Graph Neural Networks zur Vorhersage menschlicher Aktionen und Bewegungen in der Mensch-Roboter-Kollaboration bei industriellen Montageaufgaben

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des Karlsruher Instituts für Technologie (KIT) angenommene

DISSERTATION

von

M.Sc. Dimitrios Lagamtzis

Tag der mündlichen Prüfung: 12.05.2025

Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr.-Ing. Thao Dang

Danksagung

Diese Dissertation entstand während ich Doktorand am Karlsruher Institut für Technologie / Institut für Mess- und Regelungstechnik sowie wissenschaftlicher Mitarbeiter an der Hochschule Esslingen war. Betreut wurde die Arbeit zum einen von Herrn Prof. Dr.-Ing. Christoph Stiller, dem ich für die Übernahme der Betreuung, aber auch die wegweisenden Diskussionen und Anregungen sehr dankbar bin. Zum anderen wurde die Arbeit von Herrn Prof. Dr.-Ing. Thao Dang betreut. Bei Herrn Dang bedanke ich mich vor allem für die Gelegenheit, in einer für mich perfekten Arbeitsumgebung forschen zu können. Diese zeichnet sich durch exzellente fachliche Betreuung, spannende Diskussionen und Anregungen sowie der Möglichkeit zur persönlichen Entfaltung aus. Außerdem gilt ein großer Dank Herrn Prof. Dr.-Ing. Steffen Schober, der ebenfalls einen erheblichen Beitrag in der Betreuung geleistet und mich durch fachliche Diskussionen herausgefordert und unterstützt hat. Darüber hinaus möchte ich mich auch bei Prof. Dr. rer. nat. Markus Enzweiler für seine Unterstützung bedanken, der horizonterweiternd für Diskussionen und Gespräche zur Verfügung stand.

Herrn Prof. Dr.-Ing. Marcus Geimer möchte ich für den Vorsitz des Promotionsprüfungsausschusses danken.

Der Firma Festo und vor allem Dr.-Ing. Jan R. Seyler danke ich für die Unterstützung und den anwendungsbezogenen Rahmen meines Promotionsthemas. In diesem Zusammenhang möchte ich mich auch bei Dionysios Satikidis bedanken, ohne dessen Zutun, diese Kooperation nicht hätte erfolgreich stattfinden können.

Ich danke Herrn Fabian Schmidt herzlich für die Unterstützung, die wertvollen Diskussionen und Anregungen sowie das damit einhergehende geteilte Interesse an koffeinhaltigen Heißgetränken. Ebenfalls möchte ich mich bei den wissenschaftlichen Mitarbeitenden, Doktoranden und Doktorandinnen an der Hochschule Esslingen / Fakultät für Informatik und Informationstechnik, für spannende Diskussionen und Motivation bedanken. Insbesondere Herrn Philipp Köster und Herrn Prof. Dr.-Ing. Rainer Keller möchte ich für die Unterstützung

bei Fragen zu Rechenressourcen sowie den Zugriff darauf bedanken. Für die organisatorische Unterstützung und Hilfe bei der Erstellung eines Zeitplans möchte ich mich bei Frau Dr. iur. Alma Stankovic bedanken. Für die praktische Unterstützung im Labor möchte ich mich bei Herrn Peter Beltz und Herrn Udo Schlesinger bedanken.

Außerdem möchte ich mich auch bei den wissenschaftlichen Mitarbeitenden, Doktoranden und Doktorandinnen des Instituts für Mess- und Regelungstechnik, für die herzliche Aufnahme bedanken. Besonders Herrn Johannes Fischer, Herrn Royden Wagner und Herrn Jan-Hendrik Pauls möchte ich für die organisatorische Unterstützung danken.

Zuletzt der für mich wichtigste Teil: der Dank an meine Familie. Ein unendlicher Dank gilt meinen Eltern durch deren aufopfernde Erziehung und Unterstützung ich selbstbewusst und frei leben und meine hochgesteckten Ziele verfolgen konnte. In gleicher Weise möchte ich auch meiner Frau Lenja danken, ohne die das Gelingen dieser Arbeit nicht möglich gewesen wäre. Diese Arbeit ist meiner Tochter Frida gewidmet.

Kirchheim unter Teck, im Juni 2025 Dimitrios Lagamtzis

Kurzfassung

Die Kollaboration zwischen Mensch und Roboter ermöglicht das gemeinsame Lösen von Aufgaben, bei denen ein Mensch alleine an dessen Grenzen gelangt (zum Beispiel Anreichen eines benötigten Werkstücks bei Montageaufgaben). Grundvoraussetzung für eine intelligente Mensch-Roboter-Kollaboration ist die Befähigung des Roboters, den Menschen zu verstehen und folglich Entscheidungen aufgrund der momentanen und zukünftigen Aktion und der Bewegung des Menschen zu treffen.

In dieser Arbeit wird die Entwicklung einer Lernmethode für eine intelligente Kollaboration zwischen Mensch und Roboter im Kontext von Montageaufgaben in der Industrie untersucht. Konkret wird zunächst aus zeitlicher und räumlicher Szeneninformation ein Graph erzeugt, der die Zusammenhänge zwischen den Szenenobjekten und der Bewegung des Menschen und des Roboters enthält. Dieser Graph wird dann von einem Modell, das auf Graph Neural Networks basiert, zur Aktionserkennung und zur Vorhersage der Aktion und Bewegung des Menschen genutzt.

Es konnte gezeigt werden, dass durch die Verwendung von Graph Neural Networks eine Aktions- und Bewegungsvorhersage möglich ist. Auch die Kombination mehrerer Vorhersagen (im Sinne des Multi-Task-Learnings) kann dabei effizient ohne eine Abnahme der Modellgüte bereitgestellt werden. Die Methode wird dabei anhand von Baseline-Methoden und relevanten Datensätzen evaluiert, wobei durch das Collaborative Action Dataset auch ein reales Anwendungsbeispiel für die Mensch-Roboter-Kollaboration im industriellen Kontext hinzugezogen wird. Für die Bewegungsvorhersage ergibt sich dabei ein Final Displacement Error von weniger als 10cm für einen Vorhersagezeitpunkt von einer Sekunde. Bei der Aktionserkennung liefert die entwickelte Methode eine Verbesserung der Modellgüte im Vergleich zu anderen Ansätzen um bis zu 20%.

Abstract

The collaboration between a human and a robot enables jointly solving tasks, for which a human alone would reach its limits (e.g. passing a workpiece when performing assembly tasks). To enable intelligent human-robot collaboration and make suitable decisions, the robot must recognize and predict human actions and motion.

In this thesis learning methods for an intelligent human-robot collaboration in the context of industrial assembly tasks are examined. Initially, a graph is constructed given spatial and temporal scene information, which includes relations between scene objects and the human. This graph is then used by a graph neural network to predict the human's current and future action, as well as its motion trajectory.

It is shown, that action and motion prediction is possible using graph neural networks. In addition, the combination of multiple predictions (in the sense of multi-task learning) can be provided efficiently without a decrease in the accuracy of the model. The method is evaluated against baseline methods and with relevant datasets for the application at hand, including the Collaborative Action Dataset. The conducted experiments show that the developed method yields a final displacement error of less than 10cm for a prediction time of one second for motion forecasting. For action recognition, the results show an improvement in the accuracy of the model of up to 20% compared to other approaches.

Inhaltsverzeichnis

Abkürzungen und Symbole

Abkürzungen

CoAx	Collaborative Action Dataset
HPE	Human Pose Estimation
HOI	Human-Object Interaction
ROI	Region of Interest
MRK	Mensch-Roboter-Kollaboration
SEC	Semantic Event Chains
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GNN	Graph Neural Network
TCN	Temporal Convolutional Network
LSTM	Long Short-Term Memory
ERD	Encoder-Recurrent-Decoder
GRU	Gated Recurrent Unit
MLP	Multilayer Perceptron
GAN	General Adversarial Network
DCT	Discrete Cosine Transform

GCN	Graph Convolutional Network
GAT	Graph Attention Network
RGCN	Relational Graph Convolutional Network
RGAT	Relational Graph Attention Network
JK	Jumping Knowledge
GRNN	Graph Recurrent Neural Network
GGNN	Gated Graph Neural Network
ResGGCN	Residual Gated Graph ConvNet
MAE	Mean Absolute Error
MSE	Mean Squared Error
H2O	Hand-2-Object
O2O	Object-2-Object
ReLU	Rectified Linear Unit
PReLU	Parametric Rectified Linear Unit
ELU	Exponential Linear Unit
ADE	Average Displacement Error
FDE	Final Displacement Error
AR	Action Recognition
AP	Action Prediction
MF	Motion Forecasting
SMBO	Sequential Model-Based Optimization
TPE	Tree-Structured Parzen Estimator

SMAC	Sequential Model-Based Optimization for General Algorithm Configuration
Bimacs	Bimanual Actions Dataset
ZV	Zero Velocity
VBPP	Velocity-Based Position Projection
LLM	Large Language Models

1 Einleitung

Die Automatisierung in der Industrie hat sich in den letzten Jahren insbesondere durch den Einsatz von Robotern beschleunigt. Während mit traditionellen Robotern immer mehr Grenzen erreicht wurden, zum Beispiel bei Montageaufgaben, die manuell und Hand in Hand mit dem Menschen ausgeführt werden müssen, weil eine Automatisierung nicht wirtschaftlich ist [BBB+16], eröffnen kollaborative Roboter (engl. *collaborative robots*, wie in [CP99] eingeführt) durch die Unterstützung des Menschen neues Potenzial. Dieser Fortschritt ist in Abbildung 1.1 anhand der verschiedenen Stufen der Kooperation dargestellt.

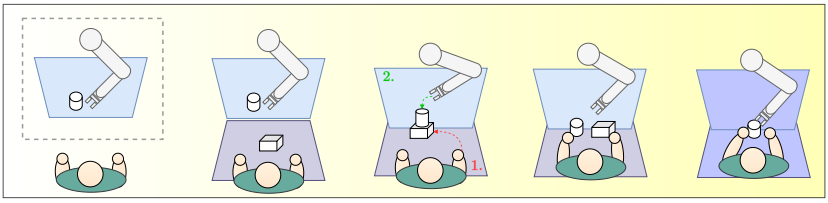


Abbildung 1.1: Stufen der Kooperation. Darstellung in Anlehnung an [BBB+16] und [MMZ+19].

Ein Roboterarm, der eine vordefinierte Aufgabe auf Basis sequentiell auszuführender Aktionen in einem eigenen abgetrennten Arbeitsbereich (auch: **Zelle**) durchführt, beschreibt eine getrennte Umgebung von Mensch und Roboter ohne jegliche Kollaboration. Die Aufhebung der räumlichen Trennung zwischen Mensch und Roboter, wobei beide Parteien einem dedizierten Arbeitsbereich zugeordnet sind und simultan gesonderte Aufgaben bearbeiten, um eigene Ziele zu erreichen, stellt eine **Koexistenz** zwischen Mensch und Roboter dar. Die **Synchronisation** bezeichnet ein gemeinsames Arbeiten von Mensch und Roboter in einem geteilten Arbeitsbereich zu unterschiedlichen Zeitpunkten. Das gleichzeitige, gemeinsame Arbeiten von Mensch und Roboter in einem geteilten Arbeitsbereich an unterschiedlichen Werkstücken bezeichnet die **Kooperation**

zwischen Mensch und Roboter. Um eine **Kollaboration** zwischen Mensch und Roboter handelt es sich, wenn zusätzlich zur Kooperation gemeinsam am demselben Werkstück gearbeitet wird [BBB+16].

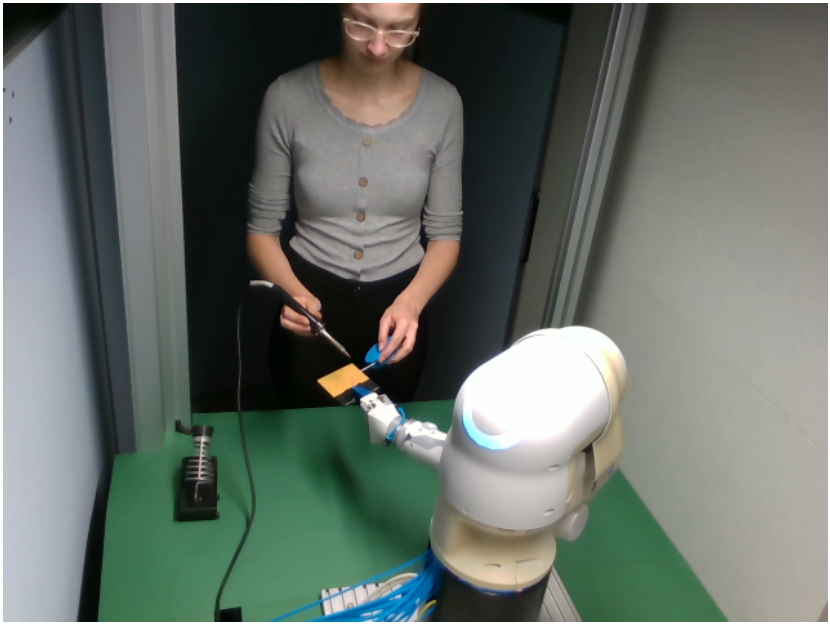


Abbildung 1.2: Abbildung einer exemplarischen industriellen Montageaufgabe im MRK-Kontext aus dem *Collaborative Action Dataset* [LSSD22] (CoAx). Dargestellt ist die Montageaufgabe „Kollaboratives Löten“, bei der ein kollaborativer Roboterarm den Menschen proaktiv beim Löten eines Kondensators auf ein Lötbrett unterstützt.

Durch diese Mensch-Roboter-Kollaboration (MRK) (vgl. Abbildung 1.1) sollen die Effizienz der kollaborativen Lösung von industriellen Montageaufgaben gesteigert, sowie die Kosten minimiert werden. Kollaborative Roboter haben in jüngster Vergangenheit sowohl in der Industrie als auch in der akademischen Welt besonders stark an Aufmerksamkeit und Interesse gewonnen, was dieses Vorhaben weiter bestärkt. Ein beispielhafter Aufbau einer MRK-Umgebung zur kollaborativen Lösung industrieller Montageaufgaben ist in Abbildung 1.2 dargestellt.

Roboter interagieren mit dem Menschen, beispielsweise zum Lösen von Montageaufgaben oder für medizinische Arbeiten. Dabei kombinieren sie die kognitiven Fähigkeiten des Menschen mit der Kraft und Wiederholbarkeit eines Roboters. Ein genereller Überblick zur MRK für Montageaufgaben, welche im Fokus dieser Dissertation stehen, wird in den Arbeiten [BBB+16], [MMZ+19] und [LZL+23] gegeben.

Um die Potenziale der Kraft und Wiederholbarkeit eines kollaborativen Roboters auszuschöpfen und ein gemeinsames Ziel von Mensch und Roboter zu erreichen, das durch die Absicht des Menschen vorgegeben ist [BWB08], muss der Roboter zum proaktiven Handeln befähigt werden. Daher spielt die Vorhersage des menschlichen Verhaltens eine wichtige Rolle und wird im nachfolgenden Kapitel 1.1 als Motivation dieser Arbeit beschrieben.

1.1 Motivation

Bei einer MRK für industrielle Montageaufgaben muss ein Roboter die Intention des Menschen erkennen. Dies beinhaltet zum einen zu verstehen, welche Aktion ein Mensch tätigt (*Aktionsvorhersage*) und zum anderen, wohin sich ein Mensch bewegen wird (*Bewegungsvorhersage*). Durch die Erkennung und Vorhersage von Aktionen kann wichtiges Wissen über die Intention des Menschen für den Roboter erlangt werden, um somit eine vorausschauende Unterstützung zu ermöglichen. Mit der Bewegungsvorhersage des Menschen kann wiederum die Bahnplanung des Roboters proaktiv zur Kollisionsvermeidung mit dem Menschen angepasst werden.

Dazu muss jegliche verfügbare Information in einer solchen MRK-Umgebung, die beispielsweise mit einer Stereokamera realitätsgetreu erfasst werden kann, genutzt werden. Dies umfasst zusätzlich die Szenenobjekte, wie zum Beispiel Werkzeuge oder Werkstücke, den kollaborativen Roboter und die beteiligten menschlichen Akteure sowie deren impliziten Relationen zueinander. Diese inhärente Information ist prädestiniert für die Prädiktion der menschlichen Intention in wissensbasierten Lernansätzen (vgl. Kapitel 4).

Für den vorliegenden Anwendungsfall von Montageaufgaben im Industrie-Kontext ist sowohl die Detektion des Menschen als auch der Szenenobjekte effizient umsetzbar, da die Erkennung des Menschen ein umfassend erforschter

Bereich ist, robust für die Vorhersage verschiedener Menschen generalisiert und leicht anwendbar ist. Auch für die Erkennung der Szenenobjekte gilt eine Vereinfachung, da diese durch eine definierte Montageanleitung feststehen und eingeplant sind und es somit keine unbekannten oder sich verändernden Objekten gibt. Auch die einzelnen Schritte und somit resultierenden Aktionen die der Mensch tätigt, sind präzise beschrieben.

1.2 Ziel und Beiträge der Arbeit

Das **Ziel** dieser Arbeit ist die Entwicklung einer Lernmethode für eine intelligente Kollaboration zwischen Mensch und Roboter im Kontext von Montageaufgaben in der Industrie, die auf der Kenntnis der Szene und der Umgebung basiert. Dazu sollen die Lernaufgaben *Aktions-* und *Bewegungsvorhersage* umgesetzt werden, wobei die implizite und explizite in der Szene vorhandene Information vordergründig verwendet werden soll. Darüber hinaus soll erarbeitet werden, ob mehr als nur eine Lernaufgabe gleichzeitig gelernt oder aber von einem einzelnen lernenden System gleichzeitig bereitgestellt werden kann.

Die **Beiträge** der Arbeit lassen sich wie folgt zusammenfassen:

- Vorstellung einer neuartigen Strukturierung der Szeneninformation in einer MRK-Arbeitsumgebung durch die Modellierung eines Graphen, in dessen Mittelpunkt die Hand des Menschen steht, mit welcher eine Aktion assoziiert wird.
- Erarbeitung einer *Graph Neural Network* (GNN)-basierten Methode zur Aktions- und Bewegungsvorhersage, wobei diese Vorhersagen unabhängig oder kombiniert gelernt werden können. Unabhängig bedeutet hierbei, dass für jedes Lernziel ein Modell trainiert werden muss. Kombiniert beschreibt hingegen a) einen mehrstufigen Ansatz nach der Theorie des Transferlernens und b) einen geteilten Encoder-Ansatz nach der Theorie des Multi-Task-Lernens.
- Darstellung der leichten Erweiterbarkeit der erarbeiteten Methode für das effiziente Lernen mehrerer Lernaufgaben ohne Verschlechterung der Vorhersagegüte. Zudem wird die Erweiterbarkeit durch datenge-

triebene Optimierung der Verlustkombination mehrerer Lernaufgaben demonstriert.

- Erstellung eines Datensatzes zum Lernen von Aktionen und Bewegungen eines Menschen in einer MRK-Umgebung für Montageaufgaben in der Industrie.

1.3 Struktur der Arbeit

Die weitere Arbeit ist wie folgt strukturiert. Der relevante Stand der Technik wird in Kapitel 2 vorgestellt. Die theoretischen Grundlagen von GNNs werden als wesentlicher Bestandteil der lernenden Komponente, welche der in dieser Arbeit entwickelten Methode entspricht, in Kapitel 3 ausführlich erläutert. Auf Basis dieser Grundlagen und dem Wissen über den aktuellen Stand der Technik wird die erarbeitete Methode zur Aktions- und Bewegungsvorhersage in Kapitel 4 dargelegt. Die Erweiterung der grundlegenden Vorhersagemethoden, vielmehr die kombinierte Vorhersage, wird daraufhin in Kapitel 5 ausgeführt. Die in Kapitel 4 und 5 erarbeiteten Vorhersagemethoden werden in Kapitel 6 evaluiert, wobei auf die in der Evaluation berücksichtigten Datensätze und Vergleichsmethoden eingegangen wird. Abschließend wird die Arbeit in Kapitel 7 zusammengefasst und ein Ausblick gegeben.

2 Stand der Technik

Eine kollaborative Zusammenarbeit zwischen Mensch und Roboter bedarf mehrerer intelligenter Komponenten, die in klassischen Roboterarchitekturen in die Bereiche „Sense-Plan-Act“ untergliedert werden. „Sense“ beinhaltet in dieser Arbeit die Szenenwahrnehmung und -Repräsentation (vgl. Kapitel 2.2 und 2.3). „Plan“ bezeichnet die Ableitung einer Handlungsstrategie. Zentral ist hier nach [LFS17] die Fähigkeit, Vorhersagen zur Intention und Bewegung des Menschen zu treffen. „Act“ entspricht der Ausführung der Strategie durch Bewegung des Roboterarms und Betätigung des Greifers. Dies wird in dieser Arbeit nicht weiter berücksichtigt. Abbildung 2.1 veranschaulicht diesen Zusammenhang anhand eines Robotersystems.

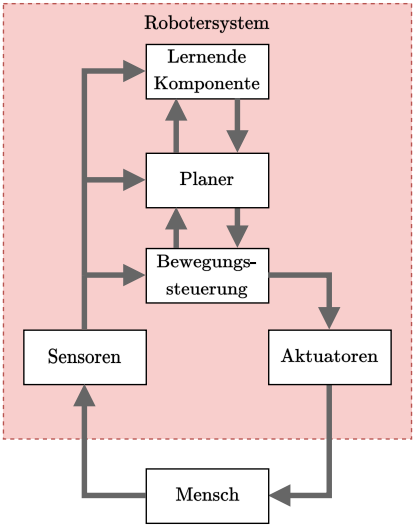


Abbildung 2.1: Robotersystem vereinfacht mit substituierbarer „lernender Komponente“. Vereinfachte Darstellung in Anlehnung an [SHSW05].

Für ein besseres Verständnis wird zunächst auf Systeme für die Kollaboration von Mensch und Roboter eingegangen. Danach werden gängige Methoden zur Wahrnehmung der Szene inklusive der darin agierenden Menschen und ihrer Repräsentation vorgestellt. Anschließend wird auf wichtige Arbeiten zu den Vorhersagemethoden für die Erkennung bzw. der Vorhersage der Intention und Bewegung von Menschen eingegangen.

2.1 Systeme für die Kollaboration von Mensch und Roboter

Zur Übersicht möglicher Systeme für die MRK werden in diesem Kapitel Systeme vorgestellt, welche die Vorhersage der ausgeführten Aktion eines Menschen, die Vorhersage der zukünftigen Bewegung des Menschen, oder aber einer Kombination dieser umsetzen. Dabei wird der Fokus auf den schematischen Aufbau dieser Systeme gesetzt. Bei der Auswahl dieser Systeme handelt es sich teilweise um Arbeiten, die auch im Kontext der Vorhersagemethoden (vgl. Kapitel 2.4) beschrieben werden, hier aber als „Lernende Komponente“ in einem Robotersystem auftreten.

Bereits in frühen Arbeiten von [SHSW05] wird eine Systemarchitektur zur Erkennung von menschlichen Intentionen beschrieben. Dieses Robotersystem ist geschlossen und verfügt über Sensoren, welche die Umgebung wahrnehmen und als Information für die Bewegungssteuerung, sowie dem Planer zur Verfügung stehen, als auch notwendige Informationen für die Intentionserkennung bereitstellen. Zusammen mit einer vordefinierten Datenbank, die Informationen über die Umgebung, wie beispielsweise der Position oder der Form von Umgebungsobjekten enthält, werden auf Basis Bayes'scher Statistik, Intentionen klassifiziert. Diese erkannten Intentionen können dann vom Planer verwendet werden, um eine Bewegungsentscheidung zu treffen, welche letztendlich von den Aktuatoren ausgeführt wird. Der Mensch steht dabei in einer Wechselwirkung mit diesem System.

Solch ein System stellt generell die Basis für eine Interaktion, genauer für eine intelligente MRK, dar. Dabei kann die intelligente Komponente, welche Schlüsse aus der Umgebung bereitstellt, durch verschiedene lernbasierte Ansätze ausgetauscht werden. Die in Kapitel 4 und 5 vorgestellten Methoden sind

Beispiele für solche Ansätze. Ein mögliches System für eine intelligente MRK kann vereinfacht, wie in Abbildung 2.1 dargestellt werden.

Eine ähnliche Abstraktion für die Integration einer lernenden Komponente für die Bewegungsvorhersage des Menschen und dementsprechend einer MRK in der Industrie, wird in der Arbeit von [ULT+18] vorgestellt. Das sogenannte menschenbewusste Robotersystem (engl. *Human-Aware Robotic System*) besteht dabei aus drei Modulen: der gemeinsamen Umgebung von Mensch und Roboter, der Komponente, welche die physikalischen Sensoren, inklusive der Detektion und Verfolgung der Menschen in der Umgebung beinhaltet, sowie einer Komponente, welche zuständig für die *Vorhersage* der Bewegung des Menschen und die Planung und Steuerung der Bewegung des Roboters ist, welche durch Algorithmen umgesetzt wird. Auch in dieser Arbeit ist eine ähnliche Aufteilung festzustellen, wie in Abbildung 2.1 gezeigt. Für die Vorhersage der Bewegung wird die Methode von [LS17] angewandt, welche ein Ansatz mit mehreren Prädiktionsmethoden (*mutliple-predictor systen* (MPS)) für eine Vorhersage ist.

In der Arbeit von [CSLT20] wird ein MRK-System vorgestellt, welches planungsbasiert mit der vorhergesagten Bewegungsklasse mittels *Long Short-Term Memory* (LSTM) eine Aktion inferiert, um Bewegungsentscheidungen zu treffen. Außerdem wird eine Bewegungsvorhersage des Menschen auf Basis einer rekursiven Least-Squares-Schätzung [CZLT19] integriert, um zusätzlich zum entschiedenen Plan eine mögliche Kollision mit dem Menschen zu vermeiden. Auch bei dieser Arbeit finden sich wiederkehrende Komponenten in den zuvor beschriebenen Robotersystemen. Die Unterscheidung liegt hierbei erneut im Gestaltungsraum der lernenden Komponente, welche final die Informationen für die Bewegungssteuerung liefert. Dabei werden hauptsächlich die ausgeführte Aktion und die Bewegung des Menschen als wichtige Information durch die lernende Komponente vorhergesagt.

