HAW HAMBURG

BACHELORTHESIS
Thomas Weigl

Klassifikation von multivariaten Zeitreihendaten mit Hilfe einer Activity Recognition Chain basierend auf Machine Learning-Verfahren

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Thomas Weigl

Klassifikation von multivariaten Zeitreihendaten mit Hilfe einer Activity Recognition Chain basierend auf Machine Learning-Verfahren

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Stephan Pareigis

Eingereicht am: 30. September 2021

Thomas Weigl

Thema der Arbeit

Klassifikation von multivariaten Zeitreihendaten mit Hilfe einer Activity Recognition Chain basierend auf Machine Learning-Verfahren

Stichworte

Activity Recognition Chain, Human Activity Recognition, Neural Networks, Deep Learning, Zeitreihen, LSTM

Kurzzusammenfassung

Personebezogene Daten, die mit Hilfe von Applikationen zur Optimierung und Überwachung eigener Interessen erhoben werden, können im Bereich der Human Activity Recognition wertvolle Erkenntnisse liefern. So ist ein Teilgebiet, welches mit diesen Daten arbeitet, das Quantified Self, eine immer beliebtere Domäne im Bereich des maschinellen Lernens. So können große Datenmengen dazu beitragen sensorgestütze Applikationen zu entwickeln, die in der Lage sind Bewegungen und Aktivitäten von Peronen zu erkennen. In der vorliegenden Arbeit wird mit Daten, welche durch am Körper getragene Sensoren erhoben wurden, eine Sportübung, die Liegestütze, in ihrer Ausführungsqualität untersucht. Hierfür wurden spezielle Sensormodule, ausgestattet mit einem Gyroskop und einem Beschleunigungssensor, am Körper von Probanden angebracht und Sätze von Liegestützen bis zur Erschöpfung von diesen durchgeführt. Für die Verarbeitung dieser temporalen Datenreihen wird ein Long Short-Term Memory eingesetzt, welches in eine Activity Recogition Chain eingebunden wird. Hierfür wird zuerst eine Hyperparameteroptimierung auf den vorliegenden Daten durchgeführt um die besten Konfigurationen zu erhalten. Basierend auf dem Datensatz werden zwei weitere syntetische Datensätze im Preprocessing von diesem abgeleitet. Auf den syntetischen Datensätzen werden dann mit den errechneten Hyperparametern aus den Ursprungsdaten weitere Modelle trainiert. Schlussendlich wird eine Gegenüberstellung der Modelle für die Auswertung der Ergebnisse betrachtet. In den durchgeführten Experimenten konnte eine maximale Genauigkeit der Vorhersage von 93,92% im Trainingsverlauf erreicht werden. Dies zeigt, dass die Ausführungsqualität von Sportübungen durchaus mit Hilfe LSTMs klassifiziert werden können.

Thomas Weigl

Title of Thesis

Classification of multivariate time series data with the help of an activity recognition chain based on machine learning methods

Keywords

Activity Recognition Chain, Human Activity Recognition, Neural Networks, time series, Deep Learning, LSTM

Abstract

Personal data that is collected with the help of applications to optimize and monitor one's own interests can provide valuable insights in the area of human activity recognition. One area that works with this data, the quantified self, is an increasingly popular domain in the field of machine learning. Large amounts of data can contribute to the development of sensor-based applications that are able to recognize movements and activities of people. In this Thesis, the quality of a sport exercise, the push-up, is investigated using data collected by sensors worn on the body. For this purpose, special sensor modules, equipped with a gyroscope and an acceleration sensor, were attached to the body of test subjects and sets of push-ups were performed by them until they were exhausted. A long short-term memory, which is integrated into an activity recognition chain, is used to process these temporal data series. For this purpose, a hyper-parameter optimization is first carried out on the available data in order to obtain the best configurations. Based on the data set, two further synthetic data sets are derived from it in the preprocessing. Further models are then trained on the synthetic data sets with the hyperparameters calculated from the original data. Finally, a comparison of the models for the evaluation of the results is considered. In the experiments carried out, a maximum accuracy of the prediction of 93.92% during training could be achieved. This shows that the quality of execution of sports exercises can certainly be classified with the help of LSTMs.

Danksagung

Ich danke Prof. Dr. Kai von Luck sowie allen Mitgliedern der Machine Learning AG an der HAW Hamburg für den regen und interessanten Wissensaustausch.

Zusätzlich möchte ich Jan Schwarzer, Andre Jeworutzki und Tobias Eichler aus dem MoGaSens Team der HAW Hamburg für die konstruktive Kritik während der Anfertigung dieser Arbeit meinen besonderen Dank aussprechen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Big Data hat sich in den letzten Jahren rasant entwickelt [9], da viele neue Geschäftsmodelle für große als auch kleine Unternehmen aus dieser Technologie heraus entstanden sind. Aus den gewonnen Daten werden Informationen mit Hilfe von Analysen und Verfahren extrahiert [21]. Doch in Anbetracht der großen Datenmengen wird es immer aufwendiger diese maschinell zu analysieren und Inforamtionen daraus zu gewinnen. Um dem Abhilfe zu leisten werden unter anderem Verfahren aus der linearen Algebra für die Datenverarbeitung eingesetzt, um sinnvolle Ergebnisse aus großen Datenbeständen zu erhalten. Eines dieser Verfahren, das Machine Learning, wurde in den letzten Jahren dank beschleunigter und preiswerter Hardware immer beliebter für diesen Zweck, sodass sich neue Geschäftsfelder und Forschungsbereiche daraus entwickelt haben. Ein Bereich nennt sich Quantified Self. Es beschreibt die Selbstquantifizierung von biologischen und verhaltensbezogenen Daten eines Individuums [48]. Diese Daten werden benutzt um gesundheitliche, gesellschaftliche sowie sportliche Problemstellungen zu lösen.

Um Informationen aus den erzeugten Datenmengen zu extrahieren wird unter anderem das Verfahren der Human Activity Recognition angewendet. Durch verschiedenste Sensoren werden menschliche Bewegungen aufgezeichnet und ausgewertet. So können externe Sensoren wie Infrarot oder am Körper getragene Sensoren, wie z.B. Beschleunigungssensoren oder Gyroskope, verwendet werden [31]. Für die vorliegene Arbeit wurde ein Datensatz verwendet, welcher durch das MoGaSens Team der HAW [3] erstellt wurde. Mit Beschleunigungssensoren und Gyroskopen wurde eine Sportübung, die Liegestütze, von mehreren Testprobanden durchgeführt und aufgezeichnet. Auf dem Datensatz wurden bisher keine maschinellen Lernverfahren angewendet um Bewegungsaktivitäten zu erkennen. Jedoch gibt es eine breite Basis an wissenschaftlichen Arbeiten, welche Deep Learning-Verfahren bereits auf ähnlich aufgebauten Datengrundlagen für die Human Activity Recognition verwendet haben. Hierzu aber mehr in Kapitel 2.

1.1 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist es, Zeitreihendaten mit einem maschinelle Lernverfahren zu untersuchen und auszuwerten. Es werden auf einem vorliegenden interpolierten Datensatz mit Hilfe einer Hyperparameteroptimierung Modelle trainiert, um die besten Konfigurationen zu extrahieren. Die trainierten Modelle sollen Liegestütze binär bezüglich deren Ausführungsqualität klassifizieren. Hierfür wird Gebrauch einer Sonderform von rekurrenten neuronalen Netzen gemacht, dem Long short-term memory (LSTM) [26]. Bisherige Untersuchungen haben bereits gezeigt, dass dieses Verfahren im Bereich der Human Activity Recognition durchaus Potenzial hat und hilfreiche Ergebnisse erzielen kann [49]. Eine Unterscheidung bezüglich der Ausführungsqualität von Liegestütze wird als Klassifikationsproblem betrachtet [34]. So wird eine korrekt ausgeführte sowie eine inkorrekt ausgeführte Liegestütze als Aktivitätsklasse bezeichnet. Die Datenbasis wird bei dessen Erzeugung gleichzeitig durch mehrere Sensoren aufgezeichnet. Daraus folgt, dass es sich um einen multivariaten Datensatz handelt. Der zugrundeliegende interpolierte Datensatz, auf dem das Hyperparametertuning durchgeführt wird, ist zudem die Basis für zwei weitere Datensätze, welche im Preprocessing erzeugt werden. Mit den Datensätzen werden dann weitere Modelle, mit den aus der Hyperparameteroptimierung entstandenen Konfigurationen, trainiert, um eine Vergleichbarkeit der unterschiedlichen Datensätze zu erreichen und diese gegenüberzustellen.

1.2 Struktur der Arbeit

Die vorliegende Arbeit ist in mehrere aufeinander aufbauende Kapitel untergliedert. In Kapitel 2 wird zuerst die Problemstellung erläutert. Darauf folgend werden die wichtigsten Begriffe charakterisiert, welche in der Arbeit verwendet werden und ein Überblick über bisherige Lösungsansätze solcher Problemstellungen gezeigt. Anschließend wird der verwendete Datensatz vorgestellt. Schlussendlich wird die Zielsetzung der Arbeit formuliert und zusammengefasst. Im 3. Kapitel wird der Aufbau und die Umsetzung des Experimentes beschrieben. Angefangen bei der Zusammensetzung der Toolchain und übergehend zur Konstruktion des Experimentes wird abgegrenzt, welche Schritte zur Erstellung nötig sind. In Kapitel 4 wird der Ablauf der Experimente untergliedert. Zuerst wird auf die Durchführung, die Hyperparameteroptimierung und die Ergebnisse der besten Modelle aus der Optimierung eingegangen. Anschließend werden die weiteren Experimente auf abgeleiteten Datensätzen durchgeführt, gegenübergestellt und ein Fazit diskutiert. Im 5.

Kapitel werden meine Erkenntnisse der vorliegenden Arbeit zusammengefasst sowie ein Ausblick in die Zukunft abgegeben.

2 Analyse

Im Analysekapitel werden die Komponenten und Verfahren, auf denen diese Arbeit aufbaut, vorgestellt und erläutert. Hierfür wird zuerst die Problemstellung erörtert. Danach werden grundlegende Methoden, welche für den Aufbau und die Durchführung des Experimentes nötig sind, beschrieben. Darauf folgend wird die Datenerzeugung und der Datensatz charakterisiert, auf dessen Grundlage diese Arbeit durchgeführt wird. Zu guter Letzt wird die inhaltliche Zielsetzung der Arbeit definiert.

2.1 Problemstellung

Menschliche Bewegungen und daraus resultierende Sportübungen können heutzutage effizient mit Sensoren, welche am Körper angebracht sind, aufgezeichnet werden [45]. Unterschieldiche Optionen zur Datenerhebung machen sich jedoch in der Präzession und im Tragekomfort bemerkbar [39]. Im Rahmen des MoGaSens Projektes der HAW-Hamburg wurden die zugrunde liegenden Daten mit Hilfe mehrerer am Körper getragenen Sensoren erfasst. Dies wurde unter Laborbedingungen mit Testprobanden, welche die Sensoren getragen haben, erreicht. Unter sportwissenschaftlichen Vorgaben wurden die ausgeführten und aufgezeichneten Liegstütze nach definierten Standarts bewertet, um eine Unterscheidung von korrekt und inkorrekt ausgeführten Übungen zu ermöglichen. Während der Aktivitätsausführung der Probanden wurden Daten von den am Körper getragenen Beschleunigungssensoren und Gyroskopen erzeugt und aufgezeichnet. Der Datensatz unterscheidet grundlegend zwei dynamische Aktivitätsklassen. Zum einen die nach definierten Kriterien einer korrekt ausgeführten Liegestütze sowie die einer inkorrekt ausgeführten Liegestütze. Zusätzlich war der Datensatz bereits interpoliert, sodass Datenlücken, welche durch technische Probleme entstanden sind, gefüllt wurden. Daraus resultieren temporale Datenreihen welche in dieser Arbeit als binäres Klassifikationsproblem betrachtet werden sollen. Hierfür wird für die Aktivitätserkennung ein Machine Learning-Verfahren, das LSTM, verwendet [19], um ein Modell mit den vorhandenen Daten zu trainieren. Das

Training des Modells und vorgelagerte Schritte werden mit Hilfe einer Prozessablaufkette, der Activity Recognition Chain, durchgeführt, welche ähnlich dem KDD-Prozess ist. Mit einer Hyperparameteroptimierung soll zusätzlich eine akzeptable Konfiguration des Suchraumes entdeckt werden. Weitere Variationen sollen zusätzlich vom interpolierten Datensatz abgeleitet werden, sodass zwei weitere Datensätze mit den gewonnen Hyperparameterkonfigurationen trainiert werden können und eine Gegenüberstellung und Auswertung der besten Modelle aller drei Datensätze stattfinden kann. So kann beobachtet werden, welche Auswirkungen verschiedene Vorverarbeitungsschritte im Preprocessing auf einem Datensatz sowie der Performance von Modellen bei gleichen Hyperparametern haben.

2.2 Vergleichbare Aufgabestellungen

Die Problemstellung, Sportübungen und Aktionen von Probanden mit Deep Learning-Systemen zu erkennen oder zu bewerten ist ein beliebter Task, um aussagekräftige Coaching-Systeme zu etablieren. So wurde bereits gezeigt, dass ein Coaching-System Probanden mit hilfreichen Korrekturen beim Tischtennis unterstützen kann, deren Performance zu verbessern [38]. Ähnliche Experimente wurden auch von Lee und Jo at. al. [37] sowie Hülsmann [28] entwickelt. Jedoch wurden hier Bench Presses sowie Tai Chie-Pushes auf deren Ausführungsqualität mit Hilfe von LSTMs evaluiert und als Coach Information-System abgebildet. So konnte ein Echtzeitfeedbacksystem für Virtual-Reality-Anwendungen [42] entwickelt werden. Bei beiden Arbeiten wurde mit sogenanten Wearable Devices gearbeitet. Ein alternativer Ansatz dazu wäre die Verarbeitung mit visuellen Daten. So hat die Arbeit von Chen at. al. [13] gezeigt, dass eine Erkennung von Aktionen während der Ausführung von sportlichen Aktivitäten durch videobasierte Verarbeitung mit Hilfe von LSTMs sowie Convolutional Neural Networks hohe Erkennungsraten erziehlt hat. Es ist somit durchaus möglich mit geeigneten Technologien brauchbare Ergebnisse für Applikation zu erarbeiten, jedoch ist dies auch von der Qualität und der Größe der verwendeten Daten abhängig. So wurden im vorliegenden Kontext der Arbeit ein Datensatz verwendet, auf dem bisher keine Deep Learning-Verfahren angewendet wurden um die Ausführungsqualität der Übungen zu beurteilen.

2.3 Activity Recognition Chain

Die Activity Recognition Chain hat zum Ziel Aktionen und Aktivitäten von einzelnen sowie mehreren Agenten zu erkennen [27]. Beginnend in den 1980er Jahren hat dieses Forschungsgebiet die Aufmerksamkeit von Wissenschaftlern auf sich gezogen, da personalisierte Anwendungen in vielen Disziplinen, wie zum Beispiel der Plan-, Verhaltens- und Absichtserkennung, einen Mehrwert generieren konnten. So werden menschliche Aktivitäten als eine Reihe von visuellen oder sensorgetriebenen Merkmalen [32] [35] in einer Prozesskette verarbeitet um in Echtzeit Aussagen über verschiedenste Aktivitäten treffen zu können. In den folgenden Unterkapiteln werden die einzelnen Prozessschritte erläutert, welche in Abbildung 2.1 gezeigt werden.

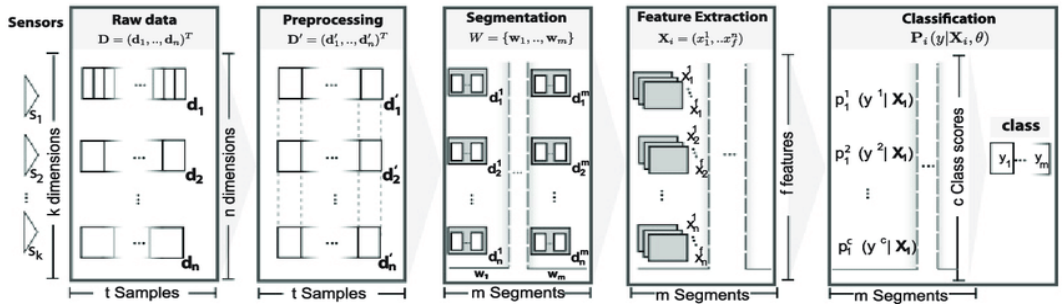

Abbildung 2.1: Prozessschritte einer Activity Recognition Chain [11]

2.3.1 Raw Data

Im ersten Schritt einer Activity Recognition Chain werden zunächst Rohdaten durch unterschiedliche Sensoren, wie zum Beispiel Beschleunigungssensoren und Gyroskopen, welche am Körper des Probanden angebracht sind, erzeugt. Zusätzlich können auch in der Umwelt angebrachte Sensoren die Datengrundlage ergänzen. Die Auswahl und Verwendung der Daten für die nachgelagerten Verarbeitungsschritte hängt starkt vom Kontext der angestrebten Funktionalität ab. So können unter anderem Teile eines Datensatzes ausgewählt oder mehrere Datensätze in Kombination betrachtet werden.

