu reduzieren. [31]

$$\{SW_k\} = \left\{ \frac{1}{w} \sum_{t=1+(k-1)m}^{k\omega+(k-1)m} x_t \right\} \tag{3.1}$$

und nach Extremwerten:

$$\{SW_k\} = Max \text{ or } Min\{x_{1+(k-1)m}, \ldots, x_{kw+(k-1)m}\} \tag{3.2}$$

wobei ω die Fenstergröße, m die Schrittgröße und k den Index der neuen Sequenz $\{SW_k\}$ darstellt und sequence $k = \left[1 + \frac{t-\omega}{m}\right]$ für alle $t \geq \omega$ und $\omega \geq m$. SW kann den groß angelegten Signaldatensatz effektiv in ein kleineres Volumen reduzieren (im Folgenden als aggregierter Datensatz bezeichnet). Der aggregierte Datensatz wird im nächsten Verarbeitungsschritt verwendet [31].

3.2.2.2 Riemann Sampling und Lebesgue Sampling

Das Abtasten und Erfassen von Signalen ist eine grundlegende Aufgabe der modernen Digitaltechnik. Von der grundlegenden Signalverarbeitung in Steuersystemen bis hin zur digitalen Kommunikation ist das Sampling ein Schlüsselfaktor für die effiziente Darstellung von Signalen aus der Natur. In der digitalen Welt besteht die traditionelle Abtastung aus einem einheitlichen Zeitabtastzyklus, der als Riemann-Sampling (RS) bekannt ist. Aufgrund der geringen Komplexität von Analyse, Design und Implementierung werden RS-Methoden häufig in computergestützten Systemen verwendet [21, 43]. Lebesgue Sampling (LS) hingegen ist einer der gängigsten ereignisgesteuerten Sampling-Mechanismen. Beim LS-methode wird der Algorithmus nur ausgeführt, wenn eine Messung von einem Lebesgue-Zustand in einen anderen übergeht oder wenn ein Ereignis eintritt. Das heißt, RS wird entlang der x-Achse integriert, während LS entlang der y-Achse integriert wird. Diese LS-methode reduziert die Anzahl der Datenpunkte, die zur Beschreibung des Signals erforderlich sind [43]. Für das Edge Device ist die Reduzierung der Datenmenge ein wichtiger Faktor. Abbildung 14 zeigt ein Beispiel für den Unterschied zwischen Riemann- und Lebesgue Sampling.

Bei RUL-Prognosen wird ein Gerät oder eine Maschine während des Betriebs aufgrund von Verschleiß oder Alterung allmählich ausfallen. Bei RUL-Prognosen wird ein Gerät oder eine Maschine während des Betriebs aufgrund von Verschleiß oder Alterung allmählich ausfallen. Die physikalischen Größen, die den Zustand ihrer Ausrüstung widerspiegeln, werden sich ebenfalls allmählich verändern. Zu Beginn der normalen Betriebsphase (ohne Berücksichtigung frühzeitiger Ausfälle) zeigt das Signal in der Regel einen gleichmäßigen Zustand mit geringer Veränderung des Signals im Laufe der Zeit. Die Zeit zwischen der Erkennung eines Fehlers und seinem Auftreten kann sehr lang sein. Eine solche kontinuierliche Überwachung kann redundant und ressourcenaufwändig sein. In Fällen, in denen eine RUL-Prognose mit herkömmlichen RS-methode nur schwer möglich ist, können Lebesgue-basierte Abtastverfahren eine neuartige Lösung für das Problem der knappen Rechenressourcen darstellen [43].

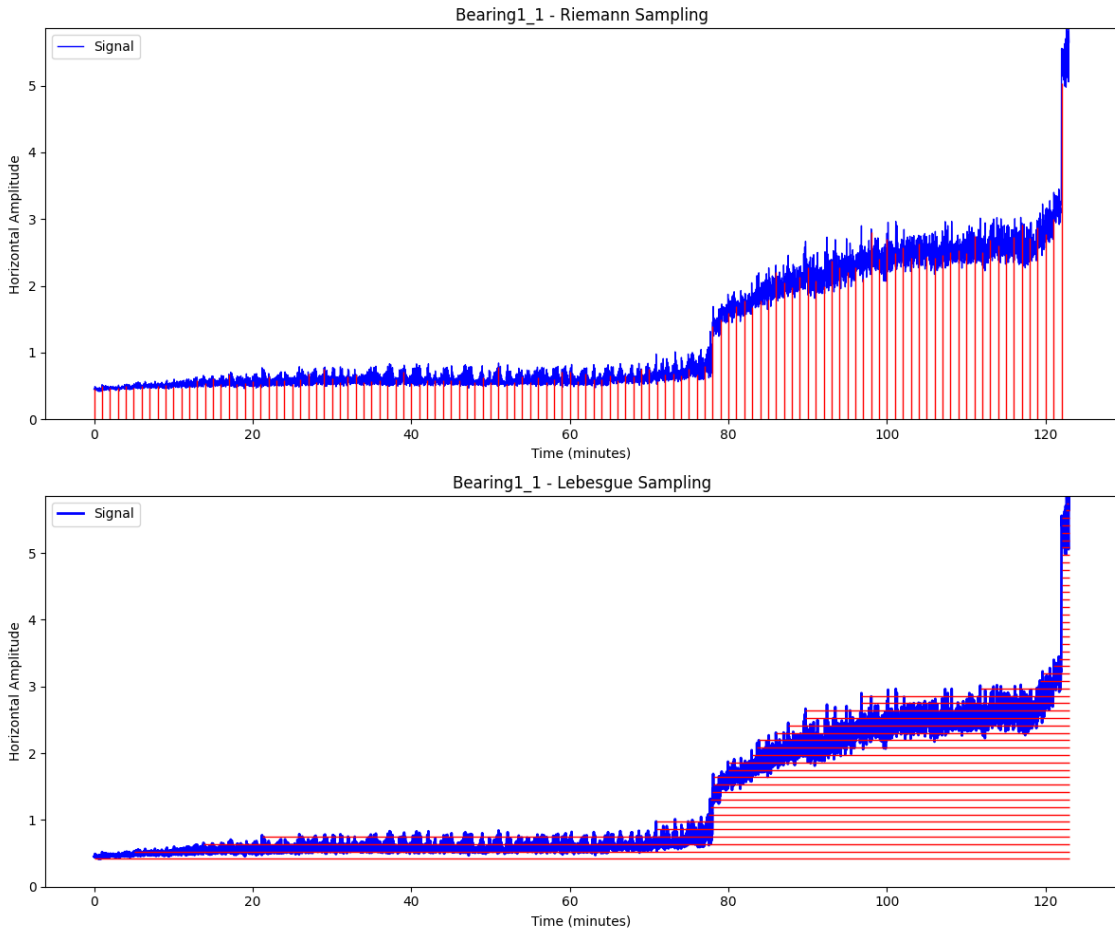


Abbildung 14: Riemann Sampling und Lebesgue Sampling

3.2.3 Datentransformation

Nach der Datenreduzierung müssen die Daten weiter transformiert werden. Der Zweck der Datentransformation besteht darin, die Daten in ein Format umzuwandeln, das für die anschließende Verwendung beim maschinellen Lernen oder beim wissensbasierten Data Mining geeignet ist. Zu den häufig verwendeten Transformationsmethoden gehören [3]:

Normalisierung und Standardisierung: Bei der Normalisierung werden die Werte auf ein bestimmtes Intervall abgebildet, z. B. wird die absolute RUL eines Wälzlagers in einen RUL-normalisierten Parameter von 0 bis 1 umgewandelt. Und die Standardisierung wandelt die Werte in eine Verteilung mit einem Mittelwert von 0 und einer Standardabweichung von 1 um [3].

Abgeleitete Attribute: Abgeleitete Attribute sind neue Attribute, die aus bestehenden Attributen oder Daten erzeugt werden. Dieses Verfahren wird häufig verwendet, wenn das neue Attribut zur Analyse beiträgt oder eine Aggregationsoperation durchgeführt wird, bei der verschiedene Attributwerte zu einem neuen Wert kombiniert werden, wie z. B. Standardabweichung oder Kurtosis [3]. In dieser Arbeit werden wir die Wälzlager aus den Amplitudenrohdaten in Zeit- und Frequenzbereichsmerkmale umwandeln.

- Zeitbereich

Die Methode der kleinsten Quadrate wird verwendet, um statistische Merkmale aus einem Intervallsampledatenbestand zu identifizieren. Gegeben sei s als Stichprobenstandardabweichung und n als Stichprobenumfang. Es gibt sechs statistische Merkmale [31]:

(1) Mittlerer quadratischer Fehler (MSE)

$$\frac{1}{n}\sum_{t=1}^{n}(x_t - \bar{x})^2 / n \tag{3.3}$$

(2) Steigung = Koeffizient β_1 der Zeitvariablen in der Methode der kleinsten Quadrate

(3) Achsenabschnitt =Koeffizient β_0 in der Methode der kleinsten Quadrate

(4) Schiefe

$$\frac{1}{n}\sum_{t=1}^{n}(x_t - \bar{x})^3 / s^3 \tag{3.4}$$

(5) Kurtosis

$$\frac{1}{n}\sum_{t=1}^{n}(x_t - \bar{x})^4 / s^4 \tag{3.5}$$

(6) Maximalwert= $\max(|x_t|)$ im Intervall

Der Zeitindex des Änderungspunkts in der Zeitreihendimension stellt den Wert in der Änderungspunktdimension dar. Im Folgenden werden vier weitere Merkmale aufgeführt:

(7) Standardabweichung (SD)

$$\sqrt{\frac{\sum_{j=1}^{n_c}(c_j-\bar{c})^2}{n_c-1}} \tag{3.6}$$

(8) Erster Veränderungspunkt=Der Zeitpunkt, zu dem der erste Veränderungspunkt auftritt.

(9) Schiefe (Skewness)

$$\frac{1}{n_c}\sum_{j=1}^{n_c}\frac{\left(c_j-\bar{c}\right)^3}{s_c^3} \tag{3.7}$$

(10) Wölbung (Kurtosis)

$$\frac{1}{n_c}\sum_{j=1}^{n_c}\frac{(c_j-\bar{c})^4}{s_c^4} \tag{3.8}$$

- Frequenzbereich

  Im Bereich der Frequenzdatenerfassung wird die schnelle Fourier-Transformation (FFT) eingesetzt. Sie transformiert jeden Abschnitt eines zufälligen Intervalls, der die Amplitude (d.h. die Leistung) im Zeitbereich enthält, in den Frequenzbereich. Die FFT ermittelt das Spektrum (d.h. die von den Datenpunkten angezeigte Leistung für jede Frequenz) sowie die entsprechenden Frequenzen. Für die Merkmalsextraktion werden die Frequenzen der maximalen Amplitude und der zweitstärksten Amplitude als Merkmale verwendet. Die ersten bis vierten Momente sind der Mittelwert, die Varianz, die Schiefe und die Wölbung des Spektrums jedes zufälligen Zeitintervalls.

  (11) Ampl1.f

  (12) Ampl1-freq.f

  (13) Ampl2.f

  (14) Ampl2-freq.f

  (15) Ampl-mean.f

  (16) Ampl-var.f

  (17) Ampl-skewness.f

  (18) Ampl-kurtosis.f

3.3 Datenanalyse

- Explorative Datenanalyse

Im Datenanalyseprozess werden zwar automatisierte Methoden weit verbreitet eingesetzt, jedoch wird vor deren Anwendung üblicherweise eine manuelle oder halbautomatische explorative Datenanalyse durchgeführt, um wertvolle Informationen aus den Daten mittels statistischer und visueller Techniken zu extrahieren. Diese umfasst drei Hauptaufgaben: Beschreibung, Auffinden und Inferenz. Zu den gängigen Werkzeugen gehören der Mittelwert, der Median, Streuungsmaße, Histogramme, Boxplots und Mosaikdiagramme [3].

- Korrelationsanalyse

Um repräsentativere Parameter zu erhalten, verwendet diese Arbeit Korrelationsanalyse und den Variance Inflation Factor (VIF) zur Datenanalyse. Ziel ist es, für die auf der RS-Methode basierende RUL weniger, aber präzisere Daten bereitzustellen, die Rechenlast zu reduzieren und die Prognosegenauigkeit zu verbessern.

Korrelationsanalyse ist eine statistische Methode zur Messung der Stärke und Richtung der Beziehung zwischen zwei oder mehr quantitativen Variablen. Ihr Hauptziel ist es, zu bewerten, ob eine Beziehung zwischen den Variablen besteht und ob diese Beziehung positiv oder negativ ist. Das zentrale Maß der Korrelationsanalyse ist der Korrelationskoeffizient, dessen Wertebereich in der Regel von -1 bis 1 reicht:
Wenn der Korrelationskoeffizient +1 ist, bedeutet dies, dass die beiden Variablen vollständig positiv miteinander verbunden sind, und wenn der Wert -1 ist, bedeutet dies, dass die beiden Variablen vollständig negativ miteinander verbunden sind. Wenn der Wert 0 ist, ist keine lineare Beziehung zwischen den beiden Variablen gegeben [14].

Um repräsentativerer Parameter zu erhalten, wird der VIF als zweiter Schritt eingeführt. Die in der Abbildung dargestellte Formel zeigt den VIF, der zur Erkennung von Multikollinearitätsproblemen zwischen Merkmalen in Regressionsmodellen verwendet wird. Die Berechnungsformel lautet:

$$\mathrm{VIF} = \frac{1}{1 - R_i^2} \tag{3.9}$$

Dabei ist R_i^2 das Bestimmtheitsmaß, das den Einfluss des i-ten Prädiktors als abhängige Variable in einem Regressionsmodell basierend auf den anderen Prädiktoren misst. Wenn der VIF-Wert 10 überschreitet, deutet dies auf ein schwerwiegendes

Multikollinearitätsproblem hin, und es sollte erwogen werden, das Merkmal zu entfernen, das den Informationsverlust am geringsten verursacht.

- Bootstrap

Die Bootstrap-Methode ist eine zentrale Technik in der nichtparametrischen Statistik zur Intervallschätzung statistischer Parameter. Ihr Kernkonzept besteht darin, durch wiederholtes Ziehen gleich großer Stichproben mit Zurücklegen aus dem Originaldatensatz die Verteilung der Originaldaten zu approximieren. Durch Wiederholen des Stichprobenprozesses N-mal werden N Bootstrap-Stichproben erzeugt. Diese Stichproben werden verwendet, um N Regressionsmodelle zu trainieren; durch Mittelung der Ausgaben jedes Modells wird der Prognosefehler RUL minimiert. Um Formel (6) effizient zu berechnen, verwendet diese Studie die nichtparametrische Bootstrap-Methode [17, 20].

3.4 Ergebnisdarstellung und -interpretation

Zur Bewertung der experimentellen Ergebnisse dieser Arbeit werden mehrere Evaluierungsparameter herangezogen. Das Fehlermodul für die Fehlerdiagnose verwendet den mittleren quadratischen Fehler (MSE), während die restlichen drei Parameter zur Bewertung der Prognose der RUL eingesetzt werden. Im Folgenden werden ihre Prinzipien kurz erläutert.

- Mittlerer quadratischer Fehler (MSE)

Der mittlere quadratische Fehler (MSE) ist eine in der Regressionsanalyse weit verbreitete Verlustfunktion, die den Durchschnitt der quadrierten Fehler zwischen den prognostizierten Werten und den tatsächlichen Werten berechnet. Die Berechnungsformel lautet wie folgt [10]:

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}\left(X_i - Y_i\right)^2 \tag{3.10}$$

Dabei ist X_i der prognostizierte Wert des $\mathrm{i\text{-}ten}$ Samples, Y_i der tatsächliche Wert des $\mathrm{i\text{-}ten}$ Samples und mmm die Gesamtzahl der Samples [10].

Der Wert des MSE ist stets nicht negativ, und je näher er an 0 liegt, desto genauer ist die Vorhersage des Modells. Da die Fehler quadriert werden, werden größere Fehler verstärkt, was bedeutet, dass der MSE sehr empfindlich gegenüber Ausreißern ist. Wenn das Modell größere Fehler aufweist, erhöht sich der MSE-Wert erheblich [10].

Der MSE ist besonders geeignet für die Fehlerdiagnose, da er die durchschnittliche Größe der Abweichungen zwischen vorhergesagten und tatsächlichen Werten quantifiziert, was entscheidend für die Bewertung der Modellgenauigkeit bei der Fehlerdiagnose ist.

- Wurzel des mittleren quadratischen Fehlers (RMSE)

Der RMSE ist die quadratische Wurzel des MSE, dessen Formel lautet:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2} \tag{3.11}$$

Der RMSE behält die gleichen Vorteile wie der MSE bei, während seine Werte mit den Einheiten der tatsächlichen Daten übereinstimmen, was eine bessere Interpretierbarkeit ermöglicht. Wie beim MSE gilt auch beim RMSE: Je näher er an 0 liegt, desto besser ist die Prognose des Modells. Der RMSE ist ebenfalls empfindlich gegenüber Ausreißern, da er die quadratische Operation des MSE übernimmt. Der RMSE ist nützlich für die RUL-Prognose, da er die Fehler in den gleichen Einheiten wie die ursprünglichen Daten ausdrückt und somit die Genauigkeit der Prognose der RUL besser interpretierbar macht [10].

- Bestimmtheitsmaß (R^2)

Das Bestimmtheitsmaß R^2, auch als Determinationskoeffizient bekannt, dient zur Messung der Fähigkeit eines Regressionsmodells, die Varianz in den Daten zu erklären. Die Formel lautet:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(X_i - Y_i)^2}{\sum_{i=1}^{m}(Y_i - \bar{Y})^2} \tag{3.12}$$

Dabei ist $\bar{Y}$ der Mittelwert der tatsächlichen Werte.

Der Wert von R^2 liegt zwischen 0 und 1. Je näher er an 1 liegt, desto stärker ist die Fähigkeit des Modells, die Daten zu erklären. Das Bestimmtheitsmaß R^2 ist geeignet für die RUL-Prognose, da es die Fähigkeit des Modells misst, die Varianz in den Daten zu erklären, was für genaue Prognose der RUL wesentlich ist [10].

- NASA-Scores

Der NASA-Score wurde ursprünglich zur Behandlung von Szenarien des Motorenverschleißes eingesetzt, bei denen Frühprognosen gegenüber Spätprognosen bevorzugt werden. Daher verwendet dieser Bewertungsalgorithmus unterschiedliche Strafmechanismen für die positiven und negativen Richtungen des Prognosefehlers. Prognosen, die nach dem tatsächlichen Ausfallzeitpunkt liegen, werden stärker bestraft, während frühzeitige Prognosen relativ mildere Strafen erhalten. Das Funktionsbild von NASA-Scores ist in Abbildung 15 dargestellt. Die spezifische Formel lautet wie folgt:

$$s = \begin{cases} \sum_{i=1}^{n} e^{-\left(\frac{d}{a_1}\right)} - 1 \text{ für } d < 0 \\ \sum_{i=1}^{n} e^{\left(\frac{d}{a_2}\right)} - 1 \text{ für } d \geq 0, \end{cases} \tag{3.13}$$

wobei
s ist die berechnete Punktzahl,
n ist die Anzahl der zu prüfenden Gerät,
d = gRUL - tRUL (geschätzte RUL - tatsächliche RUL),
a_1 = 10,
und a_2 = 13.

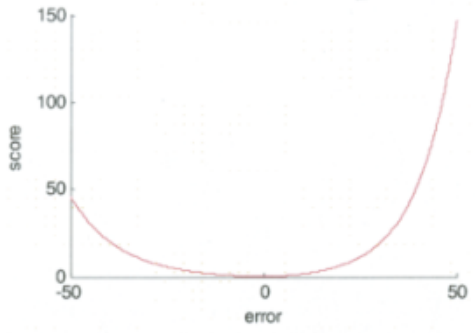


Abbildung 15: Funktionsbild von NASA-Scores[44]

Diese asymmetrische Bewertungsfunktion steuert das Ausmaß der Strafen für Fehler durch die Parameter a_1 und a_2. Mit zunehmendem Fehler wächst die Strafe exponentiell. Diese Bewertungsfunktion erfasst präzise die Präferenz für frühzeitige Prognosen und kann entsprechend angepasst werden, um diese Präferenz zu quantifizieren.

Darüber hinaus wurde die Idee vorgeschlagen, dieses Maß bei der Bewertung der Ergebnisse weiter zu optimieren. Insbesondere, da frühe Prognosen schwieriger sind und die Genauigkeit bei Geräten, die sich dem Lebensende nähern, wichtiger ist, könnte erwogen werden, Situationen mit kürzerer tatsächlicher verbleibender Lebensdauer höhere Gewichte zuzuweisen, um die Bewertungspraxis weiter zu präzisieren. Als zentrale Komponenten rotierender Maschinen haben das Versagen von Lagern und Motoren erhebliche Auswirkungen. Die RUL-Prognose von Lagern profitiert ebenfalls von der Tatsache, dass frühzeitige Prognosen gegenüber späten Prognosen bevorzugt werden. Daher ist der NASA-Scores für die RUL von Lagern äußerst geeignet [44].

4 Modul für Fehlerdiagnose und RUL-Prognose

Die Arbeit gliedert sich in zwei Hauptmodule: das Fehlerdiagnosemodul und das RUL-Prognosemodul. Das Fehlerdiagnosemodul dient dazu, den frühesten Ausfallzeitpunkt eines Lagers während seines gesamten Lebenszyklus zu bestimmen. Dies bildet die Grundlage für die anschließende RUL-Prognose. Wenn ungefilterte Schwingungsdaten des gesamten Lebenszyklus als Eingabe für das RUL-Prognosemodul verwendet werden, sind die Daten des normalen Betriebsabschnitts eine oder mehrere Konstanten. Diese nahezu konstanten Werte verschwenden einerseits Rechenressourcen und haben andererseits negative Auswirkungen auf das Training des neuronalen Netzwerks. Im RUL-Prognosemodul wird die verbleibende Lebensdauer des Lagers ermittelt. Vorverarbeitete Abbaudaten werden in Algorithmen des Deep Learnings eingespeist, um durch das Erlernen der Beziehung zwischen Zeit-Frequenz-Daten und Gesundheitszustand die entsprechende verbleibende Lebensdauer zu bestimmen.

4.1 Dataset

Das Hauptziel dieser Arbeit ist die Prognose der Restlebensdauer von Wälzlagern. Dazu ist es notwendig, einen Datensatz für den gesamten Zyklus vom Normalbetrieb bis zum Ausfall auszuwählen. Derzeit haben einige Forschungsinstitute Ermüdungsversuche und beschleunigte Lebensdauertests an Wälzlagern durchgeführt und die Versuchsdaten veröffentlicht. Die repräsentativsten sind der Wälzlager-Datensatz des IMS Centre der Universität Cincinnati, USA, und der Wälzlager-Datensatz des FEMTO-ST-Instituts, Frankreich. Tabelle 2 zeigt einen detaillierten Vergleich der Schwingungssignale in den drei Datensätzen. Wie aus Tabelle 2 ersichtlich ist, wird der IMS-Wälzlagerdatensatz nur unter einer einzigen Betriebsbedingung mit konstanten Werten sowohl für die Drehzahl als auch für die Radiallast gewonnen und weist eine geringe Stichprobengröße auf, während der FEMTO-ST-Wälzlagerdatensatz keine detaillierten Informationen über den Fehlerort enthält und eine geringe Frequenzauflösung mit einer Abtastzeit von nur 0,1 s aufweist. Diese Unzulänglichkeiten schränken den Anwendungsbereich dieser Datensätze ein. Der XJTU-SY-Datensatz bietet eine Vielzahl von Betriebszuständen auf der Basis von Schwingungsdaten über den gesamten Lebenszyklus, listet eindeutig den Fehlerort jedes ausgefallenen Lagers auf und hat eine hohe Frequenzauflösung, so dass der XJTU-SY-Datensatz für diese Arbeit ausgewählt wurde [50].

Tabelle 2: IMS, FEMTO-ST, XJTU-SY Datensatzvergleich

Beschaffenheit der Datensets	IMS	FEMTO-ST	XJTU-SY
Anzahl der Lager	4	17	15
Anzahl der Prüfbedingungen	1	3	3
Abtastfrequenz / kHz	20	25.6	25.6
Abtastintervall	10 min & 5 min	10 s	1 min
Länge der einzelnen Proben	1.024	0.1	1.28
Ob Ausfallinformationen markiert werden	Ja	Nein	Ja

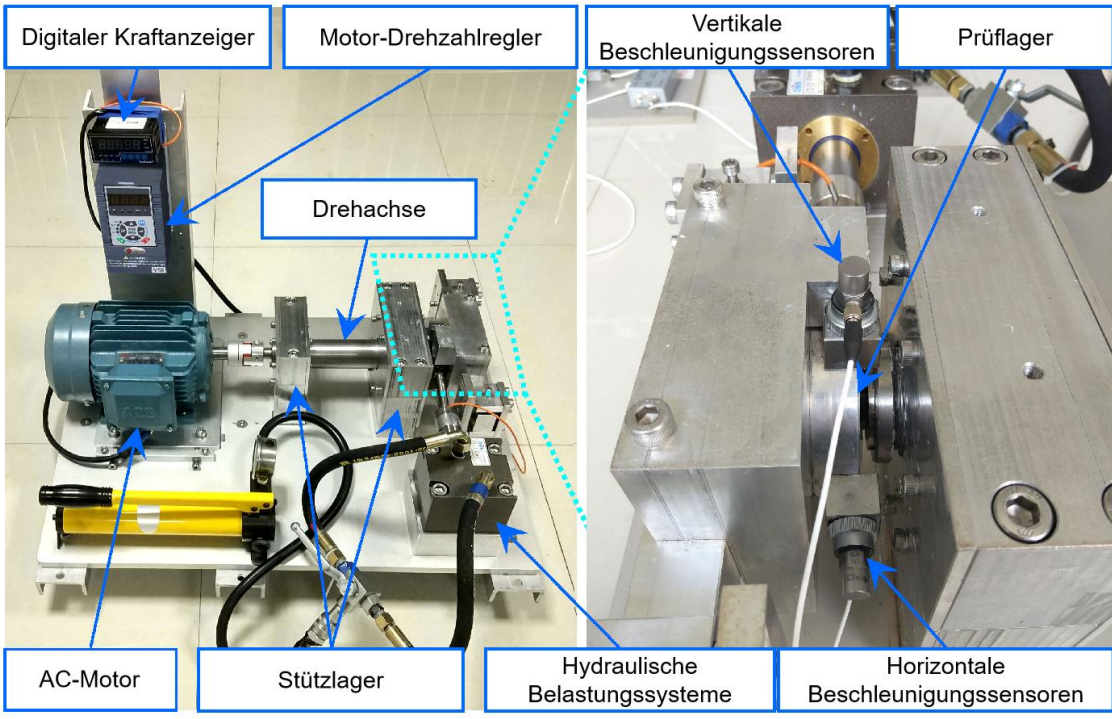


Abbildung 16: XJTU-Prüfstand für Wälzlager[1, 50]

Aus diesem Datenset wurden 15 Datensätze gesammelt, in denen jeweils fünf Wälzlager unter drei verschiedenen Betriebsbedingungen getestet wurden. Die Betriebsbedingungen umfassen (1)2100 U/min (35 Hz) und 12 kN; (2) 2250 U/min (37,5 Hz) und 11 kN; und (3) 2400 U/min (40 Hz) und 10 kN [50]. Die detaillierte Lebensdauer und die Ausfallstellen für jedes Wälzlager sind in Tabelle 3 aufgeführt.

Zur Erfassung der Schwingungssignale des zu prüfenden Wälzlagers wurden zwei Beschleunigungssensoren vom Typ PCB 352C33 in einem Winkel von 90° auf der horizontalen und der vertikalen Achse am Gehäuse des zu prüfenden Wälzlagers angebracht, wie in der Abbildung 16 dargestellt. Die Abtastrate wurde auf 25,6 kHz eingestellt. Pro Abtastung wurden 32768 Datenpunkte (d. h. 1,28 Sekunden) aufgezeichnet, wobei die Abtastzeit 1 Minute betrug. [50]

Tabelle 3: Detaillierte Informationen zu den XJTU-SY Lagerdatensätzen[20]

Betriebsbedingungen	Lagerdatensätze	Anzahl der Dateien	Lagerlebensdauer	Ausfallstellen
Bedingung 1 (35 Hz/12 kN)	Bearing1_1	123	2 h 3 min	Außenring
	Bearing1_2	161	2 h 41 min	Außenring
	Bearing1_3	158	2 h 38 min	Außenring
	Bearing1_4	122	2 h 2 min	Käfig, Innenring
	Bearing1_5	52	52 min	Innen- und Außenring
Bedingung 2 (37.5 Hz/11 kN)	Bearing2_1	491	8 h 11 min	Innenring
	Bearing2_2	161	2 h 41 min	Außenring
	Bearing2_3	533	8 h 53 min	Käfig
	Bearing2_4	42	42 min	Außenring
	Bearing2_5	339	5 h 39 min	Außenring
Bedingung 3 (40 Hz/10 kN)	Bearing3_1	2538	42 h 18 min	Außenring
	Bearing3_2	2496	41 h 36 min	Innenring, Kugel, Käfig, und Außenring
	Bearing3_3	371	6 h 11 min	Innenring
	Bearing3_4	1515	25 h 15 min	Innenring
	Bearing3_5	114	1 h 54 min	Außenring

4.2 Fehlerdiagnosemodul

Zunächst wurde das XJTU-SY-Lagerdatensatz in das Fehlerdiagnosemodul (im Folgenden FD-Modul genannt) importiert. Dieser Datensatz enthält Schwingungssignaldaten von Lagern unter verschiedenen Betriebsbedingungen, einschließlich Amplitudeninformationen in horizontaler und vertikaler Richtung. Nach dem Import der Daten erfolgte die Datenbereinigung. Dies umfasste die Entfernung ungültiger Daten, die Bearbeitung fehlender Werte und die Behandlung von Anomaliedaten, um die Vollständigkeit und Genauigkeit der Daten sicherzustellen.

Dieser Schritt ist entscheidend, da die Rohdaten möglicherweise Rauschen oder ungültige Informationen enthalten, die die nachfolgenden Analyseergebnisse beeinflussen könnten. Nach der Datenbereinigung fasst das FD-Modul alle gültigen Daten zusammen und standardisiert sie. Die Daten der verschiedenen Lagergruppen werden separat verarbeitet und die Amplitudensignale in horizontaler und vertikaler

Richtung werden entlang der Zeitachse in ein Standardformat umgewandelt, um die nachfolgenden Analyseschritte zu erleichtern. Während der normalen Betriebsphase der Lager verwendet das FD-Modul ein Zeitfenster von 500, um die Schwingungssignale jedes ausgewählten Lagers über die gesamte Phase hinweg zu analysieren und die rollende Standardabweichung zu berechnen. Gleichzeitig wird die durchschnittliche rollende Standardabweichung unter verschiedenen Betriebsbedingungen aufgezeichnet. Das FD-Modul setzt adaptive Schwellenwerte für die Anomalie Erkennung. Da selbst Lager derselben Charge unter denselben Betriebsbedingungen geringfügige Unterschiede in ihrem Schwingungsverhalten aufweisen. Basierend auf der rollenden Standardabweichung der Lager und dem Durchschnitt der rollenden Standardabweichungen der Lager derselben Gruppe unter denselben Bedingungen wurde ein harmonisierter Schwellenwert vorgeschlagen.

$$T_{harm} = \left(\sigma_{rolling}^{individuell} + \left(\sigma_{rolling}^{Gruppe} - \sigma_{rolling}^{individuell} \right) \cdot k \right) \cdot F \tag{4.1}$$

T_{harm} steht für den harmonisierten Schwellenwert. $\sigma_{rolling}^{individuell}$ bezeichnet die rollende Standardabweichung des individuellen Lagers. $\sigma_{rolling}^{Gruppe}$ bezeichnet den Durchschnitt der rollenden Standardabweichung der Lager derselben Gruppe. k ist der Harmonisierungskoeffizient. F ist der Faktor.

Wenn die Amplitude eines Lagers signifikant kleiner ist als die der anderen Lager, führt die alleinige Verwendung des Faktors der Standardabweichung zu einer zu späten Fehlererkennung. Wenn die Amplitude des Lagers hingegen groß ist, kann dies zu einer verfrühten Fehlererkennung führen. Die Wahl eines geeigneten Harmonisierungskoeffizienten erhöht die Genauigkeit des FD-Moduls. Sobald der harmonisierte Schwellenwert festgelegt ist, erkennt das Modul Anomaliepunkte, die den Schwellenwert überschreiten. Nachdem Anomaliepunkte identifiziert wurden, werden zwei Methoden zur Bestimmung der Ausfallzeit vorgeschlagen.

- Dreimal aufeinanderfolgende Anomalien

Wenn das FD-Modul mindestens dreimal aufeinanderfolgende Anomalien erkennt, wird die Ausfallzeit ausgegeben. Dieser Schritt reduziert Fehlalarme und bestätigt, dass die Signalfluktuation tatsächlich den abnormalen Zustand des Geräts widerspiegelt. In der Anfangsphase des Betriebs kann es jedoch zu Anomalie-Fehlalarmen kommen, die auf eine frühe Einlaufphase zurückzuführen sind. Daher wird eine zweite Definition der Ausfallzeit vorgeschlagen.