2.2 Szenenwahrnehmung

Eine Umgebung und die darin stattfindenden Interaktionen mit Hilfe von Sensoren und Algorithmen zu erfassen und zu interpretieren, um Schlussfolgerungen zu ziehen, wird als Szenenwahrnehmung bezeichnet. Neben dem offensichtlichen Zweck die Szene wahrzunehmen, wird die Szenenwahrneh-

mung auch zur Aufzeichnung repräsentativer Daten benötigt, welche wiederum für Lernalgorithmen verwendet werden. Besonders im Kontext der MRK, in der im dreidimensionalen Raum interagiert wird, ist es wichtig, diese räumliche Information aufzuzeichnen. Neben der Farbinformation ist für viele Robotikaufgaben, wie zum Beispiel das Greifen von Objekten oder für eine kollisionsfreie Bewegung, Tiefeninformation hilfreich [BCCI+21]. Für Szenenobjekte in einer MRK-Umgebung ist neben der Positionsinformation auch eine zusätzliche Klassifikation der jeweiligen Objektklasse zur Identifikation nützlich. Im Folgenden werden Sensoren zum Aufzeichnen einer solchen Szene beschrieben, wie auch Methoden zum Erkennen des Menschen und seiner Pose sowie die Pose und Klassifikation von Objekten.

Die Szenenwahrnehmung bildet die Basis für lernende Komponenten und Algorithmen, die einen Roboter zur Entscheidungsfindung befähigen. Ein weiterer Aspekt aus der Anwendungssicht ist die Anzahl der benötigten Sensoren und die Art der Sensorik, um hinreichende Information für eine Szenenwahrnehmung bereitzustellen. Insbesondere im Industrie-Kontext besteht die Anforderung der Verwendung einer minimalen Anzahl an Sensoren aus Gründen der Skalierbarkeit, der verfügbaren Hardware-Ressourcen zum Betreiben dieser und der damit verbundenen Kosten.

Für die Szenenwahrnehmung werden auf unterster Ebene Sensoren benötigt, die eine Aufnahme der Umgebung ermöglichen. Zu den gängigen Sensoren für autonome Systeme gehören Kameras, Lidare, Radare, Ultraschallsensoren, Inertialsensoren und GPS-Sensoren [COK+18]. Für die Szenenwahrnehmung in der Kollaborationsumgebung von Mensch und Roboter sind besonders Sensoren von Vorteil, welche leicht montiert werden und möglichst viel Information in der Szene auf einmal aufzeichnen können, um alle szenenteilnehmenden Objekte zu lokalisieren und klassifizieren.

In dieser Arbeit wird eine Stereokamera verwendet, da diese einfach zu montieren und zu handhaben ist und eine umfassende Erfassung aller Objekte in der Szene ermöglicht. Darüber hinaus können mit Hilfe einer Stereokamera sowohl räumlich versetzte RGB-Bilder aufgenommen als auch Tiefeninformation daraus berechnet werden, sodass die komplette Szene im dreidimensionalen Raum beschrieben werden kann. Exemplarisch wird dies in Abbildung 2.2 mit Bilddaten des CoAx-Datensatzes [LSSD22] (vgl. Kapitel 6.2.1) dargestellt. Die Verwendung einer einzelnen Stereokamera erfüllt zusätzlich die Anforderungen

an die Verwendung einer minimalen Anzahl an Sensoren und gewährleistet Skalierbarkeit.

(a) (b) (c)

Abbildung 2.2: Dargestellt ist die Information, die von der Stereokamera aufgezeichnet wird: RGB-Bild (a) und Tiefenbild (b), sowie die abgeleitete 3D Punktwolke (c). Bei den Bildern handelt es sich exemplarisch um eine Aufnahme aus dem Collaborative Action Dataset [LSSD22] (vgl. Kapitel 6.2.1).

2.2.1 Erkennung des Menschen und dessen Pose

Mit der Erkennung des Menschen bzw. dessen Pose (engl. *Human Pose Estimation (HPE)*) ist das Modellieren des menschlichen Körpers aus Bilddaten und die Lokalisierung dessen Gelenke (engl. *keypoints*) gemeint [SBIK16]. Aus den lokalisierten Gelenkpunkten wird folglich eine Struktur abgeleitet, welche die Pose des Menschen darstellt.

Wie in [CTH20] und [MJW+20] dargelegt, wird diese Fähigkeit in unterschiedlichen Disziplinen genutzt, die dem maschinellen Sehen zugeordnet werden. Zu diesen gehören beispielsweise die Fußgängererkennung beim automatisierten Fahren [Gav99] oder die Erkennung des Menschen für die Interaktion zwischen Roboter und Mensch [KS13b], aber auch die Bewegungsanalyse des Menschen im medizinischen Kontext, sowie die Unterstützung im betreuten Wohnen oder von Menschen mit Beeinträchtigungen.

Im Folgenden werden Methoden aus dem Stand der Technik zur gelenkbasierten Schätzung der menschlichen Pose anhand der etablierten Frameworks OpenPose [SJMS17], [CHS+21] und MediaPipe [LTN+19], [ZBV+20] erläutert. Interessierte Lesende können alle Einzelheiten in den Folgenden Arbeiten [CTH20], [MJW+20], [SBIK16], [LZBC15] und [Gav99] entnehmen.

Grundsätzlich kann bei den Methoden zur Schätzung der Gelenkpunkte des menschlichen Körpers nach bestimmten Charakteristika der Daten und der Schätzung differenziert werden. Dabei wird zwischen der Anzahl der Menschen, für die eine Pose bestimmt werden soll (Einzel- oder Mehrpersonen-Posenschätzung), der Art der Bildinformation (RGB- oder RGBD-Bild[1]) und der Dimensionen der zuletzt geschätzten Positionen (2D- oder 3D-Koordinaten der Gelenkpunkte), unterschieden [SBIK16], [CTH20].

Die Modellierung des menschlichen Körpers kann in skelettbasierte, konturbasierte und volumenbasierte Modelle unterteilt werden [CTH20].

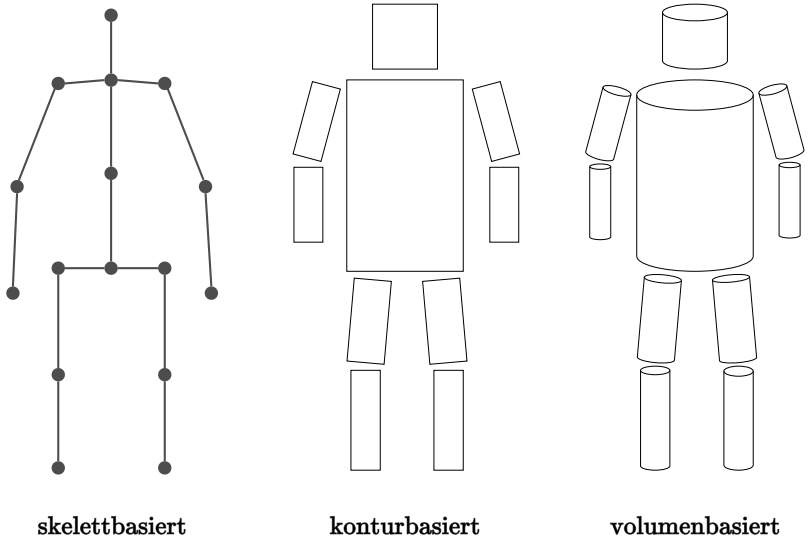


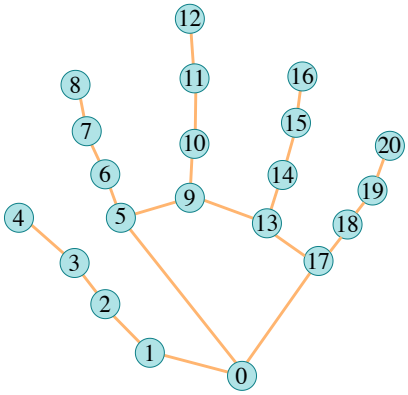
Abbildung 2.3: Arten, den Körper eines Menschen zu modellieren. Körper modelliert als Skelett, durch planare Konturen oder durch volumenbasierte Körper. Darstellung adaptiert in Anlehnung an [CTH20].

Bei der skelettbasierten Modellierung wird der menschliche Körper durch Gelenke, die in einer Skelettstruktur angeordnet werden, repräsentiert. Durch

[1] Ein Bild, das Pixel-Informationen über die Farbkanäle **R**ot, **G**rün und **B**lau sowie über die Tiefe (engl. ***D**epth*) enthält.

diese strukturierte Repräsentation kann einfach und flexibel eine Posenschätzung durchgeführt werden. Aus diesem Grund ist die skelettbasierte Modellierung Stand der Technik in der Posenschätzung und wird in dieser Arbeit verwendet.

Die Darstellung des menschlichen Körpers mittels eines Skelettmodells könnte um eine detailliertere Ausarbeitung der Handstrukturen ergänzt werden. Dabei werden zusätzliche Gelenkpunkte definiert, für welche ebenfalls die Hand-Pose bestimmt wird (vgl. MediaPipe [LTN+19], [ZBV+20] und OpenPose [SJMS17], [CHS+21]).



0	Handgelenk
1 bis 4	Daumen
5 bis 8	Zeigefinger
9 bis 12	Mittelfinger
13 bis 16	Ringfinger
17 bis 20	kleiner Finger

Abbildung 2.4: Keypoints der Hand in Anlehnung an [Goo23] (vgl. MediaPipe [LTN+19], [ZBV+20] und OpenPose [SJMS17], [CHS+21])

2.2.2 Erkennung der Objekte

Neben einer hinreichenden Erkennung des Menschen und dessen Pose in einer Kollaborationsumgebung von Mensch und Roboter, sind die darin enthaltenen Objekte ebenfalls von besonderer Bedeutung und müssen daher zu jedem Zeitpunkt erkannt und lokalisiert werden. Die Objekterkennung stellt eine weitere Grundlage für das Lernen aus dem Kontext einer Szene dar, in der ein Mensch mit Objekten interagiert (engl. *Human-Object Interaction (HOI)*) [GGDH18].

Wie in der Übersichtsarbeit [JZL+19] dargestellt, gibt es grundsätzlich verschiedene Arten der Objekt- und Objektklassendetektion. Unter Verwendung eines RGB-Bildes als Beobachtung wird zwischen verschiedenen Arten der Objekterkennung unterschieden: Der Klassifikation einzelner oder mehrerer Objektklassen, die innerhalb des Bildes vorhanden sind; der Klassifikation anhand von Begrenzungsrahmen, die einen spezifischen Bereich von Interesse (engl. *Region of Interest (ROI)*) umfassen und für die eine Objektklassifikation erforderlich ist; der Segmentierung des Bildes, sei es durch Objekte (auch: Instanzsegmentierung engl. *instance segmentation*) [HGDG17] oder anhand der Bildpixel (auch: semantische Segmentierung engl. *semantic segmentation*) [LSD15]; sowie der Kombination der beiden letztgenannten Arten der Bildsegmentierung - der panoptischen Segmentierung (engl. *panoptic segmentation*) [KHG+19].

Für die zugrundeliegende Arbeit ist eine zuverlässige Objekterkennung in Form von Begrenzungsrahmen bereits ausreichend, da die abgeleitete Positionsinformation nicht für Aufgaben verwendet wird, die eine exakte Pose erfordern, wie zum Beispiel Greifaufgaben. Die aus der vorhandenen Tiefeninformation bestimmte 3D-Position wird als Kontextinformation für die Aktions- und Bewegungsvorhersage verwendet, um die Szene und die darin enthaltenen Relationen zwischen Mensch, Roboter und Objekten zu beschreiben.

Des Weiteren können Detektoren abhängig von ihrer Architektur und dem verwendeten Grundgerüst (engl. *backbone*), in einstufige (engl. *one-stage*) oder zweistufige (engl. *two-stage*) Detektoren unterschieden werden. Besonders prominente Beispiele sind YOLO [RDGF16] und Mask-RCNN [HGDG17]. Beide Detektoren liefern für die Anwendung der Objektklassifikation und Lokalisierung im MRK-Kontext hinreichend gute Ergebnisse, wenn die Modelle mit genügend Daten trainiert wurden.

2.3 Szenenrepräsentation

Die Szenenwahrnehmung gibt die Möglichkeiten zur Repräsentation einer Szene vor. So gibt es beispielsweise im Kontext der Aktionserkennung von Menschen für Datensätze folgende Modalitäten, die anhand der wahrgenommenen Daten unterschieden werden: RGB-Bild, Tiefenbild, Skelett-Information (Positionsdaten bzw. korrespondierende Bildpixel) oder Kombinationen daraus. Zum

Beispiel Datensätze auf Basis von RGB-Bildern [KCS+17] oder RGB-Bildern mit Tiefenbildern und der Skelett-Information [SLNW16], [LSP+20], [DWA20], [LSSD22]. Die gleichen Bedingungen gelten auch für die Bewegungsvorhersage von Menschen [IPOS14].

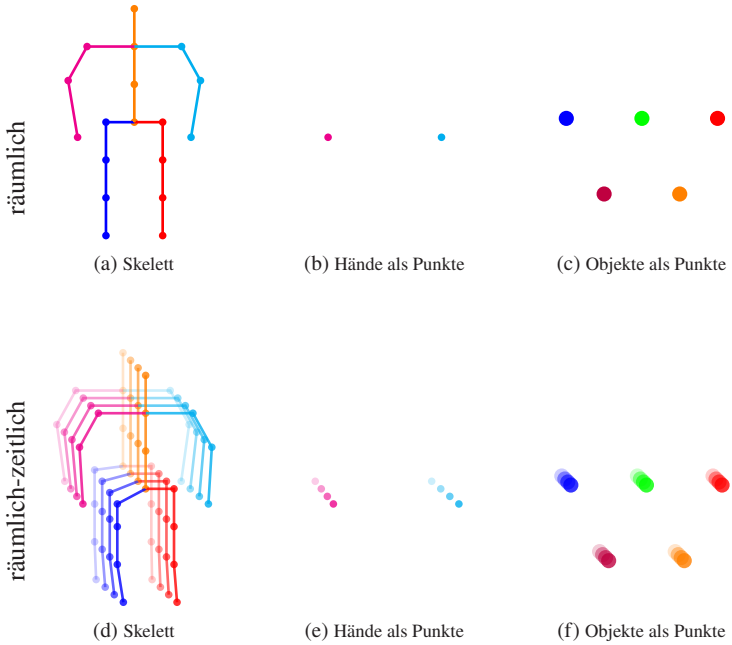
Die Szenenrepräsentation ist von besonderer Bedeutung, da diese den Speicherbedarf und die Trainingsdauer für lernende Systeme in Abhängigkeit der Datenmodalität beeinflusst. Wenn RGB-Bilder oder Tiefenbilder direkt für das Training verwendet werden, ist die benötigte Speicherkapazität um ein Vielfaches größer, als wenn beispielsweise direkt mit der extrahierten Skelett-Information des Menschen gearbeitet wird. Die Repräsentation der Information kann somit implizit (im Bild enthalten) oder explizit (aus dem Bild extrahiert) dargestellt werden. Ein weiteres Beispiel für eine explizite Darstellung neben der Skelett-Information ist die Verwendung von Trajektoriendaten, die aus zuvor detektierten Objekten extrahiert wurden.

Die Abstraktion des Menschen auf eine einzige Repräsentation, wie beispielsweise dessen Hand, kann eine weitere Gestaltungsmöglichkeit darstellen. Gerade im Kontext der MRK bei industriellen Montageaufgaben ist der Arbeitsbereich oftmals beschränkt und die unmittelbare Bewegung des Menschen geht primär aus der Handposition hervor. So kann die Abstraktion des Menschen auf die Handpose als Repräsentation zusätzlich hilfreich sein, anstatt auf das gesamte menschliche Skelett zurückgreifen zu müssen, zumal es bei beschränktem Sichtfeld nicht permanent vollständig verfügbar ist. In der folgenden Abbildung 2.5 sind die Möglichkeiten einer Szenenrepräsentation von Mensch, Roboter und Objekten innerhalb einer MRK-Arbeitsumgebung dargestellt.

Wie in der Abbildung 2.5 dargestellt, gibt es verschiedene Möglichkeiten eine Szene mit extrahierten Daten/Merkmalen explizit zu repräsentieren. Menschen, die in einer Szene wahrgenommen werden, können als Skelett repräsentiert werden [LSXW16], [KBA+17], [GZW+18], [YXL18], [CPAM20], [CSY20], [LCPW21], [LYFG21], [TXM+21], [ZVVM21], [XB22a] oder aber auch abstrakter durch ihre Handposition [AAD+11], [ZR17], [LM19], [DWA20], [LTMW22]. Außerdem können beispielsweise die Objektinformationen in Kombination mit dem Skelett des Menschen oder der Hand repräsentiert werden.

Aus den Daten, die mittels der beschriebenen Methoden in Kapitel 2.2.1 und 2.2.2 extrahiert werden, lässt sich eine Trajektorie des Menschen oder

repräsentativ für den Menschen durch dessen Hand darstellen. Die weitere Verarbeitung und das Lernen mit diesen Informationen kann dann in Form von Zeitreihen stattfinden (vgl. Kapitel 2.4). Zusätzlich können die erkannten Objekte in Form von weiteren Zeitreihen berücksichtigt werden. Eine solche Abbildung stellt nicht nur den räumlichen sondern auch den zeitlichen Verlauf der Szene dar.



(a) Skelett (b) Hände als Punkte (c) Objekte als Punkte

(d) Skelett (e) Hände als Punkte (f) Objekte als Punkte

Abbildung 2.5: Darstellung der verschiedenen Modi an Repräsentationen des menschlichen Skeletts, der Abstraktion der Hand als Repräsentation der primären Bewegung des Menschen sowie der Objekte, repräsentiert durch den Mittelpunkt des jeweiligen Begrenzungsrahmens nach der Objektdetektion. Visualisiert ist ebenfalls eine beispielhafte Repräsentation im zeitlichen Kontext für die verschiedenen Modi.

Weiterhin kann aber auch der gesamte Zustand einer Szene räumlich oder zeitlich als Graph repräsentiert werden [AAD+11], [KS13a], [KGS13], [KS13b], [YXL18], [DWA20], [LSS+23a] und [LSS+23b]. Durch die Repräsentation der Szene in Form eines Graphen können alle verfügbaren Informationen der

Szene, d.h. Objekte, Akteure (Mensch und Roboter) und die zugrundeliegenden Beziehungen untereinander berücksichtigt werden. Daraus ergibt sich neben der Szenenrepräsentation mit der Graphenkonstruktion ein weiteres wichtiges Thema, welches in Kapitel 4.1 näher beschrieben wird.

2.4 Vorhersage

Für die Vorhersage des menschlichen Verhaltens, hier im Speziellen die Intention und Bewegung des Menschen, sind die Bausteine *Szenenwahrnehmung* und *Szenenrepräsentation* von besonderer Bedeutung. Gemeinsam bilden sie die Grundlage, beispielsweise für die Vorhersage, welche Aktion ein Mensch aktuell tätigt oder wann tätigen wird, oder für die Vorhersage dessen Bewegung in Form einer Trajektorie. Im Folgenden werden dazu Vorhersagemethoden aus dem aktuellen Stand der Technik zusammenfassend beschrieben, wobei die Vorhersageziele *Aktion* und *Bewegung* als wichtige vorhersagbare Komponenten [LFS17] von besonderem Interesse sind. Ebenfalls wird für einen allgemeinen Überblick zum Themenbereich der Bewegungsvorhersage auf Basis von Trajektorien auf den Übersichtsartikel [RPH+20] und für die Aktionserkennung auf [HHP17] verwiesen.

Die Vorhersageziele werden in dieser Arbeit in Klassifikations- und Regressionsaufgaben des überwachten Lernens unterteilt. Nachfolgend sind verwandte Arbeiten nach ihrem Vorhersageziel und der Anzahl dieser eingeordnet (*Aktion*, *Bewegung* oder *Kombination*).

Aktion In der vorliegenden Arbeit wird bei der Aktionsklassifikation zwischen Erkennung und Vorhersage unterschieden. Die Fähigkeit der Erkennung und Vorhersage der Aktion, die von einem Menschen getätigt wird oder werden wird, ist die Grundlage für eine Kollaboration zwischen Mensch und Roboter. Eine naive Methode für die Klassifikation von Aktionen ist die Bayes'sche Inferenz. Dabei werden auf Basis von beobachteten Bewegungstrajektorien, welche einer bestimmten Aktion zugeordnet sind, Beobachtungen dieser Bewegungsklassen zugeordnet. Bereits in [SHSW05] wurde die Intention des Menschen mit Hilfe von Bayes'schen Netzwerken erkannt, indem Kausalitätsbeziehungen zwischen der Umwelt, Intention und Aktion in Graphenstruktur abgebildet wurden. Tatsächlich handelt es sich sogar um ein dynamisches Netzwerk, welches über

den momentanen Zustand hinaus den weiteren Verlauf zur Aktionserkennung berücksichtigt. Auch in den Arbeiten [PS15], [ZR17] wird die Bayes'sche Inferenz verwendet, um Aktionen bzw. Bewegungsklassen auf der Basis der beobachteten Bewegung der agierenden Person in der Umgebung zu klassifizieren. Die Bayes'sche Inferenz wird dabei nicht nur als Methode zur Klassifikation verwendet, sondern auch als Informationsquelle für weiterführende Ziele wie der Vorhersage von Bewegungen oder Positionen auf Basis der abgeleiteten Beobachtungen.

Semantische Ereignisketten (engl. *Semantic Event Chains* (SECs)) [AAD+11] zum Erkennen von vordefinierten Aktionen werden in [AAD+11], [ATW15] vorgestellt. Diese werden auf der Basis von Videodaten (RGB-Bild und Tiefeninformation, zur eindeutigen Kennzeichnung der Szene) mit zuvor gekennzeichneten Aktionssegmenten sowie den zeitlichen und räumlichen Zusammenhängen der Szene durch SECs modelliert und angelernt. Ein wichtiger Bestandteil dieser Methode ist die Bildung von semantischen Szenengraphen, die explizit semantische und räumliche Relationen zwischen Menschen und Objekten abbilden, aus denen anschließend zeitliche Zusammenhänge abgeleitet werden.

In [KGS13] wird mit probabilistischen Modellen die getätigte Aktion klassifiziert. Bei diesen Modellen handelt es sich um Markovsche Zufallsfelder (engl. *Markov Random Fields (MRF)*), die verwendet werden, um aus zeitlichen und räumlichen Abhängigkeiten die vergangene und zukünftige Aktion zu lernen. Besonders bedeutsam ist die Verwendung von extrahierten Mensch- und Objektdaten beispielsweise 3D-Positionen aus RGBD-Bilddaten für die Beschreibung der Szene.

Neben der Verwendung von RGB- und Tiefenbildern zur Extraktion von Merkmalen für die Klassifikation von Aktionen, werden auch aus RGB-Bildern abgeleitete Skelett-Daten eingesetzt (vgl. im Unterkapitel 2.2.1). Dabei gibt es Arbeiten, welche auf Basis dieser Merkmale mit faltenden neuronalen Netzen (engl. *Convolutional Neural Networks* (CNNs)), die ausgeführten Aktionen überwacht lernen. Dazu gehören neben den CNN-basierten Methoden [CZ17], die mit RGB-Daten und optischem Fluss (engl. *Optical Flow*) operieren, auch Methoden, die auf rekurrenten neuronalen Netzen (engl. *Recurrent Neural Networks* (RNNs)) [WWLK17], [LSXW16], [SLNW16] basieren.

Neben Methoden zur Aktionsklassifikation auf Basis von rekurrenten neuronalen Netzen, gibt es weitere Methoden auf Basis zeitlicher, räumlicher sowie

skelettbasierter Merkmale. In der Arbeit von [YXL18] wurde unter Anwendung von GNNs die skelettbasierte Aktionsklassifikation revolutioniert. Im Kontext dieser Arbeit wurden quantitative Vergleiche zwischen den Methoden, die auf RNNs [WWLK17], [LSXW16], [SLNW16], CNNs [KBA+17], aber auch auf zeitliche Faltungsnetzwerke (engl. *Temporal Convolutional Networks* (TCNs)) [vdODZ+16], [SKR17] basieren, durchgeführt, wobei der graphenbasierte Ansatz von [YXL18] die höchste Klassifikationsgüte erzielte. Auch in [LFV+17] werden TCNs zur Aktionsklassifikation verwendet, um zeitliche und räumliche Merkmale skelettbasiert zu lernen.

Neben den Methoden [YXL18], [LYFG21], [XB22a], die GNNs auf skelettbasierten Daten anwenden, gibt es auch Methoden wie in [WG18], [DWA20], [GYDD20], [MLVT21], [LZW+22], [WZL+23], die im HOI-Kontext GNNs zur Aktionsklassifikation verwenden. Dabei werden als zusätzliche Merkmale die gelernten bzw. detektierten Objekte, welche durch Methoden der Objekterkennung und -segmentierung extrahiert wurden, verwendet. Des Weiteren gibt es Arbeiten wie [GCDZ19], [XB22b], [PS23], in denen Encoder verwendet werden, welche auf Attention-Mechanismen von [VSP+17] basieren. Zusätzlich zur Verwendung von skelettbasierten Merkmalen können diese auch mit Objektinformationen aus der Szene kombiniert werden, wie in den Arbeiten [AER+21], [TXM+21], [BSCS21] und [XB22b] untersucht.

Aus Sicht der verwendeten Merkmale für die Methoden gibt es somit die folgenden Möglichkeiten:

- RGB-Bild (auch mit optischem Fluss)
- RGB- und Tiefenbild
- Skelett-Information
- Skelett-Information und detektierte Objekte der Szene
- Abstrahierte Repräsentation der Menschinformation und detektierte Objekte der Szene

Bewegung Die Bewegungsvorhersage des Menschen oder dessen Pose ist neben der Aktionsklassifikation eine wichtige Fähigkeit, um ein Szenenverständnis zu schaffen und eine intelligente Zusammenarbeit zwischen Mensch

und Roboter zu ermöglichen. Für die Vorhersage der Bewegung ist die Information über eine sich zeitlich ändernde Positionsinformation (Trajektorie), beispielsweise die des Menschen, notwendig. In der Literatur werden zur Vorhesage der zukünftigen Bewegung zwischen den skelettbasierten Methoden und den Multi-Agenten-Verfahren unterschieden, wobei die Bewegung des Menschen dabei auch auf eine einzelne Position pro Zeitschritt abstrahiert (oft der Mittelpunkt des Begrenzungsrahmens des detektierten Menschen) werden kann. Im HOI-Kontext wird der Mensch und dessen Bewegung analog durch eine repräsentative Darstellung dessen beschrieben, wie beispielsweise der Hand oder der Hände des Menschen [AAD+11], [ZR17], [LM19], [DWA20], [LTMW22], die in der Szene dominant aktiv sind. Neben der Trajektorie des Menschen oder des Hauptagenten, kann auch die Interaktion mit der Umgebung, beispielsweise mit anderen Objekten oder Agenten, berücksichtigt werden [KSM+19], [HBL+19], [DOL20], [MQEC20], [YMR+20], [GSZ+20], [LYH+20], [CHN+20], [HCZG20], [LMZT20], [LYTC20], [SICP20], [CLMT21], [LES+21].