2.3.2 Preprocessing

Der zweite Prozessschritt ist die Vorverarbeitung der ausgewählten Daten. Dieser Schritt wird dafür genutzt, um nicht relevate Daten heraus zu filtern, bestehende Daten zu bearbeiten oder weitere Daten zum Datensatz hinzuzufügen. Unvollständige oder fehlerhaft aufgezeichnete Daten können bei der Vorverarbeitung interploiert oder verworfen werden, sodass ungewollte Interpretationen beim maschinellen Lernen veringert werden.

2.3.3 Segmentation

Im Segmentierungsschritt der Activity Recognition Chain werden die vorverarbeiteten Daten in eine spezielle Form transformiert, sodass diese für maschinelle Lernverfahren verarbeitbar sind. Speziell hängt die Verarbeitungsform und Segmentierung vom Verfahren selbst ab, jedoch gibt es allgemeine Prinzipien, welche bei vielen Verfahren hilfreich sein können. Daten werden zuerst in Listen gesammelt und dort in Arrays angeordnet. Diese Arrays beinhalten numerische Features sowie deren Labels. Die numerischen Daten werden zudem meist in eine normalisierte Form gebracht. So wird jeder numerische Datenpunkt in einen Dezimalbereich zwischen 0 und 1 abgebildet. Die Labels der jeweiligen Daten müssen zudem noch einer speziellen Codierung, dem One-Hot Encoding, unterzogen werden. Das One-Hot Encoding [10] wird eingesetzt, um Labels in eine binäre Form abzubilden. Dabei wird jedes numerische Label einer Liste hinzugefügt. Die Länge der Liste entspricht der Anzahl an Kategorien, die dem Lernverfahren bekannt sein müssen. Jedes Element der Liste stellt eine der Kategorien dar. Ein Listenelement wird einer Kategorie zugeordnet, wenn das korrekte Element den Wert 1 besitzt und die restlichen Elemente der Liste mit dem Wert 0 belegt sind. Nachdem der Transformations- und Segmentierungsschritt durchgeführt wurde, ist der Datensatz für das Training Lernverfahrens verwendbar.

2.3.4 Feature Extraction

Mit Hilfe der Feature Extraction wird die Anzahl der Dimensionen reduziert, die zum Beschreiben eines großen Datensatzes erforderlich sind. Bei der Analyse komplexer Daten liegt eines der Hauptprobleme in der Anzahl der beteiligten Variablen. Die Analyse

mit einer großen Anzahl von Variablen und Unbekannten erfordert im Allgemeinen eine große Menge an Kosten und Leistung. Feature Extraction ist ein allgemeiner Begriff für Verfahren zum Konstruieren von Kombinationen der verwendeten Variablen aus den vorverarbeiteten, transformierten und segmentierten Daten, um Probleme wie Overfitting oder eine schlechte Generalisierung zu vermeiden und gleichzeitig die Daten mit ausreichender Genauigkeit zu beschreiben.

2.3.5 Classification

Ein Modell wird in einer beliebig festgelegten Anzahl von Epochen trainiert. In jeder Epoche wird der Datensatz vollständig für das Training durchlaufen. Das Training über mehrere Epochen steht vor dem Problem, dass das trainierte Modell immer besser auf die vorliegenden Trainingsdaten vorbereitet wird und Gefahr läuft, diese zu gut zu lernen, anstatt globale Muster zu erkennen. Dieses Problem wird als Overfitting bezeichnet [18]. So mag ein Modell mit starkem Overfitting gute Ergebnisse auf dem Trainingsdatensatz erzielen. Wird es jedoch auf den Daten getestet, welche dem Modell nicht aus dem bisherigen Training bekannt sind, wird dieses schlechtere Ergebnisse liefern werden. Um Overfitting vorzubeugen, können verschiedene Maßnahmen ergriffen werden. Der Datensatz wird hierzu in zwei Teile aufgeteilt. Ein größerer Anteil der Daten wird regulär als Trainingsdatensatz eingesetzt, ein kleinerer Teil wird dem Modell im Trainingsprozess vorenthalten und als Testdatensatz genutzt. Das Modell kann so auf dem Testdatensatz überprüft werden. Diese Überprüfung kann entweder nach einer Anzahl von Epochen wiederholt durchgeführt werden oder nur ein einziges Mal beim Abschluss des Trainings.
Es ist bevorzugt, die Fehlerrate des Modells für Trainings- und Testdatensatz in einem ähnlichen Spektrum zu erhalten. Ist eine klare Differenz in der Fehlerrate zwischen den Trainings- und Testdaten erkennbar, ist dieses over- oder underfittet und das trainierte Modell ist weniger nützlich für neue unbekannte Daten und somit nicht sehr aussagekräftig für reale Anwendungsfälle. Nach dem oben beschriebenen Vorgehen ist somit die Fehlerrate im Kontext des Trainings- und Testdatensatzes das Kriterium, um über die Güte eines trainierten Modells zu entscheiden. Diese bündelt jedoch viele Kriterien aus den vorangegangenen Schritten. Der Modellzustand mit der höchsten Güte, also dem geringesten Fehler, im Kontext eines minimalen Overfittings, wird nach Abschluss des Trainings auf dem Testdatensatz geprüft. Die Testgenauigkeit des Modells liefert eine Einschätzung der allgemeinen Leistung des Modells für den Datensatz mit Ausblick auf

weitere, bisher unbekannte Daten der gleichen Art.

2.4 Maschinelles Lernen mit Zeitreihen

Machine Learning ist eine Methode, welche mit Hilfe von Algorithmen Entscheidungsprozesse errechnet und wird in drei Kategorien unterteilt. Zum einen in das Unsupervised Learning [24], welche ohne im Voraus bekannter Zielwerte durchgeführt wird. Im Gegensatz dazu steht Supervised Learning [40], welches mit vorgegebenen Zielvariablen präzise Empfehlungen kalkuliert und Prognosen macht. Und als dritte Kategorie wird das Reinforcement Learning [30] betrachtet, bei dem Softwareagenten selbstständig Strategien durch definierte Zielfunktionen erlernen.

Der Hauptfokus liegt in der vorliegenden Arbeit jedoch auf dem Supervised Learning. Hier wird mit einer Eingabe und der zugeghörigen Ausgabe, dem Label, ein Modell durch das Training entwickelt. Durch den Trainingsvorgang lernt das Modell Regeln, welche es erlauben zu den Eingaben die passenden Ausgaben zu generieren. Jedoch ist diese Ausgabe mit keiner absoluten Genauigkeit korrekt. Bei komplexen Problemstellungen kann Machine Learning als Entscheidungsprozess verwendet werden, da es mit anderen Verfahren nicht immer möglich ist eine optimale Vorhersage zu treffen. Ziel ist es jedoch nicht eine perfekt Vorhersage zu treffen, sondern die Minimierung der Fehlerrate des Modells zu erreichen. Die Eigenschaften eines Machine Learning-Modells können für den Trainingsvorgang verändert werden. So lässt sich eine Modelloptimierung mit der Veränderung der Eigenschaften, auch Hyperparameter genannt, erreichen [16]. Eine optimale Hyperparametereinstellung wird mit dem Verfahren Hyperparametertuning erreicht. Dies ist jedoch ein zeit- und ressourcenaufwendiger Prozess, welcher automatisiert stattfindet kann. Es existieren hierfür mehrere Verfahren. Eine triviale Möglichkeit ist die Grid Search. Aus einer Liste mit möglichen Konfigurationen von Hyperparametern wird jede mögliche Kombination für das Training errechnet. Für große Suchräume mit vielen Hyperparametern ist diese Suche jedoch nicht effizient, da ein hoher Rechenaufwand benötigt wird. Eine weitere Möglichkeit zur Hyperparameteroptimierung ist das Random Search-Verfahren. Hier werden wie beim Grid Search Verfahren alle möglichen Parameterkonfigurationen gelistet. Jedoch ist die Reihenfolge der ausgewählten Konfigurationen zufällig. Es konnte bereits gezeigt werden, dass dieses Verfahren in kürzerer Zeit eine zufridenstellendere Konfiguration findet [8]. Neben den bereits genannten Verfahren gibt

es noch eine Reihe weiterer Methoden wie zum Beispiel die Bayesian Hyperparameter Search oder genetische Algorithmen [41].

Der Grundbaustein vieler Machine Learning-Verfahren ist das neuronale Netz [17]. Es gibt verschiedene Varianten dieser Netze wie zum Beispiel das Convolutional Neural Network [36] oder dem Recurent Neural Network [14]. Die Verwendung einer Variante hängt vom Kontext der Anwendung ab. So werden Convolutional Neural Networks hauptsächlich in der Bilderkennung und Recurent Neural Networks bei der Verarbeitung von Sequenzen eingesetzt. Eine weitere Ausprägung von Neural Networks ist das Deep Learning [47]. Die Besonderheit dieser Netze sind eine hohe Anzahl an Hidden Layers. Eine Sonderform von Recurent Neural Networks [44] ist das LSTM [26]. Dieses Netzwerk ist darauf spezialisiert Zustandsübergänge in einem Gedächtnis zu speichern und an Folgezustände weiter zu reichen [22]. Dies wird durch die Erweiterung der Recurent Neural Networks von sogenannten Gates erreicht [25]. LSTMs lösen Klassifikationsprobleme ähnlich wie Neural Networks, jedoch verarbeiten diese keine einzelnen Features, sondern Zeitreihen.

2.5 Analyse des Datensatzes

In der Analyse des Datensatzes wird beschrieben, wie dieser erzeugt wurde und aus welchen Variablen er sich zusammen setzt. So werden die einzelnen Features betrachtet und die Anforderungen formuliert, welche für die Durchführung der Klassifikation nötig sind.

2.5.1 Datenerhebung

Der Datensatz zur Erkennung der binären Ausführungsqualität von Liegestütze wurde im Rahmen des MoGaSens Projektes an der HAW Hamburg erstellt. Der gesamte Bestand wurde über eine Reihe von Experimenten gewonnen, in denen mehrere Probanden angewiesen wurden, vorher definierte Liegestütze durchzuführen. Zur Aufnahme der Daten trugen die Probanden jeweils vier Sensoren am Körper. Jeder dieser Sensoren beinhaltete Beschleunigungssensoren und Gyroskope. Die Sensoren wurden uniform über alle Probanden am linken und rechten oberen Arm, auf der Brust und am Bauch befestigt. Das Aktivitätsprotokoll jedes Probanden bestand aus jeweils drei Sätzen bei der Datenaufzeichnung der Liegestütze. Während eines Satzes wurde von den Probanden jeweils versucht Wiederholungen von Liegestütze so lange durchzuführen, bis eine Erschöpfung

eintritt. Zwischen den Sätzen wurden Ruhepausen mit einer Dauer von fünf Minuten eingelegt. Insgesamt wurden 18 Sätze von sechs Probanden mit Hilfe der Sensoren aufgenommen und liegen als Rohdaten vor. Zusätzlich wurden Lücken im Datensatz durch Interpolation aufgefüllt. Dies war jedoch nicht Bestandteil der Arbeit und wurde vorher schon von einem MoGaSens Teammitglied durchgeführt um den temporalen Kontext des Datensatzes zu verbessern.

2.5.2 Sensoren

Jeder der vier Sensoren, welche am Körper der Probanden getragen wurden, setzt sich aus einem Beschleunigungssensor und einem Gyroskop zusammen. Das Sensormodul wurde von Bosch BMI hergestellt.

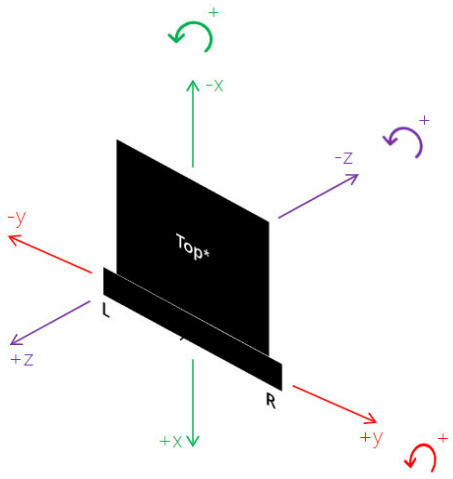

Abbildung 2.2: Freiheitsgrade: Sensormodul BMI 160

In Abbildung 2.2 sind die Freiheitsgrade des Beschleunigungssensors sowie die des Gyroskops abgebildet. In Tabelle 2.1 werden die relevanten Kanäle des Sensormoduls genauer beschrieben. Zu sehen sind die erzeugenden Datenkanäle eines Sensormuduls. Relevant für diese Arbeit sind die Daten der Beschleunigung aller drei Achsen sowie die Drehrate der Gyroskope. Der Datensatz, welcher für diese Arbeit verwendet wird, setzt sich jeweils aus vier solcher Sensoren zusammen. Das heißt die Anzahl der Spalten im Datensatz beträgt vorerst 24. Der Zeitstempel wird im Datensatz nicht betrachtet, da dieser sich

uniform um 4ms pro neu erzeugter Zeile inkrementiert. Dieser Wert würde nur linear ansteigen. Eine Synchronisation der vier Sensoren wurde bei der Datenerzeugung zusätzlich berücksichtigt. Jedoch ist nicht auszuschließen, dass die Sensoren mit einem zeitlichen Versatz Daten aufgezeichnet haben. Dies wird jedoch in dieser Arbeit vernachlässigt.

Sensorsignal	Beschreibung	Einheit
time_s	Zeitstempel	ms
xAcc	Beschleunigung in X-Richtung	$1\,\frac{\text{m}}{\text{s}^2}$
yAcc	Beschleunigung in Y-Richtung	$1\,\frac{\text{m}}{\text{s}^2}$
zAcc	Beschleunigung in Z-Richtung	$1\,\frac{\text{m}}{\text{s}^2}$
rxAcc	Gyroskop um X-Achse	$0\,^\circ/\text{s}$
ryAcc	Gyroskop um Y-Achse	$0\,^\circ/\text{s}$
rzAcc	Gyroskop um Z-Achse	$0\,^\circ/\text{s}$

Tabelle 2.1: Sensorkanäle: Bosch BMI

Zusätzlich wurde der Datensatz um zwei weitere Felder pro Sensor vom MoGaSens Team der HAW-Hamburg erweitert. Die Beschreibung erfolgt in Tabelle 2.1 . So wurde aus den drei Freiheitsgraden der Beschleunigung sowie aus den drei Freiheitsgraden des Gyroskopes jeweils ein zusätzlicher Vektor normiert. Dadurch ergeben sich acht weitere Spalten im Datensatz da diese mit der Anzahl der Sensoren multipliziert werden. Insgesamt beschreiben somit 32 Spalten den Rohdatensatz. Da bei der Datenerzeugung Sensorenwerte nicht kontinuierlich aufgezeichnet wurden war es nötig diese Datenlücken aufzufüllen, um den Charakter der unterliegenden Sportübungen nicht zu stark zu verfälschen. Dazu wurde das Verfahen der Interpolation verwendet.

Signal	Beschreibung	Einheit
normAcc	normierter Vektor Beschleunigung	$1\,\frac{\mathrm{m}}{\mathrm{s}^2}$
normGyro	normierter Vektor Gyroskop	$0\,^\circ/\mathrm{s}$

Tabelle 2.2: Ergänzung des Datensatzes

2.5.3 Anforderungen an den Datensatz

Die aufgezeichneten Rohdaten müssen nach sportwissenschaftlichen Standarts evaluiert werden. Darum werden die Daten einem Labeling unterzogen. Für das Labeln der Rohdaten wurden Kriterien von Sportwissenschaftlern vorgegeben, wann eine ausgeführte Liegestütze korrekt oder inkorrekt ausgeführt wurde. So werden in Tabelle 2.3 alle Kriterien aufgezeigt, bei denen eine Übung als nicht korrekt klassifiziert wird. Es ist jedoch zu beachten, das die fünfte Position aus Tabelle 2.3 keinen Einfluss auf die sensorischen Daten während der Ausführung hat, da die Breite der Fußstellung mit den verwendeten Sensoren nicht detektiert werden kann. Diese Fehlstellung zu erkennen wäre ausschließlich durch eine videobasierte Datenaufzeichnung möglich. Falschausführungen treten ein, wenn ein Proband den Plank nicht halten kann, also der Körper auf Hüfthöhe keinen 180° Winkel zum restlichen Körper aufweist. Des weiteren darf der Rumpfwinkel sowie der Ellenbogenwinkel keine 60° übersteigen. Die Flexion der Arme muss weniger als 90° betragen und die Handposition sollte nicht breiter oder geringer als die Schulterbreite sein.

