Fehlervorhersage für Schrittmotoren durch Methoden des maschinellen Lernens

Merkmalsaufbereitung und Daten-Augmentation bei Zeitreihendatensätzen sowie Gewinnung realer Daten

der Fakultät für Elektrotechnik, Informationstechnik und Medientechnik
der Bergischen Universität Wuppertal vorgelegte

Dissertation

---

zur Erlangung des akademischen Grades
eines Doktors der Ingenieurwissenschaften

von
M.Sc. Maxime Goubeaud
aus Kirchhain

Wuppertal 2022

Tag der mündlichen Prüfung:	29. April 2022
Erstgutachter:	Prof. Dr.-Ing. Anton Kummert
Zweitgutachter:	Prof. Dr.-Ing. Bernd Tibken

Die Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20220504-120347-6
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20220504-120347-6]

DOI: 10.25926/crqv-rb89
[https://doi.org/10.25926/crqv-rb89]

Danksagung

Zuerst möchte ich mich ausdrücklich bei meinem Doktorvater Prof. Anton Kummert bedanken, der mich während meiner Dissertation stets begleitet und bei schwierigen Fragestellungen unterstützt hat. Ohne sein Vertrauen, seine offene und lösungsorientierte Art und die exzellente wissenschaftliche Betreuung wäre diese Arbeit nicht möglich gewesen. Ich möchte mich auch bei Prof. Bernd Tibken bedanken, der sich ohne zu zögern bereit erklärt hat, als Korreferent zu fungieren und mich schon während meiner Bachelor- und Masterthesis begleitet hat.

Darüber hinaus möchte ich mich bei der Firma Vaillant bedanken, die mir das Anfertigen dieser Arbeit durch ein Stipendium ermöglicht hat. Hier möchte ich meine Führungskräfte Matthias Stursberg und Andre Maaß herausstellten, die mir jede erforderliche Unterstützung zukommen ließen. Auch Tim Grunert, Bruno Sommerfeld, Maximilian Baron und Marvin Resch möchte ich nennen, die sich mit ihrem Fachwissen in unterschiedlichen Gebieten stets für das Gelingen meiner Arbeit eingesetzt haben. Darüber hinaus möchte ich mich bei Philipp Joußen, Jan Lützenkirchen, Lucas Schelkes, Timo Krah, Sary Masri, Agasha Ratam und Julia Sudhoff bedanken, die ich während ihres Studiums betreuen durfte und die so einen wesentlichen Teil zu dieser Arbeit beigetragen haben.

Meinen größten Dank möchte ich meinen Eltern Dr. Anette und Rene Goubeaud aussprechen, die mich während meiner Studienzeit stets unterstützt und motiviert haben. Danke, dass ihr mir ermöglicht habt meinen Weg frei zu wählen und diesen losgelöst von Zweifeln und Ängsten zu gehen. Hervorheben möchte ich den Einsatz meiner Mutter, die diese Arbeit mit aufopferungsvoller Hingabe maßgeblich geprägt hat und mir in allen Lebenslagen Mut zuspricht und Hilfe zukommen lässt.

Die letzten Worte widme ich Nicolla, der Liebe meines Lebens. Danke für deine Geduld, dein Vertrauen und deine Liebe. Danke, dass du mich in jeder Situation unterstützt, für mich da bist und meinen Weg zu unserem machst. Ich könnte nicht glücklicher sein, als dich an meiner Seite zu wissen und nicht stolzer, als dich zur Frau zu nehmen.

Kurzfassung

Algorithmen des maschinellen Lernens werden im Kontext der Industrie 4.0 häufig dafür genutzt, Fehlerfälle von Maschinen zu prognostizieren und so teure Stillstandszeiten und den Defekt kostenintensiver Bauteile zu vermeiden.

Die vorliegende Arbeit greift diesen Trend auf und befasst sich mit dem Entwurf eines Systems zur frühzeitigen Fehlererkennung für Schrittmotoren mit Hilfe von Transformern. Das System ermöglicht proaktive Wartungs- und Instandhaltungsmaßnahmen einzuleiten, indem Fehlerfälle mit einer Vorlaufzeit von 30 Tagen sicher erkannt werden. Zusätzlich wurde eine Methode zur Reduktion des Verschleißes von Schrittmotoren erarbeitet, die die Erkennung und Reduktion von Betriebsbereichen außerhalb des normalen Arbeitsbereiches erlaubt. Ein Dauerversuchsaufbau, der wesentliche Betriebsparameter von 32 Schrittmotoren über ein Jahr aufzeichnet, liefert für beide Anwendungsfälle die erforderliche Datenbasis.

Darüber hinaus wurden neuartige Data Augmentation Methoden für Zeitreihen erforscht und deren Funktionalität anhand von bekannten Datensätzen erwiesen. Die erarbeiteten Werkzeuge wurden im Folgenden dafür verwendet, künstliche Daten für den Trainingsprozess der zur Fehlervorhersage sowie zur Verschleißreduktion verwendeten Algorithmen zu erzeugen. Dies führte zu einer Verbesserung der Ergebnisse und bestätigt die Funktionalität der Methoden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Acc Accurary

Adam Adaptive moment estimation

AW Attention-wheigts

CNN Convolutional Neural Network

conv Convolutional

DA Data Augmentation

ELBO Evidence Lower Bound

FC Fully connected

FH Fahrt hoch

FN Falsch negativ

FP Falsch positiv

FR Fahrt runter

GAN Generative Adversarial Network

KL Kullback-Leibler

kNN k-Nearest Neighbors

LSTM Long Short-Term Memory

MHA Multi-Head Attention

MLB MicroLabBox

NLP Natural Language Processing

OA Oberer Anschlag

ReLU Rectified Linear Unit

RGB Farbraum basierend auf den Farben Rot, Grün und Blau

RNB Random Noise Boxes

RNN Recurrent Neural Network

RUL Remaining Useful Lifetime

SM Schrittmotor

Sony SonyAIBRobotSurfaceX

SVM Support Vector Machine

TN Wahr negativ

TP Wahr positiv

UA Unterer Anschlag

Uwave UWaveGestureLibraryX

VAE Variational Autoencoder

WNW White Noise Windows

Kapitel 1

Einleitung

1.1 Hintergrund und Motivation

Künstliche Intelligenz und maschinelles Lernen werden in vielfältigen Bereichen eingesetzt. Eines der interessantesten Forschungsfelder ist die Vorhersage von Fehlerfällen an Maschinen oder Bauteilen. Durch Bewertung relevanter Betriebsparameter, welche durch spezifische Sensorik gewonnen werden, können Rückschlüsse auf den Zustand der Maschine oder der untersuchten Komponente gezogen werden. Durch frühzeitiges Erkennen von potenziellen Fehlerfällen ist es möglich, proaktiv einzugreifen und das Auftreten des Fehlers zu verhindern. Durch Verhinderung des Fehlerfalls können Defekte kostenintensiver Bauteile oder ungewünschte Stillstandszeiten von Maschinen vermieden werden. Auch eine effektivere Routenplanung für servicebasierte Unternehmen und die Optimierung von Wartungsintervallen werden so ermöglicht. Mittels richtig prognostizierter Fehlerfälle lassen sich die betriebliche Effizienz steigern, Wartungskosten verringern und Emissionen auf Grund verminderter Transport- und Fahrtstrecken der Fachhandwerker sparen.

Für die praktische Umsetzung einer Fehlervorhersage werden im Bereich des maschinellen Lernens Algorithmen verwendet. Um dem ausgewählten Algorithmus das Vorhersagen von Fehlerfällen zu ermöglichen, ist es erforderlich, diesen mit Hilfe geeigneter Datensätze zu trainieren. Während des Trainingsprozesses lernen Algorithmen die Unterschiede zwischen Daten, die kurz vor dem Auftreten eines Fehlers entstanden sind und Daten, die während des normalen Betriebs aufgenommen wurden. Erkennt der Algorithmus eine Abweichung vom Normalzustand in den Daten, gibt der Algorithmus eine Warnung aus und weist den Betreiber der Maschine so auf einen möglichen Fehlerfall hin. Als Daten kommen hierbei häufig Zeitreihen zum Einsatz, die mittels geeigneter Sensorik während des Betriebs der Maschine aufgezeichnet wurden.

Die Erstellung eines Datensatzes zur Ausfallvorhersage ist mit hohem Aufwand

verbunden. Zunächst ist es erforderlich, Daten von unterschiedlichen Betriebsstunden der zu überwachenden Maschine zu generieren. Hierbei ist wichtig, dass nicht nur Daten während des normalen Betriebes aufgezeichnet werden, sondern vor allem die Daten kurz vor Auftritt eines Fehlers. Damit keine Informationen verloren gehen, müssen diese Daten in einer hohen Qualität gesammelt werden. Es ist daher ratsam, diese in einer ausreichend hohen Frequenz sowie möglichst rauscharm aufzuzeichnen. Zusätzlich ist es von Vorteil, wenn Daten von mehreren Maschinen des zu untersuchenden Maschinentyps vorliegen. Dies begründet sich in der unterschiedlichen Beschaffenheit von Signalen baugleicher Maschinen und damit der Abbildung verschiedener Fehlerbilder. Damit wird die Erstellung solcher Datensätze zu einem langwierigen Prozess, der mehrere Jahre in Anspruch nehmen und teure Sensorik erforderlich machen kann.

Auf Grund dieser hohen Anforderungen existieren nur wenig öffentlich zugängliche Datensätze, die die Entwicklung eines Systems zur Ausfallvorhersage ermöglichen. Soll solch ein System für eine bestimmte Maschine oder ein bestimmtes Bauteil entwickelt werden, steht meist kein Datensatz zur Verfügung. In diesem Fall muss zuerst eine geeignete Datenbasis geschaffen werden.

Ein weiteres Problem ist, dass in der Regel wesentlich weniger Daten zur Verfügung stehen, die kurz vor dem Auftritt eines Fehlers aufgezeichnet wurden, als Daten, die während des reibungslosen Betriebs aufgezeichnet wurden. Für den Trainingsprozess von Algorithmen ist es jedoch von Vorteil, wenn die Datenumfänge der unterschiedlichen Ausprägungen sich nicht stark unterscheiden. Die Gestaltung eines validen Datensatzes, der sich für den Trainingsprozess eines Algorithmus zur Ausfallvorhersage eignet, stellt daher häufig eine große Herausforderung in Forschung und Industrie dar.

Einen Ansatz zur Lösung dieses Problems bietet Data Augmentation. Data Augmentation beschäftigt sich mit der Erzeugung künstlicher Daten zur Anreicherung vorhandener Datensätze. Durch die Erzeugung solcher synthetischer Fehlerdaten kann für einen Ausgleich zwischen Fehlerdaten und Daten aus dem normalen Betrieb gesorgt werden. Der ausgeglichene Datensatz erlaubt einen besseren Trainingsprozess des Algorithmus, was wiederum zu einer verbesserten Vorhersage von Fehlerfällen führt. Im Vergleich zu anderen Forschungsfeldern wie z.B. der Bildverarbeitung, existieren im Bereich für Zeitreihen bislang nur wenige Data Augmentation Methoden.

Schrittmotoren sind eine der wichtigsten Komponenten, aber auch eine der Hauptfehlerquellen von Gasheizsystemen. Der Defekt des Schrittmotors kann zum Stillstand eines kompletten Heizsystems führen. Auch in anderen Systemen, wie beispielsweise Druckern kommen Schrittmotoren zum Einsatz. Die richtige Vorhersage von Fehlerfällen ermöglicht Fachhandwerkern den proaktiven Austausch

von Schrittmotoren und trägt so dazu bei, Stillstandszeiten von Maschinen zu reduzieren. Ein System zur frühzeitigen Erkennung und Vorhersage von Fehlerfällen stellt sich daher als interessantes Forschungsfeld dar und würde der Industrie zudem kosten- und ressourcensparende Vorteile bieten und dem Endkunden eine Komfort-Verbesserung ermöglichen. Es existiert jedoch kein öffentlich zugänglicher Datensatz, der es erlaubt, eine valide Ausfallerkennung für Schrittmotoren zu entwickeln.

1.2 Zielsetzung und Aufbau der Arbeit

Ziel dieser Arbeit ist es, die beschriebenen Vorteile zu heben und ein System zur frühzeitigen Erkennung von Fehlerfällen an Schrittmotoren zu entwickeln. Hierfür ist es notwendig, einen Dauerversuch zu entwerfen, der die Generierung der erforderlichen Datenbasis ermöglicht. Darüber hinaus ist ein geeigneter Algorithmus auszuwählen, der eine Fehlervorhersage erlaubt. Um dem Mangel von Data Augmentation Methoden in diesem Bereich entgegenzuwirken, sollen zusätzlich neue Techniken zur Anreicherung von Zeitreihen-Datensätzen erprobt werden. Die Erkennung und die Reduktion von Betriebsbereichen, die den Verschleiß von Schrittmotoren begünstigen, ist wünschenswert und soll daher ebenfalls Berücksichtigung finden.

Zur Erreichung dieser Ziele gliedert sich die Arbeit in acht Kapitel.

Nach der Einleitung (Kapitel 1) werden in Kapitel 2 die für das Verständnis der Arbeit notwendigen Grundlagen erläutert. Zunächst wird das Funktionsprinzip von Permanentmagnetschrittmotoren beschrieben, ehe das Forschungsfeld Remaining Useful Lifetime Prediction vorgestellt wird. Anschließend folgen ein Überblick über verschiedene Algorithmen des maschinellen Lernens sowie die Beschreibung von Metriken, die eine Bewertung der Funktionalität von Algorithmen erlauben. Es schließt sich eine Einführung in das Themenfeld Data Augmentation an, in der ein Überblick über bestehende Techniken anhand einer Literaturrecherche gegeben wird. Abschließend werden Rekurrenzplots vorgestellt.

Nach umfassender Beschreibung dieser theoretischen Grundlagen umschließt Teil 1 die Kapitel 3 und 4, in denen der Fokus auf neuartigen Verfahren zur Erweiterung und Analyse von Zeitreihen-Datensätzen liegt.

In Kapitel 3 werden drei neue Methoden zur Erweiterung von Zeitreihen-Datensätzen vorgestellt. Hierzu werden zunächst neun bekannte und öffentlich zugängliche Datensätze gezeigt, die unterschiedliche Klassifikationsaufgaben bereitstellen. Diese werden zur Evaluation der drei neuartigen Methoden genutzt. Es folgen die detaillierte Beschreibung, Evaluation und Diskussion der vorgestellten Methoden.

Kapitel 4 zeigt einen innovativen Ansatz zur Analyse von Zeitreihen mit Hilfe von Rekurrenzplots. Es wird zunächst eine ausführliche Literaturrecherche über die Kodierung von Zeitreihendaten in Rekurrenzplots durchgeführt. Im Folgenden werden die Zeitreihen, der bereits in Kapitel 3 vorgestellten Datensätze mit unterschiedlichen Parametereinstellungen in Rekurrenzplots umgewandelt. Nachfolgend wird die Evaluation, in der ein Convolutional Neural Network als Werkzeug zur Klassifikation genutzt wird, durchgeführt. Das Kapitel schließt mit einer Diskussion über die Tauglichkeit von Rekurrenzplots zur Analyse von Zeitreihendaten.

Teil 2 beschäftigt sich mit der praktischen Umsetzung eines Systems zur Fehlervorhersage an Schrittmotoren sowie der Erkennung von schädlichen Betriebsbereichen.

Um ein System zur Fehlererkennung von Schrittmotoren umsetzen zu können, wird ein geeigneter Datensatz benötigt. Kapitel 5 beschreibt die Erstellung eines Dauerversuchsaufbaus zur Gewinnung dieses Datensatzes. Hier wird eingangs die Umsetzung des Dauerversuchs erläutert, der die Vermessung von 32 Schrittmotoren ermöglicht. Nachfolgend werden die während des Betriebs im Dauerversuch aufgetretenen Fehler untersucht. Abschließend erfolgt ein Überblick über die gewonnene Datenbasis.

In Kapitel 6 wird die Realisierung einer innovativen Anschlagserkennung von Schrittmotoren beschrieben, die helfen kann, unnötigen Verschleiß durch Betrieb außerhalb des normalen Arbeitsbereiches zu reduzieren. Hierzu werden Rekurrenzplots, erzeugt aus den in Kapitel 5 gewonnen Daten, als Werkzeug genutzt.

Die praktische Umsetzung eines Systems zur frühzeitigen Erkennung von Fehlern bei Schrittmotoren erfolgt in Kapitel 7. Hier wird zunächst eine für die Aufgabenstellung geeignete Datenbasis aus den Daten des Dauerversuchs erarbeitet. Zusätzlich wird definiert, mit welcher Vorlaufzeit vor Fehlereintritt das System eine Warnung ausgeben soll. Dann wird eine ausführliche Beschreibung des Transformer-Modells gegeben, welches als Algorithmus zur Fehlervorhersage zum Einsatz kommt. In der Evaluation folgt eine umfassende Analyse der Ergebnisse, ehe diese diskutiert werden.

Abschließend wird in Kapitel 8 das Fazit der Arbeit gezogen und ein Ausblick gegeben.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für diese Arbeit relevanten Grundlagen vorgestellt. Hierzu wird in Kapitel 2.1 die Funktionsweise eines Permanentmagnetschrittmotors erläutert, bevor in Kapitel 2.2 das Forschungsfeld der Vorhersage von Fehlerfällen präsentiert wird. In Kapitel 2.3 werden verschiedene Algorithmen des maschinellen Lernens erläutert. Anschließend wird in Kapitel 2.4 ein Einblick in das Thema Data Augmentation (DA) gegeben, bevor in Kapitel 2.5 Rekurrenzplots erläutert werden.

2.1 Permanentmagnetschrittmotor

Schrittmotoren (SM) dienen zur mechanischen Positionierung von Maschinen, wie z. B. Tintenstrahl- und 3D-Druckern, wo sie zur präzisen Positionierung des Druckkopfes oder der Plattenlaufwerke eingesetzt werden. Sie sind flexibel einsetzbar, da viele verschiedene Leistungsbereiche von wenigen mW bis hin zu einigen kW abgedeckt werden können [1]. SMs stellen eine Sonderbauform der Synchronmotoren dar und zeichnen sich dadurch aus, dass die Wicklungen über relativ hohe ohmsche Widerstände verfügen [2]. SMs kommen auch häufig in der Robotik oder in der Heiztechnik zum Einsatz. Aufgrund der stetig sinkenden Kosten für elektronische Bauteile hat sich der SM als einfaches, genaues und preiswertes Bauteil etabliert, welches jedoch in der Regel keinen Positionssensor besitzt. Die Positionsfindung erfolgt in der praktischen Anwendung über Referenzfahrten. Diese werden meist vor oder nach Betrieb des SMs ausgeführt und stellen sicher, dass dieser vollständig geschlossen oder vollständig geöffnet ist [3]. Bei Aufnahme des Betriebs wird so das Wissen über die aktuelle Position des SMs sichergestellt. Es gibt 3 Grundtypen von SMs: Permanentmagnet-SMs, Reluktanz-SMs und Hybrid-SMs [4]. Im Allgemeinen liegen die Unterschiede zwischen diesen Typen in der Art und Weise, wie das Drehmoment erzeugt wird [5]. Es wird in Lorentzkraft und Reluktanzkraft

unterschieden.

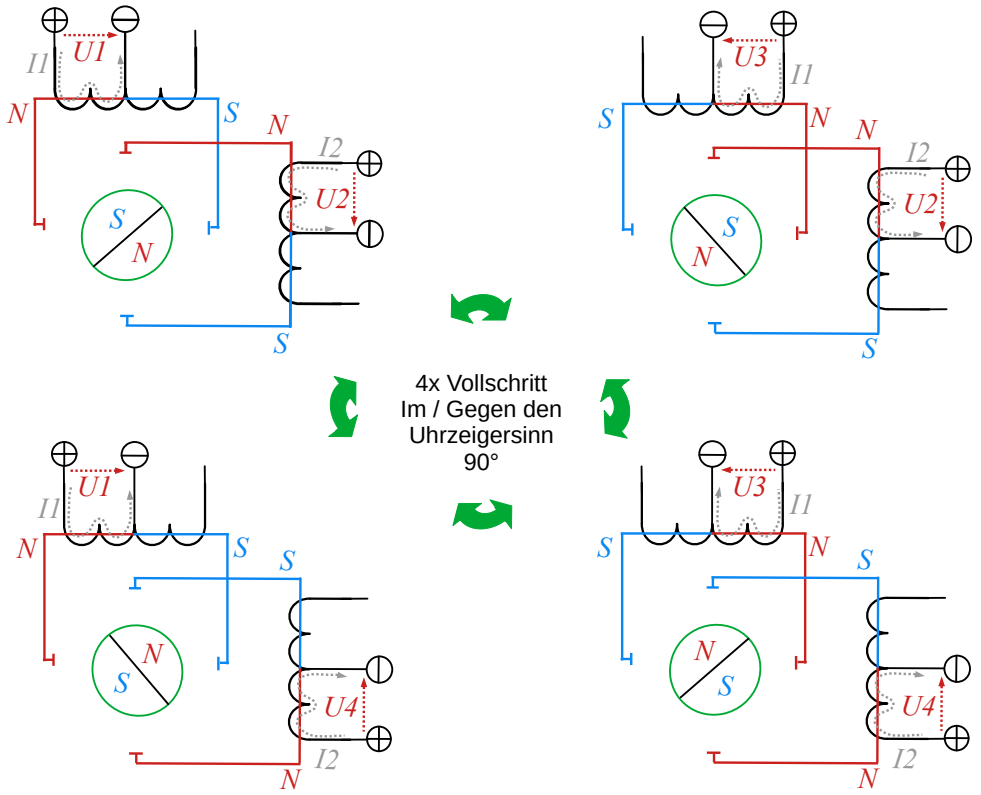

Abbildung 2.1: Funktionsweise eines unipolaren permanentmagnetischen SM.

Weiterhin wird zwischen SMs mit unipolaren und bipolaren Spulenwicklungen unterschieden. Der unipolare SM hat üblicherweise eine konstant positiv angelegte Spannung an der Mittelanzapfung und die Motorphasen sind auf ground geschaltet. Abbildung 2.1 zeigt die prinzipielle Funktionsweise eines solchen unipolaren SMs, hier dargestellt mit zwei Phasen, was auch der Typ von SM ist, der in dieser Arbeit behandelt wird. Im Folgenden wird sich daher immer auf den permanentmagnetischen, unipolaren SM mit zwei Phasen bezogen, wenn von einem SM gesprochen wird.

Um den SM zu verfahren, wird zunächst an den linken Abgriff der oberen Spule eine positive Spannung angelegt, der andere Abgriff der Spule wird inaktiv gelassen. Dies lässt den Strom $I1$ durch die erste Hälfte der Spule fließen. Die Folge ist eine messbare Spannung $U1$ und ein zweipolig magnetisierter Eisenkern. Gleichzeitig wird an der Seitenspule eine positive Spannung angelegt, die den Strom $I2$ durch diese Spulenhälfte fließen lässt. Der Effekt ist eine messbare Spannung $U2$ und ein zweipolig magnetisierter Eisenkern. Durch die Magnetisierung der Pole aus vier Richtungen, bewegt sich der Permanentmagnet-Rotor mit den daran befestig-

ten Magneten in Richtung der magnetischen Anziehungskraft. Durch Umschalten der Spannung an der oberen Spule vom linken zum rechten Abgriff (dies ist die zweite Phase), wird die Polarität des zugehörigen Eisenkerns umgekehrt. Diese erzeugt ein anderes Magnetfeld, das den Permanentmagnet-Rotor um 90° im Uhrzeigersinn drehen lässt. Beim Wiederholen des Umschaltvorgangs der Spannungen an den Spulen, dreht sich der Rotor in Vollschritten. Die Drehung kann auch gegen den Uhrzeigersinn angeregt werden, indem die Richtung der Spannung umgekehrt wird.

Es können somit vier verschiedene Betriebsmodi für den SM definiert werden

- Fahrt hoch (FH): Der SM dreht sich im Uhrzeigersinn, was diesen nach oben fahren lässt
- Fahrt runter (FR): Der SM dreht sich gegen den Uhrzeigersinn, was diesen nach unten fahren lässt
- Oberer Anschlag (OA): Der SM wird über seinen oberen mechanischen Anschlag hinaus im Uhrzeigersinn betrieben
- Unterer Anschlag (UA): Der SM wird über seinen unteren mechanischen Anschlag hinaus gegen den Uhrzeigersinn betrieben

Der normale Arbeitsbereich des SMs umfasst dabei die Betriebsmodi FH und FR. Da SMs meist keinen Positionssensor besitzen, treten in der Praxis jedoch auch häufig die Betriebsmodi OA und UA auf, welche außerhalb des normalen Arbeitsbereichs des SMs liegen.

2.2 Remaining Useful Lifetime Prediction

Das Forschungsfeld Remaining Useful Lifetime (RUL) prediction, beschäftigt sich damit, den Ausfall- oder Fehlerzeitpunkt von Bauteilen oder Gerätekomponenten vorherzusagen und somit Rückschluss auf die verbleibende Betriebszeit im ordnungsgemäßen Arbeitsbereich zu gewähren. Ein häufiger Anwendungsbereich ist es, die Lebensdauer von Batterien [6] oder Lagerschäden [7] vorauszusagen. Um dieses Ziel zu erreichen, werden Betriebsparameter mit Hilfe von Algorithmen ausgewertet und analysiert. Für eine zuverlässige Vorhersage über einen auftretenden Fehler der Betrachtungseinheit ist es daher notwendig, eine große Menge an Daten über das Betriebsverhalten der Betrachtungseinheit zu sammeln. Wichtig ist hierbei, dass Daten über verschiedene Betriebszustände vorliegen, insbesondere Daten vor Eintreten oder während des Auftretens von Fehlerfällen, damit das Modell diese mit Daten von normalen Betriebszuständen vergleichen kann und lernt, diese zu unterscheiden.

RUL prediction lässt sich dem Themenfeld Predictive Maintenance zuordnen. Predictive Maintenance, oft auch als Condition Based Maintenance, vorausschauende Instandhaltung oder vorausschauende Wartung bezeichnet, beschreibt einen zustandsbasierten Wartungs- oder Instandhaltungsvorgang. Es wird das Ziel verfolgt, Instandhaltungsmaßnahmen exakt dann zu ergreifen, wenn der qualitative Zustand der Betrachtungseinheit darauf hinweist. Predictive Maintenance grenzt sich deutlich von den bisher gezeigten reaktiven und präventiven Instandhaltungsansätzen ab und zeichnet sich durch Proaktivität aus [8].

Vorangegangene Arbeiten

Diese Arbeit konzentriert sich auf die Umsetzung von RUL prediction mit Modellen aus dem deep-learning Bereich, wobei in verwandten Arbeiten auch häufig Modelle aus anderen Forschungsfeldern, wie z.B. der Statistik [9], [10] Anwendung finden. Im Folgenden werden nun Arbeiten aufgezeigt, welche unterschiedliche deep-learning Methoden nutzen, um verschiedene Problemstellungen aus dem Bereich zu lösen.

Convolutional Neural Networks (CNNs) gelten nicht nur als eine der erfolgreichsten Methoden in der Bildverarbeitung [11], sondern wurden ebenfalls bereits erfolgreich auf Zeitreihen angewandt [12]. In [13] haben Babu et. al ein CNN als RUL-Modell vorgestellt, welches Zeitreihendaten analysiert.

Recurrent Neural Networks (RNNs) stellten einen neuen Ansatz zur RUL Vorhersage dar, indem sie langfristige zeitliche Abhängigkeiten in Daten berücksichtigen [14], unterliegen jedoch häufig dem Problem von explodierenden Gradienten [15].

Abhilfe schaffte die Einführung von Long Short-Term Memory (LSTM) Netzwerken [16], die auch das Abspeichern langfristiger Zusammenhänge erlauben. LSTMs wurden in vielfältiger Weise für prädiktive Wartungsansätze eingesetzt: Vorhersage der RUL von Flugzeugtriebwerken [17], [18], [19] von Lagern [20] oder Batterien [21], um nur einige zu nennen. Auch eine Kombination von mehreren Modellen wurde vorgeschlagen [22], [23].

2.3 Algorithmen des maschinellen Lernens

In diesem Kapitel werden verschiedene Algorithmen des maschinellen Lernens vorgestellt. Auf Grund des Fokus' dieser Arbeit, wird sich hierbei auf Klassifikationsalgorithmen oder bei Algorithmen mit vielfältigen Anwendungsmöglichkeiten, auf deren Klassifikationsfähigkeit konzentriert. Im Folgenden werden Support Vector Machines, der k-nearest neighbors Algorithmus, CNNs, Transformer sowie verschiedene Metriken zur Bewertung der Ergebnisse von Algorithmen gezeigt.

2.3.1 Support Vector Machine

Support Vector Machines (SVM) beschreiben eine Klasse weit verbreiteter, lernender Algorithmen, die sowohl zur Klassifikation als auch zur Regression eingesetzt werden können. Theoretische Grundlagen wurden bereits in der ersten Hälfte des 20. Jahrhunderts vorgestellt [24], [25]. Nachhaltige Bedeutung erlangten SVMs vor allem durch die Veröffentlichung von Vapnik et. al in den 70er [26], 80er [27] und 90er [28], [29] Jahren.

SVMs werden den Kernel-Methoden zugerechnet und finden in vielen verschiedenen Bereichen des maschinellen Lernens Anwendung. So werden diese beispielsweise im Bereich der Fehlererkennung [30], der Spracherkennung [31], [32] oder der Objekterkennung [33] genutzt. Hohe Verbreitung erlangten SVMs durch deren gute Generalisierungseigenschaften, die gute Funktionalität auch bei einer niedrigen Anzahl an Trainingsdaten im Vergleich zu anderen Algorithmen des maschinellen Lernens sowie der hohen Robustheit gegen Overfitting.

Angewandt als Klassifikator definiert die SVM eine Hyperebene, die eine vorhandene Trainingsdatenmenge in zwei Klassen unterteilt und den Abstand der Vektoren, die der Hyperebene nächstgelegen sind, maximiert [34]. Jedes Trainingsbeispiel fungiert hierbei als Vektor. Der zu maximierende Abstand wird als Margin ξ bezeichnet. Die Hyperebene ist dabei nur von denen, ihr am nächstgelegenen Vektoren, abhängig. Diese Vektoren sind ausreichend, um die Ebene mathematisch zu beschreiben und werden „Support Vektoren“ genannt. Die Funktionsweise ist in Abbildung 2.2 beispielhaft dargestellt.

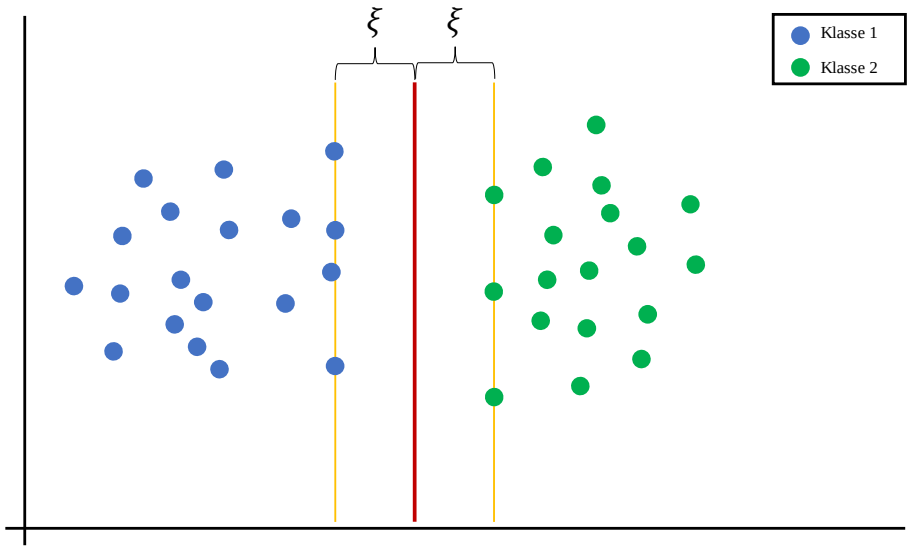

Abbildung 2.2: Funktionsprinzip einer SVM

Mit der linear separierbaren Trainingsdatenmenge $X \in \{x_1, x_2, \ldots, x_M\} \subseteq \mathbb{R}^m$

und der Klassenzugehörigkeit $y_i \in \{1, -1\}$ kann die Entscheidungsfunktion

$$D(x) = w^T x + b, \quad w \in \mathbb{R}^m, b \in \mathbb{R}$$

mit $D(x_i) > 0$, falls $y_i = 1$ und $D(x_i) < 0$, falls $y_i = -1$ aufgestellt werden. Somit gilt

$$w^T x_i + b > 0, \quad \text{für alle } i \text{ mit } y_i = 1 \tag{2.1}$$

$$w^T x_i + b < 0, \quad \text{für alle } i \text{ mit } y_i = -1. \tag{2.2}$$

Falls das Klassifizierungsproblem linear trennbar ist, so ist die zulässige Menge nicht leer und es existiert mindestens eine Lösung (w, b).

Ist die zu klassifizierende Datenmenge nicht linear, besteht der Lösungsansatz von SVMs darin, die Datenmengen in einen höher dimensionalen Raum zu transformieren, in dem sie getrennt werden können [35]. Es wird die nicht lineare Funktion $\phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_n(x))$ genutzt, die den m-dimensionalen Eingangsvektor x in einen n-dimensionalen Raum transformiert, mit $n > m$. Durch die Abbildung in einen höher dimensionalen Raum wird die Anzahl möglicher linearer Trennungen erhöht. Die Entscheidungsfunktion kann jetzt definiert werden als

$$D(x) = w^T \phi(x) + b, \tag{2.3}$$

mit $w \in \mathbb{R}^n$ und $b \in \mathbb{R}$. Hieraus resultiert jedoch, dass das quadratische Optimierungsproblem in einem hochdimensionalen Raum gelöst werden muss, was einen erheblich erhöhten Rechenaufwand zur Folge hat. Eine Möglichkeit den Rechenaufwand zu reduzieren bietet der sogenannte Kernel-Trick [35]. Dazu wird eine geeignete symmetrisch positiv semi-definite Funktion $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ mit

$$K(x_i, x_j) = \phi^T(x_i)\phi(x_j) \tag{2.4}$$

und $i, j = 1, \ldots, M$, genutzt. Je nach Datenmenge eignen sich unterschiedliche Kernelfunktionen zu deren Lösung. Nachfolgend sind drei häufig verwendete Kernelfunktionen gezeigt:

- Linearer Kernel: $K(x_i, x_j) = x_i x_j$
- Polynomialer Kernel: $K(x_i, x_j) = (x_i x_j + 1)^d$
- Radial Basis Function Kernel: $K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)$

2.3.2 k-Nearest Neighbors

Der k-nearest-neighbours (kNN) Algorithmus ist ein oft genutzter Algorithmus zur Klassifikation und Regression, wurde bereits 1951 erstmals beschrieben [36] und in den nachfolgenden Jahrzehnten stetig weiterentwickelt [37], [38], [39], [40].

Wird der kNN als Klassifikator genutzt, wird jedes Trainingsbeispiel durch einen Punkt im n-dimensionalen Raum repräsentiert. Die zu Grunde liegende Klasse der Trainingsbeispiele wird hierbei als bekannt vorausgesetzt. Soll nun die Klasse eines unbekannten Datenpunktes p bestimmt werden, wird die Klasse der k nächstliegenden, also derer mit den geringsten Abständen zu p, Trainingsbeispielen angefragt. Als Ergebnis der diskret-wertigen Klassifikation, wird die Klasse gewertet, welche bei den k nächsten Nachbarn am häufigsten vorkommt.

Das Ergebnis wird hierbei stark von der Wahl des Parameters k sowie der verwendeten Distanzmetrik beeinflusst. Abbildung 2.3 zeigt ein Beispiel für unterschiedliche k. Der Abbildung ist zu entnehmen, dass für $k = 3$ dem zu klassifizierenden Datenpunkt (gelb) die Klasse 2 (grün) zugeordnet werden würde. Für $k = 5$ jedoch die Klasse 1 (blau). Dies verdeutlicht den Einfluss von k auf das Klassifikationsergebnis.

Eine häufig verwendete Distanzmetrik ist die euklidische Distanz, welche als

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{2.5}$$

für die Berechnung der Distanz zwischen den Punkten p und q sowie der Dimension n beschrieben werden kann. p_i und q_i repräsentieren die jeweiligen Koordinaten der Datenpunkte.

Weitere, häufig verwendete Distanzmetriken sind die Mahalanobis-Distanz sowie die Chebyshev-Distanz. Der kNN-Algorithmus wird in unterschiedlichen Forschungsfeldern angewandt. So wird er beispielsweise zur Erkennung von Herzkrankheiten [41], [42], zur Texterkennung [43], [44] , für die Erkennung von Stress [45] oder zur Empfehlung für Musik [46] genutzt.

2.3.3 Convolutional Neural Network

Ein CNN wurden erstmals 1989 von LeCun et. al beschrieben [47] und die Technologie seitdem kontinuierlich weiterentwickelt [48], sowie die Berechnung auf GPUs ermöglicht [49]. Seit 2010 erlebten CNNs einen enormen Popularitätsschub, nachdem das CNN *AlexNet* einen Wettbewerb zur Erkennung von Bilddaten gewinnen konnte und dort die state-of-the-art-Methoden schlagen konnte. Krizhevsky et al. veröffentlichten *AlexNet* daraufhin [50]. Seitdem werden CNNs vor allem in der Bildklassifikation eingesetzt und zeichnen sich durch ihre convolutional (conv.)

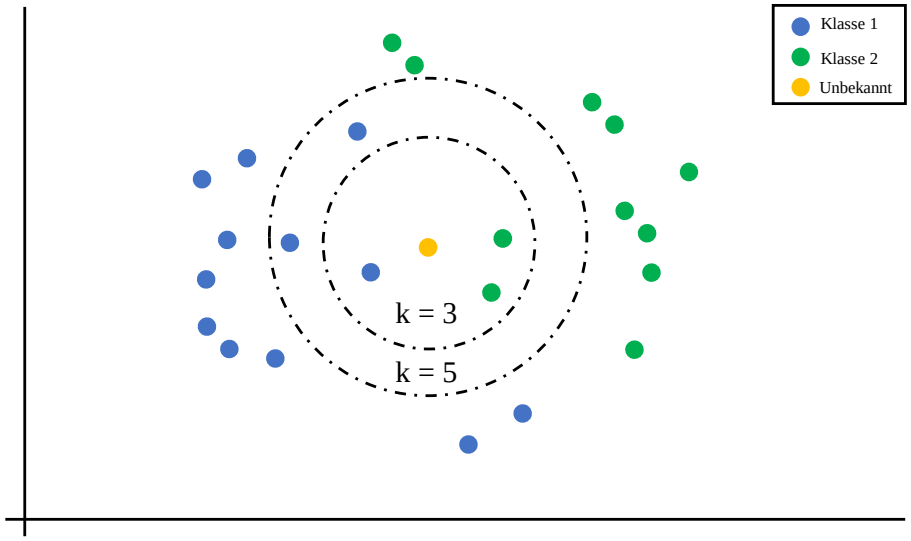

Abbildung 2.3: Funktionsprinzip des kNN-Klassifizieres

Schichten aus, in denen die Neuronen in sogenannten *feature maps* angeordnet sind. In diesen feature maps sind die Neuronen lokal verknüpft, was den Effekt hat, dass Neuronen nur kleine Teile des Inputs der vorangeganen Schicht verarbeiten. Dies führt dazu, dass jedes Neuron lernt, die für seine Region wichtigen features zu extrahieren. Die Aktivierung der feature maps kann mit

$$o^l(x,y) = \Theta\left(\sum_{x',y',c'} w^l(x',y',c') o^{l-1}(\delta^w x + x', \delta^h y + y', c') + b^l\right) \tag{2.6}$$

beschrieben werden [51]. Hierbei sind w^l die Gewichte der Faltung der Schicht l, $x', y', c' \in \mathbb{N}$ die Regionen für jedes Neuron, $\delta^w, \delta^h \in \mathbb{N}^+$ die Strides und b der Bias. Weiterhin werden in CNN-Architekturen häufig pooling-Schichten verwendet [52]. Pooling kombiniert die Ausgänge benachbarter Neuronen zu einem einzigen Wert als Input für die nächste Schicht, was zu einer Dimensionsreduktion führt. Häufig wird entweder max pooling

$$o^l(x,y,c) = \max_{x',y'} o^{l-1}(\delta^w x + x', \delta^h y + y', c), \tag{2.7}$$

min pooling

$$o^l(x,y,c) = \min_{x',y'} o^{l-1}(\delta^w x + x', \delta^h y + y', c), \tag{2.8}$$

oder average pooling

$$o^l(x,y,c) = \frac{1}{S^2} \sum_{x',y'=0}^{S-1} o^{l-1}(\delta^w x + x', \delta^h y + y', c) \tag{2.9}$$

genutzt. S^2 beschreibt die Poolingfläche, wobei S ein definierbarer Parameter ist.

2.3.4 Transformer

In diesem Kapitel werden Transformer als Klassifikatoren vorgestellt. Hierzu wird im Folgenden in Kapitel 2.3.4.1 zunächst das originale Transformer-Modell beschrieben, ehe das Embedding in Kapitel 2.3.4.2 sowie das positional encoding in Kapitel 2.3.4.3 näher erläutert werden. Nachfolgend wird in Kapitel 2.3.4.4 näher auf die self-attention, in Kapitel 2.3.4.5 die scaled dot product attention und in Kapitel 2.3.4.6 die Multi-Head Attention (MHA) eingegangen. Abschließend wird der learning rate scheduler in Kapitel 2.3.4.7 beschrieben .

2.3.4.1 Original Transformer-Modell

Transformer sind den neuronalen Netzen zuzuordnen und wurden 2017 von Vaswani et.al [53] zu Aufgaben im Bereich des Natural Language Processing (NLP) vorgestellt. Transformer beschreiben einen vollständig neuen Ansatz und heben sich von bekannten Netzstrukturen wie RNNs ab, da Sie keine Rekurrenzen haben und nur auf self-attention basieren. Der Aufbau des von Vaswani et. al vorgestellten Transformer-Modells ist in Abbildung 2.4 zu sehen.

Es ist zu erkennen, dass der Transformer aus einem Encoder und einem Decoder besteht. Der Input des Encoders ist ein Vektor, welcher nach dem Inputembedding, durch positional encoding einen Zeitstempel erhält. Der Encoder besteht aus einer MHA-, Normalisierungs- und Feed-Forward-Schicht. Der Output des Encoders hat dieselbe Dimension wie der Input und wird in die MHA-Schicht des Decoder Netzwerkes eingespeist. Während des Trainingsprozesses werden zeitgleich Zielvektoren in die Output embedding Schicht des Decoder Netzwerkes eingebettet, ebenfalls durch positional encoding mit einem Zeitstempel versehen und in den Decoder eingespeist. Dort durchlaufen die Zielvektoren zunächst eine masked-MHA-Schicht in der Teile dieser für zukünftige Iterationen des Trainingsprozesses verhüllt werden. Abgesehen von der Verhüllung ist die Schicht identisch zu der im Encoder. Anschließend folgt eine Normalisierung sowie eine weitere MHA Schicht, in die sowohl Eingangs- als auch Zielvektor zugeführt werden. Danach erfolgen weitere Normalisierungen sowie ein weitere Feed-Forward-Schicht, bevor mit Hilfe einer Softmax-Aktivierung eine linear-Schicht die Output Wahrscheinlichkeit liefert. Da in dieser Arbeit lediglich das Encoder-Netzwerk Anwendung findet, werden im Folgenden die relevanten Komponenten genauer vorgestellt.

