Zurich University of Applied Sciences

School of Engineering

Centre for Artificial Intelligence (CAI)

Projektarbeit

Erkennung von Bewegungsanomalien mit Transformer Autoencoder und Deep Embedded Clustering

---

Authors:
Nevio Roccia, Bader Eddin Loukil

Supervisors:
Dr. Jasmina Bogojeska
Prof. Dr. Eveline Graf

Submitted on
22. Dezember 2023

Study program:
Data Science, B.Sc.

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Projektarbeit an der School of Engineering

Mit der Abgabe dieser Projektarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Projektarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

Zürich, 22.12.2023

Unterschriften:

Nevio Roccia: 

Bader Eddin Loukil: BL

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Projektarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Imprint

Project: Projektarbeit
Title: Erkennung von Bewegungsanomalien mit Transformer Autoencoder und Deep Embedded Clustering
Author: Nevio Roccia, Bader Eddin Loukil
Date: 22. Dezember 2023

Study program:
Data Science, B.Sc.
Zurich University of Applied Sciences

Supervisor 1:
Dr. Jasmina Bogojeska
Zürcher Hochschule für angewandte Wissenschaften (ZHAW)
Email: bogo@zhaw.ch

Supervisor 2:
Prof. Dr. Eveline Graf
Zürcher Hochschule für angewandte Wissenschaften (ZHAW)
Email: eveline.graf@zhaw.ch

Zusammenfassung

Diese Studie beleuchtet eine neuartige Anwendung von Machine Learning im Bereich der Sportrehabilitation unter Einsatz des ExerCube-Systems. Sie erforscht die Integration von Transformer Autoencodern und Deep Embedded Clustering (DEC) zur Analyse der Bewegungsdaten von Athleten während der Rehabilitation. Im Fokus steht die Optimierung von Bewegungsmustern und die Identifizierung von Übungen, die nach einer Verletzung, insbesondere nach einem ACL (Anterior Cruciate Ligament) Tear, vorteilhaft oder potenziell schädlich sein können. Obwohl die Forschung das Potenzial von fortgeschrittenen Machine Learning-Techniken aufzeigt, um Rehabilitationsmethoden zu revolutionieren – mit tieferen Einblicken in Bewegungsmuster und einer Verbesserung des Genesungsprozesses –, waren die Ergebnisse gemischt. Herausforderungen in der Datensegmentierung und -qualität führten zu weniger definitiven Ergebnissen beim Clustering und der Anomaly Detection als erwartet. Diese Erkenntnisse heben die kritische Bedeutung der Datenintegrität bei der Anwendung komplexer Machine Learning-Modelle hervor und betonen die Notwendigkeit einer rigorosen Datenvorbereitung, um das transformative Potenzial dieser Technologien für die Sportrehabilitation vollständig auszuschöpfen.

Abstract

This study explores a novel application of machine learning in the realm of sports rehabilitation through the use of the ExerCube system. It delves into the integration of Transformer Autoencoders and Deep Embedded Clustering (DEC) for the purpose of analyzing and optimizing athletes' movement data during the rehabilitation process. The research specifically targets the refinement of movement patterns and the discernment of exercises that are beneficial or potentially detrimental post-injury, with a particular focus on anterior cruciate ligament (ACL) tears. While the study demonstrates the potential of advanced machine learning techniques to revolutionize rehabilitation practices by providing deeper insights into movement patterns and aiding in the enhancement of the recovery process, the results were mixed. Challenges in data segmentation and quality were encountered, leading to less definitive clustering and anomaly detection than anticipated. These findings highlight the critical role of data integrity in the application of complex machine learning models and underscore the need for rigorous data preparation to fully leverage the transformative possibilities these technologies offer for sports rehabilitation.

Danksagung

Wir möchten uns herzlich bei Dr. Jasmina Bogojeska vom Center for Artificial Intelligence (CAI) für ihre hervorragende Betreuung und Unterstützung während dieses Projekts bedanken. Ihr tiefgreifendes Fachwissen und ihre wertvollen Einsichten waren für den Erfolg unserer Forschung unerlässlich. Ebenso gilt unser Dank Prof. Dr. Eveline Graf vom Institut für Physiotherapie der ZHAW für ihre wertvolle Mitarbeit und Expertise, die maßgeblich zur Verbindung von maschinellem Lernen und Sportrehabilitation in unserer Studie beigetragen haben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ML	**M**achine **L**earning
LSTM	**L**ong **S**hort-**T**erm **M**emory
DEC	**D**eep **E**mbedded **C**lustering
DTW	**D**ynamic **T**ime **W**arping
HMM	**H**idden **M**arkov **M**odel
RNN	**R**ecurrent **N**eural **N**etwork
PCA	**P**rincipal **C**omponent **A**nalysis
DCLOP	**D**eep **C**lustering-based **L**ocal **O**utlier **P**robabilities approach

Kapitel 1

Einleitung

1.1 Einführung in das Projekt

Das ‘ExerUP!’ Projekt stellt einen innovativen Ansatz in der Sportlerrehabilitation dar, der sowohl physische als auch mentale Aspekte der Genesung vereint. Dieses Projekt, entstanden in Zusammenarbeit mit Dr. Eveline Graf vom Institute of Physiotherapy und Dr. Jasmina Bogojeska vom Center of Artificial Intelligence, nutzt den von Sphery entwickelten ExerCube und erweitert seine Kapazitäten durch den Einsatz von Machine Learning, um tiefere Einblicke in die Rehabilitation zu gewähren. Wir werden in dieser Arbeit trotz der deutschen Sprache regelmässig englische Begriffe verwenden, um die Konsistenz mit der Forschung im Bereich des Maschinellen Lernens zu wahren.

1.2 ExerCube: Technologie und Anwendung

Der ExerCube selbst besteht aus einer offenen Würfelstruktur, auf deren drei Seiten eine virtuelle Spielwelt projiziert wird. Im Spiel ‘Sphery Racer’ führen die Nutzer auf einer futuristischen Rennstrecke bestimmte Fitnessübungen wie Squats, Burpees und Lunges durch. Sensoren an Händen, Füssen und Beinen erfassen dabei die Genauigkeit der Bewegungen. Unser Beitrag zu diesem Projekt liegt in der Analyse dieser Sensordaten, wobei wir moderne Machine-Learning-Techniken einsetzen, um Muster zu identifizieren, die sonst möglicherweise unentdeckt bleiben würden. In der Welt des Sports ist die Rückkehr zum Wettkampf nach einer Verletzung nicht nur eine Frage der körperlichen Erholung, sondern auch der mentalen Stärke. Mentale Hindernisse sind häufig ebenso anspruchsvoll wie physische Herausforderungen. Der ExerCube adressiert diese Problematik, indem er ein kontrolliertes Umfeld schafft, das echte Sportszenarien präzise simuliert. Nähere Informationen zum Aufbau und zur Funktionsweise des ExerCubes finden sich in Kapitel 3.1.

Abbildung 1.1: Ein proband im Exercube

1.3 Maschinelles Lernen in der Sportrehabilitation

Machine Learning gewinnt in der Welt der Sportrehabilitation zunehmend an Bedeutung. Es ermöglicht Forschern, fortschrittliche Analysen durchzuführen, die mit traditionellen Methoden nicht möglich sind. Die Komplexität, Dynamik, Multidimensionalität und Nichtlinearität menschlicher Bewegungen stellen dank der Anwendung dieser Methoden keine grossen Hürden mehr dar [1].

1.4 Weiterentwicklung bestehender Methoden und neue Ansätze

Unsere Forschungsarbeit baut auf den Erkenntnissen einer Vorgängergruppe auf, die Machine Learning zur Analyse von Bewegungsdaten für die Ermüdungsdetektion einsetzte [2].

Im Gegensatz zu früheren Ansätzen, bei denen LSTM-Autoencoder im Mittelpunkt standen [3], erkunden wir die Anwendung von Transformer-Autoencodern sowie Deep Clustering-Techniken, inspiriert von den vielversprechenden Ergebnissen in den Bereichen der Satellitentelemetrie [4] und der Anomalieerkennung im Energieverbrauch [5]. Die Transformer-Architektur, bekannt für ihre herausragenden Fähigkeiten im Umgang mit sequenziellen Daten [6], bietet eine ausgezeichnete Basis, um die zeitabhängigen Muster der Bewegungsdaten zu erfassen und zu analysieren.

Die Umstellung auf den Transformer-Autoencoder ermöglicht es uns, die komplexen Muster in den Bewegungsdaten umfassender zu erfassen. Zusätzlich haben wir den Ansatz des Deep Embedded Clustering (DEC) integriert, um die Identifikation von Clustern innerhalb der Bewegungsdaten zu optimieren und eine feinere Trennung zwischen normalen und anomalen Bewegungsmustern zu erreichen.

Die Analyse von Rekonstruktionsfehlern bleibt ein zentrales Element unserer Arbeit, jedoch mit dem Ziel, durch die Nutzung fortschrittlicherer Modelle präzisere und

aussagekräftigere Erkenntnisse zu gewinnen. Durch diese methodische Neuausrichtung streben wir danach, Anomalien in den Bewegungsdaten, die auf Erschöpfung oder erhöhte Verletzungsgefahr hinweisen könnten, noch genauer zu identifizieren.

1.5 Fazit und Ausblick

Die Integration dieser fortschrittlichen Machine-Learning-Methoden hat das Potenzial, Rehabilitationsverfahren grundlegend zu verändern. Das «ExerUP!» Projekt konzentriert sich auf die Behandlung von Kreuzbandrissen, einer der häufigsten Sportverletzungen, die einen signifikanten Einfluss auf die Leistungsfähigkeit der Athleten haben. Unser Ziel ist es, Bewegungsmuster zu optimieren und zu erkennen, welche Übungen während der Rehabilitation förderlich oder potenziell schädlich sein könnten und so das Therapieangebot um eine wertvolle Dimension zu erweitern. In den folgenden Abschnitten werden wir erforschen, wie unser Ansatz zur Verbesserung dieser Methoden beitragen kann.

Kapitel 2

Theoretische Grundlagen

2.1 Rehabilitation

Unter Rehabilitation versteht man den Prozess, der darauf abzielt, Menschen beim Wiederaufbau ihrer physischen, sensorischen und mentalen Fähigkeiten, die durch Verletzungen, Krankheiten oder Behinderungen verloren gegangen sind, zu unterstützen. Speziell in der Welt des Sports spielt die Rehabilitationstherapie eine grosse Rolle, wenn es darum geht, Athleten dabei zu helfen, zu ihrer Leistung vor einer Verletzung zurückzukehren. Sie beinhaltet eine Vielzahl an Praktiken – von physischer Therapie und Übungen über kognitives Training bis hin zu Massagen und Entspannungsmethoden. Je genauer diese Vorgehensweisen an den Patienten angepasst werden können, desto effektiver wirkt die Therapie.

2.2 Technologie und Rehabilitation

Die Integration von Technologie in den Rehabilitationsprozess ist seit Jahren ein zentrales Thema in der Forschung. Verschiedenste Ansätze wurden bereits untersucht, um technologischen Fortschritt zum Vorteil solcher Therapien zu nutzen. Ein prägnantes Beispiel ist die Studie „Robotic Technologies and Rehabilitation: New Tools for Stroke Patients' Therapy“ aus dem Jahr 2013, in der die Effekte robotisch unterstützter Methoden auf die Genesung von Schlaganfallpatienten erforscht wurden. Diese Methoden zeigten das Potential, personalisierte und effektivere Behandlungspläne zu ermöglichen [7].