Label	Beschreibung
nicht korrekt	Plank kann nicht gehalten werden
nicht korrekt	Rumpfwinkel/Ellenbogen übersteigt 60 Grad
nicht korrekt	In Startposition sind Arme weniger als 170 Grad gestreckt
nicht korrekt	Weniger als 90 Grad Flexion der Arme
nicht korrekt	Fußsstellung breiter/geringer als Schulterbreite
nicht korrekt	Handposition breiter/geringer als Schulterbreite

Tabelle 2.3: Repetition Asseesment (Falschausführung)

Nach den in Tabelle 2.3 festgelegten Kriterien wurde vom MoGaSens Team ein Repetition Assessement durchgeführt. So wurde jeder der 18 Sätze betrachtet und jede einzelne Liegestütze überprüft ob eine oder mehrere der in Tabelle 2.3 aufgeführten Beschreibungen auf eine Liegestütze zutrifft. Für ein maschinelles Lernverfahren muss zudem eine Übersetzung des Assessements in Integer wie in Tabelle 2.4 durchgeführt werden. Hierbei wird nun unterschieden, welches Assessement einer Übung zugewiesen wurde.

Assessement	Label
Liegestütz - korrekte Ausführung	1
Liegestütz - inkorrekte Ausführung	0

Tabelle 2.4: Abbildung der Aktivitäten auf ganzzahlige Labels.

2.6 Zielsetzung

Das Ziel dieser Arbeit ist die Implementierung einer Activity Recognition Chain, um die von den Probanden ausgeführten Liegestütze anhand vordefinierter medizinischer Maßstäbe binär nach ihrer Ausführungsqualität einordnen zu können. Mit Hilfe dieser Prozessschritte ist es möglich Datensätze mit unterschiedlichen Vorverarbeitungsschritten zu trainieren und diese bei gleichen Hyperparameterkonfigurationen gegenüber zu stellen. Dafür wird ein maschinelles Lernverfahren, das LSTM verwendet, welches sich für die Verarbeitung von Zeitreihen eignet [20]. Das LSTM soll mit den interpolierten Trainingsdaten [12] trainiert werden. Eine statische Segmentierung der temporalen Datenpunkte ist zudem nötig, da eine fest definierte Dimension für die Verarbeitung durch das LSTM Netzwerk erforderlich ist.

Sobald ein Hyperparametertuning auf dem interpolierten Datensatz durchgeführt wurde, sollen die Konfigurationen mit den stärksten Modellen dazu dienen zwei weitere Datensätze zu trainieren. Diese werden vom interpolierten Datensatz abgeleitet. Ein Datensatz wird mit einer Moving Average-Methode aus dem interpolierten Datensatz erzeugt. Ein weiterer wird aus der Anwendung eines Butterworth-Filters [46] auf dem interpolierten Datensatz gewonnen. Mit beiden werden weitere Modelle erzeugt, um eine Vergelichbarkeit aller drei Datensätze mit den gleichen Hyperparametern zu erreichen. Anschließend sollen diese drei Fälle bezüglich ihrer Performance analysiert und verglichen werden sowie eine Abschätzung über die Tauglichkeit und potenziellen Verbesserungen des Systems abgegeben werden.

3 Aufbau und Umsetzung des Experimentes

Im folgenden Kapitel wird der Aufbau und die Umsetzung der experimentellen Umgebung beschrieben. Zuerst wird der Technologiestack und die Rahmenbedingungen zusammengefasst. Darauf basierend wird gezeigt, aus welchen Komponenten sich die Toolchain zusammensetzt. Im Anschluss wird die Konstruktion der Experimente und deren Aufbau vorgestellt.

3.1 Rahmenbedingungen

Für die Umsetzung der experimentellen Umgebung wurde die Programmiersprache Python gewählt [2]. Für Python existieren eine Vielzahl von Machine Learning-Bibliotheken, mit denen die Sprache die Umsetzung von Anwendungen unterstützt. Diese Arbeit nutzt für die benötigten Machine Learning-Methoden, wie die Erstellung eines LSTMs, die Bibliothek TensorFlow [1] von Google. Für die benötigte Datentransformation vor dem Training wird unterstützend die Bibliothek numpy [7] und sklearn [4] eingesetzt, welche diverse mathematische Funktionen und Listenoperationen zur Verfügung stellen. Zur Erstellung von Graphen für die Visualisierung und Auswertung wird die Bibliothek matplotlib [6] und seaborn [5] eingebunden. Das Hyperparametertuning wird auf dem GPU Rechnencluster der HAW-Hamburg [43] durchgeführt. Hierfür wird die Anwendung gekapselt in einem Jupyter Notebook in einem Dockercontainer [15] auf dem Cluster virtualisiert und betrieben. Der Zugriff auf das Cluster wurde über einen VPN mit SSH Zugang gewährleistet.

3.2 Konstruktion der Experimente

Die Konstruktion der Experimente erfolgt nach den Prozessschritten einer Activity Recognition Chain. Diese Vorgehensweise ist an den Knoweledge Discovery in Databases Prozess (KDD) [29] angelehnt. Die experimentelle Konstruktion der Prozesssschritte, wie in Kapitel 2.1 aufgezeigt, werden im folgendem beschrieben. Im vorliegenden Kontext der Arbeit steht hier der in Kapitel 2.5 vorgestellte Datensatz. Hier wurden die Rohdaten der Beschleunigungssensoren und Gyroskope gesammelt und einer Interpolation unterzogen, um fehlende Sensorwerte auszugleichen. Ein Labeling nach der Erzeugung der Daten wurde mit einem Repetition Assesement durchgeführt. Mit den gelabelten und interpolierten Daten wird beim ersten Experiment keine Anpassung beim Preprocessing gemacht. Jeder der Liegestützsätze musste zudem gesondert berbeitet werden um Daten zu bereinigen, welche nicht zu den Übungen gehörten. In Abbildung A.1 ist solch ein Satz mit drei Parametern der Beschleunigung eines Sensors visualisiert. So wurde der Beginn und das Ende jeder Aufzeichnung abgeschnitten. Dieser Vorgang wurde manuel und für jeden Satz individuell ausgeführt um nur Daten zu erhalten, welche auch Liegestütze beinhalten. In Abbildung A.2 ist ein gekürzter Satz mit drei Signalen zu sehen und in Abbildung A.3 ein Satz mit allen visualisierten Signalen. Zusätzlich wurden die fehlerhaften Liegestützen aus jedem Satz extrahiert. Hierzu wurde mit Hilfe des Repetition Assessments eine Aufteilung der Sätze vorgenommen. So sind in Abbilung A.4 drei aufeinander folgende falschausgeführte Liegestützen zu sehen, welche durch die Aufteilung entstanden sind.

Alle korrekt ausgeführten Beispiele, sowie die Fehlerhaften wurden nach dieser Vorgehensweise erzeugt. Zusätzlich wurden alle Features einer Skalierung zwischen 0 und 1 unterzogen. Nach der Hyperparameteroptimierung mit dem interpolierten und bearbeiteten Datensatz werden auf diesem im Preprocessing weitere Operationen durchgeführt, sodass zwei weitere geglättete Datensätze erzeugt werden. Zum einen wird ein Moving Average über jeden Inputparameter des interpolierten Datensatzes berechnet. Es wird immer der Durchschnitt über fünf temporal aufeinander folgende Werte gebildet. Dies zieht eine Glättung der Kurven nach sich. Je größer der Moving Average gewählt wird, desto Glatter wird eine Kurve über den zeitlichen Verlauf. Zum anderen wird zur Datenmanipulation des interpolierten Datensatzes auf jedem Inputparameter ein Butterworth-Filter eingesetzt. Die Ordnung des Filter wurde auf fünf festgelegt. Die Sample Rate betrug 100Hz. Abbildung A.5 zeigt die Signale des Moving Average Datensatzes und Abbildung A.6 die Signale nach Anwendung des Butterworth-Filters. Es werden zusätzlich noch

weitere Operation auf dem Datensatz durchgeführt. Da der Datensatz mehr positive Beispiele als Negative enthält wird ein Downsampling der positiven Beispiele durchgeführt. Dies wird solange wiederholt bis beide Beispielklassen eine ähnliche Größe haben. Ein Upsampling wäre in diesem Konetext zu aufwendig gewesen.

Im Segmentierungsschritt der Activity Recognition Chain werden die vorverarbeiteten Daten in eine spezielle Form, wie in 2.3.3 beschrieben, transformiert. Zuerst wird eine feste Länge eines Samples definiert. Diese beträgt 1000 aufeinander folgende Datenpunkte. Um mehr Samples aus den 18 vorhanden Sätzen zu generieren wurde das Fenster mit einer größe von 1000 Datenpunkten pro Segment um jeweils einen Datenpunkt in die Zukunft verschoben. So wurden ingesamt 94712 Segmente erzeugt. Es ist zu beachten, dass hier nicht nur eine Liegestütze abgegerenzt betrachtet wird, sondern auch die Übergänge zwischen zwei ausgeführten Liegestützen. Dieser Schritt war jedoch nötig, denn sonst wäre die Datengrundlage nicht ausreichend [50]. Um durch die im Preeprocessing konkatenierten Daten nach dem Segmentieren in einer zufällige Reihenfolge anzuordnen, werden alle Samples des Datensatzes gemischt. Des weiteren wird eine Aufteilung der gesamten Daten in Trainings- und Testdaten durchgeführt. Das Aufteilungsverhältnis wurde auf 80% Trainingsdaten festgelegt. Die Testdaten betrugen 20%.

Bei der Feature Extraction wird die Anzahl der Dimensionen reduziert, die zum Beschreiben eines großen Datensatzes erforderlich sind. Bei der Analyse komplexer Daten liegt eines der Hauptprobleme in der Anzahl der beteiligten Variablen, welche beim vorliegenden Datensatz sich jedoch auf 32 beschränken. Ein Feature Extraction Vorgang wird in diesem Experiment nicht weiter betrachtet.

Die Klassifikation wird jeweils durch ein trainiertes Modell durchgeführt. Um ein aussagekräftiges Modell zu erhalten wird darum eine Hyperparameteroptimierung auf dem interpolierten Datensatz durchgeführt. Das Verfahren der Optimierung ist eine Eigenimplementierung eines Grid Search Verfahrens. Hierbei werden sequenziell alle Kombinationen der Wertebereiche aller Hyperparameter durchlaufen und jeweils ein Modell trainiert. Die besten Konfiguration bezogen auf die Testgenauigkeit des Modells werden visualisiert und diskutiert. Stehen die Konfigurationen mit den besten Ergebnissen fest, so werden mit den zusätzlich erzeugten Datensätzen, welche im Preprocessing vom verwendeten interpolierten Datensatz abgeleitet wurden, trainert. Die besten Ergebnisse aus allen drei trainierten Datensätzen mit den selben Hyperparameterkonfigurationen werden anschließend visualisiert und aufbereitet. Jedes Modell wird mit Hilfe eines Tesdatensatzes in seiner Fehlerrate überprüft. Jedes Modell wird zudem mit den errechneten

Ergebnissen und dessen Hyperparameterkonfiguration persistiert, sodass die Ergebnisse des Trainings festgehalten werden. Hierdurch soll sichergestellt werden, dass die Experimente zukünftig replizierbar sind.

3.3 Experimenteller Ablauf

Zuerst werden wie in Abbildung 3.1 mehrere LSTM Modelle auf dem interpolierten Daten trainiert. Um eine optimale Parametereinstellung zu erreichen, wird ein Hyperparametertuning eingesetz. Es wird kein Preeprocessing und keine Feature Extraction durchgeführt. Sobald eine Auswertung der optimalen Einstellungen ergeben hat welche Konfigurationen für weitere Modelle übernommen werden können, werden vom interpolierten Datensatz zwei weitere Datenreihen abgeleitet.

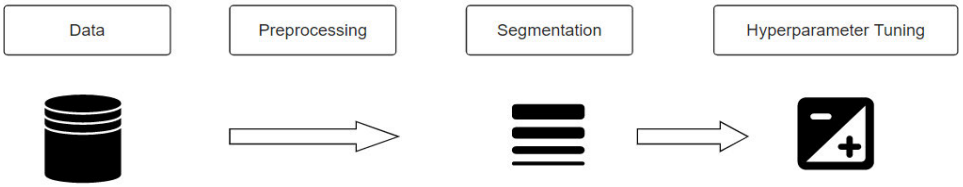

Abbildung 3.1: Prozessablauf Hyperparametertuning

In Abbildung 3.2 wird der weitere Experimentenablauf aufgezeigt. Vom interpolierten Datensatz werden im Preprocessing unter (1) und (2) ein Moving Average und ein Butterworth-Filter Datensatz [46] erzeugt. Beide Datensätze werden zudem wieder segmentiert. Eine Feature Extraction findet auch hier nicht statt. Danach werden weitere Modelle mit den extarhierten Hyperparametern aus dem Prozess aus Abbildung 3.1 für das Training der neuen Datensätze verwendet.

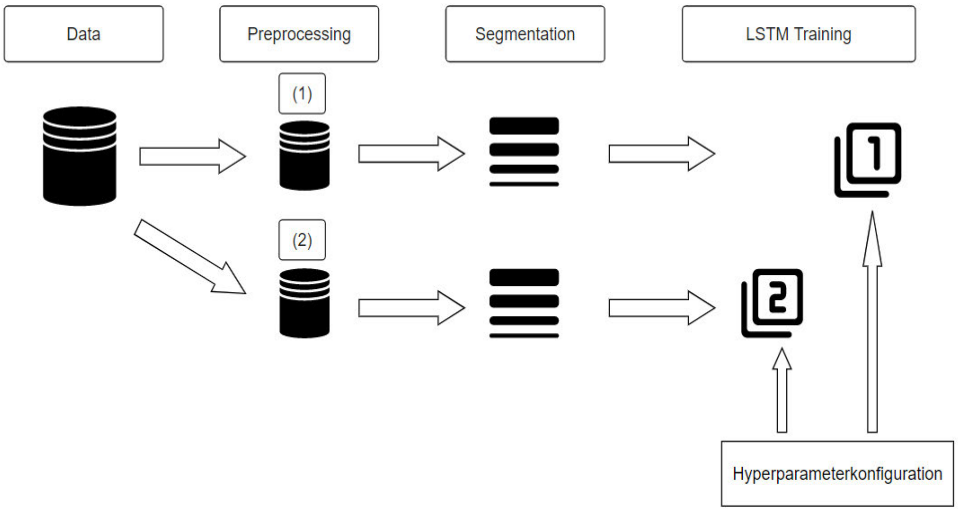

Abbildung 3.2: Prozessablauf für abgeleitete Datensätze

4 Evaluation

In Kapitel 3 wurde beschrieben, nach welchem Vorgehen die Experimente aufgebaut werden sollen. In Kapitel 4 wird die Durchführung der Experimente erläutert und die daraus entstandenen Ergebnisse evaluiert. Hierfür wird beginnend gezeigt, wie die Durchführung des in Kapitel 3 beschriebenen Experimentenaufbaus umgesetzt wird und wie der Trainingsvorgang für den LSTM Klassifikator aufgebaut wurde. Nachgelagert wird detailiert der Prozess des Hyperparameter Tunings auf dem interpolierten Datensatz beschrieben und die fünf besten Hyperparameterkonfiguration ermittelt. Aus dem interpolierten Datensatz werden dann im Preprocessing zwei weitere Datensätze erzeugt. Diese werden mit den drei besten Hyperparameterkonfigurationen, wie in Abbildung 3.2 gezeigt, aus dem Hyperparametertuning verwendet um weitere Modelle zu trainieren. Anschließend werden drei Modelle mit der höchsten Trainings und Testgenauigkeit vorgestellt und diskutiert. Eine Gegenüberstellung der Ergebnisse auf den unterschiedlichen Datensätzen folgt unter Absatz 4.3.

4.1 Durchführung der Experimente

Das Experiment unterteilt sich in drei Schritte. Im ersten Schritt wird eine geeignet Hyperparameterkonfiguration auf dem interpolierten Datensatz ohne ein Preprocessing gesucht. Die daraus hervorgegangenen Parameter mit der besten Performance werden für die darauf folgenden zwei Fälle weiter verwendet. Anschließend werden die Ergebnisse diskutiert und gegenübergestellt. Die experimentelle Umgebung wurde durch ein Jupyter Notebook Projekt aufgebaut. Dadurch konnte die komplette Umgebung mit Hilfe eines Dockercontainers virtualisiert werden. Die Anwendung wurde auf dem GPU Rechencluster der HAW Hamburg deployed. Eine Persistenz während der Durchführung ist durch Docker gegeben. Dies muss mit ensprechenden Flags beim Start der Anwendung berücksichtigt werden. Außerdem wurden alle Ergebnisse verwaltet und in .csv Dateien dokumentiert. Es wurden alle Daten zuvor mit den passenden Labels versehen, sodass

eine Klassifizierung durch das maschinelle Lernverfahren möglich ist.
Nachdem Einlesen der Inputvektoren und den zugehörigen Labels wird eine Aufteilung der Daten vorgenommen. Hier wird in Trainings- und Testdaten unterschieden. Trainingsdaten werden ausschließlich zum Training des LSTMs verwendet und Testdaten ausschließlich, um die Performance oder die Fehlerrate des trainierten Modelles zu testen. Zusätzlich kann man mit der Differenz der Ergebnisse aus den Trainings- und Testdaten darauf schließen ob ein Overfitting entstanden ist.