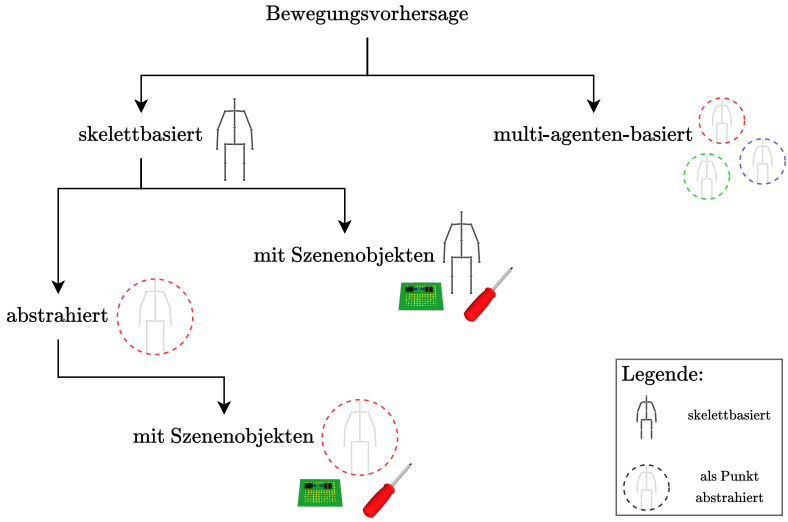


Abbildung 2.6: Übersicht der verschiedenen Verfahrensklassen für die Bewegungsvorhersage.

Die **skelettbasierte** Posenvorhersage ist sehr weit verbreitet, da sowohl mit einer 2D- als auch 3D-Repräsentation (vgl. Kapitel 2.2.1) effizient gearbeitet werden kann, sofern der Mensch sichtbar ist. Außerdem funktioniert die Detektion des Menschen umgebungsagnostisch. Auch die strukturierte Gegebenheit der Gelenkpunkte des Skeletts helfen bei der Vorhersage. Auf Basis der Skelett-Information wird mit tiefen neuronalen Netzen die Bewegungstrajektorie vorhergesagt. Dabei gibt es Methoden, die dies auf Basis von Encoder-Decoder-Architekturen [CvG+14], [CvBB14] umsetzen, wobei die Kernkomponente der Architektur variabel ist. In [FLFM15] vorgestellten Encoder-Recurrent-Decoder (ERD)-Architektur wird ein LSTM-Netzwerk [HS97] verwendet, um die Dynamik von Bewegungen zu lernen. Ein *Gated Recurrent Unit* (GRU) [CvG+14], [CvBB14] wird in der ERD-Architektur von [MBR17] verwendet. Genauer gesagt handelt es sich bei dieser Arbeit um eine Sequence-To-Sequence [SVL14] basierte ERD-Architektur, um mit variablen Sequenzlängen der Eingangsdaten umzugehen.

Neben RNN- gibt es auch CNN- bzw. mehrlagige Perzeptron (engl. *Multilayer Perceptron* (MLP))-basierte Architekturen [MHRL17], [LZLL18], [CZLT19], [CGM+20], [GDS+23], die auf Basis von Skelett-Daten die Bewegung des Menschen vorhersagen. Dabei werden die Skelett-Daten als Sequenzen verarbeitet [LZLL18], [CGM+20], [GDS+23]. Beispielsweise wird in [CZLT19] ein offline trainiertes MLP zum Lernen eines menschlichen Bewegungsmodells mit einer rekursiven Least-Squares-Schätzung des Prädiktionsfehlers kombiniert, um online die Modellparameter der letzten MLP-Schicht anzupassen und damit die Modellunsicherheit zu reduzieren. Auch der Einsatz von General Adversarial Networks (GANs) [GPM+14] und GAN-basierten Architekturen, welche ebenfalls auf Sequenz-Daten vom menschlichen Skelett operieren [GZW+18], [RLL+23], findet Verwendung.

Wie auch bei der Aktionsklassifikation mit skelettbasierten Daten, ist die Verwendung von Methoden, welche die natürliche Struktur eines Skeletts verarbeiten, naheliegend. GNNs werden auch für die skelettbasierte Bewegungsvorhersage verwendet, um aus der räumlichen Beziehung der Gelenke des Körpers und aus der zeitlichen Beziehung des Skeletts in aufeinander folgenden Zeitschritten zu lernen [MLSL19], [CSY20], [LCPW21], [APRB22], [SdA+22], [FYD+23].

So wird in [MLSL19] der verwendeten GNN-Komponente eine diskrete Kosinus-Transformation (engl. *Discrete Cosine Transform* (DCT)) vor- und nachgeschalten. Durch eine DCT werden zeitliche Zusammenhänge natürlich erfasst und

als Merkmale in einem GNN verwendet, um darüber hinaus die räumlichen Zusammenhänge zu lernen. Abschließend werden mit einer inversen DCT die skelettbasierten Bewegungen extrahiert.

In [LCPW21] werden die mit einem GNN-basierten Encoder gelernten Repräsentationen mit einem GRU-basierten Decoder kombiniert, um die zukünftigen Menschen-Posen vorherzusagen. Auch in der Verbindung mit einem TCN werden GNNs in Encoder-Decoder-Architekturen zum Lernen verwendet [CSY20], [SdA+22].

Ein GRU-basierter Encoder kombiniert mit dem Attention-Mechanismus nach [VSP+17] wird in [DFD22b] verwendet, um eine latente Repräsentation auf Grundlage von Skelett-Daten zu lernen. Daraus wird dann für jedes Gelenk eine Wahrscheinlichkeitsdichte abgeleitet, aus der dann die jeweilige Gelenkposition extrahiert und zusammen mit einem vorhergesagten Offset für jedes Gelenk korrigiert geschätzt wird. Dabei wird die Vorhersage der Menschen-Pose durch die Reihenfolge der Skelett-Bestandteile autoregressiv bedingt.

In [AER+21], [BSCS21] wird **skelettbasierte** Poseninformation und **Szenenobjekte** kombiniert und mittels zusammengesetzter Attention-basierter GNNs oder GANs mit RNNs zur Bewegungsvorhersage verwendet. Dabei werden tiefe Merkmale aus der Skelettrepräsentation und den Szenenobjekten gelernt, die dann für die Vorhersage der Bewegung verwendet werden. In [CPAM20] wird die Information der menschlichen Bewegung und der Szenenumgebung zweigeteilt verarbeitet, um die Bewegung des Menschen und der Objekte vorherzusagen. In einem Zweig wird die zeitliche Dynamik der menschlichen Bewegung mit einem RNN-basierten Ansatz nach [MBR17] gelernt und vorhergesagt. Im anderen Zweig wird der Umgebungskontext, der als Graph strukturiert ist, mithilfe eines Attention-Mechanismus gelernt. Anschließend werden die Ergebnisse beider Zweige zusammengeführt, um die vorhergesagte Bewegung des Menschen und der Objekte zu erhalten.

Neben der skelettbasierten Beschreibung der menschlichen Pose und damit der Trajektorie, werden auch **Abstraktionen** dieser verwendet, um die Bewegung zu beschreiben und vorherzusagen [AAD+11], [ZR17], [LM19], [DWA20], [LTMW22], [LSS+23a] und [LSS+23b]. Dies kann eine Punktrepräsentation sein (Mittelpunkt des Begrenzungsrahmens der erkannten Person) oder es wird das Körperteil des Menschen verwendet, das hauptsächlich aktiv ist und in der

Szene mit dem Roboter und den Objekten interagiert (beispielsweise die Hand des Menschen).

Die Vorhersage kann filter-basiert auf Basis von Annahmen über die Bewegungsgeschwindigkeit geschätzt werden [LS17], [SG64], aber auch durch prototypenbasierte Methoden realisiert werden, welche meist durch Bayes'sche Inferenz mit nachgestellter geschwindigkeitsbasierter Bewegungsschätzung umgesetzt werden [ZR17], [LS17], verwendet in [ULT+18].

Die Verwendung von Transformern, die auf Basis von Handtrajektorien und Objektpositionen aus der Szene Bewegungen lernen, wird in [LTMW22] vorgestellt. Dabei wird eine Transformer-basierte Encoder-Decoder-Architektur verwendet, die mit dem Encoder tiefe Merkmale aus der Hand-, Objekt- und Umgebungskontext-Information lernt. Der Decoder wird in Kombination mit einem *Conditional Variational Auto-Encoder* [KW13], [SLY15] für die Vorhersage der zukünftigen Handpositionen und der Objekt-Kontaktpunkte in der Szene verwendet.

Die **Multi-Agenten-Verfahren** beschäftigen sich mit der Bewegungsvorhersage von Fußgängern oder aber von Ego-Fahrzeug und Verkehrsteilnehmenden in einer dynamischen Szene, in der eine Wechselwirkung zwischen diesen vorliegt. Gerade in einer solchen zugrundeliegenden komplexen Struktur bieten sich Methoden an, die Zusammenhänge abbilden und daraus lernen können, um Lernziele wie das der Bewegungsprädiktion zu erreichen. Eine Vielzahl von Arbeiten setzt dabei GNNs für die Bewegungsvorhersage von Fußgängern [KSM+19], [LES+21], [DOL20], [MQEC20], [YMR+20], [HBL+19], von Fahrzeugen [CLMT21], [GSZ+20], [LYH+20] oder für beide [CHN+20], [HCZG20], [LMZT20], [LYTC20], [SICP20] ein. Dabei werden Encoder-Decoder-Architekturen in Kombination aus GNNs mit und ohne Attention-Mechanismus sowie aus RNN-basierten Netzen im Decoder verwendet. Durch die zeitliche Natur der Bewegungstrajektorien und den Wechselwirkungen werden ausschließlich Graphen mit zeitlicher und räumlicher Information konstruiert.

Kombination Eine weitere Kategorie stellen Methoden dar, die nicht nur eine Aufgabe sondern mehrere Aufgaben *kombiniert* lösen, um eine intelligente Zusammenarbeit zu gewährleisten.

Die Arbeit [KGS13] wurde zur Klassifikation von zukünftigen Aktionen durch komplexere Modelle erweitert, die sequenzielle Strukturen lernen können. Dazu gehören antizipatorische, temporäre, bedingte Zufallsfelder (engl. *Anticipatory Temporal Conditional Random Fields (ATCRF)*) [KS13a], [KS16] bzw. bedingte Zufallsfelder (engl. *Conditional Random Fields (CRF)*) [KS13b]. Somit werden in beiden Arbeiten auf Basis von räumlich-zeitlichen Graphen Szenenabhängigkeiten modelliert und gelernt. Diese Arbeiten sind im HOI Kontext einzuordnen.

Die Aktionsklassifikation und Bewegungsvorhersage wird in der Arbeit von [LM19] mit einem zweistufigen Modell umgesetzt. Mit Hilfe von dynamischen Bewegungsprimitiven (engl. *Dynamic Movement Primitives* (DMP)) und probabilistischen DMPs (PDMP) werden unüberwacht aus den Bewegungstrajektorien der Menschenhand Primitive offline gelernt, die zur Erkennung der Aktion/Intention und online Vorhersage der Bewegung verwendet werden können. Dabei wird die erkannte Aktion als Information zur Bestimmung der PMDPs verwendet, wodurch daraufhin die zukünftige Bewegung berechnet werden kann.

Eine weitere Arbeit [TXM+21] kombiniert die Klassifikation der momentanen Aktion mit der Vorhersage der zukünftigen Bewegung. Um die Beziehungen zwischen den Gelenken des menschlichen Skeletts und der Umgebung in Form von Objekten zu lernen, werden ein GNN und ein GRU, welches zur Modellierung der zukünftigen Pose des Menschen und der momentanen Aktion verwendet wird, kombiniert. Neben der Skelett-Information wird hier demnach zusätzlich das Wissen über die Szenenobjekte verwendet.

In [ZVVM21] wird eine Encoder-Decoder-Architektur für die Aktionsklassifikation und die Bewegungsvorhersage vorgestellt. In dieser Architektur wird ein GNN-LSTM-Encoder vorgestellt, der ausschließlich Sequenzen des menschlichen Skeletts nutzt, um sowohl räumliche Beziehungen mit Hilfe von GNNs für Elemente der Sequenz als auch die Dynamik der Bewegung durch ein LSTM zu erlernen. Diese erlernte Repräsentation wird dann mittels zweier Decoder für die jeweilige Vorhersage weiterverarbeitet. Der erste Decoder basiert auf einem CRF-Modell und wird zur Schätzung der Aktion verwendet. Der zweite Decoder nutzt zur Prädiktion der Bewegung ein LSTM-Modell.

Die kombinierte Vorhersage der zukünftigen Aktion und der 3D-Pose des Menschen wird in der Arbeit [DFD22a] vorgestellt. Hierbei wird durch drei

MLP-basierte Encoder aus einer Sequenz von RGB-Bildern, der Information über die aktuell ausgeführte Aktion sowie der 2D-Pose der sichtbaren Person eine reichhaltige Repräsentation gelernt. Diese beinhaltet die Historie der 2D-Pose, die Aktionen und die sichtbaren Objekte. Der Encoder für die Historie von 2D-Posen ist dabei durch mehrere residuale Schichten aufgebaut. Über einen geteilten MLP-basierten Decoder wird anschließend die zukünftige Aktionsklasse und die 3D-Pose extrahiert, inklusive der projizierten 2D-Pose.

Die gleichzeitige Vorhersage der momentanen und zukünftigen Aktion mit Hilfe des *Variational Graph Auto-Encoders* [KW16b] wird in [ASA21] vorgestellt. Mit einem GNN-basierten Encoder wird eine reichhaltige Repräsentation der Szene gelernt. Aus dieser wird gleichzeitig die Aktion erkannt und vorhergesagt sowie mit einer kombinierten Verlustfunktion das Netzwerk trainiert.

Zusammenfassung Motiviert durch das Anwendungsszenario, die Zusammenarbeit von Mensch und Roboter für Montageaufgaben in der Industrie zu erleichtern bzw. zu verbessern, ergeben sich für eine Methode, die dem Roboter die Intention des Menschen näher bringen soll, bestimmte Kriterien.

Zu den Wichtigsten gehört der Umgang mit einem beschränkten Sichtfeld für die Szenenwahrnehmung, wodurch unter Umständen lediglich die Hand des Menschen und die Szenenobjekte lokalisierbar sind, da durch eine einzelne Kamera die gesamte Szene inklusive des Menschen betrachtet wird.

Daraus ergibt sich auch, dass bereits eine spezifische Komponente zur Detektion der Werkstücke für die anwendungsbezogenen Montageaufgaben benötigt wird. Weiterhin gibt es auch ein Kriterium bezüglich der zur Verfügung stehenden Ressourcen. Dies zeigt sich durch die Verwendung von Eingebetteten-Geräten, welche oftmals mit hochdimensionalen Eingangsdaten wie Bildern schnell an ihre Verarbeitungsgrenzen gelangen.

Die Intention des Menschen schlüsselt sich nicht nur durch die Erkennung der Aktion, die ein Mensch tätigt, auf. Vielmehr ergibt sich die Intention aus einer Kombination mehrerer Aussagen, wie etwa dem Wissen über die aktuelle und zukünftige Aktion, die der Menschen tätigt oder tätigen wird, aber auch durch die Vorhersage dessen Bewegung oder dem Zeitpunkt eines Aktionswechsels, die Roboter verstehen können muss.

Aus diesen Kriterien ergibt sich eine Methode, welche die Szene durch explizite Information abbildet, ohne dabei wichtige Information zwischen Szenenobjekten, inklusive des Menschen, zu verlieren. Darüber hinaus sollte die Methode in der Lage sein, auch nur mit der Abstraktion des menschlichen Skeletts zu operieren und sowohl für die Detektion des Skeletts als auch für die Detektion der Szenenobjekte mit extrahierten Daten zu arbeiten. Zum einen, weil die vollständige Pose des Menschen nicht immer sichtbar sein wird, und zum anderen, um die Dimensionalität der Eingangsdaten gering zu halten. Außerdem ist eine weitere wichtige Eigenschaft der Umgang mit 3D-Information der Szene, da die Tiefeninformation für das Verständnis über Szene oder die Intention des Menschen unerlässlich ist. Schließlich sollte die Methode nicht nur auf eine Vorhersage wie der Aktionsklassifikation beschränkt sein, sondern durch eine kombinierte Architektur gleichzeitig auch andere Vorhersagen, wie etwa die Bewegung des Menschen, ermöglichen.

Die Arbeit von [DWA20] erfüllt Teile dieser Kriterien und demonstriert dies im Kontext der Aktionserkennung. Aus extrahierter Information (Objektklassifikation und -lokalisierung) der Szene, werden Szenengraphen konstruiert, welche die räumliche Information für einen bestimmten Zeitpunkt sowie die zeitliche Information über mehrere Zeitpunkte hinweg beinhalten. Eine Szene enthält dabei die Repräsentation des Menschen durch die jeweiligen Hände und der teilnehmenden Szenenobjekte. Die Information bzw. die Relation zwischen Mensch und Szenenobjekten wird durch semantische, räumliche Relationen nach [ZKTW18] abgebildet. Zeitliche Relationen werden zwischen den selben Szenenobjekten und der Hände über Zeitpunkte hinweg innerhalb eines Szenengraphen kodiert. Ein Szenengraph bildet somit Relationen zwischen allen verfügbaren Szenenobjekten inklusive der Hände des Menschen ab und ist somit vergleichbar mit einem vollständigen Graph (vgl. Kapitel 4.4 unter Vernachlässigung der Kanten- und Knotenmerkmale). Diese Szenengraphen werden von GNNs nach [BHB+18] (vgl. Kapitel 3.2.5) verarbeitet und zum Lernen eines Aktionskennzeichens verwendet, das der spezifischen Aktion der jeweiligen Hand zu einem bestimmten Zeitpunkt entspricht und jedem Szenengraphen zugeordnet ist.

Aufbauend auf der Szenenrepräsentation durch Szenengraphen und der Methode zur Vorhersage der Aktion der Grundlagenarbeit von [DWA20], werden in den folgenden Kapiteln (vgl. Kapitel 4 und Kapitel 5) nicht nur diese Aspekte erweitert, sondern neue Ansätze verfolgt, die ausführlich beschrieben werden.

Dabei wird unter anderem ein alternativer und weit verbreiteter Ansatz für GNNs nach [KW16a] im Rahmen einer neuartigen Architektur untersucht. Anders als bisher wird vorwiegend die Information der Knoten verwendet, wodurch die Komplexität in der Verarbeitung von Graphen durch GNNs abnimmt, der Trainingsprozess beschleunigt wird und die Anforderungen an die Ressourcen abnehmen, ohne dass dabei die Modellgüte abnimmt.

Die Graphenkonstruktion (vgl. Kapitel 4.1) wird ebenfalls untersucht und erweitert, um sich explizit auf den Menschen bzw. dessen Hand als die agierende Komponente in Montageaufgaben und damit zentrales Glied in der Graphenmodellierung zu konzentrieren. Auch bei der Gestaltung der Kanten eines Graphen wird die räumliche Distanz zwischen Objekten explizit als Attribut kodiert.

Gegeben durch das Anwendungsszenario, dass ein Roboter den Menschen proaktiv unterstützen soll, wird zusätzlich eine Architektur untersucht, die auch für eine Bewegungsvorhersage geeignet ist. Ferner soll die Methode nicht nur eine, sondern mehrere Lernaufgaben gleichzeitig lernen können. Neben der Aktionserkennung gehören zu diesen beispielsweise die Bewegungsvorhersage, die Vorhersage der zukünftigen Aktion zu einem festen oder dynamischen Zeitpunkt oder aber die Vorhersage des Zeitpunktes des Aktionswechsels. Diese Lernaufgaben sollen gleichzeitig während der Inferenz verfügbar sein und sich dabei auf Architekturebene gelernte Gewichte und die zugrundeliegende Information teilen, um eine MRK zu ermöglichen.

Insgesamt liegt der Fokus dieser Dissertation auf der Entwicklung einer Methode, welche sich in ein Robotersystem (vgl. Kapitel 2.1 und Abbildung 2.1) eingliedern lässt, um eine intelligente MRK zu ermöglichen. In Kapitel 4 bzw. in Kapitel 4.2, wird die in dieser Arbeit erarbeitete Methode vorgestellt und explizit auf mögliche Lernaufgaben, die darin umgesetzt werden, eingegangen.

3 Graph Neural Networks

GNNs sind eine Methode des maschinellen Lernens mit der aus Daten, die in Form von Graphen strukturiert sind, Muster und Repräsentationen gelernt werden können. Graphen sind mathematische Strukturen, die eine Menge von Elementen (Knoten) und deren Beziehungen (Kanten) zueinander darstellen. In diesem Kapitel werden die Grundlagen von GNNs erarbeitet, verschiedene GNN-Varianten und weiterführende Konzepte dazu diskutiert sowie eine prototypische GNN-Architektur vorgestellt. Hierzu werden zunächst die notwendige Graphentheorie und wichtige Begriffe eingeführt.

3.1 Graphentheorie und Definition

Der Einsatz von Graphen und deren Theorie sind ein wichtiges Forschungsgebiet für Wissenschaften, die sich mit der Darstellung und Beschreibung komplexer Systeme und der Analyse von Beziehungen zwischen Objekten und ihren Interaktionen beschäftigen. Im Beispiel von Zachary's Karate-Club Netzwerk [Zac77] (vgl. Abbildung 3.1) lässt sich mit Hilfe von Graphen ein semantisches Netzwerk zur Repräsentation von Wissen darstellen, auf welches dann verschiedene graphenbasierte Algorithmen angewendet werden können. Das allgemeine Ziel dabei ist, die Beziehungen zwischen einzelnen Teilnehmenden zu analysieren, obgleich auf Graphen-, Knoten- oder Kantenebene. In diesem Abschnitt werden Graphen und die Graphentheorie formal weitgehend nach [Ham20] beschrieben und definiert.

Graph Ein Graph $\mathcal{G}$ wird formell durch $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ beschrieben, wobei $\mathcal{V}$ eine endliche, nicht leere Menge mit $n = |\mathcal{V}|$ unterschiedlichen Knoten $\mathcal{V} = \{v_1, \ldots, v_n\}$ (engl. *vertices*) und $\mathcal{E}$ eine endliche Menge an Kanten $\mathcal{E}$ (engl. *edges*), mit $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ ist. Wenn $(u, v) \in \mathcal{E}$, wird auch $u \rightarrow v$ geschrieben. Ein Graph heißt ungerichtet, wenn $(u, v) \in \mathcal{E} \Rightarrow (v, u) \in \mathcal{E}$.

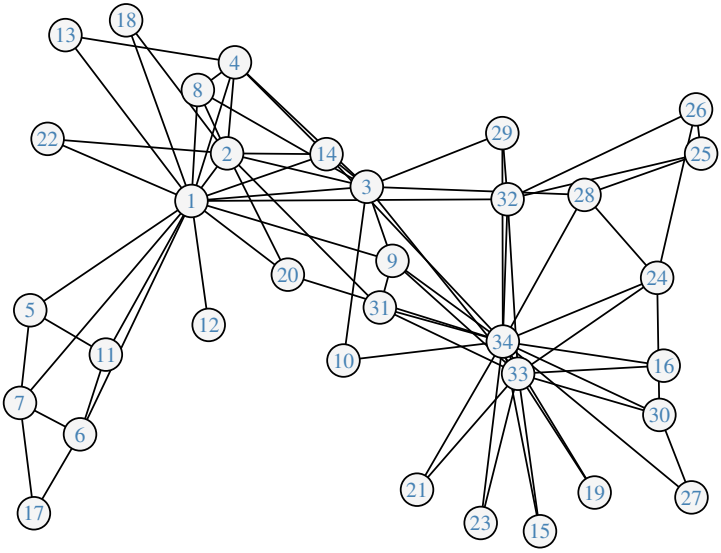


Abbildung 3.1: Darstellung von Zachary's Karate-Club-Netzwerks nach und in Anlehnung an [Zac77]. Die Position der Knoten wurde zufällig gewählt.

Graphen können durch sogenannte Adjazenzmatrizen beschrieben werden. Eine Adjazenzmatrix $\mathbf{A}$ ist durch $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ definiert, die für jeden Knoten des Graphen eine Indizierung darstellt, um vorhandene Kanten darzustellen. Diese Indizierung wird für ungerichtete Graphen definiert als

$$\mathbf{A}_{u,v} = \begin{cases} 1, & \text{falls } (u,v) \in \mathcal{E} \\ 0, & \text{falls } (u,v) \notin \mathcal{E}, \end{cases} \tag{3.1}$$

wobei hier u und v durch eine festgelegte Nummerierung der Knoten identifiziert werden.

Die Adjazenzmatrix $\mathbf{A}$ eines ungerichteten und ungewichteten Graphen ist symmetrisch, da (u,v) und $(v,u) \in \mathcal{E}$ sind. In der Abbildung 3.2a ist ein solcher Graph dargestellt, mit der Adjazenzmatrix

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Gerichtete Graphen Ein gerichteter Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (auch: *Digraph*) besteht aus einer Knotenmenge $\mathcal{V}$ und einer Menge von geordneten Knotenpaaren $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, welche auch als gerichtete Kanten bezeichnet werden, da diese im Gegensatz zu ungerichteten Graphen eine Orientierung aufweisen. Für eine gerichtete Kante $e = (u, v)$ mit $u, v \in \mathcal{V}$ entspricht der Knoten u dem Startknoten und der Knoten v dem Endknoten der Kante e. Ein beispielhafter gerichteter Graph ist in Abbildung 3.2b dargestellt, mit der zugehörigen Adjazenzmatrix

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Dabei stehen die Spaltenindizes für Startknoten und die Zeilenindizes für die Endknoten der gerichteten Kanten.

Gewichtete Graphen Ein Graph, in dem jeder Kante $(u, v) \in \mathcal{E}$ ein Kantengewicht $\varepsilon_{u,v} \in \mathbb{R}$ zugeordnet ist, wird als gewichteter Graph bezeichnet. Diese Zuordnung wird durch eine gewichtete Adjazenzmatrix $\mathbf{A}'$ beschrieben. Für jeden Eintrag $\mathbf{A}'_{u,v}$ dieser Matrix für gerichtete und ungerichtete Graphen gilt

$$\mathbf{A}'_{u,v} = \begin{cases} \varepsilon_{u,v}, & \text{falls } (u, v) \in \mathcal{E} \\ 0, & \text{falls } (u, v) \notin \mathcal{E}. \end{cases} \tag{3.2}$$

Im Folgenden wird nicht zwischen $\mathbf{A}'$ und $\mathbf{A}$ unterschieden, wenn der Kontext bekannt ist.