- Linkes 1%-Perzentil der Lognormalverteilung

Alle erkannten Anomaliepunkte werden in eine Lognormalverteilung eingepasst. Durch diese Verteilung kann die Ausfallzeit des Lagers vorhergesagt und das 1%-Perzentil als Indikator für die Früherkennung eines Ausfalls berechnet werden. Die erste Ausfallzeit wird durch eine der beiden Methoden ermittelt und ausgegeben.

Zur Bewertung der Genauigkeit des Prognosemodells berechnet das Modul den mittleren quadratischen Fehler (MSE), indem die vom FD-Modul ausgegebene Ausfallzeit mit der tatsächlichen Ausfallzeit verglichen wird. Anschließend findet das Modul die Parametergruppe mit dem kleinsten MSE und wählt sie als optimale Parametergruppe aus. Mit den optimalen Parametern kann das Modul die genauesten Ergebnisse für die Prognose der Ausfallzeit liefern. Schließlich gibt das Modul mit den optimalen Parametern die erste Ausfallzeit des Geräts aus, was die Grundlage für die Datensegmentierung im nachfolgenden RUL-Prognosemodell bildet. Abbildung 17 veranschaulicht das Flussdiagramm des FD-Moduls.

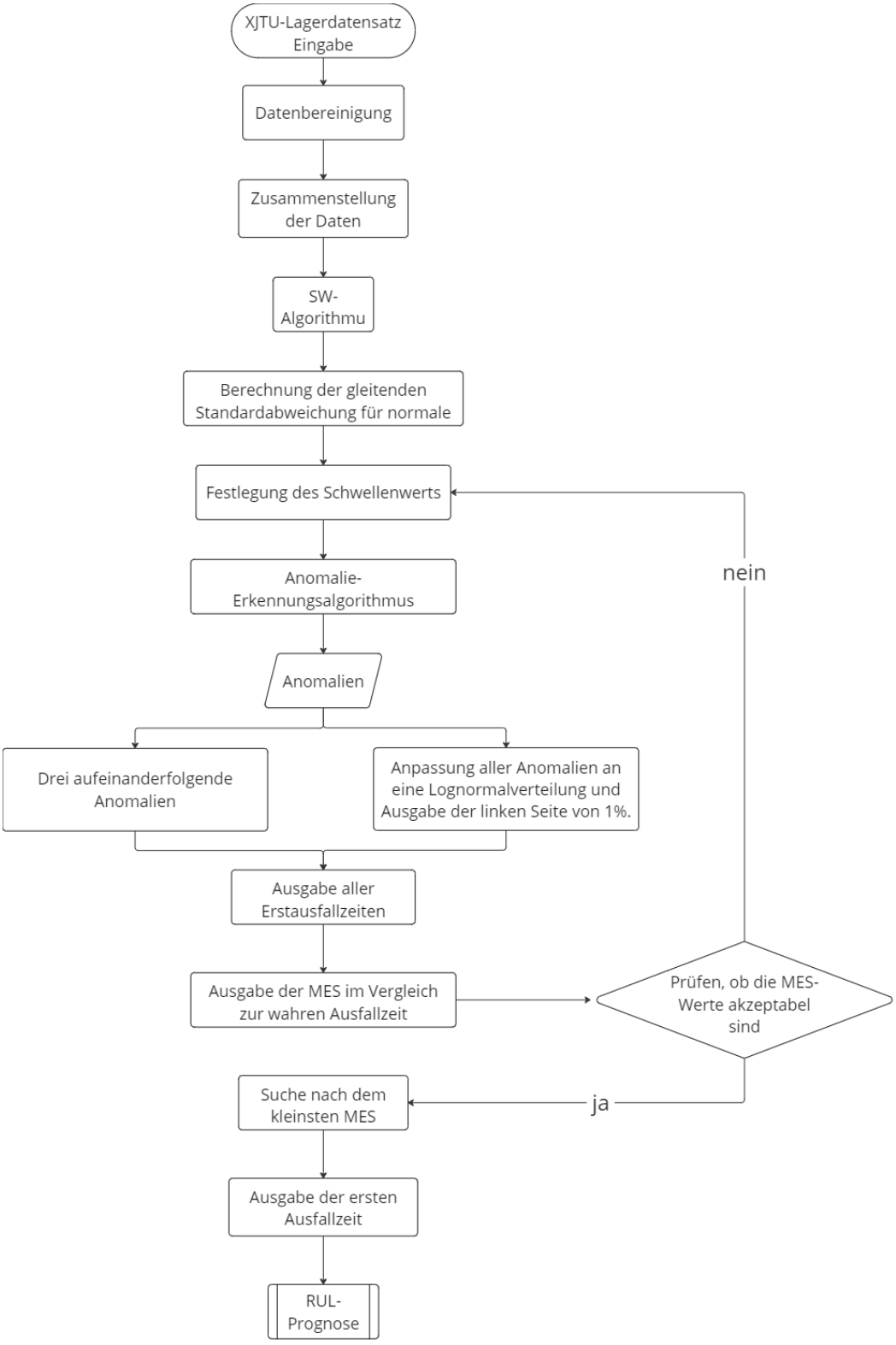


Abbildung 17: Flussdiagramm des Fehlerdiagnosemodul

4.3 RUL-Prognosemodul

4.3.1 Hintergrund der LS-basierten RUL-Prognose für Batterien

Bei der vorläufigen Literaturrecherche fiel auf, dass die Forschung im Bereich der RUL-Prognose von Batterien ebenfalls einen erheblichen Anteil einnimmt. Batterien sind als zentrale Komponenten in verschiedenen Elektrofahrzeugen, Elektrowerkzeugen und elektronischen Konsumgütern von großer praktischer Bedeutung, weshalb die RUL-Prognose für sie äußerst relevant ist. Zhang et al. [37, 52, 54] entwickelten eine auf Lebesgue-Sampling basierende Prognosemethode, die durch die Reduzierung unnötiger Berechnungen die Prognosegenauigkeit und Recheneffizienz des Batteriezustands (State of Health, SOH) verbessert. Im Gegensatz zu herkömmlichen RS-Methoden löst das LS-Methode Berechnungen nur bei signifikanten Änderungen des Batteriezustands aus. Dies ermöglicht es dem System, die Genauigkeit beizubehalten und gleichzeitig die Rechenlast erheblich zu reduzieren, was besonders für Szenarien mit langsamen Veränderungen des Batteriezustands geeignet ist. Dennoch sind Lager als wesentliche Komponenten rotierender Maschinen auch in industriellen Anwendungen von großer Bedeutung. Darüber hinaus ähneln die Veränderungen des Gesundheitszustands von Lagern während ihres Degradationsprozesses stark dem SOH-Verfall von Batterien. Beide durchlaufen nach langen Phasen normaler Betriebszustände eine Phase schneller Degradation, gefolgt von einem vollständigen Versagen. Daher ist die auf Lebesgue-Sampling basierende Prognosemethode theoretisch auch für die Überwachung des Lagerzustands und die Fehlerdiagnose geeignet.

Frühere Studien basierten teilweise auf mathematischen und statistischen Methoden. In einem Lebesguezeit-Raum-Modell wurde die Betriebszeit der Batterie in eine Lebesgue-Zeitverteilung und der Gesundheitszustand der Batterie in eine Lebesgue-Zustandsverteilung umgewandelt [52, 54]. Anschließend wurde die Beziehung zwischen diesen beiden Verteilungen angepasst, um die RUL zu prognostizieren, wie in Abbildung 18 dargestellt.

Im Rahmen der auf Riemann-Sampling basierenden Fehlerdiagnose und -prognose (RS-FDP) wird der RUL-Prognosealgorithmus zu einer Reihe fester Zeitpunkte *{t1, t2,*

t3, ..., tk} ausgeführt. Diese Zeitpunkte werden durch das Sampling des Systems oder die Merkmalsextraktionsrate bestimmt. Diese Strategie berücksichtigt nicht die Geschwindigkeit der Fehlerdegradation; unabhängig davon, wie schnell sich der Fehlerzustand ändert, werden die Algorithmen in festen Zeitintervallen ausgeführt, was hohe Anforderungen an Echtzeitberechnungen und -speicherung stellt. Im RS-FDP-Framework verwenden Diagnose und Prognose dasselbe Modell. Die zustandsbasierte Diagnose im RS-Framework bewertet den aktuellen Fehlerzustand anhand des aktuellen Zustands und des zuvor geschätzten Zustands mithilfe des Modells. Die zustandsbasierte Prognose im RS-Framework schätzt den zukünftigen Fehlerzustand, um die benötigte Zeit bis zum Erreichen der Ausfallschwelle (TTF, Time to Failure) zu prognostizieren [37].

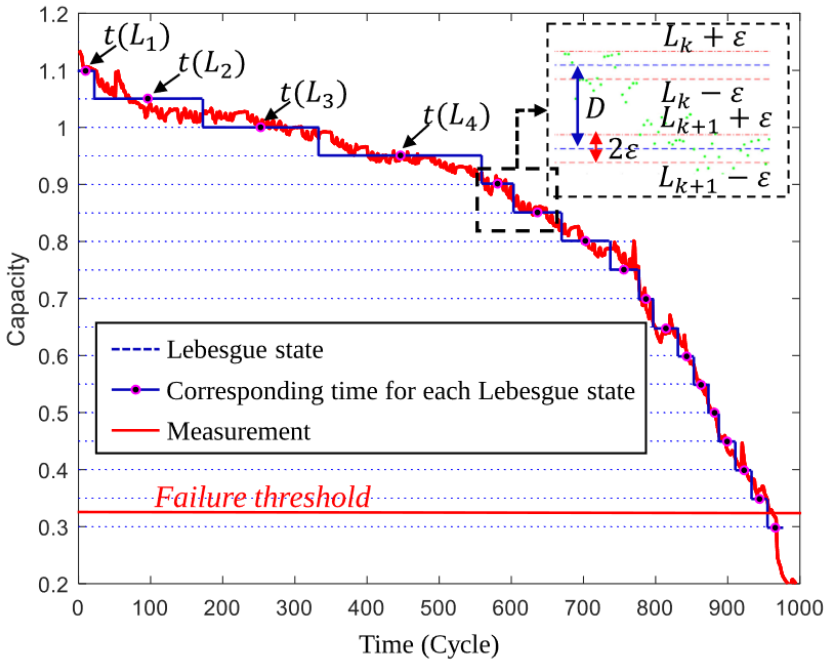
Im Rahmen von LS-FDP wird der Fehlerraum in mehrere vordefinierten Lebesgue-Zustände {F_1, F_2, F_3, …, F_k} unterteilt, die feste oder variable Lebesgue-Intervalle Δ aufweisen können. Wie in Abschnitt 3.2.2.2 beschrieben, ist LS-FDP ein ereignisbasierter Algorithmus, bei dem ein Ereignis als Änderung des Lebesgue-Zustands definiert ist. Der Diagnosealgorithmus wird bei Auftreten eines Ereignisses ausgeführt, anstatt wie herkömmliche Methoden zu festen Zeitpunkten. Die zustandsbasierte Diagnose in LS schätzt den Fehlerzustand und bewertet die Verteilung des Fehlerzustands basierend auf dem Ereigniszeitpunkt. Die zustandsbasierte Prognose in LS verwendet das Lebesgue-Zeitmodell (LZM), um direkt die Betriebszeit zu prognose, die erforderlich ist, damit der Fehlerzustand einen festgelegten Lebesgue-Zustand erreicht. Die TTF wird basierend auf dem Lebesgue-Zustand in Kombination mit dem Fehlergrenzwert geschätzt [37].

LS-FDP verwendet für Diagnose und Prognose unterschiedliche Modelle. Das Fehlerdiagnosemodell wurde in dieser Arbeit bereits auf RS-Basis entwickelt und wird hier nicht weiter erörtert. Hier wird hauptsächlich das LZM wie folgt vorgestellt:

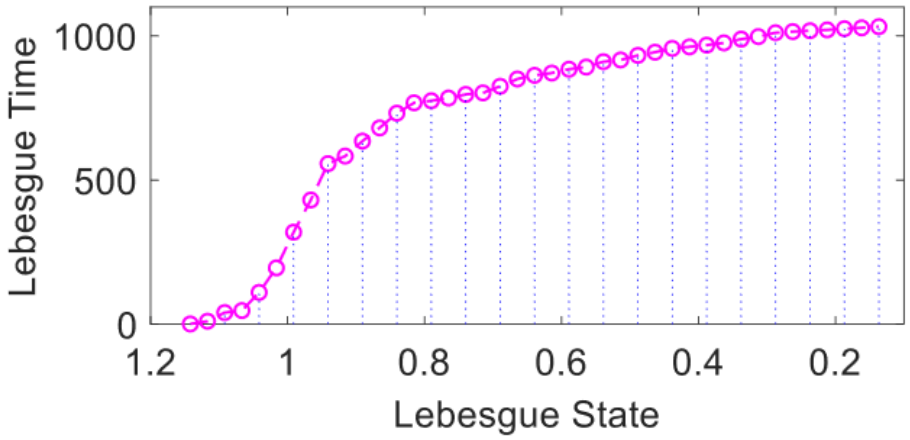
Lebesgue-Zeitmodell: Im Gegensatz zur zustandsbasierten Prognose in RS verwendet die zustandsbasierte Prognose in LS das LZM, um direkt die Betriebszeit zu prognose, die erforderlich ist, um einen bestimmten Lebesgue-Zustand zu erreichen. Die Form des LTM ist wie folgt:

$$t(L_{k+1}) = g(L_k, t(L_k), D) + \omega_t(L_k) \tag{4.2}$$

Dabei ist g die zeitliche Übergangsfunktion, die die Entwicklung des Lebesgue-Zustands über die Zeit beschreibt, L_k und L_{k+1} sind zwei aufeinanderfolgende Lebesgue-Zustände, D ist die Länge des Lebesgue-Intervalls und ωt ist das zeitliche Prozessrauschen [37].



(a)



(b)

Abbildung 18: Lebesguezeit-Raum-Modell [37] (a) Die entsprechende Zeitverteilung für jeden Lebesgue-Status (b) Lebesgue-Zeit-Übergangskurve.

Das LZM spielt eine wichtige Rolle in der zustandsbasierten Prognose von LS. Aufgrund von Kapazitätsmessrauschen und den Degradationseigenschaften der Batterie ist eine genaue zeitliche Entwicklung des Lebesgue-Zustands nicht möglich. Um dieses Modell

aus den Messdaten zu erstellen, wurde an den Lebesgue-Zuständen ein beidseitiger Prüfer definiert, der auf einem Nachbarschaftsbereich des Lebesgue-Zustands basiert, um die Zeitmessung zu bestimmen. Für den Lebesgue-Zustand L_k wird der Bereich des beidseitigen Prüfers durch den Parameter e (e ist viel kleiner als $D/2$) definiert und umfasst den Bereich $[L_k - e, L_k + e]$. Messwerte, die in diesen Bereich fallen, werden als Stichproben des Lebesgue-Zustands L_k identifiziert und aufgezeichnet. Der Mittelwert und die Varianz der aufgezeichneten Messwerte dienen als Messwerte für die zeitliche Verteilung τ_k des entsprechenden Lebesgue-Zustands. Der Parameter e sollte basierend auf dem Messrauschen festgelegt werden, um eine vernünftige und genaue Schätzung des LTM zu gewährleisten. Basierend auf dieser Strategie kann eine zeitliche Übergangskurve erstellt werden, die das LZM mittels RNNs modelliert [37].

4.3.2 Modul für die LS-basierte RUL-Prognose von Wälzlagern

Dateninput: Zunächst wird die erste Ausfallzeit (FPT) des zuvor entwickelten FD-Ausgabemodells verwendet, um den XJTU-Lagerdatensatz zu segmentieren. Die Rohdaten werden in zwei Teile aufgeteilt: einen Teil bestehend aus vollständigen Zyklen und einen anderen Teil basierend auf den Daten nach dem ersten Fehlerpunkt. Diese Segmente dienen als grundlegende Eingaben für das gesamte RUL-Prognosemodell.

Datentransformation und Merkmalsberechnung: Die beiden Datensegmente werden aus den ursprünglichen Vibrationsdaten in zeit- und frequenzdomänenspezifische Merkmale umgewandelt. Basierend auf diesen Merkmalen werden weiter standardisierte Labels erstellt, die mit der RUL in Verbindung stehen. Zur Vereinfachung der Berechnungen wird bei der RUL-Datenkennzeichnung der Informationen nach dem FPT angenommen, dass der anfängliche Gesundheitszustand 1 beträgt, also im normalen Betriebszustand bleibt der Gesundheitszustand konstant.

Sampling und Normalisierung: Nach der Datenkennzeichnung werden die Daten mittels RS und LS-Method weiterverarbeitet. Beim RS werden einerseits mehrere Parameter aus der Zeit- und Frequenzdomäne als Eingaben für das Modell verwendet. Andererseits wird die zeitliche Standardabweichung in vertikaler Vibrationsrichtung im Vergleich zu LS analysiert. Im Rahmen des Lebesgue-Samplings, ähnlich dem Prozess bei Batterien, wird die Standardabweichung als Eingabe für das Modell genutzt. Mithilfe des Lebesgue-Raumzeitmodells werden Lebesgue-Zustände abgeleitet und die normalisierten RUL-Werte in die Lebesgue-Zeitdomäne übertragen.

Tabelle 4: Kombination unterschiedlicher Datenvorverarbeitung für RUL-Prognosemodelle

Kombination	RNNs Modelle	Merkmale Typ	Sampling-Methode	Datenart
1	RNN	SD	RS	Ganze Daten
2	LSTM	SD	RS	Ganze Daten
3	BiLSTM	SD	RS	Ganze Daten
4	RNN	Multi	RS	Ganze Daten
5	LSTM	Multi	RS	Ganze Daten
6	BiLSTM	Multi	RS	Ganze Daten
7	RNN	SD	LS	Ganze Daten
8	LSTM	SD	LS	Ganze Daten
9	BiLSTM	SD	LS	Ganze Daten
10	RNN	SD	RS	Daten nach FPT

Tabelle 4: Kombination unterschiedlicher Datenvorverarbeitung für RUL-Prognosemodelle

Kombination	RNNs Modelle	Merkmale Typ	Sampling-Methode	Datenart
11	LSTM	SD	RS	Daten nach FPT
12	BiLSTM	SD	RS	Daten nach FPT
13	RNN	Multi	RS	Daten nach FPT
14	LSTM	Multi	RS	Daten nach FPT
15	BiLSTM	Multi	RS	Daten nach FPT
16	RNN	SD	LS	Daten nach FPT
17	LSTM	SD	LS	Daten nach FPT
18	BiLSTM	SD	LS	Daten nach FPT

Abbildung 19: Workflow für die Datenvorverarbeitung

Feature Engineering: Um die Leistungsfähigkeit von LS-RUL besser zu validieren, wurde in dieser Arbeit die Wichtigkeit der multiplen Merkmale mittels Korrelationsanalyse bewertet. Wenn der VIF größer als 10 ist, gelangen die Merkmale in den multiplen Verarbeitungsprozess. Die verarbeiteten Daten werden je nach Sampling- und Parametertyp mittels Bootstrap-Methode erneut sampelt, um Test- und Trainingsdatensätze zu generieren. Der gesamte Datenvorverarbeitungsprozess und die Kennzeichnung der einzelnen Datensätze sind in Tabelle 4 dargestellt. Das Sankey-Diagramm in Abbildung 19 ermöglicht ein intuitiveres Verständnis des Datenverarbeitungsprozesses. Um Verzerrungen der Datenverteilung durch einzelne Sampling-Methoden zu vermeiden, wird die Zuordnung zwischen Lebesgue-Zuständen und Lebesgue-Zeitdomäne in jedem Trainings- und Testdatensatz mittels Bootstrap-Sampling 100-mal durchgeführt, um das endgültige Ergebnis zu erhalten. Der zentrale Code lautet wie folgt Abbildung 20:

```python
import pandas as pd
import numpy as np

def preprocess_data(df):
    df_sorted = df.sort_values('RUL_normalized', ascending=False).reset_index(drop=True)
    if df_sorted.isnull().values.any():
        df_sorted = df_sorted.dropna().reset_index(drop=True)
        print("Warnung: Fehlende Werte wurden entfernt.")
    return df_sorted

def define_bins(sd_vertical, delta=0.15):
    min_sd = np.min(sd_vertical)
    max_sd = np.max(sd_vertical)
    num_bins = int(np.ceil((max_sd - min_sd) / delta))
    return min_sd + delta * np.arange(num_bins + 1)

def assign_bins(df_sorted, bin_edges):
    df_sorted['Bin_Index'] = pd.cut(df_sorted['Sd.cp.Vertical'], bins=bin_edges, labels=False, include_lowest=True)
    return df_sorted

def resample_bins(df_sorted, bin_edges, num_resamples=100):
    resampled_data = []
    num_bins = len(bin_edges) - 1
    for resample_id in range(1, num_resamples + 1):
        for bin_index in range(num_bins):
            bin_df = df_sorted[df_sorted['Bin_Index'] == bin_index]
            if bin_df.empty:
                continue
            sampled_row = bin_df.sample(n=1, replace=True).iloc[0]
            resampled_data.append({
                'Resample_ID': resample_id,
                'Bin_Index': bin_index,
                'lebesgue_State': sampled_row['Sd.cp.Vertical'],
                'lebesgue_Time': sampled_row['RUL_normalized']
            })
    return pd.DataFrame(resampled_data)

def aggregate_datasets(resampled_df, bearing):
    resampled_df['Bearing'] = bearing
    return resampled_df[['Bearing', 'Resample_ID', 'lebesgue_State', 'lebesgue_Time']]

def main(data_dict):
    LEB_THRESH = 0.25
    NUM_RESAMPLES = 100
    train_dataframes = []
    test_dataframe = None

    for bearing, df in data_dict.items():
        print(f"Verarbeite {bearing}...")
        df_sorted = preprocess_data(df)
        bin_edges = define_bins(df_sorted['Sd.cp.Vertical'].values, delta=LEB_THRESH)
        df_sorted = assign_bins(df_sorted, bin_edges)
        resampled_df = resample_bins(df_sorted, bin_edges, num_resamples=NUM_RESAMPLES)

        if resampled_df.empty:
            print(f"{bearing}: Keine Stichproben nach Resampling generiert.")
            continue

        lebesgue_df = aggregate_datasets(resampled_df, bearing)

        if bearing in ['Bearing1_1', 'Bearing1_2', 'Bearing1_3', 'Bearing2_1', 'Bearing2_2', 'Bearing2_3', 'Bearing2_4', 'Bearing2_5']:
            train_dataframes.append(lebesgue_df)
        elif bearing == 'Bearing1_5':
            test_dataframe = lebesgue_df

    if train_dataframes:
        train_df = pd.concat(train_dataframes, ignore_index=True)
        print("Trainingsdaten generiert.")

    if test_dataframe is not None:
        print("Testdaten generiert.")

if __name__ == "__main__":
    data_dict = {
        'Bearing1_1': pd.DataFrame({'RUL_normalized': np.random.rand(100), 'Sd.cp.Vertical': np.random.rand(100)}),
        'Bearing1_5': pd.DataFrame({'RUL_normalized': np.random.rand(100), 'Sd.cp.Vertical': np.random.rand(100)}),
    }
    main(data_dict)
```

Abbildung 20: Lebesgue Sampling zentrale Code

Modelltraining und Ergebnisdarstellung: Es wurden RNN-, LSTM- und BiLSTM-Modelle verwendet, um verschiedene Datenverarbeitungs- und Sampling-Methoden zu trainieren. Die Prognoseergebnisse der Modelle bestehen aus dem Vergleich von prognostizierten

Werten und tatsächlichen Werten, wodurch die Anpassungsgenauigkeit der Prognosewerte an den Normalisierungsgrad direkt sichtbar wird.

Ergebnisinterpretation und -analyse: Abschließend werden die Fehlprognosen basierend auf den Modellergebnissen interpretiert und analysiert. Es erfolgt ein Vergleich der verschiedenen Sampling-Methoden, Datenvorverarbeitungsansätze sowie unterschiedlicher Deep-Learning-Modelle (RNN, LSTM, BiLSTM) hinsichtlich ihrer RMSE, R^2 und NASA-Scores im gesamten Prognoseauftrag, um die optimale Modellkonfiguration zu ermitteln. Abbildung 21 veranschaulicht das Flussdiagramm des FD-Moduls.

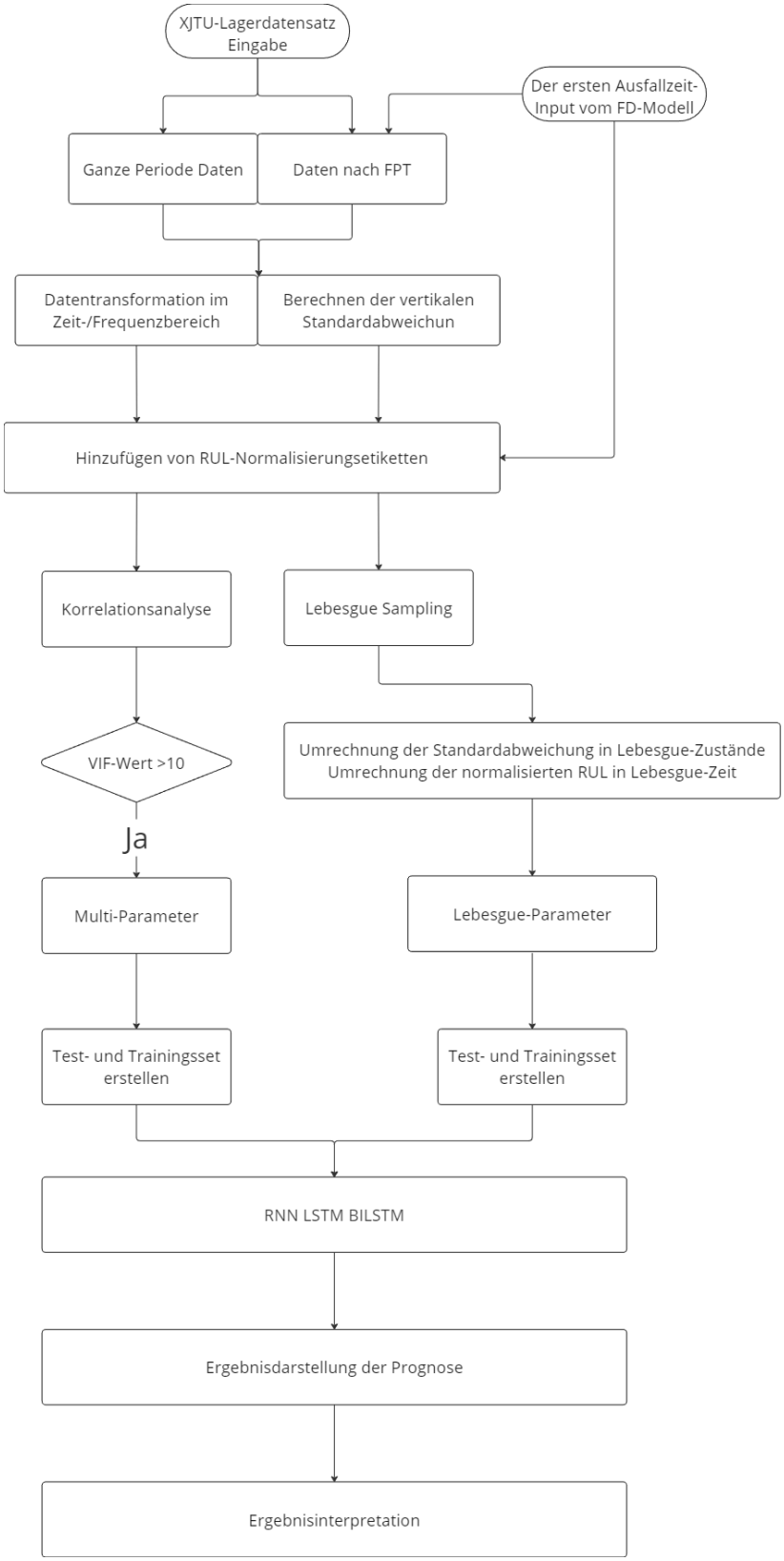


Abbildung 21: Das Flussdiagramm des RUL-Moduls.

5 Anwendungsbeispiele und Ergebnisse

5.1 Anwendungsbeispiele für Fehlerdiagnose-module

5.1.1 Datenverarbeitung

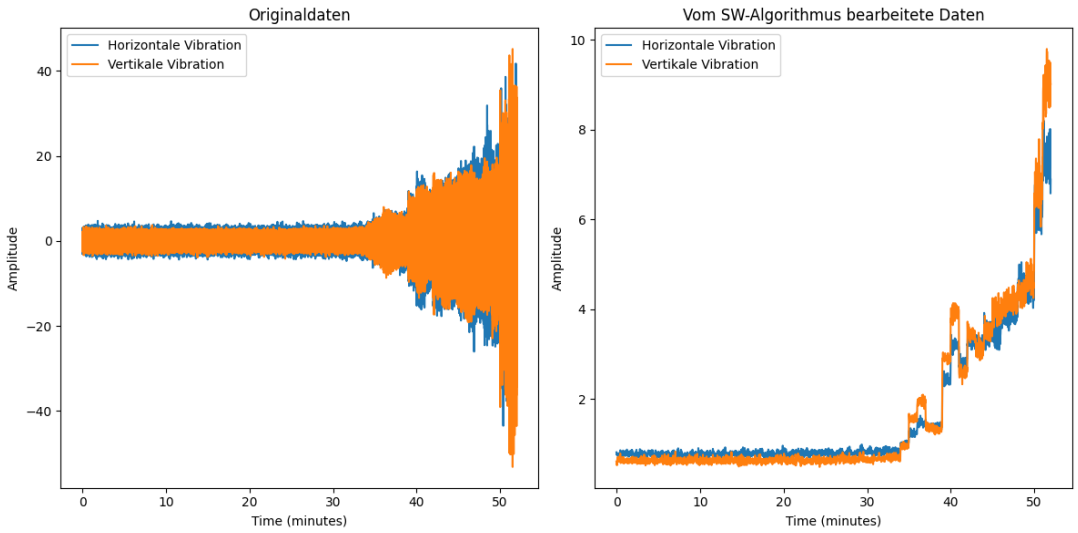


Abbildung 22: Daten vor und nach der Verarbeitung durch den SW-Algorithmus

Aufgrund der großen Datenmenge des ursprünglichen Vibrationssignals wird in dieser Arbeit ein gleitendes Fensterverfahren verwendet, um in praktischen Anwendungen Rechen- und Speicherkapazitäten zu sparen. Das Ziel des SW-Algorithmus ist es, den Mittelwert der Amplitude innerhalb jedes Fensters zu berechnen, indem ein festes Zeitfenster festgelegt wird. Da die Schwingungsdaten ein Muster das Hin- und Herbewegen in einer Achse zeigen, wird in dieser Arbeit zunächst der Absolutwert der Daten genommen, bevor der Mittelwert berechnet wird. Dadurch wird vermieden, dass der ermittelte Mittelwert nahe Null liegt und die meisten Details verloren gehen. Auf diese Weise werden die Hauptmerkmale der Wälzlagersignale effektiv extrahiert und die Datenmenge reduziert. Abbildung 22 zeigt die Daten der Wälzlager1_5 nach dem SW-Algorithmus und die Originaldaten.

5.1.2 Datenanalyse

Lei et al. [36] stellten in ihrer Arbeit eine Methode zur Bestimmung der ersten Prognosezeit vor, die die Schiefe und das 3-fache der Standardabweichung kombiniert. Die konkreten Schritte sind wie folgt: Zunächst wird für jede Lagerlaufzeit die Schiefe und deren Standardabweichung berechnet. Anschließend wird der Schwellenwert durch

das 3-fache der Standardabweichung der Schiefe festgelegt. Werte, die den Schwellenwert überschreiten und drei aufeinanderfolgende Ausreißer zeigen, werden als Beginn der Lagerdegradationsphase angesehen. Da in dieser Arbeit der SW-Algorithmus verwendet wurde, um die Datenmenge zu reduzieren und die Rechenzeit zu verkürzen. Der Glättungseffekt des gleitenden Fensters führte zu einer Veränderung der Datenform und -verteilung, was eine Änderung der Schiefe zur Folge hatte. Diese Veränderung ist in Abbildung 23 zu beobachten.