Das Modell besteht in der vorliegenden Arbeit aus einem sogenannten mehrschichtigen oder auch gestackted LSTM, welches direkt vom Tensorflow Framework bereitsgestellt wird. Für das Einlesen der Daten ist es nötig, dass diese in Form von dreidimensionalen Arrays an das LSTM übergeben werden. Die erste Dimension repräsentiert die Anzahl der Sampels, die zweite Dimension die Anzahl der temporalen Schritte pro Sample und die dritte Dimension beschreibt die Größe des Inputvektors pro Zeitschritt. Ein Sample selbst stellt dabei eine beliebige Anzahl an Zeitschritten dar, die während des Trainings einem Modell übergeben werden, ohne dass der Zustand dessen zurückgesetzt wird. Die Sample Size, also die Länge eines Samples ist entscheidend für das sogenannte Gedächtnis eines LSTM's. Die Labels aus dem Datensatz werden zusätzlich noch durch ein One-Hot Encoding transformiert. Die Impelmentierung des LSTMs wurde bezogen auf die Anzahl der Hidden Layers in einer dynamischen Programmierung umgesetzt. So ist es möglich, dass beliebig viele Hidden Layers in das Netzwerk eingefügt werden können. Begonnen beim Input Layer hängt die Anzahl der Neuronen von der Eingabeform des Inputvektors ab.

Um Overfitting beim Trainigsvorgang zu vermeiden werden zusätzlich nach jedem Hidden Layer ein sogenanntes Dropout Layer eingefügt. Als letztes LSTM Layer wurde ein Dense Layer eingebaut, um das Ergebnis des Modells in die korrekte binäre Labelform zu transformieren. Für die Loss Funktion des Modelles wurde eine Softmax Cross Entropy eingesetzt. Als Optimizer wurde der Adam Optimizer [33] mit dynamischer Lernrate und einer Decay Rate verwendet. Die sogenannte Decay Rate stellt hier eine Ergänzung dar, welche der Adam Optimizer bereitstellt. So wird mit Hilfe dieser die Lernrate im Verlauf einer Epoche verringert. Als weitere Variable wurde die Batch Size konfiguriert. Die Menge der Hidden Layers, die Lernrate und die Batch Size wurden als dynamische Hyperparameter festgelegt.

Der Testdatensatz für das maschinelle Lernen wurde auf 20% spezifiziert. Dieser Anteil wird dem Modell während des Traingsvorganges vorenthalten, um die Fehlerquote des

Modells zu ermitteln. Alle Samples der aufgeteilten Datensätze wurden zufällig angeordnet. Zusätzlich wurde für das Training eine Funktion eingefügt, die einen Trainingsvorgang beendet, sollte sich das Modell nach einer festgelegten Anzahl an Trainingsepochen nicht verbessern. Diese Funktion wird auch Early Stop Mechanismus genannt und dient zur zeitlichen Optimierung beim Training. Diese Grenze wurde durch Experimentieren auf 65 Epochen festgelegt. Sollte nach 65 Epochen die Performance unter 80 Prozent liegen, wird Gebrauch vom Early Stop Mechanismus gemacht. Dies wurde verwendet, da bei manchen Trainingsdurchläufen keine Verbesserung der Performance, auch nach vielen Epochen festgestellt wurde. Zusätzlich verhindert diese Funktion ein Overfitting, denn falls nach einer endlichen Zahl an Epochen keine merkliche Verbesserung mehr auftritt wird das Training auch gestoppt. Wenn ein Modell trainiert und mit dem Testdatensatz auf dessen Performance geprüft wurde hat eine Funktion die Zwischenschritte und Ergebnisse in einer .csv Datei persistiert. Des weiteren wurden Daten wie die Länge des Trainingsdurchlaufes, die Trainings- und Testgenauigkeit sowie die jeweilige Hyperparameterkonfiguration eines Modelles abgespeichert.

Um das Training mit unterschiedlichen Hyperparametern effizient zu implementieren wurde zuerst im .csv Format eine Tabelle generiert, welche alle verwendetetm Hyperparameter beinhaltet. Dazu wurden die Ranges aller Parameter und deren schrittweise Anhebung berücksichtigt. Das Verfahren nachdem die Parameter abgearbeitet werden nennt sich Grid Search und wurde mit Schleifen implementiert. Diese Methode ist jedoch nur für kleine Suchräume effizient, da alle Kombinationen an gegebenen Parametern sequenziell abgearbeitet werden. Eine weitere Methode der Optimierung ist das Random Search Verfahren. Dieses führt nachweislich zu einem effizienteren Suchen bei großen Suchräumen, jedoch ist der Implementierungsaufwand höher und findet somit keine Verwendung in der vorliegenden Arbeit.

4.1.1 Hyperparameteroptimierung

Der Suchraum wurde dynamisch von drei Hyperparametern definiert um die Fehlerquote eines Modells zu minimieren. Der erste Parameter war die Netzwerktiefe. Hierdurch wird ermöglicht, dass eine dynamisch konfigurierbare Anzahl an Hidden Layers in einem LSTM verwendet werden kann. Zusätzlich wurde als zweiter Hyperparameter die

Batch Size ausgewählt, da diese Einfluss auf die Optimierung der Trainingsgeschwindigkeit hat. Als dritter Hyperparameter wurde die Learning Rate ausgewählt. Als statische Parameter wurden die Anzah der Neuronen, Decay sowie das Dropout festgelegt. Der Dropout Parameter ist als Maßnahme gegen Overfitting zu verstehen. Die Ermittlung des Suchraumes wird also von den in Tabelle 4.1 aufgelisteten Parametern beeinflusst. Alle anderen Parameter werden statisch implementiert.

Dynamische Hyperparameter
Learning Rate
Batch Size
Hidden Layers

Tabelle 4.1: Dynamische Hyperparameter

Um bei den nachfolgenden Experimenten die selben Hyperparameterkonfiguration verwenden zu können wurden im Vorfeld die Intervalle der Hyperparameter festgelegt. Experimente von Greff et. al. [23]haben gezeigt, dass Hyperparameter bei LSTMs unabhängig zueinander sind. Hierdurch wird es möglich beliebige Konfigurationen einzelner Hyperparameter anzupassen, ohne einen negaitven Effekt in der Leistung des Netzwerkes zu erfahren. Im nachfolgenden Abschnitt werden die untersuchten Konfigurationen, welche auf dem interpolierten Datensatz aus 2.5 durchgeführt wurden, für jeden Parameter untersucht sowie deren Auswirkung auf die Performance betrachtet.

In Tabelle 4.2 wird aufgezeigt, welche Konfigurationen bezüglich der Batch Size untersucht wurden. Die linke Spalte der Tabelle zeigt die Konfigurationseinstellung des untersuchten Modells, die mittlere Spalte beschreibt die Menge der durchschnittlichen Epochen und die beiden rechten Spalten das durchschnittliche Ergebnis für die Konfigration über alle Trainingsdurchläufe beim Hyperparametertuning. Hierdurch soll eine möglichst klare Interpretation der Ergebnisse ermöglicht werden.

Die Untersuchung der Batch Size zeigte, das diese eine deutliche Auswirkung auf das Ergebnis des Modells hat. Dieses variiert je nach Parameterkonfiguration mehr oder weniger. Unter einer Batch Size von 50 bewegt sich die Trainingsperformance zwischen 87,02% und 85,63%. Jedoch nimmt die Performance ab einer Parametergröße von 10 deutlich ab. Die Ergebnisse der besten Modelle ab einer Size von 30 bewegen sich zwischen 89,47% und 91,32% für die Trainingsgenauigkeit. Zusätzlich wirkt sich eine kleinere Batch Size positiv auf die Trainingsgeschwindigkeit aus. Eine Suche größer einer Batch Size von 90 wäre im aktuellen Kontext nicht sinnvoll, da eine tendenzielle Abnahme der Performance

zu erkennen ist. Die Testperformance ist mit 84,72% 6,61% geringer als die Trainingsperformance. Dies lässt darauf schließen, dass das LSTM im Trainigsvorgang zu Overfitting neigt. Die Performance liegt ab einer Batch Size von 50 immer unter 80%.

Batch Size			
Konfig	Epochen ∅	Trainingsperformance ∅	Testperformance ∅
25	146	91,32%	84,71%
40	121	91,18%	83,19%
20	141	90,91%	83,02%
30	133	89,47%	82,35%
10	137	89,25%	81,49%
50	147	88,29%	79,34%
70	83	87,02%	77,25%
90	67	85,63%	72,78%

Tabelle 4.2: Batch Size Hyperparameter

Ein weitere Parameter, welcher untersucht wurde, war die Learning Rate. Aus Tabelle 4.3 geht hervor, dass über alle möglichen Hyperparameter die Learning Rate den größten Einfluss auf die durchschnittliche Leistung des Modells hat. So wurde bei einer Learning Rate von 0.0025 ein durchschnittlicher Höchstwert von 92,27% beim Trainingsvorgang erreicht. Weitere Ergebnisse mit einer Performance über 90% wurden in einem Intervall zwischen 0,001 und 0,01 erreicht. Wächst die Learning Rate an und übersteigt den Wert von 0.025, dann verschlechtert sich das Ergebnis im Training signifikant. Und ab einer Learning Rate von 0.1 sinkt die durchschnittliche Performance sogar unter 52,49%. Die Testperformance erreicht mit 86,56% ihren Höchstwert. Ab einer Learning Rate von 0.05 fällt sie unter 80%. Bei 0.15 sinkt diese sogar unter 33%. Wie man bei der Betrachtung der Batch Size schon feststellen konnte ist auch hier ein starkes Overfitting zu verzeichnen.

Es wurden zusätzlich statische Parametereinstellungen vorgenommen. Das waren die Anzahl der Neuronen pro Layer, die Sample Rate eines Samples, der Dropout Parameter und der Decay Parameter. Die Parameter wurden vor dem Hyperparameter Tuning bei mehreren Testdurchläufen festgelegt. So haben Voruntersuchungen gezeigt, dass die Anzahl der Neuronen pro Hidden Layer auf 150 Neuronen zufridenstellende Ergebnisse erzielt. Ist die Anzahl der Neuronen kleiner als 100 waren die Ergebnisse signifikant schlechter. Gleiches gilt für eine Neuronenanzahl von über 200. Dies führte zum Teil zu starken Einbußen in der Performance und einem Anstieg von Rechenzeit. Ein weiterer statischer Parameter ist die Sample Size. Diese wurde auf 1000 festgelegt, da im vorliegenden Da-

Learning Rate			
Konfig	Epochen ∅	Trainingsperformance ∅	Testperformance ∅
0.001	142	92,74%	86,56%
0.0025	137	92,27%	85,52%
0.005	162	92,02 %	85,09%
0.0005	147	91,76%	84,89%
0.01	214	90,55 %	83,93%
0.025	131	89,39%	81,22%
0.05	112	86,01%	77,91%
0.1	59	52,49 %	41,66%
0.15	28	45,23%	32,38%

Tabelle 4.3: Learning Rate Hyperparameter

tensatz eine Liegestütze im Schnitt 1000 Datenpunkte beinhaltet. Eine große Sample Size hat zudem Vorteile, da hierdurch das LSTM ein größeres Gedächtnis entwickelt als bei kleineren Samples. Der Dropout Parameter wurde auf 0.001 spezifiziert, da Voruntersuchungen gezeigt haben, dass kein signifikanter Unterschied in der Performance bei unterschiedlichen Dropout Belegungen festgestellt wurde. Für den Decay Parameter haben die Voruntersuchungen gezeigt, dass es auch hier keine große Abweichungen bei den Ergebnissen gibt, wenn ein Decay variabel gewählt wird. Aus diesem Grund wurde der Wert auf 0.00025 festgelegt.

Nachdem die ersten Hyperparameter betrachtet wurden ergibt sich ein finaler Suchraum in Kombination mit der Netzwerktiefe. Da es schon bei geringen Netzwerktiefen eine große Menge an möglichen Zusammenstellungen von Parameterkombination gibt, wurde die Netzwerktiefe auf 8 Hidden Layer begrenzt. Ein tieferes Netzwerk benötigt einen längeren zeitlichen Rechenaufwand pro Modell. In 4.4 ist zu sehen, dass die Anzahl von zwei sowie drei Hidden Layern die besten Ergenisse auf dem Datensatz liefert. Ab vier Hidden Layern sinkt die Performance stark ab und erreicht mit acht Hidden Layern den tiefsten Wert der Trainingsperformance von 69,79%.

Hidden Layer			
Konfig	Epochen ∅	Trainingsperformance ∅	Testperformance ∅
2	139	90,67%	82,92%
3	143	89,21%	81,84%
1	157	88,73%	81,02%
4	136	86,33%	80,31%
5	68	82,91%	73,59%
6	61	76,54%	68,44%
7	51	73,82%	64,15%
8	43	69,79%	61,28%

Tabelle 4.4: Hidden Layer Hyperparameter

4.1.2 Die besten Modelle der Optimierung

Nachdem in Abschnitt 4.1.1 zunächst der Suchraum für die weiteren Experimente aus den möglichen Hyperparameterkonfigurationen erprobt wurde, werden in diesem Teil der Arbeit die Ergebnisse der Optimierung zusammengefasst. Insgesamt wurden 9547 Modelle untersucht. In Tabelle 4.5 wird dargestellt, wie sich die Anzahl der trainierten Modelle über die Netzwerktiefe aufteilt. Es wurde bis zu einer Netzwerktiefe von maximal acht untersucht. Nachdem während des Trainings ersichtlich wurde, dass ab einer Tiefe von fünf, sich die Performance nicht verbessert, wurde die Anzahl der Modelle, welche trainiert werden sollen nochmals eingeschränkt, da der zeitliche Aufwand unnötig zunehmen würde.

Untersuchte Modelle	
Hidden Layer	Trainierte Modelle
1	72
2	216
3	648
4	1944
5	5832
6	502
7	223
8	110

Tabelle 4.5: Anzahl der trainierten Modelle

Die besten Ergebnisse der durchgeführten Optimierung werden in Tabelle 4.6 abgebildet. Die Übersicht zeigt die Modelle zu den verwendeten dynamischen Hyperparametern mit der besten Performance. Beim verwendeten interpolierten Datensatz werden die besten Ergebnisse bei einer Netzwerktiefe von drei erziehlt. Bei weniger als drei Hidden Layern ist ein leichter Abfall in der Performance zu vermerken. Gleiches gilt ab einer Netzwerktiefe von vier. Bei jeder Vergrößerung sinkt die Performance weiter. Da der Datensatz horizontal betrachtet mit 32 Inputparametern klein ist könnte dies ein Grund für den Performanceverlust bei einer höheren Netzwerktiefen sein, da nur eine geringe Netzwerkkomplexität nötig ist um die zugrunde liegenden Muster abzubilden.

Performance in %	**Parameter**		
	Batch Size	**Learning Rate**	**Hidden Layer**
93,67	25	0,0025	3
93,18	40	0,0025	3
92,74	25	0,001	2
92,02	25	0,005	2
91,76	40	0,0005	2

Tabelle 4.6: Konfigurationen mit der höchsten Trainingsgenauigkeit

Zu sehen ist, dass die besten Ergebnisse der Optimierung mit einer geringen Learning Rate von 0,0025 erziehlt werden. Des weiteren ist die Batch Size zwischen 25 und 40 die Größe mit den besten Ergebnissen aus der Optimierung. Bei der Netzwerktiefe ist zu sehen, dass diese ausschließlich mit zwei oder drei Hidden Layern vorteilhafte Ergebnisse auf dem Datensatz liefert.

Die Konfigurationen mit der höchsten Testgenauigkeit sind in Tabelle 4.7 abgebildet. Es ist ein deutlicher Unterschied im Vergleich zur Performance der Trainingsgenauigkeit zu sehen. Dies lässt wie schon beobachtet auf ein Overfitting während des Trainings schließen. Auch hier ist die Konfiguration mit einer Learning Rate von 0,0025 am besten in Bezug auf die Performance. Zudem weisen die Konfigurationen mit drei Hidden Layern auch hier darauf hin, das komplexe Features in den Hidden Layern gelernt werden.

Mit den Hyperparameterkonfigurationen der besten drei Modelle werden nun weitere Modelle auf Datensätzen, welche auf dem Urspungsdatensatz basieren, trainiert und in den Kontext eingereiht um einen Vergeleich zwischen diesen Modellen zu erfassen.