In Abbildung 3.2c ist exemplarisch ein gewichteter Graph dargestellt, mit der Adjazenzmatrix

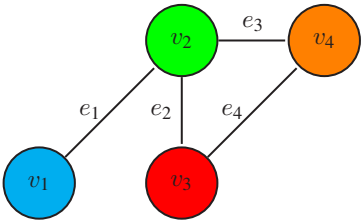
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 3 \\ 0 & 2 & 0 & 5 \\ 0 & 3 & 5 & 0 \end{bmatrix}.$$

Relationelle Graphen Bei sogenannten relationellen Graphen wird die Beschreibung der Kantenmenge $\mathcal{E}$, um Kantentypen τ zu $(u, \tau, v) \in \mathcal{E}$ erweitert. Für jeden Kantentyp τ wird eine Adjazenzmatrix $\mathbf{A}_\tau$ definiert. Somit werden relationelle Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, mit $\mathcal{R}$ als der Menge an Relationen, durch eine Adjazenzmatrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{R}| \times |\mathcal{V}|}$ beschrieben. Abbildung 3.2d zeigt exemplarisch einen relationellen Graphen mit zwei Relationen, dargestellt durch die unterschiedlich gefärbten Kanten (rot und blau). Nachfolgend werden mit heterogenen und multiplexen Graphen zwei Ausprägungen relationeller Graphen beschrieben.

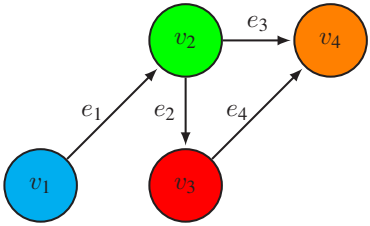
Heterogene Graphen Ein heterogener Graph definiert neben den Kantentypen $\tau \in \mathcal{R}$, auch Knotentypen und partitioniert somit die Knotenmenge für einen Graphen in disjunkte Mengen $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots \cup \mathcal{V}_k$ also, $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset, \forall i \neq j$, für $i, j \in \{1, 2, \ldots, k\}$. Es kann durch diese Definition verschiedener Knotentypen erzwungen werden, dass bestimmte Kanten bzw. Kantentypen nur als Verbindung bestimmter Knotentypen verwendet werden dürfen: $(u, \tau_i, v) \in \mathcal{E} \rightarrow u \in \mathcal{V}_j, v \in \mathcal{V}_k$. Ferner kann dies zusätzlich eingeschränkt werden, indem die Verbindung der Knoten nur durch Kanten zwischen verschiedenen Knotentypen erzwungen wird $(u, \tau_i, v) \in \mathcal{E} \rightarrow u \in \mathcal{V}_j, v \in \mathcal{V}_k \wedge j \neq k$, wie im Speziellen bei den multipartiten Graphen.

Multiplex-Graphen Bei den Multiplex-Graphen wird angenommen, dass ein Graph $\mathcal{G}$ in mehrere Schichten k aufgeteilt werden kann. Zu jeder Schicht k gehört eine Untermenge an Knoten $\mathcal{V}_k$. Für diese Menge an Knoten $\mathcal{V}_k = \{v_1, \ldots, v_n\}$ bestehen schichtinterne Verbindungen (engl. *intra-layer connections*) über die Kanten $\mathcal{E}^k$. Des Weiteren gibt es auch schichtübergreifende Verbindungen (engl. *inter-layer connections*), die den gleichen Knoten über Schichten hinweg mit Kanten verbinden. Im Rahmen dieser Arbeit sind insbesonders homogene Multiplex-Graphen wichtig und bilden die Untersuchungs-

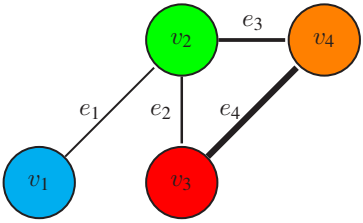
grundlage in Form von zeitlich und räumlich strukturierten Daten (vgl. Kapitel 4.1).



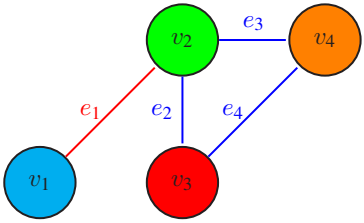
(a) Ungerichteter, ungewichteter Graph



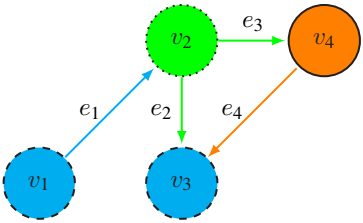
(b) Gerichteter Graph



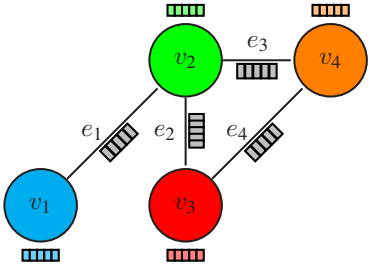
(c) Gewichteter Graph



(d) Relationeller Graph



(e) Heterogener Graph



(f) Graph mit Knoten- und Kanten-Merkmalen

Abbildung 3.2: Überblick über die verschiedenen Arten von Graphen, die in dieser Arbeit von besonderer Bedeutung sind.

Nachbarschaft Mit der Nachbarschaft (engl. *neighborhood*) eines Knoten u eines ungerichteten Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ wird die Menge an Knoten $\mathcal{V}_u$

beschrieben, die mit dem Knoten u durch eine Kante verbunden sind, also $\mathcal{N}(u) = \{v \in \mathcal{V} - u : (u, v) \in \mathcal{E}\}$. Für gerichtete Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ kann die Nachbarschaft analog als $\mathcal{N}^+(u) = \{v \in \mathcal{V} - u : (v, u) \in \mathcal{E}\}$ Eingangs-Knoten-Nachbarschaft (engl. *in-neighborhood*) und $\mathcal{N}^-(u) = \{v \in \mathcal{V} - u : (u, v) \in \mathcal{E}\}$ Ausgangs-Knoten-Nachbarschaft (engl. *out-neighborhood*) für Eingangs- und Ausgangs-Knoten gerichteter Kanten definiert werden. Insgesamt ergibt sich die Nachbarschaft $\mathcal{N}$ eines Knotens u zu $\mathcal{N}(u) = \mathcal{N}^+(u) \cup \mathcal{N}^-(u)$ [BG09]. Hier wird für gerichtete und ungerichtete Graphen, die Formulierung explizit ohne Schleife am Knoten u selbst (engl. *self-loops*) zur Nachbarschaftsbestimmung dargestellt.

Grad Der Grad eines Knotens beschreibt die Anzahl der Kanten, durch die ein Knoten u eines Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ mit $v \in \mathcal{V}$ verbunden ist. Für ungerichtete Graphen entspricht der Grad eines Knotens der Kardinalität der Nachbarschaft $\mathcal{N}(u)$. Ausgedrückt durch die Adjazenz gilt für den Grad eines Knotens

$$\text{degree}(u) = \sum_{v \in \mathcal{V}} \mathbf{A}_{u,v} \tag{3.3}$$

Zur Vereinfachung wird der Operator $\text{degree}(\cdot)$ für den Knotengrad verwendet.

Die sogenannte **Gradmatrix** (engl. *degree matrix*) $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ beschreibt eine mögliche Darstellung des Grades der Knoten $u \in \mathcal{V}$ eines Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Für ungerichtete Graphen $\mathcal{G}$, wie in Abbildung 3.2a dargestellt, kann der Grad eines Knotens $u \in \mathcal{V}$ durch die Hauptdiagonale der Gradmatrix beschrieben werden und lautet

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}.$$

Somit ergibt sich für einen Knoten u ohne Adjazenz, der Knotengrad $\text{degree}(u) = 0$.

Für gerichtete Graphen können diese Sachverhalte in [BG09], [Die17] detailliert nachvollzogen werden.

Knoten- und Kanten-Merkmale Neben den erläuterten Möglichkeiten Graphen zu beschreiben, gibt es die Option über Merkmale (engl. *features*) zusätzliche Informationen durch Assoziation mit den Knoten (engl. *node features*) sowie den Kanten (engl. *edge features*) abzubilden. Somit können Merkmale in vektor-kodierter Form für Knoten und Kanten definiert werden. Für einen Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ werden die Knoten-Merkmalsmatrix als $\mathbf{X}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d}$ und die Kanten-Merkmalsmatrix als $\mathbf{X}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d}$ sowie $\boldsymbol{x}_u, \forall u \in \mathcal{V}$ als Knotenmerkmalsvektoren und $\boldsymbol{x}_{u,v}, \forall u, v \in \mathcal{V}$ als Kantenmerkmalsvektoren definiert. Dabei ist d die jeweilige Dimension eines Merkmalsvektors der Knoten und Kanten.

Durch die Definition von Merkmalen im Kontext von Graphen kann Information in den Graphen eingebettet werden, welche als Ziel für Aufgaben des maschinellen Lernens verwendet werden können. Gemeint sind Einbettungen (engl. *embeddings*), die durch lernende Methoden zum Lösen von beispielsweise Klassifikationsproblemen genutzt werden können. Die Grundlagen zum Lernen auf Basis dieser Embeddings im Kontext von Graphen werden im nachfolgenden Kapitel 3.2 erläutert.

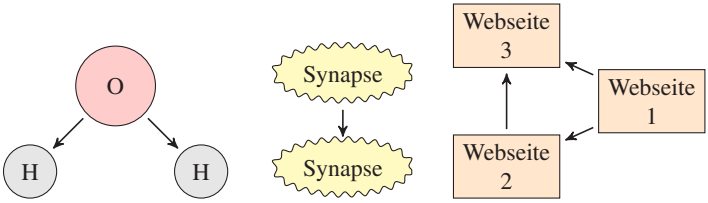
3.2 Lernen von Repräsentationen

Beim Lernen von Repräsentationen gibt es neben den Methoden auf Basis von GNNs auch filterbasierte Methoden (engl. *graph kernel methods*), die auf der spektralen Graphentheorie (engl. *spectral graph theory*) [HVG11], [DBV16] basieren. Details hierzu finden sich ebenfalls in [Ham20].

Im Folgenden wird auf Basis von [Ham20] kurz auf die Bedeutung von Encodern und Decodern im Rahmen des Lernens von Graphrepräsentationen eingegangen, um ein Verständnis für GNNs und die dahinterstehenden Methodologien sowie für wichtige Vertreter dieser zu schaffen. Das zentrale Fundament in der Graphentheorie wurde durch die Arbeit von Weisfeiler und Lehman [WL68] gelegt, in der die grundlegenden Prinzipien zur Bestimmung der Gleichheit oder Ähnlichkeit von Graphen, auch bekannt als Graphen-Isomorphietest, durch den Weisfeiler-Lehman-Algorithmus beschrieben werden. Schlüsselkonzepte umfassen dabei das *iterative* Sammeln von Informationen aus der Nachbarschaft von Knoten, die Berücksichtigung der zugrundeliegenden Graphenstruktur (vgl. Abbildung 3.3) und somit der *Relationen* zwischen den Daten. Ebenso

beinhaltet es die Fähigkeit, mithilfe des Prinzips des Graphen-Isomorphietests auf unbekannte oder neue Graphen zu *generalisieren*.

Diese grundlegenden Prinzipien können als Inspiration für die Entwicklung von GNNs gesehen werden.



(a) Wassermolekül (b) Synapsen (c) Webseiten

Abbildung 3.3: Beispielhafte Darstellung graphartiger Topologien: Moleküle, Verbindungen von Synapsen oder den Verknüpfungen von Webseiten.

3.2.1 Encoder, Decoder und Embeddings

Die folgenden Definitionen beziehen sich auf die Ebene der Knoten (vgl. Abbildung 3.6) und werden später in den Kapiteln 3.2.3 und 3.2.4 auf die Graphenebene erweitert.

Das Ziel im Lernen von Repräsentationen ist es, aus gegebener Information (in diesem Fall als Graphen strukturierte Daten), repräsentative Beschreibungen, Embeddings zu lernen, die zur Rekonstruktion der Information genutzt werden können. Embeddings finden in Bereichen des maschinellen Lernens, wie der natürlichen Sprachverarbeitung und anderen Gebieten, Anwendung.

Es gibt verschiedene Ansätze zum Lernen von Embeddings. Einige Ansätze erzeugen Embeddings mit Hilfe von Nachschlagetabellen (engl. *shallow embeddings*). Andere Ansätze, wie die in GNNs verwendeten, erzeugen Embeddings, die eine generalisierbare Repräsentation der Daten ermöglichen.

Im Folgenden werden zunächst die Begriffe Encoder, Decoder und Embeddings im Kontext von Shallow-Embedding-Ansätzen erläutert. In den Kapiteln 3.2.2 und 3.2.3 werden diese Konzepte im Rahmen von GNNs erweitert und vertieft.

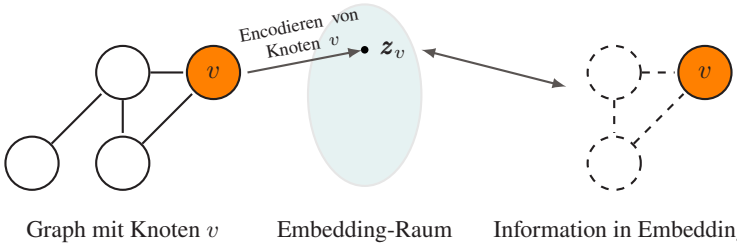


Abbildung 3.4: Schematische Darstellung eines Graphen, dessen Knoten v durch einen Encoder in den Embedding-Raum überführt wird. Es resultiert das Knoten-Embedding $\boldsymbol{z}_v$ mit der aggregierten Information aus der direkten Nachbarschaft von v.

Encoder Ein Encoder ist eine Funktion zur Transformation oder Abbildung von Information auf eine Repräsentation im latenten Raum oder Embedding-Raum (engl. *latent space* oder *embedding space*). Für einen Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ werden seine Knoten $v \in \mathcal{V}$ jeweils auf ein Embedding bzw. einen Embedding-Vektor $\boldsymbol{z}_v \in \mathbb{R}^d$ abgebildet. Die Abbildungsfunktion des Encoders wird formal als

$$\text{ENC} : \mathcal{V} \rightarrow \mathbb{R}^d \tag{3.4}$$

beschrieben.

Für die einzelnen Embedding-Vektoren $\boldsymbol{z}_v$ ergibt sich somit insgesamt

$$\text{ENC}(v) = \mathbf{Z}[v], \tag{3.5}$$

wobei $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ der sogenannten Embedding-Matrix $\mathbf{Z}$ entspricht, welche die Embedding-Vektoren für alle Knoten $v \in \mathcal{V}$ beinhaltet. Dabei entspricht $\mathbf{Z}[v]$ der Zeile der Embedding-Matrix $\mathbf{Z}$, die den Embedding-Vektor des Knotens v beinhaltet.

Die Embedding-Matrix $\mathbf{Z}$ muss im Fall von Shallow-Embedding-Methoden vollständig definiert sein, um zur Rekonstruktion von Knoten $v \in \mathcal{V}$ verwendet werden zu können. Im Gegensatz dazu muss dies bei GNNs *nicht* gewährleistet sein (vgl. Kapitel 3.2.2).

Decoder Ein Decoder wird für die Rekonstruktion der Information des Graphens und seiner Knoten auf Basis der vom Encoder generierten Knoten-Embeddings verwendet. Im Kontext des maschinellen Lernens könnte ein Decoder dazu verwendet werden, um eine Eigenschaft des Knotens u, beispielsweise dessen Nachbarschaft $\mathcal{N}(u)$, auf Basis des Knoten-Embeddings $\boldsymbol{z}_u$ vorherzusagen. Dazu kann beispielsweise die Ähnlichkeit eines Knotenpaars (u, v) herangezogen werden. Die Abbildungsfunktion eines Decoders für ein Knotenpaar kann formal durch

$$\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+ \tag{3.6}$$

beschrieben werden.

Insgesamt kann somit auf Basis eines gegebenen Ähnlichkeitsmaßes für ein Paar von Knoten-Embeddings eine Zielfunktion

$$\text{DEC}(\text{ENC}(u), \text{ENC}(v)) = \text{DEC}(\boldsymbol{z}_u, \boldsymbol{z}_v) \approx \text{S}[u, v] \tag{3.7}$$

formuliert werden, wobei $\text{S}[u, v]$ das graphenbasierte Ähnlichkeitsmaß zwischen den Knoten u und v mit $\text{S} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ist.

Für eine Menge an Knotenpaaren $\mathcal{D}$ kann diese Zielfunktion (vgl. Gl. (3.7)) verwendt werden, um eine Verlustfunktion

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell(\text{DEC}(\boldsymbol{z}_u, \boldsymbol{z}_v), \mathbf{S}[u, v]) \tag{3.8}$$

für die Rekonstruktion zu definieren, die als Optimierungsproblem minimiert werden soll. Dabei ist $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ eine Verlustfunktion, welche der Abweichung von geschätzten bzw. generierten Werten $\text{DEC}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ zu den wahren Werten $\text{S}[u, v]$ entspricht. Die Verlustfunktion ℓ kann dabei variabel gewählt werden. In Kapitel 3.2.4 werden für die verschiedenen Ebenen eines Graphen Verlustfunktionen definiert.

Ein Überblick über die verschiedenen Shallow-Embedding-Methoden und die zugehörigen Decoder DEC, den Ähnlichkeitsmaßen S und den gewählten Verlustfunktionen ℓ, findet sich ausführlich in [Ham20].

3.2.2 Basis Graph Neural Network und Methodologien

Basierend auf den Shallow-Embedding-Methoden gibt es weitere Methoden, die iterativ eine umfassende Repräsentation lernen, um eine Inferenz auf ungesehenen Daten zu ermöglichen. Diese Methoden werden auch als GNNs bezeichnet. Der grundlegende Unterschied liegt in der Verwendung des sogenannten neuronalen Message-Passings. Dabei wird die als Nachricht formatierte Information über Knoten und/oder Kanten zwischen den Knoten iterativ ausgetauscht und in angereicherter Form für die jeweiligen Knoten und Kanten mit Hilfe neuronaler Netze aktualisiert [Ham20], [GSR+17].

Im Folgenden werden die grundlegenden Mechanismen von GNNs beschrieben und eine Basisformulierung eines GNNs vorgestellt, um Knoten-Embeddings $\boldsymbol{z}_u, \forall u \in \mathcal{V}$ bzw. die Embedding-Matrix $\mathbf{Z}$ zu erzeugen.

Message-Passing Das Message-Passing beschreibt einen Ansatz zum Nachrichtenaustausch zwischen Knoten in einem Graphen. Dabei wird ein GNN mit K Message-Passing-Iterationen betrachtet. Jedem Knoten $u \in \mathcal{V}$ ist das versteckte (engl. *hidden*) Embedding $\boldsymbol{h}_u^{(k)}$ zur k-ten Iteration zugeordnet. Jede Iteration besteht aus einem Update-Schritt, bei dem die Information des Knoten $\boldsymbol{h}_u^{(k)}$ zusammen mit der aggregierten Information aus der Nachbarschaft $\mathcal{N}(u)$ des Knotens u aktualisiert wird. Die Aggregation der Information aus der Nachbarschaft $\mathcal{N}(u)$ des Knotens u wird als Nachricht $\boldsymbol{m}_{\mathcal{N}(u)}$ (engl. *message*) bezeichnet und mathematisch beschrieben durch

$$\boldsymbol{m}_{\mathcal{N}(u)}^{(k)} = \textsc{aggregate}\left(\{\boldsymbol{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\}\right). \tag{3.9}$$

Die Nachricht $\boldsymbol{m}_{\mathcal{N}(u)}^{(k)}$, welche die aggregierte Information aus der Nachbarschaft $\mathcal{N}(u)$ des Knotens u beinhaltet, wird in der UPDATE-Funktion, zusammen mit dem Embedding $\boldsymbol{h}_u^{(k)}$ des Knotens u selbst, aus der vorherigen Iteration kombiniert, um das Embedding $\boldsymbol{h}_u^{(k+1)}$ zu generieren. Diese Aktualisierung bzw. das Update, um das versteckte Embedding $\boldsymbol{h}_u^{(k+1)}$ im nächsten Schritt zu erhalten, lässt sich ausdrücken durch

$$\boldsymbol{h}_u^{(k+1)} = \textsc{update}^{(k)}\left(\boldsymbol{h}_u^{(k)}, \boldsymbol{m}_{\mathcal{N}(u)}^{(k)}\right). \tag{3.10}$$

Die Funktionen UPDATE und AGGREGATE sind dabei beliebige, differenzierbare Funktionen, wobei die AGGREGATE-Methode für die Generierung der Nachricht $\boldsymbol{m}_{\mathcal{N}(u)}$ permutationsinvariant sein muss, da es keine eindeutige Ordnung der Knoten in der Nachbarschaft $\mathcal{N}(u)$ eines beliebigen Knotens u eines Graphen $\mathcal{G}$ gibt. Das resultierende Embedding nach jeder Message-Passing-Iteration ist somit invariant gegenüber der Verarbeitungsreihenfolge der Knoten. Beispiele für solche AGGREGATE-Methoden werden in Kapitel 3.2.3 erläutert.

Der Message-Passing-Algorithmus beginnt mit der Iteration $k = 0$ und verfügt somit initial über die versteckten Embeddings $\boldsymbol{h}_u^{(k)} = \boldsymbol{h}_u^0 = x_u, \forall u \in \mathcal{V}$ und dementsprechend die initialen Knotenmerkmale. Bei einer festgelegten Anzahl an Iterationen K ist das finale Embedding für jeden einzelnen Knoten $u \in \mathcal{V}$ definiert als

$$\boldsymbol{z}_u = \boldsymbol{h}_u^{(K)}, \forall u \in \mathcal{V}. \tag{3.11}$$

Verdeutlicht wird dies anhand eines Beispielgraphens in Abbildung 3.5, wobei der Message-Passing-Algorithmus für $K = 2$ Iterationen durchgeführt wird.

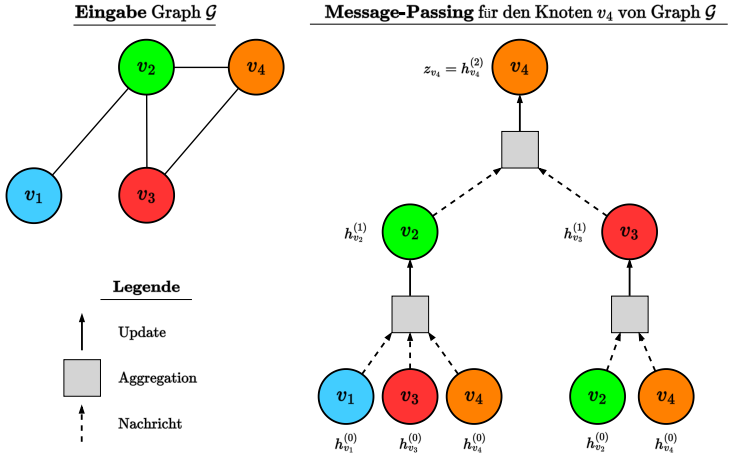


Abbildung 3.5: Visualisierung des Message-Passing-Algorithmus für einen Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Für den Knoten v_4 wird durch $K = 2$ Message-Passing-Iteration die Information der Nachbarschaft $\mathcal{N}(v_4)$, welche über zwei Schritte erreicht werden kann, iterativ zusammengefasst. Zunächst die Information der direkten Nachbarn (Knoten v_2 und v_3) und dann der indirekten Nachbarn (v_1).

Nach K Message-Passing-Iterationen hat das Knoten-Embedding somit alle Informationen von benachbarten Knoten aggregiert, die in k Schritten (engl. k-*hops*) erreichbar sind. Neben der strukturellen Information jedes Knotens werden auch die spezifischen Merkmale der Knoten aggregiert. Dabei kann das Verhalten des Message-Passings nach K Iterationen als Analogon zum Verhalten eines Faltungskernels in Bezug auf CNNs betrachtet werden. In diesem Kontext werden die Iterationen des Message-Passings auch als Schichten (engl. *layer*) eines GNNs verstanden.

Basis-GNN Das beschriebene Verfahren zum Message-Passing auf Knotenebene mit der UPDATE- und AGGREGATE-Funktion (vgl. Gln. (3.10) und (3.9)) kann wie folgt zur Formulierung eines Basis-GNNs (nach [Ham20], ursprünglich vorgeschlagen von [ML05], [GMS05], [SGT+09]), zusammengesetzt werden:

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\boldsymbol{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)}\sum_{v\in\mathcal{N}(u)}\boldsymbol{h}_v^{(k-1)} + \boldsymbol{b}^{(k)}\right), \tag{3.12}$$

wobei $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)}\times d^{(k-1)}}$ jeweils parametrisierbare Gewichtungsmatrizen (dies entspricht den trainierbaren Parametern in neuronalen Netzen), σ eine elementweise Nichtlinearität (z.B. Aktivierungsfunktion, wie das *Rectified Linear Unit* (ReLU) [Fuk69], [NH10], [GBB11]) und $\boldsymbol{b}^{(k)} \in \mathbb{R}^{d^{(k)}}$ einen optionalen Verzerrungsterm (engl. *bias term*), darstellen. Hier wird das Message-Passing-Verfahren unter Verwendung von k als aktueller und $k-1$ als vorheriger Iteration dargestellt.

Folglich können die UPDATE- und AGGREGATE-Funktion inklusive der Nachricht $\boldsymbol{m}_{\mathcal{N}_{(\cdot)}}$ durch

$$\boldsymbol{m}_{\mathcal{N}(u)} = \sum_{v\in\mathcal{N}(u)}\boldsymbol{h}_v, \tag{3.13}$$

$$\text{UPDATE}\left(\boldsymbol{h}_u, \boldsymbol{m}_{\mathcal{N}(u)}\right) = \sigma\left(W_{\text{self}}\boldsymbol{h}_u + W_{\text{neigh}}\boldsymbol{m}_{\mathcal{N}(u)}\right) \tag{3.14}$$

formuliert werden. Zur Vereinfachung werden in der mathematischen Darstellung die Superskripte für die jeweilige Iteration k weggelassen, da sich die UPDATE-Funktion auf den aktuellen Schritt k für alle Komponenten bezieht.

Durch die Verwendung von Selbst-Schleifen für den zu aktualisierenden Knoten $u \in \mathcal{V}$, also eine Kante $e = (u, u)$ mit $e \in \mathcal{E}$, kann eine Message-Passing-Iteration weiter vereinfacht werden, sodass die UPDATE-Funktion durch die Aggregation

$$\boldsymbol{h}_u^{(k)} = \text{AGGREGATE}(\{\boldsymbol{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u) \cup \{u\}\}) \tag{3.15}$$

dargestellt werden kann. Dabei ist die Information des Knoten u selbst in der Aggregation, also der Verarbeitung der Nachbarschaft $\mathcal{N}(u) \cup \{u\}$, enthalten.

Somit wird für diesen Spezialfall das Embedding für die Iteration k mit der angepassten Gl. (3.12) wie folgt bestimmt

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \boldsymbol{h}_v^{(k-1)}\right), \tag{3.16}$$

wobei lediglich eine Gewichtungsmatrix $\mathbf{W}^{(k)}$ benötigt wird, was für die Umsetzung mit neuronalen Netzen bedeutet, dass die Parameter der Gewichte für die Information des Nachbarn und die des Knoten selbst, geteilt werden.