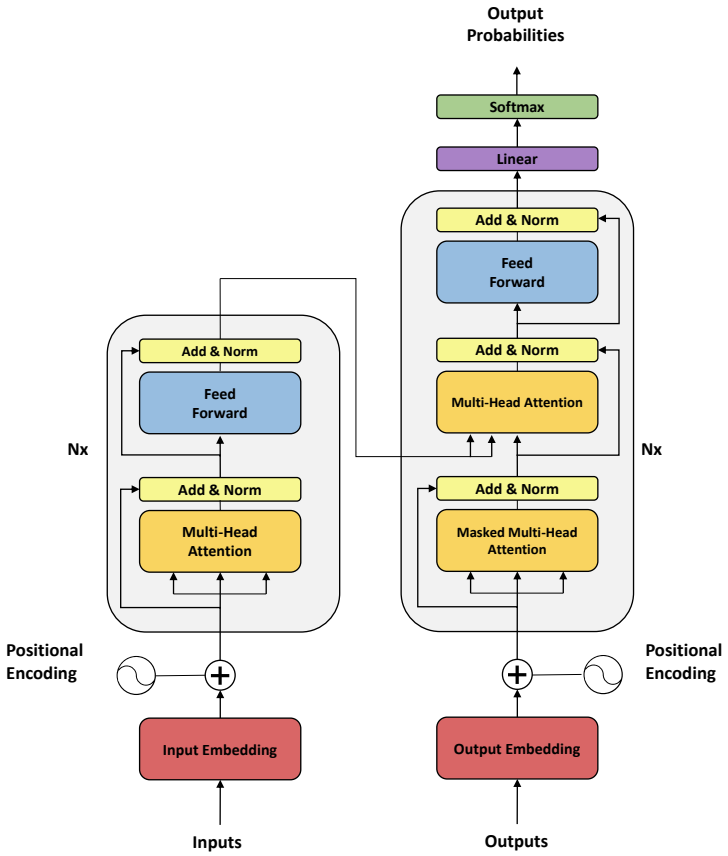

Abbildung 2.4: Schema des originalen Transformer-Modells, in Anlehnung an [53]

2.3.4.2 Embedding

Mit Embedding bei einem Transformer bezeichnet man die Transformation jedes Punktes eines Eingangsvektors in einen höher dimensionalen Raum. Die Eingangsvektoren $\mathbf{x}_i$ der Dimension $\mathbb{R}^{seq\_len}$, wobei *seq_len* der Länge der Eingangsvektoren entspricht, werden im Embedding der Dimension *d_model* zur Form $\mathbb{R}^{seq\_len \times d\_model}$.

Das Embedding ist nötig, damit Normalisierungsschichten im Transformer genutzt werden können.

2.3.4.3 Positional Encoding

Rekurrenzen in neuronalen Netzen, wie in RNNs und LSTMs, kodieren die Reihenfolge der Sequenz, die in das Modell eingegeben wird. Ein Transformer hat

jedoch keine Information über Reihenfolge oder Zeitstempel der ihm zugeführten Daten. Da die Reihenfolge von Wörtern im NLP, aber auch die zeitliche Abfolge von Zeitreihen, großen Einfluss auf das Ergebnis haben kann, wurde positional encoding eingeführt, um dem Transformer diese Informationen zuzuführen. Ein einfacher Ansatz wäre aufsteigende Integer-Werte auf jeden Eingangsvektor zu addieren, so dass für Eingangsvektoren $\mathbf{x}_j \in \mathbb{R}^{d\_model}$, mit der embedding Dimension *d_model* und der Länge $j = 1, \ldots, seq\_len$, ein neuer Eingangsvektor mit PE

$$\mathbf{x}_j + \mathbf{PE} = [x_1, \ldots, x_{seq\_len}]_j + [0, 1, \ldots, seq\_len] \qquad \forall\ j = 1, \ldots, seq\_len\ \text{-}\ 1 \tag{2.10}$$

entsteht. Dies führt jedoch dazu, dass bei langen Zeitreihen sehr hohe Werte addiert werden müssen. Dies behindert das Lernen der Transformer erheblich, da es zu verlangsamtem Training auf Grund von explodierenden Gradienten führt. Ein erster Ansatz dieses Problem zu umgehen ist es, den Bereich $[0, 1]$ in Abhängigkeit der Zeitreihenlänge zu unterteilen, was zu dem Lösungsansatz

$$\mathbf{PE} := \left[0, \frac{1}{seq\_len}, \frac{2}{seq\_len}, \ldots, 1\right] \tag{2.11}$$

führt. Dieser Lösungsansatz bringt jedoch ein neues Problem mit sich, da so jede Zeitreihe von unterschiedlicher Länge eine andere Kodierung zu Grunde liegen würde. Dies kann zu schlechteren Lernergebnissen des Modells führen. Eine Lösung für ein funktionales PE liefern ebenfalls Vaswani et. al in [53]. Die Autoren schlagen die Verwendung von alternierenden Sinus- und Cosinusfunktionen vor, denen verschiedene Periodizitäten zu Grunde liegen. Hierzu wird eine feste PE-Matrix $\mathbf{PE} \in [-1, 1]^{seq\_len \times d\_model}$ aufgestellt, welche durch

$$\begin{aligned} \mathbf{PE}_{(pos,2i)} &:= \sin(pos/10000^{2i/d\_model}) \\ \mathbf{PE}_{(pos,2i+1)} &:= \cos(pos/10000^{2i/d\_model}), \end{aligned} \tag{2.12}$$

definiert ist. Hierbei gilt $pos = 0, \ldots, seq\_len$ und $i = 0, \ldots, \lfloor \frac{d\_model}{2} \rfloor - 1$. Mit Hilfe der je nach Anwendungsfall anpassbaren Matrix, lassen sich auch Zeitreihen unterschiedlicher Länge für das Transformer-Modell verständlich kodieren. Die PE-Matrix muss in einer ausreichenden Größe initialisiert werden. Jede Dimension i entspricht einer Sinus- oder Cosinus-Funktion. Je größer i wird, desto größer wird die entsprechende Wellenlänge. Die Periodizität bildet den Bereich von 2π bis $2\pi \cdot 10000$ ab. PE-Werte, die in höheren Dimensionen i mit großer Differenz vorkommen, deuten auf Zeitreihen-Datenpunkte hin, die ebenfalls weit auseinanderliegen. Wenn die PE-Werte einen ähnlichen Wert annehmen, liegen auch die Datenpunkte der Zeitreihe nah beieinander. Darüberhinaus existieren noch weitere Methoden für ein sinnvolles PE. Einige davon werden in [54] verglichen.

2.3.4.4 Self-attention

Self-attention ist die Hauptkomponente von Transformern und das, was diese neu, einzigartig und interessant für unterschiedliche Applikationen macht [55].

Self-attention erzeugt attention-wheigts (AW), welche zwischen unterschiedlichen Zeitreihen-Datenpunkten berechnet werden. AWs geben Informationen über die Wichtigkeit der jeweiligen Datenpunkte und ihren Einfluss auf andere Datenpunkte. In den beschriebenen MHA-Schichten, wird der Eingangsvektor dahingehend untersucht und entschieden, welche Datenpunkte relevant sind und welche nicht. Die Berechnung der Attention erfolgt durch das so genannte scaled dot product attention.

2.3.4.5 Scaled dot product attention

Sei als Input der Eingangsvektor $\mathbf{x}_1, \ldots, \mathbf{x}_{seq\_len}$, mit $\mathbf{x}_i \in \mathbb{R}^{d\_model}$ für alle $i = 1, \ldots, seq\_len$, was ebenso als Matrix $\mathbf{X} \in \mathbb{R}^{seq\_len \times d\_model}$ definiert werden kann. Im Trainingsprozess werden drei Gewichtsmatrizen im Transformer initialisiert: Die Abfragegewichte, auch query weights genannt, $\mathbf{W}^{(Q)} \in \mathbb{R}^{d\_model \times key\_dim}$, die Schlüsselgewichte, auch key weights genannt, $\mathbf{W}^{(K)} \in \mathbb{R}^{d\_model \times key\_dim}$ und die Wertegewichte, auch value weights genannt $\mathbf{W}^{(V)} \in \mathbb{R}^{d\_model \times key\_dim}$. Die Tiefe aller drei Matrizen ist identisch und ein wählbarer Hyperparameter.

Der Input $\mathbf{X}$ wird mit jeder Gewichtsmatrix multipliziert und gibt *queries* $\mathbf{Q} := \mathbf{X}\mathbf{W}^{(Q)}$, die *keys* $\mathbf{K} := \mathbf{X}\mathbf{W}^{(K)}$ und die *values* $\mathbf{V} := \mathbf{X}\mathbf{W}^{(V)}$, so dass allen dieselbe Dimension (seq_len, key_dim) zu Grunde liegt. Der Lernprozess der Gewichtsmatrizen kann hierbei über eine einfache lineare Schicht erfolgen.

Die erste Rechenoperation für die Attention besteht in der Matrixmultiplikation $\mathbf{Q}\mathbf{K}^T$, oder dem Skalarprodukt zwischen q_i und k_i für alle i. Im nächsten Schritt wird ein Skalierungsfaktor installiert, der für stabilere Gradienten während der Trainingsprozesses sorgt. Schließlich erfolgt eine Normalisierung durch eine Softmax-Funktion. Diese Komponente, $\text{softmax}(\mathbf{Q}\mathbf{K}^T/\sqrt{d_k})$, wird als attention weights bezeichnet. Diese werden zu den Gewichten $\mathbf{V}$ hinzugefügt, so dass jeder Datenpunkt v_i mit einem attention score gewichtet ist. Attention lässt sich somit abschließend beschreiben als

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.13}$$

Werte mit hoher Aufmerksamkeit haben ein hohes Gewicht und bleiben bestehen, während Werte mit niedriger Aufmerksamkeit ein geringes Gewicht haben und schließlich abnehmen. Auf diese Weise kann der Transformator wichtige Informationen über die gesamte Zeitreihe hinweg lernen.

2.3.4.6 Multi-Head Attention

Ein *Attention Head* wird die Kombination von $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)}$ und $\mathbf{W}^{(V)}$ mit der scaled dot-product attention genannt. Transformer-Modelle können $h \in \mathbb{N}$ gleichzeitige Attention Heads besitzen, von diesen der i^{th} Attention Head $\mathbf{h}_i$ $(i = 1, \ldots, h)$ definiert ist als

$$\begin{aligned}\mathbf{h}_i &= \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \\ &= \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i \\ &= \text{softmax}\left(\frac{\mathbf{X}\mathbf{W}_i^{(Q)}\mathbf{X}\mathbf{W}_i^{(K)^T}}{\sqrt{d_k}}\right)\mathbf{X}\mathbf{W}_i^{(V)} \qquad \forall\ i = 1, \ldots, h, \end{aligned} \tag{2.14}$$

mit $\mathbf{h}_i \in \mathbb{R}^{seq\_len \times key\_dim} \quad \forall\ i = 1, \ldots, h$. Dies wird als Multi-Head Attention bezeichnet. Ein attention head kann attention mit dem Fokus auf ein spezifisches feature lernen. Mit einer Vielzahl von attention heads können daher auf unterschiedliche Features der Eingangsdaten geachtet werden. Im Gegensatz zu den sequenziellen Berechnungen von RNNs können Transformers diese mehreren attention heads parallel berechnen, da sie nicht voneinander abhängig sind. Dies beschleunigt das Training und ermöglicht ein hohes Maß an Skalierbarkeit.

Der Output aller h attention heads werden verkettet und bilden einen Vektor der Form $(seq\_len, key\_dim \cdot h)$. Damit gewährleistet ist, dass die Ausgangsform der Eingangsform entspricht, erfolgt eine letzte Transformation in einer abschließenden linearen Schicht durchgeführt, welche aus *d_model* versteckten Neuronen besteht. Dies entspricht einer linearen Transformation der Form

$$\Big[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_h\Big]\,\mathbf{W}_o =: \mathbf{Z} \in \mathbb{R}^{seq\_len \times d\_model} \tag{2.15}$$

mit einer lernbaren Gewichtsmatrix $\mathbf{W}_o \in \mathbb{R}^{(key\_dim \cdot h) \times d\_model}$. Der erhaltene Output $\mathbf{Z}$ hat dieselbe Form wir der MHA-input.

2.3.4.7 Learning Rate Scheduler

In Deep-Learning Modellen wird häufig ein sogenannter Learning Rate Scheduler genutzt. Dieser hat die Fähigkeit, die Lernrate von Modellen während des Trainingsprozesses zu variieren. Häufig wird dieser verwendet, um das Training mit einer hohen Lernrate zu starten, welche dann im Laufe des Trainingsprozesses langsam verringert wird [56]. Der in [53] vorgestellte Transformer nutzt den Adam-Optimizer [57] sowie einen Learning Rate Scheduler der Form

$$lrate = d\_model^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}). \tag{2.16}$$

Es ist zu sehen, dass sich die Lernrate während des Trainings ändert. Zunächst erfolgt ein linearer Anstieg für die sogenannten warm-up steps, bevor die Lernkurve entsprechend der inversen Quadratwurzel der trainings steps abfällt.

2.3.5 Variational Autoencoder

Variational Autoencoder (VAE) wurden von Kingma und Welling [58] als eine Kombination eines neuronalen netzwerkbasierten Inferenzmodells und eines neuronal netzwerkbasierten generativen Modells zur generativen Modellierung von Daten vorgeschlagen. Goodfellow et al. beschreiben VAE als ein Modell, das gelernte approximative Inferenz verwendet, die rein mit gradientenbasierten Methoden trainiert werden kann [59]. VAEs wurden bisher beispielsweise zur Erkennung von Anomalien eingesetzt [60], [61].

VAEs bestehen aus einem Encoder und einem Decoder. Hierbei existiert ein Encoder $q(z|x)$, der die wahre a posteriori Verteilung $p(z|x)$, mit der Eingabe x sowie der versteckten Repräsentation z approximiert. Es werden Trainingsdaten in den niederdimensionalen und versteckten Repräsentationsraum kodiert, dessen Parameter einer gaußschen Wahrscheinlichkeitsdichtefunktion entsprechen. Dieser Repräsentationsraum wird als *latentspace* bezeichnet. Hier liegen alle z. Das Decoder-Netzwerk, im Folgenden als $p(x|z)$ bezeichnet, nimmt eine Probe aus dem *latenspace* als Eingabe und generiert hieraus unter Berücksichtigung der Wahrscheinlichkeitsverteilung ein synthetisches Datenbeispiel. Dieser Prozess ist schematisch in Abbildung 2.5 dargestellt.

Zusätzlich zu einem Rekonstruktionsterm wird die Kullback-Leibler Divergenz (KL-Divergenz) in der Verlustfunktion verwendet, um die approximative a posteriori Verteilung $q(z|x)$ und $p(z|x)$ anzunähern. Die KL-Divergenz ist hierbei immer ≥ 0, und 0 wenn $q(z|x)$ gleich $p(z|x)$ ist. Die Optimierung wird durch Maximierung der unteren Schranke der Evidenz, auch bezeichnet als *evidence lower bound* (ELBO) erreicht, die aus einem Term zur Maximierung der Rekonstruktionswahrscheinlichkeit und der negativen KL-Divergenz besteht. Durch Minimierung der KL Divergenz kann $q(z|x)$ zu einer besseren Annäherung an die wahre Posterior-Verteilung finden. Um backpropagation nutzen zu können, verwenden VAEs einen Reparamitrierungs-Trick, der erlaubt, dass das Decoder-Netzwerk Erfahrung über seine Ergebnisse gewinnt und sich entsprechend der Trainingsdaten anpasst. Die Input-Parmeter für den Decoder werden hierbei zufällig dem latentspace entnommen.

Nach dem ein VAE trainiert worden ist, können synthetische Trainingsbeispiele erzeugt werden, in dem latentspace Beispiele entnommen und in das Decoder-Netzwerk eingespeist werden.

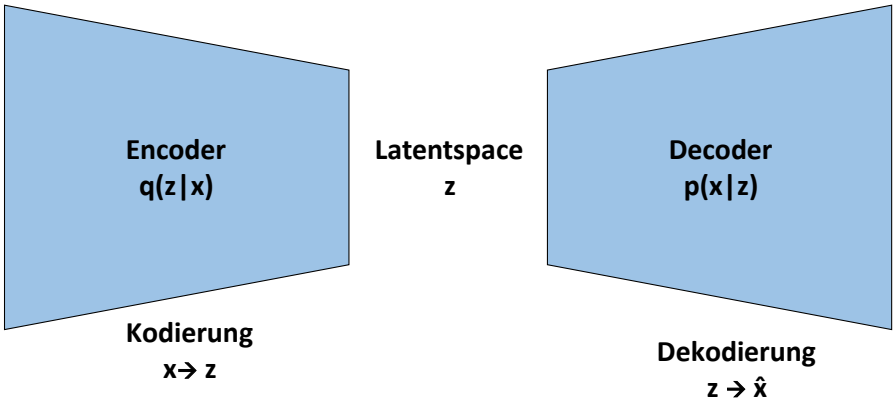

Abbildung 2.5: Funktionsprinzip eines VAE mit Encoder, Decoder und Latentspace.

2.3.6 Metriken zur Bewertung von Klassifikationsalgorithmen

Zur Bewertung von Klassifikationsalgorithmen aus dem Bereich der künstlichen Intelligenz existieren unterschiedliche Metriken um Ergebnisse der Algorithmen bewerten und diese untereinander vergleichen zu können. Die bekanntesten und häufig genutzten werden im Folgenden vorgestellt.

2.3.6.1 Accuracy

Die Accurary (Acc) beschreibt den Anteil der korrekt klassifizierten Daten eines Datensatzes im Verhältnis zum vollständigen Datensatz. Die Acc lässt sich als

$$\text{Accuracy} = \frac{\text{Anzahl richtig klassifizierte Daten}}{\text{Gesamtanzahl Daten}} \tag{2.17}$$

definieren. Selbst wenn der Algorithmus jedoch eine hohe Trainingsgenauigkeit erreicht, bedeutet dies nicht zwangsläufig, dass dieser das den Daten zugrundeliegende Muster richtig erfasst hat, da eine Über- oder Unteranpassung vorliegen kann. Aus diesem Grunde wird häufig die Confusion-Matrix genutzt um die Ergebnisse von Algorithmen genauer zu analysieren [62].

2.3.6.2 Confusion Matrix

Für binäre Klassifizierungsaufgaben ist die Confusion Matrix eine 2x2 Matrix $\mathbf{C}$, die die Anzahl der richtig vorhergesagten Proben beider Klassen (Wahr negative (TN) und Wahr Positiv (TP)) auf der Diagonalen und falschen Prädiktionen beider Klassen (Falsch Negative (FN) und Falsch Positive (FP)) auf der Gegendiagonale. Konkret wird $\mathbf{C}$ definiert als

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}, \tag{2.18}$$

wobei der Eintrag $\mathbf{C}_{i,j}$ der Anzahl der Proben entspricht, von denen bekannt ist, dass sie der Klasse i angehören und eine Klasse j Zugehörigkeit durch das Modell vorhergesagt wurde. TP und TN entsprechen einer richtigen Vorhersage der jeweiligen Klasse. Eine FP Prädiktion tritt dann auf, wenn das Modell positiv vorhersagt, die Probe jedoch negativ angehört. Dies gilt andersherum auch für FN Prädiktionen, hier wird negativ vorhergesagt, die Datenprobe gehört jedoch positiv an. Im Folgenden werden noch drei weitere häufig genutze Bewertungsmaße vorgestellt, die insbesondere Aufschluss über die Generalisierungsfähigkeit geben.

2.3.6.3 Precision, Recall und F1-Score

Um zu überprüfen, welche Anteile von positiven Vorhersagen auch wirklich positiv waren, misst die Precision den Anteil an positiven Vorhersagen, die richtig vorhergesagt worden sind. Der Recall gibt den Anteil an allen positiven Proben an, die durch das Modell korrekt vorhergesagt werden konnten [63]. Der F1-Score, welcher beide Metriken kombiniert, ist definiert als

$$\mathrm{F1} \;=\; 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}, \tag{2.19}$$

mit

$$\mathrm{Precision} \;=\; \frac{TP}{TP + FP} \;\in [0, 1], \tag{2.20}$$

$$\mathrm{Recall} \;=\; \frac{TP}{TP + FN} \;\in [0, 1], \tag{2.21}$$

und gibt oftmals aufschlussreiche Information über die wirkliche Lernfähigkeit des Modells. Besonders bei Datensätzen mit unausgewogenen Klassengrößen kommt es häufig vor, dass Klassifizierer alle Testsamples einer Klasse zuordnen und so eine hohe Acc erreichen [64]. Dieses ungewünschte Verhalten kann mit Hilfe des F1-Scores aufgedeckt werden.

2.4 Data Augmentation

Für die meisten Aufgaben des maschinellen Lernens ist ein großer Datensatz von Vorteil, da vor allem Algorithmen aus dem Bereich des deep learning für einen erfolgreichen Trainingsprozess viele Trainingsdaten benötigen. Bei kleinen Datensätzen neigen viele Algorithmen zu Overfitting und schaffen es nicht die gewonnen Informationen zu verallgemeinern. Für viele Applikationen, gerade in der Industrie, stehen jedoch nur kleine Datensätze zur Verfügung. Darüber hinaus profitieren die meisten Klassifikationsalgorithmen von Datensätzen, die über Klassen mit einer ausgewogenen Anzahl an Daten verfügen. Auch dies stellt oft ein Problem dar, da beispielsweise im Bereich der Fehlervorhersage oder Anomalieerkennung oftmals wesentlich mehr fehlerunbehaftete als fehlerbehaftete Daten vorliegen. Data Augmentation beschäftigt sich mit der Erzeugung synthetischer Daten und kann dazu beitragen, die adressierten Schwierigkeiten zu begrenzen und ist daher in den letzten Jahren ein wichtiges Forschungsfeld im Bereich des maschinellen Lernens geworden.

Vor allem im Bereich der Bildverarbeitung wurden in den letzten Jahren zahlreiche Methoden entwickelt um Datensätze mit synthetischen Daten anzureichern. So können Bilder durch einfaches Rotieren oder Spiegeln als synthetische Daten genutzt werden, ohne viel an Information zu verlieren. Auch durch Vergrößern oder Krümmen können neue Daten erzeugt werden. Komplexere Methoden durch generative Modelle wie beispielsweise General Adversarial Networks (GAN) [65], [66] werden in der Wissenschaft vor allem für die Erzeugung künstlicher Bilder, wie zum Beispiel von Verkehrskennzeichen [51] oder Werkstücken [67], genutzt.

Auch im Bereich der Sprachverarbeitung existieren viele DA-Methoden. Häufig genutzt wird eine einfache Stauchung oder Streckung der Datenprobe, was einer Verlangsamung oder Beschleunigung der Sprache entspricht [68]. Hier entsteht durch relativ einfach anzuwendende Methoden ebenfalls nur wenig Informationsverlust.

Im Gegensatz hierzu ist die Anzahl der DA-Methoden im Bereich der Zeitreihen-Klassifikation noch unterrepräsentiert. Eine große Herausforderung bei der Erweiterung von Zeitreihen-Datensätzen im Vergleich zu der Erweiterung von Bilddatensätzen besteht darin, dass einfache Methoden, wie beispielsweise das Spiegeln eines Datenbeispiels, häufig zu einer anderen Information für den Klassifikator führt und daher nicht angewandt werden können. Das Problem ist beispielhaft in Abb. 2.6 dargestellt.

Das Ausgangsbeispiel für ein Bild (a) zeigt einen Löwen. Nach der Spiegelung ist in (b) immer noch ein Löwe zu erkennen, jedoch in der gespiegelten Version von (a). Datenbeispiel (b) könnte somit weiterhin für die Klasse „Löwe“ als neues, synthetisches Trainingsmuster verwendet werden. Bild (c) zeigt als Ausgangsbeispiel

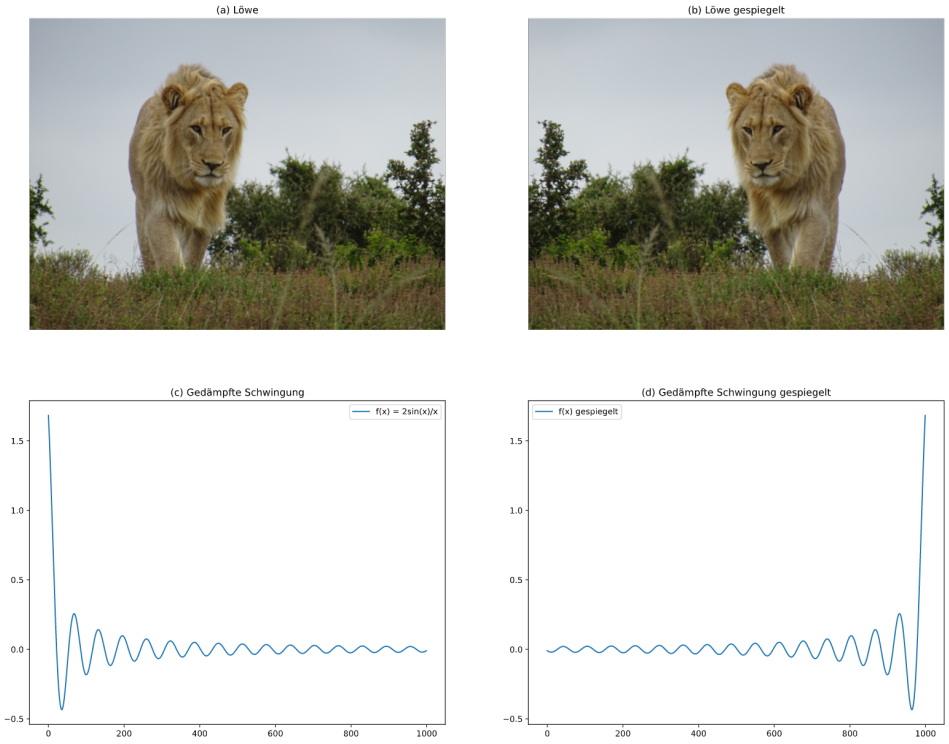

Abbildung 2.6: Beispiel für Herausforderungen von DA für Zeitreihen

für eine Zeitreihe eine gedämpfte Sinuskurve, die z.B. eine gedämpfte Bewegung eines Pendels darstellen kann. Die gespiegelte Version (d) stellt jedoch kein Pendel mehr dar, welches seine Schwingung verringert, sondern eines, das in seiner Auslenkung beschleunigt wird und ist daher als neues Trainingsbeispiel für eine abnehmende Pendelbewegung unbrauchbar.

In den letzten Jahren konnten trotzdem verschiedene Methoden von Wissenschaftlern zur Erweiterung von Zeitreihen-Datensätzen vorgestellt werden. Beispiele hierfür sind verschiedene Anwedung von Time Warping [69] oder Dynamic Time Warping [70]. Auch die Simulation von Zeitreihen stellt eine Möglichkeit dar [71], [72]. Oh et al. [73] konnten in ihrer Arbeit mit Hilfe von Interpolation die Klassifikationsgenauigkeit für sechs Datensätze erhöhen. Auch Modelle wie CNNs [74], [75] oder das T-CGAN [76] finden in der aktuellen Forschung Anwendung. Darüber hinaus gibt es Ansätze, die darauf spezialisiert sind, Klassenungleichgewichte auszugleichen. Besonders neuronale Netze als Klassifizierungsmodelle neigen dazu, den Trainingsprozess auf Klassen auszurichten, die eine signifikant größere Anzahl von Trainingsproben im Vergleich zu anderen Klassen eines Trainingssatzes haben. Das Erweitern der unterrepräsentierten Klasse mit synthetischen Proben kann helfen, die Klassifizierungsverzerrung gegenüber relativ überrepräsentierten

Klassen zu vermeiden. Ein Beispiel hierfür ist SMOTE (Synthetic Minority Oversampling Technique) [77]. Diese zeichnet sich dadurch aus, dass nicht nur die unterrepräsentierte Klasse angereichert wird, sondern auch die überrepräsentierte Klasse gezielt reduziert wird.

2.5 Rekurrenzplots

Rekurrenzplots wurden in den 1980er Jahren als eine Technik zur nichtlinearen Datenanalyse vorgestellt [78] und werden häufig in Bereichen wie Astrophysik, Geologie, Biologie, Kardiologie oder Neurowissenschaften verwendet.

Sie bieten eine Möglichkeit, Matrizen zu visualisieren, deren Elemente $\neq 0$ den Zeitpunkten entsprechen, an denen ein Zustand eines dynamischen Systems wiederkehrt. Eckmann et al. beschrieben den Prozess der Berechnung von Rekursionspunkten in [78], wobei $\vec{x_i}$ ein Punkt im Phasenraum ist, der ein dynamisches System in d Dimensionen mit insgesamt N Punkten beschreibt. Die Phasenraumtrajektorie der Zeitreihe wird erstellt mit der Zeitverzögerungseinbettungsmethode [79]. Marwan [80] erklärt die Notwendigkeit der Phasenraumrekonstruktion mit der Tatsache, dass in experimentellen Anordnungen oft nur zeitdiskrete Messungen verfügbar sind, was zu einer diskreten Zeitreihe $u_i = u(i\Delta t)$ führt, mit $i = 1, \ldots, N$ und der Abtast-Schrittweite Δt. In solchen Fällen wird der Phasenraum mit Hilfe der Zeitverzögerungsmethode

$$\hat{\vec{x_i}} = \sum_{j=1}^{m} u_{i+(j-1)\tau} \vec{e_j} \tag{2.22}$$

rekonstriert, wobei m die Einbettungsdimension und τ die Zeiverzögerung ist sowie $\vec{e_j}$ den Einheitsvektor beschreibt.

Rekurrenzplots besitzen zwei Zeitachsen und sind als

$$R_{i,j} = \Theta(\epsilon - ||\vec{x_i} - \vec{x_j}||), \quad \vec{x_i}, \vec{x_j} \in \mathrm{R}^m, \quad i, j = 1, ..., N, \tag{2.23}$$

definiert. N ist die Anzahl der betrachteten Zustände $\vec{x_i}$, ϵ der Schwellwert und $||\cdot||$ eine Norm für den betrachteten Abstand. Θ ist die Heaviside Funktion, welche 0 für Eingabewerte ≤ 0 und 1 für Eingabewerte > 0 ist.

Diese $N \times N$ Matrix, mit den Einträgen bei (i, j), wobei $\vec{x_j}$ hinreichend nahe an $\vec{x_i}$ liegt, wird als Rekurrenzplot bezeichnet. Der Schwellwertparameter ϵ definiert hierbei den zulässigen Abstand, in dem Punkte $\vec{x_j}$ als Nachbarn eines bestimmten Punktes $\vec{x_i}$ für den Rekurrenzplot betrachtet werden. Diese Punkte werden als Punkte in den Rekurrenzplot übernommen, während Punkte außerhalb der Region nicht aufgetragen werden.

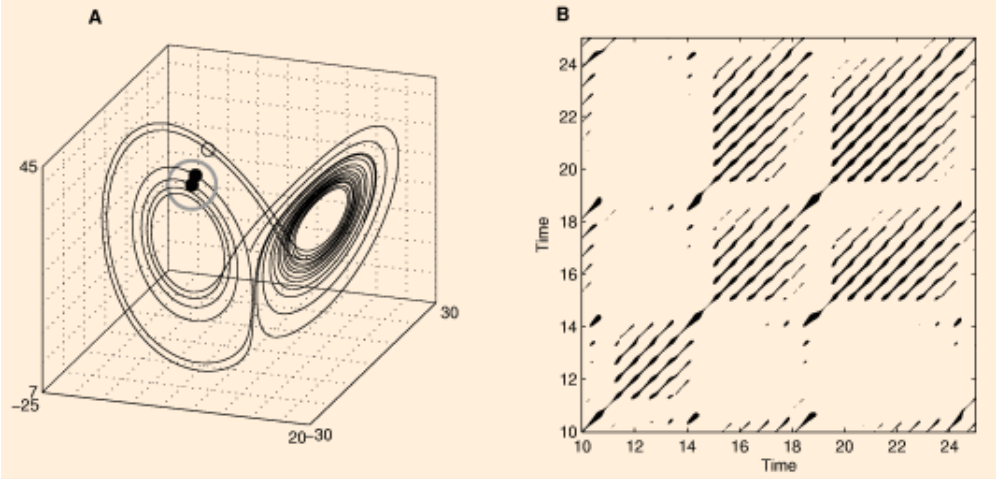

Abbildung 2.7: Phasenraumtrajektorie des Lorenz-Systems (A) mit Rekurrenzplot (B), entnommen aus [81].

Abbildung 2.7 zeigt eine Phasenraumtrajektorie des Lorenz-Systems und den daraus entstandenen Rekurrenzplot.

Ein weiterer einstellbarer Parameter ist die *step_size*. Diese legt die Anzahl der *Schritte* oder *steps* fest, d.h dem Zahlenbereich den Werte in einer Rekurrenz-Matrix annehmen können. Dies spiegelt sich in der Anwendung durch eine verbreiterte Farbpalette wieder, die über maximal *step_size* Farben verfügt.

Oft wird als Norm die euklidische Distanz zwischen den Punkten $\vec{x_i}, \vec{x_j}$ verwendet, um zu berechnen, ob der Abstand kleiner oder größer als der Schwellenwert ist. Wenn die Differenz zwischen dem Schwellenwert und dem Abstand positiv ist, wird der jeweilige Punkt für den Rekurrenzplot berücksichtigt, andernfalls liefert die Heaviside-Funktion den Wert 0 zurück.

Abbildung 2.8 zeigt die charakteristischen Typologien von Rekurrenzplots.

Ein Sinussignal, als Beispiel für schwingende Systeme, hat einen Rekurrenzplot, bei dem die Strukturen diagonal ausgerichtet sind und sich periodisch wiederholen. Weißes Rauschen führt zu einer homogenen Typologie, da es in jedem Bereich des Signals viele Zeitpunkte gibt, an denen sich Zustände wiederholen. Chaotische Rekurrenzplots können mit Signalen erstellt werden, die abrupte Änderungen in der Dynamik oder stationäre Intervalle aufweisen, wie die Brownsche Bewegung.

Die Beschaffenheit von Rekurrenzplots liefert Erkenntnisse auf einer detaillierteren Ebene und kleinräumige Strukturen wie einzelne Punkte, horizontale, vertikale oder diagonale Linien können analysiert werden. Basierend auf den beschriebenen Eigenschaften können Rekurrenzplots ein nützliches Werkzeug sein, um Einblicke in die zeitliche Entwicklung von Phasenraumtrajektorien zu gewinnen und Änderungen in der Dynamik eines Systems zu erkennen.

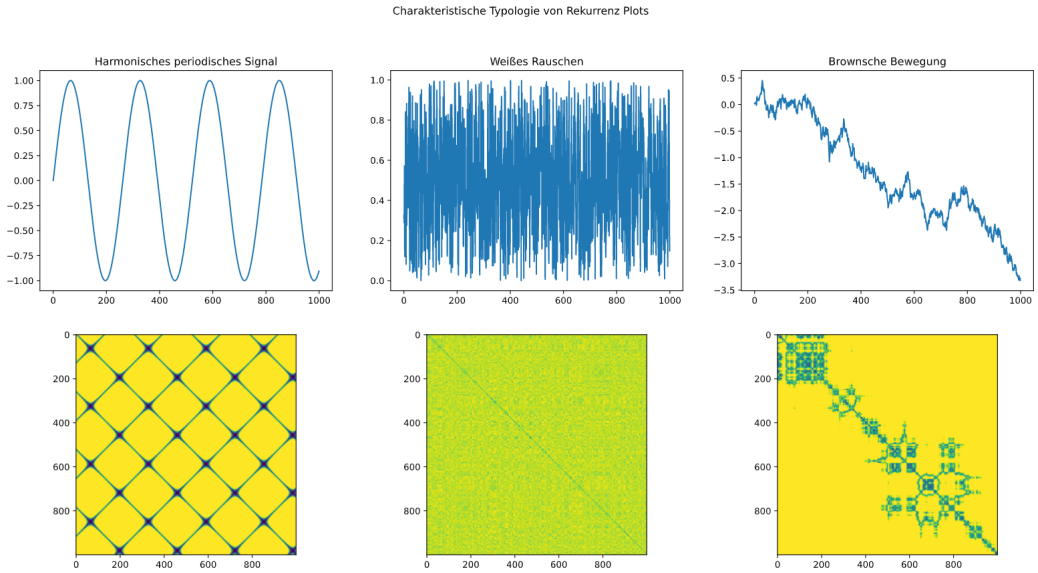

Abbildung 2.8: Charakteristische Typologien von Rekurrenzplots. Zu sehen ist links ein harmonisches, periodisches Signal, in der Mitte weißes Rauschen und rechts die Brownsche Bewegung mit ihren zugehörigen Rekurrenzplots mit Einbettungsdimension $m = 1$ (jeweils darunter abgebildet).

Teil I

Neue Verfahren zur Erweiterung und Analyse von Zeitreihen-Datensätzen

Kapitel 3

Neuartige Methoden zur Erweiterung von Zeitreihen

3.1 Problemstellung

Viele Klassifikationsalgortihmen profitieren von Datensätzen mit einer hohen Anzahl an Daten. Da in vielen Anwendungsfällen sowie in bekannten Datensätzen oftmals nicht ausreichend Trainings- oder Testsamples vorhanden sind, gibt es Ansätze die Anzahl an Daten eines Datensatzes mit Hilfe von synthetisch erzeugten Datensamples zu erhöhen. Einige davon sind in Kapitel 2.4 vorgestellt worden.

Besonders in Fällen, in denen nur kleine Datensätze für eine Anwendung zur Verfügung stehen, ist DA ein wichtiges Forschungsgebiet geworden. Im Bereich der Bildverarbeitung, aber auch in der Spracherkennung, gibt es viele unkomplizierte und einfach zu implementierende Methoden zur Datenvergrößerung. Das Drehen oder Spiegeln von Bildern zum Beispiel kann als eine informationserhaltende Transformation betrachtet werden, die in der Lage ist, auf einfache Weise Vervielfältigungen einer Trainingsmenge zu erzeugen [82].

Gerade im Bereich der Fehlervorhersage stehen oftmals lediglich kleine Datenmengen zur Verfügung oder es liegt ein Klassenungleichgewicht vor, da im Regelfall wesentlich weniger Daten von Schadfällen als Daten aus dem Normal-Betrieb gesammelt werden können. Darüber hinaus unterliegen Zeitreihendaten, die von Sensorik aufgezeichnet wurden um Fehler zu erkennen, der Herausforderung, dass viele einfache Methoden aus anderen Forschungsfeldern nicht adaptiert werden können, da diese die Ursprungsinformationen und somit die Fehlercharakteristik stark verzerren. Im Vergleich zu Bereichen wie z.B der Bildverarbeitung, in denen ein weitaus größeres Portfolio an Werkzeugen zur Erzeugung synthetischer Daten zur Verfügung steht, besteht hier ein Mangel an Methoden.

Um diesem Mangel entgegenzuwirken sind im Zuge dieser Arbeit drei neue DA-

Ansätze für Zeitreihen entwickelt und an bekannten Datensätzen mit verschiedenen Spezifika untersucht worden. Diese stehen Forschern als weitere Werkzeuge zur Anreicherung von Zeitreihen-Datensätzen zur Verfügung und tragen zur Weiterentwicklung des Forschungsgebiets bei. Darüber hinaus soll hiermit dem Anspruch der Arbeit, neuartige DA-Methoden zu erproben, genüge getan werden.

Im Folgenden werden in Kapitel 3.2 die zur Evaluierung genutzten Datensätze vorgestellt. In den Kapiteln 3.3, 3.4 und 3.5 werden anschließend die neuartigen Ansätze zur Erweiterung von Zeitreihen-Datensätzen beschrieben und evaluiert. In Kapitel 3.6 werden die Ergebnisse aller Methoden abschließend diskutiert.

3.2 Zur Evaluierung herangezogene Datensätze

Zur nachvollziehbaren und rekonstruierbaren Evaluierung der Methoden sind neun bekannte Zeitreihen-Datensätze ausgewählt worden. Alle Datensätze wurden dem UCR Time Series Classification Archive entnommen [83] und stehen online zur kostenlosen Verfügung. Die Datensätze wurden so ausgewählt, dass diese Unterschiede in den Merkmalen Klassenanzahl, Trainingsdatenmenge, Testdatenmenge und Samplelänge (SL) aufweisen, so dass eine belastbare Ergebnisdiskussion möglich ist. Tabelle 3.1 gibt eine Übersicht über die ausgewählten Datensätze und deren wichtigste Merkmale. Im Nachfolgenden wird jeder Datensatz kurz beschrieben.

Tabelle 3.1: Aufstellung der verwendeten Datensätze und den dazugehörigen Charakteristika

Datensatz	Klassen	Training Samples	Test Samples	SL
ECG200	2	100	100	96
ECG5000	5	500	4500	140
Gunpoint	2	50	150	150
Inline Skate	7	100	550	1882
Mote Strain	2	20	1252	84
SonyAIBORobotSurface1	2	20	601	70
Starlight Curves	3	1000	8236	1024
TwoLeadECG	2	23	1139	82
UWaveGestureLibraryX	8	896	3582	315

- **ECG 200**: Der ECG200 Datensatz wurde von R. Olszewski im Rahmen seiner Ph.D.-Thesis *Generalized feature extraction for structural pattern recognition in time-series data* an der Carnegie Mellon University im Jahr 2001

veröffentlicht [84]. Jede Zeitreihe repräsentiert die elektrische Aktivität, die während eines Herzschlages aufgezeichnet wird. Der Datensatz verfügt über zwei Klassen. Klasse 1 ist ein normaler Herzschlag, Klasse 2 ein Herzinfarkt.

- **ECG 5000**: Dieser Datensatz beinhaltet 5000 Samples, welche jeweils einen Herzschlag eines Patienten mit einer schweren kongestiven Herzinsuffizienz repräsentieren. Die 5000 Herzschläge wurden aus einem ursprünglich 20 Stunden langen EKG von Physionet [85] extrahiert und anschließend durch Interpolation auf dieselbe Samplelänge gebracht. Die fünf Klassen repräsentieren hierbei unterschiedliche Herzprobleme. Dieser Datensatz wurde ursprünglich in der Arbeit *A general framework for never-ending learning from time series streams* verwendet [86].

- **TwoLeadECG**: Das TwoLeadECG-Dataset ist ebenfalls aus einem bei Physionet zur Verfügung stehendem EKG-Signal entnommen. Konkret handelt es sich um Daten aus der MIT-BIH Langzeit-EKG-Datenbank (ltdb) Record ltdb/15814. Der Datensatz weist eine sehr geringe Trainingsdatenmenge von lediglich 23 Samples auf, verfügt hingegen jedoch über 1139 Testdaten. Es stehen zwei Klassen zur Verfügung.

- **StarLightCurves**: Das StarlightCurves-Dataset verfügt über insgesamt 9260 Samples mit jeweils 1024 Datenpunkten, welche die Helligkeit eines Himmelsobjekts als Funktion der Zeit beschreiben. Die Daten teilen sich in Trainingsmenge (1000 Samples) und Testmenge (8236 Samples) auf. Der Datensatz verfügt über drei Klassen, wobei jede der Klassen einen unterschiedlichen Himmelskörper repräsentiert.

- **SonyAIBRobotSurfaceX**: Der SonyAIBRobotSurfaceX (Sony) Datensatz wurde von Manuela Veloso und Douglas Vail von der Carnegie Mellon University veröffentlicht [87]. Die Daten wurden von Roll- / Neigungs- sowie Gier- und Beschleunigungssensoren eines Roboters aufgenommen. Der vorgestellte Datensatz repräsentiert nur die x-Achse der aufgezeichneten Werte. Für das Experiment wurde der Roboter zuerst über Teppich und anschließend über Zement bewegt. Die Aufgabe besteht darin, die richtige Oberfläche zu bestimmen.

- **MoteStrain**: Die Sensordaten stammen ursprünglich aus der Arbeit *Online Latent Variable Detection in Sensor Networks* [88] und wurden anschließend für eine Klassifizierung aufbereitet. Die Aufgabe besteht darin, zwischen dem Sensor q8calibHumid (einer Feuchtigkeitsmessung) und dem Sensor q8calibHumTemp (Temperaturmessung) zu unterscheiden.

- **GunPoint**: Bei diesem Datensatz handelt es sich um Zeitreihen, die die Handbewegung eines weiblichen und eines männlichen Darstellers repräsentieren. Die beiden Klassen sind Gun-Draw und Point: Bei Gun-Draw haben die Schauspieler ihre Hände an der Seite. Sie ziehen eine nachgebildete Pistole aus einem an der Hüfte befestigten Holster, richten sie etwa eine Sekunde lang auf ein Ziel, legen dann die Pistole wieder in das Holster und die Hände an die Seite. Für Point haben die Schauspieler ihre Waffe an der Seite. Sie zeigen mit ihren Zeigefingern etwa eine Sekunde lang auf ein Ziel und legen dann ihre Hände wieder an die Seite. Zur Aufnahme der Daten beider Klassen wird jeweils der Schwerpunkt der rechten Hand des Schauspielers verfolgt. Bei dem im UCR Archiv zur Verfügung stehenden Daten handelt es sich nur um die Datenpunkte der X-Achse der verwendeten Sensorik.