Performance in %	Parameter		
	Batch Size	Learning Rate	Hidden Layer
86,89	25	0,0025	3
85,21	40	0,0025	3
84,65	25	0,001	2
83,52	40	0,0005	2
82,97	25	0,005	2

Tabelle 4.7: Konfigurationen mit der höchsten Testgenauigkeit

4.2 Vergleich der Experimente

Mit den drei besten Hyperparameterkonfigurationen aus 4.6 werden jeweils zwei weitere Datensätze, welche auf dem Urspungsdatensatz basieren, trainiert sowie untersucht und in den Kontext eingereiht um einen Vergleich zwischen diesen Modellen und den Modellen des interpolierten Datensatzes zu diskutieren. Der interpolierte Datensatz, auf dem das Hyperparametertuning durchgeführt wurde, wird mit Hilfe der implementierten Preprocessingschritte der Activity Recogition Chain mit zwei unterschiedlichen Verfahren manipuliert. Für den ersten Datensatz wird ein Moving Average auf jeden Inputparameter angewendet. Beim zweiten Datensatz wird analog ein Butterwort-Filter verwendet. Danach werden für jeden Datensatz die drei besten Modelle bezüglich Ihrer Performance im Detail betrachtet. Dazu werden für jedes Modell diese in einer Abbildung aufgeführt. Konkret wird der Trainingsvorgang mit den Validierungsgenauigkeiten über die Epochen gelistet. So kann der Trainingsverlauf der Modelle und deren zeitliche Entwicklung über die Epochen genauer betrachtet werden. Zusätzlich wird eine Konfusionsmatrix mit einer Gegenüberstellung der Datenlabel betrachtet um das Verhältnis der Vorhersage bezüglich der Labels betrachten zu können und Stärken sowie Schwächen des Modells zu erkennen.

4.2.1 Interpolierter Datensatz

Die drei besten Modelle werden folgend in absteigender Reihenfolge bezüglich ihrer Trainings- und Testgenauigkeit aufgelistet und diskutiert sowie deren Schwächen und Stärken untereinander betrachtet. Aus diesen Ergebnissen leiten sich weitere Erkenntnisse über den gewählten Datensatz ab.

Erstes Modell

Das beste Modell mit dem interpolierten Datensatz erreichte eine Trainingsgenauigkeit von 93,67% zum ersten mal bei Epoche 27. Insgesamt wurden 168 Epochen trainiert. Der Verlauf in Tabelle 4.1 zeigt die Trainings- und Testgenauigkeit des Modells. Zu beginn sind kurze Einbrüche zu erkennen. Jedoch stabilisieren diese sich ab der Epoche 30. Zwischen Epoche 44 und 55 sinkt der Wert sinkt nochmal auf unter 80% bei der Trainingsperformance und stabilisiert sich jedoch danach wieder. Ab dann konvergiert diese gegen den Wert 93,67%. Um ein stabiles Ergebnis zu erreichen benötigt die Einstellung auf dem Modell mindestens 75 Epochen. Die Testperformance beträgt im Maximum 86,89%. Hier im Kurvenverlauf wird sichtbar, wie sich das Overfitting auf die Testgenauigkeit auswirkt. Die Performance bei den vorenthaltenen Daten liegt mit 86,89% 6,5% unter dem Maximum während des Trainings.. Dies kann durch mehrere Faktoren bedingt sein. Ein größerer Datensatz könnte gegebenenfalls abhilfe schaffen.

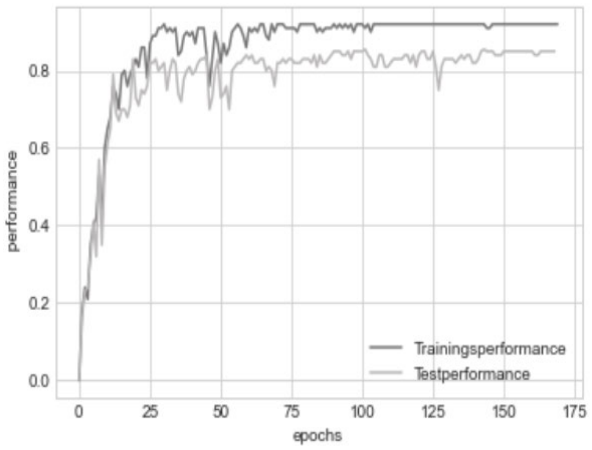

Abbildung 4.1: Trainings- und Testverlauf - interpoliert: Modell 1

Die Label der Konfusionsmatrix aus Abbildung 4.2 haben folgende Bedeutung. CORRECT steht für eine korrekt ausgeführte Liegestütze und INCORRECT für eine falsch ausgeführte. Die false negatives mit einer Anzahl von 797 sind größer als die false positives mit einer Anzahl von 743. Es werden also Übungen mit dem Label INCORRECT öfter falsch erkannt als Übungen mit dem Label CORRECT. Es ist zu beobachten, dass falsch ausgeführte Übungen öfter falsch klassifiziert werden als richtig ausgeführte Übungen. Die true positives betrugen 5223 und die true negatives 4977. Es ist ein leichtes Ungleichgewicht zu erkennen. Im Modell besteht somit eine größere Unsicherheit gegenüber falsch ausgeführten Liegestützen. Wenn man dem Datensatz zusätzlich negative Beispiele

hinzufügen würde, könnte man in zukünftigen Experimenten evaluieren, ob sich die true neagtives den true positives weiter annähern. Analog gilt das auch für die false positives und die false negatives.

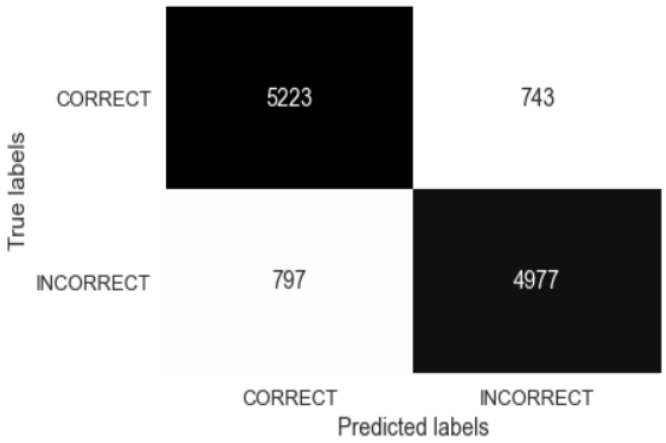

Abbildung 4.2: Heatmap: Testgenauigkeit Modell 86,89%

Zweites Modell

Das zweite Modell unter Verwendung des interpolierten Datensatzes erreichte eine Trainingsgenauigkeit von 93,18%. Insgesamt wurden 216 Epochen trainiert. Die Learning Rate betrug bei diesem Modell genau wie beim ersten Modell 0,0025. Die Anzahl der Hidden Layer war auf 3 festgelegt. Nur die Batch Size unterschied sich in der Einstellung der Hyperparameter bei diesem Modell und betrug 40.

Der Trainingsverlauf in Tabelle 4.3 zeigt den Verlauf des Modells. Zu Beginn ist ein rapider Anstieg wie beim ersten Modell in der Trainingsgenauigkeit zu erkennen. Bei Epoche 43 ist ein Einbruch zu verzeichnen, jedoch stabilisiert sich der Wert ab Epoche 55 und konvergiert gegen 93,18%. Ab Epoche 110 ist eine Stabilisierung beobachtbar. Beim besten Modell wurde diese Stabiliserung eher ereicht. Die Trainingsperformance stieg beim zweitbesten Modell zu Beginn des Trainings schneller an als beim erstbesten Modell. Das zweite Modell benötigen im Vergleich zum ersten Modell mehr Zeit zum trainieren. Die Testperformance mit 85,21% deutlich unter dem Ergebnis aus dem Training. Der Verlauf der Performance bleibt jedoch stabil im Bereich zwischen 83 und 85%.

Der Abbildung 4.4 ist zu entnehmen, dass die false negatives mit einer Anzahl von 856 kleiner als die false positives mit einer Anzahl von 874 sind. Das Ungleichgewicht zwischen

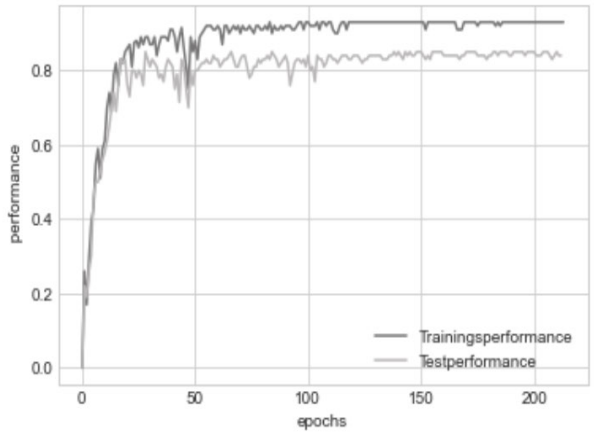

Abbildung 4.3: Trainings- und Testverlauf - interpoliert: Modell 2

den false positives und false negatives ist hier fast nicht mehr vorhanden. Bei den true positives und true negatives ist das Verhältnis wieder sehr ähnlich zum ersten Modell. Jedoch gab es hier eine ausgeglichenere Aufteilung. So betrugen die true positives 5107 und die true negatives 4901.

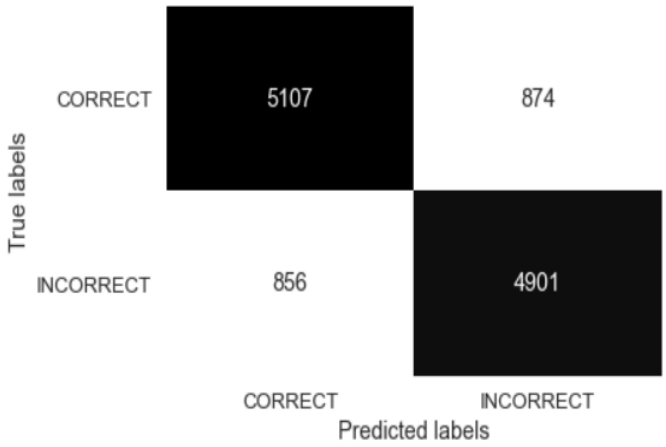

Abbildung 4.4: Heatmap: Testgenauigkeit Modell 85,21%

Drittes Modell

Das dritte Modell mit dem interpolierten Datensatz erreichte eine Trainingsgenauigkeit von 92,74%. Insgesamt wurden 163 Epochen trainiert. Die Learning Rate betrug bei diesem Modell genau wie beim besten Modell 0,001. Die Anzahl der Hidden Layer war

auf 2 festgelegt. Nur die Batch Size war wie beim ersten Modell auf 25 eingestellt. Der Trainingsverlauf in Tabelle 4.5 zeigt die Genauigkeit des Modells. Zu Beginn ist ein langsamerer Anstieg als beim ersten und zweiten Modell zu erkennen. Ab Epoche 40 ist ein kurzer Einbruch in der Genauigkeit zu verzeichnen, jedoch stabilisiert sich der Wert ab Epoche 50 wieder und konvergiert gegen 92,74%. Der Einbruch bei diesem Modell ist schwächer als der bei den ersten beiden Modellen. Eine Stabilisierung ist jedoch erst ab Epoche 100 beobachtbar. Die Testperformance lag mit 84,64% bei ihrem Höchststand. Das bisher beobachtete Overfitting ist auch hier zu erkennen.

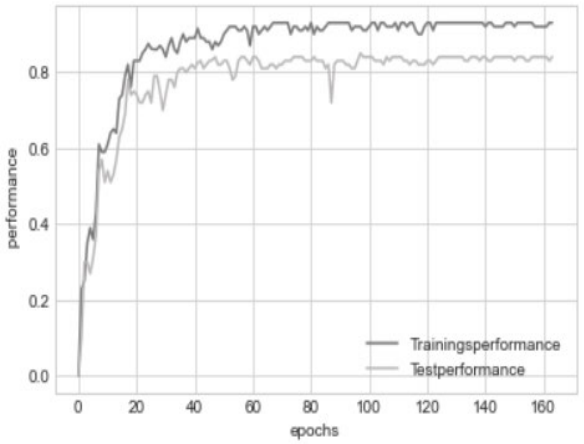

Abbildung 4.5: Trainings- und Testverlauf - interpoliert: Modell 3

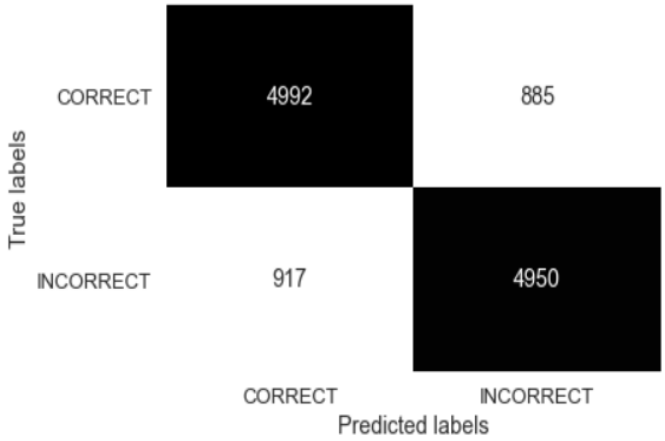

Abbildung 4.6: Heatmap: Testgenauigkeit Modell 84,64%

In Tabelle4.6 wird gezeigt, dass die false negatives mit einer Anzahl von 459 am größten ist von allen drei Modellen. Die false positives mit einer Anzahl von 393 aber weniger

zugenommen haben im Vergleich zum ersten und zweiten Modell. Außerdem ist wieder ein Ungleichgewicht zwischen den false positives und false negatives wie beim ersten und zweiten Modell zu beobachten. Bei den true positives und true negatives ist dieses Verhältnis ausgewogener. So betrugen die true positives 5495 und die true negatives 5409. Auch hier ist eine leichte Asymmetrie zu erkennen, was die Beobachtungen aus dem ersten und zweiten Modell bestätigen.

4.2.2 Moving Average Datensatz

Die drei dynamischen Hyperparameterkonfigurationen und die zwei statischen mit der höchsten Testgenauigkeit auf dem interpolierten Datensatz werden auch für den Moving Average Datensatz verwendet. Im Preprocessing wurde der interpolierte vorliegende Datensatz so manipuliert, dass jeder Inputparameter der zugrundeliegenden Daten mit einem Moving Average bearbeitet wurden. Mit diesem Datensatz und den vorhandenen Hyperparametereinstellungen der drei besten Modelle aus dem Hyperparametertuning werden analog wie beim interpolierten Datensatz neue Modelle trainiert sowie deren Ergebnisse untersucht. Auch hier werden absteigend nach ihrer Testgenauigkeit die drei Modelle diskutiert.

Erstes Modell

Das beste Modell mit dem Moving Average Datensatz erreichte eine Trainingsperformance von 93,92% ab Epoche 50. Insgesamt wurden 153 Epochen trainiert. Die Learning Rate bei diesem Training wurde auf 0.0025 festgelegt. Die Batch Size betrug 25 und die Anzahl der Hidden Layer war drei. Es ist beim Moving Average Datensatz eine leichte Verbesserung der Performance gegenüber dem interpolierten Datensatz mit den gleichen Hyperparametereinstellungen zu erkennen. Der Trainingsverlauf in Tabelle 4.7 zeigt das beste Modell mit dem Moving Average Datensatz. Zu beginn ist um die 25. Epoche ein kurzer Einbruch in der Performance zu erkennen. Dieser stabilisiert sich jedoch ab der Epoche 35 wieder. Ab Epoche 35 konvergiert der Wert gegen 93,92% und bleibt auch stabil bis auf wenige Ausreißer über die restlichen Epochen. Die Testgenauigkeit lag mit 85,02% beim Maximum, jedoch konnte dieser Wert nicht kontinuierlich erreicht werden.

Verglichen mit den Modellen des interpolierten Datensatzes zeigt Tabelle4.8, dass die false negatives mit einer Anzahl von 914 zugenommen haben verglichen zum ersten Modell

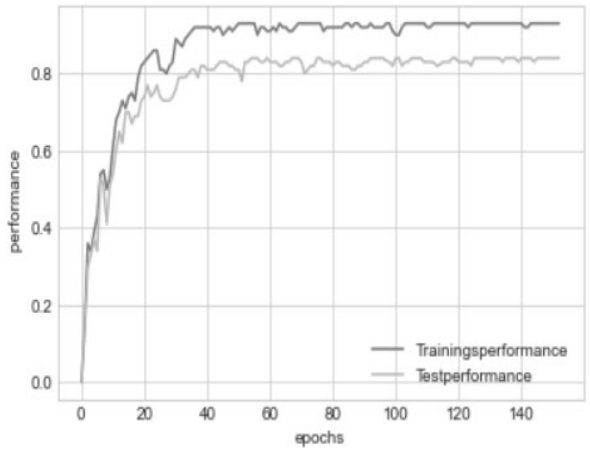

Abbildung 4.7: Trainings- und Testverlauf - Moving Average: Modell 1

des interpolierten Datensatzes. Bei den false positives hat sich diese Tendenz mit 844 auch weiter fortgesetzt. Es ist jedoch immer noch ein Ungleichgewicht zwischen den false positives und den false negatives zu erkennen. Bei den true positives und true negatives ist das Verhältnis kleiner. So betrugen die true positives 5561 und die true negatives 5471.