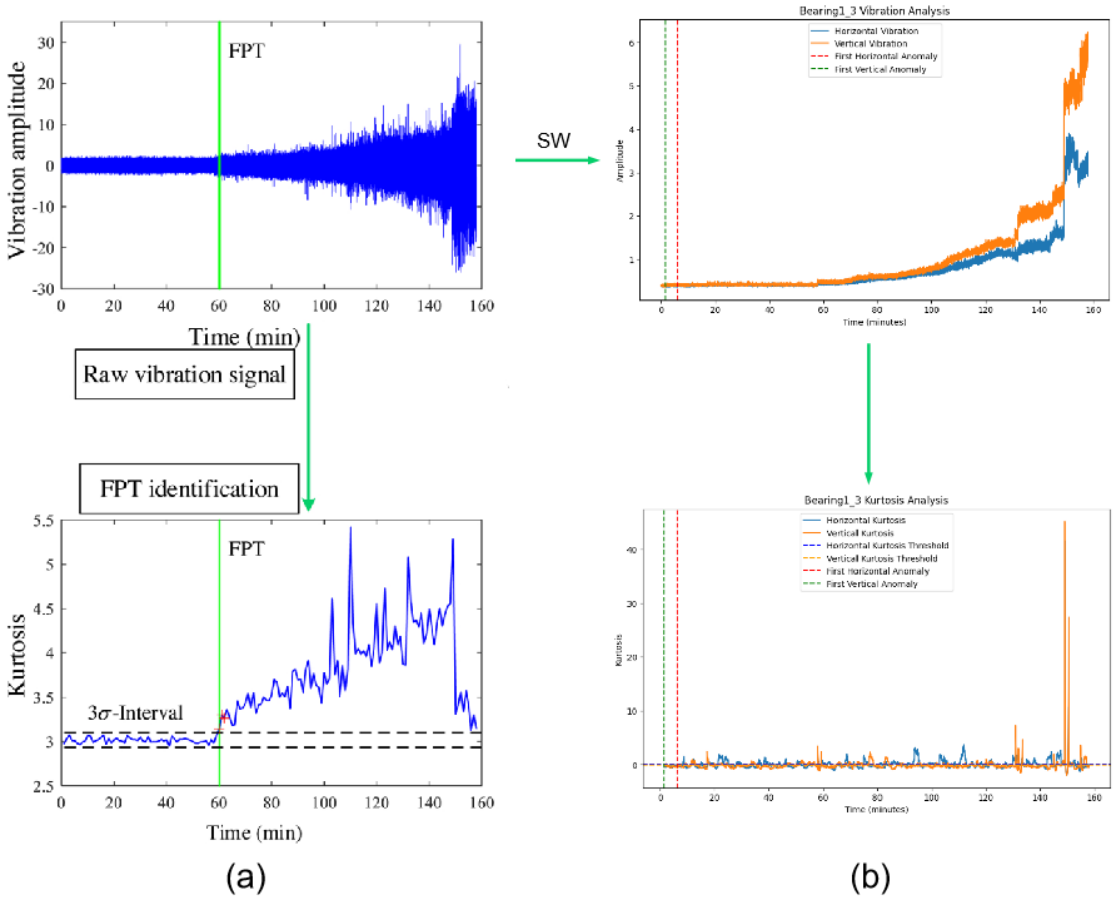


Abbildung 23: Vergleich der Fehlerdiagnose anhand der Kurtosis (a) der Originaldaten[20] und (b) der Daten nach Anwendung des SW-Algorithmus

Bei demselben Bearing1_3 zeigte sich in der Originalarbeit eine Schiefe, die einer rechtsschiefen lognormalen Verteilung entspricht. Nach Anwendung des SW-Algorithmus wurden die Daten durch das gleitende Fenster geglättet, sodass nur der Peak übrigblieb. Die ursprüngliche Methode ist in dieser Arbeit nicht mehr anwendbar. Daher wird in dieser Arbeit ein gemischter Parameter aus der Standardabweichung der Lagerrotation und der Gruppenstandardabweichung unter verschiedenen Betriebsbedingungen im Normalbetrieb als Schwellenwert zur Fehlerdiagnose und zur Abgrenzung von Ausreißern vorgeschlagen.

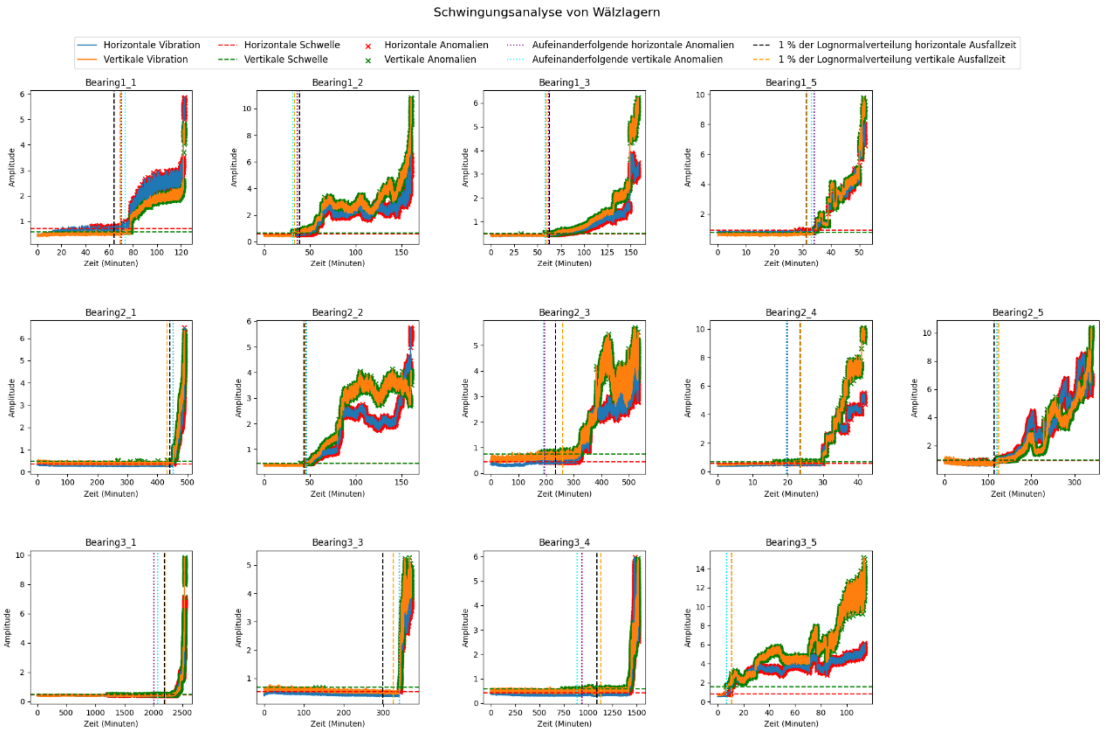


Abbildung 24: Die Diagnoseergebnisse der ersten Ausfallzeiten aller Lager (3σ)

Unter Berücksichtigung der Abbildung 24 und Abbildung 25 sowie Tabelle 4 wurde nach dem Vergleich der Fehler bei der Fehlerprognose mit einer Schwelle von 3- und 4-fachen Standardabweichungen festgestellt, dass die Prognoseergebnisse bei einem Schwellenwert von 4-fachen Standardabweichungen genauer sind. Insgesamt ist der mittlere quadratische Fehler (MSE) beim Schwellenwert von 4-fachen Standardabweichungen sowohl für die Leistung einzelner Lager als auch für die Gesamtergebnisse niedriger als beim Schwellenwert von 3-fachen Standardabweichungen, was darauf hinweist, dass der 4-fache Schwellenwert die Prognosefehler besser reduzieren kann. Daher wird in den folgenden Experimenten der Schwellenwert von 4-fachen Standardabweichungen verwendet, um die Genauigkeit des Modells zu verbessern.

Tabelle 5: Vergleichstabelle des kumulativen MES bei 3,4-facher Abstimmungsparameter

Lager	Summe der MES bei 4-facher Abstimmungsparameter	Summe der MES bei 3-facher Abstimmungsparameter
Bearing1_1	0,021082	0,012908
Bearing1_2	0,120613	0,122042
Bearing1_3	0,946886	0,433256
Bearing1_5	1,183258	1,183258
Bearing2_1	0,003863	0,005542
Bearing2_2	0,033625	0,023
Bearing2_3	0,444283	0,472
Bearing2_4	0,461847	4,62128
Bearing2_5	0,47358	0,669762
Bearing3_1	0,074392	0,280164
Bearing3_3	0,005159	0,003197
Bearing3_4	0,971498	2,036285
Bearing3_5	0,011485	0,012485
Total	4,751571	9,875179

Nach Anpassung des Schwellenwerts der Standardabweichung wird die Experimente erneut durchgeführt (sieht Anhang 1); die Diagnoseergebnisse der ersten Ausfallzeiten aller Lager sind in Abbildung 25 dargestellt. Es wurde festgestellt, dass die Diagnoseergebnisse bei der überwiegenden Mehrheit der Lager sehr zufriedenstellend sind und die vier vorgeschlagenen Methoden eine relativ konzentrierte Verteilung entlang der Zeitachse aufweisen. Allerdings werden aufgrund deutlich vorhandenen Rauschens in der frühen Phase einige Lager (wie Bearing2_3, Bearing2_4, Bearing3_4) zu früh als ausgefallen diagnostiziert, was in der Abbildung ersichtlich ist. Im Gegensatz dazu wird bei Lagern mit geringeren Schwankungen während des Betriebs (wie Bearing1_3) der Zeitpunkt des Fehlerauftritts relativ spät definiert.

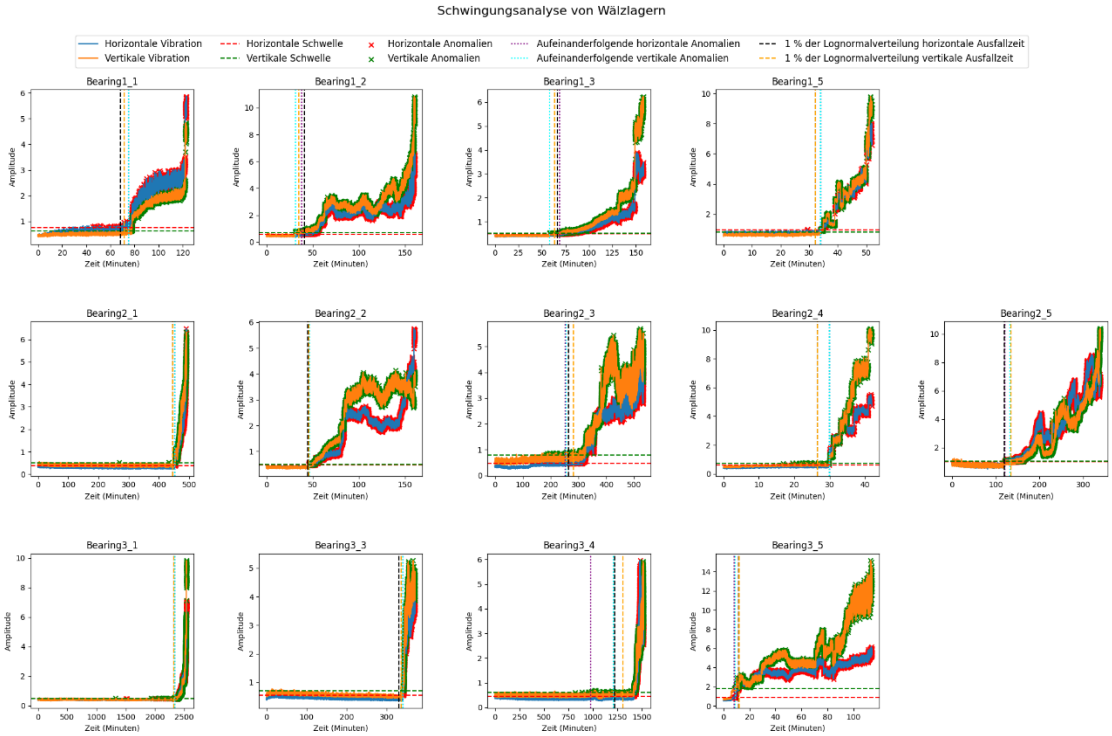


Abbildung 25: Die Diagnoseergebnisse der ersten Ausfallzeiten aller Lager (4σ)

5.1.3 Ergebnisdarstellung und -interpretation

Wie kann der erste Ausfallzeitpunkt eines Lagers am genauesten bestimmt werden? Zu diesem Zweck wird einen Harmonisierungsparameter eingeführt. Konkret wurde die Ausfallzeit von 0 bis 1 in 100 Abschnitte unterteilt; für jeden Abschnitt wurde das Experiment wiederholt, und die Ergebnisse wurden mit den in anderen Studien angegebenen ersten Ausfallzeiten verglichen. Die Vergleichsergebnisse der vier verschiedenen Bewertungsmethoden sind in Abbildung 26 dargestellt.

Oben links (MSE bei kontinuierlichen horizontalen Anomalien): Diese Abbildung zeigt die MSE-Leistung bei der Erkennung kontinuierlicher Anomalien in horizontaler Richtung. Man kann erkennen, dass die meisten Lager bei kleineren Werten des Harmonisierungsparameters einen niedrigen MSE beibehalten. Allerdings zeigen einige Lager bei höheren Parameterwerten signifikante Fehlerpeaks. Dies weist darauf hin,

dass die Methode unter bestimmten Bedingungen empfindlich auf frühes Rauschen reagiert, was zu instabilen Diagnosen führen kann.

Oben rechts (MSE mit 1 %-Schwellenwert in horizontaler Richtung): Unter einem Fehler-Schwellenwert von 1 % sinkt der MSE in horizontaler Richtung insgesamt, und die Fehlerfluktuation ist stabiler als bei der vorherigen Methode. Dennoch gibt es einige Lager, die in bestimmten Parameterbereichen höhere Fehler aufweisen. Dies zeigt, dass trotz der Einführung eines Fehler-Schwellenwerts die allgemeine Diagnosegenauigkeit verbessert wurde, aber die Methode für die Fehlerdiagnose bestimmter Lager immer noch nicht präzise genug ist.

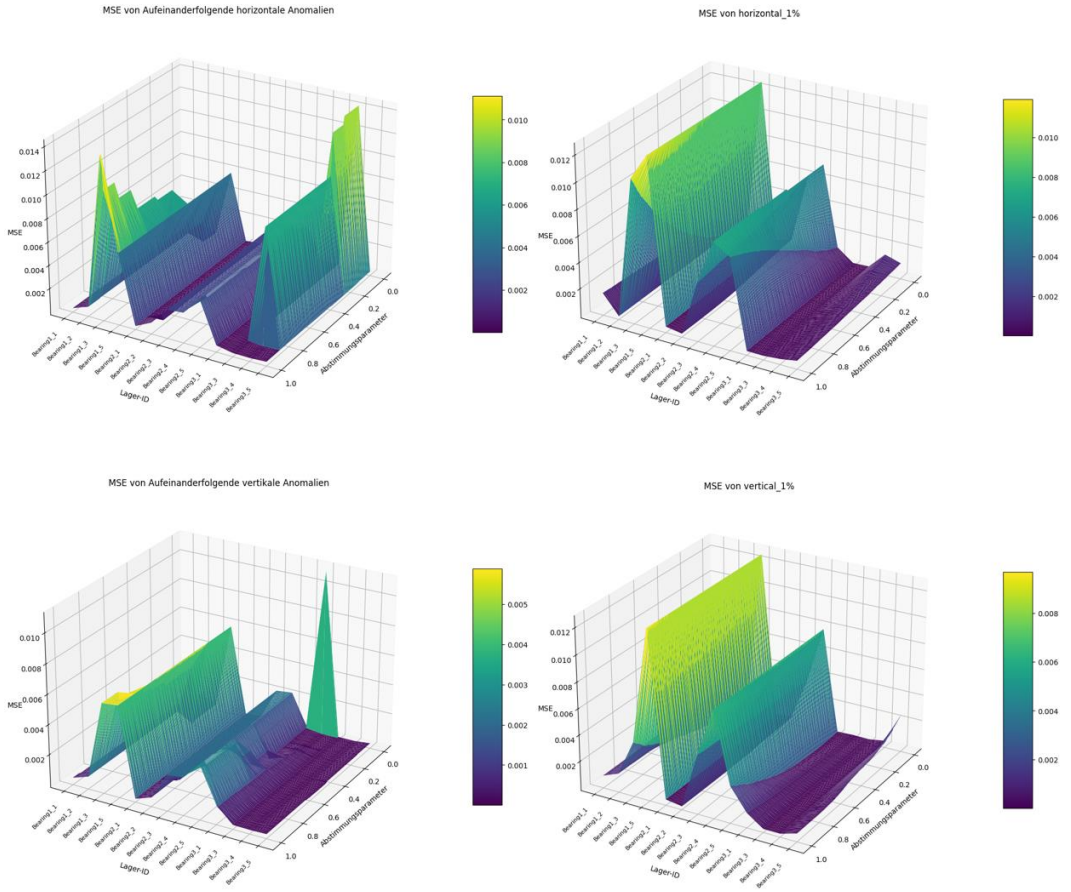


Abbildung 26: MSE-Bewertung der Fehlererkennungsmethoden bei Lagerdaten unter verschiedenen Betriebsbedingungen

Unten links (MSE bei kontinuierlichen vertikalen Anomalien): Die MSE bei der Erkennung kontinuierlicher Anomalien in vertikaler Richtung zeigt relativ stabilere Ergebnisse. Über den gesamten Bereich des Harmonisierungsparameters von 0 bis 1 schwankt der MSE wenig. Insbesondere im mittleren Bereich nähern sich die Diagnosefehler mehrerer

Lager nahezu null an. Dies deutet darauf hin, dass die vertikale Richtung in der Anomalieerkennung zuverlässiger ist und weniger von Rauschen beeinflusst wird.

Unten rechts (MSE mit 1 %-Schwellenwert in vertikaler Richtung): Unter einem Fehler-Schwellenwert von 1 % sinkt der MSE in vertikaler Richtung weiter, und nahezu alle Lager halten über den gesamten Bereich des Harmonisierungsparameters einen niedrigen Fehler aufrecht. Im Vergleich zu den anderen drei Methoden zeigt diese Methode die beste Leistung mit höherer Diagnosegenauigkeit. Insbesondere nach der Optimierung des Harmonisierungsparameters kann sie übermäßige oder verspätete Fehlerdiagnosen deutlich reduzieren.

Insgesamt zeigt die Detektion in vertikaler Richtung, insbesondere die vertikale Anomalieerkennung unter einem 1 %-Schwellenwert, die beste Diagnosegenauigkeit und Stabilität unter den vier Methoden. Im Gegensatz dazu ist die Detektion in horizontaler Richtung bei einigen Lagern empfindlicher gegenüber Rauschen, und die Fehler schwanken stärker.

Schließlich wurde der Punkt mit dem minimalen relativen Fehler unter den vier Bedingungen als erster Ausfallzeitpunkt des Lagers ausgewählt; die entsprechenden Daten sind in Tabelle 6 zusammengefasst.

Tabelle 6: Lager Früheste Ausfallzeiten

Lager ID	**Früheste Ausfallzeit (Minuten):**
Bearing1_1	75
Bearing1_2	44
Bearing1_3	60
Bearing1_5	35
Bearing2_1	454
Bearing2_2	48
Bearing2_3	302
Bearing2_4	30
Bearing2_5	141
Bearing3_1	2344
Bearing3_3	341
Bearing3_4	1418
Bearing3_5	9

5.2 Anwendungsbeispiele für RUL-Prognosemodul

5.2.1 Datenverarbeitung

Zunächst wurden alle 36 Daten der 18 Parameter in den beiden Vibrationsrichtungen berechnet und eine Korrelationsanalyse dieser Daten durchgeführt; die Ergebnisse sind in Abbildung 27 dargestellt. Durch die Korrelationsanalyse wurde festgestellt, dass eine beträchtliche Anzahl von Faktoren miteinander korreliert sind, was zu Multikollinearität führen und die Trainingseffekte des nachfolgenden RNN-Modells beeinflussen kann. Um die Parameter weiter auszuwählen, wurde der Varianzinflationsfaktor (VIF) zur Datenselektion verwendet; die Ergebnisse sind in Tabelle 7 dargestellt (sieht Anhang 2).

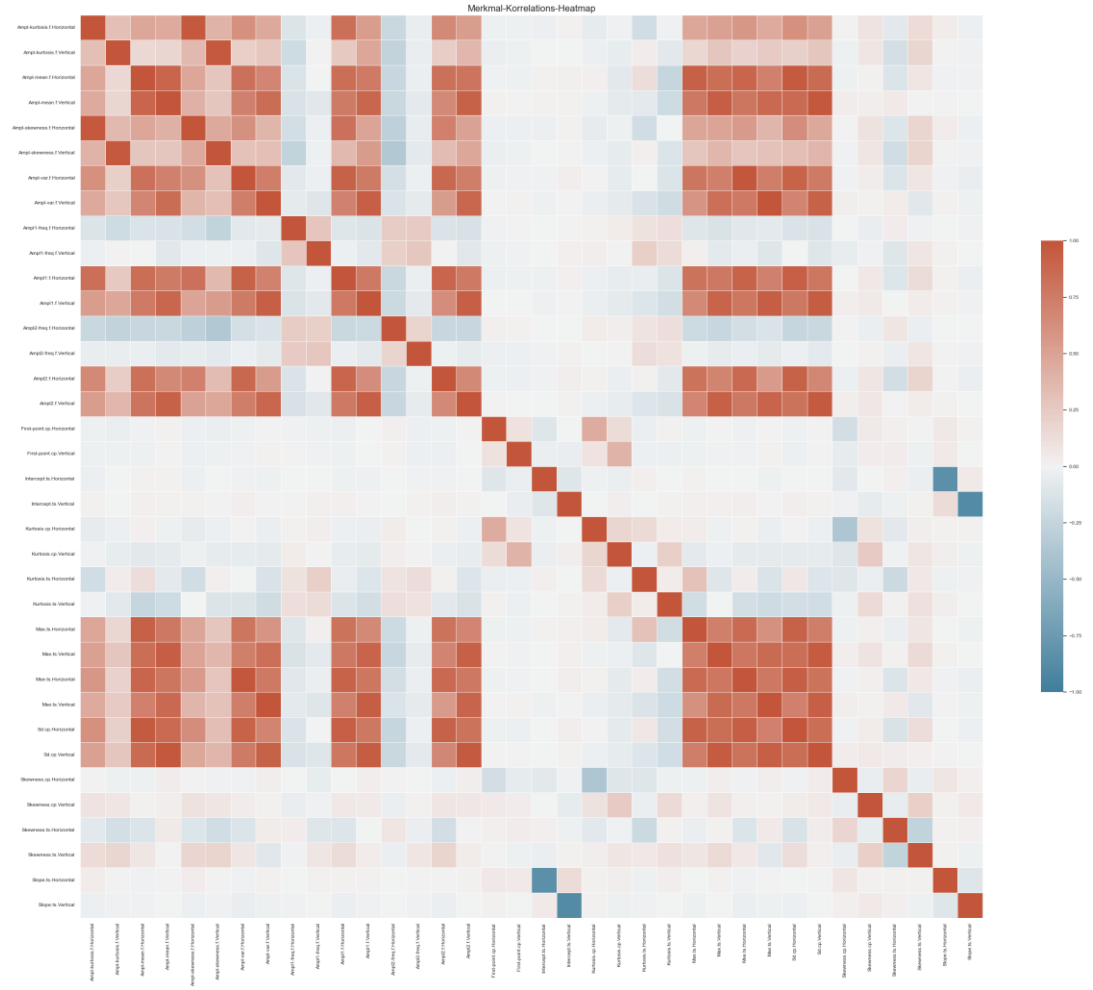


Abbildung 27: Heatmap für die Korrelationsanalyse

Parameter mit einem VIF-Wert kleiner als 10 wurden als Eingabe für die mehrparametrische Riemann-Stichprobe verwendet. Als Trainingsdatensatz wurde Lager 1_3 verwendet, um Prognosen für Lager 1_5 zu erstellen. Bezüglich LS wurden

nach der vorherigen Methode schließlich 18 verschiedene Datensätze vorbereitet. Der Größenvergleich der Datensätze ist in Tabelle 8 dargestellt; es ist ersichtlich, dass nach der LS-Stichprobe die Datengröße erheblich reduziert wurde. Dies trägt wesentlich dazu bei, die Anforderungen des Edge-Computings an erhöhte Rechengeschwindigkeit und reduzierten Speicherbedarf zu erfüllen.

Tabelle 7: VIF-Vergleich für alle Parameter

Merkmal	**VIF＜10**	**Merkmal**	**VIF＞10**
Skewness.cp.Vertical	1,171429	Ampl2.f.Horizontal	65,900276
Skewness.cp.Horizontal	1,301871	Max.ts.Horizontal	67,206329
Kurtosis.cp.Horizontal	1,772475	Ampl2.f.Vertical	91,665179
Skewness.ts.Vertical	2,001608	Max.ts.Vertical	111,712252
Ampl2-freq.f.Horizontal	2,027942	Ampl-kurtosis.f.Vertical	162,893995
Skewness.ts.Horizontal	2,194864	Ampl1.f.Vertical	217,566031
Kurtosis.cp.Vertical	2,35714	Ampl-kurtosis.f.Horizontal	222,079847
Kurtosis.ts.Vertical	2,393775	Ampl1.f.Horizontal	316,953453
Ampl2-freq.f.Vertical	2,552171	Ampl-skewness.f.Horizontal	582,926729
Ampl1-freq.f.Horizontal	2,596939	Ampl-skewness.f.Vertical	582,961009
Slope.ts.Horizontal	3,567953	Sd.cp.Vertical	1898,3323
Intercept.ts.Horizontal	3,590038	Ampl-mean.f.Vertical	2889,55023
First-point.cp.Vertical	3,608198	Ampl-var.f.Horizontal	3122,302568
First-point.cp.Horizontal	3,634578	Sd.cp.Horizontal	3365,725637
Ampl1-freq.f.Vertical	3,769236	Mse.ts.Horizontal	4453,930744
Kurtosis.ts.Horizontal	4,258772	Ampl-mean.f.Horizontal	4748,539896
Slope.ts.Vertical	4,788857	Ampl-var.f.Vertical	11363,93748
Intercept.ts.Vertical	4,83461	Mse.ts.Vertical	13987,51884

Tabelle 8: Größenvergleich der Datensätze

Kombination	123	456	789	101112	131415	161718	Neuer Trainingssatz mit sieben Lagern
Größe der Trainingsdaten (MB)	115	401	0,487	7	236	0,436	1,2
Größe der Testdatensätze (MB)	14	48,8	0,262	4,6	15,6	0,295	0,295

5.2.2 Datenanalyse

Nach der Eingabe dieser 18 Datensätze in die drei Modelle der RNN-Serie wurden für jedes Lager in seinem jeweiligen Zeitbereich die RUL-Prognosekurven erhalten. Die Hyperparametereinstellungen für das Modell sind in Anhang 3 aufgeführt. Die zugehörigen Ergebnisbilder sind in den Abbildungen 28-34 dargestellt. Anschließend werden sie gruppenweise diskutiert.

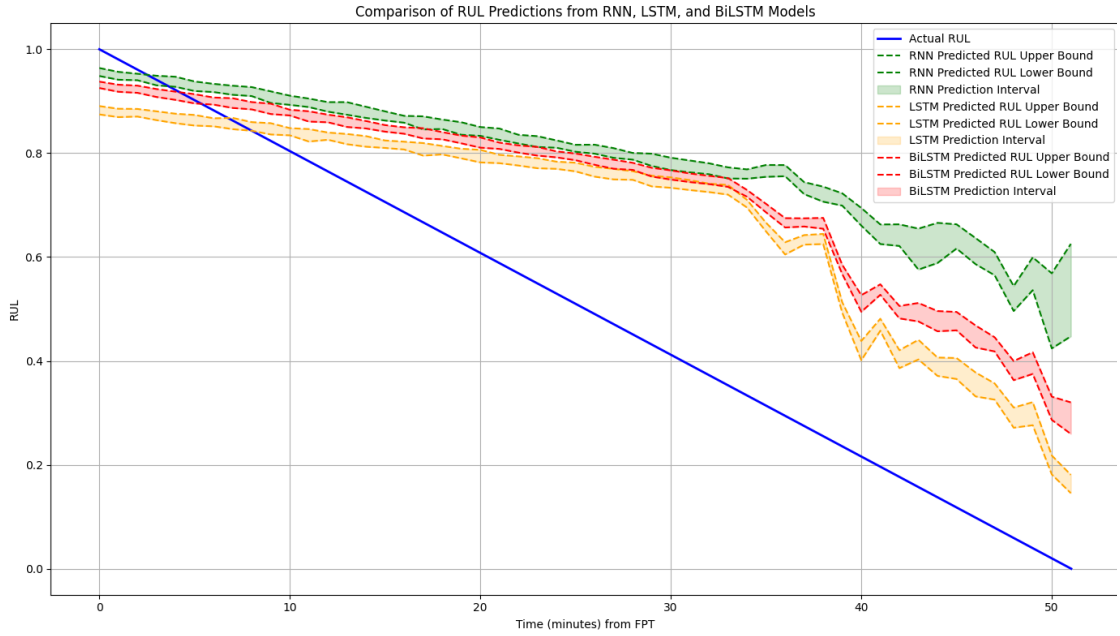


Abbildung 28: Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 1 2 3

Zunächst werden die Ergebnisse des gesamten Zeitraums und der Auswahl nur der zweiten Hälfte des Zeitraums betrachtet. Durch den Vergleich dieser beiden Bildsätze zeigt sich, dass der RUL-Anpassungsgrad beim Verwenden der Daten des gesamten Zeitraums als Trainings- und Prognosedatensatz sehr unbefriedigend ist. Dies liegt daran, dass die Zustandsparameter des Lagers während der anfänglichen normalen Betriebsphase relativ stabil sind, aber die verbleibende Zeit tatsächlich abnimmt. Das Modell kann den abnehmenden Trend nicht effektiv erlernen, was durch die Eigenschaften der Eingabedaten bestimmt wird und vom Modell schwer zu überwinden ist. Im Gegensatz dazu kann das Modell durch die Eingabe von Daten nur nach Auftreten

des Fehlers aufgrund der größeren Parameteränderungen den Verlauf besser erlernen, sodass das Ergebnis im Vergleich zu vorher verbessert wird.

Im Vergleich von RS und LS zeigt sich, dass der Anpassungsgrad der LS-Prognosewerte deutlich besser ist als der der auf RS basierenden Prognose. Einerseits wird in beiden Datensätzen die normalisierte RUL prognostiziert, was nicht die Stärke der auf RS basierenden Modelle ist; andererseits schwanken die Daten im Lebesgue-Raum-Zeit-Modell ständig, selbst während der normalen Betriebsphase, nur dass der Abwärtstrend verlangsamt ist.

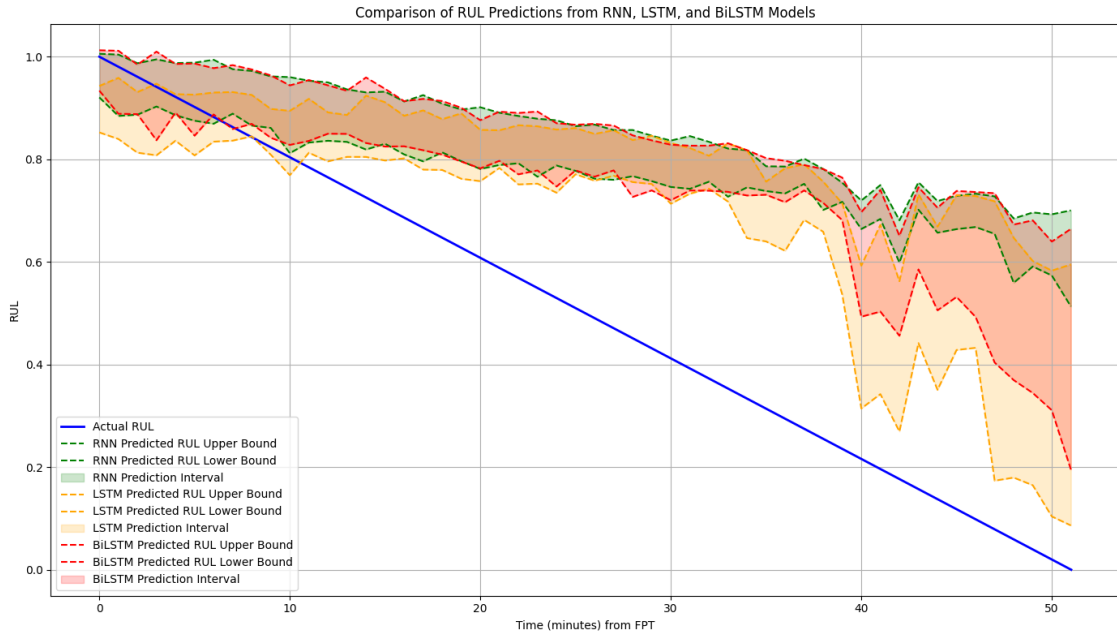


Abbildung 29: Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 4 5 6

Es wurde jedoch auch festgestellt, dass gemäß dem Vergleich von Abbildung 30 und Abbildung 33 das Modell theoretisch nach dem Fehler eine bessere Leistung zeigen sollte, jedoch beginnt der initiale Teil der Daten nicht mit einem Abfall von einem Gesundheitsindex von 1. Die Analyse legt nahe, dass dies auf eine zu geringe Anzahl von Trainingsmustern zurückzuführen sein könnte, wodurch nicht genügend Merkmale erlernt wurden. Daher wurde die Anzahl der Trainingsdatensätze von drei Lagern auf sieben Lager erhöht, und die Trainingsergebnisse verbesserten sich sofort. Dies zeigt, dass das Modell mehr Merkmale gelernt hat und der Anpassungseffekt ebenfalls besser ist.