Der ExerCube stellt eine innovative Ergänzung in dieser Technologielandschaft dar. Er demonstriert den Trend weg von einem „one-size-fits-all“-Ansatz hin zu massgeschneiderten und persönlichen Therapieformen. Der ExerCube ermöglicht durch seine interaktive und realitätsnahe, aber dennoch kontrollierte Form der Bewegungstherapie, eine elegante Verbindung moderner Technologien mit herkömmlichen Therapiemethoden, was diese adaptiver und effektiver macht.

Im nächsten Abschnitt werden wir die Rolle des Machine Learnings im Kontext dieser technologischen Erweiterung von physiotherapeutischen Verfahren näher beleuchten.

2.3 Machine Learning und Rehabilitation

Wir haben bereits die Rolle der Technologie in der Rehabilitation erörtert, insbesondere wie sie die Durchführung von Übungen unterstützt. Ein wesentliches Element, die Analyse und Bewertung der Patientenleistung während der Übungen, war bislang oft ein manueller, zeitaufwendiger Prozess. Der Fortschritt im Bereich des Maschinellen Lernens erlaubt jedoch zunehmend eine Automatisierung dieser Aufgaben. Solche Methoden ermöglichen es, komplexe Muster in den Daten zu erkennen und tiefere Einblicke zu gewinnen, die zuvor nicht zugänglich waren.

Das Paper „Machine Learning in Physiotherapy: A New Approach to Evaluate Performance During Exergames“ illustriert, wie Machine Learning-Algorithmen wie DTW und HMM effektiv eingesetzt werden können, um in technologiegestützten Umgebungen wie dem ExerCube die Leistung von Patienten zu bewerten. Dies eröffnet Möglichkeiten für Patienten, ihre Therapie auch ausserhalb klinischer Einrichtungen durchzuführen, ohne an Qualität einzubüssen. Gleichzeitig können dadurch Kosten gesenkt und Physiotherapeuten entlastet werden.[8]

Im nächsten Abschnitt werden wir die Funktionsweise solcher Machine Learning-Techniken näher beleuchten und deren Anwendung im Kontext des ExerCube-Projekts betrachten.

2.4 Anomalie Detektion in Machine Learning

Gemäss [9] wird unter einer Anomalie eine signifikante Abweichung von der normalen Datenverteilung verstanden. Diese zeigen sich entweder in einer einzelenen Beobachtung, oder in einer Abfolge von Beobachtungen, die deutlich von dieser Normverteilung unterschiedlich sind.

Die Erkennung von Anomalien ist eine wesentliche Funktion von Machine-Learning-Algorithmen. Dieser Prozess ist in vielen Bereichen von Bedeutung, wie beispielsweise der Betrugserkennung, Zustandsüberwachungssystemen und Finanzprognosen, und findet Anwendung in unterschiedlichen Datentypen. Forscher verwenden verschiedene Methoden zur Identifikation von Anomalien in diesen Daten. Unser Fokus liegt auf der Anomalieerkennung in Zeitreihendaten, die in univariate und multivariate Zeitreihen unterteilt werden können: Univariate bestehen aus einer einzelnen, multivariate aus mehreren, über die Zeit veränderlichen und voneinander abhängigen Variablen. Deep Neural Networks sind besonders geeignet zur Modellierung von Abhängigkeiten für Daten mit komplexen Strukturen. Abbildung 2.1 zeigt verschiedene deep learning Architekturen für time-series-anomaly Erkennung.

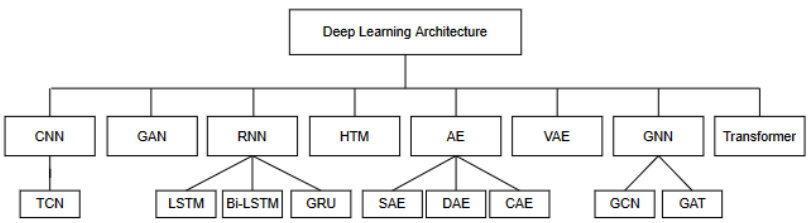

Abbildung 2.1: Deep Learning Architekturen für time series anomaly detection, adaptiert von [10]

Das Paper „Deep Learning for Time Series Anomaly Detection: A Survey“ bietet einen Überblick über Deep-Learning-Methoden für die Anomalieerkennung und präsentiert eine neue Taxonomie, die Deep-Anomaly-Detection-Modelle in die Kategorien vorhersagend (forecasting-based), rekonstruierend (reconstruction-based) und hybride Methoden unterteilt. Jede Kategorie wird weiterhin basierend auf den verwendeten Deep-Neural-Architekturen differenziert. Zudem werden state-of-the-art Ansätze in diesem Bereich untersucht, die Vor- und Nachteile verschiedener Methoden hervorgehoben und ihre Anwendungen in verschiedenen Domänen diskutiert. Es wird die Effizienz von Deep-Learning-Modellen betont, insbesondere jener, die auf neuronalen Netzwerken basieren, im Umgang mit komplexen Zeitreihendaten. In Anbetracht der Tatsache, dass unser Datensatz aus 3D-Positionsdaten von 28 Markern besteht, aufgenommen mit einer Rate von 240 Frames pro Sekunde, erwies sich das Transformer-Modell als eine solide Wahl. Transformer sind eine Unterkategorie neuronaler Netzwerke und zeichnen sich durch ihre Fähigkeit aus, Muster und Anomalien in solchen Daten zu erkennen, da sie langfristige Abhängigkeiten zwischen Datenpunkten erfassen können.[10]

Des Weiteren behandelt das Paper „Power Consumption Predicting and Anomaly Detection Based on Transformers and K-Means” von Zhang et al. (2021) die Idee, Anomalien in Stromnutzungsdaten mit Hilfe eines Transformers und K-Means-Clustering zu erkennen. Dies ermöglicht es, die Funktionsweise der Transformer durch das Clustern der Daten zu ergänzen und somit die Genauigkeit der Anomaliedetektion zu erhöhen. Zhang et al. haben demonstriert, dass ein solches Modell, trainiert auf historischen Daten, normale Abweichungen vorhersagen kann. Im Falle einer starken Abweichung kann von einer Anomalie ausgegangen werden. Diese Herangehensweise, um Ausreisser in komplexen multivariaten Zeitreihendaten isolieren zu können, nutzen wir für unsere Arbeit.[5]

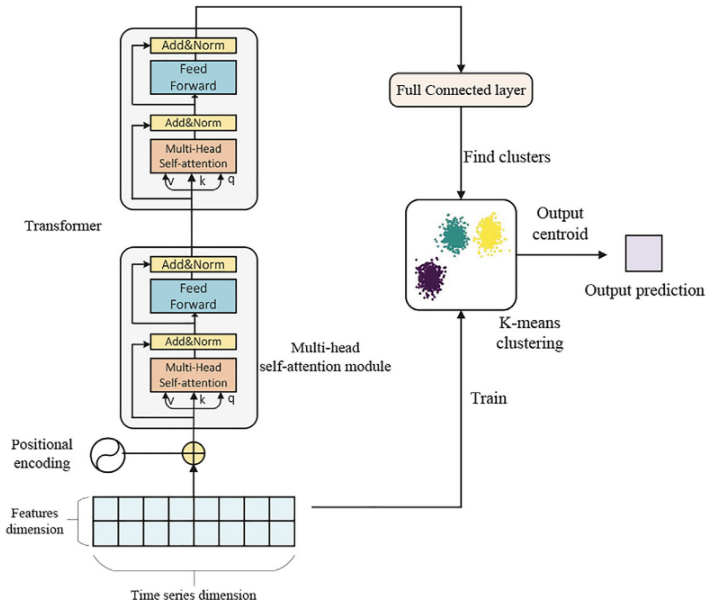

Abbildung 2.2: Transformer und K-means Archtitektur, adaptiert von [5]

2.5 Transformer

Das Transformer-Modell wurde in dem einflussreichen Paper „Attention Is All You Need“ von Vaswani et al. (2017) vorgestellt und löste einen revolutionären Wandel in der Welt des maschinellen Lernens aus, insbesondere in der Verarbeitung von sequenziellen Daten. Im Gegensatz zu früheren Modellen wie Recurrent Neural Networks (RNNs) und Long Short-Term Memory Networks (LSTMs), die Daten sequenziell verarbeiten, ermöglichen Transformer mittels eines Mechanismus namens „Attention“ die parallele Verarbeitung ganzer Datensequenzen. Dieser fundamentale Unterschied erlaubt es Transformers, Abhängigkeiten zwischen Elementen einer Sequenz effizienter und effektiver zu erkennen.

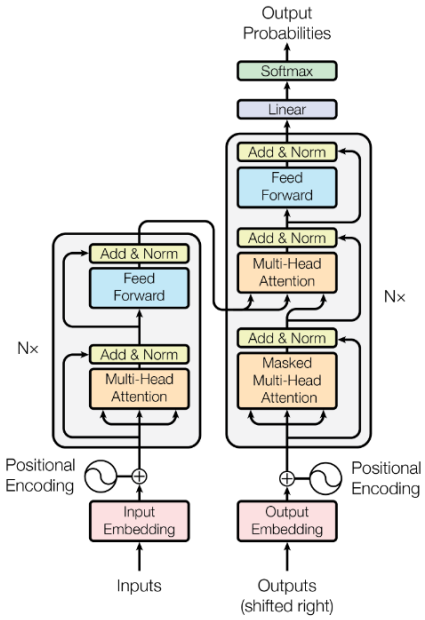

Abbildung 2.3: Architektur des Transformer-Modells

Zu den wesentlichen Elementen der Architektur gehören:

1. **Encoder und Decoder Stacks**: Der Encoder und Decoder bestehen jeweils aus sechs identischen Schichten, die wiederum aus zwei Sub-Layern bestehen. Der Encoder verarbeitet die Eingabesequenz, während der Decoder die Ausgabesequenz erzeugt, wobei ein dritter Sub-Layer Multi-Head Attention auf den Output des Encoders anwendet.

2. **Self-Attention-Mechanismus**: Dieser ist ein zentraler Bestandteil der Transformer-Architektur. Er ermöglicht es jeder Position in der Eingabesequenz, auf alle anderen Positionen innerhalb derselben Sequenz zu achten.

3. **Multi-Head Attention**: Durch den Einsatz mehrerer „Heads“ im Attention-Mechanismus kann das Modell gleichzeitig auf verschiedene Teile der Sequenz achten und so ein tieferes Verständnis des Inputs erlangen.

4. **Position-wise Feed-Forward Networks**: Jede Schicht im Encoder und Decoder wendet auf jede Position ein vollständig verbundenes Feed-Forward-Netzwerk an.

5. **Positional Encoding**: Da Transformer Daten nicht sequenziell verarbeiten, werden Positional Encodings hinzugefügt, um dem Modell Informationen über die Position jedes Elements in der Sequenz zu vermitteln.