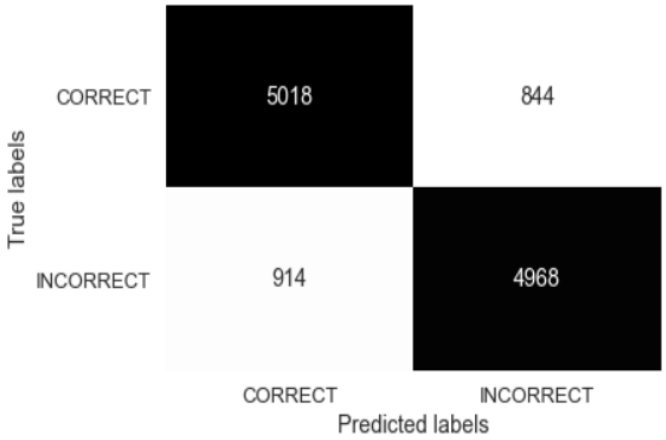

Abbildung 4.8: Heatmap: Testgenauigkeit Modell 85,02%

Zweites Modell

Das zweite Modell mit dem Moving Average Datensatz erreichte eine Trainingsgenauigkeit von 93,41% schon ab Epoche 56. Insgesamt wurden 143 Epochen trainiert. Die

Learning Rate bei diesem Modell betrug 0.001. Die Netzwerktiefe wurde auf zwei Hidden Layer festgelegt und ein Batch war 25 groß. In Abbildung 4.9 ist die Performance des zweiten Modell auf dem Moving Average Datensatz zu sehen. Zwischen der zehnten und 20. Epoche ist ersichtlich, dass eine leichte Schwankung in der Performance besteht. Jedoch normalisiert sich ab Epoche 20 diese und erreicht nach 52 Epochen das erste mal eine Genauigkeit von 93,41%. Diese stabilisiert sich im weiteren Verlauf, jedoch ist zu beobachten, dass bei Epoche 92 eine kurzer Abfall der Testgenauigkeit eingetreten ist. Dieser Abfall wird jedoch nach wenigen Epochen wieder korigiert. Das Modell benötigt 100 Epochen um konstant stabile Ergebnisse zu erzielen. Die Testperformnace lag mit 83,58% bei Epoche 60 bei ihrem Maximum. Eine Stabilisierung der Testgenauigkeit ist aber erst ab Epoche 110 zu erkennen.

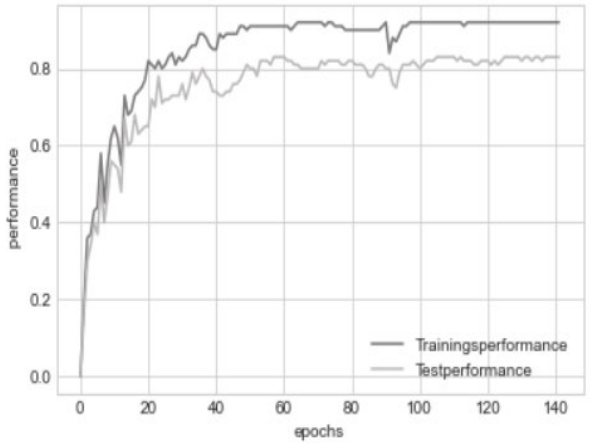

Abbildung 4.9: Trainings- und Testverlauf - Moving Average: Modell 2

Abbildung 4.10 zeigt, dass die false negatives mit einer Anzahl von 996 zum ersten Modell zugenommen haben. Bei den false positives hat sich der Wert zum ersten Modell mit 932 auch verschlechtert. Es ist jedoch immer noch ein Ungleichgewicht zwischen den false positives und den false negatives zu erkennen. Die true positives betrugen 4991 und die true negatives 4827. Signifikante Unterschiede zum ersten Modell sind jedoch nicht fest zu stellen.

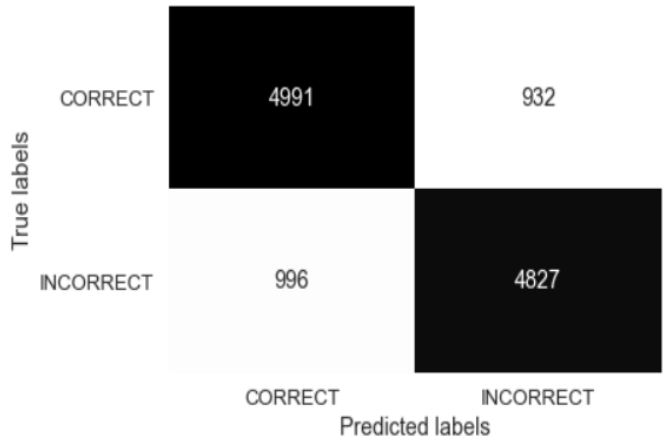

Abbildung 4.10: Heatmap: Testgenauigkeit Modell 83,58%

Drittes Modell

In Abbildung 4.11 ist die Testgenauigkeit des dritten Modells auf dem Moving Average Datensatz zu sehen. Dieses erreichte eine Trainingsgenauigkeit von 93,19% schon ab Epoche 54. Insgesamt wurden 162 Epochen trainiert. Die Learning Rate bei diesem Modell betrug 0.0025. Die Netzwerktiefe wurde auf drei Hidden Layer festgelegt und ein Batch war 40 groß. Es ist ein schneller Anstieg der Performance zu verzeichnen, jedoch wird dieser ab Epoche 17 verlangsamt und steigt erst wieder ab Epoche 24 an. Danach stabilisiert sich das Netz zunehmend ab Epoche 83. Die Testgenauigkeit hat einen maximalen Wert von 83,07

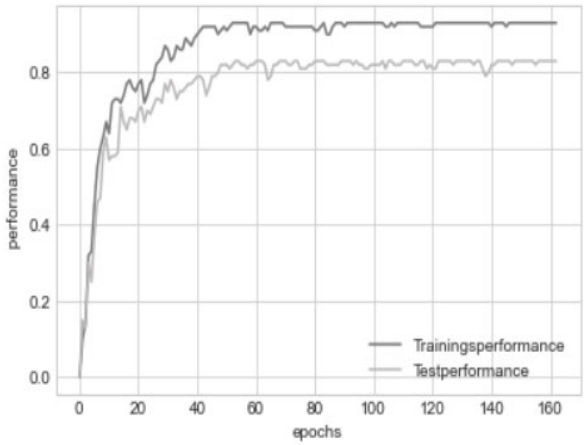

Abbildung 4.11: Trainings- und Testverlauf - Moving Average: Modell 3

Die Tabelle 4.12 zeigt, dass die false negatives mit einer Anzahl von 1077 zum zweiten Modell zugenommen haben. Bei den false positives hat sich der Wert jedoch minimal verbessert und beträgt 921. Bei den true positives und true negatives ist die Differenz geringer als beim zweiten Modell, jedoch größer als beim ersten. So betrugen die true positives 4938 und die true negatives 4830.

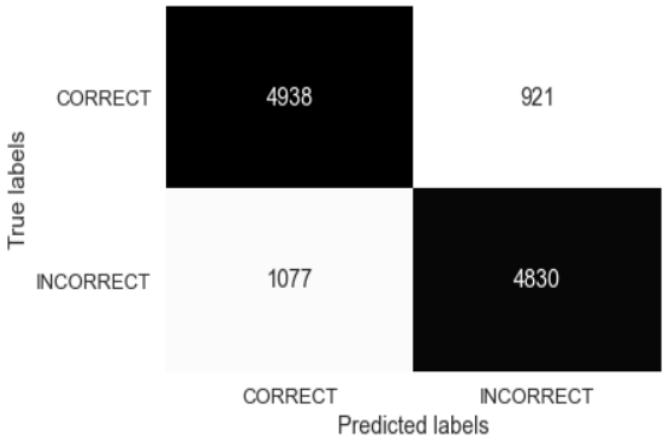

Abbildung 4.12: Heatmap: Testgenauigkeit Modell 83,07%

4.2.3 Butterworth-Filter Datensatz

Die Hyperparameterkonfigurationen mit den drei besten Testgenauigkeiten auf dem interploierten Datensatz werden nun ein zweites mal für weitere Experimente verwendet um eine Vergleichbarkeit der Modelle bezüglich ihrer Performance gegeneinander zu erreichen. Im Preprocessing wurde der interpolierte vorliegende Datensatz so manipuliert, dass jeder Inputparameter der zugrunde liegenden Daten mit einem Butterworth-Filter bearbeitet wurde. Die drei besten Modelle aus dem Hyperparameter Tuning werden für neue Modelle verwendet und deren Ergebnisse wie unter Kapitel 4.2.1 absteigend untersucht.

Erstes Modell

Das beste Modell mit dem Butterworth-Filter Datensatz erreichte eine Trainingsperformance von 93,12% bei Epoche 49. Insgesamt wurden 159 Epochen trainiert. Die Learning Rate bei diesem Training wurde auf 0.0025 festgelegt. Die Batch Size betrug 25 und die

Anzahl der Hidden Layer war 3. Es ist beim Butterworth-Filter Datensatz eine leichte Verschlechterung der Performance gegenüber dem interpolierten- und dem Moving Average Datensatz zu erkennen. Der Trainingsverlauf in Tabelle 4.13 zeigt die Testgenauigkeit des besten Modells mit dem Butterworth-Filter Datensatz. Ab Epoche 18 findet bis zur Epoche 45 ein flacher Anstieg der Performance statt. Diese stabilisiert sich jedoch ab der Epoche 46 wieder. Danach konvergiert die Testgenauigkeit gegen den Wert von 93,12% und bleibt ab Epoche 110 auch stabil. Die Testperformance lag mit 83,76% wieder deutlich unter dem Ergebnis aus dem Trainingsvorgang. Stabil wird diese erst ab Epoche 110.

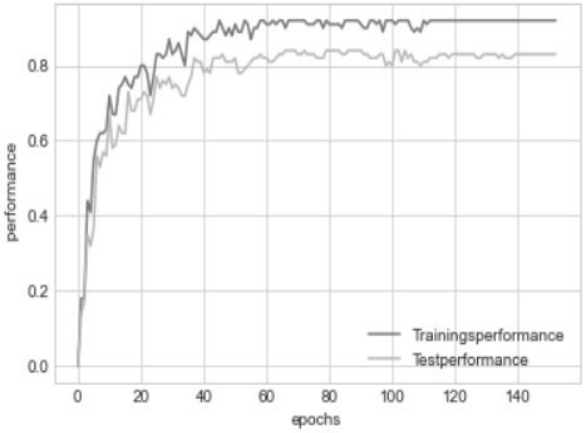

Abbildung 4.13: Trainings- und Testverlauf - Butterworth: Modell 1

In Abbildung 4.14 wird gezeigt, dass die false negatives mit einer Anzahl von 979 größer sind als bei den Ergebnissen der anderen beiden Datensätzen. Dies liegt an der allgemein schlechteren Performance. Die false positives haben analog mit einem Wert von 927 auch zugenommen. Außerdem ist wieder ein leichtes Ungleichgewicht zwischen den false positives und false negatives auffällig. Gleiches wurde bei den vorangegangen Experimenten auch beobachtet. Bei den true positives und true negatives ist keine signifikante Abweichung zu beobachten. So betrugen die true positives 4965 und die true negatives 4883.

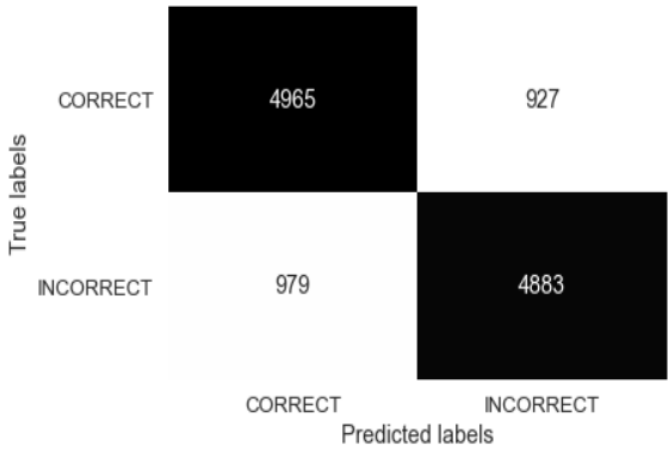

Abbildung 4.14: Heatmap: Testgenauigkeit Modell 83,76%

Zweites Modell

Das zweite Modell mit dem Butterworth-Filter Datensatz erreichte eine Trainingsgenauigkeit von 92,79% bei Epoche 54. Insgesamt wurden 171 Epochen trainiert. Die Learning Rate bei diesem Training wurde auf 0.0025 festgelegt. Die Batch Size betrug 40 und die Anzahl der Hidden Layer war 3. Der Trainingsverlauf in Tabelle 4.15 zeigt die Performance des zweiten Modells mit dem Butterworth-Filter Datensatz. Um Epoche 30 verlangsamt sich der Anstieg der Genauigkeit. Diese stabilisiert sich jedoch ab der Epoche 40 wieder. Ein Einbruch der Performance ist bei Epoche 55 zu beobachten. Danach konvergiert die Trainingsgenauigkeit gegen den Wert 92,79%. Die Testperformance erreichte ein Maximum von 82,89% und war ab Epoche 60 weitestgehend stabil.

Tabelle 4.16 zeigt, dass die false negatives mit einer Anzahl von 1031 mehr als beim ersten Modell sind. Die false positives haben zum ersten Modell mit 977 auch zugenommen. Außerdem ist wieder ein Ungleichgewicht zwischen den false positives und false negatives wie beim ersten Modell zu beobachten. Bei den true positives und true negatives ist dieses Verhältnis sehr ähnlich zum ersten Modell. So betrugen die true positives 4920 und die true negatives 4816.

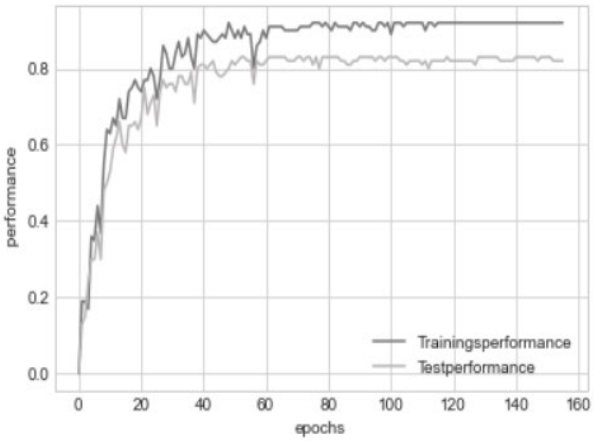

Abbildung 4.15: Trainings- und Testverlauf - Butterworth: Modell 2

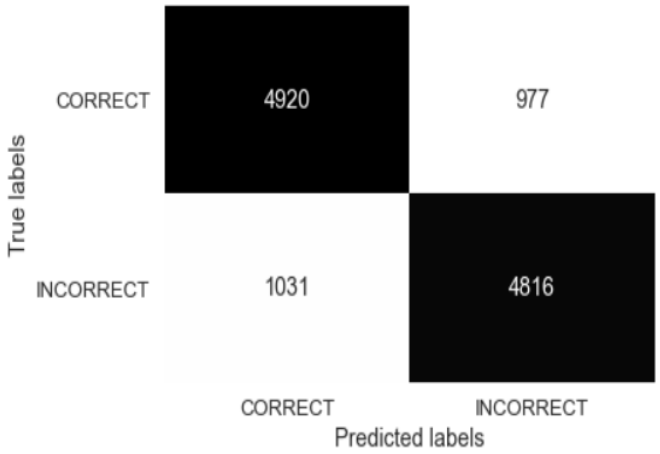

Abbildung 4.16: Heatmap: Testgenauigkeit Modell 82,89%

Drittes Modell

Das dritte Modell mit dem Butterworth-Filter Datensatz erreichte eine Testgenauigkeit von 92,45% bei Epoche 50. Insgesamt wurden 137 Epochen trainiert. Die Learning Rate bei diesem Training wurde auf 0.001 festgelegt. Die Batch Size betrug 25 und die Anzahl der Hidden Layer war auf 2 festgelegt. Der Trainingsverlauf in Tabelle 4.17 zeigt die Testgenauigkeit des dritten Modells mit dem Butterworth-Filter Datensatz. Ab Epoche 20 verlangsamt sich das Modell nach einem Anstieg bis zur Epoche 33. Danach ist bei Epoche 42 ein Einbruch in der Genauigkeit zu verzeichnen, welcher sich jedoch nach wenigen Epochen wieder normalisiert und danach sich dem Wert 92,45% annähert. Ab

Epoche 100 ist das Modell stabil und zeigt keine weiteren Abweichungen. Die Testperformance des Modells lag mit 81,26% deutlich unter der Trainingsperformnace. Jedoch verhielt sich diese ab Epoche 45 schon stabil.