Die beschriebenen Gln. (3.12) und (3.16) können auf den gesamten Graphen bezogen, durch die Matrix $\mathbf{H}^{(k)} \in \mathbb{R}^{|\mathcal{V}| \times d}$ analog zur finalen Embedding-Matrix $\mathbf{Z}$, welche die Embeddingvektoren für jeden Knoten $u \in \mathcal{V}$ der Schicht k eines GNNs enthält, beschrieben werden. Für den allgemeinen Fall ohne Selbst-Schleifen gilt für jede Schicht eines GNNs

$$\mathbf{H}^{(k)} = \sigma\left(\mathbf{A}\mathbf{H}^{(k-1)}\mathbf{W}_{\text{neigh}}^{(k)} + \mathbf{H}^{(k-1)}\mathbf{W}_{\text{self}}^{(k)}\right), \tag{3.17}$$

wobei $\mathbf{A}$ die Adjazenzmatrix des Graphen ist. Wenn Selbst-Schleifen berücksichtigt sind, dann vereinfacht sich die Formulierung der Schicht k zu

$$\mathbf{H}^{(k)} = \sigma\left((\mathbf{A} + \mathbf{I})\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right). \tag{3.18}$$

3.2.3 Varianten von GNNs

In diesem Unterkapitel werden Varianten von GNNs vorgestellt und aufbauend auf den bisher beschriebenen Eigenschaften und Charakteristika des Basis-GNN-

Modells (vgl. Gl. (3.17)) mathematisch formuliert. Von besonderer Bedeutung ist zum einen die Verarbeitung der Informationen in der Nachbarschaft der Knoten (AGGREGATE) und zum anderen die Aktualisierung der Knoten mit der aggregierten Information aus der Nachbarschaft (UPDATE). In beiden dieser Teilaufgaben des Message-Passings können unterschiedliche Ansätze aus dem tiefen maschinellen Lernen zur Generalisierung angewandt werden. Folglich werden ausgewählte Ansätze, die für diese Arbeit von besonderer Bedeutung sind, nach der jeweiligen AGGREGATE und UPDATE Spezifikation kategorisiert und erläutert.

Generalisierung der Nachbarschafts-Aggregation

In diesem Abschnitt werden weiterführende Mechanismen zur Aggregation der Information aus der Nachbarschaft beschrieben und erläutert. Diese Mechanismen basieren auf Methoden, die aus dem tiefen maschinellen Lernen bekannt sind und werden ergänzend in die bisher vorgestellte Aggregation integriert. Für ein weiterführendes Verständnis wird auf das Grundlagenwerk [GBA16] verwiesen.

Nachbarschafts-Normalisierung Die Normalisierung der Nachbarschaft (engl. *neighborhood normalization*) ist eine Methode, um die Aggregation der Nachbarschaft (vgl. Gl. (3.13)) robuster gegenüber numerischen Instabilitäten zu machen, welche sich in den versteckten Embeddings der Nachbarschaft zeigen können. Dabei wird bei der *asymmetrischen* Normalisierung der Knotengrad (vgl. Gl. (3.3)) von u berücksichtigt, sodass die Nachricht $\boldsymbol{m}_{\mathcal{N}(u)}$ aus den versteckten Embeddings $\boldsymbol{h}_v, \forall v \in \mathcal{N}(u)$ erstellt wird, die durch den Knotengrad $\text{degree}(u)$ normalisiert sind. Beschrieben wird dies durch

$$\boldsymbol{m}_{\mathcal{N}(u)} = \frac{\sum_{v \in \mathcal{N}(u)} \boldsymbol{h}_v}{|\mathcal{N}(u)|}. \tag{3.19}$$

Diese Methode zur Normalisierung wurde von [KW16a] weitergeführt und als *symmetrische* Normalisierung durch

$$\boldsymbol{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\boldsymbol{h}_v}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}} \tag{3.20}$$

beschrieben. Dies ist durch die Approximation der spektralen Faltung auf Graphen erster Ordnung [HVG11], [DBV16] motiviert. Diese symmetrische Normalisierung definiert das sogenannte *Graph Convolutional Network* (GCN) mit einer GCN-Schicht[1] und wird auf Knotenebene durch

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\boldsymbol{h}_v}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}}\right) \tag{3.21}$$

beschrieben. Die GCN-Schicht wird auf Graphen-Ebene durch

$$\mathbf{H}^{(k)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k)}\right) \tag{3.22}$$

definiert, wobei auch hier, wie in Gl. (3.18) vorgestellt, $\mathbf{W}^{(t)}$ eine lernbare Gewichtungsmatrix ist, und

$$\tilde{\mathbf{A}} = (\mathbf{D} + \mathbf{I})^{-\frac{1}{2}}(\mathbf{I} + \mathbf{A})(\mathbf{D} + \mathbf{I})^{-\frac{1}{2}} \tag{3.23}$$

der normalisierten Adjazenzmatrix $\tilde{\mathbf{A}}$ mit Selbst-Schleifen, $\mathbf{D}$ der Gradmatrix und $\mathbf{I}$ der Identität entspricht.

Nachbarschafts-Aggregation Neben der Möglichkeit die Nachbarschafts-Aggregation durch eine Normalisierung (vgl. Gl. (3.19)) zu verbessern, gibt es weitere Möglichkeiten für eine Robustifizierung der Nachbarschafts-Aggregation. Alternativ kann beispielsweise die Menge der Knoten-Embeddings $\{\boldsymbol{h}_v, \forall v \in \mathcal{N}(u)\}$ nicht durch Aufsummierung, sondern durch eine alternative AGGREGATE-Funktion, aggregiert werden. Die geltende Bedingung der Permutationsinvarianz muss dabei für jede alternative AGGREGATE-Funktion gewahrt sein. In [HYL18] wird passend hierzu eine Platzhalterfunktion AGGREGATE_k für jede k-te GNN-Schicht/Message-Passing-Iteration eingeführt (vgl. auch Gl. (3.9)), die durch verschiedene Varianten umgesetzt werden kann. Die intuitivsten Varianten (vgl. S. 51) sind dabei die Summen-, Mittelwertsbildung oder das Maximum (elementweise im Vektor) als AGGREGATE_k-Funktion für jede Schicht k zu wählen. Außerdem wird in [HYL17], [HYL18] eine alternative UPDATE-Funktion vorgestellt, welche die Information aus der Nachbarschaft

[1] Beschrieben mit Selbst-Schleifen: $\mathcal{N}(u) \cup \{u\}$

$\{\boldsymbol{h}_v, \forall v \in \mathcal{N}(u)\}$ und der Information des Knoten u selbst aus der vorherigen Message-Passing-Iteration, nicht wie üblich durch eine Aufsummierung, sondern durch eine Konkatenation zusammenfasst. Dadurch kann mehr Information über das einzelne Knoten-Embedding und somit den Knoten erhalten bleiben. Auf S. 49 bzw. in Gl. (3.37) wird dies ausführlicher beschrieben.

Über die intuitiven Varianten der Aggregation der Nachbarschaft hinaus (Aufsummierung, Mittelwertsbildung und Maximum), können in den Arbeiten von [HYL18], [ZKR+17], [MSRR18], [XLT+18] weitere Varianten nachvollzogen werden, welche beispielsweise MLPs oder aber LSTMs zur Aggregation der Informationen aus der Nachbarschaft hinzuziehen.

Nachbarschafts-Aufmerksamkeit Unter dem Begriff der Nachbarschafts-Aufmerksamkeit (engl. *neighborhood attention*) ist zunächst ein Mechanismus zum Lernen von Aufmerksamkeit (engl. *attention*) gemeint, wobei hier zwischen *additiver* Aufmerksamkeit [BCB16] oder *multiplikativer* Aufmerksamkeit [BGLL17], [VSP+17] unterschieden wird. Beschrieben wird dabei in beiden Varianten das Lernen einer Gewichtung für die Verwertung von Informationen.

Im Kontext von GNNs wurde dieser Mechanismus in der Arbeit von [VCC+18] durch das sogenannte *Graph Attention Network* (GAT) vorgestellt, welcher auf den Aufmerksamkeitsmechanismus von [BCB16] zurückzuführen ist. Das GAT verwendet den Aufmerksamkeitsmechanismus, um zusätzliche Aufmerksamkeitsgewichte (engl. *attention weights*) für den Einfluss der einzelnen Nachbarn $v \in \mathcal{N}(u)$ während der Aggregation zu berücksichtigen. Somit wird die Nachricht $\boldsymbol{m}_{\mathcal{N}(u)}$ wie folgt definiert

$$\boldsymbol{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \boldsymbol{h}_v, \tag{3.24}$$

wobei $\alpha_{u,v}$ das Gewicht für jeden Nachbarn $v \in \mathcal{N}(u)$ während der Bestimmung des Embeddings $\boldsymbol{h}_u$ des Knoten u ist. In [VCC+18] werden die Aufmerksamkeitsgewichte als

$$\alpha_{u,v} = \frac{\exp\left(\sigma_{\text{LeakyReLU}}\left(\mathbf{a}^\top [\boldsymbol{W}\boldsymbol{h}_u \oplus \boldsymbol{W}\boldsymbol{h}_v]\right)\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\sigma_{\text{LeakyReLU}}\left(\mathbf{a}^\top [\boldsymbol{W}\boldsymbol{h}_u \oplus \boldsymbol{W}\boldsymbol{h}_{v'}]\right)\right)}. \tag{3.25}$$

definiert, wobei **a** einem lernbaren Vektor für die Aufmerksamkeit, das sogenannte *Leaky Rectified Linear Unit* (LeakyReLU) [MHN+13] einer Aktivierungsfunktion, **W** einer lernbaren Gewichtungsmatrix und $\oplus$ der Konkatenation zweier Vektoren entsprechen. Das Knoten-Embedding $\boldsymbol{h}_u^{(k)}$ wird somit definiert als

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{W}^{(k)} \boldsymbol{h}_v\right). \tag{3.26}$$

In der Arbeit [BAY22] wurde mit dem sogenannten GATv2 eine Verbesserung des Einflusses des lernbaren Aufmerksamkeitsvektors **a** bei der Bestimmung der Aufmerksamkeitsgewichte α vorgeschlagen. Die resultierenden Aufmerksamkeitsgewichte $\alpha^{(\mathrm{v2})}$ für das GATv2 werden definiert als

$$\alpha_{u,v}^{(\mathrm{v2})} = \frac{\exp\left(\mathbf{a}^\top \sigma_{\text{LeakyReLU}}\left(\boldsymbol{W}\left[\boldsymbol{h}_u \oplus \boldsymbol{h}_v\right]\right)\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{a}^\top \sigma_{\text{LeakyReLU}}\left(\boldsymbol{W}\left[\boldsymbol{h}_u \oplus \boldsymbol{h}_{v'}\right]\right)\right)}. \tag{3.27}$$

Die Verwendung des Aufmerksamkeitsmechanismus kann durch die Nutzung von mehrfachen Aufmerksamkeitsköpfen (engl. *multi-head attention*) erweitert werden, wie in der Arbeit von [VSP+17] zu Multi-Head-Attention und Transformern vorgestellt. Für die Aggregation der Nachricht $\boldsymbol{m}_{\mathcal{N}(u)}$ bedeutet dies, dass Ξ unterschiedliche Aufmerksamkeitsgewichte je Attention-Head für jeden Nachbarn $v \in \mathcal{N}(u)$, also $\alpha_{u,v,\xi}$ für $\xi \in \Xi$ gelernt werden. Somit gilt

$$\boldsymbol{m}_{\mathcal{N}(u)} = \left[\mathbf{a}_1 \oplus \mathbf{a}_2 \oplus \cdots \oplus \mathbf{a}_\Xi\right] \tag{3.28}$$

$$\mathbf{a}_\xi = \mathbf{W}_\xi \sum_{v \in \mathcal{N}(u)} \alpha_{u,v,\xi} \boldsymbol{h}_v. \tag{3.29}$$

Demnach kann das Embedding im Schritt k (vgl. [VCC+18]) definiert werden als

$$\boldsymbol{h}_u^{(k)} = \bigoplus_{\xi=1}^{\Xi} \sigma\left(\sum_{v \in \mathcal{N}(u)} \alpha_{u,v,\xi} \mathbf{W}_\xi^{(k)} \boldsymbol{h}_v\right), \tag{3.30}$$

wobei $\oplus$ der Konkatenation (vgl. Gl. (3.28)) entspricht. Die Dimension der Knoten-Merkmale wird für einen Knoten $v \in \mathcal{V}$ um den Faktor Ξ zu $\Xi \cdot d$ größer. Somit muss für eine Vorhersage in der letzten Schicht, beispielsweise der Durchschnitt über Ξ Köpfe gebildet werden, um die ursprüngliche Dimension

für jedes Knoten-Embedding wiederherzustellen. Mathematisch kann dies durch eine Mittelwertsbildung über die Summe $\sum_{\xi=1}^{\Xi}$ als

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\frac{1}{\Xi}\sum_{\xi=1}^{\Xi}\sum_{v\in\mathcal{N}(u)}\alpha_{u,v,\xi}\mathbf{W}_{\xi}^{(k)}\boldsymbol{h}_v\right) \tag{3.31}$$

beschrieben werden.

Eine detaillierte Anwendung des Multi-Head-Attention-Mechanismus kann in [VCC+18] nachvollzogen werden, wo das vorgestellte GAT um Multi-Head-Attention erweitert wurde.

Die im Kontext von Transformern [VSP+17] vorgestellte *Scaled Dot-Product Attention* ist eine weitere Variante Aufmerksamkeitsgewichte zu lernen, die auch im *Graph Transformer* von [SHF+21] angewandt wird.

Im Kontext der Sprachverarbeitung besteht ein Attention-Mechanismus aus Abfrage **Q** (engl. *query*), Schlüssel **S** (engl. *key*) und Wert (engl. *value*), welche zur Verarbeitung der Eingabeinformation verwendet werden. Dieser Sachverhalt setzt sich für den Graphkontext dabei wie folgt zusammen. Der Abfragevektor $\boldsymbol{q} \in \mathbf{Q}$ wird durch die Information über den Knoten u und der Schlüsselvektor $\boldsymbol{s} \in \mathbf{S}$ über die Information über den Knoten v und der optionalen Information der Kante $e = (u, v)$ dargestellt. Somit werden die Aufmerksamkeitsgewichte $\alpha_{u,v,\xi}$ für jedes Knotenpaar (u, v) zum Multi-Head ξ mit $\boldsymbol{q}_{u,\xi} \in \mathbb{R}^d$ und $\boldsymbol{s}_{v,\xi} \in \mathbb{R}^d$, analog zur Arbeit [SHF+21], definiert als

$$\begin{aligned}
\boldsymbol{q}_{u,\xi} &= \mathbf{W}_{\boldsymbol{q},\xi}\boldsymbol{h}_u \\
\boldsymbol{s}_{v,\xi} &= \mathbf{W}_{\boldsymbol{s},\xi}\boldsymbol{h}_v \\
e_{u,v,\xi} &= \mathbf{W}_{e,\xi}e_{u,v} \\
\alpha_{u,v,\xi} &= \frac{\left\langle \boldsymbol{q}_{u,\xi}, \boldsymbol{s}_{v,\xi} + e_{u,v,\xi}\right\rangle}{\sum_{v'\in\mathcal{N}(u)}\left\langle \boldsymbol{q}_{u,\xi}, \boldsymbol{s}_{v',\xi} + e_{u,v',\xi}\right\rangle}
\end{aligned} \quad , \tag{3.32}$$

wobei $\langle \boldsymbol{q}, \boldsymbol{s}\rangle = \exp\left(\frac{\boldsymbol{q}^T s}{\sqrt{d}}\right)$ und d der Dimension jedes Vektors aus ξ entspricht. Analog zum GAT (vgl. Gln. (3.30) und (3.31)) kann beim Graph Transformer die Schicht des Knoten-Embeddings $\boldsymbol{h}_u^{(k)}$ als

$$\boldsymbol{h}_u^{(k)} = \bigoplus_{\xi=1}^{\Xi} \sigma\left(\sum_{v \in \mathcal{N}(u)} \alpha_{u,v,\xi}(\mathbf{W}_\xi^{(k)} \boldsymbol{h}_v + e_{u,v,\xi})\right) \tag{3.33}$$

und

$$\boldsymbol{h}_u^{(k)} = \sigma\left(\frac{1}{\Xi}\sum_{\xi=1}^{\Xi}\sum_{v \in \mathcal{N}(u)} \alpha_{u,v,\xi}(\mathbf{W}_\xi^{(k)} \boldsymbol{h}_v + e_{u,v,\xi})\right) \tag{3.34}$$

definiert werden.

Nachbarschafts-Relationen Wie in Kapitel 3.1 beschrieben, gibt es neben homogenen Graphen auch heterogene Graphen, welche über verschiedene Kantentypen bzw. Relationen $r \in \mathcal{R}$ verfügen und besondere Verbindungen zwischen Knotenpaaren (u, v) ausdrücken. Diese Information kann explizit in der Nachbarschafts-Aggregation berücksichtigt werden, wie in [SKB+18] durch sogenannte *Relational Graph Convolutional Networks* (RGCN) vorgeschlagen wurde. Dabei wird die AGGREGATE-Funktion im Message-Passing-Verfahren durch die Relationstypen um eine separate Aggregation pro Relation $r \in \mathcal{R}$ zu

$$\boldsymbol{m}_{\mathcal{N}(u)} = \sum_{r \in \mathcal{R}} \sum_{v \in \mathcal{N}_r(u)} \frac{\mathbf{W}_r \boldsymbol{h}_v}{f_n(\mathcal{N}(u), \mathcal{N}(v))} \tag{3.35}$$

erweitert. Dabei ist f_n eine Funktion zur Normalisierung der Nachbarschaften $\mathcal{N}(u)$ und $\mathcal{N}(v)$. In [KW16a] wurde diese beispielsweise als symmetrische Normalisierung eingeführt (vgl. Gl. (3.20)). Weitere Varianten werden in [SKB+18] diskutiert. Dieses grundlegende Prinzip, Relationen im Lernvorgang zu berücksichtigen, hat jedoch den Nachteil, dass mit steigender Anzahl an Relationen auch die Anzahl der lernbaren Parameter durch die separaten Gewichtungsmatrizen steigen. Dazu werden in [SKB+18] zwei Ansätze zur entsprechenden Regularisierung vorgeschlagen. Der eine unterteilt die Gewichte in relationsspezifische Matrizen (definiert als *basis-decomposition*), während der andere relationsspezifische Gewichte in Diagonal-Blöcke innerhalb einer einzelnen Matrix strukturiert (definiert als *block-diagonal-decomposition*).

Auch das beschriebene GAT-Netzwerk [VCC+18] kann analog zum RGCN [SKB+18] zur Berücksichtigung von Relationen erweitert werden. In der Arbeit von [BSCH19] kann dieses GNN, welches als *Relational Graph Attention Network* (RGAT) definiert ist, detaillierter nachvollzogen werden.

Generalisierte Update-Methoden

Neben den Erweiterungs- und Verbesserungsmöglichkeiten für die AGGREGATE-Methode (vgl. S. 43) im Rahmen des Message-Passings gibt es auch Mechanismen, um die propagierte Basis-UPDATE-Methode (vgl. Gl. (3.14)) zu optimieren. Hauptsächlich soll dabei eine Überglättung (engl. *over-smoothing*) [OS20] der lokalen Information während des Message-Passings verhindert werden.

Wie in Gl. (3.14) dargestellt, besteht die UPDATE-Methode im einfachsten Fall lediglich aus einer Linearkombination aus der Selbst-Information eines Knotens u und der Information aus dessen Nachbarschaft $\mathcal{N}_{(u)}$. Abhängig von der Anzahl der Message-Passing-Iterationen kann es bei zunehmend vielen Iterationen zur Überglättung der ursprünglichen Knoten-Information kommen. Im Folgenden werden Methoden besprochen, die diesem Problem entgegenwirken. Für eine ausführliche Beschreibung der Problematik werden interessierte Lesende auf [XLT+18] [Ham20] verwiesen.

Skip-Verbindungen Eine Methode zur Vermeidung von Überglättung oder Informationsverlust sind sogenannte Skip-Verbindungen (engl. *skip-connections*). Das Ziel ist es, die Information des Knotens, dessen Embedding aktualisiert werden soll, explizit in das UPDATE miteinzubeziehen, bevor dessen Information durch das Message-Passing irreversibel ausgeblendet wird . Ausgehend von der regulären UPDATE-Methode (vgl. Gl. (3.14)), zur Lesbarkeit definiert als $\text{UPDATE}_{\text{basis}}\left(\boldsymbol{h}_u, \boldsymbol{m}_{\mathcal{N}(u)}\right)$, ist somit die einfachste Art von Skip-Verbindung eine simple Addition des Knoten-Embeddings $\boldsymbol{h}_u$ auf das Ergebnis der UPDATE-Methode, definiert als

$$\text{UPDATE}_{\text{add}}\left(\boldsymbol{h}_u, \boldsymbol{m}_{\mathcal{N}(u)}\right) = \text{UPDATE}_{\text{basis}}\left(\boldsymbol{h}_u, \boldsymbol{m}_{\mathcal{N}(u)}\right) + \boldsymbol{h}_u. \tag{3.36}$$

Diese Art von Skip-Verbindung ist ähnlich zu den residualen Verbindungen (engl. *residual connections*) [HZRS16a], jedoch angepasst für graphstrukturierte Daten.

Neben der Addition des Knoten-Embeddings zur $\textsc{update}_{\text{basis}}$-Funktion ist eine zweite Möglichkeit die Konkatenation. Formuliert wird dies durch

$$\textsc{update}_{\text{concat}}\left(\boldsymbol{h}_u, \boldsymbol{m}_{\mathcal{N}(u)}\right) = \left[\textsc{update}_{\text{basis}}\left(\boldsymbol{h}_u, \boldsymbol{m}_{\mathcal{N}(u)}\right) \oplus \boldsymbol{h}_u\right]. \tag{3.37}$$

Dieser Ansatz geht auf die Arbeit von [HYL18] zurück, in welcher die GNN-Variante *GraphSage* vorgestellt wurde. Diese Variante verfolgt das Ziel, während des Message-Passings die Information der Nachbarn $\boldsymbol{m}_{\mathcal{N}}(u)$ von den knotenspezifischen Embeddings $\boldsymbol{h}_u$ getrennt zu verarbeiten und somit mehr Information zu erhalten als bei einer Addition. Für eine weiterführende Auseinandersetzung mit Skip-Verbindungen wird auf die Arbeiten [XZJK21] [XLT+18], [Ham20] verwiesen.

Jumping-Knowledge-Verbindungen Ein weiterer Ansatz zur verbesserten Bildung der Knoten-Embeddings ist das Anwenden sogenannter Jumping-Knowledge-Verbindungen (JK) [XLT+18], welche die Knoten-Embeddings aus jedem Message-Passing-Schritt $k \in \{0, 1, 2, \ldots, K\}$ nutzen, statt nur die des letzten Schrittes K ($\boldsymbol{z}_u = \boldsymbol{h}_u^K, \forall u \in \mathcal{V}$) zu nutzen. Bei Verwendung von JK wird das finale Embedding $\boldsymbol{z}_u$ somit als

$$\boldsymbol{z}_u = f_{\text{JK}}\left(\boldsymbol{h}^{(0)} \oplus \boldsymbol{h}^{(1)} \oplus \cdots \oplus \boldsymbol{h}^{(K)}\right) \tag{3.38}$$

definiert, wobei $f_{\text{JK}}(\cdot)$ eine beliebige differenzierbare Funktion ist.

Gated-Updates Unter dem Konzept von *Gated-Updates* ist insbesondere die Verwendung von Toren (engl. *gates*) zur Regularisierung des Informationsflusses innerhalb neuronaler Netzwerke gemeint und bezieht sich dabei auf die in den Arbeiten zu LSTMs [HS97] und zu GRUs [CvG+14], [CvBB14] eingeführten Gate-Mechanismen. Auch im Zusammenhang mit graphstrukturierten Daten und insbesondere zur Verbesserung der $\textsc{update}$-Methode im Message-Passing von GNNs können Gates verwendet werden, um den Informationsfluss zu regulieren. Der Message-Passing-Algorithmus innerhalb von RNNs lässt sich dahingehend verstehen, dass die $\textsc{aggregate}$-Methode die Beobachtungen oder Zustände der Nachbarschaft zusammenfasst, um dann den versteckten Zustand (engl. *hidden state*) eines Knotens zu aktualisieren. Somit könnte zur Aktualisierung, genauer als $\textsc{update}$-Methode für den versteckten Zustand,

ein RNN-basiertes GNN-Modul verwendet werden. In [LTBZ16] wird das sogenannte *Gated Graph Sequence Neural Network* vorgestellt, welches ein GNN mit einer UPDATE-Methode auf Basis eines GRUs [CvG+14], [CvBB14] für das Knoten-Embedding $\boldsymbol{h}_u^{(k)}$ als

$$\boldsymbol{h}_u^{(k)} = \text{GRU}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{m}_{\mathcal{N}(u)}^{(k)}\right) \tag{3.39}$$

beschreibt. Im Rahmen dieser Arbeit werden GNNs mit einem zugrundeliegenden rekurrenten Mechanismus als *Graph Recurrent Neural Network* (GRNN) bezeichnet. Dabei können diese GRNNs auch über Gates verfügen.