Diese Verarbeitungsweise unterscheidet sich deutlich von der sequenziellen Verarbeitung durch RNNs und LSTMs, die es erschweren, Zusammenhänge zwischen weit auseinanderliegenden Daten in einer Sequenz festzustellen – ein Problem, das oft als „Vanishing Gradient Problem“ bezeichnet wird. Im Kontext von Bewegungsdaten ermöglicht diese Funktionsweise des Transformers, Anomalien über die gesamte Bewegung hinweg parallel zu detektieren. Zudem sind Transformer hochgradig skalierbar und effizient, was für das Arbeiten mit komplexen Datensätzen wie in Ihrem Projekt unerlässlich ist.[6]

2.6 K-means clustering

Der Begriff „k-means“ wurde erstmals 1967 von James MacQueen eingeführt, obwohl der Algorithmus bereits früher von Stuart Lloyd im Jahr 1957 und Edward W. Forgy im Jahr 1965 vorgestellt wurde. Seitdem hat er sich stetig weiterentwickelt und ist zu einem fundamentalen Werkzeug in der Welt der Data Science geworden.[11] Der Name „k-means“ besteht aus zwei Teilen: „K“ bezieht sich auf die Anzahl der verschiedenen Cluster, in die die Datenpunkte eingeteilt werden sollen. „Means“ bedeutet, dass der Durchschnitt aller Datenpunkte in einem Cluster das Zentrum oder Centroid dieser Datenmenge bildet, welches vom Algorithmus bestimmt wird.[12] Der Prozess des k-means Clustering verläuft in folgenden Schritten:

1. **Initialisierung**: Der Algorithmus beginnt mit der zufälligen Auswahl von k Anfangscentroids aus den Datenpunkten.

2. **Zuweisung**: Jeder Datenpunkt wird dem Centroid zugewiesen, zu dem die euklidische Distanz am geringsten ist.

3. **Update**: Die Centroids der Cluster werden neu berechnet, basierend auf dem Mittelwert aller dem jeweiligen Cluster zugewiesenen Datenpunkte.

4. **Iteration**: Die Schritte 2 und 3 werden wiederholt, bis sich die Centroids nicht mehr signifikant verändern.

Der Algorithmus zielt darauf ab, die „within-cluster sum of squares“ zu minimieren, ein Mass für die Varianz innerhalb eines Clusters. Diese Varianz sollte möglichst gering sein, um sicherzustellen, dass die Datenpunkte innerhalb eines Clusters einander so ähnlich wie möglich sind.[12]

Zu den Einschränkungen des k-means Clustering gehört die Annahme, dass alle Cluster sphärisch und von gleicher Grösse sind, was in realen Datensätzen nicht immer der Fall ist. Die Qualität der Ergebnisse kann zudem stark variieren, je nach Wahl der anfänglichen zufälligen Centroids. Daher werden oft mehrere Durchläufe mit unterschiedlichen Initialisierungen durchgeführt, um optimale Ergebnisse erzielen zu können.[11]

2.7 DEC-Modell (Deep Embedded Clustering)

Das DEC-Modell, vorgestellt von Junyuan Xie, Ross Girshick und Ali Farhadi, ist ein fortschrittlicher Ansatz im Bereich des unüberwachten maschinellen Lernens. Es kombiniert Deep Learning mit Clustering, um eine effiziente Gruppierung von Daten zu ermöglichen. Der Prozess beginnt mit der Verwendung eines tiefen neuronalen Netzwerks zur Einbettung der Daten in einen niedrigdimensionalen Raum. Anschließend werden die Datenpunkte iterativ und selbstorganisierend zu Clustern gruppiert. Diese Methode ist besonders effektiv bei der Entdeckung latenter Strukturen in komplexen Datensätzen und findet Anwendung in Bereichen wie der Bilderkennung und der Anomalieerkennung. Das DEC-Modell ermöglicht es, tiefere Einblicke in ungelabelte Daten zu gewinnen, was in der Rehabilitationstechnologie für die Analyse von Patientendaten von Bedeutung sein kann [13].

Kapitel 3

Methodik

3.1 Datenerfassung

Wie bereits in Abschnitt 1.2 erwähnt, wurden die Daten für dieses Projekt mit Hilfe des Exercubes von Sphery aufgezeichnet. Zu Beginn der Datenerfassung wurden Sensoren an den Probanden und Probandinnen angebracht. An Hüften, Oberschenkeln und Unterschenkeln wurde jeweils eine Platte mit vier Markern befestigt. Zusätzlich wurden an beiden Schuhen jeweils fünf Marker angebracht, wobei vier auf der Rückseite und einer an der Spitze des Schuhs positioniert wurden. Diese Marker wurden mit einer Erfassungsrate von 240 Hz durch das Motion-Capture-System Vantage von Vicon aufgezeichnet [14]. Jede Messung erfasste dabei Werte für jeden Marker auf den X-, Y- und Z-Achsen. Die Positionen der einzelnen Sensoren sind in Abbildung 3.1 dargestellt.

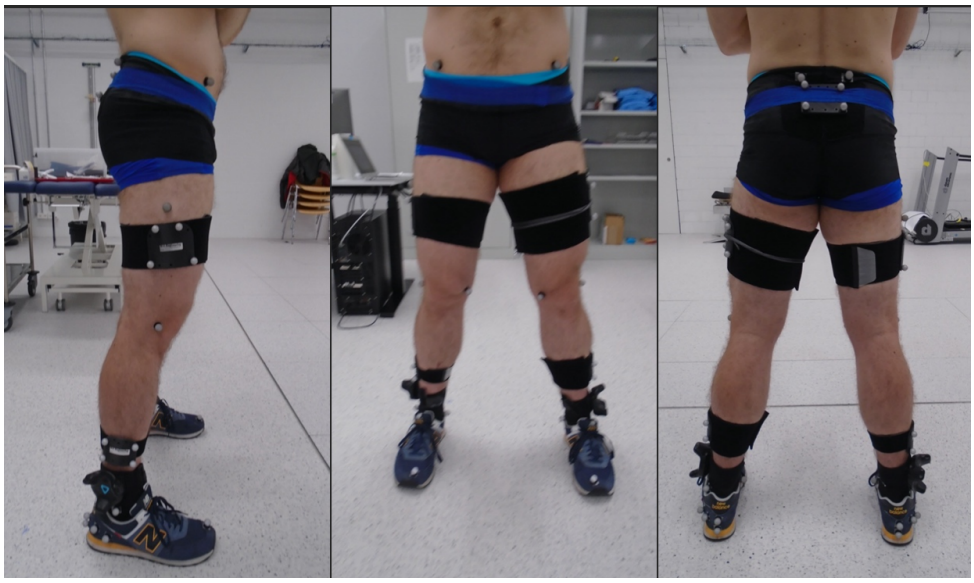

Abbildung 3.1: Proband mit befestigten Sensoren

Die Probanden und Probandinnen standen für den Versuch auf einer mit einem X markierten Stelle in der Mitte des Exercubes und absolvierten einen 25-minütigen Durchlauf des Spiels ‘Sphery Racer’. Während des gesamten Ablaufs wurden die Positionen aller Sensoren durch ein System von Kameras rund um den Exercube im Raum erfasst. Das Institut für Physiotherapie erhob auf diese Weise Bewegungsdaten von 20 Probandinnen und Probanden zwischen November 2021 und Juni 2022, die uns anschliessend im .c3d Format [15] zur Verfügung gestellt wurden. Zur Auswertung

haben wir ein Array mit allen verfügbaren Daten erstellt, das folgende Dimensionen aufweist:

$$\text{Dim}(\text{Array}) = 20 \times 90 \times 360000$$

Die erste Dimension bezieht sich auf die 20 Probanden, die zweite auf die 30 Marker in 3 Dimensionen (30 Marker × 3 Dimensionen = 90 Features) und die dritte auf die Anzahl der Frames pro Patient (25 Minuten × 240 Frames pro Sekunde = 360000 Frames).

Abbildung 3.2: Kamerasystem im Bewegungslabor

3.2 Datenbereinigung

3.2.1 Entfernte Marker

Aus Erkentnissen aus der Arbeit unserer Vorgruppe war uns bekannt, dass bei vier Probanden die Sensoren an den Fussspitzen jeweils fehlten. Wir haben die Datenpunkte dieser Sensoren deswegen bei allen Probanden entfernt und berücksichtigen diese fortlaufend nicht.[2] Dementsprechend ist die Dimension unseres Arrays für den Rest der Arbeit gegeben durch:

$$\text{Dim}(\text{Array}) = 20 \times 84 \times 360000$$

3.2.2 Explorative Datenanalyse

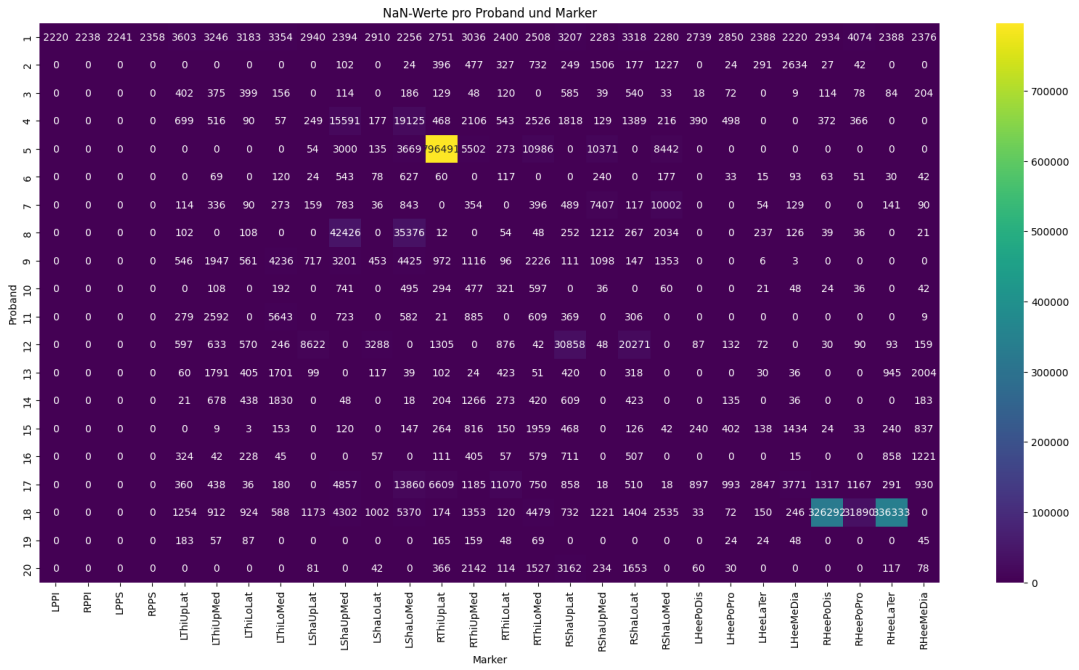

Abbildung 3.3: Heatmap der NaN-Werte im Datensatz

Um ein umfassendes Verständnis über die Integrität unseres Datensatzes zu gewinnen, haben wir eine explorative Datenanalyse durchgeführt. Ziel war es, fehlende Werte in den Daten zu identifizieren, da diese aufgrund der komplexen und dynamischen Natur von Bewegungsmustern eine bedeutende Herausforderung darstellen. Zur Visualisierung der fehlenden Werte haben wir eine Heatmap verwendet, die einen sofortigen, grafischen Überblick über die Komplexität unserer Daten bietet. Die Heatmap (siehe Abbildung 3.3) verwendet einen Farbgradienten, um die Anzahl fehlender Werte über alle Teilnehmer und Sensoren darzustellen. Auf der Y-Achse sind die Teilnehmernummern und auf der X-Achse die Sensoren abgebildet. Jede Zelle repräsentiert die Anzahl fehlender Werte pro Teilnehmer und Sensor: Je heller die Farbe, desto mehr fehlende Werte sind vorhanden. Diese Einblicke sind entscheidend für die nachfolgenden Schritte des Daten-Preprocessings und beeinflussen unsere Strategie im Umgang mit den fehlenden Daten. Zudem erleichtert die Visualisierung die vorläufige Bewertung potenzieller Verzerrungen, die durch die fehlenden Daten entstehen könnten,

was für die Integrität unserer statistischen Analyse und die Validität der Ergebnisse von grosser Bedeutung ist.