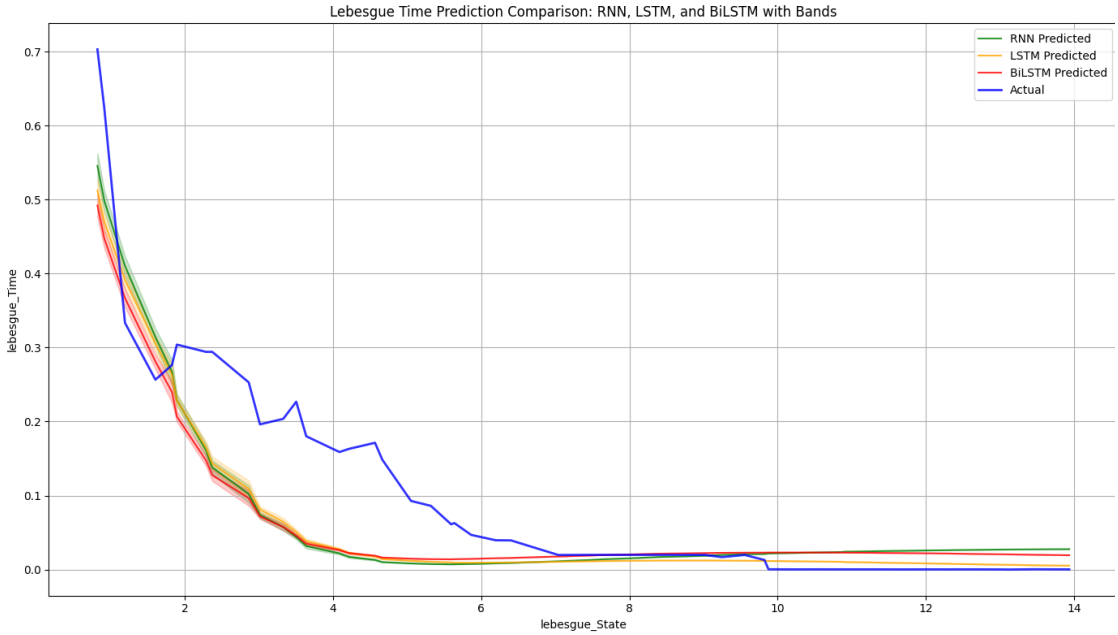


Abbildung 30: Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 7 8 9

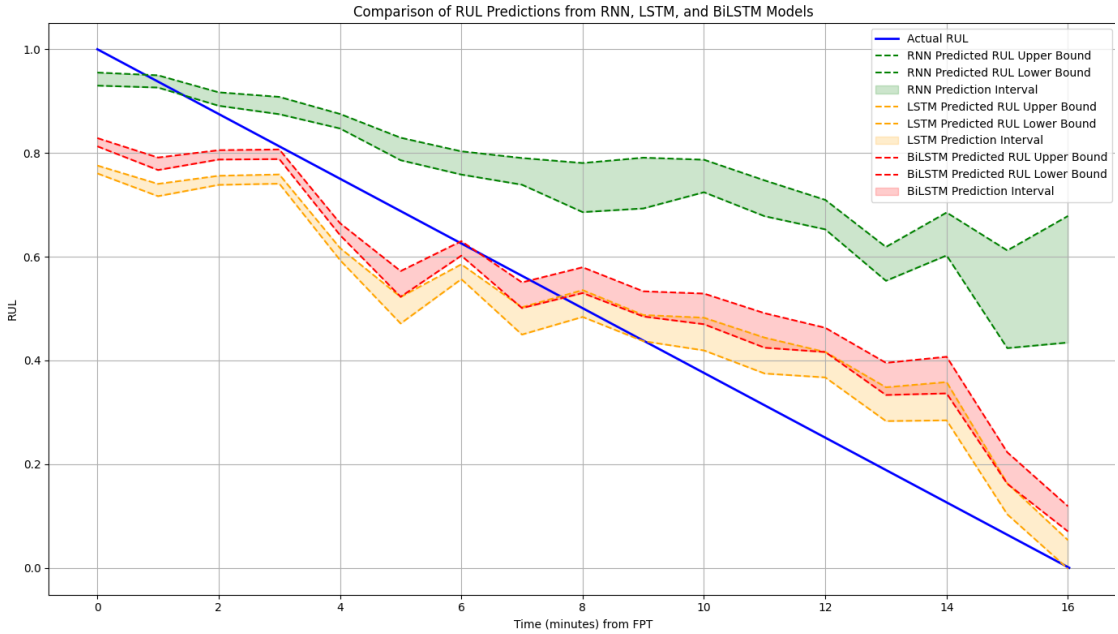


Abbildung 31: Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 10 11 12

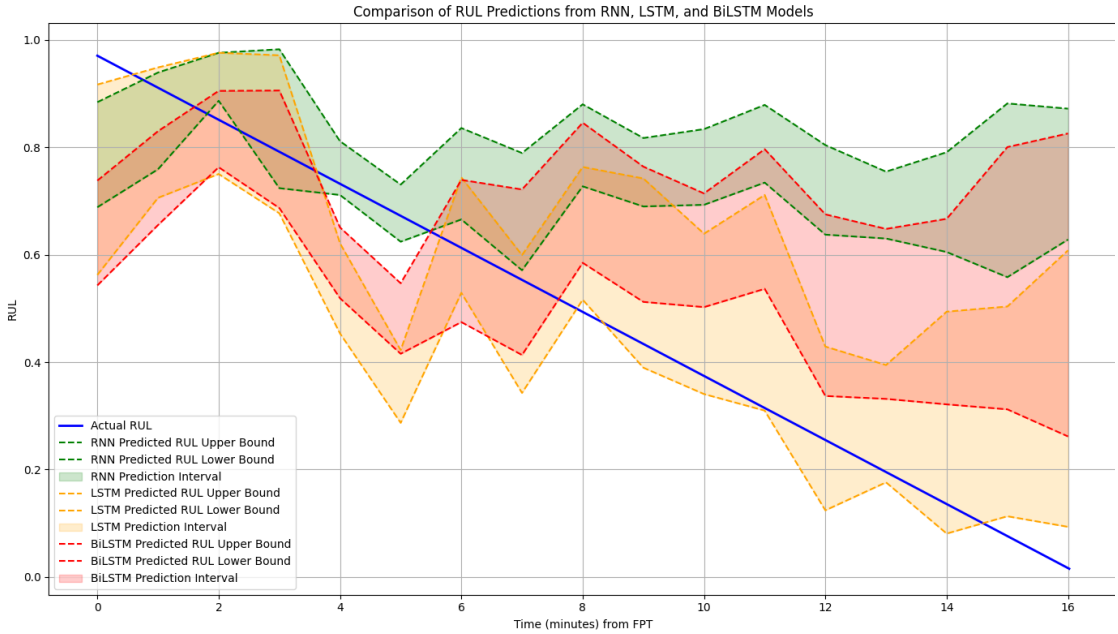


Abbildung 32: Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 13 14 15

Im Hinblick auf den Vergleich von Mehrparameter- und Einzelparameter-Eingaben wurde festgestellt, dass die Kurvenbandbreite der Mehrparameter-Prognose im Vergleich zur Einzelparameter-Eingabe deutlich breiter ist, was möglicherweise auf eine unvollständige Dimensionsreduktion der Dateneingabe und die Einführung von Rauschen zurückzuführen ist. Obwohl die Kurve einen Abwärtstrend aufweist, ist der Anpassungsgrad unzureichend, sodass möglicherweise weitere Arbeiten bei der Merkmalsauswahl erforderlich sind.

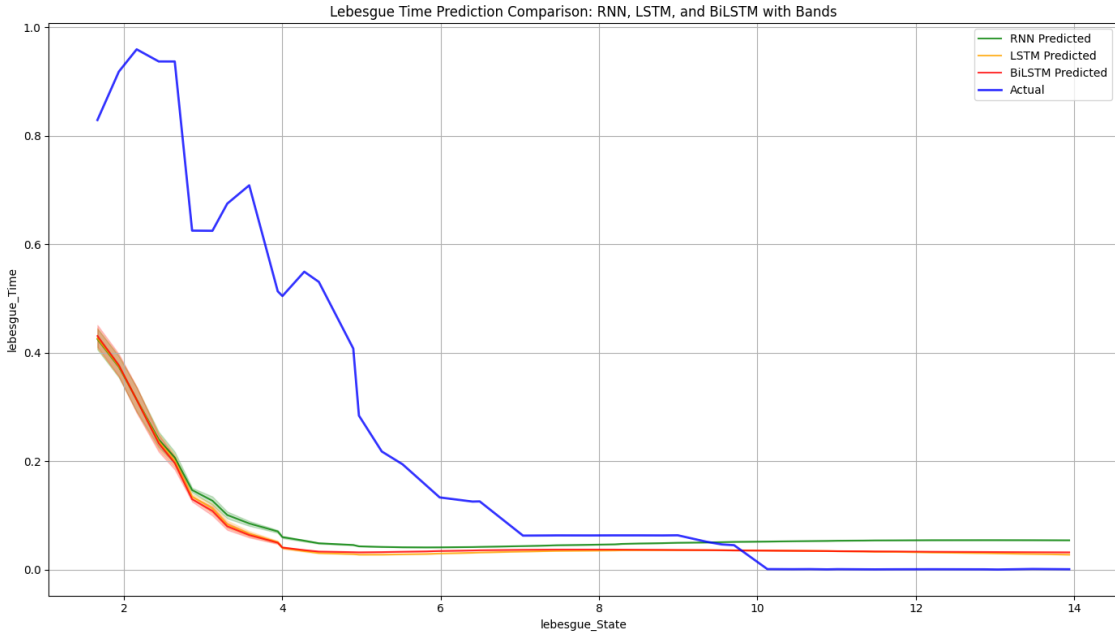


Abbildung 33: Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 16 17 18

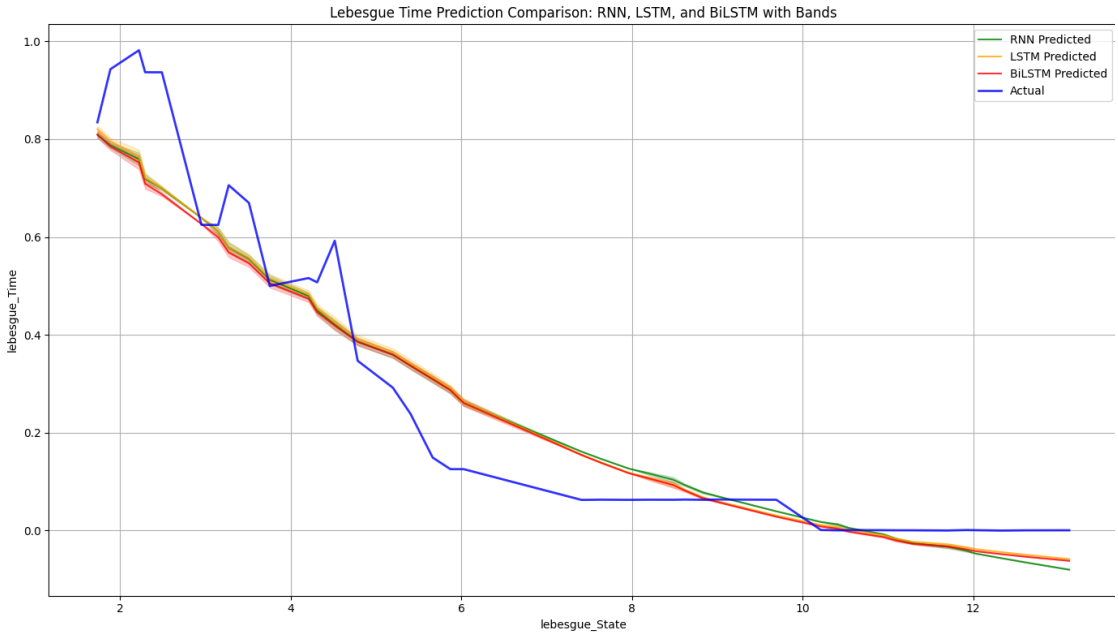


Abbildung 34 Prognoseergebnisse von RNN, LSTM und BiLSTM mit Datenvorverarbeitungskombination 19 20 21

5.2.3 Ergebnisdarstellung und -interpretation

Die Abbildung 35 zeigt die Prognosefehler der RMSE-Wert in verschiedenen Datensätzen. Ein niedrigerer RMSE-Wert weist auf geringere Prognosefehler des Modells hin. Im Trainingsdatensatz weisen BiLSTM-Modelle generell niedrigere RMSE-Werte auf als andere Modelle, insbesondere in komplexen Datensätzen (z. B. 10, 11 und 16) ist die Leistung besonders bemerkenswert. Dies deutet darauf hin, dass BiLSTM-Modelle komplexe Zeitreihenprognoseaufgaben besser bewältigen und die Akkumulation von Fehlern verringern können. In den meisten Datensätzen kann die LS-Methode den RMSE effektiv senken, insbesondere bei Prognosen nach Fehlerpunkten, indem die LS-Methode durch feinere Zeitsegmentierung die Veränderungen des Systemgesundheitszustands effektiv erfassen kann.

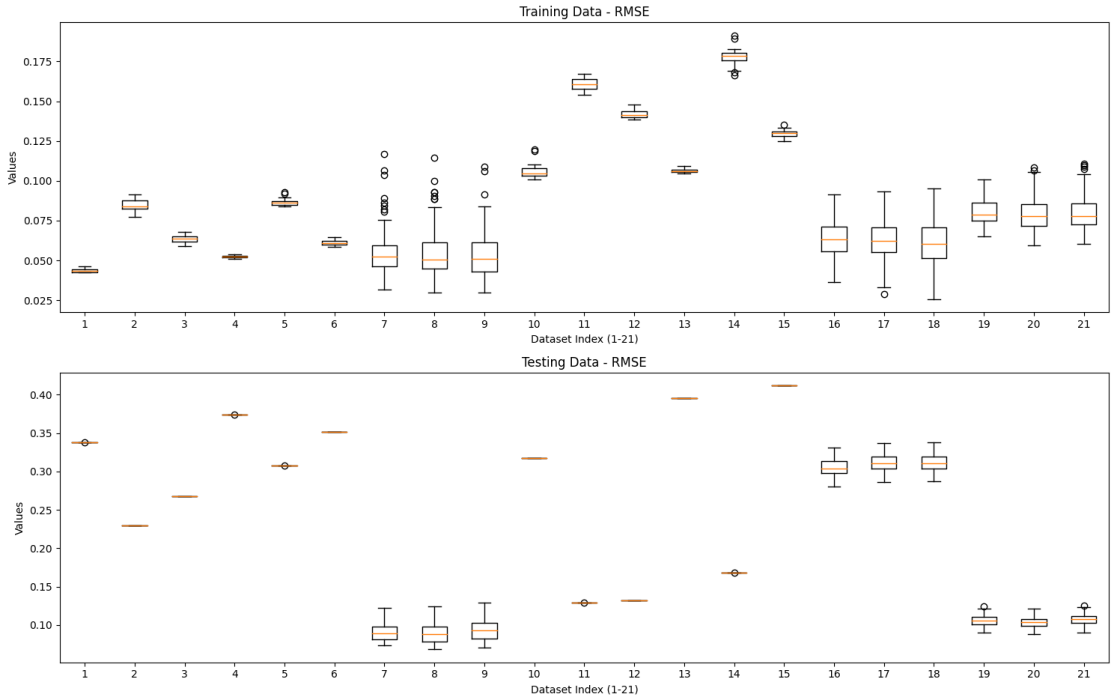


Abbildung 35: Boxplot des RMSE (Trainings- und Testdaten)

Die Abbildung 36 zeigt ein Boxplot der R^2-Werte (Bestimmtheitsmaß), die die Leistungen im Trainings- und Testdatensatz vergleichen. Ein R^2-Wert näher an 1 zeigt eine bessere Modellanpassung. Aus der Abbildung ist ersichtlich, dass die BiLSTM-Modelle in den meisten Datensätzen höhere R^2-Werte aufweisen als RNN- und LSTM-Modelle, was darauf hinweist, dass sie eine stärkere Anpassungsfähigkeit bei Zeitreihenprognoseaufgaben besitzen, insbesondere bei der Schulung von Modellen mit mehreren Parametern. In komplexen Datensätzen wie 1, 2, 5 und 10 führt die Kombination von LS-Methode mit BiLSTM-Modellen zu einer signifikanten Steigerung der R^2-Werte, was darauf hinweist, dass die LS-Methode die nichtlinearen Merkmale in den Zeitreihen besser erfassen kann, insbesondere die feinen Veränderungen in komplexen Vibrationssignalen.

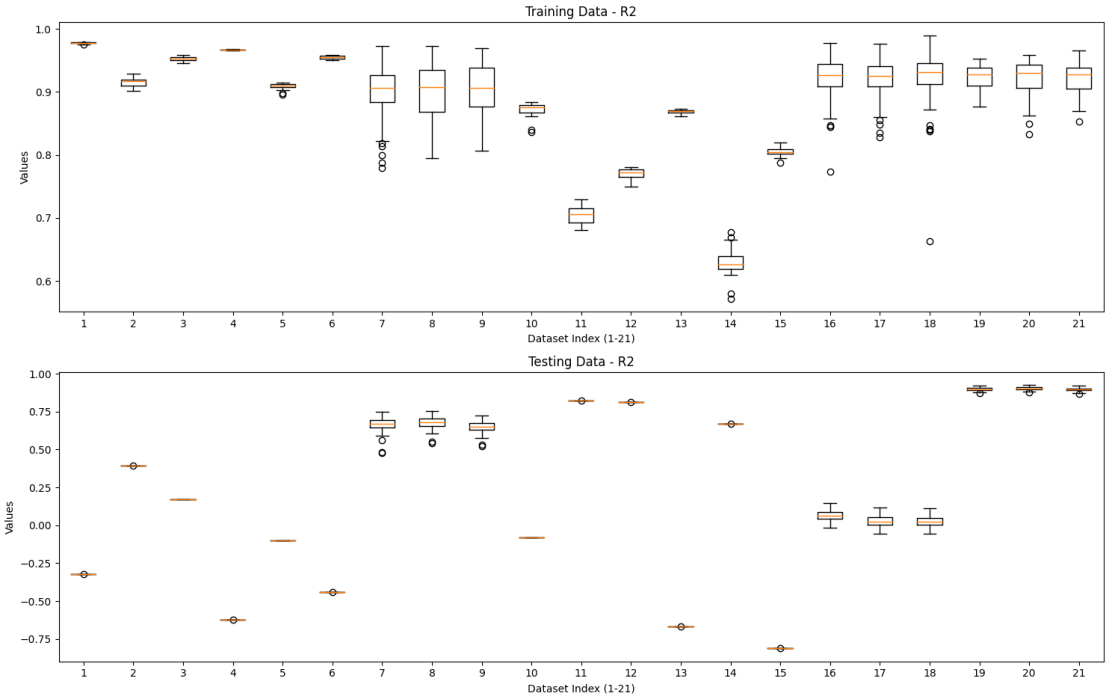


Abbildung 36: Boxplot der R^2-Werte (Trainings- und Testdaten)

Die Abbildung 37 zeigt ein Boxplot der NASA-Scores, unterteilt in Trainings- und Testdatensätze. Jeder Datensatz entspricht den Leistungen der RNN-, LSTM- und BiLSTM-Modelle unter verschiedenen Abtastmethoden (RS vs. LS). Im Trainingsdatensatz weisen alle Modelle allgemein hohe NASA-Scores auf, insbesondere die Datensätze 1, 2 und 11 zeigen eine besonders stabile Leistung. Dies deutet darauf hin, dass die Modelle die Trainingsdaten gut anpassen können. Allerdings zeigen einige Datensätze (z. B. 7, 14, 17) im Testdatensatz einen signifikanten Bewertungsabfall, was darauf hindeutet, dass diese Datensätze eine höhere Merkmalskomplexität aufweisen und Überanpassung auftreten könnte. In den meisten Datensätzen erzielte die LS-Methode höhere NASA-Scores, insbesondere in komplexen Datensätzen (z. B. 10, 11, 16) ist die Leistung der LS-Methode besonders bemerkenswert. Dies zeigt, dass die LS-Methode den Gesundheitszustand nach Fehlerpunkten besser erfassen und die Prognosestabilität der Modelle verbessern kann.

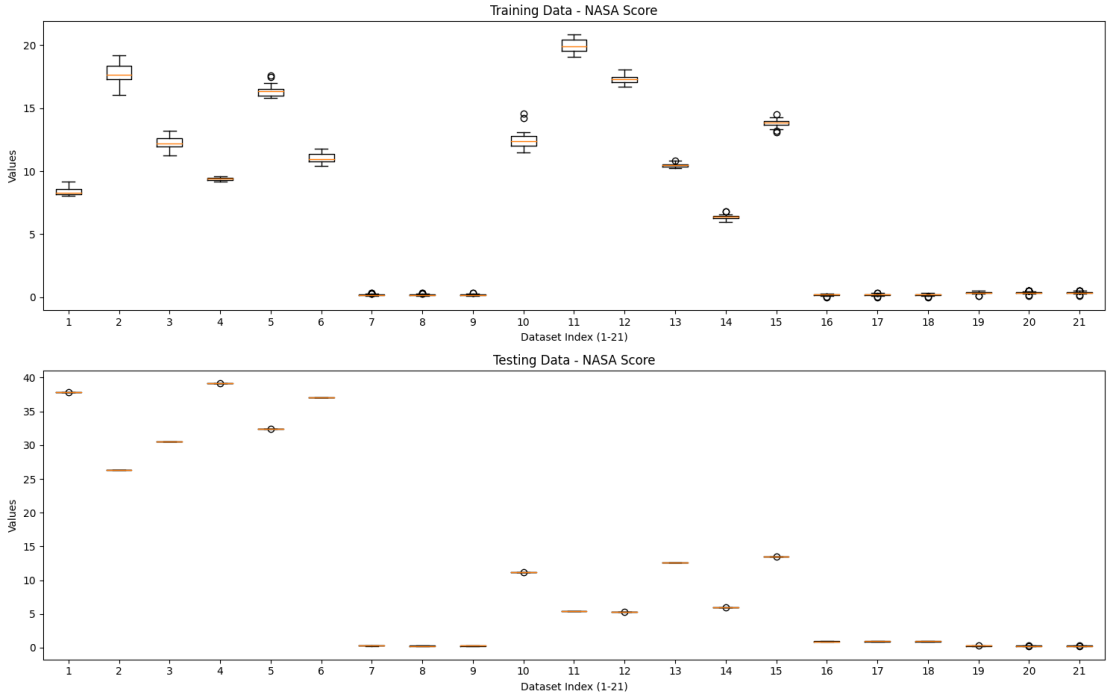


Abbildung 37: Boxplot der NASA-Scores (Trainings- und Testdaten)

6 Fazit

In dieser Masterarbeit wurde die Anwendung von RNNs in Kombination mit Lebesgue-Sampling zur Prognose der verbleibenden Nutzungsdauer RUL von Wälzlagern im Kontext der Industrie 4.0 untersucht. Ziel war es, eine effiziente und präzise Methode zur prädiktiven Instandhaltung zu entwickeln, die sowohl die Genauigkeit der Prognosen als auch die Rechenressourcen optimiert.

Zusammenfassung der Ergebnisse

Die durchgeführten Experimente zeigten, dass die Verwendung von BiLSTM-Modellen in Verbindung mit der LS-Methode signifikante Vorteile gegenüber traditionellen RNN- und LSTM-Modellen sowie der herkömmlichen RS-Methode bietet. Insbesondere in komplexen Datensätzen (z. B. Datensätze 10, 11 und 16) erzielten die BiLSTM-Modelle höhere R^2-Werte und niedrigere RMSE-Werte, was auf eine verbesserte Anpassungsfähigkeit und geringere Prognosefehler hinweist. Zudem führten die LS-Methode zu einer effektiven Reduktion der Datenmenge, ohne die Prognosegenauigkeit zu beeinträchtigen, was insbesondere für den Einsatz in Edge-Computing-Umgebungen von Vorteil ist. Die NASA-Scores bestätigten die Überlegenheit der LS-Methode, indem sie eine bessere Erfassung des Gesundheitszustands nach Fehlerpunkten und eine höhere Prognosestabilität der Modelle aufzeigten. Die detaillierte Analyse einzelner Datensätze (z. B. 14 und 16) verdeutlichte, dass die LS-Methode insbesondere bei der Verarbeitung von Vibrationssignaldaten komplexe nichtlineare Beziehungen besser modellieren kann, was zu präziseren RUL-Prognosen führt.

Ausblick und zukünftige Forschung

Basierend auf den erzielten Ergebnissen ergeben sich mehrere vielversprechende Forschungsrichtungen:

Parameterwahl und Optimierung

Eine vertiefte Untersuchung der Parameterwahl könnte dazu beitragen, die Modelle weiter zu optimieren. Durch den Vergleich und die Verbesserung der ausgewählten Parameter besteht die Möglichkeit, die Leistung der RUL-Prognosemethoden sogar über die bisherigen Ergebnisse bei Batterien hinaus zu steigern.

Wirtschaftliche Effekte

Die Reduktion der Anforderungen an die Datenverarbeitung führt zu einer Senkung der Betriebskosten. Eine wirtschaftliche Analyse könnte aufzeigen, wie durch die Verringerung der Rechenressourcen und die damit verbundenen Kosteneinsparungen eine relativ genaue prädiktive Instandhaltung realisiert werden kann. Dies würde die Attraktivität und Umsetzbarkeit der entwickelten Methoden in der Industrie weiter erhöhen.

Integration des Fehlerdiagnosemoduls (FD)

Die Kombination des Fehlerdiagnosemoduls mit den RUL-Prognosemodulen bietet die Möglichkeit, ein umfassenderes Instandhaltungssystem zu entwickeln. Durch die Erweiterung des Systems um ein Fehlerdiagnosemodul können nicht nur Ausfälle vorhergesagt, sondern auch deren Ursachen identifiziert werden. Dies ermöglicht eine gezieltere und effizientere Wartung, wodurch die Gesamteffizienz der Instandhaltungsprozesse weiter verbessert wird.

Echtzeit-Implementierung

Die Implementierung der entwickelten Methoden in Echtzeitsystemen und deren Validierung in industriellen Umgebungen stellen wichtige nächste Schritte dar, um die Praxistauglichkeit der Ansätze zu bestätigen.

Durch die Berücksichtigung dieser zukünftigen Forschungsrichtungen kann die Effizienz und Genauigkeit der prädiktiven Instandhaltungssysteme weiter gesteigert werden, was einen erheblichen Mehrwert für die Industrie 4.0 darstellt.

Die vorliegende Arbeit demonstriert, dass die Kombination von BiLSTM-Modellen mit Lebesgue-Sampling eine effektive Methode zur RUL-Prognose von Wälzlagern darstellt. Diese Methode bietet nicht nur eine hohe Prognosegenauigkeit, sondern auch eine effiziente Datenverarbeitung, was sie zu einer wertvollen Komponente in modernen prädiktiven Instandhaltungssystemen macht. Trotz bestehender Herausforderungen zeigt die Studie das Potenzial dieser Ansätze auf und legt den Grundstein für weiterführende Forschungen im Bereich der Zuverlässigkeitstechnik in der Industrie 4.0.

Literatur

1. (2023) XJTU-SY Bearing Datasets. https://biaowang.tech/xjtu-sy-bearing-datasets/. Zugegriffen: 03. Mai 2024
2. [Attoui et al. 2017]. Attoui I, Fergani N, Boutasseta N, Oudjani B, Deliou A (2017) A new time–frequency method for identification and classification of ball bearing faults. Journal of Sound and Vibration 397:241–265. doi:10.1016/j.jsv.2017.02.041
3. [Ayaz et al. 2022]. Ayaz BK, Huber M, Kröckel J, Ingold R, Oppermann H, Reimann D, Taschek H, Schulz T (2022) Analytics in der Industrie. Schlüsseltechnologie für die digitale Transformation, 1. Aufl. Vogel Communications Group GmbH & Co. KG, Würzburg
4. [Bastiaans 1985]. Bastiaans M (1985) On the sliding-window representation in digital signal processing. IEEE Trans. Acoust., Speech, Signal Process. 33(4):868–873. doi:10.1109/TASSP.1985.1164653
5. [Bergs et al. 2020]. Bergs T, Klocke F, Trauth D, Rey J (2020) Fertigungstechnik 4.0: Mit sicheren Audit-Trails und verteilten Fertigungsketten zur Fertigungsökonomie. In: Frenz W (Hrsg) Handbuch Industrie 4.0: Recht, Technik, Gesellschaft. Springer Berlin Heidelberg, Berlin, Heidelberg, S 517–541
6. [Bracke 2022]. Bracke S (2022) Technische Zuverlässigkeit. Springer Berlin Heidelberg, Berlin, Heidelberg
7. [Broy 2010]. Broy M (2010) Cyber-Physical Systems. Innovation Durch Softwareintensive Eingebettete Systeme. Acatech DISKUTIERT Ser. Springer Berlin / Heidelberg, Berlin, Heidelberg
8. [Chen et al. 2013]. Chen Y-K, Wu A-Y, Bayoumi MA, Koushanfar F (2013) Editorial Low-Power, Intelligent, and Secure Solutions for Realization of Internet of Things. IEEE J. Emerg. Sel. Topics Circuits Syst. 3(1):1–4. doi:10.1109/JETCAS.2013.2244771
9. [Chen et al. 2014]. Chen M, Mao S, Liu Y (2014) Big Data: A Survey. Mobile Netw Appl 19(2):171–209. doi:10.1007/s11036-013-0489-0
10. [Chicco et al. 2021]. Chicco D, Warrens MJ, Jurman G (2021) The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Comput Sci 7:e623. doi:10.7717/peerj-cs.623
11. [Correa 2020]. Correa JCJ (2020) Mechanical vibrations and condition monitoring. Academic Press, London, San Diego, CA, Cambridge, MA, Kidlington, Oxfrod

12. [Garcia-Perez et al. 2011]. Garcia-Perez A, Romero-Troncoso RdJ, Cabal-Yepez E, Osornio-Rios RA (2011) The Application of High-Resolution Spectral Analysis for Identifying Multiple Combined Faults in Induction Motors. IEEE Trans. Ind. Electron. 58(5):2002–2010. doi:10.1109/TIE.2010.2051398
13. [Gartner 28.11.2023]. Gartner (2023) Definition of Edge Computing - Gartner Information Technology Glossary. https://www.gartner.com/en/information-technology/glossary/edge-computing. Zugegriffen: 28. November 2023
14. [Gogtay und Thatte 2017]. Gogtay NJ, Thatte UM (2017) Principles of Correlation Analysis. J Assoc Physicians India 65(3):78–81
15. [Goyal und Pabla 2016]. Goyal D, Pabla BS (2016) The Vibration Monitoring Methods and Signal Processing Techniques for Structural Health Monitoring: A Review. Arch Computat Methods Eng 23(4):585–594. doi:10.1007/s11831-015-9145-0
16. [Guo et al. 2017]. Guo L, Li N, Jia F, Lei Y, Lin J (2017) A recurrent neural network based health indicator for remaining useful life prediction of bearings. Neurocomputing 240:98–109. doi:10.1016/j.neucom.2017.02.045
17. [Guo et al. 2023]. Guo J, Wang J, Wang Z, Gong Y, Qi J, Wang G, Tang C (2023) A CNN‐BiLSTM‐Bootstrap integrated method for remaining useful life prediction of rolling bearings. Quality & Reliability Eng 39(5):1796‐1813. doi:10.1002/qre.3314
18. [Hering und Schönfelder 2018]. Hering E, Schönfelder G (2018) Sensoren in Wissenschaft und Technik. Springer Fachmedien Wiesbaden, Wiesbaden
19. [Hoffmann Souza et al. 2020]. Hoffmann Souza ML, Da Costa CA, Oliveira Ramos G de, Da Rosa Righi R (2020) A survey on decision-making based on system reliability in the context of Industry 4.0. Journal of Manufacturing Systems 56:133–156. doi:10.1016/j.jmsy.2020.05.016
20. [Huang et al. 2021]. Huang C-G, Huang H-Z, Li Y-F, Peng W (2021) A novel deep convolutional neural network-bootstrap integrated method for RUL prediction of rolling bearing. Journal of Manufacturing Systems 61:757–772. doi:10.1016/j.jmsy.2021.03.012
21. [Institute of Electrical and Electronics Engineers und IEEE Control Systems Society 2002]. Institute of Electrical and Electronics Engineers, IEEE Control Systems Society (2002) Comparison of Riemann and Lebesgue sampling for first order stochastic systems. IEEE Service Center, Piscataway, NJ