Als *Gated Graph Neural Networks* (GGNNs) werden GNNs bezeichnet, die über einen Gate-Mechanismus verfügen. In [BL18] werden *Residual Gated Graph ConvNets* (ResGGCN) vorgestellt, die als Verbesserung der UPDATE-Methode von GNNs dienen. Dabei wird ein GGNN beschrieben, welches ohne rekurrente Einheit für die Aktualisierung des versteckten Zustands verantwortlich ist. Zentral ist hierbei die Definition der UPDATE-Methode, welche durch

$$\boldsymbol{h}_u^{(k)} = \mathbf{W}_1 \boldsymbol{h}_u^{(k-1)} + \sum_{v \in \mathcal{N}(u)} \eta_{u,v} \odot \mathbf{W}_2 \boldsymbol{h}_v^{(k-1)} \text{ mit} \tag{3.40}$$

$$\eta_{u,v} = \sigma\left(\mathbf{W}_3 \boldsymbol{h}_u^{(k-1)} + \mathbf{W}_4 \boldsymbol{h}_v^{(k-1)}\right) \tag{3.41}$$

beschrieben wird, wobei $\mathbf{W}_{1\ldots4}$ Gewichtungsmatrizen, $\eta_{u,v}$ die Gates sind und $\odot$ die elementweise Multiplikation darstellt. Das vollständige ResGGCN mit der residualen Verbindung (vgl. [HZRS16a]) wird definiert durch

$$\boldsymbol{h}_u^{(k)} = \mathbf{W}_1 \boldsymbol{h}_u^{(k-1)} + \sum_{v \in \mathcal{N}(u)} \eta_{u,v} \odot \mathbf{W}_2 \boldsymbol{h}_v^{(k-1)} + \boldsymbol{h}_u^{(k-1)}. \tag{3.42}$$

Graph-Pooling

Die bisherige besprochene Theorie zu GNNs beschreibt den Lernvorgang bzw. die Generierung von Embeddings $\boldsymbol{z}_u, \forall u \in \mathcal{V}$ für jeden einzelnen Knoten $u \in \mathcal{V}$ eines Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Wie bereits in Kapitel 3.2.1 erläutert, gibt es beispielsweise die Möglichkeit für Knotenpaare (u, v) eine Aussage über die Ähnlichkeit durch eine Verlustfunktion zu treffen (vgl. Rekonstruktionsfehler Gl. (3.8)). Somit wird durch die Ähnlichkeit zweier Knoten eine Aussage

über den Fehler gemacht, der sich aus der Differenz zwischen der wahren Ähnlichkeit und derjenigen des Decoders ergibt. In diesem Unterkapitel werden gängige Ansätze für das Lernen/Ableiten einer Repräsentation für den gesamten Graphen $\boldsymbol{z}_{\mathcal{G}}$ (auch: *Graph-Pooling*) erläutert.

Wie in den Kapiteln 3.2.2 und auf S. 43 erläutert, gibt es unterschiedliche Ansätze zur Realisierung der AGGREGATE-Methode für die permutationsinvariante Nachbarschafts-Aggregation im Message-Passing-Algorithmus. Um ein Graph-Embedding $\boldsymbol{z}_{\mathcal{G}}$ aus der Menge an Knoten-Embeddings $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{|\mathcal{V}|}\}$ abzuleiten, wird eine POOLING-Funktion f_p benötigt. Die intuitivsten Ansätze für eine POOLING-Funktion f_p zum Ableiten eines Graph-Embeddings $\boldsymbol{z}_{\mathcal{G}}$ sind beispielsweise Summe, Mittelwert oder Maximum

$$\boldsymbol{z}_{\mathcal{G}} := f_{\text{sum}}(\mathbf{Z}) = \sum\nolimits_{i=1}^{|\mathcal{V}|} \boldsymbol{z}_i, \tag{3.43}$$

$$\boldsymbol{z}_{\mathcal{G}} := f_{\text{mean}}(\mathbf{Z}) = \frac{1}{|\mathcal{V}|} \sum\nolimits_{i=1}^{|\mathcal{V}|} \boldsymbol{z}_i, \tag{3.44}$$

$$\boldsymbol{z}_{\mathcal{G}} := f_{\text{max}}(\mathbf{Z}) = \max\nolimits_{i=1}^{|\mathcal{V}|} \boldsymbol{z}_i. \tag{3.45}$$

Neben den deterministischen Ansätzen zur Bildung eines Graph-Embeddings $\boldsymbol{z}_{\mathcal{G}}$ gibt es weitere Methoden, welche meist für komplexere und größere Graphen geeignete POOLING-Funktionen (meist unter Verwendung von MLPs zur Aggregation) darstellen. Für einen weiterführenden Einblick zu Graph-Pooling-Methoden wird auf die Arbeiten [MSRR18], [GZBA22], [LZW+23], [Ham20] verwiesen.

3.2.4 Anwendung auf verschiedenen Ebenen von Graphen

Basierend auf der bisher besprochenen Theorie zu Message-Passing und GNNs kann die Frage nach der Formulierung der Zielgröße für die jeweilige Lernaufgabe und einer Verlustfunktion beschrieben werden. Die drei Ebenen des Graphen (vgl. Abbildung 3.6) werden durch die Embeddingvektoren $\boldsymbol{z}_u \in \mathbb{R}^{d_u}$ für die Knoten-Ebene, $\boldsymbol{z}_{\mathcal{G}} \in \mathbb{R}^{d_{\mathcal{G}}}$ für die Graph-Ebene und $\boldsymbol{z}_{u,v} \in \mathbb{R}^{d_{u,v}}$ für die Kanten-Ebene nach dem letzten Schritt des Message-Passings bzw. der letzten Schicht des GNNs repräsentiert.

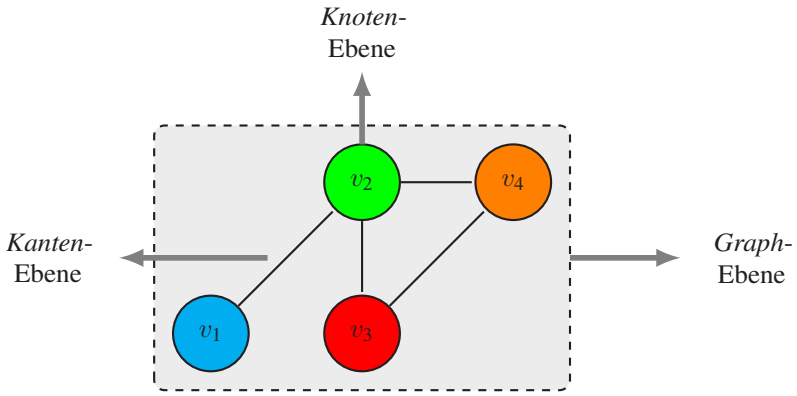


Abbildung 3.6: Die Prädiktions-Ebenen von graphstrukturierten Daten umfassen die *Knoten*-, *Kanten*- und *Graph*-Ebene.

Im tiefen maschinellen Lernen wird die Zielgröße oftmals als $\boldsymbol{y}$ beschrieben, wobei $\boldsymbol{y}$ der Wahrheit und $\hat{\boldsymbol{y}}$ der geschätzen Größe (Prädiktion) entspricht. Um aus dem gelernten Embeddingvektor $\boldsymbol{z}$ die Schätzung $\hat{\boldsymbol{y}}$ abzuleiten, wird eine Schicht zum Auslesen (engl. *readout layer*) benötigt:

$$\hat{\boldsymbol{y}} = \textsc{readout}(\boldsymbol{z}). \tag{3.46}$$

Die READOUT-Schicht selbst kann dabei durch eine lineare Schicht, ein MLP, eine SOFTMAX-Schicht oder weitere Schichten umgesetzen werden, abhängig von der Lernaufgabe.

Im Folgenden wird besonders auf die Lernaufgaben im Kontext von GNNs und der verschiedenen Prädiktions-Ebenen (vgl. Abbildung 3.6) eingegangen. Interessierte Lesende werden auf [Ham20] [BHB+18] verwiesen.

Knoten-Ebene

Betrachtet werden hierbei einzelne Knoten $u \in \mathcal{V}$ bzw. die resultierenden Knoten-Embeddings $\boldsymbol{z}_u$ aus der letzten Schicht des GNNs. Für das Knoten-Embedding $\boldsymbol{z}_u$ wird durch die Verlustfunktion das Lernziel determiniert. Dies

wird nachfolgend für die *Knoten*-Ebene als die Knoten-Klassifikation und Knoten-Regression beschrieben.

Knoten-Klassifikation Bei der Knoten-Klassifikation (engl. *node classification*) handelt es sich um die Vorhersage der Kennzeichnung für jeden Knoten $u \in \mathcal{V}$ eines Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$.

Gegeben ist für jeden Knoten u ein Kennzeichen $\boldsymbol{y}_u$, welches normalerweise als One-Hot-Kodierung (engl. *one-hot encoding*), also einer Vektorrepräsentation in der alle Komponenten den Wert 0 und eines den Wert 1 hat, abgebildet wird. Aus dem gelernten Embedding $\boldsymbol{z}_u \in \mathbb{R}^d$ wird die Schätzgröße $\hat{\boldsymbol{y}}_u$ ausgelesen. Bei der Klassifikation wird dabei eine READOUT-Schicht benötigt, welche hinsichtlich der Klassifikation eine C-dimensionale Klassenverteilung für jede Klasse $c \in \{0, 1, 2, \ldots, C\}$, ausgibt. Dabei kann beispielsweise ein MLP für eine unnormalisierte Klassenverteilung, oder eine SOFTMAX-Schicht für eine normalisierte Klassenverteilung verwendet werden. Die SOFTMAX-Schicht wird durch

$$\text{SOFTMAX}(\hat{\boldsymbol{y}}_u) = \frac{\exp(\hat{\boldsymbol{y}}_{u,c})}{\sum_{i=1}^{C} \exp(\hat{\boldsymbol{y}}_{u,i})} \tag{3.47}$$

definiert. Um den Verlust zwischen der Wahrheit $\boldsymbol{y}_u$ und der Prädiktion $\hat{\boldsymbol{y}}_u$ zu bestimmen, kann die *Kreuzentropie*-Verlustfunktion (oder auch *Log*-Verlust) $\mathcal{L}_{\text{CE}}$ für den Knoten u und für C Klassen verwendet werden

$$\mathcal{L}_{\text{CE},u}(\boldsymbol{y}_u, \hat{\boldsymbol{y}}_u) = -\sum_{c=1}^{C} \boldsymbol{y}_{u,c} \log\left(\text{SOFTMAX}(\hat{\boldsymbol{y}}_u)\right) \tag{3.48}$$

wobei $\boldsymbol{y}_u = (\boldsymbol{y}_{u,1}, \ldots, \boldsymbol{y}_{u,C})$ und analog für $\hat{\boldsymbol{y}}_u$. Dabei beinhaltet die *Kreuzentropie*-Verlustfunktion die SOFTMAX-Schicht, um die vorhergesagten Werte in eine Wahrscheinlichkeitsklassenverteilung zu transformieren. Dies sorgt dafür, dass die Verlustberechnung direkt auf Wahrscheinlichkeiten basiert und die numerischen Berechnungen stabilisiert werden.

Knoten-Regression Analog zur Knoten-Klassifikation kann mit einem Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$ auch eine Knoten-Regression durchgeführt werden. Nachdem die Embedding-Matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ durch eine READOUT-Schicht (vgl. Gl. (3.46)) in die Dimension der Wahrheit $\boldsymbol{y}_u \in \mathbb{R}^3$ für eine 3D-Position transformiert

wurde, kann durch eine Verlustfunktion der Fehler zwischen den vorhergesagten und den tatsächlichen Werten für jeden Knoten bestimmt werden.

Für die Regression auf Knotenebene können die gängigen Verlustfunktionen verwendet werden, welche nachfolgend genauer beschrieben werden. Eine Verlustfunktion $\mathcal{L}$ wird dabei für eine Menge an Knoten $\mathcal{V}$ durch ein Fehlermaß für jeden Knoten $u \in \mathcal{V}$ formuliert.

Mittlerer quadratischer Fehler Der mittlere quadratische Fehler (engl. *Mean Squared Error*) (MSE) wird durch

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} (\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)^2 \tag{3.49}$$

beschrieben. Dabei repräsentiert $\boldsymbol{y}_i$ den wahren Wert für den Knoten i mit $\boldsymbol{y}_i \in \mathbb{R}^3$ und $\hat{\boldsymbol{y}}_i$ die zugehörige Vorhersage für den Knoten i mit $\hat{\boldsymbol{y}}_i \in \mathbb{R}^3$.

Mittlerer absoluter Fehler Ein weiteres Fehlermaß ist der mittlere absolute Fehler (engl. *Mean Absolute Error*) (MAE)

$$\mathcal{L}_{\text{MAE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} |\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i| \,. \tag{3.50}$$

Hier wird der Betrag (auch: *L1-Norm*) anstelle des Fehlerquadrats für alle Knotenpaare aus Wahrheit und Vorhersage gebildet und deren Summe gemittelt.

Die Verlustfunktion wird oft anhand des Problems gewählt. Ist der Hintergrund für eine Regression beispielsweise die Vorhersage von Bewegungstrajektorien, welche durch n-dimensionale Punkte beschrieben werden, können auch spezifisch für diese weitere Verlustfunktionen herangezogen werden. Eine Übersicht diesbezüglich lässt sich in der Arbeit von [RPH+20] finden. Da im Rahmen dieser Arbeit ein Verfahren zur Bewegungsvorhersage auf Basis von 3D-Positionsdaten erarbeitet wurde, wird eine weitere Verlustfunktion für die Umsetzung des Verfahrens zur Bewegungsvorhersage im Kapitel 4.2.4 näher beschrieben.

Kanten-Ebene

Analog zur Betrachtung der Knoten-Ebene (vgl. S. 53) können die Kanten $e \in \mathcal{E}$ eines Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$ auch als Zielgröße verwendet werden. Im bisher beschriebenen Message-Passing-Verfahren (vgl. Kapitel 3.2.2) werden nur Embeddings für die einzelnen Knoten $\boldsymbol{z}_u$ sowie für den gesamten Graphen $\boldsymbol{z}_\mathcal{G}$ gelernt und anschließend durch Pooling-Mechanismen (vgl. S. 51) für die Gesamtheit aller Knoten abgeleitet. Dennoch ist es über die Kombination der Embeddings zweier Knoten (z_u, z_v) möglich, auch Lernaufgaben und Verlustfunktionen auf Kantenebene zu formulieren. Dies wird im Nachfolgenden für die Kanten-Vorhersage beschrieben.

Kanten-Vorhersage Bei der Kanten-Vorhersage (engl. *link prediction*) handelt es sich um die Vorhersage $\hat{\boldsymbol{y}}_{u,v}$, ob eine Kante zwischen zweier Knoten $(u, v) \in \mathcal{V}$ eines Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$ existiert oder nicht [ZC18], [GL16]. Diese Vorhersage kann dann mit der tatsächlichen Kennzeichnung $\boldsymbol{y}_{u,v} = [e \in \mathcal{E}]$ zur Wahrheitsprüfung verglichen werden. Für eine solche Aufgabe kann die *Kreuzentropie*-Verlustfunktion für binäre Klassen (engl. *binary cross-entropy loss*) $\mathcal{L}_{\text{BCE}}$ verwendet werden, welche einen Spezialfall der in Gl. (3.48) eingeführten *Kreuzentropie*-Verlustfunktion $\mathcal{L}_{\text{CE}}$ darstellt. Die binäre Kreuzentropie-Verlustfunktion wird für den vorliegenden Fall als

$$\mathcal{L}_{\text{BCE},u,v}(\boldsymbol{y}_{u,v}, \hat{\boldsymbol{y}}_{u,v}) = -(\boldsymbol{y}_{u,v} \log(\hat{\boldsymbol{y}}_{u,v}) + (1 - \boldsymbol{y}_{u,v}) \log(1 - \hat{\boldsymbol{y}}_{u,v})) \quad (3.51)$$

definiert, wobei $\hat{\boldsymbol{y}}_{u,v}$ die vorhergesagte Wahrscheinlichkeit für das Vorhandensein einer Kante zwischen den Knoten u und v ist. Die tatsächliche Kennzeichnung $\boldsymbol{y}_{u,v}$ für das Vorhandensein einer Kante zwischen den Knoten u und v ist 1, wenn die Kante existiert ($e \in \mathcal{E}$), andernfalls 0.

Die Vorhersage auf Basis von Kanten-Merkmalen und -Embeddings ist im Gegensatz zur Verwendung von Knoten-Merkmalen und -Embeddings nicht direkt mit dem bisher besprochenen Message-Passing-Verfahren möglich. Die Generalisierung des Message-Passing-Verfahrens (vgl. Kapitel 3.2.2), womit Kanten- und auch Graph-Embeddings direkt gelernt werden können, wird in Kapitel 3.2.5 beschrieben. Für einen ausführlichen Überblick hierzu wird auf die Arbeit von [BHB+18] verwiesen.

Graph-Ebene

Auch die Graph-Ebene kann analog zur Knoten-Ebene betrachtet werden. Hierbei werden die resultierenden Knoten-Embeddings eines Graphs $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$ durch ein repräsentatives Embedding $\boldsymbol{z}_{\mathcal{G}}$ zusammengefasst, das durch die Anwendung von Pooling-Mechanismen (vgl. S. 51) erzeugt wird. Die Verlustfunktionen für die Klassifikation und die Regression auf Graph-Ebene sind dabei analog zu der für Knoten (vgl. S. 53). Im Folgenden wird die Klassifikation auf Graph-Ebene näher beschrieben, da diese für die vorliegende Arbeit von besonderer Bedeutung ist.

Graph-Klassifikation Unter Graph-Klassifikation ist die Vorhersage einer Kennzeichnung gemeint, die mit einem spezifischen Graphen $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{R})$ assoziiert ist. Gegeben ist für jeden Graphen ein Kennzeichen $\boldsymbol{y}_{\mathcal{G}}$, welches analog zur Knoten-Klassifikation, häufig als One-Hot-Kodierung abgebildet wird. Aus dem gelernten und abgeleiteten Embedding $\boldsymbol{z}_{\mathcal{G}} \in \mathbb{R}^d$ wird die Schätzgröße $\hat{\boldsymbol{y}}_{\mathcal{G}}$ ausgelesen. Dazu wird ebenfalls eine READOUT-Schicht benötigt, welche hinsichtlich der Klassifikation eine C-dimensionale Wahrscheinlichkeitsklassenverteilung ausgibt. Dabei kann konkret ein MLP für eine unnormalisierte Klassenverteilung oder eine SOFTMAX-Schicht für eine normalisierte Klassenverteilung oder ähnliche Mechanismen verwendet werden.

Analog zur Verlustfunktion für die Knoten-Klassifikation (vgl. Gl. (3.48)) kann der Verlust zwischen der Wahrheit $\boldsymbol{y}_{\mathcal{G}}$ und der Vorhersage $\hat{\boldsymbol{y}}_{\mathcal{G}}$ durch die *Kreuzentropie*-Verlustfunktion $\mathcal{L}_{\text{CE}}$ für den Graphen $\mathcal{G}$ und für die vorliegenden Klassen C durch

$$\mathcal{L}_{\text{CE},\mathcal{G}}(\boldsymbol{y}_{\mathcal{G}}, \hat{\boldsymbol{y}}_{\mathcal{G}}) = -\sum_{c=1}^{C} \boldsymbol{y}_{\mathcal{G},c} \log\left(\frac{\exp(\hat{\boldsymbol{y}}_{\mathcal{G},c})}{\sum_{i=1}^{C} \exp(\hat{\boldsymbol{y}}_{\mathcal{G},i})}\right) \tag{3.52}$$

bestimmt werden, wobei $\boldsymbol{y}_{\mathcal{G}} = (\boldsymbol{y}_{\mathcal{G},1}, \ldots, \boldsymbol{y}_{\mathcal{G},C})$ und in analoger Weise für $\hat{\boldsymbol{y}}_{\mathcal{G}}$ gilt. In der Arbeit [MRF+19] wird diese Art von Klassifikation auf der Graphen-Ebene, für die Kennzeichnung von Moleküleigenschaften verwendet.

3.2.5 Generalisiertes Message-Passing-Konzept

Die bisher besprochene Theorie zu Message-Passing und GNNs, bezieht sich hauptsächlich auf die Verwertung der Knoteneigenschaft von Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ und dem Lernen von Repräsentationen in Form von Knoten-Embeddings. Es gibt einzelne Erweiterungen für das Message-Passing-Verfahren, um über Knotenmerkmale hinaus explizit beim Lernen der Repräsentation auch die Informationen über die Kanten zu berücksichtigen. Dazu gehören Kantentypen und Relationen (vgl. Kapitel 3.2.3) oder aber eine Repräsentation als gewichteten Graphen (vgl. Kapitel 3.1). Dazu kann beispielsweise analog zu Gl. (3.24) die Nachricht mittels statischer Gewichte $\varepsilon_{u,v}$ aggregiert werden zu

$$\boldsymbol{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \varepsilon_{u,v} \boldsymbol{h}_v. \tag{3.53}$$

Diese Kantengewichte $\varepsilon_{u,v}$ können auf Basis einer vorliegenden Eigenschaft einer Kante e zwischen zweier Knoten (u, v) beschrieben werden und sind analog zu den Gewichten gewichteter Graphen (vgl. Gl. (3.2)).

Eine tatsächliche Berücksichtigung der Information über die Knoten-Ebene hinaus, kann durch eine generalisierte Betrachtung des Message-Passing-Algorithmus auf Ebene der Knoten, Kanten [GSR+17], [KFW+18] und des Graphs [BHB+18] realisiert werden. In [BHB+18] wird dies durch eine sukzessive Bildung von Embeddings durch mehrere UPDATE-Schritte und der Einführung von globalen Attributen, die dem Graph $\mathcal{G}$ zuzuschreiben sind, umgesetzt. Im Kern steht weiterhin die AGGREGATE- und UPDATE-Methode auf Knoten-Ebene. Eine generalisierte Message-Passing-Iteration ist wie folgt definiert (nach [Ham20] ursprünglich vorgeschlagen von [BHB+18]) durch

$$\boldsymbol{h}_{u,v}^{(k)} = \text{UPDATE}_{\text{edge}}(\boldsymbol{h}_{u,v}^{(k-1)}, \boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{h}_{\mathcal{G}}^{(k-1)}) \tag{3.54}$$

$$\boldsymbol{m}_{\mathcal{N}(u)} = \text{AGGREGATE}_{\text{node}}(\{\boldsymbol{h}_{u,v}^{(k)} \forall v \in \mathcal{N}(u)\}) \tag{3.55}$$

$$\boldsymbol{h}_u^{(k)} = \text{UPDATE}_{\text{node}}(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{m}_{\mathcal{N}(u)}, \boldsymbol{h}_{\mathcal{G}}^{(k-1)}) \tag{3.56}$$

$$\boldsymbol{h}_{\mathcal{G}}^{(k)} = \text{UPDATE}_{\text{graph}}(\boldsymbol{h}_{\mathcal{G}}^{(k-1)}, \{\boldsymbol{h}_u^{(k)} \forall u \in \mathcal{V}\}, \{\boldsymbol{h}_{u,v}^{(k)} \forall (u,v) \in \mathcal{E}\}). \tag{3.57}$$

Während einer Message-Passing-Iteration beschreibt demnach Gl. (3.54) die Kanten-Embeddings, Gl. (3.56) die Knoten-Embeddings und Gl. (3.57) das

Graph-Embedding. Zum einen ist somit keine weitere Methode notwendig, um ein für den Graphen repräsentatives Embedding abzuleiten, da dieses direkt gelernt wird. Zum anderen kann durch den direkten Zugriff auf die Embeddings für Knoten, Kanten und den Graph selbst, direkt eine Verlust- und Zielfunktion formuliert werden.

Im weiteren Verlauf der Arbeit wird sich jedoch auf das knotenbasierte Message-Passing beschränkt, da die meisten Arbeiten, die GNNs anwenden, auf das vorgestellte GCN (vgl. Gln. (3.21) und (3.22)) von [KW16a] zurückzuführen sind. Schon ab einer Anzahl von vier Knoten in einem Graphen, ist die Anzahl der Knoten deutlich geringer als die der möglichen Kanten und somit auch der Aufwand zum Lernen von knotenbasierten Embeddings, als der Aufwand zum Lernen kantenbasierter Embeddings.

Wie bereits in Kapitel 2.1 beschrieben, wird beispielsweise in der Arbeit von [DWA20] dieser erweiterte Ansatz, propagiert von [BHB+18], verwendet. Ein quantitativer Vergleich bezüglich der Ziel-Metrik zwischen den beiden Methoden wird indirekt durch die Arbeit von [LSS+23a] durchgeführt (vgl. Tabelle 6.7). Weitere Details zum erweiterten Message-Passing-Konzept kann der Arbeit von [BHB+18] entnommen werden, die diesen Sachverhalt explizit differenziert und erläutert.

3.2.6 Prototypische GNN-Architektur

In diesem Unterkapitel wird eine zusammenfassende Darstellung des Lernens von Repräsentationen in Form einer prototypischen GNN-Architektur zusammen mit den zentralen Begrifflichkeiten, die in Kapitel 3.2 vorgestellt wurden, präsentiert.

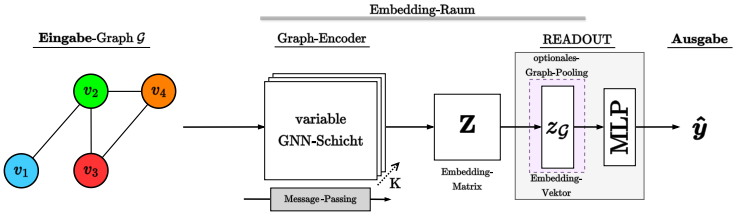


Abbildung 3.7: Prototypische GNN-Architektur mit Eingabe, Encoder und Ausgabe.

Eine typische GNN-Architektur zum Lernen von Gewichten für eine Vorhersageaufgabe kann wie in Abbildung 3.7 dargestellt aufgebaut werden. Initial wird eine Eingabe (engl. *input*) benötigt, welche als graphstrukturierte Daten dargestellt sind (vgl. dazu detailliert Kapitel 4.1). Die Knotenmerkmale eines Eingabe-Graphs $\mathcal{G}$ werden über den Encoder in den Embedding-Raum transformiert. Der Encoder, auch *Graph-Encoder*, besitzt eine *variable* Anzahl an GNN-Schichten, welche die Anzahl der Message-Passing-Iterationen K widerspiegeln. Nach K Message-Passing-Iterationen, enthält die Embedding-Matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ genau $|\mathcal{V}|$ Zeilen (Embeddingvektoren $\boldsymbol{z}_u, \forall u \in \mathcal{V}$) und d Spalten entsprechend der Dimension der Embeddingvektoren. Der Encoder kann weiter in einen Teil, welcher rein für die Transformation in den latenten Raum zuständig ist und in einen weiteren Teil, welcher zum tiefen Lernen verwendet wird, unterteilt werden. Je nach Lernaufgabe und der damit verbundenen Ebene des Graphen (vgl. S. 57), die für die Verlust- und Zielfunktion aufgestellt wird, wird optional eine Graph-Pooling Schicht verwendet, um die Embeddingvektoren $\boldsymbol{z}_u, \forall u \in \mathcal{V}$ in eine für den Graphen repräsentative Form zu aggregieren (vgl. S. 51). Der finale Baustein einer typischen GNN-Architektur ist die READOUT-Schicht (vgl. Gl. (3.46)), welche die Ausgabe (engl. *output*) $\hat{\boldsymbol{y}}$ in eine für die Verlust- und Zielfunktion passende Form transformiert. Die überwachten Lernaufgaben der Klassifikation und Regression sind für die Arbeit von besonderem Interesse, sodass die READOUT-Schicht durch die Anzahl der Klassen, dagegen im Falle der Regression durch die Dimension des Regressionsziels definiert wird.

Die prototypische GNN-Architektur wird in den Kapiteln 4 und 5 für die Methode und Anwendung sowie auch in Kapitel 6 für die Evaluation, die Grundlage bilden.