Wie in Abbildung 3.3 ersichtlich, treten bei jedem Teilnehmer Datenlücken auf. Eine mögliche Erklärung dafür ist, dass beim Aufzeichnen Sensoren nicht ausreichend oder gar nicht von den Kameras erfasst werden, wodurch die betreffenden Datenpunkte auf NaN (Not a Number) gesetzt werden.

3.2.3 Rekonstruktion

Die Rekonstruktion unserer Daten kann in zwei Hauptprozesse eingeteilt werden: Die Rekonstruktion von fehlenden Sensoren auf Platten und von solchen, die nicht auf Platten angebracht wurden (siehe 3.1).

Die Rekonstruktion der Sensoren auf Platten wurde nach folgenden Schritten umgesetzt:

1. **Laden der Daten:** Als Erstes luden wir die kompletten Daten in ein Array ein.

2. **Bestimmung örtlicher Beziehungen:** Eine von uns geschriebene Funktion berechnet mit Hilfe der NumPy Library den Normalvektor der Platte, auf welcher sich der fehlende Sensor befindet.

3. **Iterative Rekonstruktion:** Wir iterierten über alle Frames für jeden Durchgang in unserem Array und identifizierten Platten mit exakt einem fehlenden Wert, um die Genauigkeit der Rekonstruktion zu gewährleisten. Mit Hilfe der Position der anderen 3 Sensoren der Platte, einem Referenzframe und dem vorher berechneten Normalvektor, berechneten wir die Position des fehlenden Wertes.

4. **Speichern des Arrays:** Schlussendlich speicherten wir das rekonstruierte Array ab für weitere Preprocessing-Schritte.

Die Rekonstruktion von fehlenden Werten der Sensoren, die nicht auf einer Platte angebracht wurden, konnten wir nicht geometrisch umsetzen. Leider erlaubte uns die vorgegebene Zeitspanne für dieses Projekt nicht, anspruchsvolle Interpolationsmethoden anzuwenden, um diese zu ergänzen. Aus diesem Grund entschieden wir uns, die restlichen fehlenden Werte mithilfe des Mittelwerts zu ersetzen. Dazu luden wir die kompletten Daten erneut ein und teilten sie für jeden Probanden in Blöcke von 120 Frames auf. Falls nur NaN-Werte für einen Sensor in einem Block vorkamen, expandierten wir den Block, um ihn mit den benachbarten Datenpunkten ergänzen zu können. Anschliessend berechneten wir den Mittelwert und setzten diesen für alle NaN-Werte in diesen Blöcken ein. Diese Methode der dynamischen Anpassung von Datenblöcken und der Verwendung von kontextuellen Durchschnittswerten zur Imputation erwies sich als effektiv, um fehlende Werte, die nicht mit spezifischen Sensorplatten verknüpft sind, zu behandeln und verbesserte somit die Gesamtintegrität und den Nutzen unserer Bewegungsdaten.

3.3 Erstellen der Trainingsdaten

Um Anomalien in den Bewegungen präziser identifizieren zu können, entschieden wir uns, unsere nun rekonstruierten Rohdaten nach einzelnen distinkten Bewegungen aufzuteilen. Die Inspiration für diesen Ansatz basiert auf der Arbeit von Jeon et al. in dem Paper „Anomalous Gait Feature Classification From 3-D Motion Capture Data“ (2022). Ihre Studie zeigte, dass die Analyse von 3D-Bewegungsdaten effektiver ist, wenn diese in distinkte Bewegungsvektoren unterteilt werden [16]. Michelle Haas, wissenschaftliche Mitarbeiterin des Bewegungslabors am Institut für Physiotherapie der ZHAW, erläuterte uns am 14. November 2023, dass ihr Team derzeit die Auswertungen und Segmentierungen der Daten manuell vornimmt. Aufgrund zeitlicher Begrenzungen in unserem Projekt mussten wir auf eine alternative Methode ausweichen. Zusätzlich zu den Bewegungsdaten erhielten wir LogFiles für jeden Durchlauf der Probanden. Struktur und Inhalt dieser LogFiles sind in Abbildung 3.4 dargestellt. Sie enthalten Timestamps, die exakt festhalten, zu welchem Zeitpunkt innerhalb des 25-minütigen Trainings eine bestimmte Übung durchgeführt und ob sie korrekt ausgeführt wurde. Die Rohdaten der Sensoren umfassen pro Proband eine Dauer von genau 25 Minuten (360.000 Frames) und enthalten keine Timestamps. Die LogFiles sind hingegen einige Sekunden länger. Dies führte zu Unklarheiten darüber, wie eine exakte Anpassung gewährleistet werden konnte. Eine detaillierte Diskussion dieser Problematik findet sich im Abschnitt über die Limitationen unserer Arbeit 5.2.1.

```
Logfile created: 2021.11.16 10:04:00
Sphery Racer (version: 0.15.9, unity: 2018.1.9f2)
Model: System Product Name (System manufacturer), OS: Windows 10  (10.0.0) 64bit
Platform: WindowsPlayer, type: MININT-MGSEQUC
[2021-11-16 11:05:02 +1] Training     | Permission to use personal data:     True
[2021-11-16 11:05:02 +1] Training     | Workout mode:                        Standard
[2021-11-16 11:05:02 +1] Training     | Game mode:                           Competition
[2021-11-16 11:05:02 +1] Training     | Training:                            DualFlow
[2021-11-16 11:05:02 +1] Training     | Difficulty:                          Expert
[2021-11-16 11:05:02 +1] Training     | Used adaptivity routine:             Competition
[2021-11-16 11:05:02 +1] Training     | Race track identifier:               'DualFlow_Competition'
[2021-11-16 11:05:02 +1] Training     | Pre-selected seed:                   508
[2021-11-16 11:05:02 +1] Training     | Pit-Stop tutorials enabled:          True
[2021-11-16 11:05:02 +1] Training     | Training time:                       25min
[2021-11-16 11:05:02 +1] Training     | Pause time:                          0,0min
[2021-11-16 11:05:02 +1] Training     | Start speed (normalized):            0,5
[2021-11-16 11:05:02 +1] Training     | HR sensor connected:                 False
[2021-11-16 11:05:02 +1] Training     | HRMax value:                         194bpm
[2021-11-16 11:05:02 +1] Training     | HRTarget value:                      90%
[2021-11-16 11:05:37 +1] Training     | No HR sensor selected.
[2021-11-16 11:06:15 +1] Track-Gen    | Generator initialized with seed: 508.
[2021-11-16 11:06:15 +1] Track-Gen    | Start generating race beginning with pit stop.
[2021-11-16 11:06:15 +1] Race         | Finished race stage...
[2021-11-16 11:06:15 +1] Sphery ExerCube. Started new race!
[2021-11-16 11:06:44 +1] Race         | Exiting pitstop...
[2021-11-16 11:06:44 +1] Manipulation | All exercises correct for the last 20s. Changed cognition value to 0,5 (normalized).
[2021-11-16 11:06:57 +1] Evaluation   | Exercise correct: 'Punch Left'
[2021-11-16 11:06:57 +1] Evaluation   | Exercise perfectly timed: 'Punch Left'!
[2021-11-16 11:06:58 +1] Evaluation   | Exercise completed (reset pose)! Evaluation finished.
[2021-11-16 11:07:00 +1] Evaluation   | Exercise correct: 'Squat'
[2021-11-16 11:07:00 +1] Evaluation   | Exercise perfectly timed: 'Squat'!
[2021-11-16 11:07:01 +1] Evaluation   | Exercise not completed (timeout)! Evaluation finished.
[2021-11-16 11:07:03 +1] Evaluation   | Exercise correct: 'Touch Left'
```

Abbildung 3.4: Ausschnitt eines LogFiles

In einem ersten Schritt teilten wir unser Datenarray entlang der ersten Dimension auf und speicherten die 20 neu entstandenen Arrays als Dataframes in der Pandas-Bibliothek, um zwei zusätzliche Spalten für die Timestamps und die ausgeführten Übungen hinzufügen zu können. Da bis zur letzten Sekunde in den LogFiles Bewegungen aufgezeichnet wurden, wählten wir den letzten Timestamp als Anhaltspunkt, um sicherzustellen, dass alle Bewegungen berücksichtigt werden konnten. Von diesem Zeitpunkt zogen wir 25 Minuten ab und definierten diesen berechneten Zeitpunkt als Startpunkt für unsere Rohdaten. Mit einer Aufzeichnungsfrequenz von 240 Hz entstanden pro Sekunde 240 Frames für jede Bewegung. So konnten wir die ersten 240

Frames mit dem ersten berechneten Timestamp versehen, eine Sekunde hinzurechnen und die folgenden 240 Frames annotieren, bis wir das Ende der Rohdaten erreicht hatten. Diesen Prozess wiederholten wir iterativ für jeden Probanden. Das Hinzufügen dieser Timestamps ermöglichte uns nun einen Abgleich mit den LogFiles. Diese enthielten zusätzlich Informationen darüber, ob die Bewegung korrekt ausgeführt wurde oder nicht. Da es für unser Modell wichtig war, korrekte Bewegungsabläufe als Baseline zu lernen, berücksichtigten wir nur Übungen, bei denen der Eintrag „Exercise correct:“ vorlag, extrahierten den Namen der Übung und füllten die neue Spalte in unserem Dataframe zum entsprechenden Zeitpunkt mit dem Namen der Übung. Schlussendlich entfernten wir sämtliche Spalten, in denen in der Übungsspalte kein Eintrag zu finden war (also alle Bewegungen, die zwischen spezifischen Übungen gemacht wurden) und fügten alle einzelnen Dataframes wieder zu einem grossen Frame zusammen. Das Endresultat war ein Array mit den Rohdaten aller korrekt ausgeführten Sportübungen aller 20 Probanden über ihre gesamte Trainingszeit.

3.3.1 Varianz vs Overfitting und Underfitting

Daten mit hoher Varianz können zu Modellen führen, die versuchen, jedes Detail zu erlernen, was das Risiko von Overfitting erhöht. Andererseits bieten Daten mit sehr geringer Varianz nicht genügend 'Noise', was zur Entwicklung eines Modells führen kann, das nicht robust genug ist, um subtilere Anomalien zu erkennen. Um ein Gleichgewicht zu finden, haben wir die Varianz unserer Daten pro Übung visualisiert (siehe Abbildung 3.5). Auf den ersten Blick fielen die Übungen "Jump" und "Squat" als geeignete Kandidaten auf.