22. [Janiesch et al. 2021]. Janiesch C, Zschech P, Heinrich K (2021) Machine learning and deep learning. Electron Markets 31(3):685–695. doi:10.1007/s12525-021-00475-2
23. [Jauregui Correa und Lozano Guzman 2020]. Jauregui Correa JCA, Lozano Guzman AA (2020) Guidelines for the implementation of a predictive maintenance program Mechanical Vibrations and Condition Monitoring. Elsevier, S 133–145
24. [Javaid et al. 2021]. Javaid M, Haleem A, Singh RP, Rab S, Suman R (2021) Significance of sensors for industry 4.0: Roles, capabilities, and applications. Sensors International 2:100110. doi:10.1016/j.sintl.2021.100110
25. [Jia et al. 2018]. Jia F, Lei Y, Guo L, Lin J, Xing S (2018) A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines. Neurocomputing 272:619–628. doi:10.1016/j.neucom.2017.07.032
26. [Jieyang et al. 2023]. Jieyang P, Kimmig A, Dongkun W, Niu Z, Zhi F, Jiahai W, Liu X, Ovtcharova J (2023) A systematic review of data-driven approaches to fault diagnosis and early warning. J Intell Manuf 34(8):3277–3304. doi:10.1007/s10845-022-02020-0
27. [Kumar et al. 2018]. Kumar S, Goyal D, Dang RK, Dhami SS, Pabla BS (2018) Condition based maintenance of bearings and gears for fault detection – A review. Materials Today: Proceedings 5(2):6128–6137. doi:10.1016/j.matpr.2017.12.219
28. [Laney Doug 2001]. Laney Doug (2001) 3-D Data Management: Controlling Data Volume, Velocity and Variety
29. [Lazarova-Molnar und Mohamed 2019]. Lazarova-Molnar S, Mohamed N (2019) Reliability Assessment in the Context of Industry 4.0: Data as a Game Changer. Procedia Computer Science 151:691–698. doi:10.1016/j.procs.2019.04.092
30. [Lee et al. 2015]. Lee J, Ardakani HD, Yang S, Bagheri B (2015) Industrial Big Data Analytics and Cyber-physical Systems for Future Maintenance & Service Innovation. Procedia CIRP 38:3–7. doi:10.1016/j.procir.2015.08.026
31. [Lee et al. 2019]. Lee C-Y, Huang T-S, Liu M-K, Lan C-Y (2019) Data Science for Vibration Heteroscedasticity and Predictive Maintenance of Rotary Bearings. Energies 12(5):801. doi:10.3390/en12050801
32. [Liang et al. 2024]. Liang J, Liang Q, Wu Z, Chen H, Zhang S, Jiang F (2024) A Novel Unsupervised Deep Transfer Learning Method With Isolation Forest for Machine Fault Diagnosis. IEEE Trans. Ind. Inf. 20(1):235–246. doi:10.1109/TII.2023.3258966

33. [Madakam und Uchiya 2019]. Madakam S, Uchiya T (2019) Industrial Internet of Things (IIoT). Principles, processes and protocols The Internet of Things in the industrial sector. Springer Nature Switzerland AG, 2019, Cham
34. [Montero Jimenez et al. 2020]. Montero Jimenez JJ, Schwartz S, Vingerhoeds R, Grabot B, Salaün M (2020) Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. Journal of Manufacturing Systems 56:539–557. doi:10.1016/j.jmsy.2020.07.008
35. [Murshed et al. 2022]. Murshed MGS, Murphy C, Hou D, Khan N, Ananthanarayanan G, Hussain F (2022) Machine Learning at the Network Edge: A Survey. ACM Comput. Surv. 54(8):1–37. doi:10.1145/3469029
36. [Naipeng Li und Yaguolei 2015]. Naipeng Li, Yaguolei (2015) An Improved Exponential Model for Predicting Remaining Useful Life of Rolling Element Bearings. IEEE Transactions on Industrial Electronics 62(12):7762–7773
37. [Niu et al. 2022]. Niu G, Wang X, Liu E, Zhang B (2022) Lebesgue Sampling Based Deep Belief Network for Lithium-Ion Battery Diagnosis and Prognosis. IEEE Trans. Ind. Electron. 69(8):8481–8490. doi:10.1109/TIE.2021.3109527
38. [Office et al. 2023]. Office JE, Gebraeel N, Lei Y, Li N, Si X, Zio E (2023) Prognostics and Remaining Useful Life Prediction of Machinery: Advances, Opportunities and Challenges. JDMD. doi:10.37965/jdmd.2023.148
39. [Pinciroli et al. 2023]. Pinciroli L, Baraldi P, Zio E (2023) Maintenance optimization in industry 4.0. Reliability Engineering & System Safety 234:109204. doi:10.1016/j.ress.2023.109204
40. [Pistorius 2020]. Pistorius J (2020) Industrie 4.0 – Schlüsseltechnologien für die Produktion. Springer Berlin Heidelberg, Berlin, Heidelberg
41. [Qiu et al. 2023]. Qiu S, Cui X, Ping Z, Shan N, Li Z, Bao X, Xu X (2023) Deep Learning Techniques in Intelligent Fault Diagnosis and Prognosis for Industrial Systems: A Review. Sensors (Basel) 23(3). doi:10.3390/s23031305
42. [Ran et al. 2019/12/12]. Ran Y, Zhou X, Lin P, Wen Y, Deng R (2019) A Survey of Predictive Maintenance: Systems, Purposes and Approaches
43. [Reyes et al. 2018]. Reyes C, Jaramillo F, Zhang B, Kulkarni C, Orchard M (2018) Just-in-time Point Prediction Using a Computationally-efficient Lebesgue-sampling-based Prognostic Method. PHM_CONF 10(1). doi:10.36001/phmconf.2018.v10i1.484
44. [Saxena et al. 2009]. Saxena A, Goebel K, Simon D, Eklund N (2009) Damage propagation modeling for aircraft engine run-to-failure simulation. In: Engineers

IOEAE (Hrsg) 2008 international conference on prognostics and health management (phm). John Wiley, [Place of publication not identified], S 1–9

45. [Schaeffler Technologies AG & Co. KG und Vereinigte Fachverlage GmbH 2015]. Schaeffler Technologies AG & Co. KG, Vereinigte Fachverlage GmbH (2015) Wälzlagerpraxis. Handbuch zur Gestaltung und Berechnung von Wälzlagerungen, 4. Aufl. Vereinigte Fachverl., Mainz
46. [Schütze und Helwig 2017]. Schütze A, Helwig N (2017) Sensorik und Messtechnik für die Industrie 4.0. tm - Technisches Messen 84(5):310–319. doi:10.1515/teme-2016-0047
47. [Schwendemann et al. 2021]. Schwendemann S, Amjad Z, Sikora A (2021) A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines. Computers in Industry 125:103380. doi:10.1016/j.compind.2020.103380
48. [Schwendemann et al. 2021]. Schwendemann S, Amjad Z, Sikora A (2021) A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines. Computers in Industry 125:103380. doi:10.1016/j.compind.2020.103380
49. [Sessa et al. 2022]. Sessa MR, Malandrino O, Fenza G, Caminale G, Risso C (2022) The Adoption of Industry 4.0 Technologies Through the Implementation of Continuous Improvement Tools. In: Hussain CM (Hrsg) Handbook of Smart Materials, Technologies, and Devices : Applications of Industry 4. 0. Springer International Publishing AG, Cham, S 1185–1220
50. [Wang et al. 2020]. Wang B, Lei Y, Li N, Li N (2020) A Hybrid Prognostics Approach for Estimating Remaining Useful Life of Rolling Element Bearings. IEEE Trans. Rel. 69(1):401–412. doi:10.1109/TR.2018.2882682
51. [Wikibon Research 2015]. Wikibon Research (2015) The Vital Role of Edge Computing in the Internet of Things - Wikibon Research. https://wikibon.com/the-vital-role-of-edge-computing-in-the-internet-of-things/. Zugegriffen: 07. Dezember 2023
52. [Yan et al. 2017]. Yan W, Zhang B, Dou W, Liu D, Peng Y (2017) Low-Cost Adaptive Lebesgue Sampling Particle Filtering Approach for Real-Time Li-Ion Battery Diagnosis and Prognosis. IEEE Trans. Automat. Sci. Eng. 14(4):1601–1611. doi:10.1109/TASE.2017.2666202
53. [Yan et al. 2017]. Yan W, Zhang B, Dou W, Liu D, Peng Y (2017) Low-Cost Adaptive Lebesgue Sampling Particle Filtering Approach for Real-Time Li-Ion

Battery Diagnosis and Prognosis. IEEE Trans. Automat. Sci. Eng. 14(4):1601–1611. doi:10.1109/TASE.2017.2666202

54. [Yan et al. 2019]. Yan W, Zhang B, Zhao G, Tang S, Niu G, Wang X (2019) A Battery Management System With a Lebesgue-Sampling-Based Extended Kalman Filter. IEEE Trans. Ind. Electron. 66(4):3227–3236. doi:10.1109/TIE.2018.2842782
55. [Zhao et al. 2021]. Zhao B, Zhang X, Zhan Z, Wu Q (2021) A robust construction of normalized CNN for online intelligent condition monitoring of rolling bearings considering variable working conditions and sources. Measurement 174:108973. doi:10.1016/j.measurement.2021.108973
56. [Zonta et al. 2020]. Zonta T, Da Costa CA, Da Rosa Righi R, Lima MJ de, Da Trindade ES, Li GP (2020) Predictive maintenance in the Industry 4.0: A systematic literature review. Computers & Industrial Engineering 150:106889. doi:10.1016/j.cie.2020.106889
57. Geisberger E, Broy M (Hrsg) (2012) agendaCPS. Integrierte Forschungsagenda Cyber-Physical Systems. acatech STUDIE, Bd 1. Springer Berlin Heidelberg, Berlin, Heidelberg
58. Reichel J, Müller G, Haeffs J (Hrsg) (2018) Betriebliche Instandhaltung, 2. Aufl. VDI-Buch. Springer Berlin Heidelberg, Berlin, Heidelberg
59. Reinhart G (Hrsg) (2017) Handbuch Industrie 4.0. Geschäftsmodelle, Prozesse, Technik. Hanser, München
60. Schulz T (Hrsg) (2021) Industrie 4.0. Potenziale erkennen und umsetzen, 2. Aufl. Vogel Communications Group, Würzburg
61. Spura C, Fleischer B, Wittel H, Jannasch D (Hrsg) (2023) Roloff/Matek Maschinenelemente. Springer Fachmedien Wiesbaden, Wiesbaden

Anhang

Anhang1 Code für die Fehlerdiagnose von Wälzlagern

In [44]:
```python
import os
import pandas as pd
import numpy as np

# Basis-Ordnerpfad
base_folder_path = r'...\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets'
output_folder_path = r'...\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets'
# Lagergruppen
groups = {
    'Group1': ['Bearing1_1', 'Bearing1_2', 'Bearing1_3', 'Bearing1_5'],
    'Group2': ['Bearing2_1', 'Bearing2_2', 'Bearing2_3', 'Bearing2_4', 'Bearing2_5'],
    'Group3': ['Bearing3_1', 'Bearing3_3', 'Bearing3_4', 'Bearing3_5']
}

# Ordnerpfade für die Gruppen
group_paths = {
    'Group1': os.path.join(base_folder_path, '35Hz12kN'),
    'Group2': os.path.join(base_folder_path, '37.5Hz11kN'),
    'Group3': os.path.join(base_folder_path, '40Hz10kN')
}

output_folder_path = 'output'
if not os.path.exists(output_folder_path):
    os.makedirs(output_folder_path)
# Erstellen eines Wörterbuchs, um die vorverarbeiteten Daten zu speichern
dataset = {}

for group_name, bearings in groups.items():
    group_path = group_paths[group_name]
    for bearing in bearings:
        folder_path = os.path.join(group_path, bearing)
        csv_files = sorted([f for f in os.listdir(folder_path) if f.endswith('.csv')], key=lambda x: int(x.split('.')[0]))
        time_in_minutes = []
        horizontal_amplitude = []
        vertical_amplitude = []

        for file_index, file in enumerate(csv_files, start=1):
            df = pd.read_csv(os.path.join(folder_path, file))
            num_points = len(df)
            current_file_times = np.linspace((file_index - 1) * num_points, file_index * num_points, num_points) / num_points
            time_in_minutes.extend(current_file_times)
            horizontal_amplitude.extend(df.iloc[:, 0])
            vertical_amplitude.extend(df.iloc[:, 1])

        horizontal_amplitude = pd.Series(horizontal_amplitude)
        vertical_amplitude = pd.Series(vertical_amplitude)

        # Erstellen eines Pandas-DataFrames für die Daten
        df = pd.DataFrame({'Time (minutes)': time_in_minutes, 'Horizontal Amplitude': horizontal_amplitude, 'Vertical Amplitude': vertical_amplitude})

        # Speichern der Daten in einer CSV-Datei
        output_file_path = os.path.join('output', f'{bearing}.csv')
        df.to_csv(output_file_path, index=False)

print("Daten gespeichert in CSV-Dateien!")
```

```
Daten gespeichert in CSV-Dateien!
```

In [9]:
```python
# Funktion, um das Sliding-Window-Algorithmus auf horizontale und vertikale Amplituden anzuwenden
def sliding_window_with_time(df, window_size, step_size, statistic):
    result_h = []
    result_v = []
    times = []
    for start in range(0, len(df) - window_size + 1, step_size):
        end = start + window_size
        window = df.iloc[start:end]
        times.append(df['Time (minutes)'].iloc[start])  # Verwenden des Startzeitpunkts des Fensters

        if statistic =='mean':
            result_h.append(window['Horizontal Amplitude'].abs().mean())
            result_v.append(window['Vertical Amplitude'].abs().mean())
        elif statistic =='max':
            result_h.append(window['Horizontal Amplitude'].abs().max())
            result_v.append(window['Vertical Amplitude'].abs().max())
        elif statistic =='min':
            result_h.append(window['Horizontal Amplitude'].abs().min())
            result_v.append(window['Vertical Amplitude'].abs().min())

    result_df = pd.DataFrame({
        'Time (minutes)': times,
        'Horizontal Amplitude': result_h,
        'Vertical Amplitude': result_v
    })
    return result_df

# Pfade
base_folder_path = r'D:\Jupyter\xjtu\Untitled Folder\output'
output_folder_path = r'D:\Jupyter\xjtu\output'

# Erstellen des Ausgabeverzeichnisses, wenn es nicht existiert
if not os.path.exists(output_folder_path):
    os.makedirs(output_folder_path)

# Parameter für das Sliding-Window-Verfahren
window_size = 1000  # Anpassen wie erforderlich
step_size = 500     # Anpassen wie erforderlich
statistic ='mean'  # Ändern auf'max' oder'min' wie erforderlich

# Verarbeiten jedes Lagers
for group_name, bearings in groups.items():
    group_path = group_paths[group_name]
    for bearing in bearings:
        file_path = os.path.join(base_folder_path, f'{bearing}.csv')
```

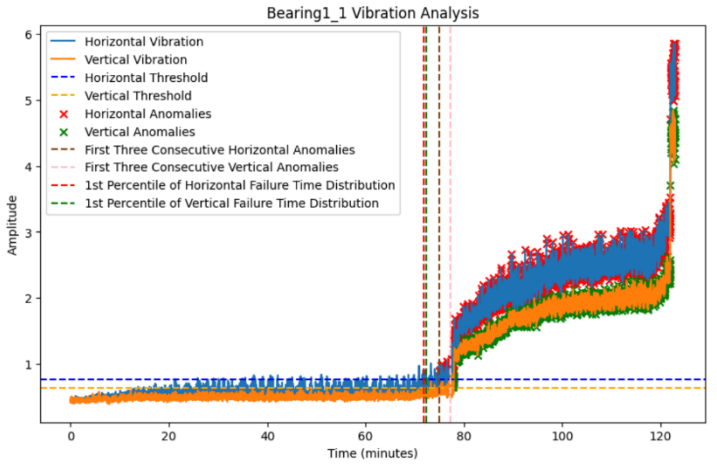
```
        df = pd.read_csv(file_path)

        # Anwenden des Sliding-Window-Algorithmus mit Zeit
        result_df = sliding_window_with_time(df, window_size, step_size, statistic)

        # Speichern der verarbeiteten Daten in einer neuen CSV-Datei
        output_file_path = os.path.join(output_folder_path, f'{bearing}_processed.csv')
        result_df.to_csv(output_file_path, index=False)

print("Sliding-Window-Verarbeitung abgeschlossen und Daten in CSV-Dateien gespeichert!")
```

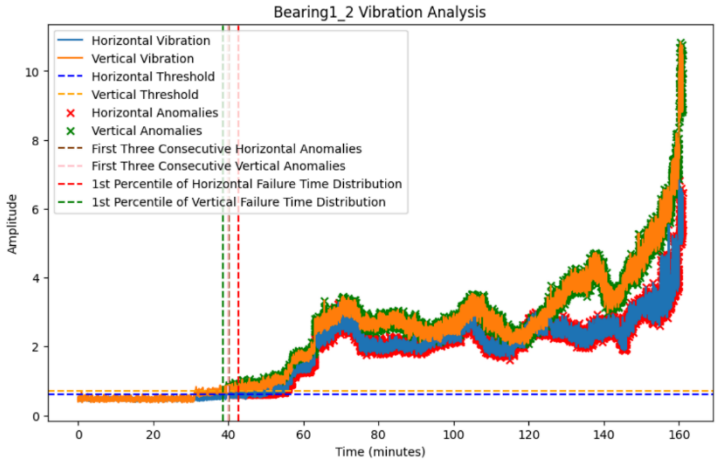
```
Sliding-Window-Verarbeitung abgeschlossen und Daten in CSV-Dateien gespeichert!
```

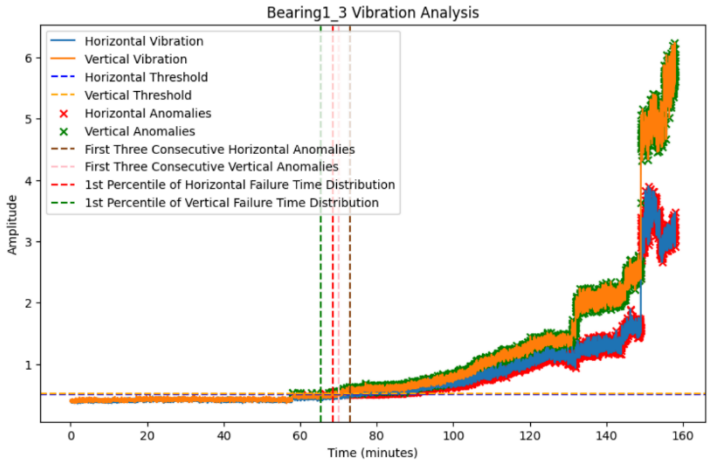
In [61]:

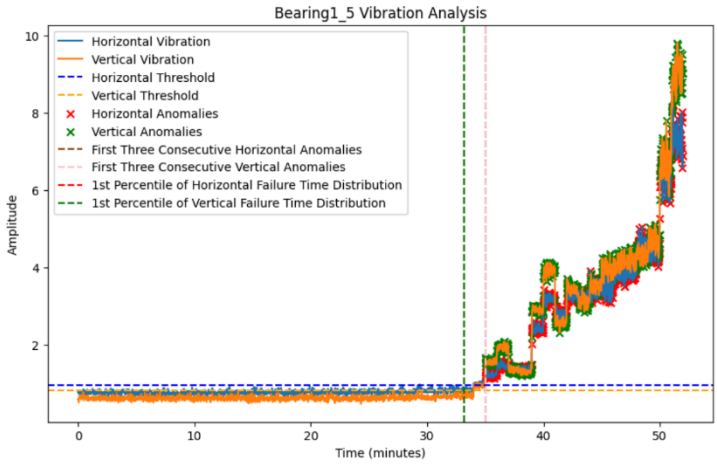
```
import matplotlib.pyplot as plt
from scipy.stats import lognorm
import numpy as np
import pandas as pd
import os

# Definiere den Basisordnerpfad und die Lagergruppen
base_folder_path = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets'

# Setze die normale Betriebsdauer für jedes Lager (in Minuten)
normal_operation_durations = {
    'Bearing1_1': 75,
    'Bearing1_2': 40,
    'Bearing1_3': 50,
    'Bearing1_5': 35,
    'Bearing2_1': 430,
    'Bearing2_2': 40,
    'Bearing2_3': 300,
    'Bearing2_4': 25,
    'Bearing2_5': 100,
    'Bearing3_1': 2000,
    'Bearing3_3': 200,
    'Bearing3_4': 1200,
    'Bearing3_5': 8,
}

# Lade die Daten
dataset = {}
groups = {
    'Group1': ['Bearing1_1', 'Bearing1_2', 'Bearing1_3', 'Bearing1_5'],
    'Group2': ['Bearing2_1', 'Bearing2_2', 'Bearing2_3', 'Bearing2_4', 'Bearing2_5'],
    'Group3': ['Bearing3_1', 'Bearing3_3', 'Bearing3_4', 'Bearing3_5']
}
for group, bearings in groups.items():
    for bearing in bearings:
        file_path = os.path.join(base_folder_path, f'{bearing}_processed.csv')
        dataset[bearing] = pd.read_csv(file_path)

# Funktion zur Erkennung von Anomalien
def detect_anomalies(data, mean, threshold):
    anomalies = np.where(data > mean + threshold)[0]
    return anomalies

# Funktion zum Finden der ersten drei aufeinanderfolgenden Anomalien
def find_first_three_consecutive_anomalies(anomalies):
    for i in range(len(anomalies) - 2):
        if anomalies[i+2] - anomalies[i+1] == 1 and anomalies[i+1] - anomalies[i] == 1:
            return anomalies[i:i+3]
    return []

# Berechne den durchschnittlichen rollenden Standardabweichung für jede Gruppe
group_rolling_std_dev_normal = {}
for group, bearings in groups.items():
    group_rolling_std_dev_normal[group] = {'horizontal': [], 'vertical': []}
    for bearing in bearings:
        time_in_minutes = dataset[bearing]['Time (minutes)']
        horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
        vertical_amplitude = dataset[bearing]['Vertical Amplitude']

        normal_operation_duration = normal_operation_durations[bearing]
        normal_operation_phase = time_in_minutes <= normal_operation_duration
        window_size = 5

        horizontal_std_dev_normal = horizontal_amplitude[normal_operation_phase].rolling(window=window_size).std().mean()
        vertical_std_dev_normal = vertical_amplitude[normal_operation_phase].rolling(window=window_size).std().mean()

        group_rolling_std_dev_normal[group]['horizontal'].append(horizontal_std_dev_normal)
        group_rolling_std_dev_normal[group]['vertical'].append(vertical_std_dev_normal)

    group_rolling_std_dev_normal[group]['horizontal'] = np.mean(group_rolling_std_dev_normal[group]['horizontal'])
    group_rolling_std_dev_normal[group]['vertical'] = np.mean(group_rolling_std_dev_normal[group]['vertical'])

# Liste zur Speicherung der Ergebnisse
results = []

# Hauptanalyse-Schleife
for group, bearings in groups.items():
    for bearing in bearings:
        time_in_minutes = dataset[bearing]['Time (minutes)']
        horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
        vertical_amplitude = dataset[bearing]['Vertical Amplitude']

        # Nutzung der normalen Betriebszeit des Lagers
        normal_operation_duration = normal_operation_durations[bearing]
        normal_operation_phase = time_in_minutes <= normal_operation_duration

        # Berechnung des Durchschnitts und der Standardabweichung während der normalen Betriebsphase
        horizontal_mean_normal = horizontal_amplitude[normal_operation_phase].mean()
        vertical_mean_normal = vertical_amplitude[normal_operation_phase].mean()

        # Berechnung der rollenden Standardabweichung für das Lager
        horizontal_std_dev = horizontal_amplitude[normal_operation_phase].rolling(window=500).std().mean()
        vertical_std_dev = vertical_amplitude[normal_operation_phase].rolling(window=500).std().mean()
```

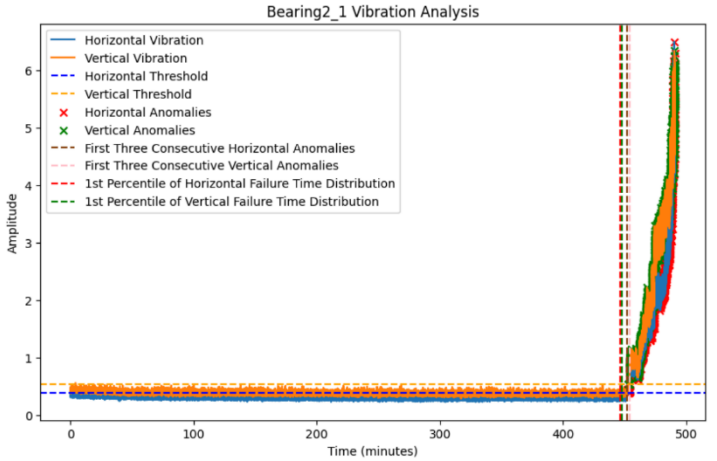
```
        # Integrieren der Anpassung des Schwellenwerts aus dem zweiten Algorithmus
        adj_horizontal_threshold = (horizontal_std_dev + (group_rolling_std_dev_normal[group]['horizontal'] - horizontal_std_dev) * 0.4) * 4
        adj_vertical_threshold = (vertical_std_dev + (group_rolling_std_dev_normal[group]['vertical'] - vertical_std_dev) * 0.4) * 4

        # Erkennen von Anomalien mit dem angepassten Schwellenwert
        horizontal_anomalies = detect_anomalies(horizontal_amplitude, horizontal_mean_normal, adj_horizontal_threshold)
        vertical_anomalies = detect_anomalies(vertical_amplitude, vertical_mean_normal, adj_vertical_threshold)

        threshold_time = normal_operation_duration * 1
        horizontal_anomalies = horizontal_anomalies[time_in_minutes[horizontal_anomalies] > threshold_time]
        vertical_anomalies = vertical_anomalies[time_in_minutes[vertical_anomalies] > threshold_time]

        # Finden der ersten drei aufeinanderfolgenden Anomalien
        first_three_horizontal_anomalies = find_first_three_consecutive_anomalies(horizontal_anomalies)
        first_three_vertical_anomalies = find_first_three_consecutive_anomalies(vertical_anomalies)

        # Berechnung der lognorm Verteilung der Ausfallzeiten und Ermittlung des 1%-Perzentils
        horizontal_failure_times = time_in_minutes[horizontal_anomalies]
        vertical_failure_times = time_in_minutes[vertical_anomalies]

        if len(horizontal_failure_times) > 0:
            shape_horizontal, _, scale_horizontal = lognorm.fit(horizontal_failure_times, floc=0)
            horizontal_percentile_1 = lognorm.ppf(0.01, s=shape_horizontal, scale=scale_horizontal)
        else:
            horizontal_percentile_1 = np.nan

        if len(vertical_failure_times) > 0:
            shape_vertical, _, scale_vertical = lognorm.fit(vertical_failure_times, floc=0)
            vertical_percentile_1 = lognorm.ppf(0.01, s=shape_vertical, scale=scale_vertical)
        else:
            vertical_percentile_1 = np.nan

        # Zeichne und speichere den Plot für jedes Lager
        plt.figure(figsize=(10, 6))
        plt.plot(time_in_minutes, horizontal_amplitude, label='Horizontal Vibration')
        plt.plot(time_in_minutes, vertical_amplitude, label='Vertical Vibration')

        # Hinzufügen von horizontalen Schwellenwertlinien (Mittelwert + Schwellenwert)
        plt.axhline(y=horizontal_mean_normal + adj_horizontal_threshold, color='blue', linestyle='--', label='Horizontal Threshold')
        plt.axhline(y=vertical_mean_normal + adj_vertical_threshold, color='orange', linestyle='--', label='Vertical Threshold')

        # Markierung von Anomalie-Punkten
        plt.scatter(time_in_minutes[horizontal_anomalies], horizontal_amplitude[horizontal_anomalies], c='red', marker='x', label='Horizontal Anomalies'
        plt.scatter(time_in_minutes[vertical_anomalies], vertical_amplitude[vertical_anomalies], c='green', marker='x', label='Vertical Anomalies')

        # Hinzufügen von Linien, die die ersten drei aufeinanderfolgenden Fehlerpunkte verbinden
        if len(first_three_horizontal_anomalies) == 3:
            plt.axvline(x=round(time_in_minutes[first_three_horizontal_anomalies[0]], 2), color='#8B4513', linestyle='--', label='First Three Consecutiv
        if len(first_three_vertical_anomalies) == 3:
            plt.axvline(x=round(time_in_minutes[first_three_vertical_anomalies[0]], 2), color='#FFC0CB', linestyle='--', label='First Three Consecutive

        # Hinzufügen der 1%-Perzentillinie der lognorm Verteilung
        if not np.isnan(horizontal_percentile_1):
            plt.axvline(x=round(horizontal_percentile_1, 2), color='r', linestyle='--', label='1st Percentile of Horizontal Failure Time Distribution')
        if not np.isnan(vertical_percentile_1):
            plt.axvline(x=round(vertical_percentile_1, 2), color='g', linestyle='--', label='1st Percentile of Vertical Failure Time Distribution')

        plt.xlabel('Time (minutes)')
        plt.ylabel('Amplitude')
        plt.title(f'{bearing} Vibration Analysis')
        plt.legend()

        # Speichern des Plots
        output_folder_path = 'individual_bearing_plots'
        os.makedirs(output_folder_path, exist_ok=True)
        plt.savefig(os.path.join(output_folder_path, f'{bearing}_vibration_analysis.png'))

        # Zeigt das gespeicherte Plot an
        plt.show()

        # Schließt den aktuellen Plot, um für den nächsten Plot Platz zu schaffen
        plt.close()

        # Speichern der Ergebnisse
        results.append({
            'Bearing': bearing,
            'Time': time_in_minutes.tolist(),
            'Horizontal Amplitude': horizontal_amplitude.tolist(),
            'Vertical Amplitude': vertical_amplitude.tolist(),
            'Horizontal Threshold': horizontal_mean_normal + adj_horizontal_threshold,
            'Vertical Threshold': vertical_mean_normal + adj_vertical_threshold,
            'Horizontal Anomalies': time_in_minutes[horizontal_anomalies].tolist(),
            'Vertical Anomalies': time_in_minutes[vertical_anomalies].tolist(),
            'First Horizontal Anomalies': time_in_minutes[first_three_horizontal_anomalies].tolist() if len(first_three_horizontal_anomalies) == 3 else
            'First Vertical Anomalies': time_in_minutes[first_three_vertical_anomalies].tolist() if len(first_three_vertical_anomalies) == 3 else np.nan
            '1% Horizontal Failure Time': horizontal_percentile_1,
            '1% Vertical Failure Time': vertical_percentile_1
        })

# Konvertiere Ergebnisse in ein DataFrame und speichere sie in eine CSV-Datei
results_df = pd.DataFrame(results)
output_csv_path = 'bearing_analysis_results.csv'
results_df.to_csv(output_csv_path, index=False)
print(f"Ergebnisse gespeichert unter '{output_csv_path}'.")
```