- **InlineSkate**: Die InlineSkate-Daten wurden von Dr. Olaf Hoos (Abteilung Sportmedizin, Universität Marburg) aus Tests im professionellen Inlineskating erhoben und in diesem Zusammenhang von Fabian Mörchen in seiner Doktorarbeit *Time Series Knowledge Mining* an der Universität Marburg verwendet [89]. Ein Sportler führte einen standardisierten Hallentest mit einer Geschwindigkeit von 7,89 m/s auf einem großen motorbetriebenen Laufband durch. Elektromyographische- und kinematische Daten der rechten Körperseite wurden 30 Sekunden lang gemessen, was 19 vollständigen Bewegungszyklen entspricht. Die elektromyographischen Daten der sieben wichtigsten Beinstrecker- und -beugemuskeln der rechten Körperseite wurden mit bipolaren Oberflächen-elektroden erfasst, die an den entsprechenden Muskeln angebracht wurden. In den vorhandenen Daten repräsentiert jede Zeitreihe eine Winkelmessung des Knöchels während eines Bewegungszyklus. Die Daten wurden auf gleiche Länge normiert.

- **UWaveGestureLibraryX**: Der UWaveGestureLibraryX (UWave) Datensatz umfasst Zeitreihen, die acht unterschiedliche Gesten repräsentieren, generiert von Beschleunigungssensoren. Die acht Gesten repräsentieren gleichzeitig die acht Klassen, zwischen denen es zu unterscheiden gilt. Die Daten bestehen aus den x-Koordinaten jeder Bewegung. Die Samplelänge einer jeden Zeitreihe beträgt 315. Der Datensatz ist in der Arbeit *uWave: Accelerometer-based personalized gesture recognition and its applications* ,im Jahr 2009 veröffentlicht worden [90].

Es ist darauf hinzuweisen, dass der Ursprung von nicht allen Datensätzen bekannt ist. Alle Datensätze sind jedoch im Bereich DA für Zeitreihen oder ähnlichen Forschungsfeldern bekannt und werden häufig genutzt.

3.3 Erweiterung von Zeitreihen-Datensätzen mit White Noise Windows

In diesem Kapitel wird die erste neu entwickelte Methode zur Erweiterung von Zeitreihen, *White Noise Windows* (WNW), vorgestellt [91]. Im Folgenden wird in Abschnitt 3.3.1 zuerst die Methodik beschrieben, ehe in Kapitel 3.3.2 die Evaluierung durchgeführt wird. Abschließend erfolgt die Diskussion der Ergebnisse in Kapitel 3.3.3.

3.3.1 Methodik

WNW ist eine einfache Methode zur Anreicherung von Zeitreihen-Datensätzen. Gegeben sei der Datensatz D, der aus dem Trainingsdatensatz D_{train} und dem Testdatensatz D_{test} besteht. Für den Trainingsdatensatz $D_{train} = \{T_1, ..., T_M\}$ wird nun zunächst jede Zeitreihe T_i normalisiert, wobei $1 \leq i \leq M$ ist. Dann wird jede Zeitreihe in D_{train} zu einer beliebigen Anzahl n von identischen Zeitreihen multipliziert, so dass der erweiterte Datensatz $D_{aug} = \{T_{1,1}, T_{1,2}, ..., T_{M,n}\}$ ist. Die Labels bleiben hierbei unangetastet. In den Experimenten dieser Arbeit wurde sich auf $n = 2$, $n = 5$, und $n = 10$ konzentriert. Im nächsten Schritt wird ein zufälliger Teil jeder Zeitreihe in D_{aug} durch weißes Rauschen ersetzt. Die Fenstergröße des Rauschens wird hierbei als Hyperparameter behandelt. In der Auswertung wurden Fenstergrößen von 5 %, 10 % und 20 % der zu augmentierenden Zeitreihe untersucht. Der Testdatensatz D_{test} bleibt bis auf die Normalisierung unverändert.

Ziel der Technik ist es, mit Hilfe der Rauschfenster die Generalisierungsfähigkeit von Klassifizieren zur steigern, da diese gezwungen werden, sich nicht nur auf kleinste Details der zur Verfügung stehenden Merkmale der Trainingsdaten zu fokussieren, sondern die vollständigen Merkmale der Eingabe zu berücksichtigen. Mit Hilfe der Vervielfältigung der Zeitreihen durch die Multiplikatornummer n soll sichergestellt werden, dass dem Klassifikator trotzdem alle Informationen jedes Trainingsbeispiels im Verlauf des Trainingsprozesses zugeführt werden. Auf beide Faktoren zahlt daher die zufällige Platzierung der Rauschfenster ein, da so, bei einer ausreichend großen Multiplikatornummer davon ausgegangen werden kann, dass annähernd alle Informationen trotz Rauschfenster bereitgestellt werden. Der Teil des Trainingsprozesses, an dem sich alle Rauschfenster der Trainingsbeispiele $\{T_{1,1}, T_{1,2}, ..., T_{1,n}\}$ exakt überlagern, wird als vernachlässigbar angenommen.

Um die Anwendung der Methode zu demonstrieren, wurde das UWave-Datenset [90] als Beispiel gewählt: Dieser Datensatz enthält Bewegungsdaten von acht verschiedenen Gesten. Der Datensatz umfasst 4478 Zeitreihen mit jeweils 315 Datenpunkten. Jede Zeitreihe repräsentiert eine Geste und ist einer von acht Klassen zugeordnet. Jede Geste stellt eine Klasse dar. Von den insgesamt 4478 Zeitrei-

hen sind 896 Trainingsdaten und 3582 Testdaten. Im ersten Schritt werden alle Zeitreihen auf Werte zwischen 0 und 1 normiert. Anschließend werden die 896 Trainingsdaten genutzt um weitere $n - 1$ identische Zeitreihen aus jeder Zeitreihe im UWave Trainingsdatensatz zu erzeugen. Dann wird ein zufälliger Teil jeder Zeitreihe durch 5 %, 10 % oder 20 % weißes Rauschen ersetzt.

Mit WNW werden so neun neue Trainingsdatensätze wovon je drei 1792, 4480 und 8960 Trainingsdaten enthalten, erzeugt, die aus den ursprünglichen 896 Trainingsdaten erstellt wurden. Für jede Trainingsdatensatzgröße existiert je ein Datensatz in dem alle Zeitreihen mit 5 % Rauschen, 10 % Rauschen und 20 % Rauschen versehen worden sind. Der Testdatensatz wird nicht verändert. Abbildung 3.1 zeigt eine Beispielanwendung von WNW auf eine zufällige Zeitreihe des UWave-Datensatzes mit verschiedenen Fenstergrößen. Für die Evaluierung werden alle neun, in Kapitel 3.2 vorgestellten Datensätze, nach demselben Schema präpariert.

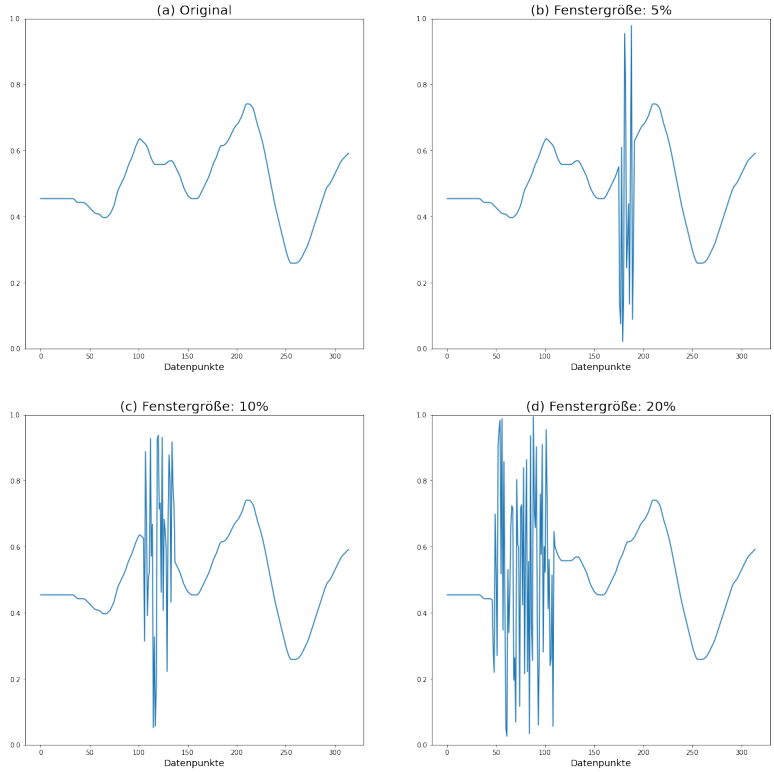

Abbildung 3.1: Beispielhafte Anwendung von WNW auf eine Zeitreihe des UWave-Datensatzes. (a) zeigt die originale Zeitreihe, (b), (c) und (d) die Zeitreihe versehen mit 5 %, 10 % oder 20 % Rauschen.

Tabelle 3.2: Evaluationsergebnisse von WNW für den kNN-Klassifizierer. Zu sehen sind die Benchmarkergebnisse sowie die Ergebnisse für $n = 2$, $n = 5$ und $n = 10$ mit einer Fenstergröße von 5 %, 10 % und 20 %.

Dataset	metric	original data	n = 2			n = 5			n = 10		
			5 %	10 %	20 %	5 %	10 %	20 %	5 %	10 %	20 %
ECG200	Acc	0.9000	0.8900	0.9000	0.8800	**0.9300**	0.9200	0.8700	0.9100	0.9200	0.8700
	F1	0.8986	0.8869	0.8989	0.8857	**0.9283**	0.9193	0.8696	0.9092	0.9191	0.8696
ECG5000	Acc	0.9391	0.9382	0.9362	0.9386	0.9311	**0.9441**	0.9380	0.9395	0.9396	0.9268
	F1	0.9292	0.929	0.9272	0.9282	0.9265	**0.9313**	0.9287	0.9290	0.9292	0.9192
TwoLeadECG	Acc	0.5981	0.6501	0.6741	0.6618	**0.7076**	0.6417	0.6777	0.6733	0.6321	0.6450
	F1	0.5626	0.6312	0.6511	0.6468	**0.6737**	0.6508	0.6454	0.6432	0.6055	0.6287
StarlightCurves	Acc	0.8451	0.8527	0.8324	0.8313	0.8492	0.8490	0.8372	**0.8568**	0.8528	0.8319
	F1	0.8596	0.8655	0.8399	0.8382	0.8543	0.8521	0.8374	**0.8601**	0.8553	0.8315
SonyAIBORobotSurface	Acc	0.4692	0.5790	0.5257	0.5474	0.6821	0.6539	0.5873	**0.6938**	0.6722	0.6405
	F1	0.3422	0.5253	0.4821	0.4797	0.6638	0.6332	0.5166	**0.6785**	0.6522	0.5872
MoteStrain	Acc	0.8514	0.8554	0.8474	0.8362	0.8658	0.8586	0.8690	**0.8849**	0.8730	0.8714
	F1	0.8502	0.8546	0.8463	0.8353	0.8652	0.8579	0.8688	**0.8846**	0.8697	0.8692
GunPoint	Acc	0.7900	0.7700	0.7833	0.7867	0.7900	0.8333	0.8133	**0.8677**	0.7933	0.7666
	F1	0.7887	0.7700	0.7829	0.7832	0.7897	0.8331	0.813	**0.8666**	0.7931	0.7663
InlineSkate	Acc	0.2255	0.2400	0.2327	0.2400	0.3109	0.2781	0.2072	**0.3309**	0.2909	0.2182
	F1	0.2274	0.2388	0.2312	0.2376	0.3098	0.2779	0.2061	**0.3340**	0.2901	0.2156
UWaveGestureLibraryX	Acc	0.7289	0.7294	0.7262	0.6842	0.7311	0.7260	0.6829	**0.7333**	0.7331	0.7018
	F1	0.6794	0.6982	0.6887	0.6223	0.7011	0.7001	0.6321	**0.7025**	0.7017	0.6485

3.3.2 Evaluierung

Für die Evaluierung wurden zwei verschiedene Klassifikatoren, kNN und SVM, aus der Python-Bibliothek für maschinelles Lernen sci-kit learn [92] implementiert. Die Funktionsweisen der Klassifizierer sind in Kapitel 2.3.1 (SVM) und 2.3.2 (kNN) erläutert. Im ersten Schritt werden die Klassifikatoren genutzt, um eine Benchmark für jeden nicht erweiterten Datensatz zu erstellen. Dann wird jeder Datensatz mit WNW gemäß dem im vorangegangenen Abschnitt beschrieben Verfahren präpariert. Anschließend werden die Klassifizierer mit den augmentierten Trainingsdatensätzen trainiert, mit denselben Testdaten getestet und die Ergebnisse mit den zuvor erstellten Benchmarks verglichen. Insgesamt werden so zehn verschiedene Ergebnisse für jeden Datensatz und Klassifikator generiert: das Benchmark-Ergebnis mit den Originaldaten, sowie neun weitere Ergebnisse, die jeweils einer anderen Fenstergröße von WNW und einer dazugehörigen Multiplikationszahl n des jeweiligen Datensatzes entsprechen. Tabelle 3.2 zeigt die Bewertungsergebnisse von kNN und Tabelle 3.3 von SVM unter Verwendung von Acc und F1-Score als Metriken.

Im Folgenden werden die wichtigsten Erkenntnisse für jeden der Klassifikatoren beschrieben:

- **kNN**

  Tabelle 3.2 zeigt, dass mit WNW signifikante Leistungsverbesserungen für

Tabelle 3.3: Evaluationsergebnisse von WNW für die SVM. Zu sehen sind die Benchmarkergebnisse sowie die Ergebnisse für $n = 2$, $n = 5$ und $n = 10$ mit einer Fenstergröße von 5 %, 10 % und 20 %.

Dataset	metric	original data	n = 2			n = 5			n = 10		
			5 %	10 %	20 %	5 %	10 %	20 %	5 %	10 %	20 %
ECG200	Acc	0.8200	0.7600	0.7400	0.6600	**0.8600**	0.8400	0.7600	0.8400	0.8300	0.7900
	F1	0.8191	0.7432	0.7222	0.5595	**0.8486**	0.8225	0.7044	0.8241	0.8105	0.7316
ECG5000	Acc	0.9044	0.9169	0.9173	0.9162	0.9202	0.9202	0.9188	0.9211	**0.9213**	0.9209
	F1	0.9032	0.8868	0.887	0.8866	0.8903	0.8903	0.8872	0.8921	**0.9041**	0.8921
TwoLeadECG	Acc	0.4707	0.4996	0.4996	0.4996	0.5057	0.4996	0.4996	**0.5856**	0.5031	0.5812
	F1	0.4701	0.3714	0.3701	0.3211	0.4464	0.4338	0.3464	**0.5436**	0.5406	0.5346
StarlightCurves	Acc	0.8468	0.8547	0.8539	0.8533	**0.8553**	0.8547	0.8542	0.8555	0.8551	0.8545
	F1	0.7933	0.7943	0.7940	0.7921	**0.7992**	0.7965	0.7912	0.7952	0.7956	0.7915
SonyAIBORobotSurface	Acc	0.5706	0.4293	0.4293	0.4293	0.5929	**0.6135**	0.4293	0.5929	0.5929	0.4293
	F1	0.5556	0.5147	0.3782	0.3091	0.5587	**0.5625**	0.3122	0.5591	0.5567	0.3255
MoteStrain	Acc	0.8182	0.8355	0.8011	0.8371	0.8083	0.8123	**0.8642**	0.8332	0.8091	0.8395
	F1	0.6556	0.8320	0.7994	0.8321	0.8031	0.8001	**0.8611**	0.8298	0.8077	0.8315
GunPoint	Acc	0.7667	0.4933	0.4933	0.4933	0.7800	**0.8067**	0.5867	0.6733	0.6467	0.7400
	F1	0.7667	0.3260	0.3260	0.3260	0.7695	**0.8042**	0.5642	0.6232	0.5938	0.7205
InlineSkate	Acc	0.1764	0.1873	0.1873	0.1873	0.1973	**0.2074**	0.1873	0.1928	0.1909	0.1891
	F1	0.1698	0.1552	0.1552	0.1552	0.1628	**0.1799**	0.1672	0.1705	0.1698	0.1677
UWaveGestureLibraryX	Acc	**0.7599**	0.6281	0.6287	0.6254	0.6304	0.6279	0.6301	0.6329	0.6326	0.6388
	F1	**0.7418**	0.5424	0.5421	0.5398	0.5485	0.5462	0.5401	0.5599	0.5591	0.5510

den kNN-Klassifikator erzielt werden können. Insgesamt zeigen alle Datensätze bessere Ergebnisse mit WNW, wenn diese mit der richtigen Parameterkombination von WNW angereichert wurden. Es lässt sich daher festhalten, dass WNW daher grundsätzlich für Zeitreihen-Datensätzen mit unterschiedlichen Charakteristika geeignet scheint. Beispielsweise kann für Datensätze mit unterschiedlichen Klassengrößen eine Verbesserung in Acc und F1-Score der Klassifikation erreicht werden. So konnte sowohl die Performance auf dem Sony Datensatz (2 Klassen) als auch auf dem UWave Datensatz (8 Klassen) verbessert worden. Weiterhin ist festgestellt worden, dass die Samplelänge keinen Einfluss auf die Anwendbarkeit von WNW hat. Dies zeigt sich in den Verbesserungen, die für den TwoLeadECG-Datensatz (SL = 82) und für den InlineSkate-Datensatz (SL = 1882) erreicht werden konnten. Auffällig ist jedoch, dass hohe Leistungsverbesserungen vor allem für Datensätzen erreicht werden konnten, die nur eine geringe Menge an Trainingsdaten zur Verfügung stellen (z. B. TwoLeadECG, MoteStrain Sony). Bei Datensätzen, die bereits viele Trainingsdaten enthalten, konnten dagegen nur geringe Verbesserungen erzielt werden.

Die Ergebnisse zeigen, dass mit einer Fenstergröße von 5 % oder 10 % die besten Ergebnisse erreicht werden konnten und diese Größen daher zu empfehlen sind. Zu große Rausch-Fenster können zu Verschlechterungen in der Acc. führen. Dies kann durch Betrachtung des F1-Scores bestätigt werden,

der bei einer Fenstergröße von 20 % die schlechtesten Ergebnisse aufweist. Außerdem liefern $n = 5$ oder $n = 10$ im Vergleich zu $n = 2$ bessere Ergebnisse. Bei der Parameterkombination von $n = 10$ und einer Fenstergröße von 5 % wurden in sechs von neun Fällen die besten Ergebnisse erzielt. Insgesamt konnte durch Anwendung von WNW die Acc für alle Datensätzen gesteigert werden, wenn eine geeignete Parameterkombination verwendet wurde. Besonders signifikante Verbesserungen der Genauigkeit konnten für den TwoLeadECG (Acc +10.95 %) und den Sony (Acc +22.46 %) Datensatz erreicht werden.

Beachtet werden muss, dass sich WNW auch negativ auswirken kann, wenn die Fenstergröße zu groß oder n zu klein gewählt wird. Die Leistung auf dem UWave-Datensatz war beispielsweise deutlich reduziert (Acc -4:47 %), wenn $n = 2$ und eine Fenstergröße von 20 % gewählt worden ist. Es ist daher zu empfehlen, WNW für den kNN-Klassifikator mit einer Fenstergröße von 5 % und $n = 10$ anzuwenden. Für die Vergrößerung eines Datensatzes mit ECG-Daten ist $n = 5$ ebenfalls zu empfehlen, da mit dieser Einstellung das beste Ergebnis für alle drei ECG-Datensätze erreicht werden konnte.

- **SVM**

  Auch die Performance der SVM kann mit WNW verbessert werden. Tabelle 3.3 zeigt, dass bei acht von neun Datensätzen ein besseres Ergebnis mit WNW erzielt werden konnte. Hier ist auffällig, dass WNW die Leistung für den UWave Datensatz, welcher der untersuchte Datensatz mit den meisten Klassen ist, in allen Experimenten verschlechtert. Auch für den InlineSkate-Datensatz (7 Klassen) wurde bei vielen Experimenten eine Verschlechterung des F1-Scores verzeichnet. Die Länge der Zeitreihe scheint hingegen nicht entscheidend für die Leistungsentwicklung zu sein. Verbesserungen wurden sowohl für die Datensätze ECG200 (SL = 96) als auch StarlightCurves-Datensatz (SL = 1024) erreicht. Wie für den kNN wird auch für die SVM eine Fenstergröße von 5 % oder 10 % empfohlen, da Fenster der Größe 20 % oftmals zu einer Verschlechterung geführt haben.

  Insgesamt sind leichte Verbesserungen für die meisten Auswertungen festgestellt worden, jedoch ist auch eine signifikante Verschlechterung der Leistung für den UWave-Datensatz in Genauigkeit und F1-Score beobachtet worden. Ein möglicher Grund ist die hohe Anzahl von Klassen. Es fällt schwerer eine Parameterempfehlung für den SVM-Klassifikator zu geben als für den kNN-Klassifikator, da kein eindeutiger Trend beobachtet werden konnte. Ein Indiz ist, dass mit n = 5 und einer Fenstergröße von 5 % bzw. 10 % das beste Ergebnis für 5 von 9 der untersuchten Datensätzen erreicht werden konnte.

Insgesamt wurde für meisten Evaluations-Tests eine Verbesserung erreicht.

WNW kann besonders für den kNN-Klassifikator nützlich sein. Hier konnte jedes Benchmark-Ergebnis durch WNW verbessert werden, wenn die Parameter gut gewählt sind. Die Auswertungsergebnisse zeigen, dass eine Fenstergröße von 5 % oder 10 % sinnvoll ist. Es wird angenommen, dass bei einer Fenstergröße von 20 % zu viele wichtige Informationen verloren gehen. Diese These wird gestützt durch den oft geringeren F1-Score für einer Fenstergröße von 20 % im Gegensatz zu Fenstergrößen von 5 % und 10 %. Außerdem sollte n nicht zu klein gewählt werden, da mit $n = 2$ in keiner Aufgabe das beste Ergebnis erzielt werden konnte. In einigen Fällen gab es sogar schlechtere Ergebnisse. WNW scheint daher grundsätzlich geeignet für Datensätze mit unterschiedlichen Stichprobenlängen, Klassengrößen und Klassenanzahlen. Allerdings sollte WNW vorsichtig eingesetzt und die Leistung des Klassifikators genau überwacht werden, da WNW auch einen negativen Einfluss auf die Acc sowie den F1-Score haben kann. Diesem Verhalten kann in den meisten Fällen jedoch durch die Untersuchung verschiedenen Parameterkombinationen vorgebeugt werden.

3.3.3 Diskussion

Die Evaluation hat gezeigt, dass WNW helfen kann, die Leistung von kNN und SVM zu verbessern und Overfitting zu vermeiden. In den meisten Evaluierungsaufgaben wurde eine Verbesserung der Genauigkeit und des F1-Score durch WNW erreicht. Auf Grund des oftmals gesteigerten F1-Scores kann festgehalten werden, dass die Generalisierungsfähigkeit der Klassifikatoren durch WNW angehoben werden kann.

Für die untersuchten Klassifikatoren zeigt die Auswertung, dass insbesondere der kNN Klassifikator von WNW profitiert, während leichte Leistungsverbesserungen für SVM erzielt werden konnten. Die Anwendung von WNW für Datensätzen mit vielen Klassen scheint nicht sinnvoll wenn die SVM als Klassifikator eingesetzt werden soll.

Für die Parameter Einstellung von WNW hat sich gezeigt, dass eine Fenstergröße von 5 % oder 10 % in Kombination mit $n = 5$ oder $n = 10$ über beide Klassifikatoren hinweg die besten Ergebnisse liefert. Für den kNN-Klassifikator lässt sich dies auf $n = 10$ mit einer Fenstergröße von 5 % präzisieren. Diese Einstellung erzielte die besten Ergebnisse in Genauigkeit und F1-Score in sechs von neun Fällen und erzielte auch das höchste Durchschnittsergebnis. Es ist nicht ratsam, eine zu große Fenstergröße oder einen zu kleinen Wert für n zu wählen. Dies lässt sich damit begründen, dass die Ergebnisse für $n = 2$ und eine Fenstergröße von 20 % im Vergleich zu den anderen getesteten Parameter Einstellungen abfallen. Es ist anzunehmen, dass bei Fenstergrößen von 20 % dem Klassifikator zu viele Informationen der Trainingsdaten vorenthalten werden. Für $n = 2$ gilt dies ebenfalls, da nur eine ausreichend große Multiplikatorzahl dazu führt, dass dem

Klassifikator im Trainingsprozess trotz der Rauschfenster alle Merkmale der Trainingsdaten zur Verfügung gestellt werden. Treten diese Faktoren in Kombination auf (große Rauschfenster, kleines n), ist es daher erklärbar, dass die Leistung der Klassifikatoren abnehmen.

Es ist daher zu beachten, dass trotz der vielfach positiven Ergebnisse auch erhebliche Leistungsverschlechterung durch die Verwendung von WNW auftreten können. Deshalb ist es für die erfolgreiche Anwendung von WNW erforderlich, verschiedene Parameterkombinationen zu prüfen und die Entwicklung der Leistung des verwendeten Klassifikators genau zu beobachten. Für weitere Experimente kann es interessant sein, WNW auf Trainingsdaten für Algorithmen aus dem deep-learning Bereich anzuwenden.

3.4 Erweiterung von Zeitreihen-Datensätzen mit Variational Autoencoder

In diesem Kapitel werden VAEs als neuartige Methode zur Erweiterung von Zeitreihen-Datensätzen auf Basis der bereits in [93] veröffentlichten Ergebnisse präsentiert. Hierzu wird in Abschnitt 3.4.1 zuerst die Methodik vorgestellt ehe in Kapitel 3.4.2 die Evaluation erfolgt. Abschließend werden die Ergebnisse in Kapitel 3.4.3 diskutiert.

3.4.1 Methodik

Die grundlegende Funktionsweise von VAEs ist in Kapitel 2.3.5 beschrieben. Für die Experimente dieser Arbeit wurden sowohl Encoder- als auch Decoder-Netzwerke mit Tensorflow implementiert, wobei fully connected (FC) Schichten mit ReLu-Aktivierung verwendet wurden.

Zunächst werden Zeitreihen aus dem Trainingsdatensatz in das Encoder-Netzwerk eingespeist, welches eine multivariate Normalverteilung mit den Parametern μ und σ zurückgibt, die aus der letzten versteckten Schicht gewonnen werden. Die Varianz wird unter Verwendung der Softplus-Aktivierung als nicht-negativ parametrisiert. Die Prior-Verteilung ist als eine Gaußsche Einheitsverteilung und der latentspace ist als 2-dimensional definiert. Das symmetrisch zum Encoder aufgebaute Decoder-Netzwerk nimmt zweidimensionale Samples aus dem latentspace als Eingabe. Diese wird durch die versteckten Schichten des Decoder geführt, welche dieselbe Größe wie die versteckten Schichten des Encoder haben. Unter Verwendung des Adam-Optimierers [57] wird ELBO durch Minimierung von -(ELBO) maximiert. Abb. 2 und Abb. 3 zeigen Beispiele aus dem ursprünglichen StarLightCurves-Datensatz sowie daraus generierte Daten mit dem VAE.

(a) Originale Zeitreihen

(b) Generierte Zeitreihen

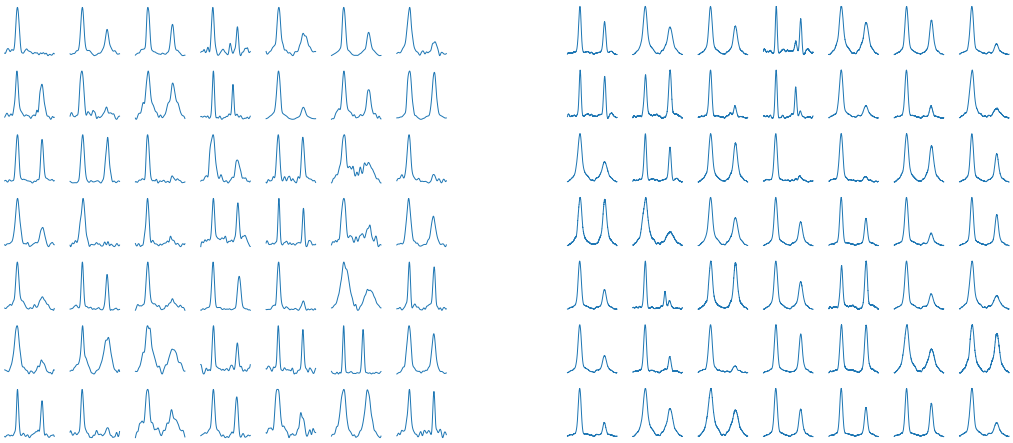

Abbildung 3.2: Beispielhafte Darstellung von synthetisch erzeugten Daten mit einem VAE anhand des StarLightCurves-Datensatzes. (a) zeigt einen Ausschnitt der originalen Zeitreihen, (b) zeigt die synthetisch erzeugten Zeitreihen.

Mit Hilfe der Trainingsdatenmenge des zu Grunde liegenden Datensatzes wird der VAE trainiert. Anschließend werden Datenpunkte zur Erstellung der synthetischen Trainingsdatensätze aus dem Latenspace entnommen und in das Decoder-Netzwerk eingespeist. Das Ergebnis ist ein erweiterter Trainingsdatensatz, der n-mal so groß ist wie der des ursprünglichen Trainingsdatensatz, wobei der ursprüngliche Trainingsdatensatz selbst sowie synthetische Trainingsdaten mit der $n-1$-fachen Anzahl des ursprünglichen Trainingsdatensatzes aufgenommen werden. Die Testdatensätze werden nicht verändert.

Zu beachten ist, dass der VAE für jede Klasse separat trainiert wird, so dass nur synthetische Beispiele der jeweiligen Klasse des Trainingsprozess erzeugt werden. Dieser Prozess wird für alle Klassen nach demselben Muster vollzogen. Anschließend werden die separat erzeugten Trainingsdaten jeder Klasse zusammengefasst. Die Proportionen der Klassen bleiben in den augmentierten Datensätzen gleich, da jeder Klasse eines Datensatzes der gleiche Parameter n zu Grunde liegt.

In den Experimenten dieser Arbeit wurden drei unterschiedliche VAE Architekturen verwendet, die sich in der Anzahl der Schichten im Encoder- und Decoder-Netzwerk unterscheiden.

Das erste Modell hat zwei versteckte Schichten in jedem Netzwerk, wobei die Anzahl der Neuronen in jeder Schicht der halben Länge der Trainingssamples des jeweiligen Datensatzes in jeder Schicht entsprechen. Das zweite Modell hat zwei

versteckte Schichten in jedem Netzwerk, in denen die Anzahl der Neuronen der Hälfte der Länge des Inputs in der ersten Schicht entspricht und in der zweiten Schicht des Netzwerks nochmals halbiert wird. Das dritte Modell hat eine versteckte Schicht in jedem Netzwerk, wobei die Anzahl der Neuronen gleich der halben Länge eines Eingabebeispiels ist.

Für jeden getesteten UCR-Datensatz wurden drei augmentierte Trainingssätze mit einer Multiplikationszahl von $n = 2$, $n = 5$ und $n = 10$ erzeugt.

3.4.2 Evaluation

Für die Evaluation werden die zwei Klassifikationsalgorithmen kNN und SVM, sowie die neun in Kapitel 3.2 vorgestellten Datensätze verwendet. Die Benchmark-Ergebnisse in Acc und F1-Score für diese Klassifikation entsprechen daher den Benchmark-Ergebnissen aus Kapitel 3.3.

Zunächst werden die drei unterschiedlichen VAEs genutzt, um die für die Auswertung verwendeten Datensätze entsprechend der Erläuterung im vorangegangenem Kapitel vorzubereiten. Im Folgenden werden die erweiterten Datensätze zum Trainieren von SVM und kNN genutzt um anschließend die Ergebnisse an Hand der unveränderten Testdatensätzen mit den zuvor erstellten Benchmarks zu vergleichen. Zusammengefasst gibt es zehn verschiedene Ergebnisse für jeden Datensatz und Klassifikator: Die Benchmark-Ergebnisse, die mit den unveränderten Trainingsdatensätzen erstellt wurden, sowie neun zusätzliche Ergebnisse, die jeweils einer der drei VAE-Architekturen und einer unterschiedlichen Multiplikationszahl n des jeweiligen Datensatzes zugeordnet sind. Tabelle 3.4 zeigt die Bewertungsergebnisse von kNN und Tabelle 3.5 von SVM unter Verwendung von Acc und F1-Score als Metriken. Im Folgenden werden die wichtigsten Erkenntnisse für jeden der Klassifikatoren erläutert:

3.4.2.1 kNN

Mit den erweiterten Datensätzen können Verbesserungen für die meisten Tests unter Verwendung des kNN als Klassifikator erzielt werden. Mit $n = 5$ oder $n = 10$ konnten die besten Ergebnisse erzielt werden, $n = 2$ fällt im Vergleich dazu ab. Diese Beobachtung wurde für alle drei verwendeten Architekturen gemacht. Für vier Datensätze (StarLightCurves, Sony, GunPoint, InlineSkate) konnten die Ergebnisse in Acc und F1-Score in jedem Experiment gesteigert werden.

Im Hinblick auf die unterschiedlichen Eigenschaften der Datensätze konnte festgestellt werden, dass die Samplelänge keinen Einfluss auf die grundsätzliche Funktionalität der VAEs als Datenerweiterungsmethode zu haben scheint, da Verbesserungen sowohl für Datensätze mit langen Samples (z.B. Inlineskate, SL =

Tabelle 3.4: Evaluationsergebnisse für den kNN. Zu sehen sind die Benchmarkergebnisse sowie die Ergebnisse für die drei verwendeten Architekturen mit $n = 2$, $n = 5$ und $n = 10$.

Dataset	metric	original data	Architecture 1			Architecture 2			Architecture 3		
			$n = 2$	$n = 5$	$n = 10$	$n = 2$	$n = 5$	$n = 10$	$n = 2$	$n = 5$	$n = 10$
ECG200	Acc	0.9000	0.9000	0.9020	**0.9120**	0.9020	0.9040	0.9060	0.8980	0.9080	0.9060
	F1	0.8986	0.8986	0.8998	**0.9055**	0.8998	0.9010	0.9020	0.8965	0.9032	0.9022
ECG5000	Acc	0.9391	0.9389	0.9391	0.9395	0.9368	**0.9405**	0.9397	0.9389	0.9379	0.9398
	F1	0.9292	0.9288	0.9286	0.9293	0.9288	**0.9307**	0.9298	0.9288	0.9291	0.9306
TwoLeadECG	Acc	0.5981	0.5882	**0.6253**	0.5979	0.5940	0.6047	0.6116	0.5960	0.5961	0.5979
	F1	0.5626	0.5833	**0.5969**	0.5626	0.5709	0.5712	0.5798	0.5667	0.5669	0.5626
StarlightCurves	Acc	0.8451	0.8467	0.8492	0.8483	0.8459	0.8481	0.8542	0.8457	0.8481	**0.8564**
	F1	0.8596	0.8490	0.8643	0.8616	0.8605	0.8480	0.8549	0.8480	0.8485	**0.8628**
SonyAIBORobotSurface	Acc	0.4692	0.5048	0.5325	**0.5648**	0.4709	0.4848	0.4986	0.4788	0.4889	0.5109
	F1	0.3422	0.4080	0.4614	**0.5160**	0.3454	0.3721	0.3987	0.3641	0.3721	0.3987
MoteStrain	Acc	0.8514	0.8482	0.8557	**0.8580**	0.8477	0.8516	0.8537	0.8474	0.8496	0.8579
	F1	0.8502	0.8465	0.8540	**0.8563**	0.8460	0.8499	0.8520	0.8457	0.8479	0.8562
GunPoint	Acc	0.7900	0.8027	0.8053	0.8027	0.0828	0.8096	0.8053	0.8013	0.8027	**0.8107**
	F1	0.7887	0.7991	0.8020	0.8018	0.8020	0.8089	0.8047	0.8004	0.8018	**0.8100**
InlineSkate	Acc	0.2255	0.2273	0.2284	0.2287	0.2280	0.2289	**0.2291**	0.2255	0.2280	0.2287
	F1	0.2274	0.2295	0.2306	0.2311	0.2303	0.2312	**0.2314**	0.2274	0.2303	0.2310
UWaveGestureLibraryX	Acc	0.7289	0.7267	0.7291	**0.7318**	0.7224	0.7291	0.7301	0.7244	0.7264	0.7281
	F1	0.6794	0.6794	0.6898	**0.7050**	0.6705	0.6898	0.6891	0.6726	0.6746	0.6998

Tabelle 3.5: Evaluationsergebnisse für die SVM. Zu sehen sind die Benchmarkergebnisse sowie die Ergebnisse für die drei verwendeten Architekturen mit $n = 2$, $n = 5$ und $n = 10$.

Dataset	metric	original data	Architecture 1			Architecture 2			Architecture 3		
			$n = 2$	$n = 5$	$n = 10$	$n = 2$	$n = 5$	$n = 10$	$n = 2$	$n = 5$	$n = 10$
ECG200	Acc	0.8200	0.8700	0.8760	0.8880	0.8700	0.8780	0.8880	0.8600	0.8800	**0.8900**
	F1	0.8191	0.8687	0.8752	0.8862	0.8676	0.8769	0.8862	0.8580	0.8792	**0.8880**
ECG5000	Acc	0.9044	0.9377	0.9352	0.9336	0.9385	**0.9394**	0.9350	0.9391	0.9362	0.9384
	F1	0.9032	0.9269	0.9238	0.9219	0.9279	**0.9300**	0.9236	0.9286	0.9250	0.9298
TwoLeadECG	Acc	0.4707	0.5882	0.6372	0.6699	0.5991	0.6481	0.6645	0.5994	0.6444	**0.6699**
	F1	0.4701	0.5833	0.6272	0.6564	0.5931	0.6366	0.6515	0.5935	0.6337	**0.6564**
StarlightCurves	Acc	0.8468	0.8593	0.8938	0.9356	0.8596	0.8918	0.9343	0.8602	0.8924	**0.9365**
	F1	0.7933	0.7994	0.8772	0.9356	0.8003	0.8742	0.9322	0.8017	0.8755	**0.9347**
SonyAIBORobotSurface	Acc	0.5706	0.6878	0.7754	0.7045	0.6955	0.7720	0.6689	0.7022	**0.7787**	0.7178
	F1	0.5556	0.6705	0.7711	0.6880	0.6800	0.7675	0.6573	0.6879	**0.7748**	0.7045
MoteStrain	Acc	0.8182	0.8187	0.8458	0.8586	0.8179	0.8474	0.8578	0.8195	0.8530	**0.8594**
	F1	0.6556	0.8168	0.8453	0.8586	0.8160	0.8470	0.8586	0.8177	0.8529	**0.8595**
GunPoint	Acc	0.7667	0.7867	0.8133	0.8240	0.7867	0.8212	0.8240	0.7840	0.8067	**0.8267**
	F1	0.7667	0.7866	0.8132	0.8214	0.7862	0.8200	0.8240	0.7838	0.8066	**0.8267**
InlineSkate	Acc	0.1764	0.2164	0.2491	0.2582	0.2164	0.2493	**0.2582**	0.2236	0.2509	0.2462
	F1	0.1698	0.2129	0.2438	0.2550	0.2121	0.2451	**0.2561**	0.2202	0.2445	0.2426
UWaveGestureLibraryX	Acc	0.7599	0.7398	0.7635	0.7694	0.7395	0.7856	0.7736	0.7451	**0.7859**	0.7728
	F1	0.7418	0.7508	0.7539	0.7547	0.7501	0.7819	0.7594	0.7563	**0.7818**	0.7585

1882) als auch für kurze Samples (z.B. ECG200, SL = 96) erreicht werden konnten. Es ist zu erkennen, dass besonders hohe Verbesserungen für Datensätze mit wenigen Trainingsdaten (z. B. TwoLead, Sony) erreicht werden konnten.

Beim Vergleich der Ergebnisse der verschiedenen Architekturen ist auffällig, dass Architektur 1 die besten Ergebnisse liefert. Für fünf von neun Datensätzen konnte das beste Ergebnis mit Architektur 1 erzielt werden. Zusätzlich wurde das Benchmark Ergebnis nur in wenigen Fällen nicht erreicht. Alle Trainingsdatensätze, bei denen das Benchmark-Ergebnis mit Architektur 1 nicht erreicht werden konnte, basieren auf $n = 2$. Mit $n = 10$ konnte für jeden Datensatz ein verbessertes Ergebnis erzielt werden. Auch für die Architekturen 2 und 3 wurden die besten Ergebnisse mit $n = 10$ erzielt. Architektur 2 weist ebenfalls sehr wenige Experimente auf, bei denen Verschlechterungen in Acc oder F1-Score aufgetreten sind. Auch lag allen aufgetretenen Verschlechterungen $n = 2$ zu Grunde. Die Datensätze, die mit $n = 5$ oder $n = 10$ Trainingsdaten angereichert wurden, zeigen gute Ergebnisse, wenn gleich diese im Vergleich zu Architektur 1 leicht abfallen. Bei Architektur 3, treten auch für $n = 5$ Verschlechterungen auf. Mit $n = 10$ konnten hingegen auch mit Architektur 3 die Ergebnisse für alle Datensätze, außer das des UWave-Datensatzes, verbessert werden.

Insgesamt kann gefolgert werden, dass VAEs sich als Modell eignen, um Zeitreihen-Datensätze mit synthetischen Samples anzureichern, die mit dem kNN klassifiziert werden sollen. Diese Aussage stützt sich auf die überzeugenden Ergebnisse, die mit den angereicherten Datensätzen im Vergleich zu den Benchmark-Datensätzen erreicht werden konnten. Die Ergebnisse zeigen, dass Architektur 1 die beste Anpassung für kNN in Kombination mit den verwendeten Datensätzen ist. Es ist deutlich zu sehen, dass mit $n = 10$ sehr gute Ergebnisse erzielt werden können und fast keine Verschlechterung eintritt. Architektur 1 in Kombination mit $n = 10$ kann daher als Datenanreicherungsmethode für vergleichbare Zeitreihen-Datensätze empfohlen werden.

3.4.2.2 SVM

Insgesamt lässt sich sagen, dass sehr gute Ergebnisse für die SVM erzielt werden konnten. Die Ergebnisse der angereicherten Datensätze konnten die Benchmark-Ergebnisse in fast allen Fällen übertreffen. Nur für den UWave-Datensatz wurde das Benchmark-Ergebnis in drei Experimenten nicht erreicht. Es ist zu erkennen, dass sehr gute Ergebnisse für $n = 5$ und $n = 10$ erzielt wurden. Im Vergleich fallen die Ergebnisse, die mit $n = 2$ erreicht wurden, ab. Dies bestätigt die Beobachtungen, die mit dem kNN gemacht wurden.

Im Hinblick auf die unterschiedlichen Eigenschaften der Datensätze kann festgestellt werden, dass die Anzahl der Klassen, die verschiedenen Typen, sowie die Samplelänge keinen signifikanten Einfluss auf die Funktionalität haben. Verbes-

serungen konnten sowohl bei Datensätzen mit wenigen Klassen (z. B. ECG200) als auch mit vielen Klassen (z.B. InlineSkate), mit kurzen Samples (z.B. TwoLeadECG) als auch mit langen (StarLightCurves) erreicht werden. Für die Anzahl der Trainingsdaten, die im Ausgangs-Datensatz zur Verfügung stehen, ist jedoch auffällig, dass bei Datensätzen mit wenigen Trainingsdaten, besonders hohe Verbesserungen in Acc und F1-Score erzielt werden konnten (z.B. Sony oder TwoLeadECG).

Ein Vergleich der verschiedenen Architekturen zeigt, dass angereicherte Datensätze, unabhängig von der Architektur, die zur Erzeugung der synthetischen Samples verwendet wird, in unseren Tests besser abschneiden als die originalen Trainingsdatensätze. Dennoch läßt sich beobachten, dass die Ergebnisse von Architektur 3 besonders gut ausfallen. Mit dieser Architektur wurde das beste Gesamtergebnis für sieben von neun Datensätzen erzielt. Die Ergebnisse, die mit $n = 10$ erzielt werden konnten, stechen positiv hervor. Insgesamt lässt sich also sagen, dass die Verwendung von VAEs zur Erzeugung synthetischer Trainingsdaten eine gute Methode ist, wenn eine SVM als Klassifikator verwendet wird. Die Ergebnisse sind vielversprechend über alle Datensätze hinweg. Mit $n \geq 5$, wurden die Benchmark-Ergebnisse in allen Tests verbessert. Nur bei $n = 2$ trat eine Verschlechterung für den UWave-Datensatz auf.