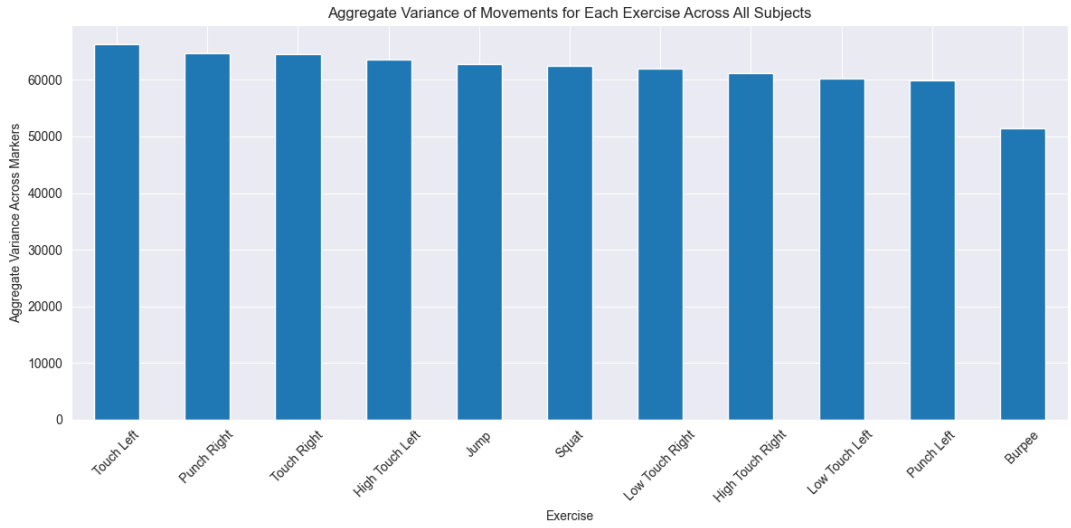

Abbildung 3.5: Varianz der Bewegungsdaten pro Übung

Zusätzlich war es uns wichtig zu gewährleisten, dass genügend Datenpunkte vorhanden waren, um das Modell effektiv trainieren zu können. In Tabelle 3.1 haben wir visualisiert, wie häufig jede Übung in den Daten vorkommt. Aufgrund der Erkenntnis, dass die Übungen "Jump" und "Squat" eine angemessene Varianz aufweisen und eine hohe Anzahl an Datenpunkten bieten, entschieden wir uns, mit diesen beiden Übungen fortzufahren.

Übung	**Anzahl der Datenpunkte**
Punch Right	379200
Jump	376560
Punch Left	358320
Squat	340080
High Touch Left	235920
High Touch Right	215520
Low Touch Left	206640
Low Touch Right	202560
Touch Left	201360
Touch Right	193920
Burpee	81600

Tabelle 3.1: Anzahl der Datenpunkte pro Übung im Array

3.4 Modell Architektur

Basierend auf der initialen Datenerfassung und -aufbereitung war der nächste Schritt in unserer Modellentwicklung das Preprocessing der Daten für den Einsatz im Transformer Autoencoder Modell. Dieser Abschnitt erläutert diesen Prozess sowie die Konfiguration der Modellarchitektur. Wie im Abschnitt 2.4 erwähnt, basiert unser Modell auf der Studie von Zhang et al., die die Wirksamkeit eines Transformer Modells in Verbindung mit K-Means Clustering demonstrierten [5]. Im Laufe unserer Forschung vertieften wir den Clustering-Aspekt und stiessen auf die Arbeit 'Deep Clustering-Based Anomaly Detection and Health Monitoring for Satellite Telemetry' von Obied et al. (2023), die einen neuen Ansatz, DCLOP, vorstellt, welcher Deep Clustering mit Anomaly Detection kombiniert, um Satelliten-Telemetriedaten zu analysieren. Dieser Ansatz nutzt einen deep-embedding Autoencoder in Kombination mit einer dynamisch gewichteten Verlustfunktion für effizientes Deep Clustering [4]. Die folgenden Abschnitte beschreiben, wie wir die Grundlagen dieser beiden Studien auf unser Projekt adaptiert haben.

3.4.1 Normalisierung und Segmentierung

Der erste Schritt in unserer Pipeline war die Normalisierung der Daten. Mit dem StandardScaler von Scikit-learn haben wir jedes Feature normalisiert, sodass es einen Mittelwert von Null und eine Varianz von Eins aufweist [17]. Diese Normalisierung gewährleistet, dass jede Dimension der 3D-Bewegungsdaten auf einer ähnlichen Skala ist, was entscheidend für die Präzision und Effektivität des Transformer Autoencoder-Modells ist.

Nach der Normalisierung der numerischen Features war der nächste Schritt in unserer Datenverarbeitungspipeline die Behandlung der kategorischen Daten, speziell die `Exercise` Spalte in unserem Datensatz. Da unser Transformer Autoencoder-Modell numerische Eingaben erfordert, war es notwendig, die kategorischen Daten in eine für das Modell geeignete Form zu konvertieren. Hierzu setzten wir das One-Hot-Encoding-Verfahren ein.

Mithilfe der Funktion `pd.get_dummies()` aus der Pandas-Bibliothek wandelten wir die kategorische `Exercise` Spalte in mehrere binäre Spalten um. Jede Spalte repräsentiert eine Übungsart (z.B. 'Squat', 'Jump'), wobei der Wert 1 anzeigt, dass die betreffende Übung in der jeweiligen Sequenz ausgeführt wurde, und 0 das Gegenteil darstellt. Diese Transformation führte zu einer Erhöhung der Eingabedimension von ursprünglich 84 auf 86, indem zwei zusätzliche Spalten für die One-Hot-codierten Übungsarten hinzugefügt wurden [18].

Anschliessend segmentierten wir die Daten in distinkte Sequenzen. Jede Sequenz repräsentiert eine Bewegungseinheit. Aufgrund der Aufzeichnungsfrequenz des Motion-Capture-Systems von 240 Hz und der Beschränkung der Bewegungsanalyse auf eine Sekunde gemäss den Logfiles ergaben sich Sequenzen mit einer Länge von 240 Frames und jeweils 86 features.

3.4.2 Train/Test/Validation Aufteilung

Die Sequenzen wurden in ein Trainings-, ein Validierungs- und ein Testset aufgeteilt. Die Grösse aller drei Sets wurde folgendermassen gewählt:

- Trainingsset: 60% der Daten, verwendet um das Modell zu trainieren.
- Validierungsset: 30% der Daten, verwendet für das Hyperparameter-Tuning und um Overfitting zu verhindern.
- Testset: 10% der Daten, verwendet um das Modell auf neuen, unbekannten Daten zu testen.

3.4.3 Transformer Autoencoder Konfiguration

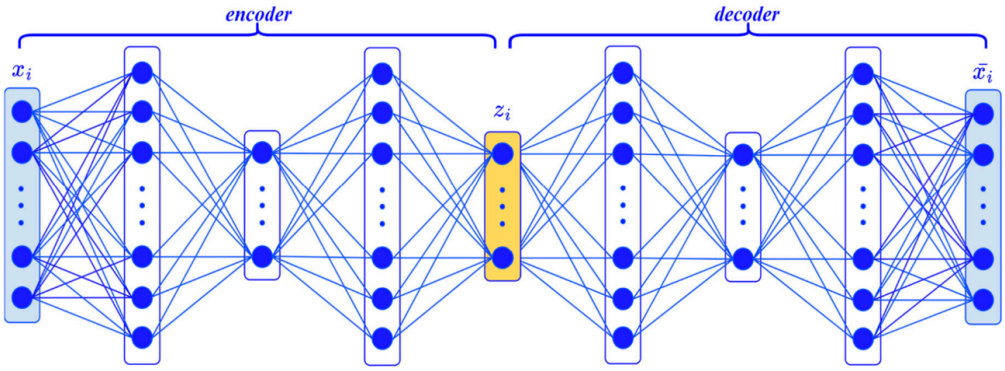

Abbildung 3.6: Transformer Autoencoder Archtiektur adaptiert von [4]

Nach der Vorbereitung der Daten setzten wir das Transformer Autoencoder Modell auf. Unsere Architektur folgt dem Konzept von Obied et al. (siehe Abbildung 3.6) und basiert auf PyTorch, einer weit verbreiteten Open-Source-Bibliothek für maschinelles Lernen. PyTorch bietet ein robustes und flexibles Framework für Deep Learning Anwendungen. Die Kernkomponenten – der Encoder und der Decoder – wurden unter Verwendung von Standardmodulen und Hilfsfunktionen des PyTorch Neural Network Moduls (`torch.nn`) konstruiert [19].

Encoder: Der Encoder beginnt mit einem Positional Encoding Layer, der den Input-Daten sequenzielle Informationen hinzufügt. Dies ist entscheidend, um die Reihenfolge der Sequenzen im Attention-Mechanismus beizubehalten. Für diesen Zweck implementierten wir eine eigene Klasse namens `PositionalEncoding` im PyTorch-Framework. Anschliessend reduziert ein Linear Reduction Layer die Eingabedimension (`input_dim` = 86) auf einen niedrigerdimensionalen latenten Raum (`latent_dim` = 46). Insgesamt kodieren acht Encoder-Layers (`nn.TransformerEncoderLayer`) die komplexen Abhängigkeiten in unseren Input-Daten in den sogenannten Latent Space. Das Feed-Forward-Netzwerk in jedem Layer besitzt dieselbe Dimension wie unsere latente Dimension (`latent_dim`), enthält einen Multi-Head-Self-Attention-Mechanismus mit jeweils zwei Attention Heads und wendet Layer Normalization an, um den Lernprozess zu stabilisieren.

Decoder: Der Decoder bildet eine Spiegelung des Encoders und besteht ebenfalls aus acht Decoder-Layers mit Multi-Head Self-Attention-Mechanismen, einem Feed-Forward-Netzwerk in der Dimension von `latent_dim` und Layer Normalization. Nach den acht Encoder-Layers durchlaufen die Daten zwei zusätzliche lineare Schichten. Die erste Schicht expandiert die Dimension von 46 (`latent_dim`) auf 92 (`latent_dim * 2`), und die zweite reduziert sie wieder auf 46 (`latent_dim`). Die Integration dieser zusätzlichen linearen Schichten nach den Transformer-Decoder-Layers ermöglicht eine komplexere Transformation der Daten. Dieser Ansatz verbessert die Fähigkeit des Modells, komplexere Muster und Beziehungen innerhalb der Daten zu erfassen, was zu einer potenziell besseren Rekonstruktionsgenauigkeit führt.

Abschliessend transformiert der Output Layer des Decoders die Daten zurück in die ursprüngliche Eingabedimension (`input_dim` = 86), womit der Prozess der Datenrekonstruktion vollendet wird.

Training Setup: Das Modell wurde unter Verwendung folgender Hyperparameter trainiert:

1. Optimizer: Adam Optimizer. Transformer-Modelle arbeiten insbesondere bei der Anwendung auf 3D-Daten in hochdimensionalen Räumen. Der Adam Optimizer, mit seinem adaptiven Lernratenmechanismus, eignet sich besonders gut, um diese komplexen, hochdimensionalen Verlustlandschaften effektiv zu navigieren.

2. Learning Rate: 0,001.

3. Epochs: 15.

4. Batch Size: 64.

Die Auswahl der Hyperparameter für die Learning Rate, die Anzahl der Epochs und die Batch Size erfolgte durch Vergleich verschiedener Versuche. Hierbei wählten wir die Werte aus, die die besten Resultate erzielten.