Zusammenfassung

Es wurde ein grundsätzliches Verständnis von GNNs geschaffen, wobei besonders auf Methoden und weiterführende Mechanismen für GNNs und den Message-Passing-Algorithmus eingegangen wurde, welche sich primär auf die Bildung von knotenbasierten Embeddings beziehen. Motiviert ist diese Wahl durch die besonders einfache und ressourcenarme Formulierung und Anwendung. Bei einer Berücksichtigung bzw. Erweiterung des Message-Passing-Algorithmus um Kanten-Embeddings, die explizit gelernt werden, steigt die

Komplexität in der Ausführung einer Message-Passing-Iteration, da die Anzahl der zu berücksichtigenden Kanten unabhängig von der Anzahl der Knoten steigt.

Abschließend wurde eine prototypische GNN-Architektur vorgestellt, die als Grundlage für die entwickelte Methode und die damit verbundenen Architekturen dient.

4 Aktions- und Bewegungsvorhersage für die Mensch-Roboter-Kollaboration

In den bisherigen Kapiteln wurden der Stand der Technik und die notwendigen Grundlagen für die vorliegende Anwendung zur Aktions- und Bewegungsvorhersage für die MRK erarbeitet. In diesem Kapitel wird auf Basis dieser Grundlagen eine Umsetzung der Aktions- und Bewegungsvorhersage für die MRK mit GNNs vorgestellt. Dabei werden die Entwicklung einer Szenenrepräsentation (vgl. Kapitel 2.3) als Modellierung eines *Szenengraphen* (auch: *Graphenkonstruktion*) erläutert und anschließend die verschiedenen Aufgaben hergeleitet, die mit Hilfe dieser Szenengraphen und der vorgestellten Methode gelöst werden können.

Dieses Kapitel stützt sich teilweise auf im Rahmen dieser Arbeit frühere enstandene Veröffentlichungen [LSS+23a] und [LSS+23b], in denen Methoden zur Aktionserkennung, Aktionsprädiktion und Bewegungsvorhersage einschließlich der grundlegenden Graphenkonstruktion beschrieben sind.

4.1 Modellierung des Szenengraphen

Im vorliegenden Anwendungsfall wird die Szene durch eine Stereokamera wahrgenommen (vgl. Kapitel 2.2). Die aufgezeichneten Daten liegen dementsprechend in Form von RGB- und Tiefeninformation sowie als Merkmale, die aus diesen bereits extrahiert wurden, vor. Zu diesen Merkmalen gehören die in einem Bild (Frame) vorhandenen Objekte inklusive der detektierten Objektklasse und der lokalisierten 3D-Position $x^t \in \mathbb{R}^3$ zum Zeitpunkt t im Koordinatensystem des Roboters (Szene veranschaulicht in Abbildung 4.1). Zu den Objekten gehört auch die Information zu den menschlichen Akteuren in der Szene, welche ebenfalls detektiert und lokalisiert werden.

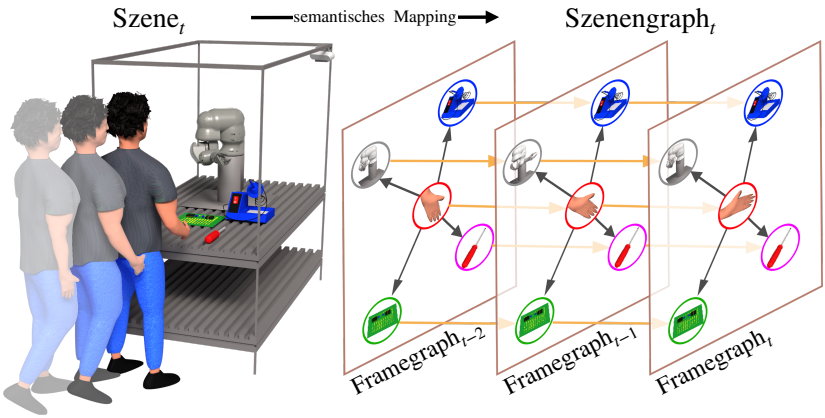


Abbildung 4.1: Veranschaulichung der Szene und der daraus abgeleiteten Szenenrepräsentation, welche durch einen räumlich-zeitlichen Szenengraphen dargestellt wird. Ein Szenengraph besteht dabei aus mehreren konsekutiv zusammenhängenden Framegraphen, welche der extrahierten Information eines Bildes aus einer abgetasteten Videosequenz der Szene entsprechen. Außerdem sind Verbindungen zwischen Knoten, innerhalb eines Framegraphen und über den Framegraphen hinaus dargestellt. Die Auswahl der abzubildenden Information wird in diesem Kapitel näher erläutert.

Um mittels GNNs bzw. einer GNN-Architektur eine Lernaufgabe für die Aktions- und Bewegungsvorhersage zu formulieren, wird eine graphstrukturierte Repräsentation der Szene benötigt, wie in Kapitel 3.2.6 dargestellt. Die Aufgabenstellung zur Konstruktion eines Graphen aus den gegeben Informationen scheint trivial, bedarf jedoch einer besonderen Aufmerksamkeit, da durch *implizite* oder *explizite* Abbildung von Information in der Graphstruktur bereits Entscheidungen getroffen werden, die fundamentalen Einfluss auf den Lernverlauf mit GNNs haben und die Modellgüte stark beeinflussen können. Im Folgenden wird die in [LSS+23a], [LSS+23b] vorgeschlagene Graphkonstruktion vorgestellt und hinsichtlich der möglichen Entscheidungen in dieser diskutiert.

Szenengraph Im Kontext dieser Arbeit ist ein Szenengraph $\mathcal{G}_S^t = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ ein Graph, der die abgeleitete räumliche Information einer Szene in Form von

Framegraphen $\mathcal{G}_F^t$ (vgl. Abbildung 4.2) zum Zeitpunkt t für $H + 1$ Zeitschritte in einer graphstrukturierten Form zeitlich darstellt (vgl. [AAD+11], [ATW15] und [DWA20] für ähnliche Definitionen von Szenengraphen). Dabei ist H die sogenannte *history size* und stellt die zeitlichen Schritte $t = 0 \dots H$ in die Vergangenheit und somit die zeitlich zusammenhängenden Framegraphen $\mathcal{G}_F^t$ eines Szenengraphen $\mathcal{G}_S^t$ dar. Die Menge an Knoten $\mathcal{V} = \{\mathcal{V}^t\}_{t=0\dots H}$ eines Szenengraphen $\mathcal{G}_S^t$ besteht aus $H + 1$ Teilmengen $\mathcal{V}^t$, welche die einzelnen Knoten eines Framegraphen $\mathcal{G}_F^t$ zum Zeitpunkt t darstellen. Dabei repräsentiert die Knotenmenge $\mathcal{V}^t = \{v_i\}_{i=1\dots N}$ mit v_i als Knoten eines Framegraphen das i-te von N Objekten in einem Bild bzw. der Szene zum Zeitpunkt t. Jeder Knoten v_i verfügt über Knoten-Merkmale $\boldsymbol{x}_i \in \mathbb{R}^d$, wobei die Dimension d durch die One-Hot-Kodierung der jeweiligen Objektklasse ($d_{\text{onehot}} \in \mathbb{R}^{\#\text{Objekte}}$) und der 3D-Position P_i^t des jeweiligen Objektes ($d_{\text{pos}} \in \mathbb{R}^3$) gegeben ist.

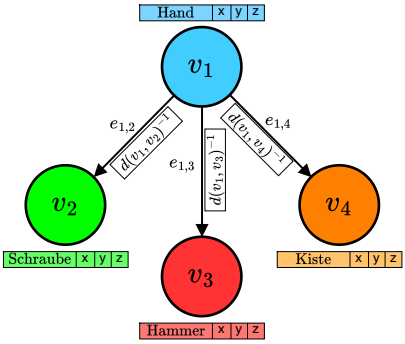


Abbildung 4.2: Exemplarischer Aufbau eines Framegraphen $\mathcal{G}_F^t$ zum Zeitpunkt t mit den Objekten einer Szene als Knoten $\mathcal{V}^t = \{v_i\}_{i=1\dots N}$, den gerichteten Kanten $\mathcal{E}$ zwischen dem *Hand*-Knoten v_1 und den übrigen Objkten $\{v_j\}_{j=2\dots N}$ (N entspricht der Anzahl an Objekten) sowie den Knoten- und Kanten-Merkmalen. Dieser Graph wird auch *Hand-2-Object* (H2O) genannt.

Die Kanten $\mathcal{E}$ eines Szenengraphen $\mathcal{G}_S^t$ bestehen aus $\mathcal{E} = \mathcal{E}_{intra} \cup \mathcal{E}_{inter}$. Dabei sind $\mathcal{E}_{intra} = \{(v_1^t, v_j^t)\}_{j=2\dots N}$ zu einem Zeitpunkt t die sogenannten *intra*-frame Kanten, welche den Knoten v_1, welcher die Hand repräsentiert, mit den Knoten v_j, welche die weiteren Objekte der Szene repräsentieren, verbindet. Der Graph, der aus dieser gerichteten Repräsentation der *intra*-frame Kanten entsteht, wird auch *Hand-2-Object* (H2O) genannt. Diese Repräsentation ist

im Framegraphen $\mathcal{G}_F^t$ in Abbildung 4.2 dargestellt. Die *inter*-frame Kanten $\mathcal{E}_{inter} = \{(v_i^{t-1}, v_i^t)\}_{t=1\dots H, i=1\dots N}$ verbinden die Knoten v_i bzw. Objekte der gleichen Klasseninstanz in zeitlich aufeinanderfolgenden Frames, um die Trajektorie eines Knotens v_i zeitlich zu modellieren. Die Kanten-Merkmale werden als Kantengewichte $\varepsilon_{i,j}^t$ (vgl. Gl. (3.53)) modelliert, welche den euklidischen Abstand $d^t(P_i^t, P_j^t)$ (kurz: $d_{i,j}^t$) der 3D-Position von zwei durch eine Kante $e \in \mathcal{E}$ verbundene Knoten $i, j \in \mathcal{V}$ zum Zeitpunkt t im euklidischen Raum darstellen.

Um die Kanten-Merkmale als Gewichtung für die Beziehung zweier Knoten abzubilden, wird die reziproke euklidische Distanz

$$\varepsilon_{i,j}^t = \frac{1}{d_{i,j}^t} \tag{4.1}$$

verwendet. Dadurch kann für räumlich nähere Objekte ein höheres Gewicht und umgekehrt für entferntere Objekte beschrieben werden.

Neben der Möglichkeit, Kanten-Merkmale $\varepsilon_{i,j}^t \in \mathbb{R}$ durch gewichtete Abstände zwischen Knoten zu beschreiben, kann die Information über $intra$- und $inter$-frame Kanten auch explizit als Relation $\mathcal{R}$ beschrieben werden, wobei diese Relationen $\mathcal{R} \in \{\textit{räumlich}, \textit{zeitlich}\}$ durch die Natur eines Szenengraphen $\mathcal{G}_S^t$ (vgl. Abbildung 4.3), welcher räumlich-zeitliche Relationen abbildet, für jede Kante eingeteilt werden können.

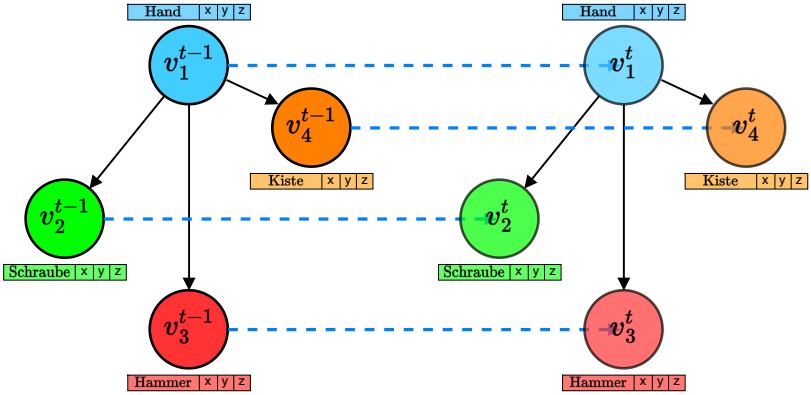


Abbildung 4.3: Exemplarischer Aufbau eines Szenengraphen $\mathcal{G}_S^t$ zum Zeitpunkt t. Dabei stellen die blauen, gestrichelten Pfeile die $inter$-frame und die schwarzen Pfeile die $intra$-frame Katen dar. Des Weiteren besteht dieser Szenengraph $\mathcal{G}_S^t$ aus den beiden Framegraphen $\mathcal{G}_F^t$ und $\mathcal{G}_F^{t-1}$.

Die Verwendung des Kantenmerkmals als skalares Gewicht zur Abbildung einer Distanz oder in vektorieller Form für eine Relation $\mathcal{R}$ hängt von der gewählten GNN-Variante ab. Es gibt GNN-Varianten, welche skalare Gewichte in der Adjazenzmatrix $\mathbf{A}$ integrieren können (vgl. Gl. (3.53)) oder aber welche, die direkt in der Nachbarschafts-Aggregation eine Differenzierung über die definierten Relationen $\mathcal{R}$ (vgl. Kapitel 3.2.3) vornehmen. Es kann im Vorfeld keine Aussage darüber getroffen werden, welche GNN-Variante gewählt werden muss und ob skalare oder vektorielle Kanten-Merkmale genutzt werden müssen. Eine systematische Annäherung an eine optimale Entscheidung bezüglich der zu wählenden GNN-Variante und der Verwendung von Kanten-Merkmalen und weiteren Parametern wird im Kontext der Hyperparameteroptimierung in Kapitel 6.1 beschrieben.

Neben der vorgestellten H2O-Modellierung des Graphen kann dieser auch als vollständiger Graph[1] modelliert werden. Alle Knoten, also sämtliche Objekte, werden dabei durch Kanten miteinander verknüpft (auch *Object-2-Object*

[1] Vollständiger Graph: Ein Graph $\mathcal{G}$ bei dem alle Knoten durch Kanten paarweise verknüpft sind [Die17].

(O2O)). Ein solcher O2O-Graph ist in Abbildung 4.4 dargestellt. Die Anzahl der Knoten bleibt dabei gleich, die Anzahl der Kanten nimmt jedoch zu. Diese Zunahme wird durch die Anzahl an $H + 1$ zu berücksichtigenden historischen Framegraphen $\mathcal{G}_F^t \dots \mathcal{G}_F^{t-H}$ weiter verstärkt. In der Arbeit [LSS+23b] wurde diese Komplexitätszunahme für einen beispielhaften Datensatz [LSSD22] quantitativ betrachtet.

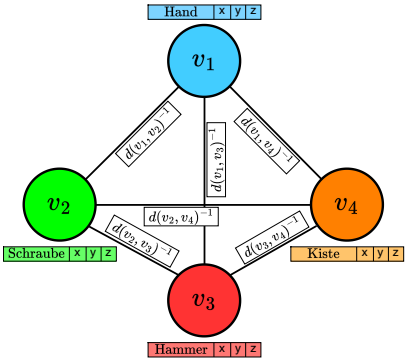


Abbildung 4.4: Exemplarischer Aufbau eines Framegraphen $\mathcal{G}_F^t$ zum Zeitpunkt t mit einer Verknüpfung aller Knoten, also sämtlicher Objekte durch Kanten. Diese Modellierung wird in dieser Arbeit auch als O2O-Graph bezeichnet und entspricht einem vollständigen Graphen.

Die konstruierten Szenengraphen dienen als Eingabedaten für die nachfolgend beschriebenen Modellarchitekturen.

4.2 Beschreibung der Modellarchitektur

Nachdem nun die Graphkonstruktion bzw. die Modellierung des Szenengraphen erläutert wurde, können analog zur eingeführten prototypischen GNN-Architektur (vgl. Kapitel 3.2.6) die einzelnen Komponenten und damit die schematische Modellarchitektur (vgl. Abbildung 4.5) vorgestellt werden. Anschließend werden die Zielgrößen für die jeweiligen Lernaufgaben anwendungsbezogen beschrieben.

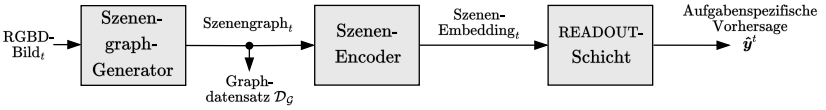


Abbildung 4.5: Schematischer Aufbau der GNN-Modellarchitektur: Der *Szenengraph-Generator* verarbeitet ein RGBD-Bild$_t$ zu einem Szenengraphen $\mathcal{G}_S^t$. Der sogenannte *Szenen-Encoder* überführt den Szenengraphen in eine latente Raumdarstellung (engl. *latent space representation*) und führt das Message-Passing durch. Im Anschluss an den Szenen-Encoder wird durch die READOUT-Schicht eine Vorhersage $\hat{\boldsymbol{y}}^t$ für die spezifische Lernaufgabe (vgl. Kapitel 4.2.3, 4.2.4 und 4.2.5) passend abgeleitet. Der erste Teil bis zum Erhalt des Szenengraphen $\mathcal{G}_S^t$ kann zur Erstellung eines graphstrukturierten Datensatzes $\mathcal{D}_\mathcal{G}$ verwendet werden.

4.2.1 Szenengraph-Generator

Die Aufgabe der Generierung von Szenengraphen ist wichtig, um graphstrukturierte Daten zu erzeugen, die sowohl für den Lernprozess in Form eines Graphdatensatzes $\mathcal{D}_\mathcal{G}$, als auch für die Inferenz verwendet werden können. Der *Szenengraph-Generator* kann durch die in Abbildung 4.6 dargestellte Pipeline beschrieben werden.

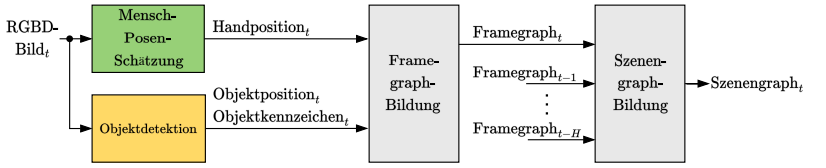


Abbildung 4.6: Beschreibung der Pipeline zur Generierung von Szenengraphen aus vorhandener RGB und Tiefeninformation. Die *grün* und *gelb* eingefärbten Komponenten deuten auf zusätzlich benötigte Komponenten während der Inferenz hin. Die Menschen-Posen-Schätzung in *grün* wird dabei durch vortrainierte Modelle realisiert (vgl. OpenPose [SJMS17], [CHS+21] und MediaPipe [LTN+19], [ZBV+20] bzw. in Kapitel 2.2.1). Die in *gelb* eingefärbte Komponente hingegen muss durch einen spezifisch auf die Szenenobjekte trainierten Objektdetektor realisiert werden, da vor allem im Industrie-Kontext spezielle Werkzeuge, Bauteile oder Werkstücke für Montageaufgaben Verwendung finden.

4.2.2 Szenen-Encoder

Der sogenannte *Szenen-Encoder* setzt die erläuterte Theorie zum Message-Passing und den GNNs um. Das Ziel ist die Überführung eines Szenengraphen $\mathcal{G}_S^t$ in eine latente Raumdarstellung (auch *Embedding-Raum* genannt). Wie in Abbildung 4.7 dargestellt, wird dies initial durch einen GNN-Block erreicht, der für die initiale Transformation der Merkmale $\mathbf{X}_\mathcal{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$ in den Ziel-Embedding-Raum $\mathbb{R}^C$, durch die Abbildung $f : \mathbb{R}^d \rightarrow \mathbb{R}^C$, zuständig ist. Eine solche Transformation f kann dabei auf unterschiedliche Arten umgesetzt werden. Obwohl in diesem Fall ein GNN-Block verwendet wird, wäre eine gewöhnliche lineare Schicht ebenfalls für die initiale Transformation in den latenten Raum ausreichend. Insgesamt besteht der Szenen-Encoder aus einer Sequenz von m aufeinanderfolgenden GNN-Blöcken, wobei bis auf den initialen Block zur Transformation in den Embedding-Raum der residuale Mechanismus [HZRS16a] in den GNN-Blöcken integriert wird. Der genaue Aufbau eines (residualen) GNN-Blocks ist in Abbildung 4.8 dargestellt.

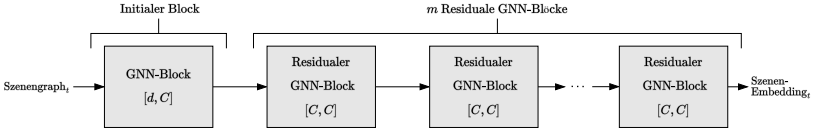


Abbildung 4.7: Der Szenen-Encoder verarbeitet einen Szenengraphen $\mathcal{G}_S^t$, um reichhaltige Embeddings der Szene zu erlernen. Dazu wird der Eingangs-Szenengraph $\mathcal{G}_S^t$ durch m aufeinanderfolgende, residuale GNN-Blöcke verarbeitet, wobei initial ein Block zur Transformation der Szenengraph-Merkmale $\mathbf{X}_\mathcal{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$ in den Ziel-Embedding-Raum $\mathbb{R}^C$ verwendet wird.

Nachfolgend wird der detaillierte Aufbau eines GNN-Blocks erläutert, wobei auf besondere Modellierungseigenschaften eingegangen wird. Es ist wichtig anzumerken, dass ein GNN-Block lediglich zur Beschreibung möglicher, aufeinanderfolgender Schichten innerhalb eines solchen Blocks dient, einschließlich der verwendeten Mechanismen wie beispielsweise der Skip-Verbindungen (vgl. Kapitel 3.2.3).

Generischer GNN-Block Der generische *GNN-Block* stellt den Kern des Szenen-Encoders dar und ist so aufgebaut, dass verschiedene Mechanismen,

bekannt aus dem tiefen maschinellen Lernen, abgebildet werden können, um graphspezifische Merkmale zu verarbeiten und Embeddings aus diesen zu lernen.

Ein GNN-Block verarbeitet die Knoten-Merkmale $\mathbf{X}_{\mathcal{V}}^{(0)} \in \mathbb{R}^{|\mathcal{V}| \times d}$ bzw. Knoten-Embeddings $\mathbf{H}_{\mathcal{V}}^{(k-1)} \in \mathbb{R}^{|\mathcal{V}| \times d}$ für $k > 0$ zu Knoten-Embeddings $\mathbf{H}^{(k)} \in \mathbb{R}^{|\mathcal{V}| \times d}$. Alternativ können statt den Kanten-Gewichten die Kanten-Typen (vgl. Kapitel 3.1 und 3.2.3) oder aber lediglich die Knoten-Merkmale ohne weitere kantenspezifische Informationen, verwendet werden.

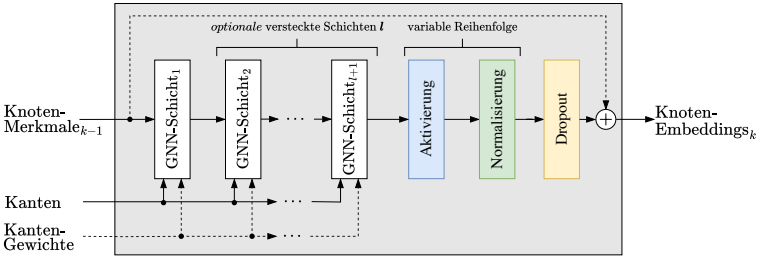


Abbildung 4.8: Aufbau des generischen GNN-Blocks: Als Eingang werden Knoten-Merkmale oder Knoten-Embeddings des vorherigen Message-Passing-Schrittes $k-1$ mit der Information über vorhandene Kanten und der zugehörigen Kanten-Merkmale zu Knoten-Embeddings verarbeitet. Der Aufbau besteht aus l optionalen, versteckten GNN-Schichten, wobei standardmäßig $l = 0$ ist, und einer Folge von Aktivierungs-, Normalisierungs-, und Dropoutschicht. Optional kann der Eingang auf den Ausgang der Dropout-Schicht addiert (Residualschicht) werden. Optionale Schritte sind durch *gestrichelte* Pfeile gekennzeichnet.

Der generische GNN-Block (Abbildung 4.8) besteht grundsätzlich aus mindestens einer GNN-Schicht und l weiteren, optionalen GNN-Schichten. Welches GNN explizit verwendet wird, wird durch die jeweilige GNN-Variante (vgl. 3.2.3) spezifiziert. Die Anzahl der optionalen GNN-Schichten ist standardmäßig $l = 0$, sodass nur eine GNN-Schicht in einem GNN-Block verwendet wird. Die Option, einen GNN-Block auf Basis mehrerer GNN-Schichten aufzubauen, eröffnet die zusätzliche Möglichkeit, ein tieferes neuronales Netz und somit Modell zu erstellen.

Auf die letzte GNN-Schicht folgt eine Aktivierungs-, Normalisierungs- und Dropout-Schicht, wobei die Reihenfolge der Aktivierungs- und Normalisie-

rungsschicht variabel ist. In der Arbeit von [IS15] zu Normalisierungsschichten (hier: *Batch-Normalisierung*), wird die Reihenfolge

$$\cdots \rightarrow \text{Normalisierung} \rightarrow \text{Aktivierung} \rightarrow \ldots$$

vorgeschlagen, in der die Normalisierung der Aktivierungsschicht folgt, um beispielsweise den Ausgang einer Faltungsschicht für die Aktivierung sinnvoll zu begrenzen. Diese Reihenfolge wurde in [IS15] etabliert. Empirisch hat sich jedoch gezeigt, dass eine Aktivierung vor der Normalisierungsschicht zu besseren Ergebnissen führen kann, daher wird die Reihenfolge im vorgeschlagen GNN-Block variabel modelliert. Die schlussendliche Verwendung in der Architektur muss auch hier durch eine Hyperparameteroptimierung (vgl. Kapitel 6.1) bestimmt werden.

Zu den abgebildeten Mechanismen gehört ebenfalls die residuale Addition [HZRS16a] (auch *Residual Layer*), welche die Eingabewerte zu den Ausgabewerten der Dropout-Schicht addiert (vereinfacht durch „Residual“ dargestellt):

$$\text{GNN} \rightarrow \text{Normalisierung} \rightarrow \text{Aktivierung} \rightarrow \text{Dropout} \rightarrow \text{Residual}.$$

Des Weiteren kann der generische GNN-Block dahingehend erweitert werden, die in [LXTG20] und ursprünglich in [HZRS16b] vorgestellte Reihenfolge umzusetzen. Diese Reihenfolge

$$\text{Normalisierung} \rightarrow \text{Aktivierung} \rightarrow \text{Dropout} \rightarrow \text{GNN} \rightarrow \text{Residual}$$

setzt die eigentliche GNN-Schicht direkt vor die Residualschicht und *hinter* die Aktivierungsschicht. Somit wird die Aktivierung vor der Verarbeitung durch das GNN durchgeführt, was auch als *pre-activation* bekannt ist (vgl. [HZRS16b]). In [LXTG20] wird diese Reihenfolge im Zusammenhang mit der GNN-Variante von [KW16a] als „ResGCN+“ vorgestellt. Dabei kennzeichnet der Zusatz des „+“-Zeichens diese spezielle Reihenfolge von Aktivierungs-, Normalisierungs-, Dropout- und GNN-Schicht unter Verwendung einer Residualschicht.