3.4.3 Diskussion

Generell konnte sowohl für den kNN als auch für die SVM in den meisten Fällen eine Verbesserung mit den erweiterten Datensätzen im Vergleich zu den Benchmark-Ergebnissen mit den Originaldatensätzen erzielt werden. Für beide Klassifikatoren hat sich gezeigt, dass $n = 10$ empfehlenswert ist und die unterschiedlichen Charakteristika der Datensätze keinen nennenswerten Einfluss auf die Funktionalität der Methode nehmen. Es fällt jedoch auf, dass besonders für Datensätze mit wenigen Trainingsdaten besonders gute Ergebnisse, sowohl in Acc als auch in F1-Score erzielt werden konnten.

Für den kNN hat sich gezeigt, dass die besten Ergebnisse für fünf von neun Datensätzen mit Architektur 1 erreicht worden sind. Mit Architektur 1 und $n = 10$ konnten darüberhinaus das Benchmark-Ergebnis in allen Experimenten übertroffen werden. Verschlechterungen traten hingegen nur mit $n = 2$ auf. Die erreichten Ergebnisse für den kNN Klassifikator sind daher insgesamt als positiv zu bewerten.

Die SVM profitiert in noch stärkerem Maße von den angereicherten Datensätzen als der kNN. Mit $n = 10$ konnte für alle Architekturen in jedem Experiment eine Verbesserung in Acc und F1-Score erzielt werden. Architektur 3 liefert für die SVM die besten Ergebnisse. Mit dieser konnte das beste Ergebnis in Acc und F1-Score für sieben von neun Datensätzen erreicht werden. Insgesamt zeigen die Ergebnisse deutlich, dass die SVM von den, mit einem VAE erzeugten, syntethisch angerei-

cherten Datensätzen stark profitiert. Hervorzuheben ist hier auch die erhebliche Steigerung der F1-Scores.

Auf Grund der ausführlichen Evaluation ist festzuhalten, dass VAEs ein starkes Werkzeug zur Anreicherung von Zeitreihen-Datensätzen sind. Beide getesteten Klassifikatoren profitieren im Trainingsprozess von den synthetisch generierten Trainingsdaten und können ihre Generalisierungsfähigkeit in den meisten Fällen steigern. Dies spiegelte sich in gesteigerter Acc und verbessertem F1-Score wieder. Es ist davon auszugehen, dass VAEs in Zukunft vermehrt als DA-Methode für Zeitreihen genutzt werden. Weitere Experimente, wie beispielsweise ein höherdimensionaler latentspace oder größere Encoder- und Decoder-Netzwerke, sind interessant sowie notwendig um detaillierte Erkenntnisse zu sammeln. Hier bietet sich viel Raum für zukünftige Arbeiten.

3.5 Erweiterung von Zeitreihen-Spektrogramm-Datensätzen mit Hilfe von Random Noise Boxes

In diesem Kapitel wird eine neue Technik zur Datenerweiterung von Spektrogrammen vorgestellt, welche *Random Noise Boxes* (RNB) genannt wird [94]. Im nachfolgenden werden in Kapitel 3.5.1 zuerst dieser Arbeit vorangegangene Veröffentlichungen vorgestellt, ehe die in dieser Arbeit entwickelte Methode in Kapitel 3.5.2 im Detail erklärt wird. Es folgt eine ausführliche Auswertung in Kapitel 3.5.3 bevor in Kapitel 3.5.4 die Ergebnisse abschließend diskutiert werden.

3.5.1 Vorangegangene Arbeiten

Spektrogramme zeigen den zeitlichen Verlauf des Frequenzspektrums eines Signals in Form einer bildlichen Darstellung. Sie werden in der Regel als Diagramm dargestellt, mit der Zeit auf der x-Achse, den Frequenzen auf der y-Achse und oft einer zusätzlichen Farbskala, die die Intensität repräsentiert.

In früheren Arbeiten wurden Spektrogramme verwendet, um Audiodaten zu transformieren und anschließend mit den kodierten Daten CNNs für Klassifikationsaufgaben zu trainieren [95], [96]. Der Ansatz, Spektrogramme mit CNNs zu klassifizieren, kann bereits weitere vielversprechende Ergebnisse bei Zeitreihendaten vorweisen, wie beispielsweise mit ECG-Daten in [97] und [98] gezeigt werden konnte.

Da CNNs für ein robustes Training jedoch große Datenmengen benötigen und Zeitreihen-Datensätze häufig lediglich kleine Datenmengen zur Verfügung stellen, kann es bei vielen Datensätzen zu Problemen führen, Spektrogramme und CNNs

zu nutzen. In solchen Fällen kann DA helfen, größere Datensätze zu erzeugen und so bessere Trainingsbedingungen für das verwendete CNN herzustellen. Bisher gibt es erst wenige Methoden zur Anreicherung von Spektrogramm-Datensätzen. Einige der bekannten Methoden zur Anreicherung von Spektrogrammen sind Frequenzmaskierung, Image-Warping, Timeshifting Translation und Skalierung [99], [100]. Park et. al konnten mit Ihrer Technik *Specaugment* [101] zeigen, wie verschiedene Blöcke im Spektrogramm maskiert werden können, was zu einem robusteren Trainingsprozess des Netzwerks und somit auch zu besseren Ergebnissen führt.

3.5.2 Methodik

RNB ist eine einfache Methode zur Anreicherung von Spektrogramm-Datensätzen. Bei einem Trainingsdatensatz von Zeitreihen $D = \{T_1, \ldots, T_M\}$ wird zunächst jede Zeitreihe T_i, wobei $1 \leq i \leq M$, normalisiert. Anschließend wird T_i in ein entsprechendes Spektrogramm S_i transformiert. Dann wird jedes S_i multipliziert, so dass eine Anzahl n von identischen Spektrogrammen entsteht. Die Labels bleiben unverändert. Dies führt zu dem vergrößerten Datensatz $S_{aug} = \{S_{i,1}, S_{i,2}, ..., S_{i,n}\}$. Schließlich wird ein zufälliger Teil jedes Spektrogramms in S_{aug} durch eine zufällig platzierte Rauschbox ersetzt. Die Zufallsrauschbox, die zufällige RGB-Pixelwerte enthält, ist eine quadratische Box mit einer vordefinierten Größe und namensgebend für die Methode *Random Noise Boxes*. Für die Experimente in dieser Arbeit sind Boxgrößen von 5 %, 10 %, und 20 % der Größe des zu augmentierenden Spektrogramms gewählt worden. Die Rauschbox wird immer zufällig innerhalb des Spektrogramms platziert und kann nie außerhalb dessen liegen.

Die Arbeitshypothese dieses Ansatzes ist es, dass das CNN durch eine Entfernung des Informationsgehalts von zufälligen Bereichen lernt, sich nicht nur auf bestimmte Bereiche der Eingabe zu konzentrieren, sondern Anhaltspunkte aus der gesamten Eingabe für den Trainingsprozess zu berücksichtigen. Dadurch wird eine bessere Generalisierungsfähigkeit der trainierten Modelle erhofft. Durch die Vervielfältigung eines jeden Spektrogramms und die zufällige Verteilung der Rauschboxen soll darüber hinaus sichergestellt werden, dass dem Algorithmus im Verlauf des Trainingsprozesses trotzdem alle verfügbaren Informationen zugeführt werden.

Um die Anwendung der Methode zu demonstrieren, wurde der StarLightCurves-Datensatz als Beispiel ausgewählt. Der Datensatz umfasst insgesamt 9236 Zeitreihen, die jeweils eine Länge von 1024 Datenpunkten haben. Jede Zeitreihe ist einer von drei Klassen zugeordnet, wobei jede Klasse eine unterschiedliche Lichtintensität repräsentiert. Von den insgesamt 9236 Zeitreihen sind 1000 Trainingsdaten und 8236 Testdaten.

Im ersten Schritt werden alle Zeitreihen auf Werte zwischen 0 und 1 normiert. Als nächstes werden aus den 1000 Trainingszeitreihen Spektrogramme erzeugt, die den Basisdatensatz bilden. Jedes Spektrogramm des Basisdatensatzes wird

anschließend in Abhänigkeit der Multiplikatornummer n vervielfacht. So werden anschließend, wie bei der WNW-Methode, drei neue Datensätze mit $n = 2$, $n = 5$ und $n = 10$ identischen Spektrogrammen, eines jeden Spektrogrammes des Basisdatensatzes, erzeugt. Nachfolgend werden aus jedem der nun bestehenden drei Datensätze wiederum drei Datensätze erzeugt, in denen jedes Spektrogramm entweder mit einer 5 %, 10 % oder 20 % Box überlagert wird. Es entstehen so also neun neue Datensätze. Es ergeben sich somit durch die Anwendung von RNB drei neue Trainingsdatensätze von 2000, 5000 und 10000 Trainingssamples für jede Boxgröße, die aus den ursprünglichen 1000 Trainingsdaten erzeugt worden sind. Der Testdatensatz wurde nicht geändert. Abbildung 3.3 zeigt die Ergebnisse der Anwendung von RNB auf den StarlightCurves-Datensatz mit verschiedenen RNB-Größen. Zur Vorbereitung der Auswertung wurden alle in Kapitel 3.2 vorgestellten Datensätze nach dem beschriebenen Schema vorbereitet.

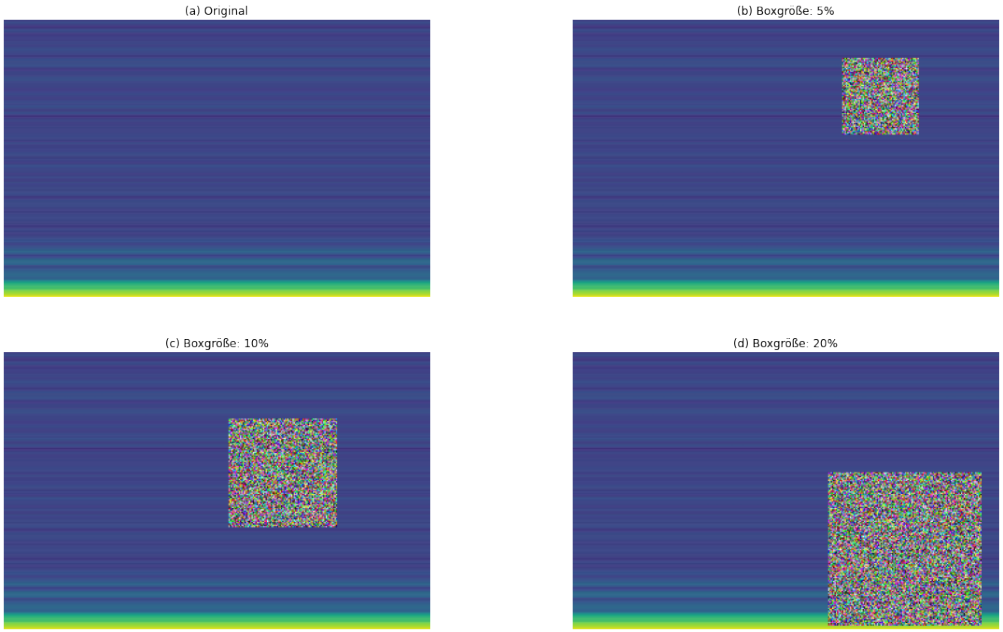

Abbildung 3.3: Beispielhafte Anwendung von RNB auf ein Spektrogramm, erzeugt von einer Zeitreihe des StarLightCurves-Datensatzes: das originale Spektrogramm (a), das Spektrogramm versehen mit einer Rauschbox von 5 % (b), 10 % (c) und 20 % (d).

3.5.3 Evaluation

3.5.3.1 Bedingungen

Auf Grund der in Kapitel 2.3.3 beschriebenen Eigenschaften, der häufigen Verwendung in Forschung und Industrie sowie der guten Ergebnisse vorangegangener Arbeiten, erfolgt die Bewertung der vorgestellten Methode mit Hilfe von CNNs. Aus Voruntersuchungen ist hervorgegangen, dass CNNs mit unterschiedlichen Architekturen benötigt werden, um angemessene Resultate für die Benchmark-Experimente in Acc und F1-Score zu erzielen. Es wurden daher drei verschiedene Architekturen unterschiedlicher Größe ausgewählt. Diese werden im Folgenden als Architektur 1, 2 und 3 bezeichnet. Der Aufbau der Architekturen ist in Tabelle 3.6 dargestellt. Zur Veranschaulichung ist Architektur 2 in Abbildung 3.4 visualisiert.

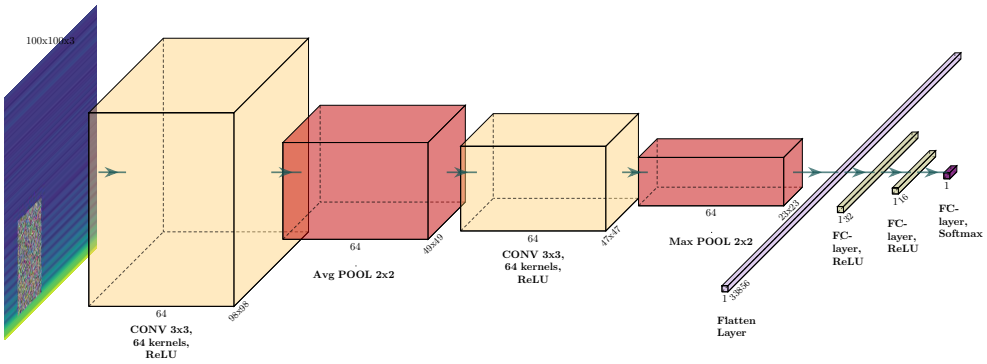

Abbildung 3.4: Schematische Darstellung des Aufbaus von CNN Architektur 2.

Architektur 1 verfügt über zwei conv Schichten mit 16 Kernel und zwei FC Schichten mit 16 Neuronen. Die conv Schichten von Architektur 2 haben 64 Kernel und die FC Schichten 32 Neuronen. Für Architektur 3 wurden conv. Schichten mit 128 Kernel sowie FC Schichten mit 32 Neuronen gewählt. Für alle Architekturen gilt, dass kein padding genutzt wurde. Als Aktivierungsfunktion wird immer ReLU genutzt, außer in der jeweils letzten Schicht, in der Softmax angewandt worden ist. Alle Gewichte werden mit He-Initialisierung [102] und Nullen als Bias-Gewichte initialisiert. Als Optimierer wird der Adam-Optimizer eingesetzt [57]. Wenn ein Datensatz über 2 Klassen verfügt, so wird als Verlustfunktion binäre Kreuzentropie genutzt, ansonsten wird kategoriale Kreuzentropie verwendet. Die Dauer des Trainingsprozesses wird auf 100 Epochen festgesetzt.

Für die Datensätze ECG5000, StarLightCurves, InlineSkate und Sony wird Architektur 1 genutzt, für UWave, TwoLeadECG und MoteStrain Architektur 2 und für GunPoint und ECG200 Architektur 3.

Tabelle 3.6: Übersicht über den Aufbau der unterschiedlichen CNN Architekuren.

Architektur	**1. Schicht Art: Conv**	**2. Schicht Art: Conv**	**3. Schicht Art: FC**	**4. Schicht Art: FC**
Kleines CNN Initialisierung: He	Filter: 3x3 #kernels: 16 Padding: nein Nachfolgend: Pooling: 2x2 Art: avg Pool	Filter: 3x3 #kernels: 16 Padding: nein Nachfolgend: Pooling: 2x2 Art: max Pool	#Neuronen: 16 Aktivierung: ReLU	
Medium CNN Initialisierung: He	Filter: 3x3 #kernels: 64 Padding: nein Nachfolgend: Pooling: 2x2 Art: avg Pool	Filter: 3x3 #kernels: 64 Padding: nein Nachfolgend: Pooling: 2x2 Art: max Pool	#Neuronen: 32 Aktivierung: ReLU	#Neuronen: 16 Aktivierung: ReLU
Großes CNN Initialisierung: He	Filter: 3x3 #kernels: 128 Padding: nein Nachfolgend: Pooling: 2x2 Art: avg Pool	Filter: 3x3 #kernels: 128 Padding: nein Nachfolgend: Pooling: 2x2 Art: max Pool	#Neuronen: 32 Aktivierung: ReLU	#Neuronen: 16 Aktivierung: ReLU

Um ein Benchmarkergebnis auf die nicht augmentierten Originaldaten zu erhalten, werden zunächst die unveränderten Orignaldatensätze für den Trainings- und Testprozess aller CNNs genutzt. Anschließend wird jeder Datensatz mit RNB gemäß dem in Kapitel 3.5.2 beschriebenen Verfahren angereichert und für die unterschiedlichen Evaluierungsaufgaben vorbereitet. Danach werden die CNNs mit den augmentierten Daten trainiert und die Ergebnisse mit den zuvor erstellten Benchmarks verglichen. Insgesamt gibt es zehn verschiedene Ergebnisse für jeden Datensatz: Das Benchmark-Ergebnis und neun zusätzliche Ergebnisse, die jeweils

Tabelle 3.7: Evaluationsergebnisse von RNB. Zu sehen sind die Benchmarkergebnisse sowie der Ergebnisse für $n = 2$, $n = 5$ und $n = 10$ mit einer Boxgröße von 5 %, 10 % und 20 %.

Dataset	**metric**	**original data**	**5 %**			**10 %**			**20 %**		
			$n = 2$	$n = 5$	$n = 10$	$n = 2$	$n = 5$	$n = 10$	$n = 2$	$n = 5$	$n = 10$
ECG200	Acc	0.5740	0.5920	0.5840	0.6680	0.6620	**0.6940**	0.6620	0.6540	0.6700	0.6140
	F1	0.4784	0.5460	0.5382	0.5558	0.5566	**0.6102**	0.6002	0.5656	0.5568	0.4922
ECG5000	Acc	0.2528	0.3484	0.3826	**0.5078**	0.3879	0.4288	0.3827	0.5057	0.2029	0.3367
	F1	0.1734	0.2720	0.2462	**0.3700**	0.2870	0.2954	0.2464	0.3696	0.1242	0.1976
TwoLeadECG	Acc	0.5306	0.5661	0.5649	**0.6727**	0.5679	0.6162	0.5593	0.5459	0.5487	0.5535
	F1	0.4308	0.4908	0.4618	**0.6308**	0.4474	0.5852	0.5092	0.4364	0.4754	0.4774
StarlightCurves	Acc	0.4939	0.5541	0.5019	**0.5603**	0.4102	0.5184	0.5509	0.4702	0.5178	0.5177
	F1	0.3900	0.4482	0.3752	**0.5524**	0.2702	0.4032	0.4522	0.3282	0.3620	0.3260
SonyAIBORobotSurface	Acc	0.5134	0.5464	0.4789	0.5181	0.5508	**0.5707**	0.5288	0.5197	0.5576	0.5651
	F1	0.3982	0.4378	0.3394	0.4116	0.4630	**0.4980**	0.4242	0.3870	0.2894	0.4266
MoteStrain	Acc	0.4799	0.4891	**0.5102**	0.4948	0.5019	0.4946	0.4789	0.4955	0.5086	0.4855
	F1	0.3925	0.4140	0.4832	0.4834	0.4838	0.4618	**0.4950**	0.4288	0.4334	0.3932
GunPoint	Acc	0.4973	0.4986	0.4933	0.5012	**0.5093**	0.4994	0.4973	0.4986	0.4983	0.4932
	F1	0.3434	0.3974	0.3260	0.3504	0.3560	0.3501	**0.3630**	0.3462	0.3459	0.3620
InlineSkate	Acc	0.1598	0.1589	0.1545	0.1618	0.1622	**0.1661**	0.1640	0.1599	0.1607	0.1556
	F1	**0.0518**	0.0480	0.0410	0.0450	0.0450	0.0452	0.0474	0.0462	0.0440	0.0444
UWaveGestureLibraryX	Acc	**0.2471**	0.1856	0.1959	0.2103	0.2120	0.1979	0.2023	0.1214	0.1631	0.2170
	F1	**0.1952**	0.0982	0.1200	0.1442	0.1290	0.1270	0.1312	0.0264	0.0854	0.1500

einer anderen Boxgröße von RNB und einer unterschiedlichen Multiplikationszahl n des jeweiligen Datensatzes entsprechen. Tabelle 3.7 zeigt die Bewertungsergebnisse unter Verwendung von Acc und F1-Score als Metriken.

3.5.3.2 Ergebnisse

Es fällt auf, dass sowohl mit den Originaldatensätzen als auch mit den augmentierten Datensätzen lediglich kleine Werte für Acc. und F1-Score erreicht wurden. Dies ist darin begründet, dass die genutzten Datensätze nicht optimal für die Transformation in Spektrogramme geeignet sind. Für die bearbeitete Problemstellung ist dies jedoch nicht von Bedeutung, da der Fokus auf der Prüfung der Funktionalität der beschriebenen DA Methode liegt.

Mit der richtigen Parametereinstellung von RNB konnte die Acc für acht von neun Datensätzen und der F1-Score für sieben von neun Datensätzen verbessert werden. Die Ergebnisse für alle Datensätze und die zugehörigen Parametereinstellung von RNB sind in Tabelle 3.7 zu sehen.

Für ECG200 und TwoLeadECG konnte eine Verbesserung in Acc und F1-Score mit allen getesteten Parameterstellungen erreicht werden. Für den MoteStrain-Datensatz konnte der F1-Score in allen Experimenten verbessert werden. Dies lässt als erste Schlussfolgerung zu, dass RNB eine geeignete Methode zur Erweiterung von Spektrogramm-Datensätzen darstellen kann. Es ist jedoch zu beachten, dass

nicht für alle Datensätze Verbesserungen erreicht worden sind. Für UWave konnte in keinem Experiment eine Verbesserung, weder in Acc noch in F1-Score, erreicht werden.

Die besten Parameterkombinationen waren eine Boxgröße von 10 % mit $n = 5$ und eine Boxgröße von 5 % mit $n = 10$. Mit diesen Parametern wurden alle Benchmark-Ergebnisse in Acc außer für dem UWave-Datensatz und alle Benchmark-Ergebnisse in F1 mit Ausnahme der UWave- und InlineSkate Datensätze übertroffen. Für sechs Datensätze konnte mit einer dieser Parametereinstellungen das beste Ergebnis aller durchgeführten Experimente erzielt werden. Geeignete Parametereinstellungen waren auch eine Boxgröße von 10 % mit $n = 10$ und 5 % bei $n = 2$. Mit diesen Einstellungen konnten die Benchmark-Ergebnisse für sieben von neun Datensätzen übertroffen werden. Die Ergebnisse für 20 % Boxgrößen fallen schlechter aus als die Ergebnisse für 5 % und 10 %, insbesondere bei der Betrachtung des F1-Scores. Dies wird darauf zurückgeführt, dass bei 20 % Rauschen zu viele Informationen durch die Box versperrt werden und dem CNN somit für den Trainingsprozess nicht zur Verfügung stehen.

Die Betrachtung der Ergebnisse im Hinblick auf die unterschiedlichen Eigenschaften der Datensätze lässt weitere Rückschlüsse über die Anwendbarkeit der vorgestellten Methode zu. Es kann festgestellt werden, dass die Samplelänge der Zeitreihen (bzw. die daraus resultierende Pixelanzahl der Spektrogramme) keinen Einfluss auf die Ergebnisse hat. Auch die Anzahl der verfügbaren Trainingsdaten scheint eine untergeordnete Rolle zu spielen da sowohl bei Datensätzen mit vielen Trainingsdaten Verbesserungen erzielt werden konnten (z. B. StarLightCurves), als auch bei Datensätzen mit wenigen Trainingsdaten (z. B. MoteStrain). Es ist jedoch auffällig, dass für Datensätze mit besonders wenigen Trainingsdaten häufig eine deutliche Verbesserung erreicht werden kann. Beispielsweise konnte die Acc des TwoLeadECG-Datensatz, der nur 23 Trainingsdaten enthält, um 14.21 % gesteigert werden. Andererseits scheint die Anzahl der Klassen im Datensatz eine wichtigere Rolle beim Erreichen der Verbesserung zu spielen. Zum Beispiel sind die Benchmark-Ergebnisse für den UWave-Datensatz (8 Klassen) mit keiner Parameterkombination erreicht worden. Auch für den Inlineskate-Datensatz (7 Klassen) gab es nur unbedeutende Verbesserungen in Acc und Verschlechterungen im F1-Score. Dies lässt darauf schließen, dass RNB vor allem für Datensätze mit einer Klassengröße < 7 interessant ist.

3.5.4 Diskussion

RNB ist eine einfach zu implementierende Methode zur Erweiterung von Spektrogrammdatensätzen. Die Evaluation hat gezeigt, dass RNB helfen kann, die Leistung, sowohl in Acc als auch in F1-Score, von CNNs die zur Klassifikation genutzt werden, zu verbessern.

Die Funktionalität wurde hierzu an neun unterschiedlichen Datensätzen mit drei unterschiedlichen CNN Architekturen überprüft.

Die Evaluation hat ergeben, dass in den meisten Fällen eine Verbesserung erreicht werden konnte, wenn die Trainingsdaten der getesteten Datensätze vor dem Trainingsprozess mit RNB angereicht worden sind. Im Besonderen mit den Parametern 5 % Boxgröße und $n = 10$, sowie 10 % und $n = 5$, wurden sehr gute Ergebnisse erzielt. Verbesserungen in Acc wurden für acht von neun Datensätzen erreicht und der F1-Score für sieben von neun Datensätzen. Die Verwendung von 20 % Boxgröße (oder größer) wird nicht empfohlen. Es wird angenommen, dass dem CNN in diesem Fall zu viele Informationen vorenthalten werden und die gewünschte Generalisierung verhindert wird. Diese Erkenntnis kann durch den häufig gesunkenen F1-Score bei Verwendung einer Boxgröße von 20 % bekräftigt werden.

Bei Betrachtung der Funktionalität von RNB in Bezug auf die unterschiedlichen Eigenschaften der verschiedenen Datensätze, hat sich gezeigt, dass RNB für Datensätze mit vielen Klassen nicht geeignet zu sein scheint. Für den UWave-Datensatz (8 Klassen) sowie für den Inlineskate Datensatz (7 Klassen), jene beiden Datensätze, die unter allen getesteten die meisten Klassen aufweisen, ist die Leistungsentwicklung als schlecht zu bewerten. Die Samplelänge und die Anzahl der verfügbaren Trainingsdaten scheinen hingegen wenig Einfluss auf die Leistungsentwicklung zu haben. Verbesserungen konnten sowohl bei Datensätzen mit wenigen (Sony, MoteStrain), als auch mit vielen Trainingsdaten (StarLightCurves) erzielt werden.

Es kann gefolgert werden, dass RNB eine geeignete Methode für die Vergrößerung von Spektrogramm-Datensätzen ist, jedoch sich auch negativ auf die Leistung auswirken kann. Es ist daher ratsam, die Entwicklung der gewählten Metriken zur Leistungsüberprüfung sorgfältig zu beobachten und verschiedene Parameterkonstellationen zu untersuchen. Von einer Anwendung für Datensätze mit vielen Klassen ist auf Grund der vorliegenden Ergebnisse abzuraten.

3.6 Zusammenfassung & Diskussion

In diesem Kapitel wurden drei neue Methoden zur Erweiterung von Zeitreihen-Datensätzen vorgestellt. Alle Methoden wurden mit Hilfe derselben neun Datensätze, gezeigt in Kapitel 3.2, auf Funktionsfähigkeit und Anwendbarkeit untersucht. Die verwendeten Datensätze wurden dem UCR Timeseries classification [103] archive entnommen und werden in der Forschung häufig für ähnliche Aufgabenstellungen oder Experimente herangezogen. WNW sowie die Erzeugung synthetischer Daten mit Hilfe eines VAE wurden auf die rohen Zeitreihendaten angewandt und anschließend mit Hilfe der bekannten Klassifikatoren kNN und SVM getestet.

Für RNB wurden die Daten zuerst in Spektrogramme umgewandelt und nachfolgend mit unterschiedlichen CNN Architekturen klassifiziert. Untersucht wurden jeweils die Entwicklung von Acc und F1-Score.

Es hat sich gezeigt, dass mit allen Methoden Verbesserungen in Acc und F1-Score erreicht werden konnten. Um die gewünschte Verbesserung zu erreichen, ist jedoch häufig die Wahl der richtigen Parameterkombinationen der unterschiedlichen Methoden entscheidend.

Mit Hilfe von WNW konnten Acc und F1-Score für alle Datensätze gesteigert werden, wenn der kNN als Klassifikator genutzt wurde. Für die SVM konnten Acc und F1-Score für acht von neun Datensätzen verbessert werden. Entscheiden war jeweils die Wahl der richtigen Multiplikatorzahl n und der Fenstergröße. Klassifikator übergreifend kann festgehalten werden, dass $n = 5$ oder $n = 10$ in Kombination mit einer Fenstergröße von 5 % oder 10 % die besten Ergebnisse lieferten. Die Charakteristika der evaluierten Datensätze haben lediglich eine untergeordnete Rolle eingenommen. Hervorzuheben ist jedoch, dass besonders für Datensätze mit wenigen Trainingsdaten hohe Verbesserungen erreicht wurden und Datensätze mit vielen Klassen zu Problemen führen können.

Die insgesamt besten Ergebnisse wurden mit den Datensätzen erzielt, die synthetische Trainingsdaten, erzeugt durch einen VAE, enthalten. Besonders für die SVM wurden vielversprechende Resultate erzielt. Hier konnte das Benchmark-Ergebnis von vier Datensätzen parameterunabhängig übertroffen werden. Als Empfehlung kann dennoch $n = 5$ oder $n = 10$ gegeben werden, da hiermit bessere Ergebnisse erzielt werden konnten, als mit $n = 2$ und sich so ein klarer Trend über alle Experimente abgezeichnet hat. Die Architektur des VAE sollte den zu erweiternden Daten angepasst werden.

Mit RNB konnte eine neuartige Methode zur Erweiterung von Spektrogramm-Datensätzen vorgestellt werden, mit der ebenfalls vielversprechende Ergebnisse erreicht wurden. Für die meisten Datensätze konnte eine Verbesserung in Acc und F1-Score erzielt werden. Voraussetzung hierfür ist die Wahl der richtigen Hyperparameter. Die besten Ergebnisse wurden mit einer Boxgröße von 5 % und $n = 10$ oder einer Boxgröße von 10 % und $n = 5$ erzielt. Zu beachten ist, dass die Ergebnisse von RNB nicht mit denen der anderen Methoden verglichen werden sollten, da die Zeitreihen in Spektrogramme transformiert wurden und daher die Klassifikation mit einem CNN anstatt mit SVM oder kNN durchgeführt wurde.

Insgesamt lässt sich festhalten, dass drei vielversprechende Methoden gefunden werden konnten, mit denen Zeitreihen-Datensätze erweitert werden können. Diese neuen Möglichkeiten geben Analysten von Zeitreihendaten neue Werkzeuge an die Hand und verkleinern die Lücke, die zwischen der Anzahl an zur Verfügung stehenden Methoden zur Erweiterung von Bilddaten gegenüber Zeitreihendaten vorliegen. Besonders VAE scheinen eine zukunftsträchtige Möglichkeit der Erweiterung

von Zeitreihendaten darzustellen. Es ist jedoch festzuhalten, dass alle Methodiken weiterer Untersuchungen und Tests benötigen, um diese zu standardisieren.

Kapitel 4

Innovativer Ansatz zur Analyse von Zeitreihen mit Hilfe von Rekurrenzplots

Die Interpretation einer Zeitreihe aus ihrer rohen Zeitbereichsdarstellung kann eine schwierige Aufgabe sein. In vielen Anwendungen, bei denen Einblicke in das Verhalten eines dynamischen Systems erforderlich sind, sind bildhafte Darstellungen wie Spektrogramme oder Skalogramme ein hilfreiches Werkzeug, um detailliertere Aussagen über das System treffen zu können. Rekurrenzplots bieten eine weitere Option zur Analyse einer Zeitreihe und sind eine Möglichkeit zur Darstellung einer Zeitreihe durch eine quadratische Matrix, deren Elemente $\neq 0$ den Zeitpunkten entsprechen, zu denen die Zustände eines dynamischen Systems wiederkehren. Sie bieten sich daher vor allem für periodische Systeme als Analysewerkzeug an. Ein theoretischer Hintergrund von Rekurrenzplots ist in Kapitel 2.5 beschrieben. Rekurrenzplots und die Informationen, die sie erfassen, können auch für Klassifikationsaufgaben von Zeitreihen verwendet werden. Da Rekurrenzplots für Klassifikationsaufgaben von neuronalen Netzen oder anderen Algorithmen bisher eine untergeordnete Rolle spielen, ist eine nähere Untersuchung der Funktionalität interessant.

In Kapitel 4.1 werden Erkenntnisse aus der Literaturrecherche zur Zeitreihen-Klassifikation mittels Rekurrenzplots vorgestellt. In Kapitel 4.2 wird die Methodik sowie das Parametersetting für die in Kapitel 4.3 durchgeführte Klassifikation näher erläutert. In Kapitel 4.4 erfolgt die abschließende Diskussion der Ergebnisse.

4.1 Vorangegangene Arbeiten

In [104] werden sechs verschiedene Zeitreihen-zu-Bild-Kodierungen, Gramian Angular Field, Markov Transition Field, Rekurrenzplot, Grey Scale Encoding, Spektrogramm und Skalogramm auf einen Datensatz mit Zeitreihen von Vibrationsdaten angewandt. Der verwendete Datensatz besteht aus Messungen von realen Hubschrauber-Flugtests, wobei die Aufgabe darin besteht, Anomalien in den Signalen zu erkennen. Der Test wird zuerst mit den rohen Zeitreihen und anschließend mit den kodierten Daten durchgeführt. Im Pre-processing wurden die Samples der Vibrationssignale zunächst in eine der 2D-Darstellungen konvertiert und in ein Convolutional Autoencoder Netzwerk eingespeist, das Rekonstruktionen der Eingangsdaten ausgibt. Wenn für einen Testsatz einer der Residuen der rekonstruierten Samples größer ist als ein Schwellenwert ϵ , wird es als anormal gekennzeichnet. Der Schritt zur Erkennung von Anomalien ist derselbe für die rohen Zeitreihen, jedoch ohne den Kodierungsschritt und mit 1D-Faltungen. Abbildung 4.1 zeigt die Ergebnisse der Zeitreihen-zu-Bild-Kodierungsschritte für die sechs genannten Methoden.

Die Bewertung der Ergebnisse erfolgt an Hand von F1-Score, Acc und Falsch-Positiv-Raten. Mit allen sechs Methoden wurden bessere Ergebnisse als mit den Rohdaten erzielt. Skalogramme lieferten für den spezifischen Datensatz und bei dem genutzten Bewertungsmetriken die beste Leistung. Diese Ergebnisse zeigen, dass wichtige Eigenschaften der Zeitreihen bei der Kodierung in Bilddarstellungen erhalten bleiben, nachdem sie in Bildrepräsentationen kodiert wurden.

Wichtig zu beachten ist, dass durch das Testen der Ansätze an nur einem bestimmten Datensatz nicht sichergestellt ist, dass die Unterschiede in der Leistung nicht darauf zurückzuführen sind, dass einige Zeitreihen-Kodierungen nur für den spezifischen Datensatz oder für einige Arten von Anomalien gut geeignet sind. Neben der Leistungsverbesserung ist zu erwähnen, dass Bilddarstellungen in Bezug auf die Berechnungseffizienz teurer sind als rohe Zeitreihen, was für Anwendungen relevant sein kann, bei denen Speicher oder Rechenleistung begrenzt sind.

In [105] werden Zeitreihen-Samples aus dem UCR time series classification archiv in Rekurrenzplots umgewandelt und in einem CNN verwendet, um die Klassifizierungsleistung mit klassischen Zeitreihen-Klassifizierungsmethoden zu vergleichen. Zunächst wird aus jeder Probe eine Phasenraumtrajektorie mit einer Zeitverzögerungseinbettung von eins erstellt. Dann wird aus der Trajektorie der Rekurrenzplot erstellt. Der Vorgang ist in Abbildung 4.2 dargestellt.

Die Klassifikationsergebnisse von verschiedenen UCR-Datensätzen werden hinsichtlich ihrer Fehlerraten gegen die Ergebnisse von klassischen Zeitreihen-Klassifikations-Algorithmen verglichen. Mit Blick auf den durchschnittlichen Rang der Fehlerrate, schnitten die Rekurrenzplots am besten ab. Die Autoren schließen aus

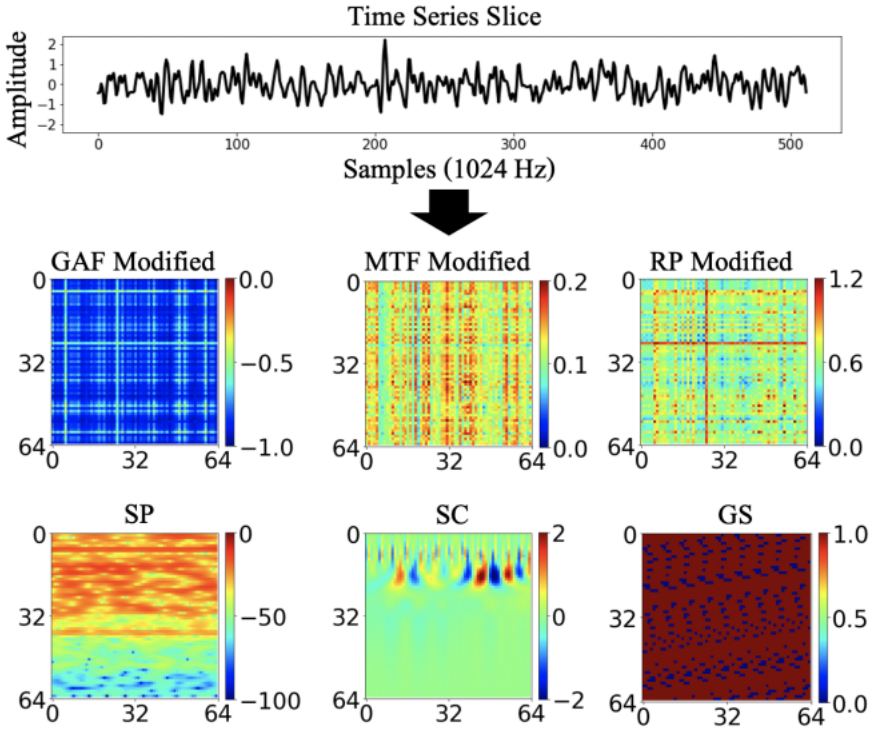

Abbildung 4.1: Beispiele der 2D-Repräsentationen der Zeitreihen von Helikopter-Flugtests. Die verwendeten Methoden sind (von oben links nach unten rechts): Gramian Angular Field, Markov Transition Field, Rekurrenzplot, Spektrogram, Skalogram und Grey Scale Kodierung , entnommen aus [104].

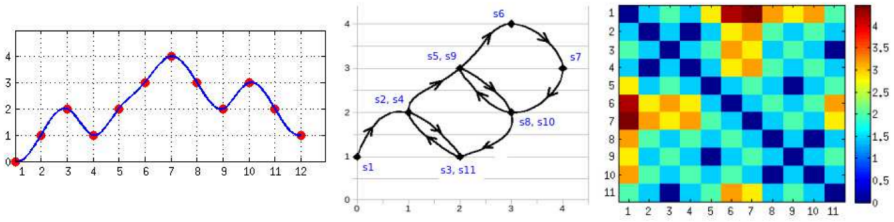

Abbildung 4.2: Prozess der Rekurrenzplot-Erstellung von einer 2D-Phasenraum-trajektorie, entnommen aus [105].

ihren Experimenten, dass Rekurrenzplots gut geeignet sind, um in der Zeitreihen-Klassifikation eingesetzt zu werden.

4.1.1 Weitere interessante Ergebnisse anderer Arbeiten

In den hier rezensierten Veröffentlichungen wurden die Experimente nur für Rekurrenzplots mit festen Parametereinstellungen durchgeführt. Es wäre interessant zu sehen, ob die Klassifikationsmodelle für verschiedene Rekurrenzplot-spezifische Parameter wie Schwellenwert, Einbettungsdimension und Zeitverzögerung Einfluss auf die Funktionalität und Anwendbarkeit von Rekurrenzplots haben. Eine Arbeit, die sich mit den Auswirkungen der Einbettungsdimension auf Rekurrenzplots von experimentellen Daten beschäftigt, kam zu dem Ergebnis, dass Rekurrenzplots qualitativ und quantitativ unabhängig von der Einbettungsdimension sind [106].

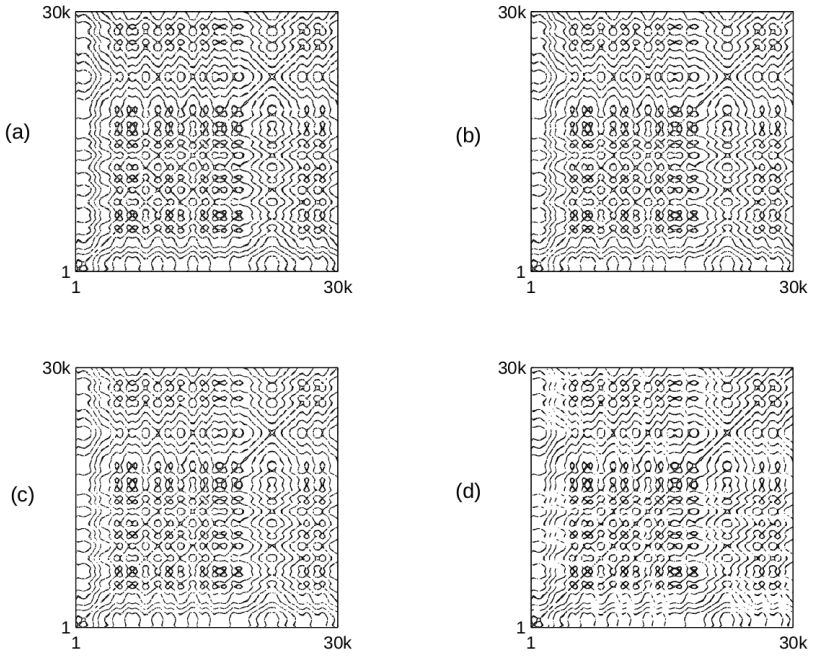

Abbildung 4.3: Rekurrenzplots von Winkelsensordaten. Die Diagramme (a)-(d) stellen die Ergebnisse für unterschiedliche Einbettungsdimensionen (a = 1, b = 2, c = 3 und d = 4) bei sonst gleichen Parametern dar, entnommen aus [106].

Abbildung 4.3 zeigt wie ähnlich Rekurrenzplots für verschiedene Einbettungsdimensionen aussehen. Die Plots wurden aus Winkelsensordaten eines angetriebenen Pendels erzeugt und variieren nur in Form einer Aufhellung mit zunehmender Einbettungsdimension. Auf Grund dieser Erkenntnis wird für alle weiteren Experimente mit Rekurrenzplots dieser Arbeit die Einbettungsdimension nicht weiter beachtet und standardmäßig mit 1 gewählt.

Weitere interessante Erkenntnisse liefert eine Arbeit, die sich mit den Effekten des Poolings von Schichten in CNNs beschäftigt [107]. Während sowohl Max- als

auch Average-Pooling-Schichten zum Zweck der Dimensionalitätsreduktion implementiert werden, können sie zu unerwünschten Ergebnissen wie Informationsverlust führen, wenn sie nicht mit den Eingabe Feature-Maps übereinstimmen. Das Problem ist beispielhaft in Abbildung 4.4 dargestellt.