Evaluation und Anomaliedetektion: Nach dem Trainingsprozess evaluierten wir das Modell hinsichtlich seiner Fähigkeit, die Input-Daten akkurat zu rekonstruieren. Wir verwendeten den Reconstruction Error als Metrik zur Detektion von Anomalien, ähnlich dem Ansatz von Zhang et al. Signifikant höhere Reconstruction Errors, ein Indikator für Abweichungen von den Standardbewegungen, die das Modell erlernt hat, wurden als potenzielle Anomalien identifiziert.

3.4.4 K-Means Clustering zur Identifizierung von Bewegungsmustern

Ein wichtiger Schritt in unserer Analyse war die Anwendung des K-Means Clustering-Algorithmus auf die latenten Repräsentationen der Daten, welche durch das Transformer Autoencoder-Modell generiert wurden. Ziel war es, zwei distinkte Bewegungsmuster in unserem Datensatz zu identifizieren und die Clusterzentren für die spätere Verwendung im DEC (Deep Embedded Clustering) Modell zu bestimmen.

Identifikation von Bewegungsmustern

Die latenten Repräsentationen der Trainingsdaten wurden mittels PCA (Principal Component Analysis) dimensionreduziert, um eine effiziente Clustering-Analyse zu ermöglichen. Auf diesen reduzierten Daten führten wir den K-Means Clustering-Algorithmus durch.

- Wir wählten zwei Cluster, um den zwei Hauptbewegungsmustern in unserem Datensatz zu entsprechen.
- Die Silhouettenanalyse unterstützte die Wahl von zwei Clustern, was auf eine klare Trennung der Bewegungsmuster hinweist.

Bestimmung der Clusterzentren für DEC

Die identifizierten Clusterzentren spielen eine wesentliche Rolle in unserem DEC-Modell. Sie dienen als anfängliche Zentren für das Deep Embedded Clustering, um eine genauere und spezifischere Anomalieerkennung zu ermöglichen.

- Die Clusterzentren wurden visualisiert und analysiert, um ein tieferes Verständnis der Datenstruktur und der inhärenten Muster zu erlangen.
- Diese Zentren bilden die Basis für das DEC-Modell, welches darauf abzielt, die Clusterzuordnung im Laufe des Trainings zu verfeinern und zu optimieren.

Zusammenfassend ermöglichte die Anwendung von K-Means Clustering auf die latenten Daten nicht nur die Identifikation von zwei grundlegenden Bewegungsmustern in unserem Datensatz, sondern auch die Vorbereitung der initialen Clusterzentren für das nachfolgende DEC-Modell.

3.4.5 DEC (Deep Embedded Clustering) Modellarchitektur

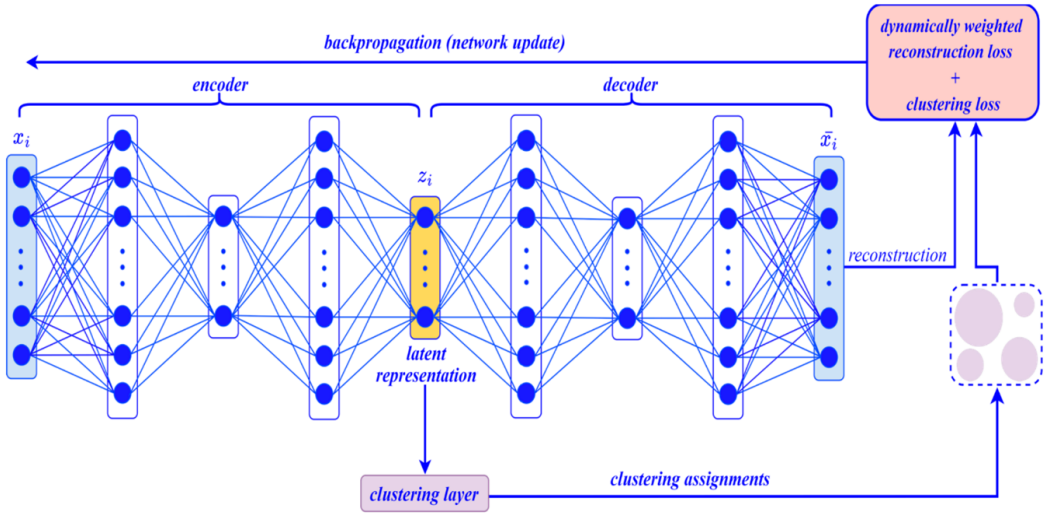

Abbildung 3.7: DEC Architektur, adaptiert von [4]

Das DEC (Deep Embedded Clustering) Modell ist ein entscheidender Bestandteil unserer Methodik zur Anomalieerkennung. Es kombiniert die Feature-Learning-Fähigkeiten unseres Transformer Autoencoders mit einer fortgeschrittenen Clustering-Methode. In diesem Abschnitt wird die spezifische Architektur und Funktionsweise des DEC-Modells beschrieben, wie es in unserem Ansatz implementiert wurde. Die Abbildung 3.7 zeigt den Aufbau des Models von Obied et al. welches von uns adaptiert wurde [4].

Architektur des DEC-Modells

Das DEC-Modell integriert eine *ClusterAssignmentLayer*, die auf Basis von initialen Clusterzentren, ermittelt durch K-Means Clustering, trainiert wird. Die Hauptkomponenten des Modells umfassen:

1. **ClusterAssignmentLayer:** Diese Schicht nutzt die initialen Clusterzentren als trainierbare Parameter. Sie berechnet eine *soft cluster assignment* für die Eingabedaten, wobei eine Student's t-Verteilung verwendet wird, um die Wahrscheinlichkeiten der Clusterzugehörigkeit zu bestimmen.

2. **Integration des Autoencoders:** Der Autoencoder wird in das DEC-Modell eingebettet, um die Eingabedaten zu kodieren und eine latente Darstellung zu erzeugen, die anschliessend für die Clusterzuweisung verwendet wird.

3. **Rekonstruktion und Clustering:** Neben der Erzeugung einer latenten Darstellung für die Clusterzuweisung rekonstruiert der Autoencoder auch die Eingabedaten, was eine simultane Optimierung von Datenrekonstruktion und Clustering ermöglicht.

Funktionsweise des DEC-Modells

Das DEC-Modell arbeitet in mehreren Schritten, um eine effiziente Anomalieerkennung zu ermöglichen:

- **Forward Pass:** Im Forward Pass werden die Eingabedaten durch den Autoencoder kodiert und rekonstruiert. Gleichzeitig wird eine *soft cluster assignment* für die latente Darstellung berechnet.
- **Target Distribution:** Auf Basis der Clusterzuweisungen berechnet das Modell eine *target distribution*, welche dazu dient, die Clusterzentren während des Trainings zu aktualisieren und zu optimieren.
- **Loss Function:** Die Kullback-Leibler-Divergenz zwischen den berechneten Zuweisungen und den Zielverteilungen dient als Loss Function. Diese Funktion führt zu einer kontinuierlichen Verbesserung der Clusterzuweisungen und der Erkennung von Anomalien.

Zusammengefasst ermöglicht das DEC-Modell eine präzise Clusterzuweisung und eine effektive Anomalieerkennung, indem es die starken Feature-Learning-Eigenschaften des Autoencoders mit einem ausgeklügelten Clustering-Algorithmus kombiniert.

Kapitel 4

Resultate

4.1 Transformer Autoencoder

Die Evaluierung der Modellleistung basierte auf dem Vergleich zwischen originalen und rekonstruierten Daten, wobei die Rekonstruktionsfehler als primäres Mass für die Genauigkeit des Modells dienten. Für das Trainingset wurde ein Rekonstruktionsfehler von 0.4398 ermittelt, während das Validierungsset einen deutlich geringeren Fehler von 0.2154 aufwies. Diese Ergebnisse lassen darauf schliessen, dass das Modell in der Lage ist, die charakteristischen Muster der Validierungsdaten effektiver zu erfassen und zu generalisieren, verglichen mit den Trainingsdaten.

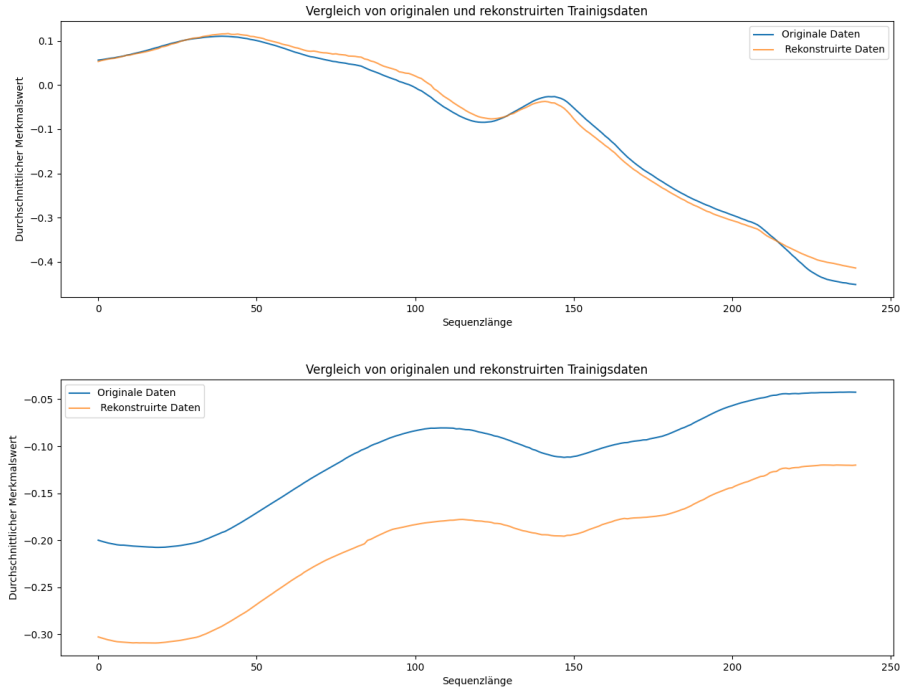

Abbildung 4.1: Stichproben für Vergleiche zwischen rekonstruierten und originalen Trainingsdaten aus dem Transformer Autoencoder Modell

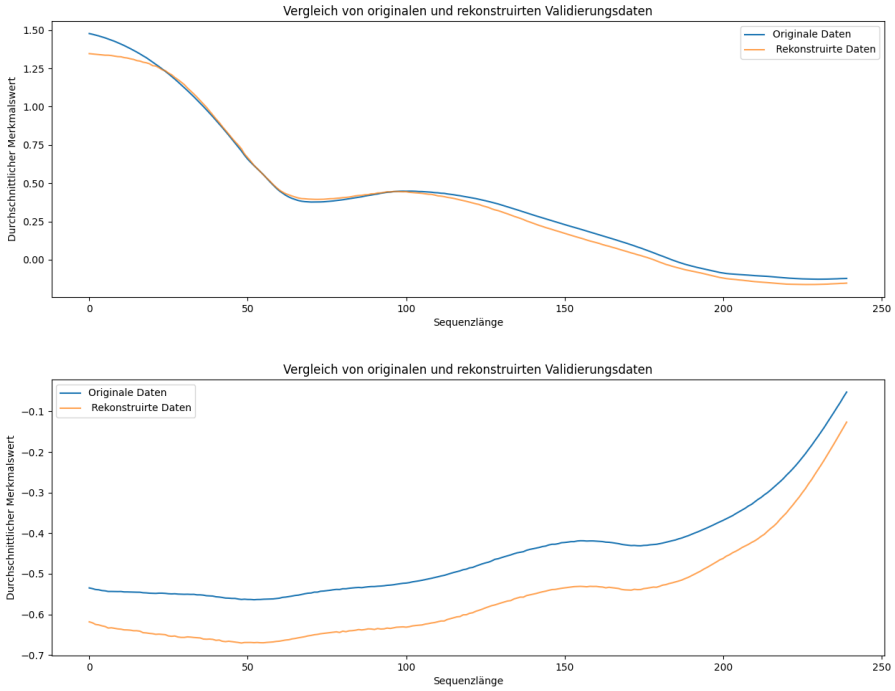

ABBILDUNG 4.2: Stichproben für Vergleiche zwischen rekonstruierten und originalen Validierungsdaten aus dem Transformer Autoencoder Modell