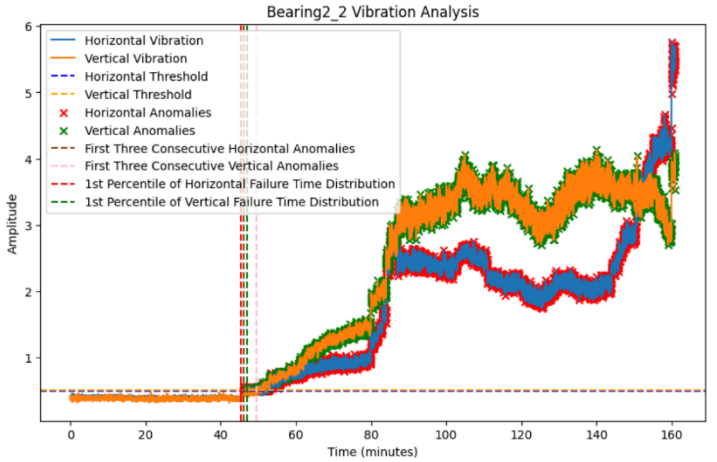

Bearing1_1 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

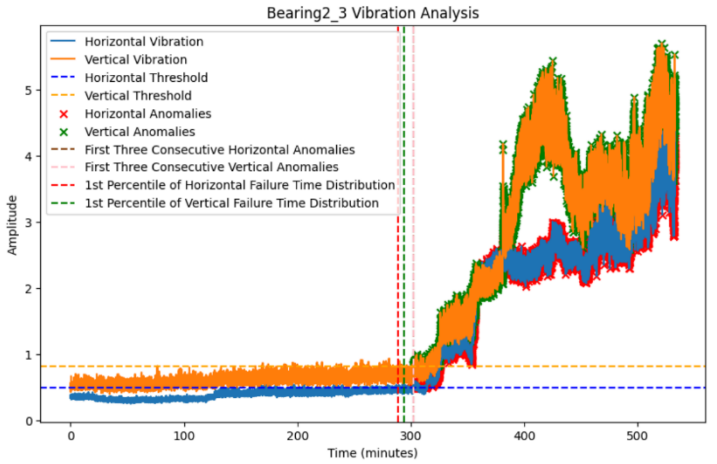

Bearing1_2 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

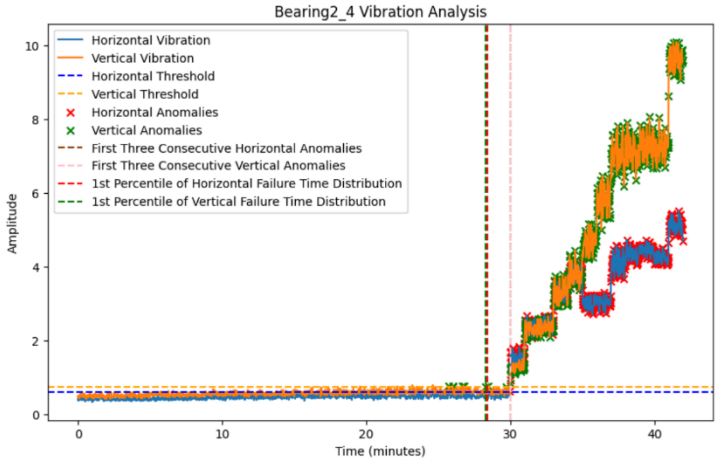

Bearing1_3 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

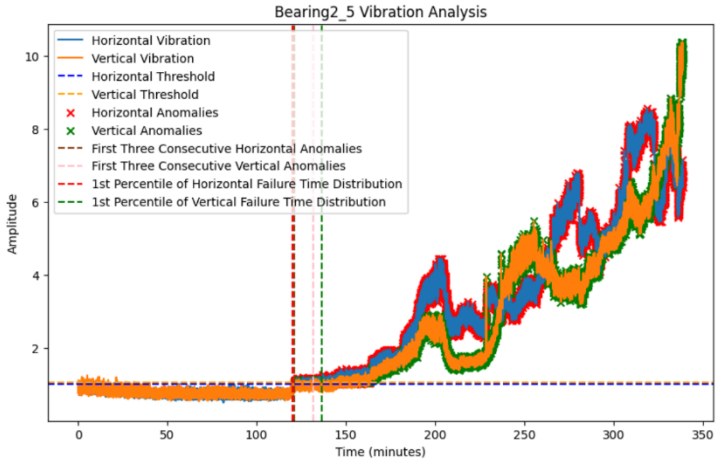

Bearing1_5 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

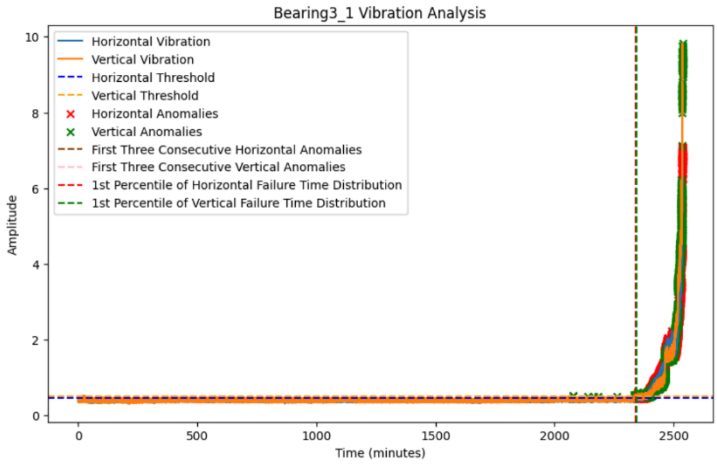

Bearing2_1 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

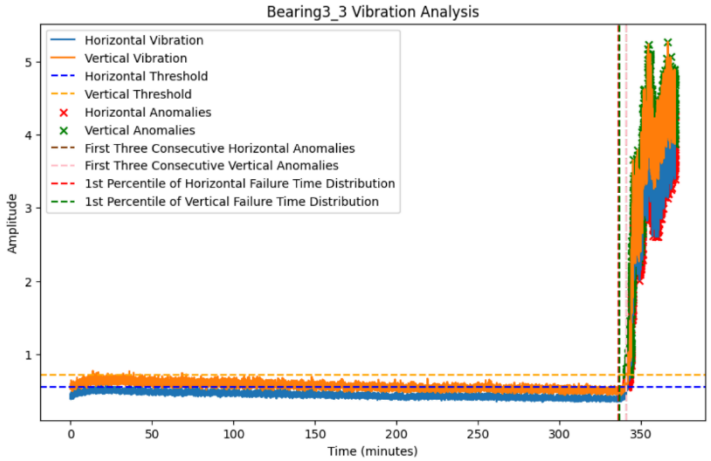

Bearing2_2 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

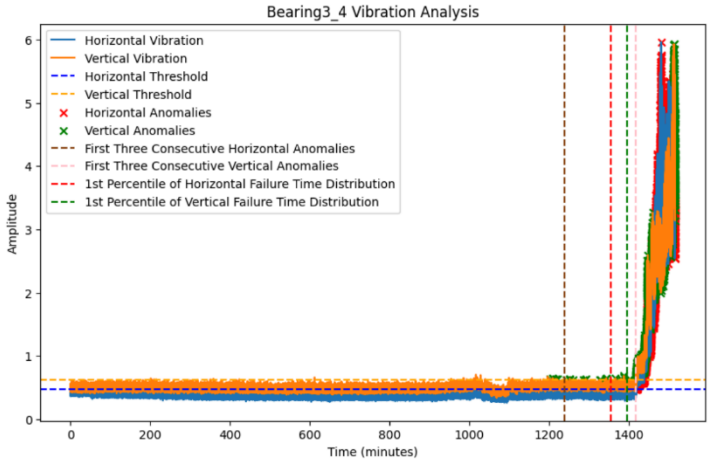

Bearing2_3 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)

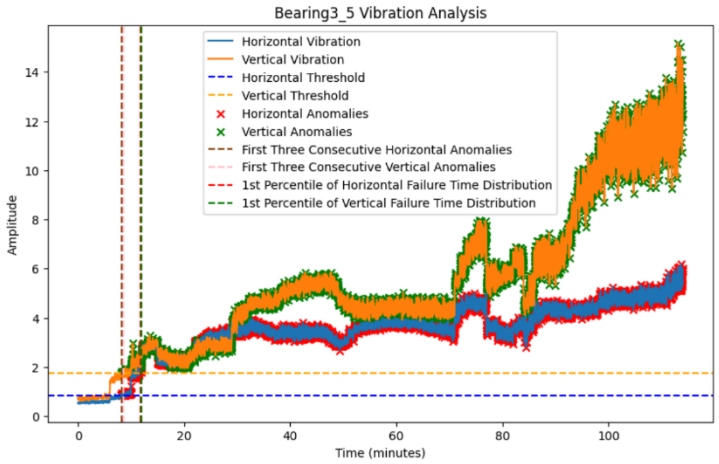

Bearing2_4 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)


Bearing2_5 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)


Bearing3_1 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)


Bearing3_3 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)


Bearing3_4 Vibration Analysis
Horizontal Vibration
Vertical Vibration
Horizontal Threshold
Vertical Threshold
Horizontal Anomalies
Vertical Anomalies
First Three Consecutive Horizontal Anomalies
First Three Consecutive Vertical Anomalies
1st Percentile of Horizontal Failure Time Distribution
1st Percentile of Vertical Failure Time Distribution
Amplitude
Time (minutes)



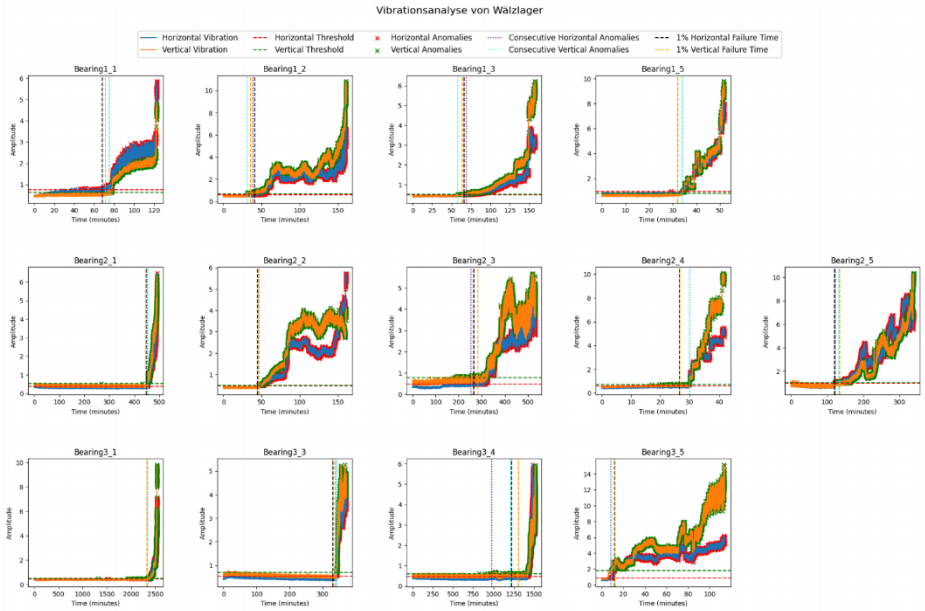

```
Ergebnisse gespeichert unter 'bearing_analysis_results.csv'.
```

In [ ]:

In [93]:
```python
import matplotlib.pyplot as plt
from scipy.stats import lognorm
import numpy as np
import pandas as pd
import os

# Definiere den Basisordnerpfad und die Lagergruppen
base_folder_path = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets'

# Setze die normale Betriebsdauer für jedes Lager (in Minuten)
normal_operation_durations = {
    'Bearing1_1': 75,
    'Bearing1_2': 40,
    'Bearing1_3': 50,
    'Bearing1_5': 35,
    'Bearing2_1': 430,
    'Bearing2_2': 40,
    'Bearing2_3': 300,
    'Bearing2_4': 25,
    'Bearing2_5': 100,
    'Bearing3_1': 2000,
    'Bearing3_3': 200,
    'Bearing3_4': 1200,
    'Bearing3_5': 8,
}

# Lade die Daten
dataset = {}
groups = {
    'Group1': ['Bearing1_1', 'Bearing1_2', 'Bearing1_3', 'Bearing1_5'],
    'Group2': ['Bearing2_1', 'Bearing2_2', 'Bearing2_3', 'Bearing2_4', 'Bearing2_5'],
    'Group3': ['Bearing3_1', 'Bearing3_3', 'Bearing3_4', 'Bearing3_5']
}
for group, bearings in groups.items():
    for bearing in bearings:
        file_path = os.path.join(base_folder_path, f'{bearing}_processed.csv')
        dataset[bearing] = pd.read_csv(file_path)

# Funktion zur Erkennung von Anomalien
def detect_anomalies(data, mean, threshold):
    anomalies = np.where(data > mean + threshold)[0]
    return anomalies

# Funktion zum Finden der ersten drei aufeinanderfolgenden Anomalien
def find_first_three_consecutive_anomalies(anomalies):
    for i in range(len(anomalies) - 2):
        if anomalies[i+2] - anomalies[i+1] == 1 and anomalies[i+1] - anomalies[i] == 1:
            return anomalies[i:i+3]
    return []

# Berechne den durchschnittlichen rollenden Standardabweichung für jede Gruppe
group_rolling_std_dev_normal = {}
for group, bearings in groups.items():
    group_rolling_std_dev_normal[group] = {'horizontal': [], 'vertical': []}
    for bearing in bearings:
        time_in_minutes = dataset[bearing]['Time (minutes)']
        horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
        vertical_amplitude = dataset[bearing]['Vertical Amplitude']

        normal_operation_duration = normal_operation_durations[bearing]
        normal_operation_phase = time_in_minutes <= normal_operation_duration
        window_size = 5

        horizontal_std_dev_normal = horizontal_amplitude[normal_operation_phase].rolling(window=window_size).std().mean()
        vertical_std_dev_normal = vertical_amplitude[normal_operation_phase].rolling(window=window_size).std().mean()

        group_rolling_std_dev_normal[group]['horizontal'].append(horizontal_std_dev_normal)
        group_rolling_std_dev_normal[group]['vertical'].append(vertical_std_dev_normal)
```

```
    group_rolling_std_dev_normal[group]['horizontal'] = np.mean(group_rolling_std_dev_normal[group]['horizontal'])
    group_rolling_std_dev_normal[group]['vertical'] = np.mean(group_rolling_std_dev_normal[group]['vertical'])

# Vorbereitung des Plotbereichs für das Gesamtbild
fig, axes = plt.subplots(3, 5, figsize=(25, 15))  # Anpassung für maximal 5 Lager pro Zeile
plt.subplots_adjust(hspace=0.5, wspace=0.4)
fig.subplots_adjust(top=0.85)  # Platz oben für den Titel lassen

# Platzhalter für die Legenden-Elemente erstellen
legend_elements = []

# Hinzufügen des Titels über der Legende
fig.suptitle('Vibrationsanalyse von Wälzlager', fontsize=16, y=0.95)

results = []
row_indices = {'Group1': 0, 'Group2': 1, 'Group3': 2}

# Hauptanalyse-Schleife
for group, bearings in groups.items():
    for idx, bearing in enumerate(bearings):
        time_in_minutes = dataset[bearing]['Time (minutes)']
        horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
        vertical_amplitude = dataset[bearing]['Vertical Amplitude']

        # Nutzung der normalen Betriebszeit des Lagers
        normal_operation_duration = normal_operation_durations[bearing]
        normal_operation_phase = time_in_minutes <= normal_operation_duration

        # Berechnung des Durchschnitts und der Standardabweichung während der normalen Betriebsphase
        horizontal_mean_normal = horizontal_amplitude[normal_operation_phase].mean()
        vertical_mean_normal = vertical_amplitude[normal_operation_phase].mean()

        # Berechnung der rollenden Standardabweichung für das Lager
        horizontal_std_dev = horizontal_amplitude[normal_operation_phase].rolling(window=500).std().mean()
        vertical_std_dev = vertical_amplitude[normal_operation_phase].rolling(window=500).std().mean()

        # Integrieren der Anpassung des Schwellenwerts aus dem zweiten Algorithmus
        adj_horizontal_threshold = (horizontal_std_dev + (group_rolling_std_dev_normal[group]['horizontal'] - horizontal_std_dev) * 0.4) * 4
        adj_vertical_threshold = (vertical_std_dev + (group_rolling_std_dev_normal[group]['vertical'] - vertical_std_dev) * 0.4) * 4

        # Erkennen von Anomalien mit dem angepassten Schwellenwert
        horizontal_anomalies = detect_anomalies(horizontal_amplitude, horizontal_mean_normal, adj_horizontal_threshold)
        vertical_anomalies = detect_anomalies(vertical_amplitude, vertical_mean_normal, adj_vertical_threshold)

        # Ignoriere Anomalien in den ersten 60% des normalen Degradationsbereichs
        threshold_time = normal_operation_duration * 0.6
        horizontal_anomalies = horizontal_anomalies[time_in_minutes[horizontal_anomalies] > threshold_time]
        vertical_anomalies = vertical_anomalies[time_in_minutes[vertical_anomalies] > threshold_time]

        # Finden der ersten drei aufeinanderfolgenden Anomalien
        first_three_horizontal_anomalies = find_first_three_consecutive_anomalies(horizontal_anomalies)
        first_three_vertical_anomalies = find_first_three_consecutive_anomalies(vertical_anomalies)

        # Berechnung der lognorm Verteilung der Ausfallzeiten und Ermittlung des 1%-Perzentils
        horizontal_failure_times = time_in_minutes[horizontal_anomalies]
        vertical_failure_times = time_in_minutes[vertical_anomalies]

        if len(horizontal_failure_times) > 0:
            shape_horizontal, _, scale_horizontal = lognorm.fit(horizontal_failure_times, floc=0)
            horizontal_percentile_1 = lognorm.ppf(0.01, s=shape_horizontal, scale=scale_horizontal)
        else:
            horizontal_percentile_1 = np.nan

        if len(vertical_failure_times) > 0:
            shape_vertical, _, scale_vertical = lognorm.fit(vertical_failure_times, floc=0)
            vertical_percentile_1 = lognorm.ppf(0.01, s=shape_vertical, scale=scale_vertical)
        else:
            vertical_percentile_1 = np.nan

        # Berechnung der vier Schlüsselzeitpunkte
        degradation_times = []
        if len(horizontal_failure_times) > 0:
            degradation_times.append(horizontal_failure_times.iloc[0])  # Erster horizontaler Ausfallzeitpunkt
        if len(vertical_failure_times) > 0:
            degradation_times.append(vertical_failure_times.iloc[0])  # Erster vertikaler Ausfallzeitpunkt
        if len(first_three_horizontal_anomalies) == 3:
            degradation_times.append(time_in_minutes[first_three_horizontal_anomalies[0]])  # Erstes Set von drei aufeinanderfolgenden horizontalen Anom
        if len(first_three_vertical_anomalies) == 3:
            degradation_times.append(time_in_minutes[first_three_vertical_anomalies[0]])  # Erstes Set von drei aufeinanderfolgenden vertikalen Anomalie
        # Zeitreihen und Schwellenwerte plotten für jeden einzelnen Lager
        plt.figure(figsize=(10, 6))
        plt.plot(time_in_minutes, horizontal_amplitude, label='Horizontal Vibration')
        plt.plot(time_in_minutes, vertical_amplitude, label='Vertical Vibration')
        plt.axhline(y=horizontal_mean_normal + adj_horizontal_threshold, color='r', linestyle='--', label='Horizontal Threshold')
        plt.axhline(y=vertical_mean_normal + adj_vertical_threshold, color='g', linestyle='--', label='Vertical Threshold')

        # Markiere Anomalien mit Streupunkten
        plt.scatter(time_in_minutes[horizontal_anomalies], horizontal_amplitude[horizontal_anomalies], color='red', marker='x', label='Horizontal Anomal
        plt.scatter(time_in_minutes[vertical_anomalies], vertical_amplitude[vertical_anomalies], color='green', marker='x', label='Vertical Anomalies')

        # Plot vertikale Linien für die ersten drei aufeinanderfolgenden Anomalien
        if len(first_three_horizontal_anomalies) == 3:
            for anomaly in first_three_horizontal_anomalies:
                plt.axvline(x=time_in_minutes[anomaly], color='purple', linestyle=':', label='Consecutive Horizontal Anomalies')
        if len(first_three_vertical_anomalies) == 3:
            for anomaly in first_three_vertical_anomalies:
                plt.axvline(x=time_in_minutes[anomaly], color='cyan', linestyle=':', label='Consecutive Vertical Anomalies')

        # Hinzufügen der 1%-Perzentillinie der lognorm Verteilung
        if not np.isnan(horizontal_percentile_1):
            plt.axvline(x=round(horizontal_percentile_1, 2), color='black', linestyle='--', label='1% Horizontal Failure Time')
        if not np.isnan(vertical_percentile_1):
            plt.axvline(x=round(vertical_percentile_1, 2), color='orange', linestyle='--', label='1% Vertical Failure Time')

        plt.xlabel('Time (minutes)')
        plt.ylabel('Amplitude')
        plt.title(f'{bearing} Vibration Analysis')
```

```
        plt.legend()

        # Speichern der Einzelgrafiken
        output_folder_path = r'D:\Jupyter\xjtu\output\individual_plots'
        os.makedirs(output_folder_path, exist_ok=True)
        plt.savefig(os.path.join(output_folder_path, f'{bearing}_vibration_analysis.png'))
        plt.close()

        # Zeitreihen und Schwellenwerte plotten für das Gesamtbild
        ax = axes[row_indices[group], idx % 5]
        line1, = ax.plot(time_in_minutes, horizontal_amplitude, label='Horizontal Vibration')
        line2, = ax.plot(time_in_minutes, vertical_amplitude, label='Vertical Vibration')
        line3 = ax.axhline(y=horizontal_mean_normal + adj_horizontal_threshold, color='r', linestyle='--', label='Horizontal Threshold')
        line4 = ax.axhline(y=vertical_mean_normal + adj_vertical_threshold, color='g', linestyle='--', label='Vertical Threshold')

        # Markiere Anomalien mit Streupunkten
        scatter1 = ax.scatter(time_in_minutes[horizontal_anomalies], horizontal_amplitude[horizontal_anomalies], color='red', marker='x', label='Horizon
        scatter2 = ax.scatter(time_in_minutes[vertical_anomalies], vertical_amplitude[vertical_anomalies], color='green', marker='x', label='Vertical An

        # Plot vertikale Linien für die ersten drei aufeinanderfolgenden Anomalien
        line5 = None
        if len(first_three_horizontal_anomalies) == 3:
            for anomaly in first_three_horizontal_anomalies:
                line5 = ax.axvline(x=time_in_minutes[anomaly], color='purple', linestyle=':', label='Consecutive Horizontal Anomalies')
        line6 = None
        if len(first_three_vertical_anomalies) == 3:
            for anomaly in first_three_vertical_anomalies:
                line6 = ax.axvline(x=time_in_minutes[anomaly], color='cyan', linestyle=':', label='Consecutive Vertical Anomalies')

        # Hinzufügen der 1%-Perzentillinie der lognorm Verteilung
        line7 = None
        if not np.isnan(horizontal_percentile_1):
            line7 = ax.axvline(x=round(horizontal_percentile_1, 2), color='black', linestyle='--', label='1% Horizontal Failure Time')
        line8 = None
        if not np.isnan(vertical_percentile_1):
            line8 = ax.axvline(x=round(vertical_percentile_1, 2), color='orange', linestyle='--', label='1% Vertical Failure Time')

        ax.set_title(f'{bearing}')
        ax.set_xlabel('Time (minutes)')
        ax.set_ylabel('Amplitude')

        # Sammle die ersten Legenden-Elemente einmal
        if idx == 0 and group == 'Group1':
            legend_elements.extend([line1, line2, line3, line4, scatter1, scatter2, line5, line6, line7, line8])

        # Ergebnisse zum Speichern als CSV sammeln
        results.append({
            'Bearing': bearing,
            'Group Average Rolling Std Dev (Horizontal)': horizontal_std_dev,
            'Group Average Rolling Std Dev (Vertical)': vertical_std_dev,
            'Adjusted Horizontal Threshold': adj_horizontal_threshold,
            'Adjusted Vertical Threshold': adj_vertical_threshold
        })

# Verstecke nicht genutzte Subplots
for i in range(3):
    for j in range(len(groups[f'Group{i+1}']), 5):
        axes[i, j].axis('off')

# Erstelle eine separate Legende für das Gesamtbild
fig.legend(handles=[le for le in legend_elements if le is not None], loc='upper center', ncol=5, fontsize=12, bbox_to_anchor=(0.5, 0.92))

# Speichern des Gesamtplots
overall_output_folder_path = r'D:\Jupyter\xjtu\output\overall_plots'
os.makedirs(overall_output_folder_path, exist_ok=True)
plt.savefig(os.path.join(overall_output_folder_path, 'overall_vibration_analysis.png'))

plt.show()

# Definiere den Ausgabepfad und Dateipfad für CSV
output_file_path = os.path.join(overall_output_folder_path, 'Bearing_Vibration_Analysis_Results.csv')

# Speichern der Ergebnisse in eine CSV-Datei
results_df = pd.DataFrame(results)
results_df.to_csv(output_file_path, index=False)
print(f"Ergebnisse gespeichert unter '{output_file_path}'.")
```




```
Ergebnisse gespeichert unter 'D:\Jupyter\xjtu\output\overall_plots\Bearing_Vibration_Analysis_Results.csv'.
```

In [94]:

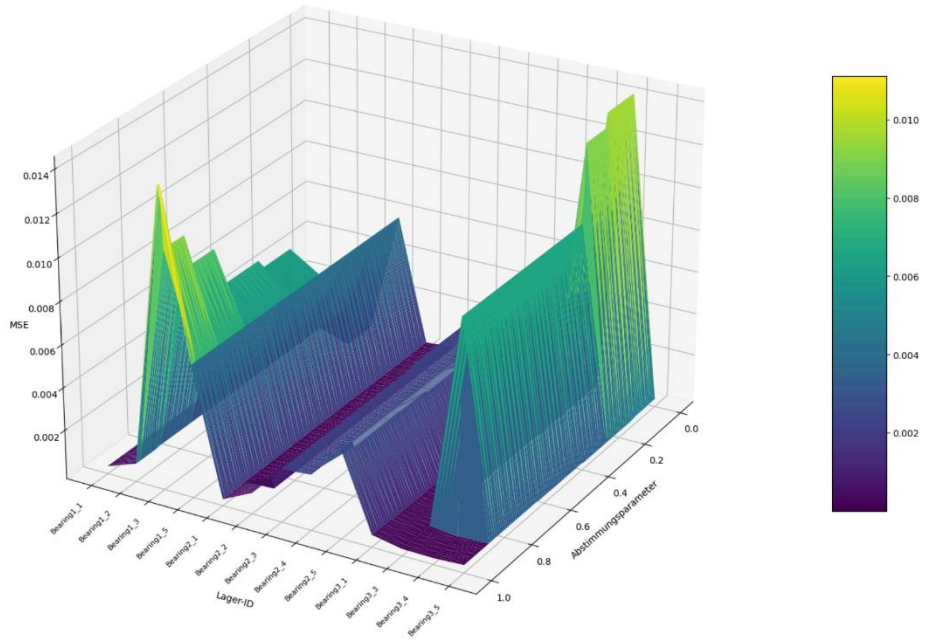
```python
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import lognorm
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

# Define base folder path and bearing groups
base_folder_path = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets'
bearing_groups = {
    'Group1': ['Bearing1_1', 'Bearing1_2', 'Bearing1_3', 'Bearing1_5'],
    'Group2': ['Bearing2_1', 'Bearing2_2', 'Bearing2_3', 'Bearing2_4', 'Bearing2_5'],
    'Group3': ['Bearing3_1', 'Bearing3_3', 'Bearing3_4', 'Bearing3_5']
}

# Set normal operation durations for each bearing (in minutes)
normal_operation_durations = {
    'Bearing1_1': 75,
    'Bearing1_2': 40,
    'Bearing1_3': 50,
    'Bearing1_5': 35,
    'Bearing2_1': 430,
    'Bearing2_2': 40,
    'Bearing2_3': 300,
    'Bearing2_4': 25,
    'Bearing2_5': 100,
    'Bearing3_1': 2000,
    'Bearing3_3': 200,
    'Bearing3_4': 1200,
    'Bearing3_5': 8,
}

# Define failure progression time (FPT)
fpt_data = {
    'Bearing1_1': 76, 'Bearing1_2': 44, 'Bearing1_3': 60, 'Bearing1_5': 39,
    'Bearing2_1': 455, 'Bearing2_2': 48, 'Bearing2_3': 327, 'Bearing2_4': 32, 'Bearing2_5': 141,
    'Bearing3_1': 2344, 'Bearing3_3': 340, 'Bearing3_4': 1418, 'Bearing3_5': 9
}

# Load data
dataset = {}
fpt_normalized = {}
normal_operation_durations_normalized = {}
for group, bearings in bearing_groups.items():
    for bearing in bearings:
        file_path = os.path.join(base_folder_path, f'{bearing}_processed.csv')
        if not os.path.exists(file_path):
            continue  # 如果文件不存在，跳过这个轴承的处理

        dataset[bearing] = pd.read_csv(file_path)

        # 单独归一化时间
        time_min = dataset[bearing]['Time (minutes)'].min()
        time_max = dataset[bearing]['Time (minutes)'].max()
        if time_min != time_max:
            dataset[bearing]['Time (minutes)'] = (dataset[bearing]['Time (minutes)'] - time_min) / (time_max - time_min)
            fpt_normalized[bearing] = (fpt_data[bearing] - time_min) / (time_max - time_min)
            normal_operation_durations_normalized[bearing] = (normal_operation_durations[bearing] - time_min) / (time_max - time_min)
        else:
            fpt_normalized[bearing] = fpt_data[bearing]
```

```
            normal_operation_durations_normalized[bearing] = normal_operation_durations[bearing]

# Anomaliedetektion-Funktion
def detect_anomalies(data, mean, threshold):
    anomalies = np.where(data > mean + threshold)[0]
    return anomalies

# Funktion, um die ersten drei aufeinanderfolgenden Anomalien zu finden
def find_first_three_consecutive_anomalies(anomalies):
    for i in range(len(anomalies) - 2):
        if anomalies[i+2] - anomalies[i+1] == 1 and anomalies[i+1] - anomalies[i] == 1:
            return anomalies[i:i+3]
    return []

# Berechnung der rollenden Standardabweichung für jede Gruppe
group_rolling_std_dev_normal = {}
window_size = 500  # Fenstergröße basierend auf den verfügbaren Daten anpassen

for group, bearings in bearing_groups.items():
    group_rolling_std_dev_normal[group] = {'horizontal': [], 'vertical': []}
    for bearing in bearings:
        time_in_minutes = dataset[bearing]['Time (minutes)']
        horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
        vertical_amplitude = dataset[bearing]['Vertical Amplitude']

        normal_operation_duration = normal_operation_durations_normalized[bearing]
        normal_operation_phase = time_in_minutes <= normal_operation_duration

        horizontal_std_dev_normal = horizontal_amplitude[normal_operation_phase].rolling(window=window_size).std().mean()
        vertical_std_dev_normal = vertical_amplitude[normal_operation_phase].rolling(window=window_size).std().mean()

        group_rolling_std_dev_normal[group]['horizontal'].append(horizontal_std_dev_normal)
        group_rolling_std_dev_normal[group]['vertical'].append(vertical_std_dev_normal)

    group_rolling_std_dev_normal[group]['horizontal'] = np.mean(group_rolling_std_dev_normal[group]['horizontal'])
    group_rolling_std_dev_normal[group]['vertical'] = np.mean(group_rolling_std_dev_normal[group]['vertical'])

# MSE-Berechnungsfunktion
def calculate_mse(actual, predicted):
    return np.mean((actual - predicted) ** 2)

# Ergebnisse generieren
results = []
for percentage in np.linspace(0, 1, 100):
    for group, bearings in bearing_groups.items():
        for bearing in bearings:
            time_in_minutes = dataset[bearing]['Time (minutes)']
            horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
            vertical_amplitude = dataset[bearing]['Vertical Amplitude']

            # Berechnung der normalen Betriebsphase
            normal_operation_duration = normal_operation_durations_normalized[bearing]
            normal_operation_phase = time_in_minutes <= normal_operation_duration

            # Berechnung des Mittelwerts und der Standardabweichung während des normalen Betriebs
            horizontal_mean_normal = horizontal_amplitude[normal_operation_phase].mean()
            vertical_mean_normal = vertical_amplitude[normal_operation_phase].mean()

            horizontal_std_dev = dataset[bearing]['Horizontal Amplitude'][normal_operation_phase].rolling(window=window_size).std().mean()
            vertical_std_dev = dataset[bearing]['Vertical Amplitude'][normal_operation_phase].rolling(window=window_size).std().mean()

            # Dynamisch angepasste Schwellenwerte
            adj_horizontal_threshold = (horizontal_std_dev + (group_rolling_std_dev_normal[group]['horizontal'] - horizontal_std_dev) * percentage) * 4
            adj_vertical_threshold = (vertical_std_dev + (group_rolling_std_dev_normal[group]['vertical'] - vertical_std_dev) * percentage) * 4

            # Anomalien erkennen
            horizontal_anomalies = detect_anomalies(horizontal_amplitude, horizontal_mean_normal, adj_horizontal_threshold)
            vertical_anomalies = detect_anomalies(vertical_amplitude, vertical_mean_normal, adj_vertical_threshold)
            threshold_time = normal_operation_duration
            horizontal_anomalies = horizontal_anomalies[time_in_minutes[horizontal_anomalies] > threshold_time]
            vertical_anomalies = vertical_anomalies[time_in_minutes[vertical_anomalies] > threshold_time]

            # Erste drei aufeinanderfolgende Anomalien finden
            first_three_horizontal_anomalies = find_first_three_consecutive_anomalies(horizontal_anomalies)
            first_three_vertical_anomalies = find_first_three_consecutive_anomalies(vertical_anomalies)

            if len(first_three_horizontal_anomalies) == 3:
                first_horizontal_failure_time = time_in_minutes[first_three_horizontal_anomalies[0]]
            else:
                first_horizontal_failure_time = np.nan

            if len(first_three_vertical_anomalies) == 3:
                first_vertical_failure_time = time_in_minutes[first_three_vertical_anomalies[0]]
            else:
                first_vertical_failure_time = np.nan

            if bearing in fpt_normalized and not np.isnan(fpt_normalized[bearing]):
                actual_fpt = fpt_normalized[bearing]

                if not np.isnan(first_horizontal_failure_time):
                    horizontal_mse = calculate_mse(np.array([actual_fpt]), np.array([first_horizontal_failure_time]))
                    results.append((percentage, bearing, 'Aufeinanderfolgende horizontale Anomalien', horizontal_mse, first_horizontal_failure_time))

                if not np.isnan(first_vertical_failure_time):
                    vertical_mse = calculate_mse(np.array([actual_fpt]), np.array([first_vertical_failure_time]))
                    results.append((percentage, bearing, 'Aufeinanderfolgende vertikale Anomalien', vertical_mse, first_vertical_failure_time))

            # Berechnung des 1%-Perzentils der Ausfallzeit mittels Log-Normal-Verteilung
            horizontal_failure_times = time_in_minutes[horizontal_anomalies]
            vertical_failure_times = time_in_minutes[vertical_anomalies]

            if len(horizontal_failure_times) > 0 and np.all(horizontal_failure_times > 0):
                shape_horizontal, _, scale_horizontal = lognorm.fit(horizontal_failure_times, floc=0)
                horizontal_percentile_1 = lognorm.ppf(0.01, s=shape_horizontal, scale=scale_horizontal)
                if not np.isnan(horizontal_percentile_1):
                    horizontal_mse_1 = calculate_mse(np.array([actual_fpt]), np.array([horizontal_percentile_1]))
                    results.append((percentage, bearing, 'horizontal_1%', horizontal_mse_1, horizontal_percentile_1))
```