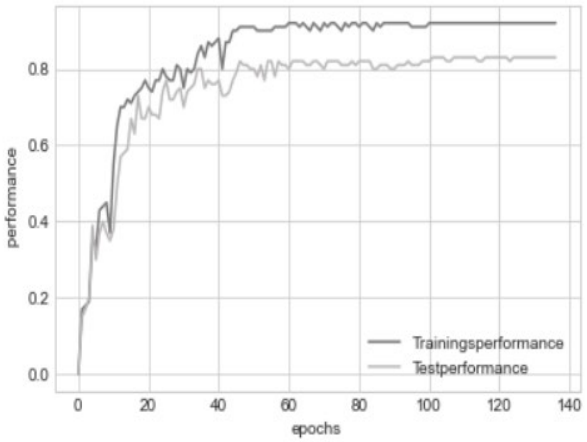

Abbildung 4.17: Trainings- und Testverlauf - Butterworth: Modell 3

In Tabelle4.18 wird gezeigt, dass die false negatives mit einer Anzahl von 1174 das schlechteste Modell ist. Die false positives haben zum ersten Modell deutlich zugenommen. Das bisher beobachtete Ungleichgewicht zwischen den false positives und false negatives bleibt weiter bestehen. Bei den true positives und true negatives ist dieses Verhältnis sehr ähnlich zu den anderen Modellen. So betrugen die true positives 4841 und die true negatives 4713.

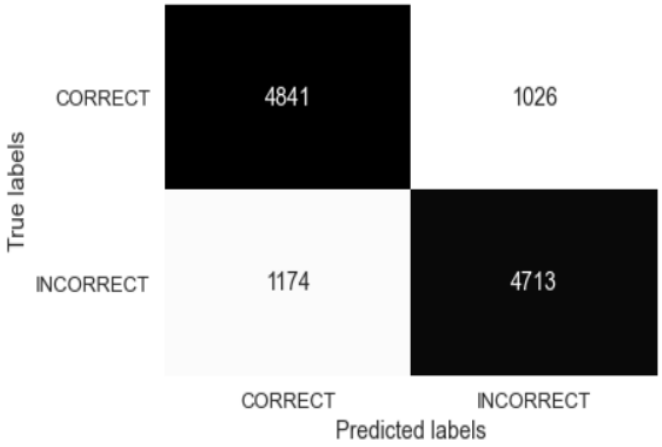

Abbildung 4.18: Heatmap: Testgenauigkeit Modell 81,26%

4.3 Auswertung

Nach der Vorstellung der einzelnen Modelle aus den Experimenten der vorliegenden Arbeit werden diese in 4.8 nun ausgewertet. Das beste Modell des interpolierten Datensatzes mit einer Trainingsgenauigkeit von 93,67% und einer Testgenauigkeit von 86,89% ist in der Gesamtbetrachtung über alle drei Datensätze auf Platz zwei. Die Hyperparameterkonfiguration war bei diesem Modell wie folgt definiert. Die Anzahl der Hidden Layer war drei, die Learning Rate 0,0025 und die Batch size 25.

Das Modell mit der höchsten Performance auf dem Moving Average Datensatz hat über alle Modelle das beste Ergebnis im Training erziehlt. Die Performance lag bei 93,92%. Dies ist jedoch nur eine Verbesserung um 0,2% zum besten Modell des interpolierten Datensatzes. Die Testperformance lag hier bei 85,02%. Dies ist in der Testperformance ein schlechterer Wert als beim interpolierten Datensatz.

Das beste Modell, welches auf dem Butterworth-Filter Datensatz entwickelt wurde, wies eine Performance von 93,12% auf. Dies ist im Vergleich zu den anderen Datensätzen das schlechteste Modell. Die Trainingsperformance ist zum interpolierten Datensatz um fast 0,6% und zum Moving Average Datensatz um 0,8% schlechter. Das Ergebnis war überraschend, da davon ausgegangen wurde, dass durch die Manipulation des interpolierten Datensatzes eine Steigerung der Performance erzielt werden kann. Jedoch muss die Verarbeitung der Daten mit dem Butterworth-Filter zu einem Informationsverlust geführt haben. Da der Konfigurationsraum der Parameter eines Butterworth-Filters viele Möglichkeiten bietet Kurven zu bearbeiten wurde wahrscheinlich eine unvorteilhafte Konfiguration ausgewählt. Abschließend hat sich jedoch gezeigt, dass die Performance der Modelle untereinander auf den drei Datensätzen nur minimale Abweichungen aufweisen. Der Unterschied der Performance während des Trainings beträgt 0,8% vom besten zum schlechtesten Modell.

Trainingsperformance in %	Datensatz
93,92	Moving Average Datensatz
93,67	Interpolierter Datensatz
93,12	Butterworth-Filter Datensatz

Tabelle 4.8: Ergebnisse der besten Modelle über alle Datensätze im Training

In Tabelle 4.9 wird die Testgenauigkeit der besten Modelle auf den Datensätzen abgebildet. Es ist zu erkennen, dass der interpolierte Datensatz mit 86,89% die höchste

Genauigkeit besitzt. Das beste Modell auf dem Moving Average Datensatz erreichte eine Performance von 85,02% und das Modell, welches den Butterworth-Filter Datensatz verwendete, war eine Genauigkeit von 83,76% festzustellen.

Testperformance in %	Datensatz
86,89	Interpolierter Datensatz
85,02	Moving Average Datensatz
83,76	Butterworth-Filter Datensatz

Tabelle 4.9: Ergebnisse der besten Modelle über alle Datensätze im Test

4.4 Fazit

Die durchgeführten Experimente haben gezeigt, dass die Verwendung von LSTMs durchaus für den Einsatz zur Erkennung der Ausführungsqualität von Sportübungen auf den verwendeten Datensätzen eingesetzt werden können. Hier ist speziell die Betrachtung des zeitlichen Kontextes der Datenreihen relevant, da dieser einen positiven Einfluss auf die Leistung des Modells hat. Die entwickelten Modelle besitzen jedoch klar abgrenzbare Einschränkungen bezüglich der Fehlertoleranz sowie der Benutzbarkeit. So wurde auf dem generierten Rohdatensatz zuerst eine aufwendige Interpolation durchgeführt um fehlende Werte im Datensatz auszugleichen, sodass keine größeren Lücken in den Datenreihen entstehen.

Die Modellarchitektur besitzt zudem auch keinen Feature Extraction Schritt in der Activity Recognition Chain. Würden zusätzlich aussagekräftige Features aus den vorhandenen Daten generiert, wäre es durchaus möglich eine höhere Genauigkeit zu erreichen. Dies könnte bei Echtzeitanwendungen jedoch aufwendige Berechnungen nach sich ziehen, welche auf mobilen Endgeräten oder Wearables ressourcensparsam sein sollten.

Eine weitere Einschränkung, welche aus zeitlichen Gründen bei der Hyperparameter Optimierung erfolgte, war die Festlegung von statischen Parametern wie dem Decay, der Sample Size, der Neuronenanzahl pro Layer und der Dropout Rate. Um die Erkennungsrate zu maximieren könnte der Suchraum erweitert werden, indem den statischen Parametern ein dynamisches Intervall zugewiesen wird. Zusätzlich könnten die Suchräume mit fünf oder mehr Hidden Layern noch komplett untersucht werden. Dies wäre jedoch sehr Ressourcenintensiv.

Eine schnellere Berechnung der Modelle könnte zusätzlich durch ein anderes Entscheidungsverfahren beim Hyperparameter Tuning erziehlt werden, da in dieser Arbeit eine triviale Grid Search implementiert wurde. So könnte man auf das Random Search Verfahren zurückgreifen, welches zufällig Parameterkonfigurationen auswählt.

Der wohl gravierendste Einflussfaktor auf die Modelle ist übergreifend das Overfitting. Es konnte beginnend ab dem Hyperparameter Tuning bis hin zum Training einzelner Modell kein Trainingsvorgang durchgeführt werden, welcher kein oder nur geringes Overfitting aufweist. Zurückzuführen ist dies auf die geringe Anzahl an Trainingsdaten. Dies lag zum einen an den begrenzt verfügbaren Testprobanden, welche die Sportübungen durchgeführt haben und zum anderem an dem starken Ungleichgewicht von positiven zu negativen Beispielen. Bei einem aufgezeichneten Satz Liegestützen wurden bei allen 18 betrachteten Sätzen nur wenige Übungen falsch ausgeführt. Um das Verhältnis von positiven und negativen Beispielen auszugleichen wurden die positiven Beispiele downgesampled, was eine signifikante Datenreduzierung nach sich zog. Um dem entgegenzuwirken kämen zwei Möglichkeiten in Betracht. Zum einen könnte man die negativen Beispiele upsamplen oder der Datenbestand würde durch Testprobanden weiter vergrößert werden.

5 Schluss

Im letzten Kapitel werden die gewonnenen Erkenntnisse der Arbeit zusammengefasst und ein Ausblick gegeben, welche Forschungsansätze künftig auf dieser Arbeit aufbauen könnten.

5.1 Zusammenfassung

Die vorliegende Arbeit hatte zum Ziel ein Klassifikationsproblem mit Hilfe von LSTMs nach dem Prozess der Acitivity Recognition Chain auf ähnlichen Datensätzen zu lösen. Die erstellten Modelle sollten in der Lage sein die Aktivitätsklassen der Datensätze möglichst genau zu erkennen. Dazu wurde in Kapitel 2 zuerst alle grundlegenden Begriffe sowie Vorgehensweisen erklärt. Der zugrundeliegene interpolierte Datensatz wurde zudem vorgestellt.

In Kapitel 3 wurde der allgemeine Aufbau und die Umsetzung der Experimente beschrieben. Zuerst wurde auf die verwendete Toolchain eingegangen, danach die Konstruktion des Experimentes erläutert. Hier kam der Prozess der Activity Recognition Chain zum Einsatz, ähnlich dem KDD Prozess. Zusätzlich wurde der genaue Ablauf der Experimente mit unterschiedlichen Vorverarbeitungsschritten dargestellt. Danach wurde der Datensatz segmentiert und für die Verarbeitung im LSTM vorbereitet.

Kapitel 4 beschrieb die Umsetzung und Evaluation der Experimente. So wurde eine Hyperparameteroptimierung, die Grid Search, implementiert um eine geeignete Konfiguration für Modelle zu suchen. Mit Hilfe der Optimierung wurden die fünf besten Modelle auf dem interpolierten Datensatz ausgewertet. Aus den Konfigurationen mit der höchsten Performance wurden die drei erfolgreichsten Konfigurationen ausgewählt und für weitere Experimente verwendet. Der interpolierte Datensatz wurde zudem in zwei zusätzliche Datensätze transformiert. Zum einen mit dem Moving Average Verfahren und zum anderen

mit einem Butterworth-Filter. Mit Hilfe der drei erfolgreichsten Hyperparameterkonfigurationen aus dem interpolierten Datensatz wurden dies Konfigurationen mit den beiden Datensätzen trainiert. Danach wurden die drei besten Modelle jedes Datensatzes vorgestellt und diskutiert. Die Ergebnisse der zusammengefassten Experimente haben gezeigt, dass die Modelle die Fähigkeit besitzen unterschiedliche Kategorien an dynamischen Aktivitätsklassen zu erkennen. Jedoch wurde ersichtlich, dass aufgrund von wenigen verfügbaren Daten die Leistung der Modelle noch gesteigert werden könnte. Overfitting hat jeden Trainingsdurchlauf stark beeinflusst. Die besten Modelle über alle Datensätze wurden zudem in der Auswertung zusammengefasst. Das beste Modell, welches im Zuge der Eperimente errechnet wurde besitzt eine Trainingsgenauigkeit von 93,92%. Damit konnte gezeigt werden, dass LSTMs eine durchaus valide Methode bei der Erkennung der Ausführungsqualität von Sportübungen sind. Danach wurden im Fazit Optionen aufgezeigt, wie Experimente, die auf dieser Arbeit aufbauen, gestaltet werden können um weitere Verbesserung der Modelle zu erreichen.

5.2 Ausblick

Die Ergebnisse dieser Arbeit haben gezeigt, dass für den verwendeten interpolierten Datensatz und dessen Variationen eine Unterscheidung der Ausführungsqualität von Sportübungen mit Hilfe von LSTMs möglich ist. Die trainierten Modelle können die unterschiedlichen Aktivitätslabel erkennen, was vermuten lässt, dass auch komplexere Problemstellungen aus dem Bereich der Human Activity Recognition mit diesem Ansatz gelöst werden können. Generel wachsen die Anwendungsfälle und Geschäftsmodelle von Deep Learning Methoden. So sind bezüglich des Quantified Selfs diverse Gesundheits- und Fittness Apps bereits in einem stark wachsenden Markt vertreten.

Der Einsatz solcher Applikationen ist zudem Bestandteil einer regen Diskussion, da sich im Kontext der Datenerhebung bei Privatpersonen für Geschäftstreibende neue Verantwortlichkeiten herausbilden. So müssen gemeinsame Standarts und Regeln beim Einsatz der Anwendungen berücksichtigt werden, sodass Rechte von Verbrauchern im Rahmen der Produktnutzung geschützt werden.

Literaturverzeichnis

[1] Google Developers. . – URL https://www.tensorflow.org/api_docs. – Zuletzt aufgerufen: 06.09.2021

[2] Python 3.9.5 Documentation. . – URL https://www.python.org/downloads/release/python-395/. – Zuletzt aufgerufen: 06.09.2021

[3] MoGaSens. (2019). – URL https://csti.haw-hamburg.de/project/mogasens/. – Zuletzt aufgerufen: 28.08.2021

[4] API Reference. (2020). – URL https://scikit-learn.org/stable/modules/classes.html. – Zuletzt aufgerufen: 21.08.2021

[5] API reference. (2021), 8. – URL https://seaborn.pydata.org/api.html. – Zuletzt aufgerufen: 17.08.2021

[6] Overview. (2021), 8. – URL https://matplotlib.org/stable/contents.html. – Zuletzt aufgerufen: 14.08.2021

[7] What is NumPy? (2021), 6. – URL https://numpy.org/doc/stable/user/whatisnumpy.html. – Zuletzt aufgerufen: 21.08.2021

[8] Bergstra, James ; Bengio, Yoshua: Random Search for Hyper-Parameter Optimization. In: *Journal of Machine Learning Research* 13 (2012), Nr. 10, S. 281–305. – URL http://jmlr.org/papers/v13/bergstra12a.html

[9] Brown, Terry: 5 Big Data Predictions. (2018), 12. – URL https://itchronicles.com/big-data/5-big-data-predictions-for-2019/. – Zuletzt aufgerufen: 21.08.2021

[10] Brownlee, Jason: How to One Hot Encode Sequence Data in Python. (2019), 12. – URL https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/. – Zuletzt aufgerufen: 23.08.2021

[11] Bulling, Andreas ; Blanke, Ulf ; Schiele, Bernt: A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors. In: *ACM Computing Surveys* 46 (2013), 01

[12] Caruso, C. ; Quarta, F.: Interpolation methods comparison. In: *Computers Mathematics with Applications* 35 (1998), Nr. 12, S. 109–126. – URL https://www.sciencedirect.com/science/article/pii/S0898122198001011. – ISSN 0898-1221

[13] Chen, Jun ; Samuel, R. Dinesh J. ; Poovendran, Parthasarathy: LSTM with bio inspired algorithm for action recognition in sports videos. In: *Image and Vision Computing* 112 (2021), S. 104214. – URL https://www.sciencedirect.com/science/article/pii/S0262885621001190. – ISSN 0262-8856

[14] Chung, Junyoung ; Gulcehre, Caglar ; Cho, KyungHyun ; Bengio, Yoshua: *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* 2014

[15] Cito, Jürgen ; Ferme, Vincenzo ; Gall, Harald: Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research, 06 2016, S. 609–612. – ISBN 978-3-319-38790-1

[16] Claesen, Marc ; Moor, Bart D.: *Hyperparameter Search in Machine Learning.* 2015

[17] Demuth, Howard B. ; Beale, Mark H. ; De Jess, Orlando ; Hagan, Martin T.: *Neural Network Design.* 2nd. Stillwater, OK, USA : Martin Hagan, 2014. – ISBN 0971732116

[18] Dietterich, Tom: Overfitting and Undercomputing in Machine Learning. In: *ACM Comput. Surv.* 27 (1995), September, Nr. 3, S. 326–327. – URL https://doi.org/10.1145/212094.212114. – ISSN 0360-0300

[19] Garcia-Gonzalez, Daniel ; Rivero, Daniel ; Fernandez-Blanco, Enrique ; Luaces, Miguel R.: A Public Domain Dataset for Real-Life Human Activity Recognition Using Smartphone Sensors. In: *Sensors* 20 (2020), Nr. 8. – URL https://www.mdpi.com/1424-8220/20/8/2200. – ISSN 1424-8220

[20] Gers, F.A.: Learning to forget: continual prediction with LSTM. In: *IET Conference Proceedings* (1999), January, S. 850–855(5). – URL https://digital-library.theiet.org/content/conferences/10.1049/cp_19991218

[21] Goddard, William: The New Gold Rush for Businesses. (2019), 4. – URL https://itchronicles.com/technology/data-the-new-gold-rush-for-businesses/. – Zuletzt aufgerufen: 02.08.2021