Eine detailliertere Analyse der Ergebnisse offenbarte, dass der Rekonstruktionsfehler zwischen den Trainings- und Validierungsdaten nicht gleichmässig verteilt war. Insbesondere wurden zwei zufällige Stichproben aus jedem Datensatz untersucht. Die erste Stichprobe aus den Trainingsdaten wies einen niedrigen Rekonstruktionsfehler auf, vergleichbar mit der ersten Stichprobe aus den Validierungsdaten, die ebenfalls einen minimalen Fehler zeigte. (siehe Abbildungen 4.1 und 4.2)

Im Gegensatz dazu zeigten die zweiten Stichproben sowohl aus den Trainings- als auch aus den Validierungsdaten höhere Rekonstruktionsfehler. Der Unterschied in der Leistung zwischen diesen und den zuvor genannten Stichproben wirft Fragen zur Konsistenz der Daten und zur Robustheit des Modells auf. Es ist möglich, dass die Sequenzen mit den höheren Fehlern Anomalien enthalten könnten, oder sie könnten auf schlechte Datenqualität, eine Komplexität, die das Modell nicht erfassen konnte, oder auf Rauschen hinweisen.

Bemerkenswert ist, dass der Validierungsdatensatz trotz seiner geringeren Grösse im Vergleich zum Trainingsdatensatz einen niedrigeren durchschnittlichen Rekonstruktionsfehler aufwies. Dies könnte darauf hindeuten, dass der Validierungsdatensatz weniger problematische Datensequenzen enthält, was die Fähigkeit des Modells beeinflusst, von den Trainingsdaten auf die Validierungsdaten zu generalisieren. Der geringere Fehler in den Validierungsdaten deutet darauf hin, dass das Modell genauer ist, wenn die Eingabedaten von höherer Qualität sind, was die Bedeutung von Datenreinheit und -konsistenz für das Training effektiver Machine-Learning-Modelle unterstreicht.

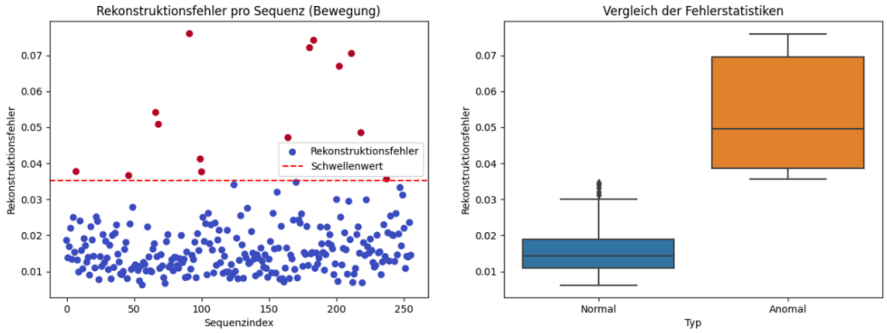

Abbildung 4.3: Analyse der Rekonstruktionsfehler der normalen und anomalen Testdaten

Der Mechanismus zur Anomalieerkennung, der in unserem Transformer-Autoencoder-Modell implementiert ist, wird in den Abbildungen 4.3 veranschaulicht. Das Modell setzt einen Schwellenwert für den Rekonstruktionsfehler fest; Sequenzen, die diesen Schwellenwert überschreiten, werden als anomal klassifiziert.

Die linke Grafik mit dem Titel 'Rekonstruktionsfehler pro Sequenz (Bewegung)' zeigt den Rekonstruktionsfehler für jede Sequenz, indiziert durch ihre jeweilige Position im Datensatz. Die Mehrheit der Sequenzen ist durch blaue Punkte dargestellt, was einen Rekonstruktionsfehler unterhalb des Schwellenwerts anzeigt, der durch die gestrichelte rote Linie dargestellt wird. Auffällig ist, dass eine kleine Anzahl von Sequenzen, markiert durch rote Punkte, einen Rekonstruktionsfehler aufweist, der diesen Schwellenwert übersteigt und sie somit als potenzielle Anomalien kennzeichnet.

Der Schwellenwert wurde empirisch bestimmt, um ein Gleichgewicht zwischen falsch positiven und falsch negativen Ergebnissen zu optimieren. Die Auswahl des Schwellenwerts ist ein kritischer Faktor für die Leistung des Modells, da er die Empfindlichkeit der Anomalieerkennung definiert.

Die rechte Grafik mit dem Titel 'Vergleich der Fehlerstatistiken' präsentiert ein Boxplot, das die Verteilung der Rekonstruktionsfehler für 'normale' und 'anomale' Sequenzen gegenüberstellt. Es ist ersichtlich, dass die 'normalen' Sequenzen, dargestellt in Blau, eine engere Verteilung mit einem niedrigeren Medianfehler aufweisen, während die 'anomalen' Sequenzen, dargestellt in Orange, einen breiteren Bereich mit einem höheren Medianfehler zeigen. Diese Trennung der Fehlerverteilungen bestätigt die Wirksamkeit des gewählten Schwellenwerts bei der Unterscheidung zwischen normalen und anomalen Sequenzen.

Das Vorhandensein von Ausreissern in den 'normalen' Daten zeigt, dass es Sequenzen mit marginal höheren Fehlern als die Mehrheit gibt, die jedoch den Schwellenwert nicht überschreiten. Umgekehrt deutet die breite Streuung in den 'anomalen' Daten darauf hin, dass einige Anomalien mit signifikant höheren Fehlern deutlich abweichen, während andere näher am Schwellenwert liegen.

Diese Analyse bestätigt, dass das Transformer-Autoencoder-Modell fähig ist, Anomalien in 3D-Bewegungsdaten zu erkennen, indem es den Rekonstruktionsfehler als diagnostisches Mass nutzt. Die Ergebnisse unterstreichen die Bedeutung einer angemessen festgelegten Schwelle in der Fehlerverteilung, um anomales Verhalten effektiv von normalen Variationen in den Daten abzugrenzen.

4.2 K-means Clustering

Abbildung 4.4: K-means-clustering auf Trainingsdaten

In der durchgeführten K-Means-Clusteranalyse auf den latenten Repräsentationen der Trainingsdaten wurden zwei Cluster identifiziert, die den one-hot-kodierten Bewegungen ‘Jumps’ und ‘Squat’ entsprechen sollen. Die resultierende Visualisierung zeigt, dass die Datenpunkte entlang der ersten beiden Hauptkomponenten gruppiert sind, was auf die zugrunde liegenden Unterschiede zwischen den beiden Bewegungsarten hinweist.

Der in Blau visualisierte Cluster zeichnet sich durch eine hohe Dichte an Datenpunkten aus, die sich um ein zentrales Clustering-Zentrum sammeln. Dies deutet darauf hin, dass das Modell die Bewegung ‘Squats’ relativ kohärent erfasst hat, wobei die Datenpunkte ähnliche Muster innerhalb dieser Bewegungsart aufweisen.

Im Gegensatz dazu weist der in Orange dargestellte Cluster eine breitere Streuung und geringere Dichte um das Clustering-Zentrum auf, was auf eine variablere Ausführung der Bewegung ‘Jumps’ schliessen lässt. Diese Beobachtung könnte auf unterschiedliche Stile oder Ausführungen der Sprünge innerhalb der Trainingsdaten hinweisen.

Die Isolierung einiger Datenpunkte, die weit abseits der beiden Hauptcluster liegen, könnte auf Ausreisser oder atypische Ausführungen der Bewegungen hindeuten. Diese könnten interessante Einblicke in Variationen der Bewegungsformen oder fehlerhafte Daten geben.

Die räumliche Trennung der beiden Clusterzentren, die in der zweidimensionalen Darstellung der Hauptkomponentenanalyse hervorgehoben wird, belegt die Kapazität des K-Means-Algorithmus, distinkte Bewegungsmuster zu erkennen. Obwohl die Gruppierung nicht äusserst präzise ist, konnten zwei zentrale Punkte identifiziert werden, die für die Weiterverarbeitung im DEC-Modell genutzt wurden.

Zusammenfassend spiegeln die Ergebnisse der K-Means-Clusteranalyse die Trennung der Trainingsdaten in zwei Bewegungstypen wider, auch wenn die Abgrenzung nicht scharf ist. Die erkannten Clusterzentren bildeten eine solide Grundlage für die darauf aufbauende DEC-Analyse und trugen dazu bei, die Datenstruktur im Hinblick auf die beiden spezifischen Bewegungsarten zu verstehen.

4.3 Deep Embedded Clustering

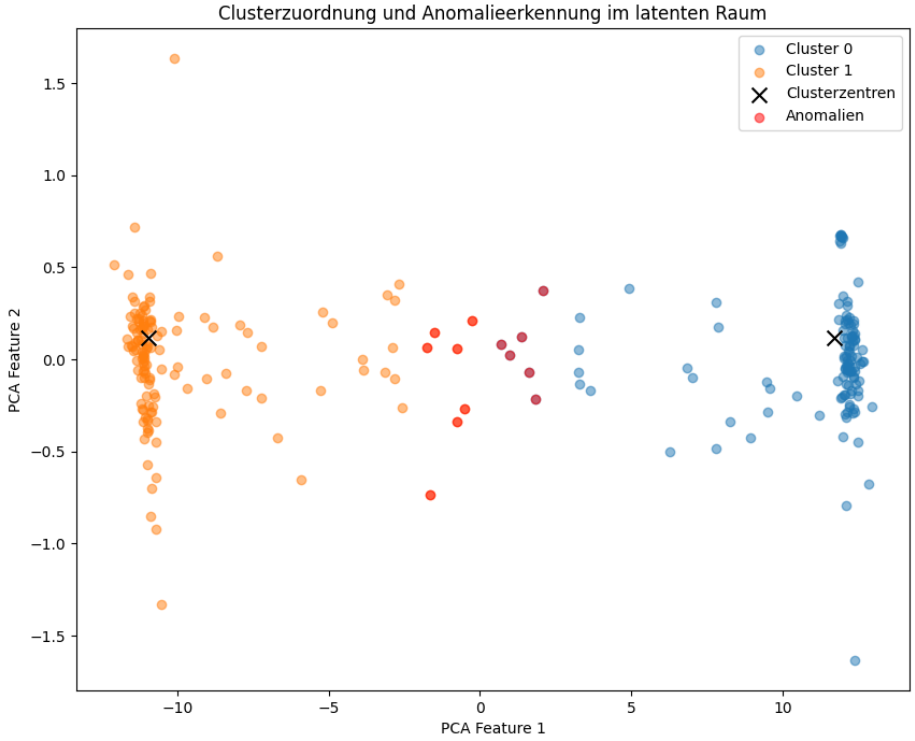

Abbildung 4.5: Deep Embedded Clustering auf Testdaten

Ein signifikantes Mass für die Qualität der Clusterbildung liefert der Silhouetten-Score, der für das DEC-Modell einen Wert von 0,52 erreicht. Dieser Wert liegt über dem Mittelpunkt des Wertebereichs und deutet darauf hin, dass die gebildeten Cluster eine moderate Abgrenzung und Zusammengehörigkeit der Datenpunkte aufweisen. Der Silhouetten-Score bestätigt somit, dass das DEC-Modell eine sinnvolle Segmentierung der Daten erreicht hat, wobei die Datenpunkte in der Regel näher an ihren eigenen Clusterzentren als an denen anderer Cluster liegen.