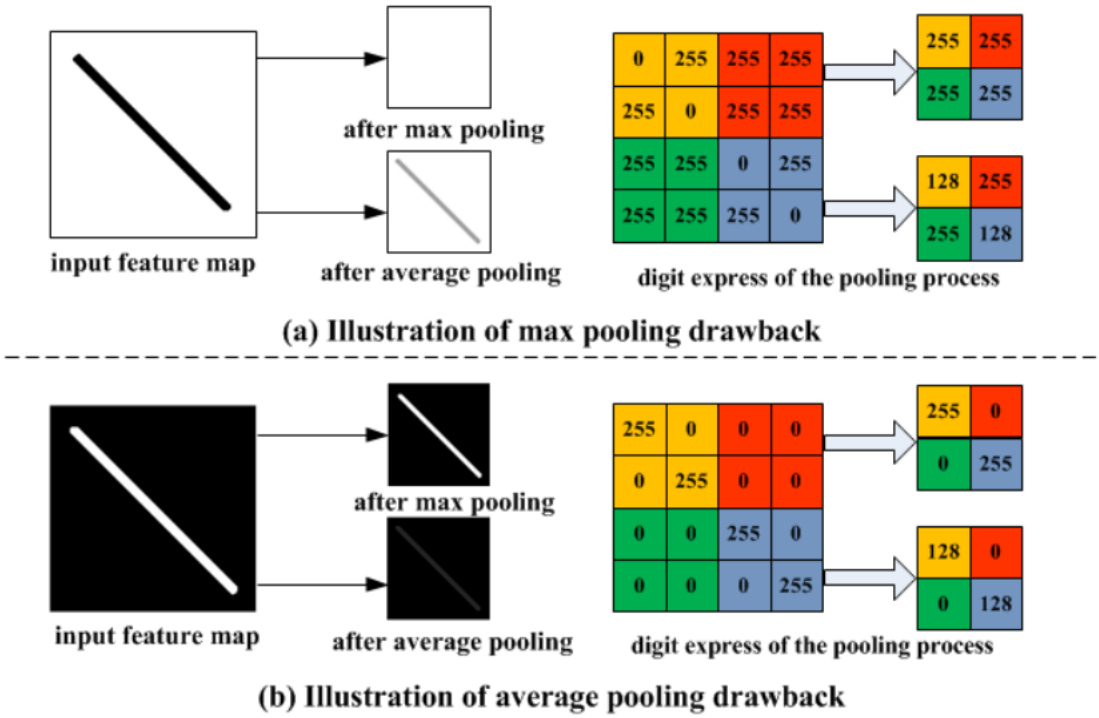

Abbildung 4.4: Visualisierung möglicher Nachteile von max- und average pooling für CNNs, entnommen aus [107].

Wenn die Mehrheit der Elemente in der Pooling-Region von hoher Größe ist, führt max pooling zu Informationsverlusten. Average pooling, das den Mittelwert der Elemente in der Pooling-Region berechnet, erhält die spezifischen Merkmale, da es auch die niedrigeren Werte berücksichtigt (Abbildung 4.4 (a)). Wenn jedoch die Mehrheit der Elemente in einer Pooling-Region von geringer Größe ist (Abbildung 4.4 (b)), bewahrt das Max-Pooling die Eigenschaft des Merkmals, während das Average-Pooling wiederum auch alle niedrigen Werte berücksichtigt, was in dem gezeigten Beispiel zu Informationsverlust führt.

Bislang wurde gezeigt, dass Rekurrenzplots vertikale, horizontale und diagonale Linien als Hauptkomponenten haben. Abhängig von der Struktur der Zeitreihe kann der Rekurrenzplot mehr oder weniger spärlich sein, mit vielen kurzen Linien oder Punkten gefüllt sein, sowie Regionen mit höherer oder niedrigerer Rekurrenz-

punktdichte aufweisen. Die Feature-Maps der Rekurrenzplots können daher unterschiedlicher Natur sein und Ihre Werte in manchen Fällen homogen verteilt und in anderen Fällen häufig 0 oder maximal sein. Die Wahl des Pooling-Schichttyps muss daher sorgfältig geprüft und der Beschaffenheit der Daten angepasst werden.

4.2 Methodik

Nachdem die Erkenntnisse der Literaturrecherche vorgestellt wurden, wird nun die Methodik für den in dieser Arbeit entwickelten Ansatz zur Analyse und Klassifikation von Zeitreihen mit Hilfe von Rekurrenzplots vorgestellt.

In den durchgeführten Experimenten, wurden CNNs verwendet, um die Daten, aus denen in Kapitel 3.2 vorgestellten Datensätzen zu klassifizieren. Eingangsdaten sind die Rekurrenzplots der Trainingsdaten. In Abbildung 4.5 sind beispielhaft Rekurrenzplots der Datensätze ECG200, StarLightCurves und GunPoint mit den zu Grunde liegenden Zeitreihen gezeigt. Das CNN hat die Aufgabe der Merkmalsextraktion um die charakteristischen Muster wie horizontale, vertikale oder diagonale Linien für bestimmte Klassen zu erkennen. Die Modelle wurden mit der Deep-Learning-Bibliothek Keras [108] erstellt.

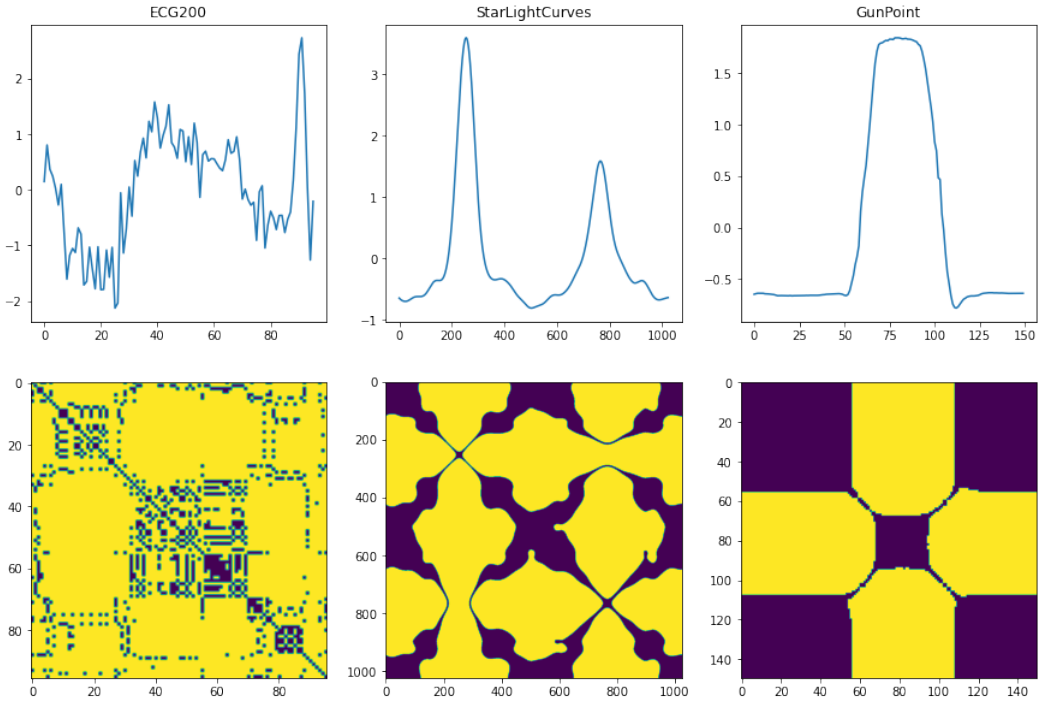

Abbildung 4.5: Beispielhafte Rekurrenzplots zufälliger Zeitreihen der Datensätze ECG200 (links), StarLightCurves (mitte) und GunPoint (rechts).

Im Rahmen dieser Arbeit wurden für jeden Datensatz zwölf Experimente mit unterschiedlichen Parametereinstellungen durchgeführt. Erste Tests mit Visualisierungen der Rekurrenzplots zeigten, dass der Schwellenwertparameter ϵ einen starken Einfluss auf das visuelle Erscheinungsbild der Rekurrenzplots hat. Die unterschiedliche Anzahl von Punkten im Phasenraum, die für die Berechnung des Abstands zu einem bestimmten Punkt berücksichtigt werden, sorgt dafür, dass die Rekurrenzmatrix spärlicher oder mit mehr Einträgen gefüllt ist. Für die Experimente wurden unterschiedlicher Werte von ϵ gewählt. Abbildung 4.6 zeigt die Auswirkungen der ausgewählten Werte auf das Erscheinungsbild der Rekurrenzplots.

Je höher der Schwellenwert gesetzt wird, desto mehr benachbarte Punkte können berücksichtigt werden, was zu einer Matrix mit mehr Einträgen $\neq 0$ führt. Der Rekurrenzplot des ECG200-Beispiels in Abbildung 4.6 zeigt eine zunehmende Anzahl von Einträgen $\neq$ Null, wenn ϵ erhöht wird. Dies könnte den Filtern in den conv. Schichten mehr Merkmale zur Unterscheidung bieten. Während für das ECG200-Beispiel Punkte in den Rekurrenzplots mit größeren Schwellenwerten erscheinen, die nicht in den Rekurrenzplots mit kleineren Schwellenwerten auftauchen, werden die Linien in den StarLightCurves-Plots nur ausgeprägter, aber es erscheinen keine Punkte in neuen Regionen. Für das GunPoint-Beispiel in Abbildung 4.6 gilt qualitativ, dass bei einem erhöhten Schwellenwert sogar einige Details verloren gehen könnten. Während die obere linke Ecke im Diagramm $\epsilon = 0{,}01$ in ein größeres Quadrat in der Mitte unterteilt ist, das von vier kleineren Quadraten umgeben ist, geht diese Information bei den Schwellenwerten 0,1 und 0,2 verloren. Wie die Beispiele zeigen, dass eine Erhöhung von ϵ zu mehr, aber auch zu weniger detaillierten Rekurrenzplots führen kann, wird in Bezug auf Acc und F1-Score angenommen, dass sich die Klassifikationsergebnisse manchmal verbessern und manchmal auch verschlechtern könnten, abhängig von der Struktur der Zeitreihe. Es scheint daher sinnvoll, unterschiedliche ϵ Werte für jeden Datensatz zu testen, um ein verbessertes Ergebnis zu erhalten.

Der zweite Parameter, der variiert wird, ist die Anzahl der Schritte, also der eindeutigen Werte in einer Rekurrenzmatrix. Während Rekurrenzmatrizen bisher binär waren, macht die Hinzunahme zusätzlicher Schrittzahlen Rekurrenzplots zunehmend komplexer. Rekurrenzplots mit einer Schrittzahl von 1 werden mit Einträgen von 0 und 1 dargestellt, Rekurrenzplots mit n Schritten bestehen aus n möglichen eindeutigen Werten. Die Schrittzahlen wurden mit 1, 10, 20 und 50 gewählt. Abbildung 4.7 zeigt Rekurrenzplots für die verschiedenen Schrittzahlen für Zeitreihen aus den Datensätzen ECG200, StarLightCurves und GunPoint bei einem festen Schwellenwert $\epsilon = 0{,}1$.

Mit zunehmender Schrittweite sind die Rekurrenzplots nun farbcodiert, wobei die Anzahl der verschiedenen Farben gleich der Schrittweite ist. Damit können

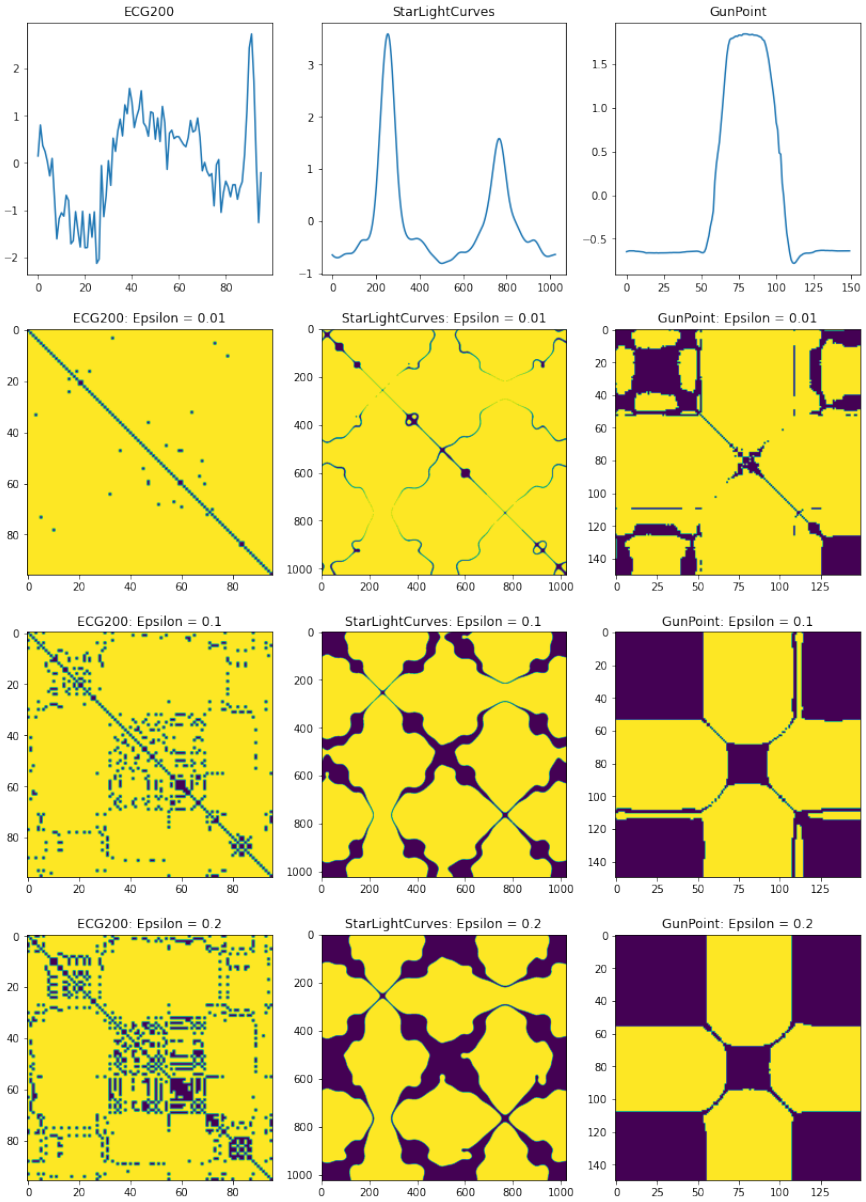

Abbildung 4.6: Rekurrenzplots der Datensätze ECG200 (links), StarLightCurves (mitte) und GunPoint (rechts) für $\epsilon = 0.01$ (oben), $\epsilon = 0.1$ (mitte) und $\epsilon = 0.2$ (unten). Die Grafiken in der obersten Zeile zeigen die zu Grunde liegenden Zeitreihen.

die Filter in den Faltungsschichten nicht nur die Struktur der Rekurrenzpunkte/-linien lernen, sondern haben auch die Schritte zwischen 0 und $n-1$ als zusätzliche Information.

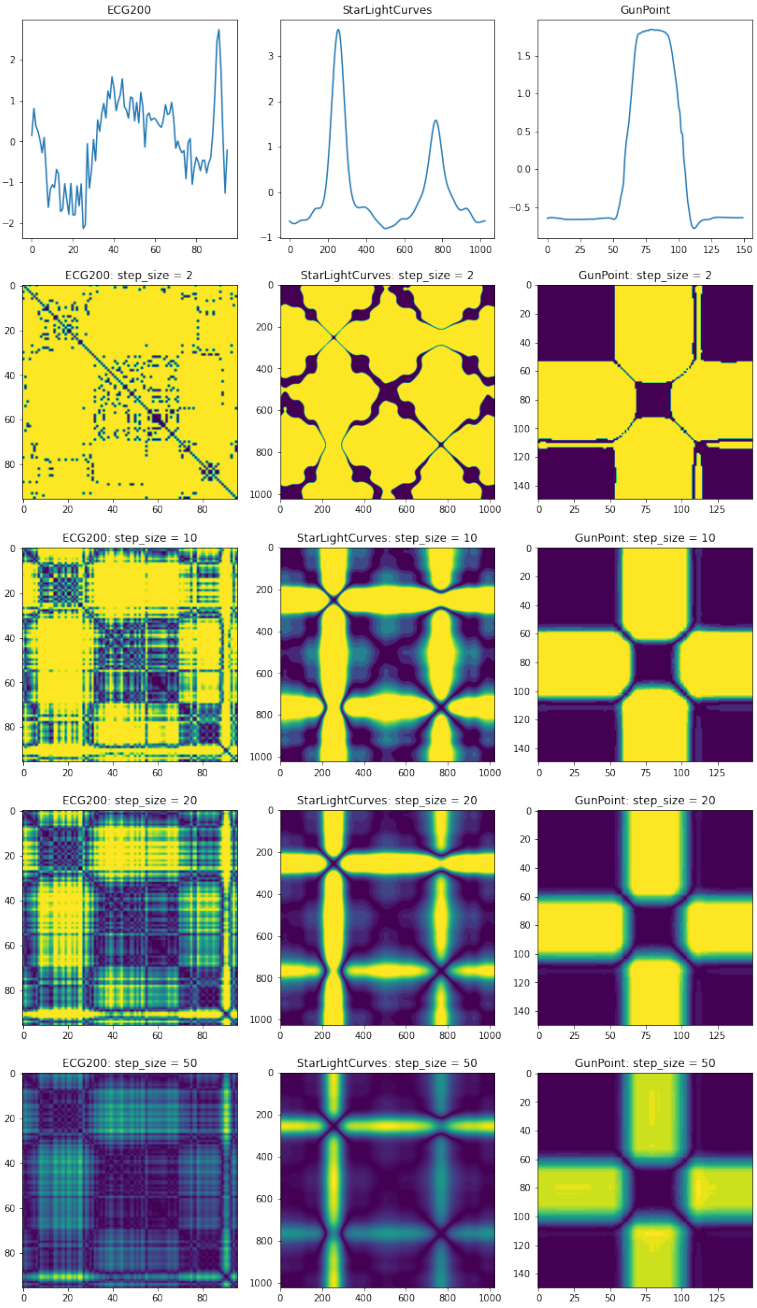

Abbildung 4.7: Rekurrenzplots der Datensätze ECG200 (links), StarLightCurves (mitte) und GunPoint (rechts) für Schrittweite = 1 (2. von oben), Schrittweite = 10 (3. von oben), Schrittweite = 20 (4. von oben) und Schrittweite = 50 (unten). Die Grafiken in der obersten Zeile zeigen die zu Grunde liegenden Zeitreihen.

Das Beispiel StarLightCurves zeigt zwei horizontale und vertikale Linien für die Schrittzahlen 10, 20 und 50, die den Maxima im Zeitbereich entsprechen (Abbildung 4.7). Während für Schrittweite = 10 die Linien gleich aussehen, zeigt der Rekurrenzplot mit Schrittweite = 50 Informationen über die Amplituden der beiden Peaks (die gelblichen Linien, die dem globalen Maximum entsprechen). In der GunPoint-Probe werden die scharfen Trennlinien zu Übergangsbereichen.

4.3 Evaluation

Die in den Experimenten verwendeten CNNs bestehen aus zwei conv. Schichten, gefolgt von einer average Pooling-Schicht und drei dense Schichten. Erste Tests mit den UCR-Datensätzen bestätigten, dass die Wahl der Pooling-Schicht starken Einfluss auf das Ergebnis nehmen kann beschrieben in Kapitel 4.1. Kombinationen von average pooling Schichten, nur max-Pooling-Schichten, beiden Typen gemischt und nur einer einzelnen max- oder average pooling Schicht wurden getestet. Insbesondere bei der ausschließlichen Verwendung von max-Pooling nahm die Klassifizierungsleistung bei den getesteten Datensätzen ab. Die besten Ergebnisse wurden bei der Verwendung von nur einer average-Pooling-Schicht nach der ersten conv. Schicht erzielt.

Beide conv. Schichten verfügen über 64 Filter mit einer Kernelgröße von 3 und ReLU als Aktivierungsfunktion. Die erste und zweite dense Schicht sind mit 32 und 16 Neuronen definiert und verwenden ebenfalls ReLU als Aktivierungsfunktion. Die Neuronenanzahl der letzten dense Schicht entspricht der Anzahl der Klassen für den gegebenen Datensatz und verwendet binäre / kategoriale Crossentropie für Zwei- / Mehrklassendatensätze und Sigmoid / Softmax als Aktivierungsfunktion.

Die Eingabe ist ein Rekurrenzplot einer Zeitreihenprobe der Form s, s, n , wobei s die Länge der Zeitreihe und n die Anzahl der Schritte ist. Die Eingabe-Zeitreihen sind normalisiert, so dass alle Werte zwischen 0 und 1 liegen. Die Netze werden für 100 Epochen trainiert mit einem Validierungssplit von 0.2 und einer Batchgröße von 10. Acc und F1-Score sind die Metriken, anhand derer die Klassifikationsergebnisse der Testdatensätze ausgewertet werden.

Tabelle 4.1 zeigt die Ergebnisse der Klassifikationsexperimente mit den CNNs.

Starke Verbesserungen in Acc und F1-Score konnten vor allem gegenüber den Ergebnissen des kNN Klassifikator auf den nicht kodierten Zeitreihendaten erreicht werden. Die Ergebnisse der unkodierten Datensätzen stellen die Benchmarkergebnisse dar und sind identisch mit den Benchmarks aus Kapitel 3.3 und Kapitel 3.4.

Insbesondere, StarlightCurves und TwoleadECG zeigten starke Steigerungen in Acc und F1-Score. 76.56 % Acc wurden in der besten Einstellung für TwoleadECG mit einem F1-score von 0.75 erreicht, verglichen mit 59.78 % Acc und F1-score von

Tabelle 4.1: Evaluationsergebnisse für die Klassifikation von Rekurrenzplots mit unterschiedlichen Schrittweiten und Schwellwertparametern ϵ.

Steps		1	1	1	10	10	10	20	20	20	50	50	50
ϵ		0.01	0.1	0.2	0.01	0.1	0.2	0.01	0.1	0.2	0.01	0.1	0.2
ECG200	Acc	0.71	0.80	0.81	0.81	0.84	0.87	0.72	0.85	0.85	0.74	0.88	0.90
	F1	0.69	0.77	0.78	0.78	0.81	0.85	0.59	0.83	0.83	0.67	0.87	0.89
ECG5000	Acc	0.88	0.8682	0.8718	0.8729	0.9029	0.894	0.8569	0.8984	0.9036	0.8871	0.9009	0.8951
	F1	0.36	0.36	0.36	0.36	0.41	0.41	0.35	0.41	0.4	0.37	0.41	0.39
GunPoint	Acc	0.88	0.8733	0.7933	0.8067	0.8067	0.78	0.8533	0.90	0.78	0.8133	0.80	0.88
	F1	0.88	0.87	0.79	0.81	0.80	0.78	0.85	0.90	0.77	0.81	0.79	0.89
InlineSkate	Acc	0.1636	0.1764	0.1688	0.1673	0.1636	0.1646	0.1673	0.2036	0.2056	0.1636	0.1891	0.1972
	F1	0.03	0.04	0.03	0.03	0.03	0.03	0.03	0.22	0.25	0.03	0.09	0.13
MoteStrain	Acc	0.5767	0.6957	0.7181	0.7069	0.8027	0.8235	0.6941	0.8171	0.8235	0.7468	0.8027	0.5687
	F1	0.44	0.65	0.72	0.7	0.8	0.8	0.65	0.82	0.82	0.74	0.8	0.54
Sony	Acc	0.4859	0.4859	0.4426	0.4859	0.5025	0.7288	0.6323	0.5574	0.7088	0.6356	0.6656	0.5108
	F1	0.44	0.44	0.34	0.44	0.44	0.73	0.63	0.52	0.71	0.63	0.66	0.45
StarlightCurves	Acc	0.9322	0.9253	0.9558	0.9182	0.9579	0.9534	0.9460	0.9508	0.9467	0.9491	0.9558	0.9451
	F1	0.9	0.9	0.93	0.9	0.94	0.93	0.92	0.93	0.93	0.93	0.93	0.92
TwoLeadECG	Acc	0.5584	0.7436	0.8235	0.6374	0.6927	0.7849	0.7586	0.7656	0.7524	0.7480	0.7612	0.7340
	F1	0.45	0.74	0.82	0.59	0.67	0.78	0.75	0.75	0.74	0.74	0.75	0.72
UWave	Acc	0.1136	0.1209	0.1209	0.1175	0.3353	0.1220	0.1192	0.1203	0.1195	0.1195	0.1212	0.1234
	F1	0.04	0.03	0.03	0.03	0.28	0.05	0.03	0.04	0.04	0.03	0.04	0.04

0.56 für den kNN-Klassifikator, trainiert mit den unkodierten Zeitreihen. Auf der anderen Seite sind die Ergebnisse für ECG5000, UWave und Motestrain deutlich schlechter ausgefallen als nach herkömmlicher Klassifikation mit SVM oder kNN.

Für die Wahl der Parameter kann festgestellt werden, dass für einige Datensätze eine Erhöhung von ϵ eindeutig vorteilhaft ist (bei einer festen Schrittweite) für andere hingegen nicht. So gibt es Datensätze, die mit $\epsilon = 0.1$ besser abschneiden als mit $\epsilon = 0.01$ und bei $\epsilon = 0.2$ jedoch in Acc und F1-Score wieder abfallen.

Der ECG200-Datensatz zeigt Verbesserungen in Acc mit steigendem ϵ für alle Schrittzahlen. Für Schritte = 50, verbessert sich die Acc sogar um 16 Prozentpunkte zwischen $\epsilon = 0.01$ und $\epsilon = 0.2$. Auch die F1-Scores sind für größere Schwellwerte höher. Die schlechtesten Ergebnisse für eine gegebene Schrittzahl werden mit Schritten = 1 und $\epsilon = 0.2$ erzielt. Hier liegt die Acc bei nur noch 81 %. Die besten Ergebnisse für diesen Datensatz wurden mit Schritten = 50 und $\epsilon = 0.2$ erzielt, bei denen die Acc 90 % beträgt und der F1-Score mit 0.89 ebenfalls maximal ist.

Die Acc für ECG5000 verhält sich chaotisch. Ein klarer Trend ist nicht erkennbar. Obwohl 90.36 % Acc in der besten Kombination (Schrittweite = 20, $\epsilon = 0.2$) erreicht werden, sind die F1-Scores für alle Experimente niedrig (um 0.4). Dies kann mit den starken Klassenungleichgewichten erklärt werden, die in diesem Datensatz auftreten. Die Analyse der Confusion-Matrix konnte hier belegen, dass lediglich drei der fünf vorhandenen Klassen gelernt wurden.

Der Sony-Datensatz zeigt für $\epsilon = 0.01$ schlechte Ergebnisse mit einer Acc von maximal lediglich 48.59 %, jedoch eine enorme Leistungssteigerung bei $\epsilon = 0.2$ und einer Schrittweite von 10. Hier liegt die Acc bei 72.88 % und der F1-Score bei 0.73 was einer enormen Leistungssteigerung verglichen mit dem Benchmark-Ergebnis

entspricht.

Die Ergebnisse für den StarLightCurves-Datensatz zeigen ein anderes Verhalten als die der bisher ausgewerteten Datensätze. Die Schwankungen in den Ergebnissen sind über alle Einstellungen hinweg relativ klein im Vergleich zu den zuvor beschriebenen Datensätzen. Der Unterschied zwischen dem besten und schlechtesten Ergebnis in Acc beträgt weniger als 4 % und der maximale Unterschied zwischen den F1-Scores für alle Parameterkombinationen liegt bei 0.04.

Der InlineSkate-Datensatz, der 7 Klassen hat, zeigt die besten Ergebnisse für Schrittweite = 20 und $\epsilon = 0.2$. Hier liegt die Acc bei nur 20.56 %, der F1-Score bei 0.25. Dies entspricht einer Verschlechterung in Acc und einer Verbesserung in F1-Score verglichen mit den Benchmarkergebnissen von kNN und SVM.

Der Datensatz UwaveGestureLibraryX (8 Klassen) erreicht bei allen Einstellungen eine Acc von etwa 12 %, außer bei Schrittweite = 10 und $\epsilon = 0.1$, wo die Acc auf 33.53 % ansteigt und der F1-Score 0.28 beträgt.

TwoLeadECG zeigte die beste Verbesserung der Acc für eine feste Schrittweite. Für Schrittweite = 1 verbesserte sich die Acc von 55.84 % für $\epsilon = 0.01$ auf 82.35 % für $\epsilon = 0.2$.

4.4 Diskussion

In den vorgestellten Experimenten dieses Kapitels wurde untersucht, ob mit Hilfe von aus Zeitreihen erstellten Rekurrenzplots bessere Klassifikationsergebnisse erzielt werden können, als wenn die Zeitreihen selbst als Trainingsdaten für einen Algorithmus verwendet werden. Hierzu wurden durch Veränderung von ϵ und der Schrittweite unterschiedliche Rekurrenzplot-Datensätze erzeugt und auf ihre Funktionalität hin untersucht. Auf Grund der unterschiedlichen Beschaffenheit von Rekurrenzplots im Gegensatz zu rohen Zeitreihen wurden für die Klassifikation von Rekurrenzplots CNNs verwendet.

In den Experimenten konnte gezeigt werden, dass in Rekurrenzplots kodierte Zeitreihen und anschließende Klassifikation durch ein CNN eine gute Methode zur Klassifikation darstellen können, jedoch sehr volatile Leistungsentwicklungen hinsichtlich Datensätzen mit unterschiedlichen Charakteristika aufweisen. Auch die Parametereinstellung bei der Erzeugung von Rekurrenzplots nimmt großen Einfluss auf die Funktionalität der vorgestellten Methode.

Ein einheitlicher Trend, welche Charakteristika der Datensätze vorteilhaft für Rekurrenzplots sind, war nicht zu erkennen. Die Ergebnisse in Acc und F1-Score für den GunPoint Datensatz konnten über alle Experimente im Vergleich zu den Benchmarkergebnissen verbessert werden, die des ECG200 Datensatzes hingegen weisen häufig Verschlechterungen auf. Beide Datensätze verfügen über 2 Klassen, eine ähnliche Samplelänge und eine ähnliche Anzahl von Test- und Trainingsdaten.

Auch die Datensätze MoteStrain und Sony, welche ähnliche Charakteristika aufweisen, unterscheiden sich deutlich in Ihrer Leistungsentwicklung. Die MoteStrain-Ergebnisse sind als schlecht zu bewerten, für den Sony Datensatz konnten jedoch sehr vielversprechende Ergebnisse mit den Parameterkombinationen $\epsilon = 0.2$ und einer Schrittweite von 10 oder 20 erreicht werden.

Darüber hinaus ist auch kein klarer Trend hinsichtlich der verschiedenen Parameterkombinationen erkennbar. Es wurden große Leistungsschwankungen für die meisten Datensätze, abhängig von der zu Grunde liegenden Parametereinstellung festgestellt. Wann eine Verbesserung und wann eine Verschlechterung auftritt, unterscheidet sich jedoch von Datensatz zu Datensatz. Auch bei Betrachtung der Ergebnisse eines einzelnen Datensatzes ist nicht immer ein Muster erkennbar. Die Datensätze ECG200, MoteStrain und TwoLeadECG profitieren beispielsweise eindeutig, je größer ϵ gewählt wurde. Die Ergebnisse von Sony und InlineSkate sind hingegen chaotisch, wenn ϵ gesteigert wird, die Schrittweite aber konstant bleibt.

Insgesamt lässt sich daher schließen, dass die vorgestellte Methode helfen kann, Klassifikationsergebnisse zu verbessern, die Funktionalität jedoch stark abhängig vom zu erweiternden Datensatz, sowie der Parameterwahl ist. Hinweise, welche Datensätze geeignet und welche Parameterkombination für welche Art von Datensatz erfolgsversprechend sind, konnten durch die Evaluation nicht aufgedeckt werden. So bleibt die Annahme bestehen, dass sich Rekurrenzplots vor allem für periodische Signale eignen. Werden diese im vorgestellten Rahmen eingesetzt, ist es erforderlich, verschiedene Parameterkombinationen zu prüfen, sowie die Leistungsentwicklung genau zu überwachen.

Teil II

Implementierung und Evaluierung von Verfahren zur Vorhersage und Vermeidung von Fehlerfällen an Schrittmotoren

Kapitel 5

Erstellung eines Dauerversuchsaufbaus zur Gewinnung neuer Daten zur Ausfall- und Anschlagserkennung von Schrittmotoren

RUL prediction ist eines der interessantesten Forschungsfelder im Bereich der künstlichen Intelligenz. Um Ausfälle mit hoher Präzision vorherzusagen, ist eine große Datenbasis über verschiedene Betriebszustände der zu überwachenden Komponente notwendig. In diesem Kapitel wird ein Dauerversuch entworfen, durch welchen Messdaten generiert werden, welche als Datenbasis, für die in Kapitel 6 und 7 durchgeführten Experimente dienen. In Kapitel 5.1 wird die Motivation für den Dauerversuchsaufbau erklärt sowie drei Arbeitshypothesen aufgestellt. Anschließend wird in Kapitel 5.2 auf die Anforderungen eingegangen, denen der Versuchsaufbau genügen soll. In Kapitel 5.3 wird die Umsetzung des Versuchsaufbaus gezeigt, ehe in Kapitel 5.4 die aufgetretenen Fehlerfälle während des Prüfzeitraums beschrieben werden. Kapitel 5.5 gibt einen Überblick über die gesammelten Daten.

5.1 Einführung

5.1.1 Motivation

SMs, die beispielsweise als Regelelement des Gasdurchflusses in Gasarmaturen von Gasthermen eingesetzt werden, sind eine der Hauptfehlerquellen in Gasheizsystemen. Funktioniert der SM nicht ordnungsgemäß, schalten Gasthermen in den Fehler-Modus, was bedeutet, dass ein Heiz- oder Warmwasser-Betrieb nicht weiter möglich ist [109]. Auch in der Robotik können SMs ausfallen, was häufig

enorme Kosten und ungewünschte Ausfallzeiten nach sich zieht. Es ist daher wünschenswert, den Verschleiß von SMs zu reduzieren, um die Anzahl von Fehlerfällen zu minimieren.

Da SM im normalen Betrieb regelmäßig genutzt werden und somit zwangsläufig Verschleiß auftritt, ist darüber hinaus das frühzeitige Erkennen von potenziellen Fehlerfällen interessant. Kann ein Defekt vor seinem Auftreten detektiert werden, können durch frühzeitige Information des Fachhandwerkers, Geräte-Ausfallzeiten für Firmen oder Endkunden vermieden werden. Darüber hinaus ist für Fachhandwerksbetriebe die frühzeitige Information über Ausfallzeitpunkt und Ausfallursache mit vielen Vorteilen verbunden. Liegen Informationen über den prognostizierten Ausfallzeitpunkt vor, können die Routen der Fachhandwerker effizienter geplant und so Fahrtkosten und schädliche Emissionen verringert werden. Zusätzlich können Transportkosten durch falsch bestellte Ersatzteile und dadurch entstehende Lagerhaltungskosten reduziert werden, was in beiden Fällen ebenfalls zu einer Verringerung von Emissionen beitragen kann. Durch frühzeitiges Wissen über die mögliche Defekt-Ursache, ist es den Fachhandwerkern möglich, direkt die passenden Ersatzteile zeitgercht zu bestellen und ohne weitere Lagerhaltung mitzuführen. So können mögliche Folgetermine vermieden werden, die in der Praxis durch nicht vorhandenes, fälschlicherweise bestelltes oder nicht mitgeführtes Ersatzteilmaterial regelmäßig auftreten.

5.1.2 Arbeitshypothesen

Mit Hilfe der durch den Dauerversuchsaufbau generierten Daten wird erwartet, dass Unterschiede oder Veränderungen im Betriebsverhalten über die Laufzeit festgestellt werden können. Dieser Erwartung liegen drei Annahmen zu Grunde:

- **Arbeitshypothese 1: Verändertes Verhalten bei Betrieb außerhalb des Arbeitsbereiches**
  Auf Grund der in Kapitel 2.1 beschriebenen Gegebenheiten, ist davon auszugehen, dass der Betrieb außerhalb des Arbeitsbereiches eine höhere Belastung des Motors verursacht, was zu einer Veränderung der Betriebsparameter führt. Es wird angenommen, dass mit Hilfe eines Klassifikationsalgorithmus, diese Veränderung erkannt werden kann. Somit wäre es möglich zu erkennen, wann der Schrittmotor außerhalb seines Arbeitsbereiches betrieben wird. Referenzfahrten könnten so verkürzt werden, was den Verschleiß des Schrittmotors im täglichen Betrieb erheblich reduziert.

- **Arbeitshypothese 2: Mechanischer Abrieb über Zeit**
  Bei sich berührenden Körpern, die gegeneinander bewegt werden, tritt Reibung auf. Diese Reibung bildet ein Drehmoment entgegen der Bewegungsrichtung. Darüber hinaus kann bei weicheren Materiealien durch Reibung

vermehrt Abrieb entstehen. In Schrittmotoren ist es häufig der Fall, dass die Antriebswelle aus Eisen auf ein Gegengewinde aus Plastik trifft. Im Laufe der Zeit bilden sich so Abreibungen des Plastik-Gegengewindes, welche zu einem Defekt führen können. Es ist davon auszugehen, dass dieser Effekt im Laufe der Zeit Einfluss auf die Betriebsparameter hat. Mit Hilfe einer Klassifikation ist davon auszugehen, dass Daten, welche von einem Schrittmotor mit wenig Abrieb aufgenommen worden sind, unterschieden werden können, von Daten, die von einem Schrittmotor mit viel Abrieb aufgenommen worden sind.

- **Arbeitshypothese 3: Reduzierung des Motor-Schmierfetts über Zeit** Um das in Arbeitshypothese 2 beschrieben Verhalten zu verhindern, wird herstellerseitig Schmierfett zwischen Rotor und Antriebswelle aufgebracht. Es wird davon ausgegangen, dass durch die schneckenförder-ähnliche Arbeitsweise der Antriebswelle, Schmierfett während des Betriebs aus dem zu schmierenden Bereich befördert werden kann. Dazu wird vermutet, dass, sollte mechanischer Abrieb entstehen, sich dieser mit dem Schmierfett vermischt und die Schmierwirkung beeinträchtigt. Verringert sich die Schmierwirkung, benötigt der Motor mehr Kraft, um die Antriebswelle zu bewegen. Steigt das erforderliche Antriebsmoment für das Bewegen durch reduzierte Schmierung über das Antriebsmoment des Motors, kann der Motor die Antriebswelle nicht mehr bewegen und es liegt ein Defekt vor.

5.2 Anforderungen

Ziel des Dauerversuchsaufbaus ist es, Daten zu generieren, mit denen sich die Arbeitshypothesen überprüfen lassen. Um Arbeitshypothese 1 überprüfen zu können, müssen Daten sowohl vom Betrieb im normalen Arbeitsbereich als auch Daten vom Betrieb außerhalb des normalen Arbeitsbereichs aufgezeichnet werden. Für Arbeitshypothese 2 ist es notwendig, den Dauerversuch so zu gestalten, dass er Daten über den kompletten Lebenszyklus eines Schrittmotors, also vom neuwertigen Gerät bis hin zum Defekt liefert. Für Arbeitshypothese 3 muss eine ausreichend lange Betriebsdauer des Motors vorliegen. Anschließend kann die Vermischung zwischen Abrieb und Schmierfett durch eine Detail-Analyse der SM-Komponenten festgestellt werden.

Für alle Arbeitshypothesen ist es zur verbesserten Abbildung des Realbetriebs sinnvoll, die jeweiligen Daten bei unterschiedlichen klimatischen Bedingungen aufzunehmen. Auch die Zeitkonsistenz der Messreihen ist eine wesentliche Anforderung. Die Daten müssen in einer ausreichend hohen Frequenz aufgenommen werden. Damit werden einerseits Informationsverluste vermieden und andererseits

den Algorithmen eine gute Trainingsgrundlage geboten. Um eine für die Klassifizierung ausreichende Anzahl Messdaten zu generieren, ist es erforderlich, dass der Versuchsaufbau die gleichzeitige Vermessung einer ausreichenden Anzahl Prüflinge zeitgleich zulässt. Eine möglichst rauscharme Messung ist wünschenswert.

5.3 Umsetzung des Dauerversuchsaufbaus

Dieses Kapitel beschreibt die Umsetzung des Dauerversuchaufbaus. Hierzu wird in Kapitel 5.3.1 zunächst auf Wahl des Prüflings eingegangen, ehe in Kapitel 5.3.2 der Aufbau des Dauerversuchs detailliert beschrieben wird. Kapitel 5.3.3 gibt Aufschluss über den Betrieb.

5.3.1 Auswahl des Prüflings

Als zu untersuchender Motor wurde ein unipolarer, permanentmagnetischer SM ausgewählt. Dieser findet häufig Anwendung in unterschiedlichen industriellen Prozessen und Maschinen. Er kommt beispielsweise in Gasthermen und Wärmepumpen zum Einsatz und ist somit ein geeignetes Prüflingsmodell. Die generelle Funktionsweise eines solchen SM ist in Kapitel 2.1 beschrieben.

Das ausgewählte SM-Modell verfügt über zwei Phasen und eine Polpaarzahl von 12, woraus sich ein Schrittwinkel von $7,5°$ pro Schritt ergibt. Der Motor wird im 2-Phasen-Vollschritt betrieben und hat einen Drehzahlbereich von 200 $\frac{1}{s}$ [109]. Abbildung 5.1 zeigt ein zerlegtes Modell des verwendeten SMs.

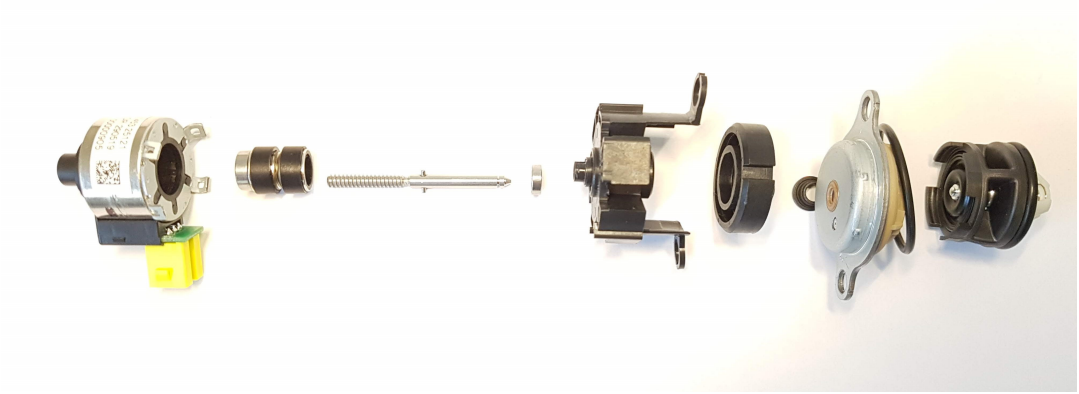

Abbildung 5.1: Zerlegter permanentmagnetischer SM

Zu erkennen sind der Stator, der Rotor, die Antriebswelle, ein Gegenstück aus Plastik, das Lager sowie ein metallischer Endblock. Hervorzuheben ist, dass das Gegengewinde des Rotors aus Plastik besteht, die Antriebswelle jedoch aus

Eisen. Dies stützt Arbeitshypothese 2, in der von einem mechanischen Abrieb über Zeit ausgegangen wird. Das Schmierfett, welches aufgetragen ist um diesem Effekt vorzubeugen und einen reibungsarmen Betrieb zu gewährleisten, ist herstellerseitig im Rotor eingebracht.

5.3.2 Aufbau

Der Versuchsaufbau bietet die Möglichkeit 32 Schrittmotoren dauerhaft zu betreiben, nach Wunsch anzusteuern und für jeden Schrittmotor sechs unterschiedliche Signale zu erfassen. Die Hauptkomponenten des Versuchsaufbaus sind die 32, auf einem Metallgestell verschraubten Prüflinge, eine für diesen Versuchsaufbau entworfene Schaltung zur Signalverarbeitung, eine zentrale Steuer- und Recheneinheit sowie 2 Labornetzteile zur Spannungsversorgung.

Als zentrale Steuer- und Recheneinheit für Dauer- und Messbetrieb wird die MicroLabBox (MLB) des Herstellers dSpace genutzt. Die MicroLabBox ist ein echtzeitfähiges System mit zahlreichen digitalen und analogen Ein- und Ausgängen und dient hauptsächlich der Umsetzung von Regel-, Test- und Messanwendungen [110]. Der in der MicroLabBox implementierte Algorithmus zur Ansteuerung und Messung der Gasventile wurde mit MATLAB und Simulink entwickelt. Abbildung 5.2 zeigt ein Bild des Versuchsaufbaus mit den genannten Hauptkomponenten.