[22] Graves, Alex: *Generating Sequences With Recurrent Neural Networks*. 2014

[23] Greff, Klaus ; Srivastava, Rupesh K. ; Koutník, Jan ; Steunebrink, Bas R. ; Schmidhuber, Jürgen: LSTM: A Search Space Odyssey. In: *IEEE Transactions on Neural Networks and Learning Systems* 28 (2017), Nr. 10, S. 2222–2232

[24] Hinton, Geoffrey ; Sejnowski, Terrence J.: *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 05 1999. – URL https://doi.org/10.7551/mitpress/7011.001.0001. – ISBN 9780262288033

[25] Hochreiter, Sepp ; Schmidhuber, Jürgen: LSTM Can Solve Hard Long Time Lag Problems. In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*. Cambridge, MA, USA : MIT Press, 1996 (NIPS'96), S. 473–479

[26] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-term Memory. In: *Neural computation* 9 (1997), 12, S. 1735–80

[27] Hsu, Yu-Liang ; Yang, Shih-Chin ; Chang, Hsing-Cheng ; Lai, Hung-Che: Human daily and sport activity recognition using a wearable inertial sensor network. In: *IEEE Access* 6 (2018), S. 31715–31728

[28] Hülsmann, Felix ; Göpfert, Jan P. ; Hammer, Barbara ; Kopp, Stefan ; Botsch, Mario: Classification of motor errors to provide real-time feedback for sports coaching in virtual reality — A case study in squats and Tai Chi pushes. In: *Computers Graphics* 76 (2018), S. 47–59. – URL https://www.sciencedirect.com/science/article/pii/S0097849318301304. – ISSN 0097-8493

[29] Imberman, Susan: Effective Use of the KDD Process and Data Mining for Computer Performance Professionals., 01 2001, S. 611–620

[30] Kaelbling, Leslie P. ; Littman, Michael L. ; Moore, Andrew W.: Reinforcement Learning: A Survey. In: *J. Artif. Int. Res.* 4 (1996), Mai, Nr. 1, S. 237–285. – ISSN 1076-9757

[31] Karantonis, Dean ; Narayanan, Michael ; Mathie, Merryn ; Lovell, Nigel ; Celler, B.G.: Implementation of a Real-Time Human Movement Classifier Using a Triaxial Accelerometer for Ambulatory Monitoring. In: *Information Technology in Biomedicine, IEEE Transactions on* 10 (2006), 02, S. 156 – 167

[32] Ke, Shian-Ru ; Thuc, Hoang Le U. ; Lee, Yong-Jin ; Hwang, Jenq-Neng ; Yoo, Jang-Hee ; Choi, Kyoung-Ho: A Review on Video-Based Human Activity Recognition. In: *Computers* 2 (2013), Nr. 2, S. 88–131. – URL https://www.mdpi.com/2073-431X/2/2/88. – ISSN 2073-431X

[33] Kingma, Diederik P. ; Ba, Jimmy: *Adam: A Method for Stochastic Optimization.* 2017

[34] Kotsiantis, Sotiris: Supervised Machine Learning: A Review of Classification Techniques. In: *Informatica (Ljubljana)* 31 (2007), 10

[35] Lara, Oscar D. ; Labrador, Miguel A.: A Survey on Human Activity Recognition using Wearable Sensors. In: *IEEE Communications Surveys Tutorials* 15 (2013), Nr. 3, S. 1192–1209

[36] LeCun, Yann ; Kavukcuoglu, Koray ; Farabet, Clement: Convolutional networks and applications in vision. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, S. 253–256

[37] Lee, Jaehyun ; Joo, Hyosung ; Lee, Junglyeon ; Chee, Youngjoon: Automatic Classification of Squat Posture Using Inertial Sensors: Deep Learning Approach. In: *Sensors* 20 (2020), Nr. 2. – URL https://www.mdpi.com/1424-8220/20/2/361. – ISSN 1424-8220

[38] Lim, Se-Min ; Oh, Hyeong-Cheol ; Kim, Jaein ; Lee, Juwon ; Park, Jooyoung: LSTM-Guided Coaching Assistant for Table Tennis Practice. In: *Sensors* 18 (2018), Nr. 12. – URL https://www.mdpi.com/1424-8220/18/12/4112. – ISSN 1424-8220

[39] Lukowicz, Paul ; Ward, Jamie A. ; Junker, Holger ; Stäger, Mathias ; Tröster, Gerhard ; Atrash, Amin ; Starner, Thad: Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. In: Ferscha, Alois (Hrsg.) ; Mattern, Friedemann (Hrsg.): *Pervasive Computing.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2004, S. 18–32. – ISBN 978-3-540-24646-6

[40] Maglogiannis, Ilias G.: *Emerging artificial intelligence applications in computer engineering: real word ai systems with applications in ehealth, hci, information retrieval and pervasive technologies.* Bd. 160. Ios Press, 2007. – 3–24 S

[41] Mockus, J. B. ; Mockus, L. J.: Bayesian Approach to Global Optimization and Application to Multiobjective and Constrained Problems. In: *J. Optim. Theory Appl.* 70 (1991), Juli, Nr. 1, S. 157–172. – ISSN 0022-3239

[42] Nebeling, Michael ; Madier, Katy: 360proto: Making Interactive Virtual Reality Augmented Reality Prototypes from Paper. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA : Association for Computing Machinery, 2019 (CHI '19), S. 1–13. – URL https://doi.org/10.1145/3290605.3300826. – ISBN 9781450359702

[43] Nietsche, Matthias: Towards German Abstractive Text Summarization using Deep Learning. (2019), 08. – URL https://users.informatik.haw-hamburg.de/~ubicomp/papers.html. – Zuletzt aufgerufen: 07.09.2021

[44] Nils, Y. H.: *Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables*. arxiv, 2016. – URL https://arxiv.org/pdf/1604.08880.pdf

[45] Poppe, Ronald: Vision-based human motion analysis: An overview. In: *Computer Vision and Image Understanding* 108 (2007), 10, S. 4–18

[46] Sandru, Florin-Daniel ; Nanu, Sorin ; Silea, Ioan ; Miclea, Razvan-Catalin: Kalman and Butterworth filtering for GNSS/INS data. In: *2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC)*, 2016, S. 257–260

[47] Schmidhuber, Jürgen: Deep learning in neural networks: An overview. In: *Neural Networks* 61 (2015), Jan, S. 85–117. – URL http://dx.doi.org/10.1016/j.neunet.2014.09.003. – ISSN 0893-6080

[48] Swan, Melanie: The Quantified Self: Fundamental Disruption in Big Data Science and Biological Discovery. In: *Big Data* 1 (2013), Nr. 2, S. 85–99. – URL https://doi.org/10.1089/big.2012.0002. – PMID: 27442063

[49] Yu, Tao ; Chen, Jianxin ; Yan, Na ; Liu, Xipeng: A Multi-Layer Parallel LSTM Network for Human Activity Recognition with Smartphone Sensors. In: *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2018, S. 1–6

[50] Zhang, Ying: A strategy to apply machine learning to small datasets in materials science. (2018)

A Anhang

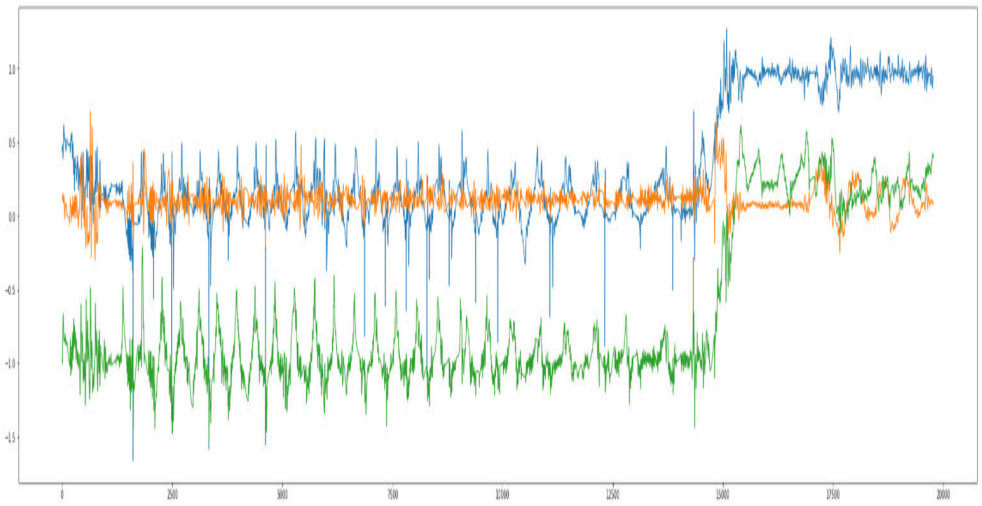

Abbildung A.1: Drei Beschleunigungssignale eines ungekürzten Satzes

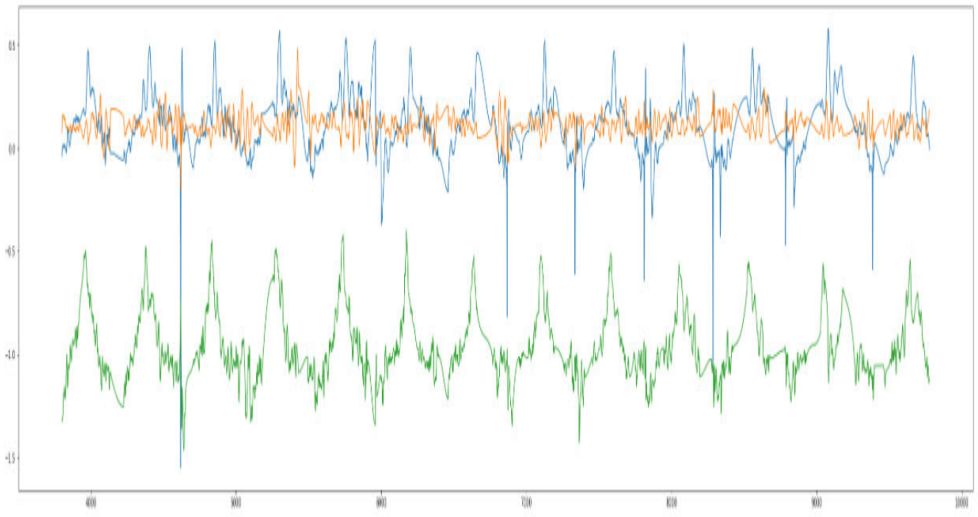

Abbildung A.2: Drei Beschleunigungssignale eines gekürzten Satzes

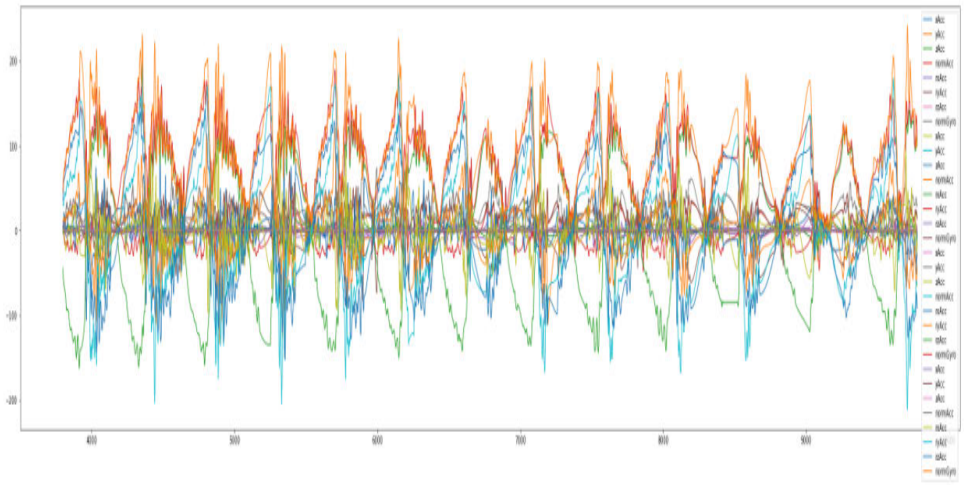

Abbildung A.3: 32 Signale eines gekürzten Satzes

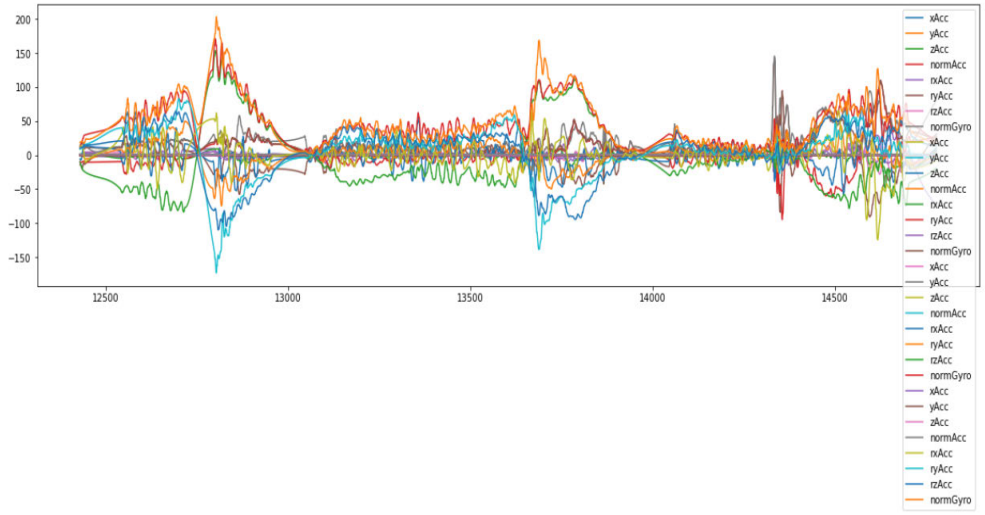

Abbildung A.4: Signale der fehlerhaften Liegestütze eines Satzes

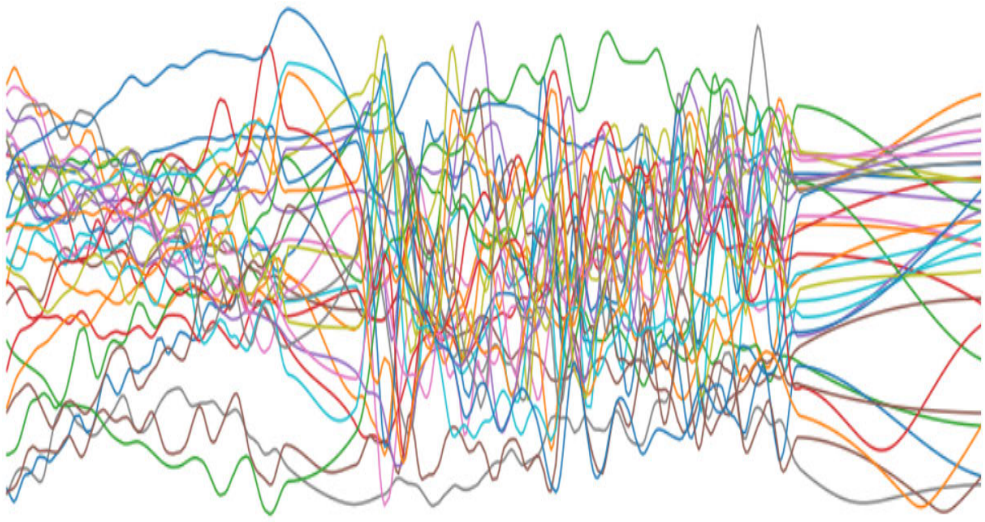

Abbildung A.5: Signale nach Moving Average-Methode

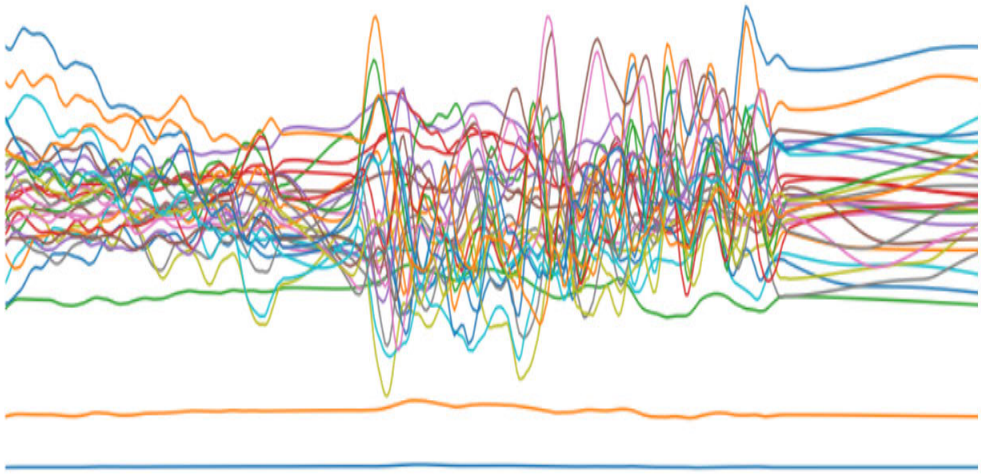

Abbildung A.6: Signale nach Butterworth-Filter-Methode

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort Datum Unterschrift im Original