In der vorliegenden Abbildung 4.5 sind die Ergebnisse unseres DEC-Modells dargestellt, das auf den Clusterzentren des K-Means-Clustering basiert, um eine tiefere

Einbettung und Clusterbildung vorzunehmen. Die Visualisierung zeigt zwei Hauptcluster: Cluster 0 in hellblau und Cluster 1 in orange, mit den jeweiligen Clusterzentren, die durch Kreuze markiert sind. Die Anomalien sind durch rote Punkte gekennzeichnet und liefern eine direkte Visualisierung ihrer Verteilung in Relation zu den Clustern. Die DEC-Analyse zeigt eine deutliche Trennung der beiden Hauptcluster entlang der ersten beiden Hauptkomponenten, wobei die Clusterzentren die aggregierten Schwerpunkte innerhalb dieser neuen Merkmalsräume darstellen. Die roten Punkte, welche die Anomalien repräsentieren, befinden sich hauptsächlich an den Rändern der Cluster oder in den Übergangsbereichen, was auf ihre Position als Ausreisser in der Datenstruktur hinweist.

Kapitel 5

Diskussion

5.1 Bewertung der Modellergebnisse

Die vorliegende Studie hat gezeigt, dass das Transformer-Autoencoder-Modell in der Lage ist, gewisse Bewegungsabläufe zu erlernen. Dennoch sind die Ergebnisse, insbesondere im Bereich des Clustering, weniger eindeutig als erwartet. Die Clusterbildung fiel unscharf aus und die erwarteten klaren Grenzen zwischen verschiedenen Bewegungstypen konnten nicht deutlich herausgearbeitet werden.

5.2 Limitationen

Die Studie wurde durch mehrere limitierende Faktoren beeinträchtigt, die sich auf die Genauigkeit und Zuverlässigkeit der Modellresultate auswirken.

5.2.1 Datenaufbereitung und Segmentierung

Eine signifikante Einschränkung lag in der Qualität der Trainingsdaten. Die Herausforderung, 3D-Rohdaten ohne Zeitstempel adäquat zu segmentieren, hat sich als erhebliches Hindernis erwiesen. Die Abhängigkeit von fehleranfälligen Logfiles zur Segmentierung führt zu einer beträchtlichen Unsicherheit hinsichtlich der Konsistenz und Verlässlichkeit der Trainings- und Validierungsdaten.

5.2.2 Unscharfe Clustergrenzen

Die K-Means-Clusteranalyse konnte zwar zwei zentrale Punkte identifizieren, jedoch waren die Grenzen zwischen den Bewegungstypen ‘Jumps’ und ‘Squats’ nicht so deutlich, wie wir es erwartet hatten. Dieser Umstand wirft Fragen bezüglich der Diskriminierungsfähigkeit des Modells auf und limitiert seine Anwendbarkeit für präzise Anomalieerkennung.

5.2.3 Fehlende Validierungsdaten

Das Fehlen von Bewegungsdaten von Probanden mit bekannten inkorrekten Bewegungsausführungen aufgrund von Verletzungen stellt eine weitere wesentliche Limitation dar. Ohne diese Daten konnte das Modell nicht angemessen gegen bekannte Anomalien validiert werden, was die Aussagekraft der Ergebnisse einschränkt.

5.3 Ausblick

5.3.1 Datenqualität und Modellverbesserung

Die Qualität der Trainingsdaten ist für die Funktionsfähigkeit eines Autoencoder-Modells entscheidend. Zukünftige Arbeiten müssen die Datenvorbereitung verbessern und Methoden entwickeln, um fehlende oder ungenaue Zeitstempel in den Rohdaten zu kompensieren.

5.3.2 Potenzial für zukünftige Forschung

Trotz der ernüchternden Aspekte zeigen die Ergebnisse, dass das Modell Potenzial für die Identifizierung von Anomalien in 3D-Bewegungsdaten aufweist. Es bedarf jedoch einer sorgfältigen Überarbeitung der Datenaufbereitungsmethoden und einer Verbesserung der Datenqualität, um dieses Potenzial voll auszuschöpfen.

5.3.3 Schlussfolgerungen

Zusammenfassend lässt sich feststellen, dass die Ergebnisse der Studie gemischt ausfallen. Während das Modell einige Bewegungen erkennen und rekonstruieren konnte, waren die Resultate insgesamt nicht so eindeutig, wie erhofft. Die Studie zeigt, dass für die zuverlässige Anwendung von Machine Learning in der Bewegungsanalyse die Datenqualität von höchster Bedeutung ist. Es ist entscheidend, dass zukünftige Forschungsarbeiten die Datenaufbereitung und -segmentierung verbessern, um die Zuverlässigkeit der Modelle zu steigern.

Anhang A

Projekt Details

A.1 Projekt Titel und Beschreibung

Titel: Deep Learning für Erkennung von Änderungen des Bewegungsverhaltens bei sportlichen Übungen (Deep Learning for identification of changes in movement behavior during sport exercises)

Beschreibung: Im Bewegungsanalyselabor der ZHAW Gesundheit werden im Rahmen von klinischen Studien eine Vielzahl von Daten erhoben, einschließlich mehrdimensionaler Bewegungsdaten und Daten zur Muskelaktivität. Die Anwendung maschinellen Lernens ermöglicht ein tieferes Verständnis der zugrundeliegenden Prinzipien und könnte die auf Datenanalyse basierenden Schlussfolgerungen revolutionieren. Diese Arbeit nutzt kinematische Daten aus einem 25-minütigen Training, um Muster zu identifizieren, die auf Veränderungen im Bewegungsverhalten (wie Ermüdung) hinweisen. Die Ergebnisse unterstützen die Entwicklung neuer Ansätze in der sportlichen Rehabilitation.

Voraussetzungen: Python, Grundlagen des maschinellen Lernens

A.2 Quellcode

Der Quellcode dieser Arbeit ist verfügbar auf dem ZHAW Github unter folgendem link: github.zhaw.ch/loukibad/PA-Exercube-ZHAW.

Anhang B

Verzeichnisse

Literatur

[1] M. Zago, A. F. R. Kleiner und P. A. Federolf. „Editorial: Machine Learning Approaches to Human Movement Analysis“. In: *Frontiers in Bioengineering and Biotechnology* 8 (2021), S. 638793. DOI: 10.3389/fbioe.2020.638793.

[2] Benjamin Stern und Florian Witschi. *Machine Learning-Based Analysis of Data from the ZHAW Movement Analysis Laboratory for Fatigue Detection during Sports Exercises*. Project Work in Computer Science. Supervised by Prof. Dr. Thilo Stadelmann and Dr. Eveline Graf. Dez. 2022.

[3] Konstantinos Balaskas und Kostas Siozios. „Fatigue Detection Using Deep Long Short-Term Memory Autoencoders“. In: *2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. IEEE. 2021. DOI: 10.1109/MOCAST52088.2021.9493378.

[4] Muhamed Abdulhadi Obied u. a. „Deep Clustering-Based Anomaly Detection and Health Monitoring for Satellite Telemetry“. In: *Big Data and Cognitive Computing* 7.39 (2023). DOI: 10.3390/bdcc7010039.

[5] Junfeng Zhang u. a. „Power Consumption Predicting and Anomaly Detection Based on Transformer and K-Means“. In: *Frontiers in Energy Research* 9 (2021), S. 779587. DOI: 10.3389/fenrg.2021.779587.

[6] Ashish Vaswani u. a. „Attention Is All You Need“. In: *arXiv preprint arXiv:1706.03762* (2017).

[7] R. Bertani u. a. „Robotic Technologies and Rehabilitation: New Tools for Stroke Patients' Therapy“. In: *BioMed Research International* 2017 (2017), S. 153872.

[8] Fernando Bello Reza Haghighi Osgouei David Soulsby. „Rehabilitation Exergames: Use of Motion Sensing and Machine Learning to Quantify Exercise Performance in Healthy Volunteers“. In: *JMIR Rehabilitation and Assistive Technologies* 7.2 (2020), e17289. DOI: 10.2196/17289.

[9] D. M. Hawkins. *Identification of Outliers*. Bd. 11. Chapman und Hall, 1980.

[10] Zahra Zamanzadeh Darban u. a. „Deep Learning for Time Series Anomaly Detection: A Survey“. In: *arXiv* 2211.05244v2 (Dez. 2022). URL: https://arxiv.org/abs/2211.05244.

[11] Wikipedia contributors. *K-means clustering - Wikipedia*. https://en.wikipedia.org/wiki/K-means_clustering. Accessed: Dec. 19, 2023. 2023.

[12] Jim Frost. *What is K Means Clustering? With an Example*. Accessed: Dec. 16, 2023. 2023. URL: https://statisticsbyjim.com/basics/k-means-clustering/#more-16787.

[13] Junyuan Xie, Ross Girshick und Ali Farhadi. „Unsupervised Deep Embedding for Clustering Analysis“. In: *Proceedings of the 33rd International Conference on Machine Learning*. Bd. 48. 2016, JMLR.org.

[14] Vicon. *Vicon Vantage Camera Range*. https://docs.vicon.com/display/Vantage/Vicon+Vantage+camera+range. Zugriff am: Dec. 16, 2023. 2023.

[15] *C3D: The Industry Standard for 3D Motion Data*. https://www.c3d.org/. Zugriff am: Dec. 16, 2023. 2023.

[16] Suil Jeon, Kyoung Min Lee und Seungbum Koo. „Anomalous Gait Feature Classification From 3-D Motion Capture Data“. In: *IEEE Journal of Biomedical and Health Informatics* 26.2 (Feb. 2022). Manuscript received February 25, 2021; revised June 27, 2021; accepted July 25, 2021. Date of publication August 4, 2021; date of current version February 4, 2022. DOI: 10.1109/JBHI.2021.3101549.

[17] Scikit-learn developers. *sklearn.preprocessing.StandardScaler*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Zugriff am: Nov. 16, 2023. 2023.

[18] *pandas.get_dummies - pandas 1.3.5 documentation*. https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html. Zugriff am: Nov. 16, 2023.

[19] PyTorch Developers. *PyTorch Documentation*. Zugriff am: Nov. 16, 2023. 2023. URL: https://pytorch.org/docs/.

Abbildungsverzeichnis

Tabellenverzeichnis