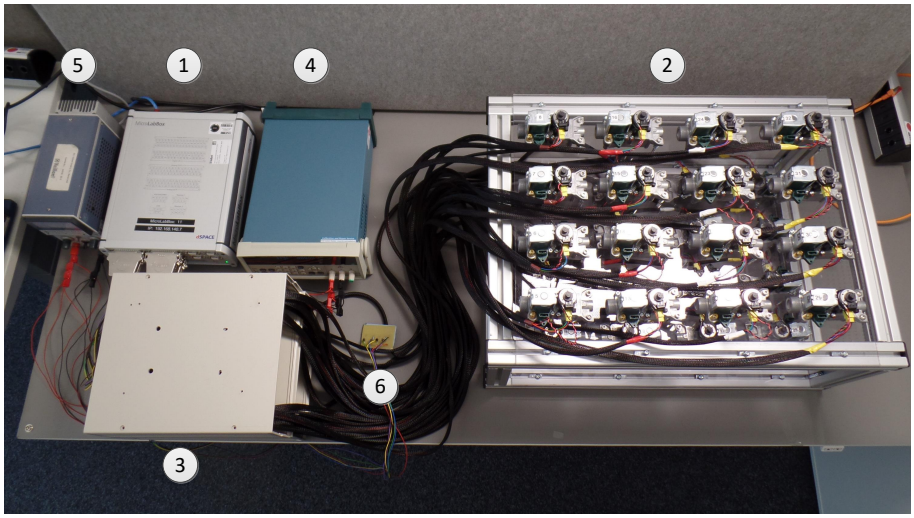

Abbildung 5.2: Hauptbestandteile des Dauerversuchaufbaus. Zu sehen ist die MLB (1), das Gestell mit den SMs (2), das Gehäuse, in dem die Platinen verbaut sind (3), die zwei Labornetzteile mit 24 V (4) und 5 V (5) und die Platine mit den Klimasensoren (6).

5.3.2.1 Elektrische Schaltung

Bei der für diesen Versuch entworfenen Schaltung handelt es sich um eine komplementäre Darlington-Schaltung, mit der große Spannungsverstärkungen erreicht werden können. Mit dieser Schaltung ist es möglich, acht SMs pro Platine seriell zu vermessen. Auf der Eingangsseite sind auf allen Platinen Eingänge für Spannungsversorgungen mit 24 V und -5 V. Für die Steuersignale der Treiberschaltung und die Steuersignale der Multiplexer, die von den digitalen Ausgängen der MLB kommen, sind weitere Eingänge auf den Platinen verbaut. Die einkommenden Messungen der Klimasensorik und der Schrittmotorsensorik werden auf der Platine aufbereitet.

Ausgangsseitig befinden sich die aufbereiteten Signale der Sensorik der SMs, welche zu den analogen Eingängen der MLB geleitet werden. Des Weiteren hat die erste Platine eine Steckverbindung zu einer Platine, auf der Klimasensoren verbaut sind. Die Stromversorgung dieser Platine erfolgt über die erste Platine. Auf der Platine mit den Klimasensoren sind jeweils zwei Luftfeuchtigkeits- und Temperaturfühler verbaut. Diese werden mit 5V versorgt und liefern einen Kontrollwert für die klimatischen Bedingungen, denen der Versuchsaufbau ausgesetzt ist. Der schematische Aufbau des Versuchsaufbaus ist in Abbildung 5.3 zu sehen. Bei den schwarzen Pfeilen handelt es sich um Steuersignale für die SMs, bei den

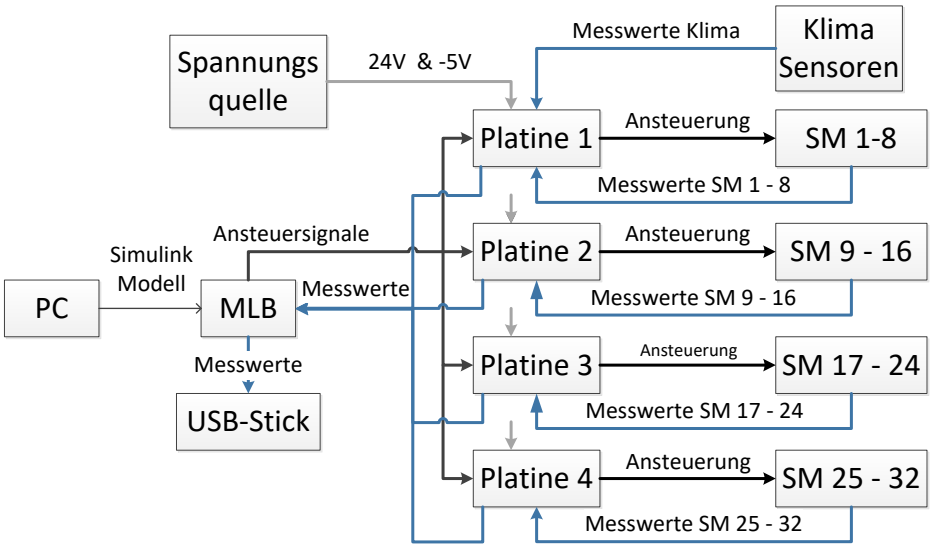

Abbildung 5.3: Aufbauschema des Dauerversuchaufbaus. Zu sehen sind die Hauptkomponenten, die vier Platinen sowie die 32 SMs. Die schwarzen Pfeile stellen Steuersignale dar, die blauen symbolisieren Messwerte.

blauen Pfeilen handelt es sich um die jeweiligen Messwerte. Abbildung 5.4 zeigt die elektrische Schaltung, die dem Versuchsaufbau zu Grunde liegt. Die Dioden

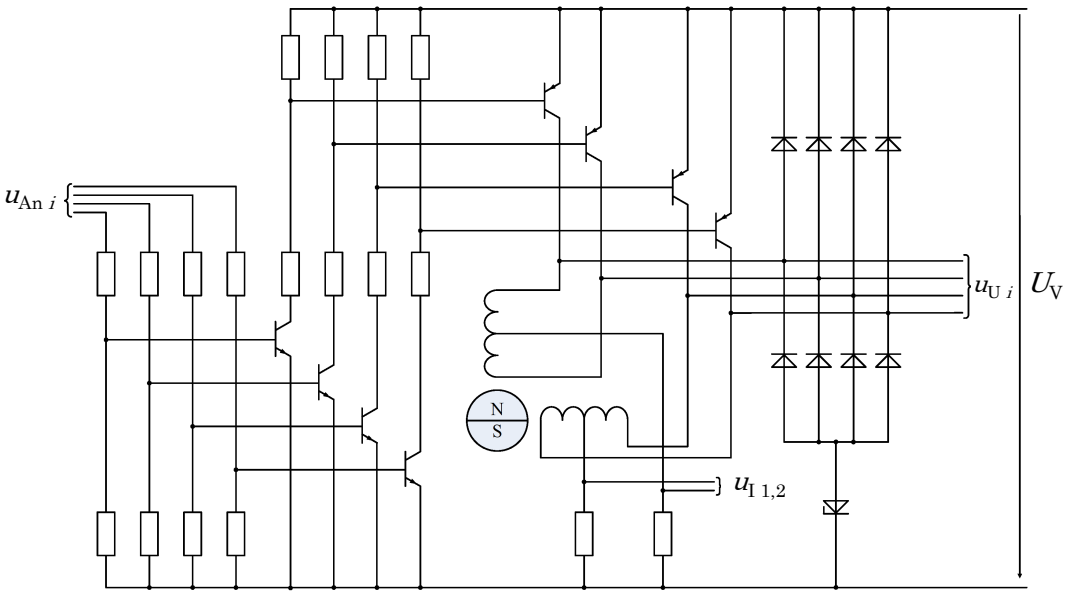

Abbildung 5.4: Elektrische Schaltung des Dauerversuchaufbaus. Zu erkennen sind die Verschaltung der Bauteile sowie die Abgriffe für die aufgezeichneten Signale.

verfügen über eine Durchbruchspannung von $U_D = 0.7V$, die Zener-Dioden über eine Zener-Spannung von $U_Z = 9.1V$. Alle verwendeten Widerstände besitzen eine Größe von $R = 20\ \Omega$. Für die Ansteuerung der MLB dienen die Spannung u_{Ani} mit $i = [1,2,3,4]$. Diese nehmen entweder ein Potential von $u_{Ani} = 0V$ oder $u_{Ani} = 5V$ an. Die Strommessung erfolgt über die Mittelanzapfungen der Motorinduktivitäten. Hierzu werden Shunt-Widerstände genutzt, deren Spannungen über den Abgriff $u_{I1,2}$ mit Hilfe von Impedanzwandler durch die MLB erfasst wird. Die Spannungsmessung erfolgt an Hand der Messstelle u_{Ui}, ebenfalls über Impedanzwandler.

Insgesamt erlaubt der Prüfstand somit die Vermessung der folgenden Signale:

- Signal 1: I1, Strom Spule 1, gemessen über Shunt-Widerstand
- Signal 2: I2, Strom Spule 2, gemessen über Shunt-Widerstand
- Signal 3: U1, Spannung Spule 1, Phase 1
- Signal 4: U2, Spannung Spule 1, Phase 2
- Signal 5: U3, Spannung Spule 2, Phase 1

- Signal 6: U4, Spannung Spule 2, Phase 2

5.3.3 Betrieb

5.3.3.1 Betriebszyklus und Messungen

Der Ansteuerung der Motoren im Prüfstand liegt ein für diesen Versuch entworfener Betriebszyklus zu Grunde, der die Schrittfolge der SM vorgibt. Der Messzyklus sowie die zugehörigen Betriebsmodi der SM sind in Abbildung 5.5 gezeigt.

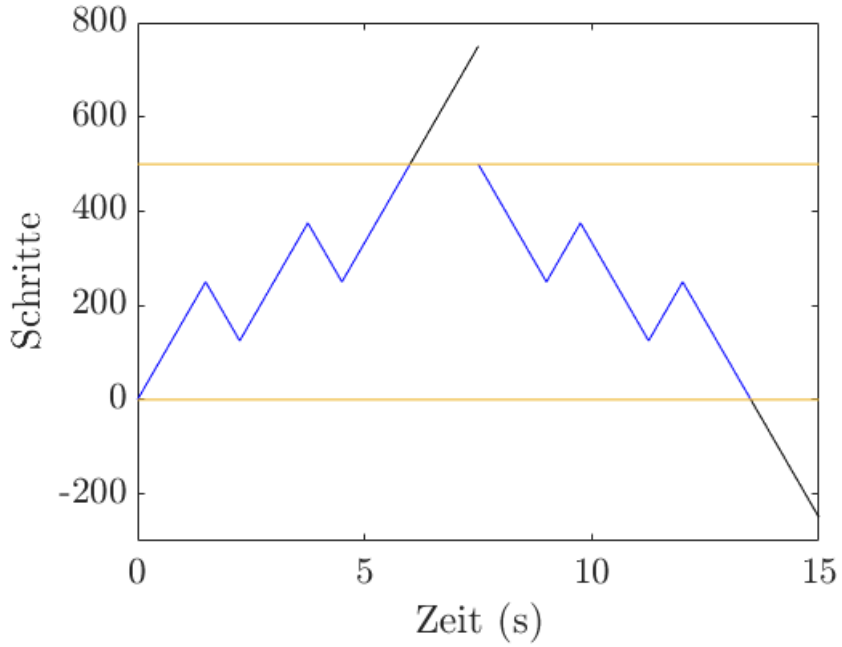

Abbildung 5.5: Betriebszyklus der SMs. FH und FR sind in blau gekennzeichnet, OA und UA in schwarz. Die gelben Linien umschließen den normalen Arbeitsbereich.

Der in dem Prüfstand implementierte Messzyklus steuert die Schrittmotoren so an, dass die gesamte Spannweite des Arbeitsbereiches vom vollständig geschlossenen bis zum vollständig geöffneten Zustand und wieder zurück in den geschlossenen Zustand durchfahren werden. Sowohl bei der öffnenden Fahrt als auch bei der schließenden Fahrt wurden zusätzlich vier Fahrtrichtungswechsel implementiert. Das Prinzip ist eine beginnende Öffnungsfahrt mit einer kurzen Schließfahrt. Dieses Verhalten wurde integriert, um anfallende Richtungswechsel in der Realität nachzubilden, da diese in praktischen Anwendungen üblich sind. Zusätzlich werden die Motoren so angesteuert, dass diese deutlich weiter geöffnet sowie geschlossen

werden, als es mechanisch möglich ist um das Verhalten einer Referenzfahrt zu simulieren. In diesem Bereich fallen die Motoren kontinuierlich außer Tritt. Während des 15-sekündigen Messzyklus bzw. der Vermessung verfahren die Motoren 2500 Schritte mit Geschwindigkeit von $6ms$ pro Schritt.

Auf Grund der hohen Datenmenge, die bei einer dauerhaften Vermessung anfallen würde, wird die Vermessung in regelmäßigen Abständen durchgeführt. Hierfür werden die sechs Signale eines Zyklus sechs mal innerhalb von 28 Stunden aufgezeichnet und dienen als Datenbasis für die geplanten Experimente.

5.3.3.2 Betriebsbedingungen

Da Umgebungsbedingungen wie Temperatur und Luftfeuchtigkeit Einfluss auf die Stromsignale der Motoren haben, erfolgt sowohl der Dauerbetrieb als auch die Vermessung der Motoren in einer klimatisierten Einheit. So können verschiedene Realbedingungen simuliert werden. Die klimatisierte Einheit erlaubt die Einstellung konkreter Temperaturen und Luftfeuchtigkeit. Abbildung 5.6 zeigt ein Bild des Versuchsaufbau in der klimatisierten Einheit. Folgende klimatische Bedingun-

Abbildung 5.6: Bild des in die klimatisierte Einheit integrierten Dauerversuchsaufbaus.

gen wurden für die komplette Betriebszeit des Versuchsaufbaus ausgewählt:

- Temperatur 10 °C, Luftfeuchtigkeit 80 %
- Temperatur 20 °C, Luftfeuchtigkeit 60 %

- Temperatur 50 °C, Luftfeuchtigkeit 40 %

Die drei klimatischen Bedingungen wechseln sich hierbei in einem 28-Stunden Rhythmus ab, so dass ein Klimazyklus entsteht, der alle drei klimatischen Zustände im Laufe einer Betriebswoche zwei Mal durchläuft. Zusätzlich wird durch die wechselnden klimatischen Bedingungen ein beschleunigter Alterungsprozess der Motoren erzeugt.

Es muss angenommen werden, dass unter Laborbedingungen die Belastungen durch Luftverunreinigungen und dadurch entstehende Ablagerungen in den Motoren geringer ausfällt als im Realbetrieb. Eine Zuführung von Staub oder ähnlichen Substanzen lässt die Beschaffenheit der klimatisierten Einheit nicht zu und kann daher nicht integriert werden.

5.3.3.3 Betriebszeit

Die Motoren wurden insgesamt über ca. 8400 Stunden nach dem in Kapitel 5.3.3.1 beschriebenen Zyklus betrieben und vermessen. Der Dauerversuch befand sich dabei immer innerhalb der klimatisierten Einheit mit den beschrieben klimatischen Bedingungen. Über den Betriebszeitraum konnten so 1802 Messreihen für jeden SM generiert werden.

5.4 Fehlerfälle

In diesem Kapitel wird ein Überblick über die aufgetreten Fehlerfälle gegeben und es erfolgt eine Analyse hinsichtlich der beschriebenen Arbeitshypothesen.

Insgesamt sind während des Dauerversuchs 20 von 32 Schrittmotoren ausgefallen. Tabelle 5.1 zeigt die ausgefallenen Motoren und die Messreihe, in der ein Defekt erstmals festgestellt werden konnte. Nach Abschluss des Dauerversuchs wurden die defekten Motoren zerlegt und mit Hilfe eines Mikroskops hinsichtlich der Arbeitshypothesen 2 und 3 analysiert. Abbildung 5.7 zeigt verschiedene Bilder von neuwertigen und defekten Komponenten der SMs.

In Abbildung 5.7 ist auf dem Bild oben links zu erkennen, dass das Schmierfett, zu erkennen als weiß-bläuliche Substanz, im neuwertigen Motor gleichmäßig im Rotor verteilt ist. Oben rechts ist die Antriebswelle eines unbenutzten Motors dargestellt. Hier ist zu erkennen, dass keine Ablagerungen, Rückstände oder ähnliches vorhanden sind.

In der Mitte und Unten sind ausgewählte Bilder von Komponenten defekter Motoren gezeigt. Auf der linken Seite ist zu erkennen, dass das ehemalig weißbläuliche Schmierfett zu einer braun-schwarzen Masse geworden ist. Dazu haben sich Ablagerungen im oberen Teil des Rotors gebildet, die so nicht mehr zur

Tabelle 5.1: Übersicht über die Ausfallzeiten der SM. Hierzu ist die Messung und die zugehörige Betriebszeit gegeben, in der ein Ausfall eines SM erstmals aufgezeichnet wurde.

SM Nr.	Ausfallmessung	Betriebszeit (h)	SM Nr.	Ausfallmessung	Betriebszeit (h)
1	1679	7835	17	-	-
2	-	-	18	-	-
3	-	-	19	-	-
4	1480	6906	20	-	-
5	1619	7555	21	958	4470
6	-	-	22	1799	8395
7	1013	4727	23	904	4218
8	904	4218	24	658	3070
9	1013	4727	25	1552	7242
10	940	4386	26	1605	7490
11	1605	7490	27	904	4218
12	-	-	28	922	4302
13	1013	4727	29	-	-
14	958	4470	30	-	-
15	-	-	31	922	4302
16	-	-	32	1015	4736

Schmierung beitragen können. Die Farbveränderung wird auf die klimatischen Belastungen sowie die Vermischung mit den in Arbeitshypothese 2 beschrieben Abreibungen zurückgeführt. Dieser Eindruck bestätigt sich durch ein Konsistenz-Vergleich der neuwertigen Schmiermasse mit der Schmiermasse defekter Motoren. Hier konnte eine erhebliche Änderung festgestellt werden. Die braun-schwarze Schmiermasse war wesentlich zäher und ließ sich nicht ohne schwarze Partikel-Rückstände verreiben. Die Partikel-Rückstände unterstützen die Annahme der Abreibung des im Rotor befindlichen Kunststoff-Gegengewindes über die Betriebszeit.

Auch die in Abbildung 5.7 auf der rechten Seite dargestellten Bilder genutzter Antriebswellen bestätigen die getroffenen Arbeitshypothesen. Hier kann gut die in Arbeitshypothese 3 aufgezeigte Problematik des Schneckenfördermechanismus erkannt werden. Schmierfett wird aus dem zu schmierenden Bereich des Rotors abtransportiert und an Stellen befördert (vor allem in den Abbildungen links zu erkennen), an denen es keine Funktion erfüllt. Das hier abgelagerte Schmierfett

ist als blass-rosa zu erkennen. Die blass-rosane Farbe wird auf Oxydation und Feuchtigkeit zurückgeführt. Dass hier keine braun-schwarze Farbgebung vorliegt unterstützt wiederum die Abriebshypothese, da Abrieb im Arbeitsbereich produziert wird und nur hier das Schmierfett durchsetzt.

Der Grundsatz von Arbeitshypothese 2 und 3 hat sich somit bestätigt. Es gilt weiterhin zu überprüfen ob die festgestellten Effekte auch Einfluss auf die Betriebsparameter haben und die Änderung mit Hilfe künstlicher Intelligenz erkannt werden kann.

5.5 Überblick über die gewonnenen Daten

Durch die regelmäßige Vermessung der Motoren über die komplette Betriebsdauer des Dauerversuchsaufbaus ist eine große Datenbasis entstanden, die sowohl Daten unterschiedlicher Signale, unterschiedlicher Arbeitsmodi sowie unterschiedlicher Alterungszustände beinhaltet.

Der in Kapitel 5.3.3.1 beschriebene Zyklus wird in 28 Stunden sechs mal zur Vermessung aller 32 Motoren genutzt. Eine Messung umfasst hierbei 2500 Schritte eines jeden Schrittmotors und dauert insgesamt 15 Sekunden. Da die Signale mit einer Frequenz von 10kHz aufgezeichnet werden, entstehen so Messreihen mit 150.000 Datenpunkten je Signal und Motor.

Während der Betriebszeit des Prüfstandes sind insgesamt 1802 Messreihen je Signal und Motor entstanden. Von diesen Messreihen können jedoch nicht alle verwendet werden, da Messreihen die nach einem Defekt eines Motors aufgezeichnet wurden für die Experimente in dieser Arbeit keine Anwendung finden. Für jeden Motor können alle Messreihen verwendet werden, die vor seiner Ausfallmessung aufgenommen worden sind. Insgesamt können 45.087 Messreihen mit je 150.000 Datenpunkten zur Analyse genutzt werden.

Anschließend werden alle Messreihen zu Datensamples mit einer Samplelänge von $SL = 240$ zerschnitten. Diese Länge wurde ausgewählt, da es genau einem Vollschritt des SM entspricht und sich somit für reale Anwendungen eignet. Die Verwendung einer vollständigen Messreihe mit 150.000 Datenpunkten wird auf Grund Ihrer Größe als nicht sinnvoll eingestuft. Es wird sich vorbehalten, die Datenstruktur für weitere Experimente nochmals anzupassen.

Durch die Festsetzung der Samplelänge auf 240 Datenpunkte stehen insgesamt ca. 169 Millionen Datensamples zur Verfügung, welche das Datenset $\mathbf{X} \in \mathbb{R}^{169\mathrm{M}\times 240\times 1}$ ergeben.

Anschließend wird $\mathbf{X}$ in der Form

$$\mathbf{X} := \frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})} \tag{5.1}$$

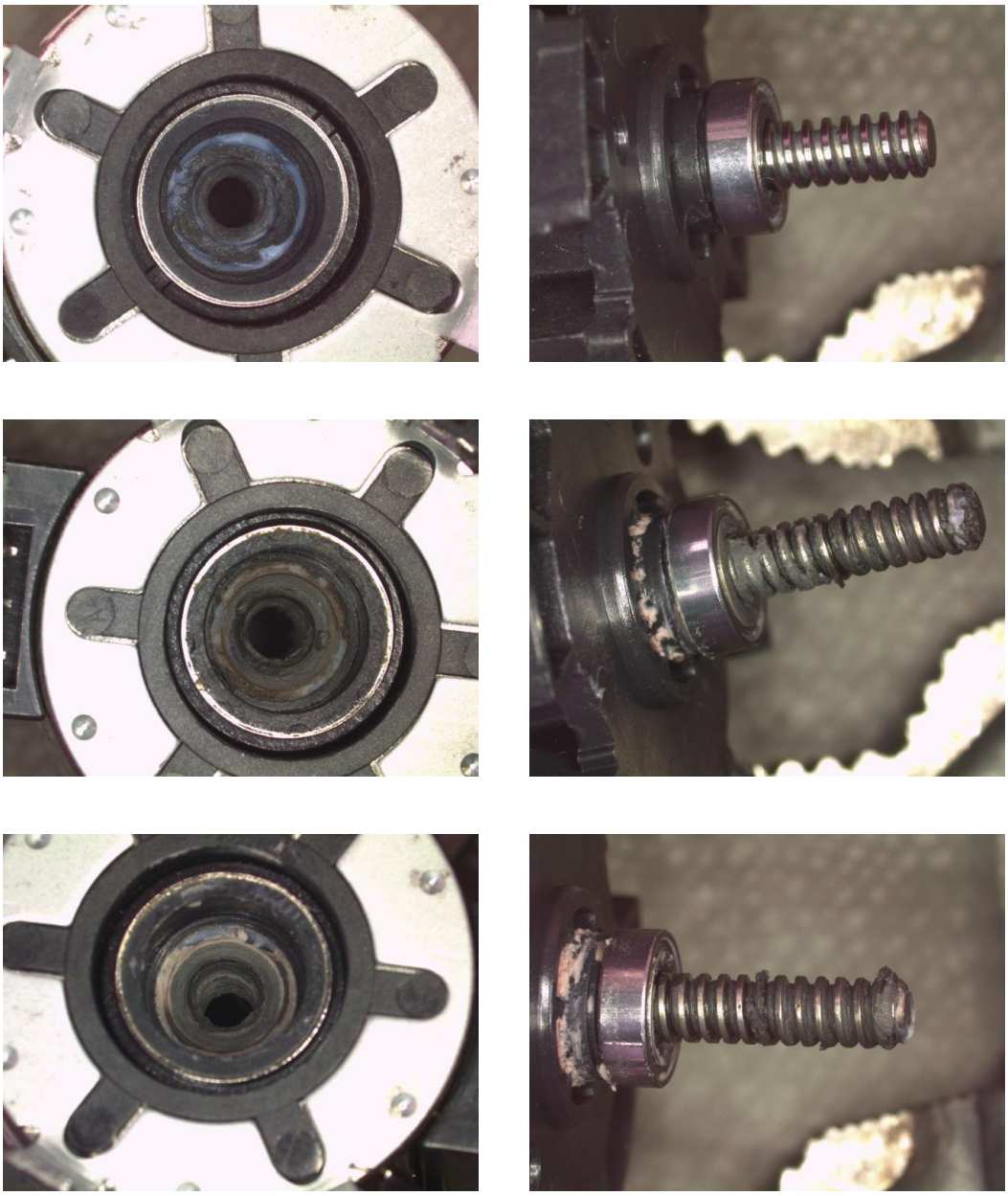

Abbildung 5.7: Mikroskop-Aufnahmen neuwertiger (oberste Reihe) und abgenutzer Komponenten (mittlere und untere Reihe) des SM. Links: Rotor, rechts: Antriebswelle.

auf Werte zwischen $[0, 1]$ skaliert, was zu $\mathbf{X} \in [0, 1]^{169\mathrm{M} \times 240 \times 1}$ führt.

Abbildung 5.8 zeigt Ausschnitte von Messreihen der Signale bei den unterschiedlichen klimatischen Bedingungen.

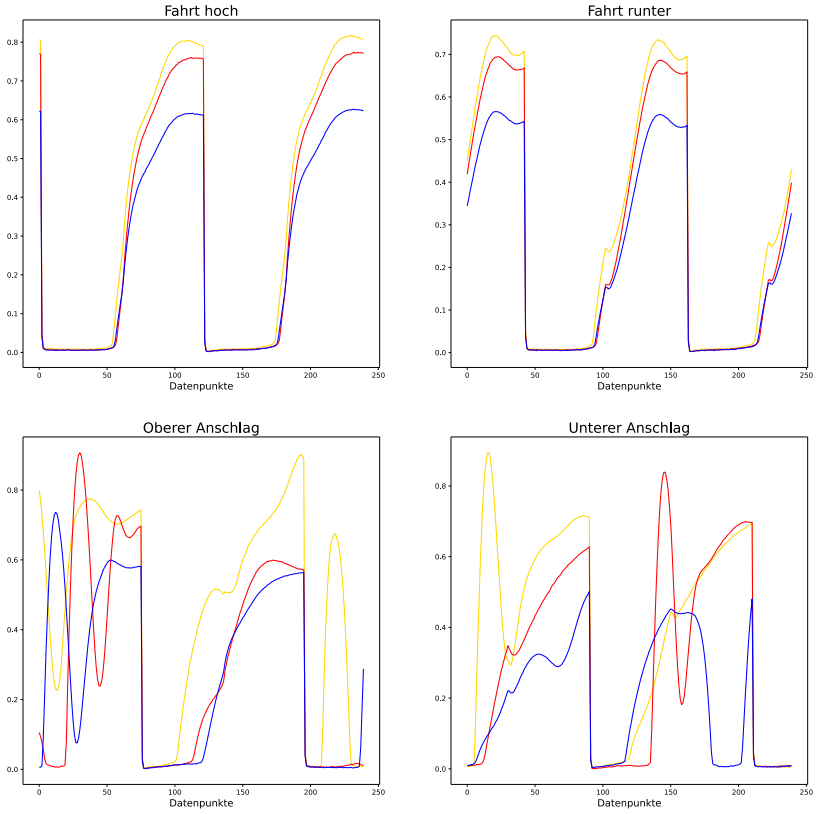

Abbildung 5.8: Beispielhafte Stromverläufe von Signal 1 für alle vier Betriebsmodi. Gezeigt werden Daten für alle klimatischen Bedingungen: rot 50°C, 40 %, gelb 20°C, 60 % und blau 10°C, 80 %.

Einschränkungen

Auf Grund der enormen generierten Datenmenge, wurde eine Voruntersuchung zur Eignung der unterschiedlichen Signale durchgeführt. Hierbei hat sich gezeigt, dass Signal 1 (I1, Spule 1, gemessen über Shunt-Widerstand) für Klassifikationsaufgaben am besten geeignet zu sein scheint. Diese Annahme bestätigte sich auch bei weitergehenden Untersuchungen und ist ebenfalls der Arbeit [111] zu entnehmen. Aus diesem Grund wird für die weiteren Experimente dieser Arbeit lediglich Signal 1 genutzt, was zu $\mathbf{X} \in \mathbb{R}^{28\mathrm{M}\times 240\times 1}$ führt. Die anderen Signale werden aus der Betrachtung ausgeschlossen.

Kapitel 6

Realisierung einer innovativen Anschlagserkennung für Schrittmotoren

In diesem Kapitel wird eine innovative Anschlagserkennung für den in Kapitel 5 untersuchten Schrittmotor entworfen. Hierzu wird auf Basis der generierten Daten ein Klassifikator trainiert, der zwischen dem normalen Betrieb und dem Betrieb über den mechanischen Anschlag hinaus, unterscheidet. Nachfolgend erfolgt in Kapitel 6.1 zuerst die Beschreibung der Problemstellung, ehe in Kapitel 6.2 die genutzte Datenbasis vorgestellt wird. In Kapitel 6.3 wird die Evaluation durchgeführt. Abschließend werden die Ergebnisse in Kapitel 6.4 diskutiert.

6.1 Problemstellung

Schrittmotorantriebe stellen eine vergleichsweise günstige und exakte Positionierungsmöglichkeit dar. Die Antriebe können ohne teure Positionsgeber und Regelelektronik betrieben werden. Es können allerdings Betriebsbedingungen auftreten, die zu einem Schrittverlust führen, sodass es für den gesteuerten Betrieb nötig ist, mit Hilfe einer Referenzfahrt eine Referenzposition zu ermitteln. Diese Referenzposition stellt je nach Anwendungsfall entweder der untere oder der obere mechanische Anschlag des SMs dar. Bei diesen Referenzfahrten sind die mechanischen Belastungen hoch, da der SM weit über seinen mechanischen Anschlag hinaus angesteuert wird. Diese sollten daher so kurz wie möglich gehalten werden.

Auch der in Kapitel 5 untersuchte SM ist in seinen typischen Einsatzgebieten Referenzfahrten ausgesetzt. Beispielsweise führen SMs in Gasgeräten auf Grund der Wichtigkeit der exakten Positionierung, im täglichen Betrieb regelmäßig Referenzfahrten in die Anschläge aus. Hierbei werden die SMs jeweils mindestens

200 Schritte über ihren normalen Arbeitsbereich hinaus betrieben [112]. Dieses Verhalten ist ebenfalls in dem beschriebenen Dauerversuch integriert worden.

Die Motoren erfahren so regelmäßig eine erhöhte Belastung, was zu erhöhtem Verschleiß und Reparaturen führt oder ein Austauschen der Motoren notwendig macht. Darüber hinaus erhöht sich der Geräuschpegel des SMs, wenn dieser außerhalb seines normalen Arbeitsbereiches betrieben wird.

Unternehmen sind daran interessiert, den Verschleiß der verbauten Komponenten soweit wie möglich zu reduzieren, um Reparaturen zu vermeiden und so Kosten sowohl für die Kunden als auch für das Unternehmen selbst zu sparen. Durch einen geminderten Geräuschpegel lässt sich zusätzlich der Komfort für den Endkunden erhöhen.

Es ist daher das Ziel, mit Hilfe der in Kapitel 5 generierten Daten, eine Anschlagserkennung für den untersuchten Schrittmotor zu entwerfen. Diese soll es möglich machen, Referenzfahrten in Zukunft zu verkürzen, indem die Betriebszeit außerhalb des normalen Arbeitsbereichs soweit wie möglich reduziert wird. Eine Vermeidung von Referenzfahrten ist nicht sinnvoll, da eine exakte Positionierung für den zuverlässigen Betrieb des SM eine sehr wichtige Komponente darstellt und nur so die exakte Position des SMs sichergestellt werden kann. Als Werkzeug zur Umsetzung der Anschlagserkennung bietet es sich an, Algorithmen aus dem Bereich der künstlichen Intelligenz einzusetzen. Die wichtigsten Grundlagen wurden hierzu im Kapitel 2.3 beschrieben.

6.1.1 Beschreibung des mechanischen Anschlags

Während des Fahrens strebt der Rotor eine Ausrichtungn entsprechend dem Statormagnetfeld an. Die resultierende induzierte Spannung verringert die Anstiegsgeschwindigkeit des Stroms. Fällt der Motor ausser Tritt, führt dies zu einem abweichenden Verhalten. Abhängig vom Winkel und der Kreisfrequenz kann dies den Strom so anregen, dass dieser weit über den Stillstandsstrom hinaus ansteigt oder den bereits aufgebauten Strom wieder verringert. In Abbildung 6.1 wird der Unterschied zwischen FH (links) und OA (rechts) an dem Stromsignal I1 veranschaulicht. Dieser wird über die Mittelanzapfung gemessen. Wenn das SM verfahren wird, erfährt der konstante Strom (65mA) eine Zunahme, die periodisch nach demselben Muster verläuft.

Aufgrund der magnetischen Flussänderung des Rotors kommt es zu Ungleichmäßigkeiten im Stromverlauf. Bei Betrieb außerhalb des normalen Arbeitsbereichs bewirkt der außer Tritt fallende Rotor starke Stromanregung als auch Stromverdrängung. Die Art und Weise des außer Tritt fallens variiert unter verschiedenen SMs, sowie auch bei jedem einzelnen SM stark, da es an unterschiedlichen Rotorpositionen auftritt.

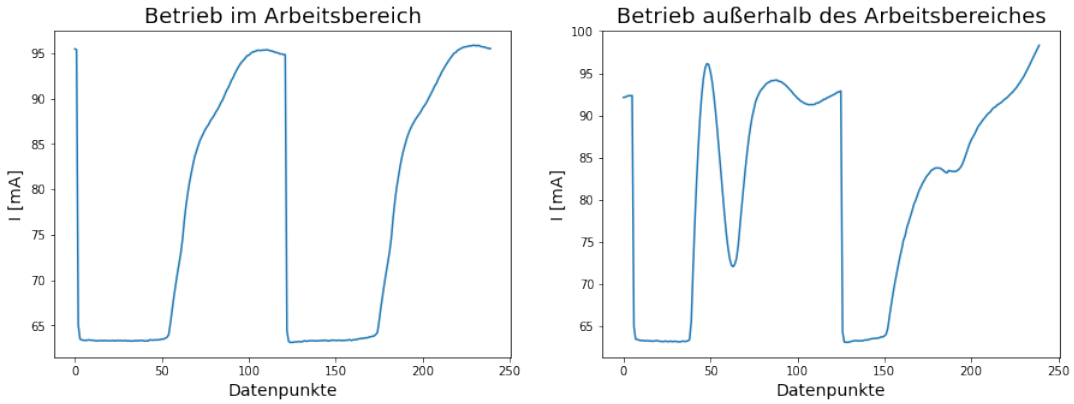

Abbildung 6.1: Vergleich der Ströme bei Betrieb im und außerhalb des Arbeitsbereiches

6.2 Verwendete Datenbasis

Um den mechanischen Anschlag zu detektieren, werden Daten aus dem in Kapitel 5 beschriebenen Gesamtdatensatz ausgewählt. Auf Grund der großen Datenmenge fand eine detaillierte Voruntersuchung der Zeitreihendaten der unterschiedlichen Signale statt. Hierbei hat sich herausgestellt, dass Signal 1 am besten geeignet ist, um den Anschlag mit Klassifikationsalgorithmen zu bestimmen.

Der ausgewählte Datensatz zur Anschlagserkennung D_{an} besteht aus 280.000 Samples. Den verwendeten Samples liegen dabei zu gleichen Teilen die unterschiedlichen klimatischen Bedingungen, welche in Kap. 5.3.3 beschrieben sind zu Grunde. Darüber hinaus wurden Daten aus allen Betriebsmodi in die Datenbasis einbezogen. Die Betriebsmodi, auf die näher in Kapitel 2.1 eingegangen wurde, sind:

- Fahrt hoch (FH): Der SM dreht sich im Uhrzeigersinn, was diesen nach oben fahren lässt
- Fahrt runter (FR): Der SM dreht sich gegen den Uhrzeigersinn, was diesen nach unten fahren lässt
- Oberer Anschlag (OA): Der SM wird über seinen oberen mechanischen Anschlag hinaus im Uhrzeigersinn betrieben
- Unterer Anschlag (UA): Der SM wird über seinen unteren mechanischen Anschlag hinaus gegen den Uhrzeigersinn betrieben

Für die Modi FH und FR sind jeweils ca. 115.000 Samples enthalten, für die Modi OA und UA jeweils ca. 25.000 Samples. Allen Samples wurde die Information über ihren zu Grunde liegenden Betriebsmodus hinzugefügt.

6.2.1 Erstellung von Rekurrenzplots

Auf Grund der in Kapitel 2.5 beschriebenen Eigenschaften von Rekurrenzplots wird angenommen, dass diese ein probates Werkzeug zur Erkennung des mechanischen Anschlags eines SM sein können. Da die Stromkurve von Schrittmotoren im normalen Betrieb (sowohl im als auch gegen den Uhrzeigersinn) eine hohe Periodizität aufweist, die Stromkurve jedoch bei Betrieb des SMs außerhalb des normalen Arbeitsbereichs chaotisch wird, ist davon auszugehen, dass mit Rekurrenzplots als Eingabe für ein neuronales Netz gute Ergebnisse für eine Klassifikation zwischen diesen Zuständen erzielt werden können. In Abbildung 6.2 ist beispielhaft sowohl die normale Zeitreihe, als auch der dazugehörige Rekurrenzplot für alle vier Betriebsmodi, dargestellt.

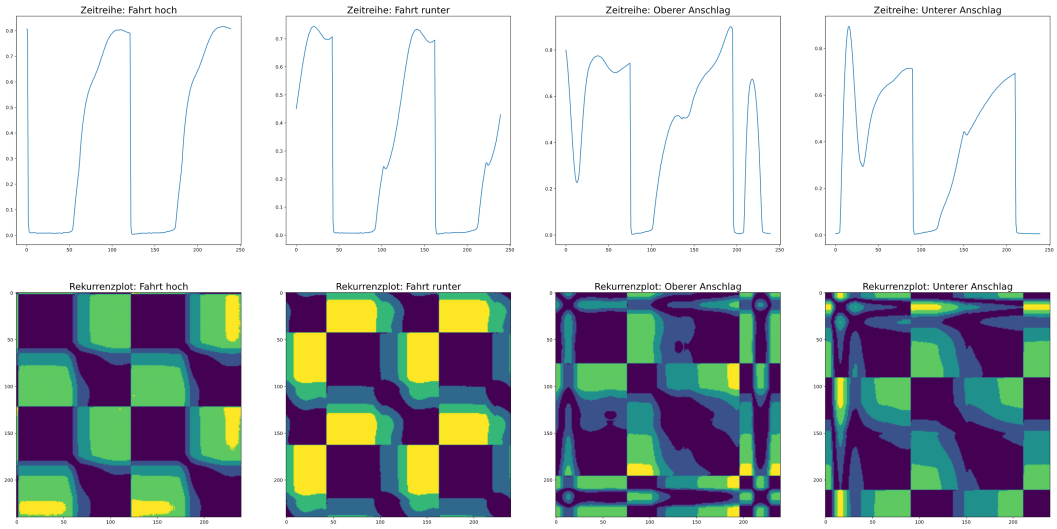

Abbildung 6.2: Zeitreihen (oben) und dazugehörige Rekurrenzplots (unten) für alle vier Betriebsmodi der SMs.

Um die aufgestellte These der Eignung von Rekurrenzplots zu validieren, werden alle Zeitreihen aus Datensatz D_{an} nach dem in Kapitel 4.2 beschriebenen Verfahren in Rekurrenzplots umgewandelt. Zur Erstellung der Rekurrenzplots wurden hierbei alle Parametereinstellungen genutzt, die auch in den Experimenten in Kapitel 4 genutzt worden sind. Diese setzen sich aus den Schrittweiten 1, 10, 20 und 50 kombiniert mit den ϵ-Werten von 0.01, 0.1 und 0.2 zusammen. Es entstehen somit insgesamt 12 weitere Datensätze D_{PS1}, D_{PS2}, ... , D_{PS12} für Experimente mit Rekurrenzplots. Jeder Datensatz repräsentiert hierbei eine Parameterselektion.

Tabelle 6.1: Ergebnisse der Evaluation für die Detektion des mechanischen Anschlags mit Rekurrenzplots.

	Steps			1	1	1	10	10	10	20	20	20	50	50	50
	Epsilon			0.01	0.1	0.2	0.01	0.1	0.2	0.01	0.1	0.2	0.01	0.1	0.2
Test	Metrik	SVM	KNN	PS1	PS2	PS3	PS4	PS5	PS6	PS7	PS8	PS9	PS10	PS11	PS12
1	Acc	0.9207	0.9951	0.9941	0.9981	0.9950	0.9953	0.9977	0.9975	0.9963	0.9982	0.9988	0.9943	0.9989	0.9975
	F1	0.9101	0.9911	0.9938	0.9980	0.9944	0.9947	0.9971	0.9975	0.9957	0.9981	0.9984	0.9939	0.9983	0.9975
2	Acc	0.9231	0.9939	0.9963	0.9961	0.9954	0.9965	0.9993	0.9996	0.9949	0.9975	0.9992	0.9945	0.9988	0.9944
	F1	0.9172	0.9929	0.9963	0.9961	0.9954	0.9961	0.9989	0.9995	0.9945	0.9974	0.9987	0.9942	0.9983	0.9943
3	Acc	0.9217	0.9952	0.9942	0.9981	0.9987	0.9967	0.9985	0.9994	0.9951	0.9986	0.9996	0.9963	0.9993	0.9956
	F1	0.9121	0.9951	0.9939	0.9981	0.9987	0.9965	0.9979	0.9989	0.9950	0.9985	0.9993	0.9959	0.9990	0.9954
4	Acc	0.9211	0.9941	0.9950	0.9964	0.9960	0.9972	0.9976	0.9973	0.9961	0.9996	0.9982	0.9981	0.9976	0.9981
	F1	0.9116	0.9963	0.9944	0.9959	0.9957	0.9969	0.9976	0.9970	0.9957	0.9993	0.9976	0.9979	0.9970	0.9977
5	Acc	0.9223	0.9929	0.9963	0.9985	0.9989	0.9962	0.9974	0.9979	0.9964	0.9972	0.9988	0.9957	0.9976	0.9986
	F1	0.9152	0.9924	0.9959	0.9982	0.9987	0.9960	0.9968	0.9974	0.9964	0.9970	0.9982	0.9955	0.9972	0.9981
Mean	Acc	0.9218	0.9942	0.9952	0.9974	0.9968	0.9964	0.9981	0.9983	0.9958	0.9982	0.9989	0.9958	0.9984	0.9968
	F1	0.9132	0.9936	0.9949	0.9973	0.9966	0.9960	0.9977	0.9981	0.9955	0.9981	0.9984	0.9955	0.9980	0.9966

6.3 Evaluierung

Für alle Experimente der Evaluation wurde die Erkennung des Anschlags als zwei Klassen Problem implementiert. Alle Samples von FH und FR werden der Klasse 1 = *normaler Arbeitsbereich* zugeordnet und alle Samples die im Bereich OA und UA aufgenommen wurden, werden der Klasse 2 = *mechanischer Anschlag* zugeordnet.

Anschließend wird zunächst D_{an} gemischt und in einem 75:25 Split in Trainings- und Testdatenmenge unterteilt. Nachfolgend wird mit den bereits aus Teil 1 bekannten Algorithmen, kNN und SVM, die Klassifikation durchgeführt.

Die Datensätze D_{PS1}, D_{PS2}, ... , D_{PS12} werden ebenfalls vor jedem Experiment gemischt und anschließend in einem 60:20:20 Verhältnis in Trainings-, Test- und Validierungsdaten unterteilt. Diese dienen als Eingabedaten für das bereits in Kapitel 4.2 genutzte CNN. Tabelle 6.1 zeigt die Bewertungsergebnisse aller Klassifikatoren für fünf Durchläufe unter Verwendung von Acc und F1-Score als Metriken.