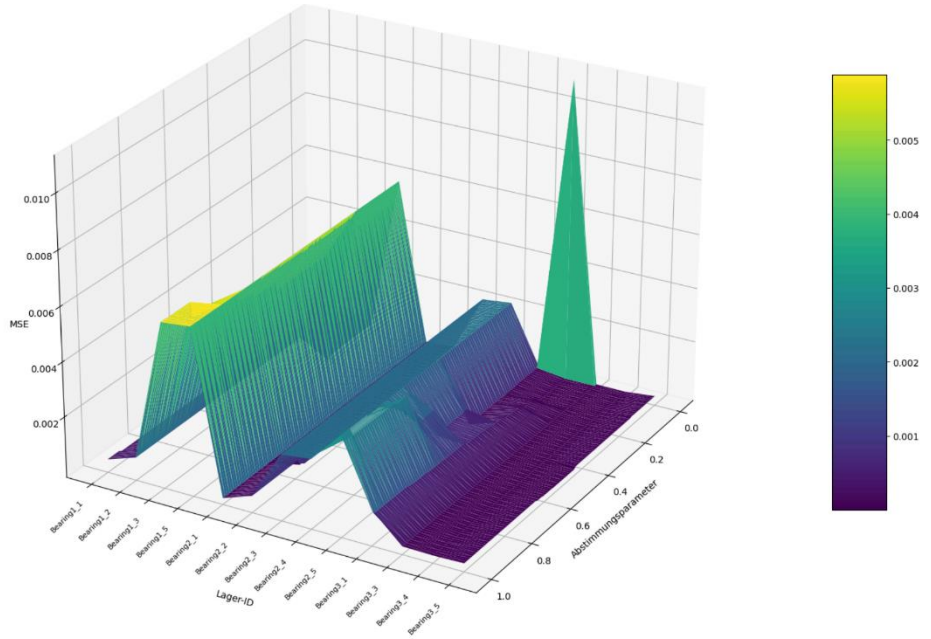
```
            else:
                horizontal_percentile_1 = np.nan

            if len(vertical_failure_times) > 0 and np.all(vertical_failure_times > 0):
                shape_vertical, _, scale_vertical = lognorm.fit(vertical_failure_times, floc=0)
                vertical_percentile_1 = lognorm.ppf(0.01, s=shape_vertical, scale=scale_vertical)
                if not np.isnan(vertical_percentile_1):
                    vertical_mse_1 = calculate_mse(np.array([actual_fpt]), np.array([vertical_percentile_1]))
                    results.append((percentage, bearing, 'vertical_1%', vertical_mse_1, vertical_percentile_1))

# Ergebnisse in DataFrame umwandeln
results_df = pd.DataFrame(results, columns=['Percentage', 'Bearing', 'Method', 'MSE', 'Failure Time'])

# Minimalen MSE für jedes Lager in verschiedenen Methoden finden
min_mse = results_df.loc[results_df.groupby(['Bearing', 'Method'])['MSE'].idxmin()]
min_mse_overall = min_mse.loc[min_mse.groupby('Bearing')['MSE'].idxmin()]

# Ergebnisse in CSV-Datei speichern
output_path_mse = 'bearing_min_mse.csv'
min_mse_overall.to_csv(output_path_mse, index=False)

print(f"Ergebnisse wurden unter '{output_path_mse}' gespeichert.")

# 3D-Oberflächenplot-Funktion
def plot_3d_surface(df, method, metric, ax):
    df_method = df[df['Method'] == method]
    X = df_method['Percentage']
    Y = df_method['Bearing'].astype('category').cat.codes
    Z = df_method[metric]

    surf = ax.plot_trisurf(X, Y, Z, cmap=cm.viridis, linewidth=0.2)
    ax.set_xlabel('Abstimmungsparameter')
    ax.set_ylabel('Lager-ID', labelpad=30)
    ax.set_zlabel(metric)

    ax.set_yticks(range(len(df['Bearing'].unique())))
    ax.set_yticklabels(df['Bearing'].unique(), rotation=45, ha='right', fontsize=8)
    ax.set_title(f'{metric} von {method}')
    ax.view_init(elev=25., azim=30)
    ax.dist = 10

    return surf

# 3D-MSE-Plot erstellen
fig = plt.figure(figsize=(28, 45))

# Plot für MSE erstellen
methods = ['Aufeinanderfolgende horizontale Anomalien', 'Aufeinanderfolgende vertikale Anomalien', 'horizontal_1%','vertical_1%']
for j, method in enumerate(methods):
    ax = fig.add_subplot(4, 1, j + 1, projection='3d')
    surf = plot_3d_surface(results_df, method, 'MSE', ax)
    fig.colorbar(surf, ax=ax, shrink=0.6, aspect=8)

    # Save the figure at each iteration
    fig.savefig(f'mse_plot_{method}.png', bbox_inches='tight')

plt.tight_layout()
plt.show()

# Funktion zum Plotten der minimalen MSE-Werte
def plot_min_points(df, metric, ax):
    X = df['Percentage']
    Y = df['Bearing'].astype('category').cat.codes
    Z = df[metric]

    ax.scatter(X, Y, Z, c='r', marker='o', alpha=0.6)
    ax.set_xlabel('Abstimmungsparameter')
    ax.set_ylabel('Lager-ID', labelpad=35)
    ax.set_zlabel(metric)

    ax.set_yticks(range(len(df['Bearing'].unique())))
    ax.set_yticklabels(df['Bearing'].unique(), rotation=45, ha='right', fontsize=10)
    ax.set_title(f'Minimale {metric}')
    ax.view_init(elev=25., azim=30)
    ax.dist = 10

# 3D-Punkte für minimale MSE plotten
fig = plt.figure(figsize=(28, 15))

ax_mse = fig.add_subplot(111, projection='3d')
plot_min_points(min_mse_overall, 'MSE', ax_mse)

plt.tight_layout()
plt.savefig('min_mse_plot.png', bbox_inches='tight')
plt.show()
```

```
Ergebnisse wurden unter 'bearing_min_mse.csv' gespeichert.
```

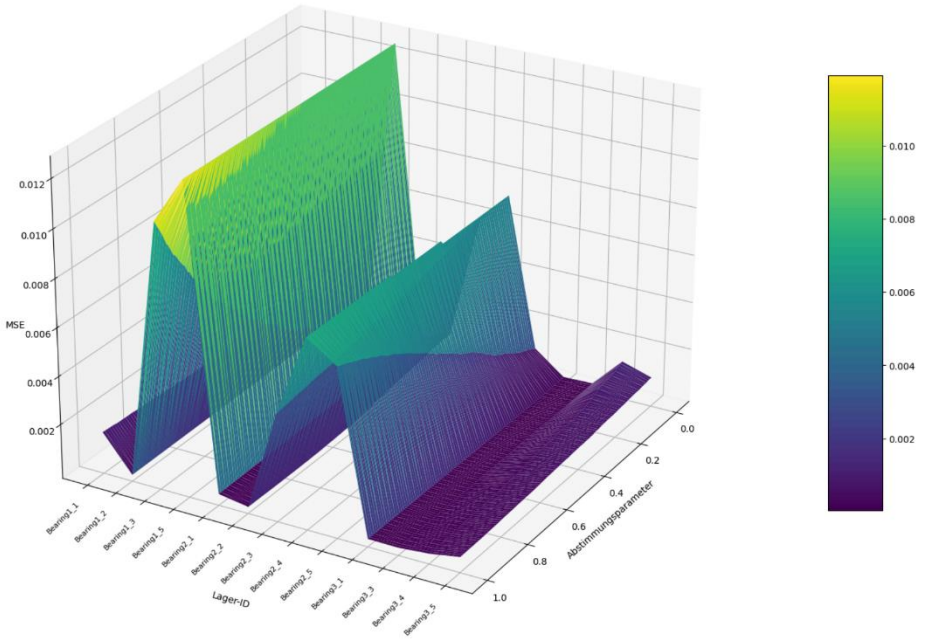
MSE von Aufeinanderfolgende horizontale Anomalien



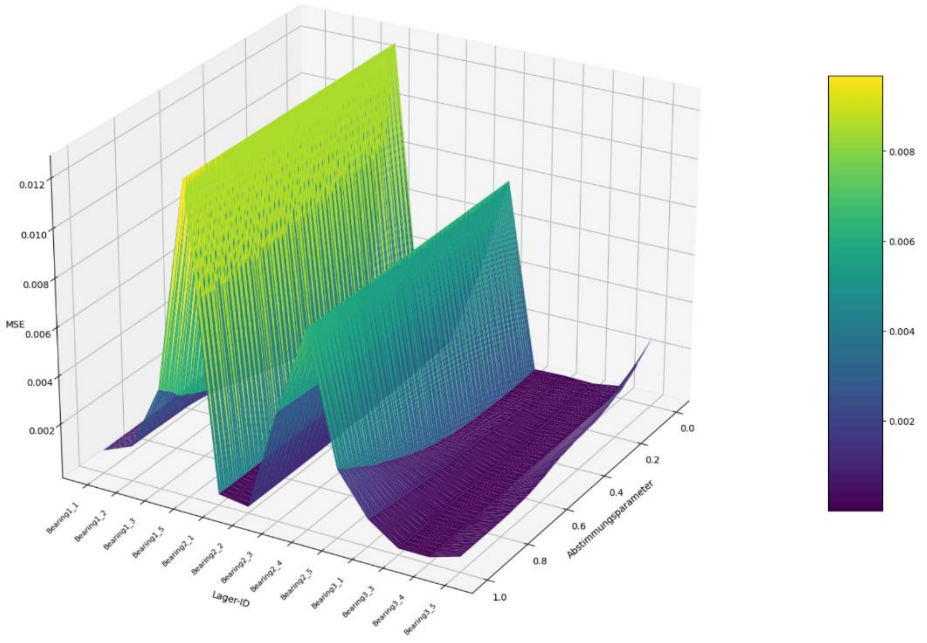
MSE von Aufeinanderfolgende vertikale Anomalien



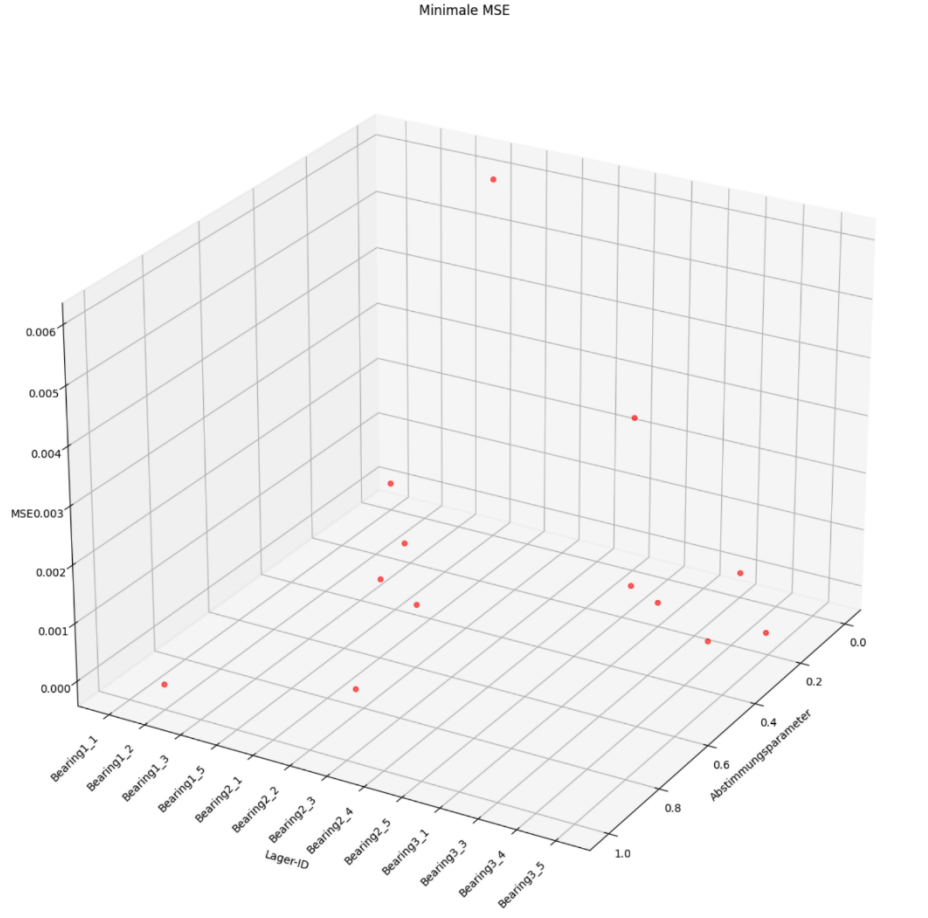
MSE von horizontal_1%



MSE von vertical_1%



```
C:\Users\KZZN\AppData\Local\Temp\ipykernel_34888\3432472702.py:251: UserWarning: Tight layout not applied. The left and right margins cannot be made large enough to accommodate all axes decorations.
  plt.tight_layout()
```



In [96]:

```python
import os
import pandas as pd
import numpy as np
from scipy.stats import kurtosis, skew
from scipy.fft import rfft, rfftfreq
from scipy.signal import find_peaks
from sklearn.linear_model import LinearRegression

# Neue Funktion zur Berechnung von Fehlern für verschiedene Schwellenwertmultiplikatoren und Gruppierung nach Lager
def calculate_errors_for_threshold_per_bearing(multiplier, percentages, dataset, fpt_normalized, bearing_groups, window_size, group_rolling_std_dev_norm
    results = []
    for percentage in percentages:
        for group, bearings in bearing_groups.items():
            for bearing in bearings:
                time_in_minutes = dataset[bearing]['Time (minutes)']
                horizontal_amplitude = dataset[bearing]['Horizontal Amplitude']
                vertical_amplitude = dataset[bearing]['Vertical Amplitude']

                # Bestimmung der normalen Betriebsphase mittels zeitbasierter Methode
                normal_operation_duration = normal_operation_durations_normalized[bearing]
                normal_operation_phase = time_in_minutes <= normal_operation_duration

                # Berechnung des Mittelwerts und der Standardabweichung während der normalen Betriebsphase
                horizontal_mean_normal = horizontal_amplitude[normal_operation_phase].mean()
                vertical_mean_normal = vertical_amplitude[normal_operation_phase].mean()

                horizontal_std_dev = dataset[bearing]['Horizontal Amplitude'][normal_operation_phase].rolling(window=window_size).std().mean()
                vertical_std_dev = dataset[bearing]['Vertical Amplitude'][normal_operation_phase].rolling(window=window_size).std().mean()

                # Angepasste Schwellenwerte mit dynamischer Anpassung
                adj_horizontal_threshold = (horizontal_std_dev + (group_rolling_std_dev_normal[group]['horizontal'] - horizontal_std_dev) * percentage)
                adj_vertical_threshold = (vertical_std_dev + (group_rolling_std_dev_normal[group]['vertical'] - vertical_std_dev) * percentage) * multip

                # Anomalien erkennen, die den Schwellenwert überschreiten
                horizontal_anomalies = detect_anomalies(horizontal_amplitude, horizontal_mean_normal, adj_horizontal_threshold)
```

```
                vertical_anomalies = detect_anomalies(vertical_amplitude, vertical_mean_normal, adj_vertical_threshold)
                threshold_time = normal_operation_duration
                horizontal_anomalies = horizontal_anomalies[time_in_minutes[horizontal_anomalies] > threshold_time]
                vertical_anomalies = vertical_anomalies[time_in_minutes[vertical_anomalies] > threshold_time]

                # Finde die ersten drei aufeinanderfolgenden Anomalien
                first_three_horizontal_anomalies = find_first_three_consecutive_anomalies(horizontal_anomalies)
                first_three_vertical_anomalies = find_first_three_consecutive_anomalies(vertical_anomalies)

                if len(first_three_horizontal_anomalies) == 3:
                    first_horizontal_failure_time = time_in_minutes[first_three_horizontal_anomalies[0]]
                else:
                    first_horizontal_failure_time = np.nan

                if len(first_three_vertical_anomalies) == 3:
                    first_vertical_failure_time = time_in_minutes[first_three_vertical_anomalies[0]]
                else:
                    first_vertical_failure_time = np.nan

                if bearing in fpt_normalized and not np.isnan(fpt_normalized[bearing]):
                    actual_fpt = fpt_normalized[bearing]

                    if not np.isnan(first_horizontal_failure_time):
                        horizontal_mse = calculate_mse(np.array([actual_fpt]), np.array([first_horizontal_failure_time]))
                        results.append((percentage, bearing, 'horizontal', horizontal_mse, first_horizontal_failure_time))

                    if not np.isnan(first_vertical_failure_time):
                        vertical_mse = calculate_mse(np.array([actual_fpt]), np.array([first_vertical_failure_time]))
                        results.append((percentage, bearing, 'vertical', vertical_mse, first_vertical_failure_time))

    return pd.DataFrame(results, columns=['Percentage', 'Bearing', 'Method', 'MSE', 'Failure Time'])

# Ergebnisse für den Multiplikator 3 und 4 generieren
results_3x = calculate_errors_for_threshold_per_bearing(3, percentages, dataset, fpt_normalized, bearing_groups, window_size, group_rolling_std_dev_norma
results_4x = calculate_errors_for_threshold_per_bearing(4, percentages, dataset, fpt_normalized, bearing_groups, window_size, group_rolling_std_dev_norma

# MSE für jedes Lager summieren für 3x und 4x Multiplikator
sum_mse_3x = results_3x.groupby('Bearing')['MSE'].sum().reset_index()
sum_mse_4x = results_4x.groupby('Bearing')['MSE'].sum().reset_index()

# Vergleichs-DataFrame für 3x und 4x MSE-Ergebnisse erstellen
comparison_df_3x = sum_mse_3x.copy()
comparison_df_4x = sum_mse_4x.copy()

# Gesamt-MSE für beide 3x und 4x berechnen
total_mse_3x = sum_mse_3x['MSE'].sum()
total_mse_4x = sum_mse_4x['MSE'].sum()

# Gesamtergebnis zum Vergleichs-DataFrame hinzufügen
comparison_df_3x.loc[len(comparison_df_3x.index)] = ['Total', total_mse_3x]
comparison_df_4x.loc[len(comparison_df_4x.index)] = ['Total', total_mse_4x]

# Ergebnisse ausdrucken
print("Ergebnisse für 3x Multiplikator:")
print(comparison_df_3x)

print("\nErgebnisse für 4x Multiplikator:")
print(comparison_df_4x)
```

```
Ergebnisse für 3x Multiplikator:
       Bearing       MSE
0   Bearing1_1  0.012908
1   Bearing1_2  0.122042
2   Bearing1_3  0.433256
3   Bearing1_5  1.183258
4   Bearing2_1  0.005542
5   Bearing2_2  0.023000
6   Bearing2_3  0.472000
7   Bearing2_4  4.621280
8   Bearing2_5  0.669762
9   Bearing3_1  0.280164
10  Bearing3_3  0.003197
11  Bearing3_4  2.036285
12  Bearing3_5  0.012485
13       Total  9.875179

Ergebnisse für 4x Multiplikator:
       Bearing       MSE
0   Bearing1_1  0.021082
1   Bearing1_2  0.120613
2   Bearing1_3  0.946886
3   Bearing1_5  1.183258
4   Bearing2_1  0.003863
5   Bearing2_2  0.033625
6   Bearing2_3  0.444283
7   Bearing2_4  0.461847
8   Bearing2_5  0.473580
9   Bearing3_1  0.074392
10  Bearing3_3  0.005159
11  Bearing3_4  0.971498
12  Bearing3_5  0.011485
13       Total  4.751571
```

In [98]:

```
combined_results = min_mse_overall.copy()
best_results = combined_results.loc[combined_results.groupby('Bearing')['MSE'].idxmin()]

# Lese die Originaldaten, um den minimalen und maximalen Zeitwert zu erhalten
base_folder_path = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets'
bearing_groups = {
    'Group1': ['Bearing1_1', 'Bearing1_2', 'Bearing1_3', 'Bearing1_5'],
    'Group2': ['Bearing2_1', 'Bearing2_2', 'Bearing2_3', 'Bearing2_4', 'Bearing2_5'],
    'Group3': ['Bearing3_1', 'Bearing3_3', 'Bearing3_4', 'Bearing3_5']
}

time_ranges = {}
for group, bearings in bearing_groups.items():
    for bearing in bearings:
```

```
        file_path = os.path.join(base_folder_path, f'{bearing}_processed.csv')
        if os.path.exists(file_path):
            df = pd.read_csv(file_path)
            time_ranges[bearing] = {
                'min': df['Time (minutes)'].min(),
                'max': df['Time (minutes)'].max()
            }

# Denormalisierungsfunktion
# Diese Funktion wandelt den normalisierten Wert in den ursprünglichen Bereich zurück
def denormalize(normalized_value, min_value, max_value):
    return normalized_value * (max_value - min_value) + min_value

# Denormalisieren und das früheste Ausfallzeit extrahieren
earliest_failure_times = {}
for _, row in best_results.iterrows():
    bearing = row['Bearing']
    normalized_failure_time = row['Failure Time']

    if bearing in time_ranges:
        min_time = time_ranges[bearing]['min']
        max_time = time_ranges[bearing]['max']
        denormalized_failure_time = denormalize(normalized_failure_time, min_time, max_time)
        # Den Wert auf die nächste ganze Zahl abrunden (nach unten)
        earliest_failure_times[bearing] = round(denormalized_failure_time)

# Ergebnisse drucken
print("Früheste Ausfallzeit (Minuten):")
for bearing, failure_time in earliest_failure_times.items():
    print(f"{bearing}: {failure_time:.2f}")

# Ergebnisse in einer CSV-Datei speichern
output_df = pd.DataFrame(list(earliest_failure_times.items()), columns=['Bearing', 'Earliest Failure Time'])
output_df.to_csv('earliest_failure_times.csv', index=False)
print("Die Ergebnisse wurden in 'earliest_failure_times.csv' gespeichert.")
```

```
Früheste Ausfallzeit (Minuten):
Bearing1_1: 75.00
Bearing1_2: 44.00
Bearing1_3: 60.00
Bearing1_5: 35.00
Bearing2_1: 454.00
Bearing2_2: 48.00
Bearing2_3: 302.00
Bearing2_4: 30.00
Bearing2_5: 141.00
Bearing3_1: 2344.00
Bearing3_3: 341.00
Bearing3_4: 1418.00
Bearing3_5: 9.00
Die Ergebnisse wurden in 'earliest_failure_times.csv' gespeichert.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Anhang 2 Code für die Feature-Engineering

In [29]:

```python
import os
import pandas as pd
import numpy as np
from scipy.stats import kurtosis, skew
from scipy.fft import rfft, rfftfreq
from scipy.signal import find_peaks
from sklearn.linear_model import LinearRegression

def calculate_features(signal, window_size=1024):
    if len(signal) < window_size:
        window = signal
    else:
        window = signal[-window_size:]

    time = np.arange(len(window))
    reg = LinearRegression().fit(time.reshape(-1, 1), window)

    fft_result = rfft(window)
    fft_freq = rfftfreq(len(window), d=1/25600)
    fft_amplitudes = np.abs(fft_result)[1:]
    fft_freq = fft_freq[1:]

    if len(fft_amplitudes) > 0:
        sorted_indices = np.argsort(fft_amplitudes)[::-1]
        sorted_freqs = fft_freq[sorted_indices]
        sorted_ampls = fft_amplitudes[sorted_indices]
    else:
        sorted_freqs = np.array([0, 0])
        sorted_ampls = np.array([0, 0])

    peaks, _ = find_peaks(window, distance=100)
    if len(peaks) > 0:
        first_peak = peaks[0]
    else:
        first_peak = 0

    features = {
        'Mse.ts': np.mean((window - reg.predict(time.reshape(-1, 1)))**2),
        'Slope.ts': reg.coef_[0],
        'Intercept.ts': reg.intercept_,
        'Skewness.ts': skew(window),
        'Kurtosis.ts': kurtosis(window),
        'Max.ts': np.max(window),
        'Sd.cp': np.std(window),
        'First-point.cp': first_peak,
        'Skewness.cp': skew(window[:first_peak]) if first_peak > 0 else skew(window),
        'Kurtosis.cp': kurtosis(window[:first_peak]) if first_peak > 0 else kurtosis(window),
        'Ampl1-freq.f': sorted_freqs[0] if len(sorted_freqs) > 0 else 0,
        'Ampl1.f': sorted_ampls[0] if len(sorted_ampls) > 0 else 0,
        'Ampl2-freq.f': sorted_freqs[1] if len(sorted_freqs) > 1 else 0,
        'Ampl2.f': sorted_ampls[1] if len(sorted_ampls) > 1 else 0,
        'Ampl-mean.f': np.mean(fft_amplitudes) if len(fft_amplitudes) > 0 else 0,
        'Ampl-var.f': np.var(fft_amplitudes) if len(fft_amplitudes) > 0 else 0,
        'Ampl-skewness.f': skew(fft_amplitudes) if len(fft_amplitudes) > 0 else 0,
        'Ampl-kurtosis.f': kurtosis(fft_amplitudes) if len(fft_amplitudes) > 0 else 0
    }
    return features

base_folder_path = r'D:\Jupyter\xjtu\rul\output'
output_folder_path = 'output'

all_bearings_features = pd.DataFrame()
test_features = pd.DataFrame()

for bearing_num in [1, 2, 3]:
    file_name = f'Bearing1_{bearing_num}_EPZ.csv'
    file_path = os.path.join(output_folder_path, file_name)

    df = pd.read_csv(file_path)

    horizontal_signal = df['Horizontal Amplitude'].to_numpy()
    vertical_signal = df['Vertical Amplitude'].to_numpy()

    for i in range(0, len(df), 1024):
        horizontal_features = calculate_features(horizontal_signal[:i+1024])
        vertical_features = calculate_features(vertical_signal[:i+1024])

        horizontal_features = {f'{key}.Horizontal': value for key, value in horizontal_features.items()}
        vertical_features = {f'{key}.Vertical': value for key, value in vertical_features.items()}
```

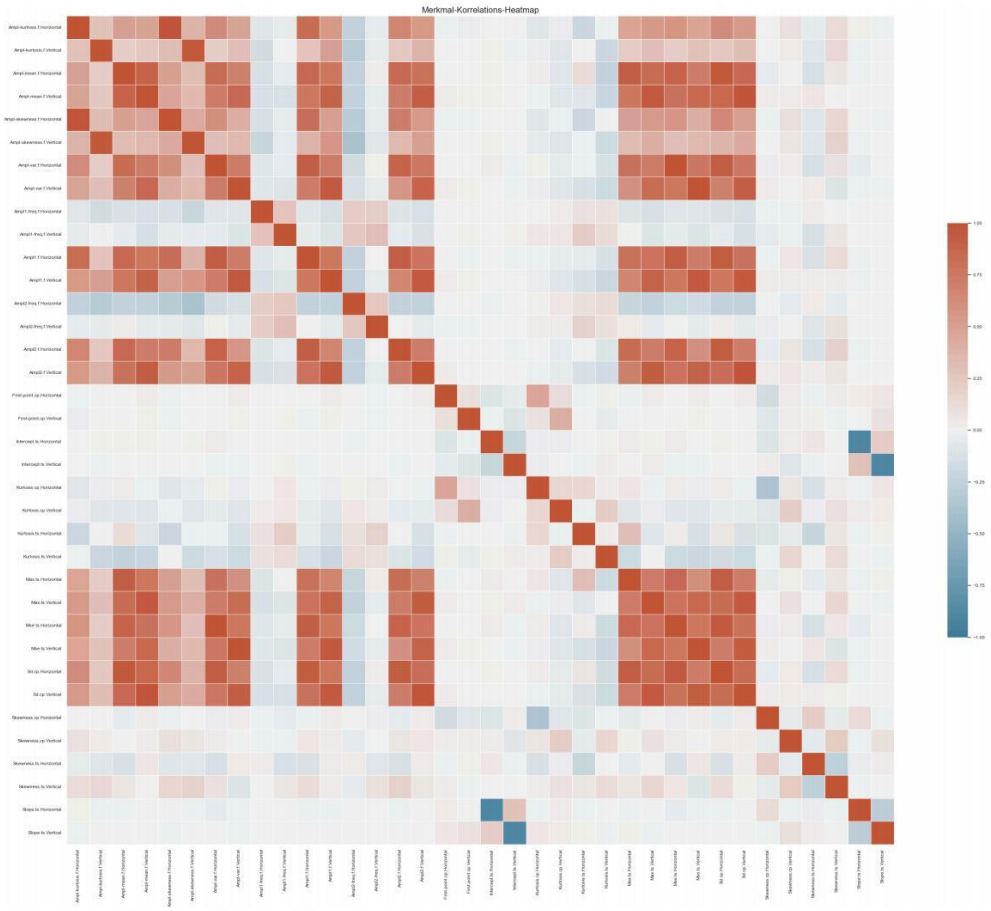
```
        combined_features = {
            'Bearing_ID': f'{bearing_num}',
            'RUL': df['RUL'].iloc[i],
            **horizontal_features,
            **vertical_features
        }
        all_bearings_features = all_bearings_features.append(combined_features, ignore_index=True)


test_file_name = 'Bearing1_5_EPZ.csv'
test_file_path = os.path.join(output_folder_path, test_file_name)

df = pd.read_csv(test_file_path)

horizontal_signal = df['Horizontal Amplitude'].to_numpy()
vertical_signal = df['Vertical Amplitude'].to_numpy()

for i in range(0, len(df), 1024):
    horizontal_features = calculate_features(horizontal_signal[:i+1024])
    vertical_features = calculate_features(vertical_signal[:i+1024])

    horizontal_features = {f'{key}.Horizontal': value for key, value in horizontal_features.items()}
    vertical_features = {f'{key}.Vertical': value for key, value in vertical_features.items()}

    combined_features = {
        'Bearing_ID': '5',
        'RUL': df['RUL'].iloc[i],
        **horizontal_features,
        **vertical_features
    }
    test_features = test_features.append(combined_features, ignore_index=True)


column_order = ['Bearing_ID', 'RUL'] + [col for col in all_bearings_features.columns if col not in ['Bearing_ID',
all_bearings_features = all_bearings_features[column_order]
test_features = test_features[column_order]

all_bearings_features.to_csv(os.path.join(base_folder_path, 'training_validation_features.csv'), index=False)
test_features.to_csv(os.path.join(base_folder_path, 'test_features.csv'), index=False)

print(f"Trainings- und Validierungsmerkmale wurden extrahiert und gespeichert.")
print(f"Testmerkmale wurden extrahiert und gespeichert.")
```

```
Trainings- und Validierungsmerkmale wurden extrahiert und gespeichert.
Testmerkmale wurden extrahiert und gespeichert.
```

In [21]:

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Merkmalsdaten lesen
basis_ordner_pfad = r'D:\Jupyter\xjtu\rul\output'
merkmals_datei = os.path.join(basis_ordner_pfad, 'training_validation_features.csv')

# CSV-Datei einlesen
df = pd.read_csv(merkmals_datei)

# 'Bearing_ID' und 'RUL' Spalten entfernen
df = df.drop(['Bearing_ID', 'RUL'], axis=1)

# Korrelation berechnen
korrelation = df.corr()

# Abbildungsgröße festlegen
plt.figure(figsize=(30, 24))

# Heatmap erstellen
sns.set(font_scale=0.8)  # Schriftgröße anpassen
farbpalette = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(korrelation, cmap=farbpalette, vmax=1, vmin=-1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5},
            annot=False)  # annot=True setzen, um konkrete Korrelationskoeffizienten anzuzeigen

# Titel setzen
plt.title('Merkmal-Korrelations-Heatmap', fontsize=16)

# Layout anpassen
plt.tight_layout()
```

```
# Abbildung anzeigen
plt.show()

# Optional: Abbildung speichern
# plt.savefig('merkmal_korrelations_heatmap.png', dpi=300, bbox_inches='tight')

# Die am stärksten korrelierten Merkmalspaare ausgeben
print("Top 10 am stärksten korrelierte Merkmalspaare:")
korrelations_paare = korrelation.unstack()
sortierte_paare = korrelations_paare.sort_values(kind="quicksort", ascending=False)
print(sortierte_paare[sortierte_paare < 1].head(10))
```