Es ist zu erkennen, dass in allen Experimenten sehr gute Ergebnisse erzielt werden konnten. Mit Hilfe des kNN-Klassifizierer konnten im Mittel 99.42 % aller Testdaten von Datensatz D_{an} der richtigen Klasse zugeordnet werden. Die SVM fällt im Vergleich zum kNN ab. Hier konnten nur 92.2 % aller Testdaten richtig zugeordnet werden. Es ist festzuhalten, dass der F1-Score diese Ergebnisse bestätigt und auf eine gute Generalisierung beider Klassen schließen lässt. Darüber hinaus ist die Standartabweichung der Ergebnisse über die 5 Durchläufe sowohl für SVM als auch für kNN als vernachlässigbar anzusehen, da die höchste Abweichung in Acc bei 0.13 % und für den F1-Score bei 0.39 % liegt.

Mit Hilfe des CNNs konnten für alle getesteten Datensätze, denen jeweils Rekurrenzplots erzeugt mit unterschiedlichen Parametereinstellungen zu Grunde liegen, sehr gute Ergebnisse erzielt werden. Nach Betrachtung der Resultate ist festzustellen, dass Parametereinstellung $PS1$ im Mittel mit einer Acc von 99.52 % sowie einem F1-Score von 99.49 die schlechtesten Ergebnisse aller Experimente liefert, die mit Rekurrenzplots durchgeführt wurden. Diese Ergebnisse sind sowohl in Acc als auch in F1-Score bereits im Mittel besser, als die Ergebnisse die mit kNN und SVM erreicht werden konnten.

Nach Analyse der Ergebnisse der unterschiedlichen Parametereinstellungen von Rekurrenzplots lässt sich feststellen, dass mit $\epsilon = 0.01$ für alle Schrittweiten das jeweils schlechteste Durchschnittsergebnis erreicht worden ist. Dieser klare Trend konnte in Kapitel 4 nicht festgestellt werden. Eine mögliche Begründung liegt in der Beschaffenheit der genutzten Datenbasis und ist somit nicht als repräsentativ zu betrachten, ist jedoch für weitere Experimente, mit der veröffentlichten Datenbasis zu beachten.

Weiterhin lässt sich feststellen, dass die besten Ergebnisse (Acc = 99.89 %; F1-Score = 99.84 %) mit der Parametereinstellung $\epsilon = 0.2$ mit einer Schrittweite von 20 erreicht werden konnten. Auch die Kombination $\epsilon = 0.1$ mit einer Schrittweite von 20 konnten sehr gute Ergebnisse (Acc = 99.82 %, F1-Score = 99.81 %) erzielt werden. Über alle Tests mit Rekurrenzplots ist darüber hinaus festzustellen, dass diese in allen Parametereinstellungen äußerst robust funktionieren, da keine großen Abweichungen in Acc und F1-Score in den durchgeführten Testdurchläufen aufgetreten sind.

Data Augmentation

Auf Grund der guten Ergebnisse mit den vorhandenen Datensets wird eine Anreicherung der Datensets als nicht sinnvoll eingestuft und somit von einer solchen abgesehen. Eine Anreicherung der Daten mit den in Kapitel 3 vorgestellten Methoden wäre denkbar gewesen, wenn die Evaluation entweder ergeben hätte, dass nicht ausreichend Trainings- oder Testdaten vorliegen oder wenn davon auszugehen wäre, dass das Klassenungleichgewicht, welches durch die unterschiedliche Anzahl an Daten für jeden Modus resultiert, Einfluss auf die Performance der genutzten Algorithmen hat. Da beide Faktoren auf Grund der Ergebnisse ausgeschlossen werden können, ist im Rahmen dieser Arbeit keine Evaluation für die Erkennung des mechanischen Anschlags auf erweiterten Datensets erfolgt.

6.4 Diskussion

Die Evaluation hat gezeigt, dass durch die Daten, die durch den Dauerversuch gewonnen wurden, eine äußerst funktionale Detektion des mechanischen Anschlags umgesetzt werden kann. Sowohl mit herkömmlichen Algorithmen wie SVM und kNN, als auch mit Hilfe von Rekurrenzplots, welche mit einem CNN ausgewertet werden, ist eine zuverlässige Klassifikation möglich. Vor allem mit Hilfe des in Kapitel 4 vorgestellten Verfahrens zur Analyse von Zeitreihen mit Hilfe von Rekurrenzplots konnten außerordentlich vielversprechende Ergebnisse erzielt werden. Hervorzuheben ist hierbei Datensatz D_{PS10}, dem Rekurrenzplots der Schrittweite 20 sowie ϵ von 0.2 zu Grunde liegen, mit dem 99.89 % aller Testdaten richtig zugeordnet werden konnten.

Dies bestätigt die in Kapitel 6.1 getätigte These, dass Rekurrenzplots ein probates Mittel zur Analyse der Schrittmotordaten darstellen können. Die Periodizität, welche in den Daten des normalen Fahrbetriebs zu finden ist, kann klar von dem chaotischen Verhalten unterschieden werden, welches auftritt, wenn der Schrittmotor außerhalb seines Arbeitsbereiches betrieben wird.

Für die zu Grunde liegende Aufgabenstellung lässt sich festhalten, dass eine Anschlagserkennung für den in Kapitel 5 untersuchten Schrittmotor zuverlässig und robust entworfen werden konnte. Dies lässt sich insbesondere auf die hohe Datenqualität zurückführen. Mit Hilfe der entworfenen Anschlagserkennung, können Referenzfahrten nun erheblich verkürzt werden. Es ist nicht mehr erforderlich den Motor viele Schritte über seinen Arbeitsbereich hinaus zu betreiben. Die Belastung, die durch das Betreiben im anormalen Bereich entsteht, kann so erheblich reduziert werden.

Hinzugefügt werden muss, vor allem mit Hinblick auf eine industrielle Anwendung, dass die Analyse mittels Rekurrenzplots wesentlich rechenintensiver ist, als die Nutzung der nur leicht bearbeiteten Rohdaten. Wird die Anwendung im Real-Betrieb implementiert, muss die Auswertung der Zeitreihen in Echtzeit erfolgen. Für die Anwendung mit Rekurrenzplots müssen so nach der Normierung der Zeitreihen erst Rekurrenzplots erstellt werden, ehe diese mit Hilfe einen CNNs ausgewertet werden könnten. Die entstandenen Rekurrenzplots sind hierbei auf Grund ihrer Matrix-Struktur sowie der Farbcodierung größer als die herkömmliche Zeitreihe. Das heißt, es fallen sowohl für die Erstellung der Rekurrenzplots, als auch für deren Auswertung erhöhte Rechenkosten an. Ist eine Anschlagserkennung auf Systemen zu implementieren, denen eine sehr eingeschränkte Rechenleistung oder nur wenig Speicherplatz zur Verfügung stehen, ist daher die Anwendung des kNN zu empfehlen.

Kapitel 7

Praktische Umsetzung eines Systems zur frühzeitigen Erkennung von Fehlern bei Schrittmotoren

In diesem Kapitel wird ein System zur frühzeitigen Erkennung von Fehlern bei SMs entworfen. Hierzu wird zunächst im Kapitel 7.1 die Problemstellung beschrieben, ehe in Kapitel 7.2 vorangegangene Arbeiten beleuchtet werden. Anschließend erfolgt die Gestaltung der Datenbasis in Kapitel 7.3, die Erläuterung der Methodik in Kapitel 7.4 sowie die Evaluation in Kapitel 7.5. In Kapitel 7.6 werden die erzielten Ergebnisse abschließend diskutiert.

7.1 Problemstellung

SMs erfüllen in verschiedenen technischen Systemen den kritischen Part des Positionsgebers oder fungieren als Regelelement. Liegt ein unerwarter Fehlerfall eines SMs vor, führt dies meist zu ungewünschten und teuren Stillstandszeiten des kompletten Systems.

RUL prediction ist eines der interessantesten Forschungsfelder der künstlichen Intelligenz und beschäftigt sich damit, Fehlerfälle, die in der Zukunft auftreten, zu prognostizieren. Durch richtig prognostizierte Fehlerfälle lassen sich die kritischen Komponenten präventiv austauschen und so der Fehlerfall vermeiden.

Transformer sind eine neuartige Architektur neuronaler Netze, die in den letzten Jahren vor allem im Bereich NLP gute Ergebnisse vorweisen konnte. Durch ihre einzigartige Fähigkeit, gewissen Datenstrukturen eine erhöhte Attention zuzuweisen, sowie zeitliche Abhängigkeit in Daten zu berücksichtigen, sind Transformer auch eine äußerst interssante Technologie für RUL prediction.

Mit Hilfe der in Kapitel 5 gewonnen Datenbasis soll daher im Folgenden eine

Fehlererkennung für SMs mit Hilfe von Transformern entworfen werden. Hierzu ist es notwendig, Stromverläufe die in einem definierten Zeitrahmen vor Fehlereintritt aufgezeichnet wurden von Stromverläufen zu unterscheiden, auf die kein Fehlerfall folgt.

Wird eine Zeitreihe erkannt, auf die ein Fehlerfall folgt, kann eine Warnung ausgegeben werden. Diese ermöglicht es dem Betreiber des SMs zu reagieren und den Austausch oder die Reperatur vorzunehmen. Mit einer funktionalen Ausfallerkennung können Stillstandszeiten gesamter technischer Systeme vermieden oder reduziert werden. Um eine praktische Anwendung zu ermöglichen, ist es erforderlich, die Information über das Auftreten eines Fehlers in einem geeigneten Zeitrahmen vor dem erwarteten Auftreten zu prognostizieren. Hierdurch soll sichergestellt werden, dass ausreichend Zeit für einen Austausch der Komponente zur Verfügung steht.

7.2 Vorangegangene Arbeiten

In Kapitel 2.3.4 wurde das originale Transformer-Modell von Vaswani et. Al [53] im Detail vorgestellt. Zur Verbreitung von Transformern war ein wichtiger Schritt, die Berechnung zu parallelisieren und auf GPUs zu ermöglichen, da Transformer eine sehr hohe Rechenkapazität benötigen. Anschließend wurden verschiedene Arbeiten veröffentlicht, die Transformer zur Vorhersage von Zeitreihen einsetzen [113], [114], [115], [116]. Auch die Kombination eines Transformer-Modells mit anderen Modellen wurde untersucht [117]. Seit 2020 entstanden einige Arbeiten, die Transformer zur RUL prediction von unterschiedlichen Anwendungsfällen einsetzen. In [118] führen die Autoren Experimente mit dem C-MAPSS-Datensatz [119] durch und zeigen, dass die Leistung von Transformern vergleichbar oder überlegen im Vergleich zu bestehenden Ansätzen ist. Die Autoren von [120] können mit ihrem Transformer-Modell Ausfälle von Triebwerken sehr gut prognostizieren und erzielen bessere Ergebnisse als die meisten herkömmlichen Methoden. In [121] nutzen die Autoren das Prinzip der self-attention um die Lebensdauer von Lithium-Ionen Batterien abzuschätzen und können ebenfalls gute Ergebnisse vorweisen.

7.3 Gestaltung der Datenbasis

In diesem Kapitel wird ein genauer Einblick in die zur Ausfallvorhersage verwendete Datenbasis gegeben. Hierzu wird zunächst ein Faktor ermittelt, der einen Zusammenhang zwischen Betriebszeit im Dauerversuch und realer Betriebszeit herstellt. Anschließend wird ein RUL Schwellwert definiert, dem entnommen werden kann, ab wann eine Messreihe als kritisch eingestuft wird und somit eine Warnung ausgegeben werden kann. Abschließend erfolgt die Klassenzuordnung sowie

ein Überblick über die verwendete Datenbasis.

7.3.1 RUL Faktor

Da die Betriebszeit der SM im Dauerversuchsaufbau auf Grund der erhöhten Belastung nicht mit der Betriebszeit in realen Anwendungen gleichgesetzt werden kann, wird im Folgenden versucht, einen Umrechnungsfaktor F_{RUL} von der Betriebszeit im Dauerversuch T_D in reale Betriebszeit T_R zu ermitteln. Es gilt die Annahme

$$T_R >> T_D. \tag{7.1}$$

Diese Annahme stützt sich auf folgende Gegebenheiten:

- Die im Dauerversuchsaufbau verbauten SMs werden durchgehend verfahren und legen daher wesentlich mehr Schritte zurück, als es bei normalen Anwendungen der Fall ist.
- Das Fahrprofil des Dauerversuchsaufbaus steuert die SMs häufig in ihre mechanischen Anschläge, was nicht dem normalen Arbeitsbereich der SMs entspricht.
- Die klimatischen Bedingungen, denen die SMs durch den Betrieb in der klimatisierten Einheit ausgesetzt sind, erzeugen eine hohe Belastung der Komponenten. Auf Grund der häufigen Wechsel zwischen den verschiedenen klimatischen Bedingungen ist davon auszugehen, dass diese die SMs mehr belasten, als es bei normalen Anwendungen der Fall ist.
- Die Abgeschlossenheit der klimatisierten Einheit kann zu einer geringen Belastung durch Staub oder anderen Verunreinigungen führen als im Realbetrieb.

Um einen geeigneten Umrechnungsfaktor F_{RUL} zwischen T_D und T_R zu ermitteln, wurden zunächst zahlreiche Gespräche mit Fachexperten geführt. Den Experten wurde Einblick in das Setup des Dauerversuchs gegeben und ihnen wurden die Fehlerbilder der SMs zur Verfügung gestellt. Zusätzlich zu diesen Fachgesprächen wurden Erfahrungswerte aus der Industrie hinsichtlich Austauschintervallen von SMs recherchiert. Hierbei konnte ermittelt werden, dass in der Industrie davon ausgegangen wird, dass SMs nach 15 Jahren ausgetauscht werden müssen [122]. In dieser Zeitspanne wird von einer totalen Schrittanzahl von ca. 980 Millionen Schritten ausgegangen [123]. Die oben aufgeführten Gegebenheiten konnten nach

mündlicher Kommunikation mit dem Industriepartner bestätigt, jedoch nur unzureichend quantifiziert werden.

Aus den eingeholten Informationen wurden drei Möglichkeiten zur Berechnung von F_{RUL} extrahiert:

1. Die im Dauerversuchsaufbau zurückgelegten Schritte ins Verhältnis zu den in der Industrie bekannten Schrittanzahl von ca. 980 Millionen Schritten setzen.

2. Die unterschiedlichen zusätlichen Belastungen durch klimatische Bedingungen, Dauerbetrieb und Fahrprofil grob quantifizieren.

3. Das bekannte Austauschintervall von 15 Jahren ins Verhältnis mit der ersten Häufung von Fehlerfällen im Dauerversuchsaufbau setzen.

Nach Evaluation der unterschiedlichen Möglichkeiten, wurde Möglichkeit 3 für die valideste befunden. Dies liegt in der Charakteristik des zeitlichen Verlaufs der im Dauerversuchsaufbau aufgetretenen Fehlerfälle, sowie der unzureichenden Quantifizierbarkeit der Einflussfaktoren im Dauerversuchsaufbau, begründet. In Abbildung 7.1 ist die Häufigkeit von Fehlerfällen an Hand der Betriebszeit in Stunden dargestellt.

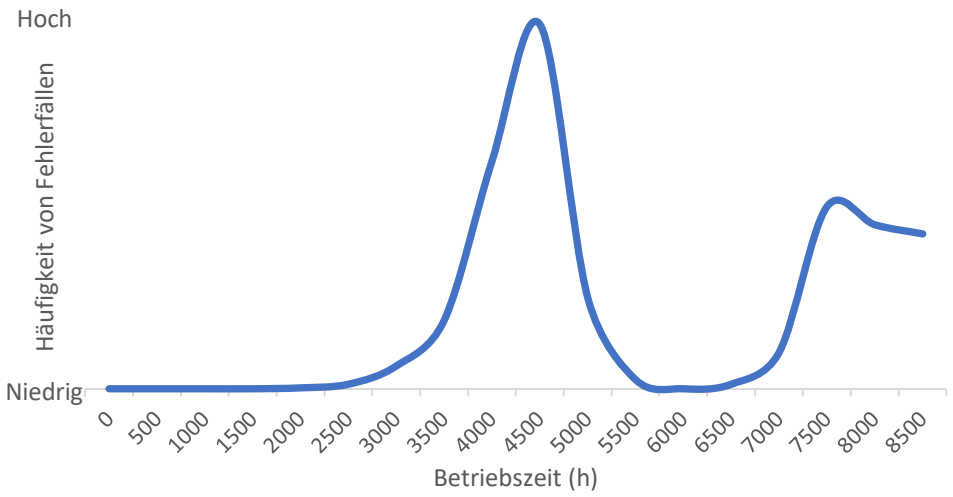

Abbildung 7.1: Ausfallhäufigkeit der SM in Abhängigkeit der Betriebszeit.

Es ist deutlich zu erkennen, dass die Struktur der Badewannenkurve folgt. Diese beschreibt oftmals die Gerätelebensdauer von technischen Geräten oder Systemen und stellt die Ausfallwahrscheinlichkeit in Abhängigkeit der Betriebszeit dar [124]. Die Begründung des Effektes der Badewannenkurve liegt darin, dass Mängel, die durch verbaute Werkstoffe, Produktion- oder Konstruktionsfehler anfallen, häufig innerhalb einer gewissen Betriebszeit auftreten. Liegen solche Mängel nicht vor,

funktioniert das Gerät einwandfrei, bis Schäden durch Verschleiß und Alterungseffekte eintreten.

Nach Rücksprache mit Fachexperten und Analyse der ausgefallenen SMs (vgl. Kapitel 5.4) wird angenommen, dass dieses Verhalten auch bei dem für diese Arbeit durchgeführten Experiment vorliegt.

Es wird daher für die folgenden Experimente zur Bestimmung der RUL die Annahme getroffen, dass die erste Fehlerwelle, welche bei ca. 4000 Betriebsstunden einsetzt, der in der Industrie bekannten Lebensdauer von $T_R = 15$ Jahre $= 131.400$ Stunden entspricht. Mit F_{RUL} folgt daher

$$F_{RUL} \cdot T_D = T_R \tag{7.2}$$

$$\Leftrightarrow \quad F_{RUL} = \frac{T_R}{T_D}$$

$$\Leftrightarrow \quad F_{RUL} = \frac{131.400}{4000}$$

$$\Rightarrow \quad F_{RUL} = 32,85. \tag{7.3}$$

7.3.2 RUL Schwellwert

Nachdem nun F_{RUL} ermittelt worden ist und erlaubt, die Betriebszeit im Dauerversuchsaufbau in reale Betriebszeit umzurechnen, muss nun noch definiert werden, welche Vorlaufzeit vor einem möglichen Defekt gewählt wird, um eine Warnung auszugeben und auf einen prognostizierten Fehler hinzuweisen. Diese Vorlaufzeit wird im folgenden als $F_{Schwell}$ bezeichnet. Um $F_{Schwell}$ sowohl für die praktische Anwendung der Ausfallerkennung, als auch für die technische Umsetzbarkeit valide zu wählen, müssen verschiedene Faktoren berücksichtigt werden.

Wird $F_{Schwell}$ sehr groß gewählt, führt dies zwar zu einer großen Datenbasis, welche vorteilhaft für den Trainingsprozess des Modells ist, bietet jedoch keine sinnvolle Option für reale Applikationen. Wird beispielsweise klassifiziert, dass innerhalb der nächsten 250 Tage ein Fehler im überwachten Gerät auftritt, ist dies für die meisten Anwendungen - wie z.B Gasthermen oder Drucker - wenig hilfreich. Auch sehr kleine Bereiche, beispielsweise von wenigen Minuten, verfehlen den Zweck einer Fehlervorhersage, da keine oder nur unzureichende Zeit für eine entsprechende Reaktion bleibt. Weiterhin führt ein sehr klein gewähltes $F_{Schwell}$ zu einer stark reduzierten Datenbasis, was den Trainingsprozess von Algorithmen stark beeinträchtigen kann.

Der Anspruch dieser Arbeit ist es, nah an realen Applikationen zu arbeiten und einen guten Trainingsprozess des Modells zu ermöglichen. Es ist daher erforderlich, $F_{Schwell}$ so zu wählen, dass sowohl bei Ausgabe einer Warnung ein praktikabler Zeitraum zur Reaktion gewährt wird, als auch ausreichend Daten zum Training des Modells zur Verfügung stehen.

Um diesen Anforderungen gerecht zu werden, wird nach Rücksprache mit dem Industriepartner

$$F_{Schwell} = 30 \text{ Tage} \tag{7.4}$$

für die nachfolgenden Experimente definiert. Das heißt, dass 30 Tage vor Auftreten eines Defekts, vor diesem gewarnt werden können soll. Dies gibt in Realität genug Zeit für eine Reaktion, beispielsweise in Form von Austausch des SMs. Eine größeres $F_{Schwell}$ wird als nicht sinnvoll erachtet.

7.3.3 Datenauswahl und Klassenzuordnung

Das Ziel dieser Arbeit ist es, vorherzusagen, ob ein Schrittmotor innerhalb einer vordefinierten Zeitspanne einen Defekt erlebt oder nicht. Diese Zeitspanne wurde auf $F_{Schwell} = 30$ Tage definiert. Die Aufgabe stellt ein binäres Klassifikationsproblem dar, in der die Klassen als

$$y = \begin{cases} 1, & \text{Defekt innerhalb } F_{Schwell} \\ 0, & \text{Kein Defekt innerhalb } F_{Schwell} \end{cases}, \tag{7.5}$$

definiert sind. Wird ein Sample Klasse 1 zugeordnet, kann dem Betreiber des SMs eine Warnung ausgegeben werden. An Hand dieser Warnung können Maßnahmen zur Verhinderung des Ausfalls unternommen werden.

Nun wird die in Kapitel 5.5 vorgestellte Datenbasis $\mathbf{X} \in \mathbb{R}^{28\text{M}\times 240\times 1}$ ausgewählt und jeder Zeitreihe eine Klasse aus y zugewiesen. Alle Datenproben, für deren Ursprungsmessreihe gilt, dass diese innerhalb von

$$\frac{F_{Schwell}}{F_{RUL}} = 0{,}913 \text{ Tage} \tag{7.6}$$

vor dem Auftreten eines Defektes aufgezeichnet wurden, werden der Klasse $y = 1$ zugeordnet. Alle anderen Datenproben werden der Klasse $y = 0$ zugeordnet. Das Schema der Klassenzuordnung ist in Abbildung 7.2 veranschaulicht. Da wesentlich mehr Samples mit Klasse $y = 0$ als mit Klasse $y = 1$ vorliegen, wird für die Datenbasis, die zur Vorhersage von Fehlerfällen genutzt wird, $\mathbf{X}_{RUL}$ ein Klassenausgleich vorgenommen. Für den vorliegenden Anwendungsfall bedeutet dies, dass alle Samples von der kleineren Klasse $y = 1$ in $\mathbf{X}_{RUL}$ belassen werden, und von der größeren Klasse $y = 0$ zufällig soviele Samples entfernt werden, dass beide Klassen in $\mathbf{X}_{RUL}$ gleichmäßig vertreten sind.

Dies führt zu einer Datenbasis $X_{RUL} \in \mathbb{R}^{100\text{k}\times 240\times 1}$ mit 50000 Samples der Klasse 0 und 50000 Samples der Klasse 1.

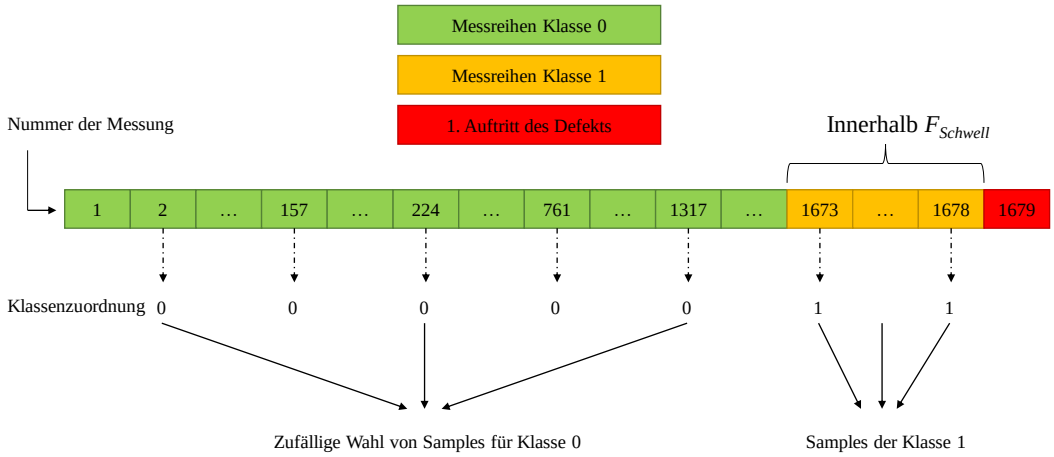

Abbildung 7.2: Schema der Klassenzuordnung

Data Augmentation

Wie in Kapitel 2.4 erläutert, wird DA häufig bei Datensätzen zur RUL Vorhersage angewandt. Dies begründet sich in der häufig auftretenden ungleichen Verteilung zwischen fehlerbehafteten Daten und Daten aus dem normalen Betrieb.

Auch die in Kapitel 5 generierte Datenbasis $\mathbf{X}_{RUL}$ unterliegt diesem Problem. Insgesamt liegen in der Ausgangsdatenbasis $\mathbf{X}$ 28 Millionen Samples vor, davon fallen unter Voraussetzung von $F_{Schwell} = 30$ Tage jedoch nur 50.000 in die Kategorie fehlerbehaftet. Da eine Datenbasis mit ausgeglichener Klassengröße von Vorteil ist, ist $\mathbf{X}_{RUL}$ wesentlich kleiner als $\mathbf{X}$.

Es liegt daher nahe, künstliche Samples aus den bestehenden Daten von Klasse 1 zu erzeugen, um $\mathbf{X}_{RUL}$ synthetisch anzureichern und dem Algorithmus somit mehr Trainingsdaten zur Verfügung zu stellen.

Als Verfahren hierfür wurde die in Kapitel 3.4 vorgestellte Methode mit VAEs ausgewählt. Da für die vorgestellten Architekturen kein klarer Trend erkennbar war, werden alle drei Architekturen zur Erzeugung synthetischer Daten genutzt. Als Multiplikatorzahl wird $n = 10$ ausgewählt, da hiermit in Kapitel 3.4 die besten Ergebnisse erzielt werden konnten. Es entstehen somit die drei angereicherten Datensätze $\mathbf{X}_{RUL-aug1}, \mathbf{X}_{RUL-aug2}$ und $\mathbf{X}_{RUL-aug3}$. Mit den VAEs werden lediglich Daten für Klasse 1 erzeugt. Weitere Daten für Klasse 0, die zum Ausgleich der Klassengröße benötigt werden, werden $\mathbf{X}$ zufällig entnommen. Die angereicherten Datensätze verfügen je über insgesamt 500.000 Samples pro Klasse.

7.4 Methodik

Nachdem nun die verschiedenen Datensätze für eine Ausfallvorhersage aufbereitet worden sind, wird nun die Methodik und das zu Grunde liegende Modell näher beschrieben. In Kapitel 2.2 wurden unterschiedliche Arbeiten vorgestellt, die verschiedene Methoden zur RUL prediciton nutzen. Darüber hinaus wurden in Kapitel 2.3.4 Transformer als neuartige Methode vorgestellt, die bisher hauptsächlich im NLP Bereich eingesetzt wurde, aber auch auf Zeitreihen angewandt werden kann. Da Transformer bisher nur in wenigen Arbeiten für RUL prediction eingesetzt wurden, wird dieses Anwendungsfeld in dieser Arbeit weiter vertieft.

Um das in Kapitel 2.3.4 vorgestelle Modell des Transformers für Zeitreihen anzupassen, werden nur die Komponenten des Encoders benötigt. Der Decoder wird im NLP zur Hinzugabe des Zielsatzes genutzt. Diese Funktionalität wird im Bereich der Zeitreihen-Klassifikation nicht benötigt.

Der für diese Arbeit entworfene Transformer besteht daher nur aus dem Encoder sowie aus Dense-Schichten zur Klassifikation. Die Struktur des Modells ist in Abbildung 7.3 dargestellt.

Die wichtigstens Komponenten des in dieser Arbeit genutzten Transformers werden im Folgenden kurz vorgestellt.

7.4.1 Embedding

Die Inputs für den Transformer sind univariate Zeitreihen $\mathbf{x} = (x_1, \ldots, x_{seq\_len})$ mit *seq_len* Datenpunkten und $x_i \in [0, 1]$ für alle $i \in \{1, \ldots, seq\_len\}$.

Für NLP-Anwendungen wird eine spezielle Embedding-Schicht verwendet, die jedoch nur mit positiven Indizes als Eingaben und nicht mit kontinuierlichen Daten genutzt werden kann. Die Einbettung von Zeitreihendaten ist eine andere Herausforderung, denn bei realen Zeitreihendaten gibt es häufig Variationen und Verschiebungen in den Daten auf Grund von Sensorungenauigkeiten und variierenden Umgebungsbedingungen.

Im vorliegenden Fall treten Verschiebungen in der Zeit- und Wertedimension bereits durch die wiederkehrenden klimatischen Veränderungen ein. Wenn nur ein Wert in zwei Zeitreihen der gleichen Klasse geringfügig abweicht, führt dies zu einer starken Abweichung des Embeddings. Dennoch sind die meisten Verschiebungen im vorliegenden Fall wiederkehrend, also vorhersehbar, so dass ein Embedding gesucht wird, welches dies berücksichtigen kann. Diese Überlegungen schließen jeden Ansatz mit festen Bezugsgrößen aus und erfordern eine dynamische Herangehensweise.

Es wird daher vorgeschlagen, ein hochdimensionales Embedding zu verwenden um ausreichend Freiheitsgrade zur Verfügung zu stellen. Von daher wird eine

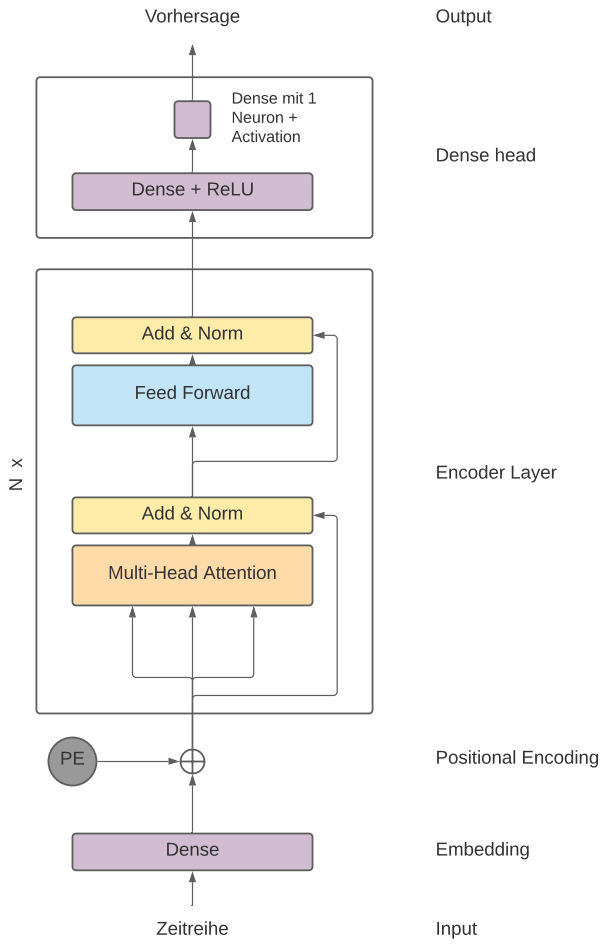

Abbildung 7.3: Struktur des Transformers zur RUL Vorhersage von Schrittmotoren

lineare Schicht mit der Anzahl Neuronen $units = d\_model \in \mathbb{N}$ gewählt. Dadurch werden die Eingangsvektoren der Form $(batch\_size, seq\_len, 1)$ transformiert zu $(batch\_size, seq\_len, d\_model)$.

Dies gibt dem Modell die Chance, sein eigenes Embedding zu lernen, welches mit der Herausforderung von variierenden Werten umgehen kann. Es wird somit dem Modell die Möglichkeit gegebenen, selbst die Komplexität an die vorliegende Aufgabenstellung anzupassen. Ein ähnlicher Ansatz für das Embedding wurde auch in [125] gezeigt.

Die Einstellung in dieser Arbeit ist $d_{model} = 8$, da diese Parameterwahl sich in zahlreichen, für diese Arbeit durchgeführten Voruntersuchungen, bei Anwendung auf ähnlichen Datensätzen, als sehr funktional erwiesen hat.

7.4.2 Positional Encoding

Für den Anwendungsfall dieser Arbeit hat das PE eine große Bedeutung, da die zeitliche Abhängigkeit der genutzten Datenbasis beibehalten werden muss. Aus dem Embedding folgt der Input $\mathbf{X} \in \mathbb{R}^{seq\_len \times d\_model}$ für das PE. In Kapitel 2.3.4.3 wurden mehrere Möglichkeiten vorgestellt, mit denen PE vorgenommen werden kann. Für die vorliegende Datenbasis, wäre sowohl das PE aus Gleichung 2.10, als auch das PE aus Gleichung 2.11 anwendbar. Um für zukünftige Anwendungsfälle die Verwendung von Zeitreihen unterschiedlicher Länge zu ermöglichen, wurde das PE aus Gleichung 2.11 gewählt. Der Output des PE ist demnach definiert als $\mathbf{X}_{PE} := \mathbf{X} + \mathbf{PE}$.

7.4.3 Dense Head

Gegeben ist der Output des Encoders $\mathbf{Z} \in \mathbb{R}^{seq\_len \times d\_model}$. Um nun eine RUL Vorhersage vornehmen zu können, wird $\mathbf{Z}$ durch verschiedene dense-Schichten geleitet. Zuerst wird $\mathbf{Z}$ in eine Flatten-Schicht geführt, ehe der Output aus dieser in eine dense-Schicht mit 128 Neuronen und einer ReLu Aktivierungsfunktion gespeist wird. Abschließend besteht die finale Dense-Schicht aus einem Neuron mit Sigmoid Aktivierungsfunktion, welches die Vorhersage für die binäre Klassifikation und die RUL Vorhersage $y \in [0, 1]$ trifft.

7.4.4 Weitere Komponenten

Der Aufbau des Transformers besteht darüber hinaus aus der MHA-Schicht, dem feed-forward Netz sowie der Normalisierungsschicht. An diesem wurden im Vergleich zu dem Transformer-Modell gemäß des in [53] beschriebenen Aufbau keine Änderungen vorgenommen.

7.5 Evaluierung

In diesem Kapitel wird das in Kapitel 7.4 erstellte Modell mit den in Kapitel 7.3 gebildeten Datensätzen evaluiert. Hierzu werden zunächst die Trainingsbedingungen beschrieben, ehe die Ergebnisse präsentiert werden. Abschließend werden die Ergebnisse diskutiert.

7.5.1 Trainingsbedingungen

7.5.1.1 Hyperparameter

Die Größe des Inputs wird auf Grund der fest definierten Länge einer Zeitreihe auf 240 gesetzt. Das Embedding wird als $d_{model} = 8$ gewählt. Die Anzahl der Attention Heads wird auf 6 festgelegt, was auch die Wahl von [53] ist. Die Anzahl der Encoderschichten wird auf 1 gesetzt, da eine höhe Anzahl in zahlreichen Versuchen kein besseres Ergebnis lieferte, den Rechenaufwand jedoch enorm erhöhte. Die innere Dimension der MHA wird mit 64 gewählt, was sich ebenfalls in verschiedenen Tests als eine gute Einstellung erwies. Hierbei ist zu beachten, dass sowohl ein zu kleiner, als auch ein zu großer Wert, den Trainingsprozess des Transformers stark beeinträchtigen kann. Zusätzlich wurde drop-out mit 0.1 implementiert. Größere Werte sind nicht zu empfehlen, da diese für einen instabilen Trainingsprozess sorgen können. Eine Übersicht aller gewählten Hyperparameter ist in Tabelle 7.1 gegeben.

Hyperparameter	Gewählte Einstellung
Eingangsgröße ($units = seq\_len$)	240
Embedding Dimension (d_model)	8
Anzahl Schichten encoder	1
Anzahl attention heads	6
Dimension MHA	64
Dimension des feed-forward Netzwerks	32
Dimension des classification head	128
Dropout	0.1

Tabelle 7.1: Übersicht über die Auswahl der Hyperparameter des Transformer-Modells

7.5.1.2 Trainingseinstellung

Als grundlegende Trainingseinstellung wurde eine Trainingszeit von 500 Epochen und eine batch size von 16 ausgewählt. Die Learning-rate wird wie in [53] bestimmt. Auf Grund der Erkenntnisse in anderen Arbeiten, dass eine große Anzahl an warm-up Steps nötig sind, um eine zu große Learning-rate zu Beginn des Trainingsprozess zu vermeiden, werden 200.000 warm-up steps gewählt [126], [127], [128], [129]. Als optimizer wird Adam [57] genutzt. Als Loss-Funktion wird binary Crossentropy loss verwendet. Zusätzlich wird eine Funktion implementiert, die

den Trainingsprozess abbricht, wenn über 20 Epochen hinweg keine Verbesserung der Acc auf den Validierungsdaten erzielt werden kann (Patience). Durch diese Funktion soll ein mögliches Overfitting über die lange Trainingsdauer von 500 Epochen verhindert werden. Eine Übersicht der Trainingsparameter ist in Tabelle 7.2 gegeben.

Trainingsparameter	Gewählte Einstellung
Warmup steps	200,000
Batch size	16
Epochen	500
Patience	20
Loss	binary Crossentropy
Optimizer	Adam

Tabelle 7.2: Übersicht über die Trainingsparameter der Evaluation

7.5.1.3 Datenaufteilung

Alle Datensätze werden zunächst gemischt und anschließend auf Trainingsdaten, Testdaten und Validierungsdaten aufgeteilt. Als Aufteilungsverhältnis von $\mathbf{X}_{RUL}$ wird ein Split von 80 % Trainingsdaten, 10 % Testdaten und 10 % Validierungsdaten gewählt, damit dem Transformer ausreichend Trainingsdaten zur Verfügung stehen. Für die angereicherten Datensätze $\mathbf{X}_{RUL-aug1}$, $\mathbf{X}_{RUL-aug2}$ und $\mathbf{X}_{RUL-aug3}$ wird ein Aufteilungsverhältnis von 60 % Trainingsdaten, 20 % Testdaten und 20 % Validierungsdaten gewählt.

7.5.2 Ergebnisse

Für alle Datensätze wurden fünf Durchläufe ausgeführt. Tabelle 7.3 zeigt die erzielten Ergebnisse. Dargestellt sind die Acc, die confusion Matrix und der F1-Score für alle Experimente. Zusätzlich ist die durchschnittliche Acc, der durchschnittliche F1-Score, die Standartabweichung σ der Acc sowie die Standartabweichung des F1-Scores über alle fünf Durchläufe eines Datensatzes gegeben. Darüber hinaus ist der durchschnittliche Anteil an FP Prädiktionen gegeben. Da bei praktischer Anwendung des Systems FP Prädiktionen zu nicht notwendigen Austauschen und damit zu unnötigen Kosten führen können, ist dieser Wert ebenfalls von hoher Signifikanz.

Für den Datensatz $\mathbf{X}_{RUL}$, der nur aus Originaldaten besteht, beträgt die durchschnittliche Acc 96.39 % und der durchschnittliche F1-Score 96.42 %. Die höchste

Nr	Metrik	$\mathbf{X}_{RUL}$	$\mathbf{X}_{RUL-aug1}$	$\mathbf{X}_{RUL-aug2}$	$\mathbf{X}_{RUL-aug3}$
1	Acc	0.9668	0.9859	0.9860	0.9840
	Conf. Matrix	[[5141 173] [159 4527]]	[[97311 1247] [1555 99887]]	[[96561 1299] [1500 100640]]	[[100041 1139] [2058 96762]]
	F1-score	0.9646	0.9862	0.9862	0.9837
2	Acc	0.9716	0.9850	0.9835	0.9820
	Conf. Matrix	[[4832 161] [123 4884]]	[[100241 1139] [1858 96762]]	[[93034 2001] [1300 103665]]	[[96301 1805] [1789 100105]]
	F1-score	0.9717	0.9847	0.9843	0.9824
3	Acc	0.9520	0.9855	0.9854	0.9795
	Conf. Matrix	[[4210 303] [177 5310]]	[[98191 1189] [1709 98911]]	[[97210 1247] [1666 99877]]	[[92234 2236] [1865 103665]]
	F1-score	0.9568	0.9856	0.9856	0.9806
4	Acc	0.9683	0.9845	0.9850	0.9835
	Conf. Matrix	[[4806 163] [154 4877]]	[[92234 2001] [1100 104665]]	[[106261 862] [2134 90743]]	[[97111 1551] [1751 99587]]
	F1-score	0.9685	0.9854	0.9838	0.9837
5	Acc	0.9610	0.9862	0.9816	0.9751
	Conf. Matrix	[[5023 124] [266 4587]]	[[103766 989] [1762 93583]]	[[96311 1895] [1789 100005]]	[[97768 2300] [2689 97243]]
	F1-score	0.9675	0.9819	0.9819	0.9750
$\emptyset$	Acc	0.9639	0.9855	0.9843	0.9808
$\emptyset$	F1-Score	0.9642	0.9854	0.9844	0.9811
σ	Acc	0.0069	0.0006	0.0016	0.0033
σ	F1-Score	0.0056	0.0005	0.0015	0.0033
$\emptyset$	Anteil FP	0.0185	0.0066	0.0073	0.0090

Tabelle 7.3: Ergebnisse der Ausfallvorhersage für den originalen Datensatz $\mathbf{X}_{RUL}$ sowie die angereicherten Datensätze $\mathbf{X}_{RUL-aug1}, \mathbf{X}_{RUL-aug2}$ und $\mathbf{X}_{RUL-aug3}$.

Acc beträgt 96.83 % und die niedrigste 95.20 %. Der beste F1-Score beträgt 96.85 % der schlechteste 95.68 %. Dies lässt auf einen robusten Trainingsprozess schließen und kann durch die niedrigen Standartabweichungen für Acc (0.69 %) und F1-Score (0.56 %) bestätigt werden. Mit Hinblick auf die confusion Matrizen lässt sich festhalten, dass kein klarer Trend erkennbar ist, welche Klasse besser detektiert werden kann. Weder FP und FN treten überproportional häufig auf. Der

Anteil an FP Prädiktionen liegt durchschnittlich bei 1.85%.

Die angereicherten Datensätze liefern im Vergleich zu $\mathbf{X}_{RUL}$ leicht verbesserte Ergebnisse in allen Metriken. Die besten Ergebnisse wurden mit $\mathbf{X}_{RUL-aug1}$ erreicht. Mit Hilfe dieses Datensatzes ist eine durchschnittliche Acc von 98.55 % und ein durchschnittlicher F1 Score von 98.54 % erreicht worden. Das beste Ergebnis in Acc wurde in Durchlauf 5 erreicht und beträgt 98.62 %. $\mathbf{X}_{RUL-aug2}$ liefert mit einer durchschnittlichen Acc von 98.43 % und einem durchschnittlichen F1-Score von 98.44 % ebenfalls sehr gute Ergebnisse. Mit $\mathbf{X}_{RUL-aug3}$ wurde eine durchschnittliche Acc von 98.08 % und ein durchschnittlicher F1-Score von 98.11 % erreicht.

Mit Hinblick auf die Standardabweichung in Acc und F1-Score ist festzuhalten, dass die Abweichungen aller erweiterten Datensätze geringer ausfallen als die von $\mathbf{X}_{RUL}$. Das robusteste Training liefert $\mathbf{X}_{RUL-aug1}$ mit einer Standartabweichung von 0.06 % für Acc und 0.05 % für F1-Score. Auch die Abweichungen für $\mathbf{X}_{RUL-aug2}$ (0.16 % Acc und 0.15 % F1-Score) $\mathbf{X}_{RUL-aug3}$ (0.33 % Acc und 0.33 % F1-Score) fallen gering aus und lassen auf einen robusten Trainingsprozess schließen.

Bei Betrachtung der unterschiedlichen confusion Matrizen bestätigt sich die Beobachtung, die für $\mathbf{X}_{RUL}$ gemacht werden konnte. Beide Klassen können zuverlässig klassifiziert werden. Ein eindeutiger Trend, welche Klasse durch den Algorithmus besser generalisiert werden konnte, ist nicht zu erkennen.

Auch die FP Prädiktionen konnten mit den erweiterten Datensätzen erheblich reduziert werden. $\mathbf{X}_{RUL-aug1}$ weist nur einen durchschnittlichen Anteil von 0.66 % FP Prädiktionen auf. Dies entspricht einem Rückgang von ca. 64 % im Vergleich zu dem mit $\mathbf{X}_{RUL}$ erreichten Ergebnis.

Insgesamt lässt sich festhalten, dass mit allen Datensätzen sehr vielversprechende Ergebnisse erreicht werden konnten. Die Anreicherung des Datensatzes mit synthetischen Samples kann ebenfalls als erfolgreich bewertet werden. Der VAE als Werkzeug zur Erzeugung künstlicher Daten hat die positiven Ergebnisse aus Kapitel 3.4 bestätigt.

7.5.3 Diskussion

Die Evaluation hat gezeigt, dass mit Hilfe des Stromsignals I1 eine präzise Ausfallvorhersage für SMs möglich ist. Wird das Stromsignal während des Betriebs vermessen und mit dem vorgestellten Modell ausgewertet, kann 30 Tage vor Auftreten eines Defekts eine Warnung ausgegeben werden. Nach Ausgabe der Warnung können Ersatzteile bestellt und Fachhandwerker mit dem Austausch des zukünftig defekten Bauteils beauftragt werden.

Insgesamt sind vor allem der hohe F1 Score, sowie der niedrige Anteil von FP-Prädiktionen hervorzuheben. Der hohe F1 Score lässt darauf schließen, dass das Modell beide Klassen gleichermaßen gut generalisiert. Der niedrige Anteil von

FP-Prädiktionen ist vor allem für den realen Anwendungsfall von hoher Wichtigkeit. Wenn ein SM-Austausch durch eine FP Prädiktion ausgelöst wird, entstehen unnötige Kosten. Bei einer FN-Prädiktion kann hingegen nur der Vorteil der Ausfallvorhersage nicht genutzt werden. Es entstehen somit keine Kosten, die ohne das System nicht angefallen wären.

Mit Hilfe angereicherter Datensätze konnten die Ergebnisse im Vergleich zum Basisdatensatz verbessert werden. Die besten Ergebnisse mit einer durchschnittlichen Acc von 98.55 % und einem Anteil von 0.66 % FP-Prädiktionen wurden mit dem Datensatz $\mathbf{X}_{RUL-aug1}$ erreicht. Die synthetischen Samples dieses Datensatzes wurden mit der größten verwendeten VAE Architektur erzeugt. Eine mögliche Begründung ist, dass im Vergleich zu den in Kapitel 3.4 durchgeführten Experimenten, wesentlich mehr Trainingsdaten zur Verfügung standen. Es ist daher denkbar, dass eine VAE mit noch mehr trainierbaren Parametern zu noch besseren Ergebnissen führen kann.

7.6 Zusammenfassung & Diskussion

In diesem Kapitel wurde ein System entwickelt, mit dem sich Fehlerfälle von SM prognostizieren lassen. Mit Hilfe der Vorhersage sollen Stillstandszeiten vermieden und Kosten reduziert werden.

Um die grundlegende Datenbasis für die Fehlervorhersage zu erstellen war es notwendig, die Betriebszeit der SM im Dauerversuchsaufbau in reale Betriebszeit zu übertragen. Hierzu wurde nach interner Kommunikation mit dem Industriepartner beschlossen, die erste Welle von Fehlerfällen mit der Lebensdauer gleichzusetzen, die in der Industrie kalkuliert wird, um Austausche vorzunehmen. Andere Ansätze sind denkbar, jedoch auf Grund mangelnder Möglichkeit zur Quantifizierung nicht valide umzusetzen.

Darüber hinaus war es notwendig zu definieren, mit welchem Vorlauf eine Messung eine Warnung auslöst. Hier wurden 30 Tage als Maß gewählt. Dieser Zeitraum gibt ausreichend Zeit für die Bestellung eines Ersatzteils und den Einsatz eines Fachhandwerkers, führt jedoch nicht zu unnötig frühen Austauschen. Die Wahl von wesentlich größeren oder kleineren Zeiträumen ist denkbar, wird jedoch für den Anwendungsfall als nicht praktikabel befunden. Anschließend wurden die Daten hinsichtlich der beschriebenen Parameter in zwei Klassen unterteilt und ein Datensatz zur Ausfallvorhersage gebildet. Mit Hilfe dieses Datensatzes, sowie der in Kapitel 3.4 vorgestellten DA Methode wurden synthetische Samples für drei weitere Datensätze erzeugt.

Daraufhin wurde ein Transformer-Modell für die binäre Klassifikationsaufgabe entworfen. Hierbei wurde das in Kapitel 2.4.4 vorgestellte Modell unter Berücksichtigung vorhandener Arbeiten sowie eigener Tests für die vorliegende Aufgaben-

stellung angepasst.

Nachfolgend konnte in der Evaluation gezeigt werden, dass das Modell zur Vorhersage von Fehlerfällen grundsätzlich geeignet ist. Dies lässt sich durch die hohe Acc von 98.55 %, den hohen F1-Score von 98.54 % sowie einer FP-Rate von 0.66 % belegen. Besonders die geringe FP-Rate ist für den realen Anwendungsfall wichtig, da bei einer FP-Prädiktion nicht notwendige Kosten entstehen würden.

Insgesamt lässt sich festhalten, dass mit Transformern nicht nur im Themenfeld natural language processing gute Ergebnisse erzielt werden können, sondern diese sich auch für die Analyse von Zeitreihen eignen. Da Transformer jedoch noch eine junge Technologie darstellen, gibt es noch viel Raum für Verbesserungen. Bisher stehen im Vergleich zu bekannten Algorithmen wie kNN oder SVM nur wenig Literatur und Hinweise für vorteilhafte Implementierungen zur Verfügung. Durch weitere Erforschung unterschiedlicher Architekturen, Parameterkombinationen und des besten Trainingsprozesses können Ergebnisse weiter verbessert und die Implementierung auf zu behandelnde Aufgabenstellungen optimiert werden. Es ist zu erwarten, dass sich Transformer als Methode des maschinellen Lernens in den nächsten Jahren zunehmend standardisieren, da in unterschiedlichen Bereichen vielversprechende Ergebnisse erzielt werden konnten. Die Ergebnisse dieser Arbeit bestätigen dies.

Für die zukünftige Weiterentwicklung des entworfenen Systems gibt es mehrere Ansatzpunkte:

- Eine Datenbasis mit einer größeren Anzahl von Fehlerfällen ist wünschenswert. Der Datensatz dieser Arbeit beschränkt sich auf 20 defekte SM. Durch einen industriellen Feldtest ist es möglich, weitaus mehr Daten von und vor Fehlerfällen zu sammeln. Diese stellen dem Algorithmus mehr und vielfältigere Trainingsdaten zur Verfügung, da bei 20 Defekten nicht davon ausgegangen werden kann, dass alle möglichen Defekt-Muster abgebildet sind.

- Die Evaluation unterschiedlicher DA-Methoden kann helfen eine bessere Datenbasis zu erstellen. DA-Methoden können jedoch nicht den angesprochenen Feldtest ersetzen, da neue Fehlerbilder dadurch nicht abgebildet werden können.

- Das entworfene Modell wurde in vielen Tests untersucht und immer weiter für die vorliegende Aufgabenstellung angepasst. Auf Grund der nur spärlich vorhanden Literatur kann davon ausgegangen werden, dass hier noch großes Potential zur Verbesserung besteht.

- Um die Implementierung auf rechenschwachen Einheiten zu ermöglichen, ist es notwendig einen anderen Algorithmus zu verwenden, da Transformer hohe Rechenkosten erzeugen.

Kapitel 8

Fazit und Ausblick

Diese Arbeit leistet einen Beitrag zu den Forschungsfeldern Data Augmentation (DA) und Remaining Useful Lifetime (RUL) Prediction. Für den Bereich DA wurden neue und innovative Ansätze zur Erweiterung von Zeitreihen-Datensätzen entwickelt und vorgestellt. Diese neu- und weiterentwickelten Methoden bieten der Forschung Möglichkeit für weitere Experimente und vielversprechende Ansätze um im Bereich DA neue standardisierte und zukunftsfähige Werkzeuge auf den Weg zu bringen. Für das Themengebiet RUL Prediction konnte eine innovative und funktionale Fehlervorhersage für Schrittmotoren unter Einsatz von Transformern entwickelt werden. Hierbei wurden Transformer als neuartiger Algorithmus für die Analyse von Zeitreihen weiter erforscht. Zusätzlich wurde ein umfangreicher Datensatz für die Fehlererkennung an Schrittmotoren generiert, welcher ebenfalls viel Raum für weitere Experimente bietet.

Ziel der Arbeit war es, ein System zur frühzeitigen Erkennung von Fehlerfällen an Schrittmotoren zu entwickeln. Darüber hinaus war die Anforderung, dem Mangel an vorhandenen Methoden zur Erweiterung von Zeitreihen-Datensätzen entgegenzuwirken und neue Methoden zu entwickeln und zu erproben. Zusätzlich bestand der Anspruch, eine Methode zur Erkennung und Reduktion schädlicher Betriebsbereiche des Schrittmotors zu entwerfen um Verschleiß vorzubeugen.

Nach der Einleitung in Kapitel 1 wurden zunächst in Kapitel 2 die notwendigen Grundlagen erläutert. Es wurde ein Einblick in verschiedene Algorithmen des maschinellen Lernens gegeben sowie in die Themenbereiche RUL Prediction, DA und Rekurrenzplots anhand einer Literaturrecherche eingeführt. Darüber hinaus wurde das Funktionsprinzip von Schrittmotoren näher beschrieben.

Um dem Anspruch an die Erforschung neuartiger DA Methoden zu genügen, wurden in Kapitel 3 drei innovative Ansätze zur Erweiterung von Zeitreihen-Datensätzen präsentiert. Zunächst wurden neun bekannte Zeitreihen-Datensätze mit un-

terschiedlichen Klassifikationsaufgaben vorgestellt, die zur Evaluation der neuen Methoden dienten. Als Klassifikationsalgorithmen für die Evaluation der Methoden wurden die Support Vector Machine (SVM) und der k Nearest Neighbors (kNN) ausgewählt.

Die erste neu entwickelte Methode trägt die Bezeichnung White Noise Windows (WNW). Ziel der Technik ist es, die Generalisierungsfähigkeit von Algorithmen zu erhöhen. Die Evaluation hat gezeigt, dass die Leistung der Algorithmen starken Schwankungen unterliegt. Es wurden sowohl verbesserte Ergebnisse als auch Leistungseinbrüche verzeichnet. Voraussetzung für ein verbessertes Ergebnis ist die richtige Wahl der Parameter von WNW.

Als nächstes wurde ein innovativer Ansatz zur Erzeugung synthetischer Trainingsdaten unter Einsatz eines Variational Autoencoder (VAE) gezeigt. Das Bestreben der Methode ist es, künstliche Daten zu erzeugen, die realen Daten ähneln und sich daher zur Anreicherung von Datensätzen eignen. Mit Hilfe dieses Ansatzes wurden sehr gute Ergebnisse erzielt. Sowohl für die SVM als auch für den kNN konnten verbesserte Ergebnisse für alle Datensätze erreicht werden.

Die dritte neuartige Technik wurde für die Erweiterung von Spektrogramm-Datensätzen entwickelt und als Random Noise Boxes (RNB) benannt. RNB hat als Ziel, Overfitting von Algorithmen zu vermeiden. In der Evaluation konnte gezeigt werden, dass mit RNB Verbesserungen aber auch Verschlechterungen auftreten können. Entscheidend hierfür ist die Wahl der richtigen Parameter von RNB sowie die Beschaffenheit des zu erweiternden Datensatzes.

Somit wurden in dieser Arbeit drei aussichtsreiche DA-Methoden zur Erweiterung von Zeitreihen-Datensätzen entwickelt und getestet. Die besten Ergebnisse konnten mit der Erzeugung von synthetischen Daten mittels VAE erreicht werden. Diese Technik liefert verbesserte Resultate für alle Evaluations-Datensätze und wird für zukünftige Anwendungen als vielversprechendste Technolgie eingestuft. Alle Methoden stellen interessante Ansätze dar und bieten viel Raum für Weiterentwicklung. Es lässt sich festhalten, dass dem Anspruch an die Entwicklung und Erprobung neuer DA-Methoden hiermit erfolgreich genügt wurde.

Kapitel 4 präsentiert eine neuartige Methode zur Analyse von Zeitreihen mit Hilfe von Rekurrenzplots. Rekurrenzplots haben die Eigenschaft, wiederkehrende Zustände eines dynamischen Systems zu visualisieren. Es wurde gezeigt, dass Rekurrenzplots durch diese Eigenschaft eine gute Möglichkeit zur Analyse von Zeitreihendaten darstellen können. Ein einheitlicher Trend, welche Charakteristika der unterschiedlichen Datensätze darauf hinweisen, dass eine Analyse mittels Rekurrenzplots erfolgversprechend ist, konnte allerdings nicht gefunden werden. Um diesen Ansatz weiter zu verfolgen, werden für zukünftige Arbeiten weitere Evaluationen mit einer größeren Anzahl an unterschiedliche Datensätzen als sinnvoll erachtet.

Hauptziel der Arbeit war es, ein System zur Fehlervorhersage an Schrittmotoren zu entwerfen. Darüber hinaus sollte eine Methode zur Erkennung schädlicher Betriebsbereiche entwickelt werden. Für beide Anwendungsfälle ist eine umfangreiche Datenbasis über das Betriebsverhalten von Schrittmotoren erforderlich, die einen guten Trainingsprozess des gewählten Modells ermöglicht. Um eine zuverlässige Fehlervorhersage mittels künstlicher Intelligenz zu gewährleisten, sollte die Datenbasis über Betriebsdaten verfügen, die über die komplette Lebensdauer des Prüflings aufgezeichnet wurden. Für die Identifikation schädlicher Betriebsbereiche müssen Betriebsdaten dieser Bereiche in der Datenbasis enthalten sein.

Kapitel 5 beschreibt Entwurf und Aufbau eines Dauerversuchs, der es erlaubt, eine solche Datenbasis für Schrittmotoren zu erstellen. Hierzu wurden unterschiedliche Signale von 32 Schrittmotoren über eine Betriebsdauer von ca. einem Jahr aufgenommen. Die Schrittmotoren wurden nach einem bestimmten Muster betrieben, welches alle Betriebsbereiche abbildet sowie Referenzfahrten simuliert. Um unterschiedliche klimatische Zustände nachbilden zu können und die Belastung der Schrittmotoren für den Dauerversuch zu erhöhen, wurden die Schrittmotoren über den kompletten Vermessungszeitraum in einer klimatisierten Einheit betrieben. Insgesamt konnte so eine Datenbasis mit ca. 169 Millionen Trainingsdaten von sechs Signalen generiert werden, die in unterschiedlichen Arbeitsbereichen des Schrittmotors sowie bei verschiedenen Temperaturen aufgezeichnet wurden. Während des Dauerversuchs sind an 20 der 32 Schrittmotoren Defekte aufgetreten, welche ebenfalls in der Datenbasis abgebildet sind und die Grundlage für ein System zur Fehlervorhersage bilden.

Kapitel 6 beschäftigt sich mit der Entwicklung einer Methode zur Erkennung schädlicher Betriebsbereiche von Schrittmotoren und greift hierzu auf die in Kapitel 5 generierten Daten zurück. Auf Grund der Periodizität des zu untersuchenden Signals im normalen Arbeitsbereich, sowie des chaotischen Verhaltens des Signals außerhalb des normalen Arbeitsbereichs, werden Rekurrenzplots genutzt, um den schädlichen Betrieb zu identifizieren. Diese werden mit einem CNN in die Klassen „Betrieb im normalen Arbeitsbereich“ und „Betrieb außerhalb des normalen Arbeitsbereichs“ klassifiziert. Die Evaluation zeigt sehr gute Klassifikationsergebnisse. Die Erkennung des schädlichen Betriebs ist zu 99.84 % möglich. Zudem konnte nachgewiesen werden, dass sich Rekurrenzplots für die gegebene Aufgabenstellung gut eignen, da die Ergebnisse eine höhere Accuracy (Acc) und einen verbesserten F1-Score aufweisen, als jene, die mit den herkömmlichen Zeitreihen erreicht wurden.

Mit Hilfe der entwickelten Methode ist es möglich, die Dauer von Referenzfahrten zu vermindern und so den Betrieb im schädlichen Bereich zu reduzieren. Dies führt zu einem geringeren Verschleiß von Schrittmotoren in der praktischen Anwendung. Es lässt sich daher festhalten, dass das Ziel, des Entwurfs eines Mo-

dells zur Erkennung schädlicher Betriebsbereiche, erreicht wurde. Mit Hinblick auf eine industrielle Anwendung sollte beachtet werden, dass die Auswertung von Rekurrenzplots mit CNNs wesentlich höhere Berechnungskosten verursacht, als die Auswertung der herkömmlichen Zeitreihe mit einem Standard-Klassifizierer wie kNN oder SVM.

In Kapitel 7 wurde ein System zur frühzeitigen Fehlererkennung entworfen. Um einen Transformer zu trainieren, der es ermöglicht Fehlerfälle vorherzusagen, wurden die in Kapitel 5 generierten Daten genutzt. Die Vorlaufzeit bis zu einem erwarteten Ausfall wurde mit 30 Tagen festgelegt. Dieser Zeitraum bietet ausreichend Möglichkeit zum Austausch der demnächst ausfallenden Komponente und lässt zugleich eine für den Trainingsprozess ausreichend große Datenbasis zu. Zusätzlich wurden drei erweiterte Datensätze mit Hilfe der in Kapitel 3 vorgestellten Methode mit VAEs erstellt. Die Evaluation zeigt, dass eine Fehlervorhersage zuverlässig umgesetzt werden konnte. Mit dem Originaldatensatz konnte eine Acc von 96.39 %, ein F1-Score von 96.42 % und einer FP-Rate von 1.85 % erreicht werden. Das beste Ergebnis lieferte ein erweiterter Datensatz mit einer Acc von 98.55 %, F1-Score von 98.54 % und einer FP-Rate von 0.66 %.

Es lässt sich festhalten, dass ein funktionales System zur Ausfallvorhersage entworfen werden konnte und damit das Hauptziel der Arbeit erreicht wurde. Das System ermöglicht in 98.55 % aller Fällen den notwendigen Austausch des Schrittmotors im Voraus zu veranlassen und so ungewünschte Fehlerauftritte zu vermeiden. Nur in 0.66 % aller Fälle wird eine noch funktionstüchtige Komponente fälschlicherweise als zukünftige Fehlerquelle identifiziert. Durch rechtzeitigen Austausch können Kosten gespart und der Kundenkomfort erhöht werden. Werden Schrittmotoren in Industrieanlagen überwacht, ermöglicht das System die Verhinderung von ungewünschten Stillstandszeiten. Zusätzlich konnten Transfomer für die Analyse von Zeitreihen mit Fokus auf die Vorhersage von Fehlerfällen als geeignetes Werkzeug identifiziert werden.

Zusammenfassend läßt sich sagen, dass alle angestrebten Ziele in dieser Arbeit erreicht wurden. Es konnten drei innovative DA-Methoden für Zeitreihen entwickelt und erprobt werden. Zusätzlich konnte eine große Datenbasis über Schrittmotoren mit Hilfe eines Dauerversuchs generiert werden. Diese kann genutzt werden, um Modelle zur Detektion von schädlichen Betriebsbereichen und zur Vorhersage von Fehlerfällen zu trainieren. Mit Hilfe von Rekurrenzplots ist ein funktionales Modell zur Erkennung von schädlichen Betriebsbereichen entwickelt worden, das die Verkürzung von Referenzfahrten ermöglicht und so unnötigem Verschleiß vorbeugt. Anschließend konnte auf Basis eines Transformers ein funktionales System zur Fehlervorhersage entwickelt werden. Das System ermöglicht proaktive Wartungs- und Instandhaltungsmaßnahmen, indem Fehlerfälle mit einer Vorlauf-

zeit von 30 Tagen sicher prognostiziert werden.

Für die Zukunft wird davon ausgegangen, dass das Themenfeld RUL Prediction weiter an Bedeutung gewinnen wird. Die steigenden Anforderungen von Kunden, sowie der Konkurrenzdruck in der Industrie untermauern dies. Die stetige Weiterentwicklung der Rechenleistung von Computern und Mikrochips bieten schon jetzt, aber vor allem in Zukunft viele Möglichkeiten für Ansätze auf eingebetteten Systemen.

Eine der großen Herausforderungen für eine erfolgreiche Umsetzung ist, dass für die Aufgabenstellung qualifizierte Datensätze zur Verfügung stehen. Schnellere Verbindungen zur Datenübertragung und die zunehmende Vernetzung von Systemen mit industriellen Rechenzentren werden es den Unternehmen in Zukunft erlauben, solche Datensätze standardmäßig zu sammeln. Dies macht RUL-Prediction für eine Vielzahl von Anwendungsfällen zunehmend interessanter.

Es ist auch davon auszugehen, dass sich DA als wichtiges Feld sowohl im Bereich Zeitreihen, als auch im Bereich der Fehlervorhersage weiter etabliert. Aufwendig zu sammelnde Daten, wie jene kurz vor und während des Eintritts von Fehlerfällen, können so kostengünstig und schnell künstlich erzeugt werden.

Transformer werden als spannende Technologie auch für Zeitreihen eingestuft. Bisher vermehrt im NLP-Bereich eingesetzt, verfügen Sie durch Ihre Eigenschaften über viel Potential im Bereich Zeitreihen. Die erreichten Ergebnisse bestätigen dies.

Literatur

[1] F. Schörlin, *Mit Schrittmotoren steuern, regeln und antreiben*. Franzis Verlag, 1995.

[2] D. Schröder, *Elektrische Antriebe - Grundlagen*. Springer-Vieweg, 2017.

[3] T. Krah, *Entwicklung eines KI-basierten Algorithmus zur Zustandserkennung von Schrittmotoren*, German, 2. Dez. 2020.

[4] A. Binder, *Elektrische Maschinen und Antriebe*. Springer Verlag, 2018.

[5] B. G. Liptak, *Instrument Engineers' Handbook, Volume Two: Process Control and Optimization*. CRC Press, 2018.

[6] S. A. Hasib, S. Islam, R. K. Chakrabortty u. a., "A Comprehensive Review of Available Battery Datasets, RUL Prediction Approaches, and Advanced Battery Management," *IEEE Access*, Jg. 9, S. 86 166–86 193, 2021. DOI: 10.1109/ACCESS.2021.3089032.

[7] A. K. Mahamad, S. Saon und T. Hiyama, "Predicting Remaining Useful Life of Rotating Machinery Based Artificial Neural Network," Jg. 60, S. 1078–1087, 2010. DOI: 10.1016/j.camwa.2010.03.065.

[8] M. Goubeaud, *Entwicklung eines Predictive Maintenance Konzepts zur Vorhersage und Vermeidung von Betriebsausfällen an wandhängenden Gasthermen*, German, 4. Okt. 2018.

[9] X. Si, W. Wang, C.-H. Hu und D. Zhou, "Remaining useful life estimation - A review on the statistical data driven approaches," *Eur. J. Oper. Res.*, Jg. 213, S. 1–14, 2011.

[10] M. Tanwar und N. Raghavan, "Lubricating Oil Remaining Useful Life Prediction Using Multi-Output Gaussian Process Regression," *IEEE Access*, Jg. 8, S. 128 897–128 907, 2020. DOI: 10.1109/ACCESS.2020.3008328.

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar und L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, S. 1725–1732.

[12] Y. Kim, *Convolutional Neural Networks for Sentence Classification*, 2014. arXiv: 1408.5882 [cs.CL].

[13] G. S. Babu, P. Zhao und X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International conference on database systems for advanced applications*, Springer, 2016, S. 214–228.

[14] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *2008 international conference on prognostics and health management*, IEEE, 2008, S. 1–6.

[15] Y. Bengio, P. Simard und P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, Jg. 5, Nr. 2, S. 157–166, 1994.

[16] S. Hochreiter und J. Schmidhuber, "Long short-term memory," *Neural computation*, Jg. 9, Nr. 8, S. 1735–1780, 1997.

[17] S. Zheng, K. Ristovski, A. Farahat und C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE international conference on prognostics and health management (ICPHM)*, IEEE, 2017, S. 88–95.

[18] J. Wang, G. Wen, S. Yang und Y. Liu, "Remaining useful life estimation in prognostics using deep bidirectional lstm neural network," in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, IEEE, 2018, S. 1037–1042.

[19] V. Mathew, T. Toby, V. Singh, B. M. Rao und M. G. Kumar, "Prediction of Remaining Useful Lifetime (RUL) of turbofan engine using machine learning," in *2017 IEEE International Conference on Circuits and Systems (ICCS)*, 2017, S. 306–311. DOI: 10.1109/ICCS1.2017.8326010.

[20] W. Mao, J. He, J. Tang und Y. Li, "Predicting remaining useful life of rolling bearings based on deep feature representation and long short-term memory neural network," *Advances in Mechanical Engineering*, Jg. 10, Nr. 12, S. 1 687 814 018 817 184, 2018.

[21] K. Park, Y. Choi, W. J. Choi, H.-Y. Ryu und H. Kim, "LSTM-based battery remaining useful life prediction with multi-channel charging profiles," *Ieee Access*, Jg. 8, S. 20 786–20 798, 2020.

[22] J. Li, X. Li und D. He, "A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction," *IEEE Access*, Jg. 7, S. 75 464–75 475, 2019.

[23] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov und H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliability Engineering & System Safety*, Jg. 183, S. 240–251, 2019.

[24] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals Eugenics*, Jg. 7, S. 179–188, 1936.

[25] N. Aronszajn, "Theory of Reproducing Kernels," *Transactions of the American Mathematical Society*, Jg. 68, Nr. 3, S. 337–404, 1950. Adresse: http://dx.doi.org/10.2307/1990404.

[26] V. Vapnik und A. Chervonenkis, *Theory of Pattern Recognition [in Russian]*. Nauka, 1974, (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

[27] V. Vapnik, *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 1982.

[28] B. E. Boser, I. M. Guyon und V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, New York, NY, USA: Association for Computing Machinery, 1992, S. 144–152. DOI: 10.1145/130385.130401.

[29] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995, ISBN: 0-387-94559-8.

[30] S. Tong, C. Yanqiao und Z. Yuan, "Fault prediction of marine diesel engine based on time series and support vector machine," in *2020 International Conference on Intelligent Design (ICID)*, 2020, S. 75–81. DOI: 10.1109/ICID52250.2020.00023.

[31] I. Hammami, L. Salhi und S. Labidi, "Pathological voices detection using Support Vector Machine," in *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2016, S. 662–666. DOI: 10.1109/ATSIP.2016.7523162.

[32] H. Hu, M.-X. Xu und W. Wu, "GMM Supervector Based SVM with Spectral Features for Speech Emotion Recognition," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, Bd. 4, 2007, S. IV-413-IV–416. DOI: 10.1109/ICASSP.2007.366937.

[33] H. Li und J. Cao, "Detection and Segmentation of Moving Objects Based on Support Vector Machine," in *2010 Third International Symposium on Information Processing*, 2010, S. 193–197. DOI: 10.1109/ISIP.2010.35.

[34] S. Abe, *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*. Berlin, Heidelberg: Springer-Verlag, 2005, ISBN: 1852339292.

[35] I. Steinwart und I. Christmann, *Support Vector Machines*. New York: Springer-Verlag, 2008, ISBN: 978-0-387-77242-4.

[36] E. Fix und J. Hodges, *Discriminatory analysis : nonparametric discrimination, consistency properties*. Texas: USAF School of Aviation, 1951.

[37] T. Cover und P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, Jg. 13, S. 21–27, 1967.

[38] M. E. Hellman, "The Nearest Neighbor Classification Rule with a Reject Option," *IEEE Transactions on Systems Science and Cybernetics*, Jg. 6, Nr. 3, S. 179–185, 1970. DOI: 10.1109/TSSC.1970.300339.

[39] K. Fukunaga und L. Hostetler, "k-nearest-neighbor Bayes-risk estimation," *IEEE Transactions on Information Theory*, Jg. 21, Nr. 3, S. 285–293, 1975. DOI: 10.1109/TIT.1975.1055373.

[40] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule," *IEEE Transactions on Systems, Man, and Cybernetics*, Jg. SMC-6, Nr. 4, S. 325–327, 1976. DOI: 10.1109/TSMC.1976.5408784.

[41] C. C, "Prediction of Heart Disease using Different KNN Classifier," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2021, S. 1186–1194. DOI: 10.1109/ICICCS51141.2021.9432178.

[42] ——, "Prediction of Heart Disease using Different KNN Classifier," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2021, S. 1186–1194. DOI: 10.1109/ICICCS51141.2021.9432178.

[43] Y. Tan, "An Improved KNN Text Classification Algorithm Based on K-Medoids and Rough Set," in *2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Bd. 01, 2018, S. 109–113. DOI: 10.1109/IHMSC.2018.00032.

[44] L. Zhou, L. Wang, X. Ge und Q. Shi, "A clustering-Based KNN improved algorithm CLKNN for text classification," in *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, Bd. 3, 2010, S. 212–215. DOI: 10.1109/CAR.2010.5456668.

[45] D. Bajpai und L. He, "Evaluating KNN Performance on WESAD Dataset," in *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2020, S. 60–62. DOI: 10.1109/CICN49253.2020.9242568.

[46] G. Li und J. Zhang, "Music personalized recommendation system based on improved KNN algorithm," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2018, S. 777–781. DOI: 10.1109/IAEAC.2018.8577483.

[47] Y. LeCun, B. Boser, J. S. Denker u. a., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, Jg. 1, Nr. 4, S. 541–551, 1989. DOI: 10.1162/neco.1989.1.4.541.

[48] Y. Lecun, L. Bottou, Y. Bengio und P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, Jg. 86, Nr. 11, S. 2278–2324, 1998. DOI: 10.1109/5.726791.

[49] K. Oh und K. Jung, "GPU implementation of neural networks," *Pattern Recognit.*, Jg. 37, S. 1311–1314, 2004.

[50] A. Krizhevsky, I. Sutskever und G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, Jg. 60, Nr. 6, S. 84–90, Mai 2017. DOI: 10.1145/3065386. Adresse: https://doi.org/10.1145/3065386.

[51] F. Ghorban, "Machine Learning in Advanced Driver-Assistance Systems," Dissertation, University of Wuppertal, 2019.

[52] D. Scherer, A. C. Müller und S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," in *ICANN*, 2010.

[53] A. Vaswani, N. Shazeer, N. Parmar u. a., *Attention Is All You Need*, 2017. arXiv: 1706.03762 [cs.CL].

[54] J. Rosendahl, V. A. K. Tran, W. Wang und H. Ney, "Analysis of positional encodings for neural machine translation,"

[55] L. Schelkes, *Comparison of Statistical and Deep Learning Methods to predict the Remaining Useful Lifetime of Vaillant Gas Boiler Components*, English, 8. Juli 2021.

[56] J. Duchi, "Derivations for linear algebra and optimization," *Berkeley, California*, Jg. 3, Nr. 1, S. 2325–5870, 2007.

[57] D. P. Kingma und J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] D. P. Kingma und M. Welling, "Auto-Encoding Variational Bayes," *CoRR*, Jg. abs/1312.6114, 2014.

[59] A. C. Ian Goodfellow Yoshua Bengio, *Deep Learning - Adaptive Computation and Machine Learning series*. MIT Press, 2016, ISBN: 0262035618.

[60] J. Pereira und M. Silveira, "Learning representations from healthcare time series data for unsupervised anomaly detection," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2019 IEEE International Conference on Big Data and Smart Computing, BigComp 2019 ; Conference date: 27-02-2019 Through 02-03-2019, United States: Institute of Electrical und Electronics Engineers, 2019.

[61] Y. Huang, C.-H. Chen und C.-J. Huang, "Motor Fault Detection and Feature Extraction Using RNN-Based Variational Autoencoder," *IEEE Access*, Jg. 7, S. 139 086–139 096, 2019. DOI: 10.1109/ACCESS.2019.2940769.

[62] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, Jg. 2, Nr. 1, S. 37–63, 2011.

[63] C. Sammut und G. I. Webb, Hrsg., *Encyclopedia of Machine Learning and Data Mining*. New York: Springer, 2017, ISBN: 978-1-4899-7685-7. DOI: 10.1007/978-1-4899-7687-1.

[64] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, Jg. 27, Nr. 8, S. 861–874, 2006, ROC Analysis in Pattern Recognition, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. Adresse: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[65] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger und H. Greenspan, "Synthetic data augmentation using GAN for improved liver lesion classification," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, S. 289–293. DOI: 10.1109/ISBI.2018.8363576.

[66] M. Podduturi, "Data augmentation for supervised learning with generative adversarial networks," 2018.

[67] D. Wagner, K. Kalischewski, S. Tilgner, J. Velten und A. Kummert, "Automatic Labeling of Industrial Images by using Generative Adversarial Networks," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, S. 1–5. DOI: 10.1109/ISCAS.2019.8702195.

[68] A. Y. Hannun, C. Case, J. Casper u. a., "Deep Speech: Scaling up end-to-end speech recognition," *CoRR*, Jg. abs/1412.5567, 2014. Adresse: http://arxiv.org/abs/1412.5567.

[69] K. M. Rashid und J. Louis, "Time-Warping: A Time Series Data Augmentation of IMU Data for Construction Equipment Activity Identification," 2019.

[70] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen und E. Keogh, "Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification," in *2014 IEEE International Conference on Data Mining*, 2014, S. 470–479. DOI: 10.1109/ICDM.2014.27.

[71] C. Kim, A. Misra, K. K. Chin u. a., "Generation of Large-Scale Simulated Utterances in Virtual Rooms to Train Deep-Neural Networks for Far-Field Speech Recognition in Google Home," in *INTERSPEECH*, 2017.

[72] J. Yeomans, S. Thwaites, W. S. P. Robertson, D. Booth, B. Ng und D. Thewlis, "Simulating Time-Series Data for Improved Deep Neural Network Performance," *IEEE Access*, Jg. 7, S. 131 248–131 255, 2019. DOI: 10.1109/ACCESS.2019.2940701.

[73] C. Oh, S. Han und J. Jeong, "Time-Series Data Augmentation based on Interpolation," in *The 17th International Conference on Mobile Systems and Pervasive Computing*, 2020, S. 64–71.

[74] T. T. Um, F. Pfister, D. Pichler u. a., "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, 2017.

[75] A. L. Guennec, S. Malinowski und R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," 2016.

[76] G. Ramponi, P. Protopapas, M. Brambilla und R. Janssen, "T-CGAN: Conditional Generative Adversarial Network for Data Augmentation in Noisy Time Series with Irregular Sampling," *ArXiv*, Jg. abs/1811.08295, 2018.

[77] N. V. Chawla, K. W. Bowyer, L. O. Hall und W. P. Kegelmeyer, "SMOTE: Synthetic Minority over-Sampling Technique," *J. Artif. Int. Res.*, Jg. 16, Nr. 1, S. 321–357, Juni 2002, ISSN: 1076-9757.

[78] J.-P. Eckmann, S. Kamphorst und D. Ruelle, "Recurrence Plots of Dynamical Systems," in *Europhysics Letters*, Bd. 4, 1987, S. 973–977. DOI: 10.1209/0295-5075/4/9/004.

[79] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980, Lecture Notes in Mathematics*, D. Rand und L.-S. Young, Hrsg., Bd. 898, Berlin Heidelberg: Springer-Verlag, 1981, S. 366–381, ISBN: 978-3-540-11171-9. Adresse: http://www.springerlink.com/index/10.1007/BFb0091924.

[80] N. Marwan, M. Romano, M. Thiel und J. Kurths, "Recurrence Plots for the Analysis of Complex Systems," English, *Physics Reports*, Jg. 438, Nr. 5-6, S. 237–329, Jan. 2007, ISSN: 0370-1573. DOI: 10.1016/J.PHYSREP.2006.11.001.

[81] N. Marwan, M. Thiel und J. Kurths, *Recurrence Plots and Cross Recurrence Plots*, ww.recurrence-plot.tk, Accessed: 2021-07-19.

[82] J. Shijie, W. Ping, J. Peiyi und H. Siping, "Research on data augmentation for image classification based on convolution neural networks," in *2017 Chinese Automation Congress (CAC)*, 2017, S. 4165–4170. DOI: 10.1109/CAC.2017.8243510.

[83] H. A. Dau, A. Bagnall, K. Kamgar u. a., *The UCR Time Series Archive*, 2019. arXiv: 1810.07758 [cs.LG].

[84] R. Olszewski, R. Maxion und D. Siewiorek, "Generalized feature extraction for structural pattern recognition in time-series data," 2001.

[85] A. Goldberger, L. Amaral, L. Glass u. a., "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals.," *Circulation*, Jg. 101 23, E215–20, 2000.

[86] Y. Chen, Y. Hao, T. Rakthanmanon, J. Zakaria, B. Hu und E. J. Keogh, "A general framework for never-ending learning from time series streams," *Data Mining and Knowledge Discovery*, Jg. 29, S. 1622–1664, 2014.

[87] A. Mueen, E. Keogh und N. Young, "Logical-Shapelets: An Expressive Primitive for Time Series Classification," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, California, USA: Association for Computing Machinery, 2011, S. 1154–1162, ISBN: 9781450308137. DOI: 10.1145/2020408.2020587.

[88] A. Mueen, E. J. Keogh und N. Young, "Logical-shapelets: an expressive primitive for time series classification," in *KDD*, 2011.

[89] F. Mörchen, "Time Series Knowledge Mining," Dissertation, University of Marburg, 2006.

[90] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya und V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," in *2009 IEEE International Conference on Pervasive Computing and Communications*, 2009, S. 1–9. DOI: 10.1109/PERCOM.2009.4912759.

[91] M. Goubeaud, P. Joußen, N. Gmyrek, F. Ghorban und A. Kummert, "White Noise Windows: Data Augmentation for Time Series," in *2021 7th International Conference on Optimization and Applications (ICOA)*, 2021, S. 1–5. DOI: 10.1109/ICOA51614.2021.9442656.

[92] F. Pedregosa, G. Varoquaux, A. Gramfort u. a., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, Jg. 12, Nr. Oct, S. 2825–2830, 2011.

[93] M. Goubeaud, P. Joußen, N. Gmyrek, F. Ghorban, L. Schelkes und A. Kummert, "Using Variational Autoencoder to augment Sparse Time series Datasets," in *2021 7th International Conference on Optimization and Applications (ICOA)*, 2021, S. 1–6. DOI: 10.1109/ICOA51614.2021.9442619.

[94] M. Goubeaud, N. Gmyrek, F. Ghorban, L. Schelkes und A. Kummert, "Random Noise Boxes: Data Augmentation for Spectrograms," in *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2021, S. 24–28. DOI: 10.1109/PIC53636.2021.9687058.

[95] S.-D. Hyun, I. Choi und N. Soo, "ACOUSTIC SCENE CLASSIFICATION USING PARALLEL COMBINATION OF LSTM AND CNN," 2016.

[96] R. Hyder, S. Ghaffarzadegan, Z. Feng, J. Hansen und T. Hasan, "Acoustic Scene Classification Using a CNN-SuperVector System Trained with Auditory and Spectrogram Image Features," in *INTERSPEECH*, 2017.

[97] J. Huang, B. Chen, B. Yao und W. He, "ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network," *IEEE Access*, Jg. 7, S. 92 871–92 880, 2019. DOI: 10.1109/ACCESS.2019.2928017.

[98] S. Biswal, J. A. Kulas, H. Sun u. a., "SLEEPNET: Automated Sleep Staging System via Deep Learning," *ArXiv*, Jg. abs/1707.08262, 2017.

[99] L. Nanni, G. Maguolo und M. Paci, "Data augmentation approaches for improving animal audio classification," *ArXiv*, Jg. abs/1912.07756, 2020.

[100] G. Maguolo, M. Paci, L. Nanni und L. Bonan, "Audiogmenter: a MATLAB Toolbox for Audio Data Augmentation," *ArXiv*, Jg. abs/1912.05472, 2019.

[101] D. Park, W. Chan, Y. Zhang u. a., "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *INTERSPEECH*, 2019.

[102] K. He, X. Zhang, S. Ren und J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, S. 1026–1034, 2015.

[103] H. A. Dau, E. Keogh, K. Kamgar u. a., *The UCR Time Series Classification Archive*, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/, Okt. 2018.

[104] G. R. Garcia, G. Michau, M. Ducoffe, J. Gupta und O. Fink, "Time Series to Images: Monitoring the Condition of Industrial Assets with Deep Learning Image Processing Algorithms," *ArXiv*, Jg. abs/2005.07031, 2020.

[105] N. Hatami, Y. Gavet und J. Debayle, "Classification of time-series images using deep convolutional neural networks," in *International Conference on Machine Vision*, 2018.

[106] J. S. Iwanski und E. Bradley, "Recurrence plots of experimental data: To embed or not to embed," *Chaos*, S. 861–871, 1998.

[107] D. Yu, H. Wang, P. Chen und Z. Wei, "Mixed Pooling for Convolutional Neural Networks," in *RSKT*, 2014.

[108] F. Chollet u. a., *Keras*, https://keras.io, 2015.

[109] J. Lützenkirchen, *Evaluierung von Konzepten zur Bauteilüberwachung von Gasventilen*, German, 16. Juni 2020.

[110] dSPACE, *MicroLabBox*, [Online; accessed 31-March-2021]. Adresse: %5Curl%7Bhttps://www.dspace.com/en/pub/home/products/hw/microlabbox.cfm#180_23633%7D.

[111] M. Goubeaud, T. Grunert, J. Lützenkirchen, P. Joußen, F. Ghorban und A. Kummert, "Introducing a New Benchmarked Dataset for Mechanical Stop Detection of Stepper Motors," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2020, S. 1–4. DOI: 10.1109/ICECS49266.2020.9294985.

[112] *Vaillant Group*, Internes Dokument, Accessed: 2021-07-16.

[113] N. Wu, B. Green, X. Ben und S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," *arXiv preprint arXiv:2001.08317*, 2020.

[114] S. Li, X. Jin, Y. Xuan u. a., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Advances in Neural Information Processing Systems*, Jg. 32, S. 5243–5253, 2019.

[115] M. Cohen, M. Charbit, S. L. Corff, M. Preda und G. Noziere, "End-to-end deep metamodeling to calibrate and optimize energy loads," *ArXiv*, Jg. abs/2006.12390, 2020.

[116] B. Lim, S. Ö. Arık, N. Loeff und T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, 2021.

[117] X. Jin, Y.-X. Wang und X. Yan, "Inter-Series Attention Model for COVID-19 Forecasting," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, SIAM, 2021, S. 495–503.

[118] Y. Mo, Q. Wu, X. Li und B. Huang, "Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit," *Journal of Intelligent Manufacturing*, S. 1–10, 2021.

[119] A. Saxena, K. Goebel, D. Simon und N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 International Conference on Prognostics and Health Management*, 2008, S. 1–9. DOI: 10.1109/PHM.2008.4711414.

[120] Z. Zhang, W. Song und Q. Li, "Dual Aspect Self-Attention based on Transformer for Remaining Useful Life Prediction," *arXiv preprint arXiv:2106.15842*, 2021.

[121] B. Zhou, C. Cheng, G. Ma und Y. Zhang, "Remaining useful life prediction of lithium-ion battery based on attention mechanism with positional encoding," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, Bd. 895, 2020, S. 012 006.

[122] *Vaillant Group*, Internes Dokument, Accessed: 2021-06-11.

[123] *Vaillant Group*, Internes Dokument, Accessed: 2021-08-02.

[124] G. Klutke, P. Kiessler und M. Wortman, "A critical look at the bathtub curve," *IEEE Transactions on Reliability*, Jg. 52, Nr. 1, S. 125–129, 2003. DOI: 10.1109/TR.2002.804492.

[125] M. Cohen, M. Charbit, S. L. Corff, M. Preda und G. Noziere, "End-to-end deep metamodeling to calibrate and optimize energy loads," *ArXiv*, Jg. abs/2006.12390, 2020.

[126] T. Q. Nguyen und J. Salazar, "Transformers without tears: Improving the normalization of self-attention," *arXiv preprint arXiv:1910.05895*, 2019.

[127] R. Xiong, Y. Yang, D. He u. a., "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*, PMLR, 2020, S. 10 524–10 533.

[128] X. S. Huang, F. Perez, J. Ba und M. Volkovs, "Improving transformer optimization through better initialization," in *International Conference on Machine Learning*, PMLR, 2020, S. 4475–4483.

[129] Q. Wang, B. Li, T. Xiao u. a., "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787*, 2019.