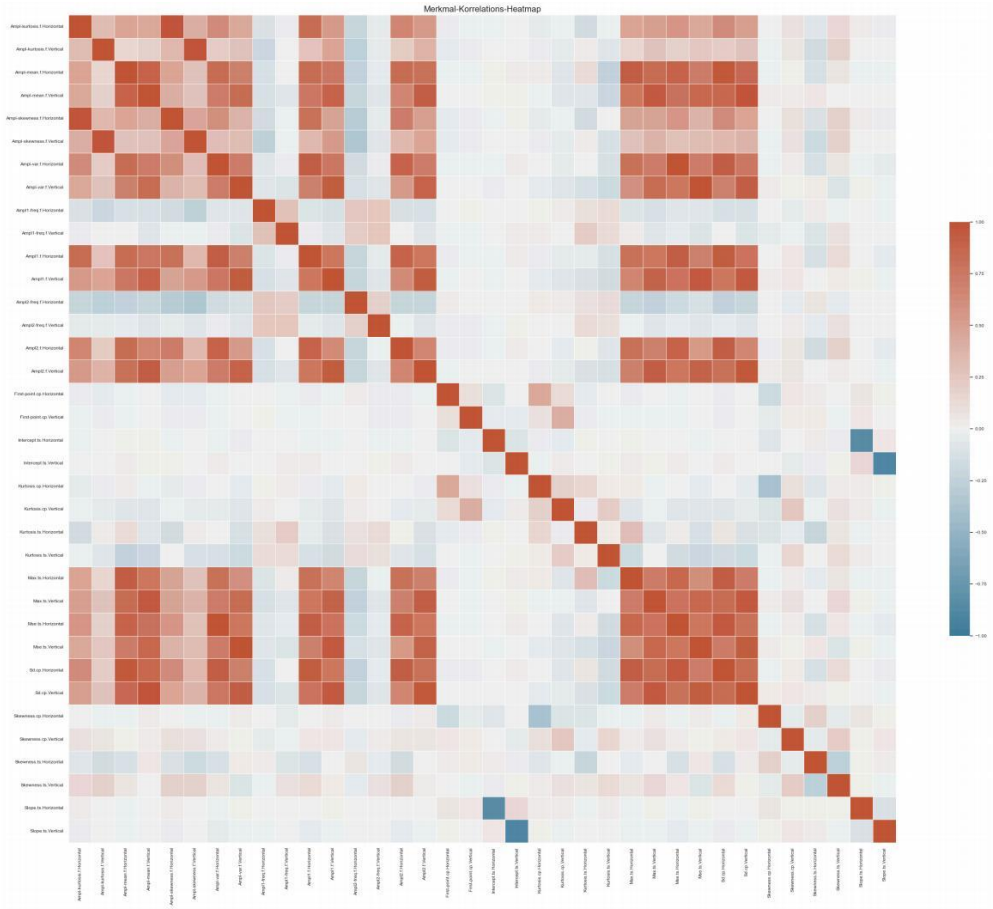

```
Top 10 am stärksten korrelierte Merkmalspaare:
Ampl-var.f.Vertical          Mse.ts.Vertical              0.997832
Mse.ts.Vertical              Ampl-var.f.Vertical          0.997832
Mse.ts.Horizontal            Ampl-var.f.Horizontal        0.986914
Ampl-var.f.Horizontal        Mse.ts.Horizontal            0.986914
Ampl-mean.f.Vertical         Sd.cp.Vertical               0.976608
Sd.cp.Vertical               Ampl-mean.f.Vertical         0.976608
                             Ampl2.f.Vertical             0.972929
Ampl2.f.Vertical             Sd.cp.Vertical               0.972929
Ampl-skewness.f.Horizontal   Ampl-kurtosis.f.Horizontal   0.969336
Ampl-kurtosis.f.Horizontal   Ampl-skewness.f.Horizontal   0.969336
dtype: float64
```

In [22]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Daten laden
basis_ordner_pfad = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets\35Hz12kN'
merkmals_datei = os.path.join(basis_ordner_pfad, 'training_validation_features.csv')
```

```
# CSV-Datei einlesen
df = pd.read_csv(merkmals_datei)

# Entfernen der 'Bearing_ID' und 'RUL'-Spalten
df = df.drop(['Bearing_ID', 'RUL'], axis=1)

# Berechnung der Korrelation
korrelation = df.corr()

# Ausgabe der am stärksten korrelierten Merkmale
print("Top 10 am stärksten korrelierte Merkmale:")
korrelations_paare = korrelation.unstack()
sortierte_paare = korrelations_paare.sort_values(kind="quicksort", ascending=False)
# Vermeidung von Eigenkorrelationen (1.0)
print(sortierte_paare[sortierte_paare < 1].head(10))

# Heatmap erstellen
plt.figure(figsize=(30, 24))
sns.set(font_scale=0.8)  # Schriftgröße anpassen
farbpalette = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(korrelation, cmap=farbpalette, vmax=1, vmin=-1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5},
            annot=False)  # annot=True setzt konkrete Korrelationskoeffizienten

# Titel setzen
plt.title('Merkmal-Korrelations-Heatmap', fontsize=16)

# Layout anpassen und anzeigen
plt.tight_layout()
plt.show()
```

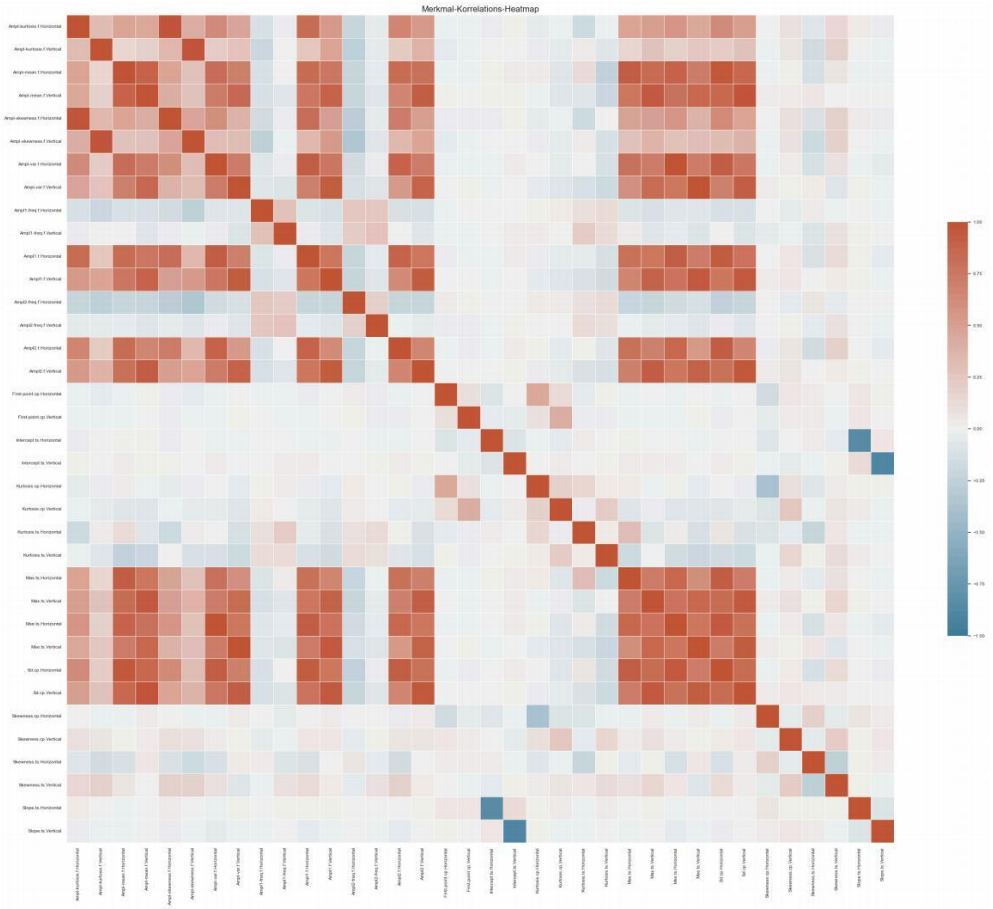
```
Top 10 am stärksten korrelierte Merkmale:
Ampl-var.f.Vertical          Mse.ts.Vertical              0.997811
Mse.ts.Vertical              Ampl-var.f.Vertical          0.997811
Ampl-var.f.Horizontal        Mse.ts.Horizontal            0.985999
Mse.ts.Horizontal            Ampl-var.f.Horizontal        0.985999
Ampl-mean.f.Vertical         Sd.cp.Vertical               0.977421
Sd.cp.Vertical               Ampl-mean.f.Vertical         0.977421
Ampl-skewness.f.Vertical     Ampl-kurtosis.f.Vertical     0.970379
Ampl-kurtosis.f.Vertical     Ampl-skewness.f.Vertical     0.970379
Ampl-kurtosis.f.Horizontal   Ampl-skewness.f.Horizontal   0.969472
Ampl-skewness.f.Horizontal   Ampl-kurtosis.f.Horizontal   0.969472
dtype: float64
```



In [31]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
import warnings

warnings.filterwarnings('ignore')

# Daten laden
basis_ordner_pfad = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets\35Hz12kN'
merkmals_datei = os.path.join(basis_ordner_pfad, 'training_validation_features.csv')

# CSV-Datei einlesen
df = pd.read_csv(merkmals_datei)

# Entfernen der 'Bearing_ID' und 'RUL'-Spalten
df = df.drop(['Bearing_ID', 'RUL'], axis=1)

# Daten bereinigen: Entfernen von Zeilen mit inf oder NaN Werten
df = df.replace([np.inf, -np.inf], np.nan).dropna()

# Berechnung der Korrelation
korrelation = df.corr()

# Ausgabe der am stärksten korrelierten Merkmale
print("Top 10 am stärksten korrelierte Merkmale:")
korrelations_paare = korrelation.unstack()
sortierte_paare = korrelations_paare.sort_values(kind="quicksort", ascending=False)
# Vermeidung von Eigenkorrelationen (1.0)
print(sortierte_paare[sortierte_paare < 1].head(10))

# Heatmap erstellen
plt.figure(figsize=(30, 24))
```

```
sns.set(font_scale=0.8)  # Schriftgröße anpassen
farbpalette = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(korrelation, cmap=farbpalette, vmax=1, vmin=-1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5},
            annot=False)  # annot=True setzt konkrete Korrelationskoeffizienten

# Titel setzen
plt.title('Merkmal-Korrelations-Heatmap', fontsize=16)

# Layout anpassen und anzeigen
plt.tight_layout()
plt.show()

# VIF berechnen, um Multikollinearität zu identifizieren
def calculate_vif(df):
    vif_data = pd.DataFrame()
    vif_data["Merkmal"] = df.columns
    vif_data["VIF"] = [variance_inflation_factor(df.values, i) for i in range(len(df.columns))]
    return vif_data

try:
    # VIF für die Merkmale berechnen
    vif_data = calculate_vif(df)

    # Ausgabe von Merkmalen mit hohem VIF (>10)
    print("Merkmale mit hohem VIF (>10):")
    print(vif_data[vif_data["VIF"] > 10])
except Exception as e:
    print(f"Fehler bei der VIF-Berechnung: {e}")
    print("Überprüfen Sie Ihre Daten auf Multikollinearität oder perfekte Korrelationen.")

# Zusätzliche Datenüberprüfung
print("\nDatenüberprüfung:")
print(df.describe())
print("\nSpalten mit konstanten Werten:")
print(df.columns[df.nunique() == 1].tolist())
```

```
Top 10 am stärksten korrelierte Merkmale:
Mse.ts.Vertical             Ampl-var.f.Vertical          0.997791
Ampl-var.f.Vertical         Mse.ts.Vertical              0.997791
Ampl-var.f.Horizontal       Mse.ts.Horizontal            0.985926
Mse.ts.Horizontal           Ampl-var.f.Horizontal        0.985926
Sd.cp.Vertical              Ampl-mean.f.Vertical         0.977383
Ampl-mean.f.Vertical        Sd.cp.Vertical               0.977383
Ampl-skewness.f.Vertical    Ampl-kurtosis.f.Vertical     0.970339
Ampl-kurtosis.f.Vertical    Ampl-skewness.f.Vertical     0.970339
Ampl-skewness.f.Horizontal  Ampl-kurtosis.f.Horizontal   0.969356
Ampl-kurtosis.f.Horizontal  Ampl-skewness.f.Horizontal   0.969356
dtype: float64
```


Merkmal-Korrelations-Heatmap

Merkmale mit hohem VIF (>10):

	Merkmal	VIF
0	Ampl-kurtosis.f.Horizontal	222.079847
1	Ampl-kurtosis.f.Vertical	162.893995
2	Ampl-mean.f.Horizontal	4748.539896
3	Ampl-mean.f.Vertical	2889.550230
4	Ampl-skewness.f.Horizontal	582.926729
5	Ampl-skewness.f.Vertical	582.961009
6	Ampl-var.f.Horizontal	3122.302568
7	Ampl-var.f.Vertical	11363.937479
10	Ampl1.f.Horizontal	316.953453
11	Ampl1.f.Vertical	217.566031
14	Ampl2.f.Horizontal	65.900276
15	Ampl2.f.Vertical	91.665179
24	Max.ts.Horizontal	67.206329
25	Max.ts.Vertical	111.712252
26	Mse.ts.Horizontal	4453.930744
27	Mse.ts.Vertical	13987.518840
28	Sd.cp.Horizontal	3365.725637
29	Sd.cp.Vertical	1898.332300

Datenüberprüfung:

	Ampl-kurtosis.f.Horizontal	Ampl-kurtosis.f.Vertical
count	8128.000000	8128.000000
mean	42.067780	44.910348
std	28.696002	22.142557
min	1.309853	2.693657
25%	18.343997	29.247694
50%	38.672063	41.451049
75%	57.653261	56.986366
max	284.336762	154.053367

	Ampl-mean.f.Horizontal	Ampl-mean.f.Vertical
count	8128.000000	8128.000000
mean	45.664326	43.063713
std	22.235689	23.522575
min	12.756284	11.990922
25%	23.725687	22.934880
50%	47.025954	40.140291
75%	61.722204	54.590667
max	133.760429	143.077692

	Ampl-skewness.f.Horizontal	Ampl-skewness.f.Vertical
count	8128.000000	8128.000000
mean	5.049676	5.658470
std	1.895727	1.397394
min	1.196156	1.614108
25%	3.412478	4.760049
50%	5.141146	5.661261
75%	6.364675	6.519341
max	15.359768	10.883669

	Ampl-var.f.Horizontal	Ampl-var.f.Vertical	Ampl1-freq.f.Horizontal
count	8128.000000	8128.000000	8128.000000
mean	4284.699049	8230.236128	1113.993701
std	4669.846530	12120.572827	1037.585269
min	92.071459	146.048884	307.200000
25%	628.859517	1283.901892	870.400000
50%	3739.559397	4535.036734	1075.200000
75%	6735.325133	9775.675891	1075.200000
max	49800.185334	147538.672971	12083.200000

	Ampl1-freq.f.Vertical	...	Mse.ts.Horizontal	Mse.ts.Vertical
count	8128.000000	...	8128.000000	8128.000000
mean	845.700787	...	6.860695	10.633086
std	618.242001	...	6.727430	14.553947
min	102.400000	...	0.267888	0.295955
25%	742.400000	...	1.212746	1.831468
50%	870.400000	...	6.243798	6.207623
75%	870.400000	...	10.427555	12.597669
max	11827.200000	...	62.264007	165.912352

	Sd.cp.Horizontal	Sd.cp.Vertical	Skewness.cp.Horizontal
count	8128.000000	8128.000000	8128.000000
mean	2.313036	2.746694	-0.108552
std	1.229461	1.758572	0.435760
min	0.517588	0.544051	-3.108763
25%	1.101378	1.353507	-0.349268
50%	2.498866	2.491900	-0.086129
75%	3.229468	3.550240	0.140894
max	7.890755	12.880838	1.816889

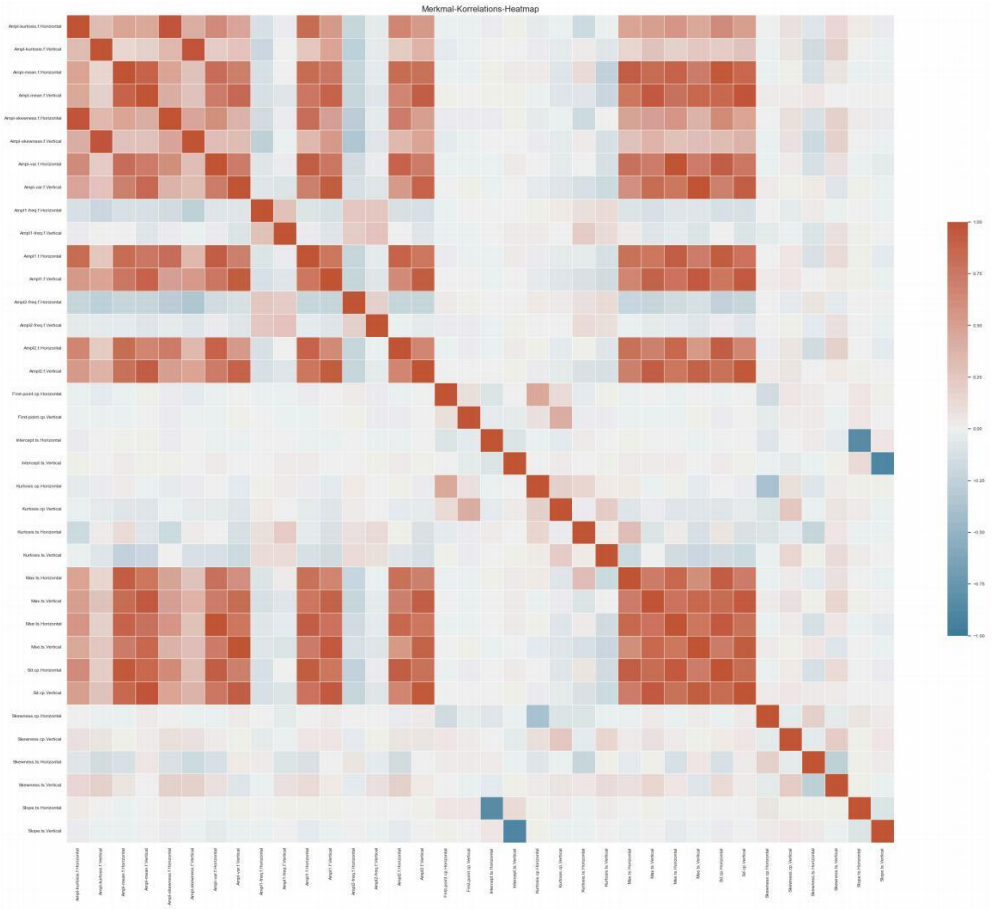
```
       Skewness.cp.Vertical  Skewness.ts.Horizontal  Skewness.ts.Vertical  \
count           8128.000000             8128.000000           8128.000000
mean               0.032901               -0.061649             -0.050049
std                0.465095                0.164537              0.158027
min               -1.851548               -0.969985             -0.545139
25%               -0.248905               -0.166267             -0.158221
50%                0.000000               -0.054677             -0.055690
75%                0.287356                0.040667              0.047461
max                2.447824                0.988863              0.481179

       Slope.ts.Horizontal  Slope.ts.Vertical
count          8128.000000        8128.000000
mean              0.000001          -0.000002
std               0.000100           0.000203
min              -0.000730          -0.001571
25%              -0.000049          -0.000104
50%               0.000001          -0.000008
75%               0.000053           0.000089
max               0.000495           0.001518

[8 rows x 36 columns]

Spalten mit konstanten Werten:
[]
```

In [30]:

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import warnings

warnings.filterwarnings('ignore')

# Daten laden
basis_ordner_pfad = r'D:\Jupyter\xjtu\XJTU-SY_Bearing_Datasets\Data\XJTU-SY_Bearing_Datasets\35Hz12kN'
merkmals_datei = os.path.join(basis_ordner_pfad, 'training_validation_features.csv')

# CSV-Datei einlesen
df = pd.read_csv(merkmals_datei)

# Entfernen der 'Bearing_ID' und 'RUL'-Spalten
y = df['RUL']  # Zielvariable speichern
df = df.drop(['Bearing_ID', 'RUL'], axis=1)

# Daten bereinigen: Entfernen von Zeilen mit inf oder NaN Werten
df = df.replace([np.inf, -np.inf], np.nan).dropna()

# Berechnung der Korrelation
korrelation = df.corr()

# Ausgabe der am stärksten korrelierten Merkmale
print("Top 10 am stärksten korrelierte Merkmale:")
korrelations_paare = korrelation.unstack()
sortierte_paare = korrelations_paare.sort_values(kind="quicksort", ascending=False)
print(sortierte_paare[sortierte_paare < 1].head(10))

# Heatmap erstellen
plt.figure(figsize=(30, 24))
sns.set(font_scale=0.8)
farbpalette = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(korrelation, cmap=farbpalette, vmax=1, vmin=-1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5},
            annot=False)

plt.title('Merkmal-Korrelations-Heatmap', fontsize=16)
plt.tight_layout()
plt.show()

# VIF berechnen
def calculate_vif(df):
    vif_data = pd.DataFrame()
    vif_data["Merkmal"] = df.columns
    vif_data["VIF"] = [variance_inflation_factor(df.values, i) for i in range(len(df.columns))]
    return vif_data

try:
    vif_data = calculate_vif(df)
```

```
    print("Merkmale mit hohem VIF (>10):")
    print(vif_data[vif_data["VIF"] > 10])
except Exception as e:
    print(f"Fehler bei der VIF-Berechnung: {e}")

# Schrittweise Regression für Merkmalsauswahl
def stepwise_selection(X, y, initial_list=[],
                       threshold_in=0.01,
                       threshold_out=0.05,
                       verbose=True):
    included = list(initial_list)
    while True:
        changed = False
        # Vorwärtsauswahl
        excluded = list(set(X.columns) - set(included))
        new_pval = pd.Series(index=excluded)
        for new_column in excluded:
            model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included + [new_column]]))).fit()
            new_pval[new_column] = model.pvalues[new_column]
        best_pval = new_pval.min()
        if best_pval < threshold_in:
            best_feature = new_pval.idxmin()
            included.append(best_feature)
            changed = True
            if verbose:
                print(f'Add {best_feature} with p-value {best_pval}')

        # Rückwärtselimination
        model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
        pvalues = model.pvalues.iloc[1:]
        worst_pval = pvalues.max()
        if worst_pval > threshold_out:
            changed = True
            worst_feature = pvalues.idxmax()
            included.remove(worst_feature)
            if verbose:
                print(f'Remove {worst_feature} with p-value {worst_pval}')
        if not changed:
            break
    return included

# Durchführung der schrittweisen Regression
try:
    ausgewählte_merkmale = stepwise_selection(df, y)
    print("Ausgewählte Merkmale:")
    print(ausgewählte_merkmale)
except Exception as e:
    print(f"Fehler bei der schrittweisen Regression: {e}")
    print("Überprüfen Sie Ihre Daten auf Multikollinearität oder andere Probleme.")

# Zusätzliche Datenüberprüfung
print("\nDatenüberprüfung:")
print(df.describe())
print("\nSpalten mit konstanten Werten:")
print(df.columns[df.nunique() == 1].tolist())
```

```
Top 10 am stärksten korrelierte Merkmale:
Mse.ts.Vertical             Ampl-var.f.Vertical           0.997791
Ampl-var.f.Vertical         Mse.ts.Vertical               0.997791
Ampl-var.f.Horizontal       Mse.ts.Horizontal             0.985926
Mse.ts.Horizontal           Ampl-var.f.Horizontal         0.985926
Sd.cp.Vertical              Ampl-mean.f.Vertical          0.977383
Ampl-mean.f.Vertical        Sd.cp.Vertical                0.977383
Ampl-skewness.f.Vertical    Ampl-kurtosis.f.Vertical      0.970339
Ampl-kurtosis.f.Vertical    Ampl-skewness.f.Vertical      0.970339
Ampl-skewness.f.Horizontal  Ampl-kurtosis.f.Horizontal    0.969356
Ampl-kurtosis.f.Horizontal  Ampl-skewness.f.Horizontal    0.969356
dtype: float64
```


Merkmal-Korrelations-Heatmap

Merkmale mit hohem VIF (>10):

	Merkmal	VIF
0	Ampl-kurtosis.f.Horizontal	222.079847
1	Ampl-kurtosis.f.Vertical	162.893995
2	Ampl-mean.f.Horizontal	4748.539896
3	Ampl-mean.f.Vertical	2889.550230
4	Ampl-skewness.f.Horizontal	582.926729
5	Ampl-skewness.f.Vertical	582.961009
6	Ampl-var.f.Horizontal	3122.302568
7	Ampl-var.f.Vertical	11363.937479
10	Ampl1.f.Horizontal	316.953453
11	Ampl1.f.Vertical	217.566031
14	Ampl2.f.Horizontal	65.900276
15	Ampl2.f.Vertical	91.665179
24	Max.ts.Horizontal	67.206329
25	Max.ts.Vertical	111.712252
26	Mse.ts.Horizontal	4453.930744
27	Mse.ts.Vertical	13987.518840
28	Sd.cp.Horizontal	3365.725637
29	Sd.cp.Vertical	1898.332300

Fehler bei der schrittweisen Regression: The indices for endog and exog are not aligned
Überprüfen Sie Ihre Daten auf Multikollinearität oder andere Probleme.

Datenüberprüfung:

	Ampl-kurtosis.f.Horizontal	Ampl-kurtosis.f.Vertical
count	8128.000000	8128.000000
mean	42.067780	44.910348
std	28.696002	22.142557
min	1.309853	2.693657
25%	18.343997	29.247694
50%	38.672063	41.451049
75%	57.653261	56.986366
max	284.336762	154.053367

	Ampl-mean.f.Horizontal	Ampl-mean.f.Vertical
count	8128.000000	8128.000000
mean	45.664326	43.063713
std	22.235689	23.522575
min	12.756284	11.990922
25%	23.725687	22.934880
50%	47.025954	40.140291
75%	61.722204	54.590667
max	133.760429	143.077692

	Ampl-skewness.f.Horizontal	Ampl-skewness.f.Vertical
count	8128.000000	8128.000000
mean	5.049676	5.658470
std	1.895727	1.397394
min	1.196156	1.614108
25%	3.412478	4.760049
50%	5.141146	5.661261
75%	6.364675	6.519341
max	15.359768	10.883669

	Ampl-var.f.Horizontal	Ampl-var.f.Vertical	Ampl1-freq.f.Horizontal
count	8128.000000	8128.000000	8128.000000
mean	4284.699049	8230.236128	1113.993701
std	4669.846530	12120.572827	1037.585269
min	92.071459	146.048884	307.200000
25%	628.859517	1283.901892	870.400000
50%	3739.559397	4535.036734	1075.200000
75%	6735.325133	9775.675891	1075.200000
max	49800.185334	147538.672971	12083.200000

	Ampl1-freq.f.Vertical	...	Mse.ts.Horizontal	Mse.ts.Vertical
count	8128.000000	...	8128.000000	8128.000000
mean	845.700787	...	6.860695	10.633086
std	618.242001	...	6.727430	14.553947
min	102.400000	...	0.267888	0.295955
25%	742.400000	...	1.212746	1.831468
50%	870.400000	...	6.243798	6.207623
75%	870.400000	...	10.427555	12.597669
max	11827.200000	...	62.264007	165.912352

	Sd.cp.Horizontal	Sd.cp.Vertical	Skewness.cp.Horizontal
count	8128.000000	8128.000000	8128.000000
mean	2.313036	2.746694	-0.108552
std	1.229461	1.758572	0.435760
min	0.517588	0.544051	-3.108763
25%	1.101378	1.353507	-0.349268
50%	2.498866	2.491900	-0.086129

```
75%             3.229468        3.550240                0.140894
max             7.890755       12.880838                1.816889

       Skewness.cp.Vertical  Skewness.ts.Horizontal  Skewness.ts.Vertical  \
count           8128.000000             8128.000000           8128.000000
mean               0.032901               -0.061649             -0.050049
std                0.465095                0.164537              0.158027
min               -1.851548               -0.969985             -0.545139
25%               -0.248905               -0.166267             -0.158221
50%                0.000000               -0.054677             -0.055690
75%                0.287356                0.040667              0.047461
max                2.447824                0.988863              0.481179

       Slope.ts.Horizontal  Slope.ts.Vertical
count          8128.000000        8128.000000
mean              0.000001          -0.000002
std               0.000100           0.000203
min              -0.000730          -0.001571
25%              -0.000049          -0.000104
50%               0.000001          -0.000008
75%               0.000053           0.000089
max               0.000495           0.001518

[8 rows x 36 columns]

Spalten mit konstanten Werten:
[]
```

In [25]:

```python
# VIF berechnen
def calculate_vif(df):
    vif_data = pd.DataFrame()
    vif_data["Merkmal"] = df.columns
    vif_data["VIF"] = [variance_inflation_factor(df.values, i) for i in range(len(df.columns))]
    return vif_data

try:
    vif_data = calculate_vif(df)

    # VIF 值按升序排序
    vif_data_sorted = vif_data.sort_values('VIF')

    print("Alle Merkmale mit ihren VIF-Werten (sortiert nach VIF):")
    print(vif_data_sorted)

    print("\nMerkmale mit niedrigem VIF (<=10):")
    print(vif_data_sorted[vif_data_sorted["VIF"] <= 10])

    print("\nMerkmale mit hohem VIF (>10):")
    print(vif_data_sorted[vif_data_sorted["VIF"] > 10])

except Exception as e:
    print(f"Fehler bei der VIF-Berechnung: {e}")
```

Alle Merkmale mit ihren VIF-Werten (sortiert nach VIF):

	Merkmal	VIF
31	Skewness.cp.Vertical	1.171429
30	Skewness.cp.Horizontal	1.301871
20	Kurtosis.cp.Horizontal	1.772475
33	Skewness.ts.Vertical	2.001608
12	Ampl2-freq.f.Horizontal	2.027942
32	Skewness.ts.Horizontal	2.194864
21	Kurtosis.cp.Vertical	2.357140
23	Kurtosis.ts.Vertical	2.393775
13	Ampl2-freq.f.Vertical	2.552171
8	Ampl1-freq.f.Horizontal	2.596939
34	Slope.ts.Horizontal	3.567953
18	Intercept.ts.Horizontal	3.590038
17	First-point.cp.Vertical	3.608198
16	First-point.cp.Horizontal	3.634578
9	Ampl1-freq.f.Vertical	3.769236
22	Kurtosis.ts.Horizontal	4.258772
35	Slope.ts.Vertical	4.788857
19	Intercept.ts.Vertical	4.834610
14	Ampl2.f.Horizontal	65.900276
24	Max.ts.Horizontal	67.206329
15	Ampl2.f.Vertical	91.665179
25	Max.ts.Vertical	111.712252
1	Ampl-kurtosis.f.Vertical	162.893995
11	Ampl1.f.Vertical	217.566031
0	Ampl-kurtosis.f.Horizontal	222.079847
10	Ampl1.f.Horizontal	316.953453
4	Ampl-skewness.f.Horizontal	582.926729
5	Ampl-skewness.f.Vertical	582.961009
29	Sd.cp.Vertical	1898.332300
3	Ampl-mean.f.Vertical	2889.550230
6	Ampl-var.f.Horizontal	3122.302568
28	Sd.cp.Horizontal	3365.725637
26	Mse.ts.Horizontal	4453.930744
2	Ampl-mean.f.Horizontal	4748.539896
7	Ampl-var.f.Vertical	11363.937479
27	Mse.ts.Vertical	13987.518840

Merkmale mit niedrigem VIF (<=10):

	Merkmal	VIF
31	Skewness.cp.Vertical	1.171429
30	Skewness.cp.Horizontal	1.301871
20	Kurtosis.cp.Horizontal	1.772475
33	Skewness.ts.Vertical	2.001608
12	Ampl2-freq.f.Horizontal	2.027942
32	Skewness.ts.Horizontal	2.194864
21	Kurtosis.cp.Vertical	2.357140
23	Kurtosis.ts.Vertical	2.393775
13	Ampl2-freq.f.Vertical	2.552171
8	Ampl1-freq.f.Horizontal	2.596939
34	Slope.ts.Horizontal	3.567953
18	Intercept.ts.Horizontal	3.590038
17	First-point.cp.Vertical	3.608198
16	First-point.cp.Horizontal	3.634578
9	Ampl1-freq.f.Vertical	3.769236
22	Kurtosis.ts.Horizontal	4.258772
35	Slope.ts.Vertical	4.788857
19	Intercept.ts.Vertical	4.834610

Merkmale mit hohem VIF (>10):

	Merkmal	VIF
14	Ampl2.f.Horizontal	65.900276
24	Max.ts.Horizontal	67.206329
15	Ampl2.f.Vertical	91.665179
25	Max.ts.Vertical	111.712252
1	Ampl-kurtosis.f.Vertical	162.893995
11	Ampl1.f.Vertical	217.566031
0	Ampl-kurtosis.f.Horizontal	222.079847
10	Ampl1.f.Horizontal	316.953453
4	Ampl-skewness.f.Horizontal	582.926729
5	Ampl-skewness.f.Vertical	582.961009
29	Sd.cp.Vertical	1898.332300
3	Ampl-mean.f.Vertical	2889.550230
6	Ampl-var.f.Horizontal	3122.302568
28	Sd.cp.Horizontal	3365.725637
26	Mse.ts.Horizontal	4453.930744
2	Ampl-mean.f.Horizontal	4748.539896
7	Ampl-var.f.Vertical	11363.937479
27	Mse.ts.Vertical	13987.518840

Anhang 3 Parameter für RNNs

Parameter	RS-RNN	LS-RNN
input_size	1	1
hidden_size	32	32
output_size	1	1
Epoch	20	100
Learning rate	0,0001	0,0001
batch_size	128	64
num_layers	2	2

Parameter	RS-LSTM	LS-LSTM
input_size	1	1
hidden_size	64	64
output_size	1	1
Epoch	20	100
Learning rate	0,0001	0,0001
batch_size	128	64
num_layers	1	1

Parameter	RS-BILSTM	LS-BILSTM
input_size	1	1
hidden_size	64	64
output_size	1	1
Epoch	20	100
Learning rate	0,0001	0,0001
batch_size	128	64
num_layers	1	1