Entsprechend dem hier beschriebenen generischen GNN-Block werden in der Arbeit von [LXTG20] vergleichbare Modellierungseigenschaften für GNN-Blöcke vorgestellt und um weitere ergänzt. Eine Ausnahme bildet dabei die Möglichkeit die GNN-Schichten innerhalb eines GNN-Blocks tief zu schichten.

Für eine weiterführende Auseinandersetzung mit GNN-Blöcken als Bausteine für tiefergehende GNN-Architekturen wird auf [LXTG20] verwiesen.

Aktivierung Mögliche Aktivierungsfunktionen sind beispielsweise das ReLU, das LeakyReLU, das *Parametric Rectified Linear Unit* (PReLU) [HZRS15] oder das *Exponential Linear Unit* (ELU) [CUH16]. Grundsätzlich wird in jedem GNN-Block eine Aktivierungsfunktion verwendet, wobei die Auswahl einer bestimmten Funktion durch eine Hyperparameteroptimierung (vgl. Kapitel 6.1) empirisch bestimmt wird.

Die verschiedenen Aktivierungsfunktionen unterscheiden sich in der Regel im Umgang mit „negativen“ x-Werten (auch: Eingabewerten). Die ReLU-Funktion behandelt diese Eingabewerte alle gleich und aktiviert die Neuronen mit dem Funktionswert von 0 (*inaktiv*). Dadurch kann es vorkommen, dass für bestimmte x-Werte (die negativ sind), der Gradient auf den Funktionswert 0 gesetzt wird und die Neuronen somit inaktiv sind (auch bekannt als *dying ReLU problem*). Dieses Problem wird durch die LeakyReLU-, PReLU- oder ELU-Funktion auf unterschiedliche Arten angegangen. Die LeakyReLU-Funktion führt einen konstanten Wert $\alpha \neq 0$ ein, der für negative x-Werte den Gradienten mit α aktiviert, wobei α ein statischer Hyperparameter ist. Die PReLU-Funktion parametrisiert diesen konstanten Wert α als lernbaren Parameter im Training selbst. Die ELU-Funktion basiert auf der Exponentialfunktion e^x, welche den Gradient als Exponential-Wert für negative x-Werte aktiviert. Wie bei der LeakyReLU-Funktion wird ein statischer Hyperparameter α als konstanter Wert für negative x-Werte verwendet.

Normalisierung Die Normalisierung kann ergänzend in einer Architektur verwendet werden, um den Trainingsprozess zu stabilisieren und zu beschleunigen [IS15], [BKH16]. Die Batch-Normalisierung [IS15] operiert auf Basis von Batches, also auf Bündeln an Trainingsdaten, und verfolgt das Ziel durch die statistische Information des Batches (Mittelwert und Varianz) eine Normalisierung durchzuführen, die sich positiv auf die Robustheit im Lernen auswirkt und eine größere Lernrate zulässt. Diese Strategie ist abhängig von der Batchgröße und wird nur im Training verwendet und nicht während der Inferenz, da üblicherweise eine Batch-Strategie für die Inferenz mit unbekannten Daten nicht möglich ist. Stattdessen wird die während der Trainingsphase erfahrene

Statistik für die Inferenz verwendet. Die Layer-Normalisierung [BKH16] ist unabhängig von den Batches und operiert auf den statistischen Informationen einer jeden Schicht. Hierbei muss keine Unterscheidung zwischen der Trainings- und der Inferenzphase vorgenommen werden.

Grundsätzlich kann bei Limitierungen in der Verwendung von Batches eine Batch-Normalisierungstrategie ausgeschlossen werden. Wenn jedoch beide Normalisierungsstrategien für den Anwendungsfall umsetzbar sind, sollte analog zur Auswahl der Aktivierungsfunktion die Auswahl der Normalisierungsstrategie empirisch durch eine Hyperparameteroptimierung (vgl. Kapitel 6.1) getroffen werden.

Dropout Mit *Dropout* [SHK+14] ist ein Mechanismus gemeint, welcher ähnlich wie eine Normalisierungsstrategie dabei hilft, Robustheit in den Lernprozess zu integrieren. Die Verarbeitung des Eingangs einer Dropoutschicht wird unter Ausschluss einer zufällig festgelegten deaktivierten Menge an Neuronen, durchgeführt, um so das Lernen robuster zu machen und eine Überanpassung zu verhindern. Analog zur Batch-Normalisierung wird eine Unterscheidung zwischen Trainingsphase und Inferenz gemacht. Während der Trainingsphase wird die Dropout-Funktion verwendet und während der Inferenz deaktiviert.

Szenen-Embedding Das Szenen-Embedding $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ entspricht der vorgestellten Embedding-Matrix (Gl. (3.5)), die über die einzelnen finalen Knoten-Embeddings der Szene S, gegeben dem verarbeiteten Szenengraphen $\mathcal{G}_S^t = (\mathcal{V}, \mathcal{E})$, nach der Durchführung des Message-Passings verfügt. Das Szenen-Embedding $\mathbf{Z}_S^t$ kann als reiches Merkmal für unterschiedliche Lernaufgaben, die in den folgenden Kapiteln 4.2.3, 4.2.4 und 4.2.5 näher beschrieben werden, genutzt werden.

4.2.3 Aktionsklassifikation

Die Aktionsklassifikation stellt eine Lernaufgabe zur Ermöglichung intelligenter MRK dar und kann dabei abhängig von der wahren Zielgröße entsprechend der Klassifikation näher spezifiziert werden. So kann etwa die Klassifikation der Ak-

tion durch die Vorhersage einer Aktionsklassenverteilung für A Aktionsklassen durchgeführt werden.

Die untersuchten Varianten für eine solche Aktionsklassifikation im Kontext dieser Arbeit können wie folgt formuliert werden:

- *Erkennung* der aktuellen Aktion a^t, gegeben des Szenengraphen $\mathcal{G}_S^t$ als Eingabe und der vorhergesagten Aktionsklassenverteilung $\hat{\boldsymbol{y}}^t$ als Ausgabe.
- *Vorhersage* der Aktion a^{t+P} zum *statischen* Prädiktionszeitpunkt P, gegeben des Szenengraphen $\mathcal{G}_S^t$ als Eingabe und der vorhergesagten Aktionsklassenverteilung $\hat{\boldsymbol{y}}^{t+P}$ als Ausgabe.
- *Vorhersage* der *nächsten* Aktion $a^{t+\tau}$ zum *dynamischen* Prädiktionszeitpunkt τ, wann die nächste Aktion stattfindet, gegeben des Szenengraphen $\mathcal{G}_S^t$ als Eingabe und der vorhergesagten Aktionsklassenverteilung $\hat{\boldsymbol{y}}^{t+\tau}$ mit $\tau \in \mathbb{N}_0$ als Ausgabe.

Dabei entspricht der statische Prädiktionszeitpunkt P dem letzten Zeitpunkt des Prädiktionshorizonts T_P. Der Prädiktionshorizont ist somit definiert als $T_P := t + t_P,$ wobei $t_P \in \{1, \ldots, P\}$ und im Folgenden $P = H + 1$.

Die Abbildung 4.9 veranschaulicht für diese Varianten den Aufbau einer möglichen READOUT-Schicht, welche auch bereits in der GNN-Modellarchitektur (vgl. Abbildung 4.5) verwendet wird.

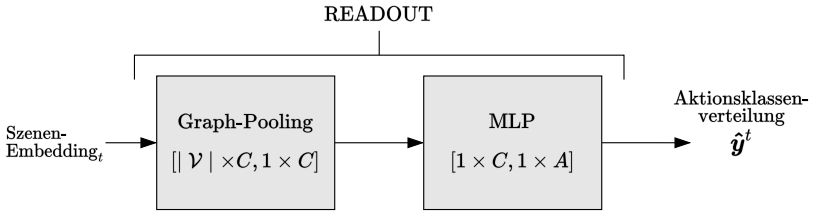


Abbildung 4.9: Schematischer Aufbau der READOUT-Schicht für die Aktionsklassifikation. Das Szenen-Embedding $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ wird durch ein Graph-Pooling auf eine Repräsentation auf Graphenebene $\boldsymbol{z}_{\mathcal{G}}^t$ transformiert (vgl. Gln. (3.43)–(3.45)). Durch ein MLP wird diese Repräsentation auf Graphenebene $\boldsymbol{z}_{\mathcal{G}}^t \in \mathbb{R}^C$ folglich in eine Aktionsklassenverteilung $\hat{\boldsymbol{y}}^t \in \mathbb{R}^A$ überführt.

Nachfolgend werden diese drei angeführten Varianten für eine Aktionsklassifikation näher beschrieben.

Aktionserkennung Bei der Aktionserkennung handelt es sich um die *Erkennung* der aktuellen Aktion a^t. Als Beobachtung ist der Verlauf der Szene für $H + 1$ historische Zeitschritte als Szenengraph $\mathcal{G}_S^t$ gegeben. Nach dem Erlernen des Szenenembeddings $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ wird dieses durch Graph-Pooling als eine Repräsentation auf Graphenebene $\boldsymbol{z}_\mathcal{G}^t \in \mathbb{R}^C$ abgebildet. Durch eine READOUT-Schicht kann folglich die Aktionsklassenverteilung $\hat{\boldsymbol{y}}^t \in \mathbb{R}^A$ vorhergesagt werden. Dabei steht jedes Neuron für die unnormalisierte Wahrscheinlichkeit der jeweiligen Aktionsklasse $a \in A$.

Die unnormalisierte Aktionsklassenverteilung $\hat{\boldsymbol{y}}^t$ kann dann durch eine Verlustfunktion verwendet werden, um die Prädiktion mit der Wahrheit abzugleichen. Somit kann beispielsweise die Kreuzentropie-Verlustfunktion, analog zur Gl. (3.52), als $\mathcal{L}_{\text{CE},\mathcal{G}}(\boldsymbol{y}^t, \hat{\boldsymbol{y}}^t)$ formuliert werden. Dabei stellt $\boldsymbol{y}^t$ die wahre Aktion als One-Hot-Kodierung dar.

Aktionsprädiktion mit statischem Prädiktionszeitpunkt Ähnlich zur Erkennung der Aktion, beschreibt die *Vorhersage* der Aktion a^{t+P} die Klassifikation der zukünftigen Aktion. Dabei wird jedoch die Aktion zum *statischen* Prädiktionszeitpunkt P vorhergesagt, gegeben des Szenengraphens $\mathcal{G}_S^t$ analog zur Aktionserkennung als Eingabe. Nach dem Erlernen des Szenenembeddings $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ wird dieses durch Graph-Pooling als eine Repräsentation auf Graphenebene $\boldsymbol{z}_\mathcal{G}^t \in \mathbb{R}^C$ abgebildet und durch eine READOUT-Schicht als eine Aktionsklassenverteilung $\hat{\boldsymbol{y}}^{t+P} \in \mathbb{R}^A$ dargestellt.

Das bedeutet, dass für die Vorhersage der Aktion a^{t+P} die Basis durch die gleiche Beobachtung gegeben ist und keine weitere Information zur Verfügung steht, als für die Aktionserkennung. Die Kreuzentropie-Verlustfunktion gilt hier analog zur Gl. (3.52) als $\mathcal{L}_{\text{CE},\mathcal{G}}(\boldsymbol{y}^{t+P}, \hat{\boldsymbol{y}}^{t+P})$. Dabei stellt $\boldsymbol{y}^{t+P}$ die wahre Aktion als One-Hot-Kodierung zum Zeitpunkt $t + P$ dar.

Aktionsprädiktion mit dynamischem Prädiktionszeitpunkt Die Vorhersage der Aktion zu einem statischen Prädiktionszeitpunkt P bedeutet, dass keine Rücksicht auf die eigentliche Folge bzw. den Wechsel von Aktionen während

einer gesamten Videosequenz, in der eine Aufgabe bearbeitet wird, genommen werden kann. Daher ist eine weitere untersuchbare Aufgabe die Vorhersage der *nächsten* Aktion, die durch einen dynamischen Prädiktionszeitpunkt τ gegeben ist. Dies ist der Fall, wenn ein Aktionswechsel stattfindet. Für die Kreuzentropie-Verlustfunktion gilt analog $\mathcal{L}_{\text{CE},\mathcal{G}}(\boldsymbol{y}^{t+\tau}, \hat{\boldsymbol{y}}^{t+\tau})$. Dabei stellt $\boldsymbol{y}^{t+\tau}$ die wahre Aktion als One-Hot-Kodierung zum Zeitpunkt $t + \tau$ dar.

Besonders interessant ist der dynamische Prädiktionszeitpunkt auch im Hinblick auf einen Vergleich mit dem statischen Prädiktionszeitpunkt. So kann durch den dynamischen Prädiktionszeitpunkt τ für statische Prädiktionszeitpunkte P, ein Vergleich geführt werden, falls gilt $\tau > P$. Dies wird im Evaluationskapitel 6 näher beschrieben.

4.2.4 Bewegungsvorhersage

Die Bewegungsvorhersage stellt eine weitere Lernaufgabe für die Umsetzung einer intelligenten MRK dar. Das Ziel entspricht dem Lernen der Merkmale auf Knotenebene, sowohl für einen als auch für mehrere Knoten eines Szenengraphen $\mathcal{G}_S^t$, über eine Regression zur Vorhersage der zukünftigen Position. Genauer gesagt soll der Teil eines Knoten-Merkmals $\boldsymbol{x}_u \in \mathbb{R}^d$ eines Knoten $u \in \mathcal{V}$, der die 3D-Position des jeweiligen Knoten und somit Objektes abbildet, durch eine Knotenregression gelernt werden. Je nach Spezifikation des Prädiktionshorizontes T_P kann gegeben eines Szenengraphen $\mathcal{G}_S^t$ die zukünftige Bewegungstrajektorie über die Zeitschritte $t + 1$ bis $t + P$ eines oder mehrerer Knoten $u \in \mathcal{V}$ gelernt werden, wobei die Vorhersage $\hat{\boldsymbol{y}} \in \mathbb{R}^3$ ist.

Für die Bewegungsvorhersage kann eine mögliche READOUT-Schicht, wie in Abbildung 4.10 dargestellt, realisiert werden. Die Ausgabe entspricht dabei der 3D-Position aller Knoten über alle Zeitschritte $t + 1$ bis $t + P$.

Die Verlustfunktion auf Knotenebene für eine Knotenregression ist dabei analog zu den in Kapitel 3.2.4 beschriebenen Gleichungen (vgl. Gln. (3.49) und (3.50)). Da in dieser Arbeit Bewegungstrajektorien basierend auf 3D-Positionen gelernt werden sollen, wird die sogenannte *Average Displacement Error* (ADE)-Verlustfunktion verwendet. Durch die Information des Eingangs-Szenengraphen $\mathcal{G}_S^t$ können die vorhergesagten 3D-Positionen verwendet werden, um, analog zur in Kapitel 4.1 beschriebenen Szenengraph-Konstruktion, den zukünftigen Szenengraphen $\mathcal{G}_S^{t+P}$ zu rekonstruieren.

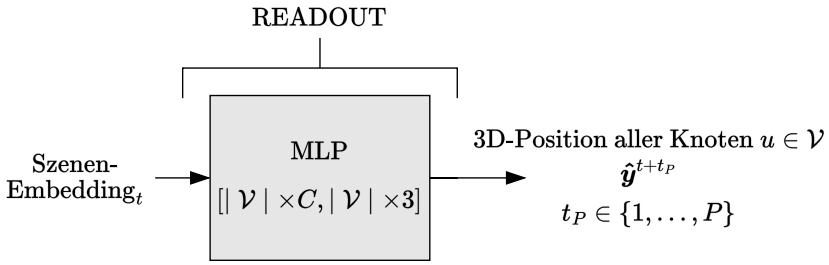


Abbildung 4.10: Schematischer Aufbau der READOUT-Schicht für die Bewegungsvorhersage. Das Szenen-Embedding $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ wird durch ein MLP auf die Dimension | $\mathcal{V}$ | $\times 3$ transformiert, um eine 3D-Position für alle Knoten $u \in \mathcal{V}$ auszulesen. Somit entspricht $\hat{\boldsymbol{y}}^{t+t_P} \in \mathbb{R}^{|\mathcal{V}| \times 3}, t_P \in \{1, \ldots, P\}$ einer Repräsentation auf Knotenebene.

Average Displacement Error Die ADE-Verlustfunktion [PESVG09] ist in der Literatur als Verlustfunktion für Bewegungstrajektorien weit verbreitet [RPH+20]. Die ADE-Verlustfunktion kann auf Basis aller Knoten $u \in \mathcal{V}$ eines Szenengraphen $\mathcal{G}_S^t = (\mathcal{V}, \mathcal{E})$ durch

$$\mathcal{L}_{\text{ADE}}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{P} \sum_{P=1}^{P} d\left(\boldsymbol{y}_n^{t+t_P}, \hat{\boldsymbol{y}}_n^{t+t_P}\right) \tag{4.2}$$

berechnet werden, wobei hier die Knotenmenge $\mathcal{V}$ zeitlich durch den Prädiktionshorizont T_P und räumlich durch die darin vorkommenden N Objekte der Szene aufgeschlüsselt ist. Dadurch soll verdeutlicht werden, dass für jeden Prädiktionsschritt $t_P \in \{1, \ldots, P\}$ der Verlust zwischen dem Knoten des gleichen Objektes zwischen der wahren und der vorhergesagten Position berechnet wird.

Falls lediglich die Positionsinformation eines bestimmten Knotens u von Interesse ist, wie beispielsweise dem Knoten, der die Handposition des Menschen darstellt, so kann für jeden Prädiktionsschritt $t_P \in \{1, \ldots, P\}$ der Verlust zwischen der wahren und vorhergesagten 3D-Position durch

$$\mathcal{L}_{\text{ADE}}(\boldsymbol{y}_u, \hat{\boldsymbol{y}}_u) = \frac{1}{P} \sum_{t_P=1}^{P} d\left(\boldsymbol{y}_u^{t+t_P}, \hat{\boldsymbol{y}}_u^{t+t_P}\right) \tag{4.3}$$

berechnet werden. Dabei wird der ADE auf Basis der euklidischen Distanz $d(\cdot,\cdot)$ zwischen der Wahrheit $\boldsymbol{y}_u$ und der Prädiktion $\hat{\boldsymbol{y}}_u$ für die 3D-Punkte bestimmt.

In diesem Zusammenhang kann auch der sogenannte *Final Displacement Error* (FDE) [PESVG09] angeführt werden. Dieser bestimmt den Fehler des *zeitlich* letzten Prädiktionsschritts P einer oder mehrerer Trajektorien und kann im Kontext von Graphen für alle Knoten mit

$$\mathcal{L}_{\text{FDE}}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{n=1}^{N} d\left(\boldsymbol{y}_n^P, \hat{\boldsymbol{y}}_n^P\right), \tag{4.4}$$

oder für den Knoten u des letzten Prädiktionsschritts P mit

$$\mathcal{L}_{\text{FDE}}(\boldsymbol{y}_u,\hat{\boldsymbol{y}}_u) = d\left(\boldsymbol{y}_u^P, \hat{\boldsymbol{y}}_u^P\right) \tag{4.5}$$

berechnet werden.

In dieser Arbeit liegt dabei vor allem die Hand des Menschen als dynamisches Szenenobjekt zur Erfassung der menschlichen Handlung im Fokus. Die übrigen Objekte, mit Ausnahme des handlungsweisenden Objekts, das vom Menschen manipuliert wird, bleiben weitgehend statisch (vgl. Abbildung 1.2, in der die Aufgabe „kollaboratives Löten“ dargestellt ist und der Mensch den Lötkolben mit der Hand festhält. Die übrigen Objekte bewegen sich nicht und sind daher *statisch*). Daher wird die Formulierung einer Verlustfunktion mit Fokus auf die dynamische Komponente „Hand“ als besonders sinnvoll angesehen. Das Ziel ist es, die Positionen der Hand bzw. die Handtrajektorie hinreichend gut vorherzusagen.

4.2.5 Zeitpunktsvorhersage des Aktionswechsels

Bei der Zeitpunktsvorhersage des Aktionswechsels handelt es sich um die *Vorhersage* des *Aktionswechselzeitpunkts* τ, wann die nächste Aktion stattfindet, gegeben dem Szenengraphen $\mathcal{G}_S^t$. Ähnlich zu der Vorhersage der Bewegungstrajektorie, kann durch die Vorhersage des Aktionswechselzeitpunkts eine weitere Aussage über die Intention des Menschen und den Konsequenzen in und für eine Kollaborationsumgebung mit einem Roboter getroffen werden. So

ist gleichzeitig klar, wann die bisher ausgeführte Aktion endet und wann die nächste Aktion ausgeführt wird. Die zugrundeliegende Lernaufgabe kann auch hier in Form einer Regression beschrieben werden. Der schematische Aufbau kann analog zu den bisherigen Lernaufgaben (vgl. Abbildung 4.11) dargestellt werden.

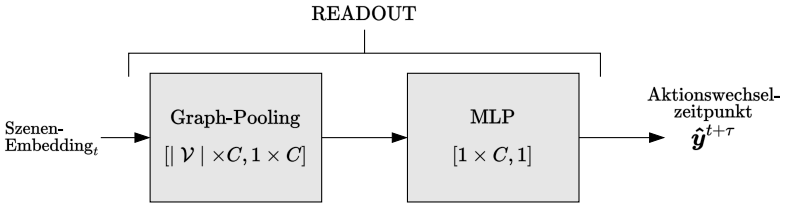


Abbildung 4.11: Schematischer Aufbau der READOUT-Schicht für die Zeitpunktsvorhersage des Aktionswechsels. Das Szenen-Embedding $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ wird durch ein Graph-Pooling zu einer Repräsentation auf Graphenebene $\boldsymbol{z}_{\mathcal{G}}^t$ transformiert (vgl. Gln. (3.43)–(3.45)). Durch ein MLP wird die Repräsentation auf Graphenebene auf die Dimension $\mid \mathcal{V} \mid \times 1$ transformiert, um den Zeitpunkt des Aktionswechsels auszulesen. Somit entspricht $\hat{\boldsymbol{y}}^{t+\tau} \in \mathbb{R}$ einer skalaren Repräsentation.

4.2.6 Zusammenfassung zu den Vorhersagen mit GNNs

Die in dem Kapitel 2.4 beschriebenen Möglichkeiten zur Vorhersage, die durch maschinelle Lernaufgaben abgebildet werden, können als Modell durch die jeweilige beschriebene Architektur eigenständig oder kombiniert verwendet werden. Neben der einzelnen Verwendung und Umsetzung eines Modells für jede Lernaufgabe ist von besonderem Interesse, ein kombiniertes System bzw. eine Architektur zu modellieren, welche gleichzeitig mehrere Lernaufgaben lösen kann. Diese Thematik wird im nachfolgenden Kapitel 5 näher erläutert und mögliche Kombinationen werden beschrieben.

5 Kombination von Vorhersagen für die Mensch-Roboter-Kollaboration

In diesem Kapitel werden aufbauend auf den Vorhersagekategorien in Kapitel 2.4 verschiedene Möglichkeiten beschrieben mehrere Vorhersagen in einer Architektur zu kombinieren. Übergeordnet werden dabei zwei grundsätzliche Ansätze herangezogen, welche durch die beiden Arbeiten [LSS+23a] und [LSS+23b] motiviert sind. Zum einen handelt es sich um eine n-stufige Architektur (vgl. [LSS+23a]), welche die Aktionserkennung und Bewegungsvorhersage mit jeweils eigenständigen Encodern kombiniert. Zum anderen handelt es sich um eine Architektur mit einem einzelnen kombinierten Encoder und mehreren Prädiktionsköpfen (engl. *prediction heads*), die zum kombinierten Lernen der Aktionserkennung, Aktionsvorhersage und Bewegungsvorhersage verwendet werden. Neben den direkt resultierenden Varianten aus diesen beiden veröffentlichten Arbeiten, werden noch weitere vorgestellt, die diesen beiden generellen Varianten zugeordnet werden können. Dazu gehört die Kombination von Lernaufgaben auf Basis geteilter Gewichte des Encoders mit einer folglich zusammengesetzten Verlustfunktion für die jeweilige Aufgabe. In diesem Zusammenhang werden zudem verschiedene Arten, um mehrere Verlustfunktionen zusammenzusetzen, diskutiert.

5.1 Kombination als n-stufige Architektur

Mit einer *n-stufigen* Architektur ist im Kontext dieser Arbeit eine mehrstufige, sequentiell ausführbare Architektur gemeint. Dabei werden die gelernten Merkmale und der Ausgang aus der vorherigen Stufe, von der nachfolgenden Stufe, als zusätzliche Merkmale verwendet. Das vortrainierte Modell wird aus der Teilarchitektur $n-1$ mit *eingefrorenen* Modellgewichten in der Teilarchitektur n wiederverwendet. Dieser Vorgang kann der Thematik des Transferlernens (engl. *transfer learning*) zugeordnet werden. Somit kann der Sachverhalt auch

als n-stufiger Transfer-Learning-Ansatz beschrieben werden. Es wird auf die Übersichtsarbeit [PY10] für eine vertiefte Auseinandersetzung mit der Transfer-Learning-Thematik und in der Literatur zu tiefergehendem maschinellem Lernen auf [GBA16] verwiesen.

Nachfolgend wird die zweistufige Architektur für Aktionserkennung und Bewegungsvorhersage, veröffentlicht in der Arbeit [LSS+23a], näher beschrieben.

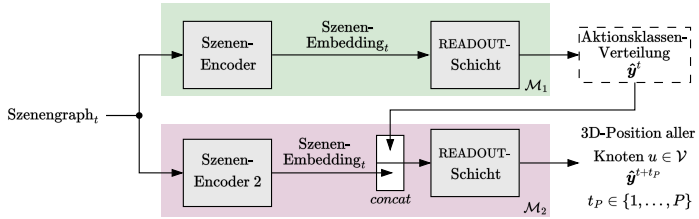


Abbildung 5.1: Abbildung der zweistufigen, GNN-basierten Architektur. In der ersten Stufe wird eine Aktionsklassenverteilung gelernt. Die Information über die gelernte Aktion wird dann in einer zweiten Stufe, zusammen mit den Szenen-Embeddings aus einem zweiten Szenen-Encoder zum Lernen der 3D-Positionen verwendet.

Wie in [LSS+23a] beschrieben, kann eine gemeinsame Aktionserkennung und Bewegungsvorhersage, durch eine zweistufige Architektur umgesetzt werden. Diese zweistufige Architektur ist schematisch in Abbildung 5.1 dargelegt. Analog zu den eigenständigen Lernaufgaben und den schematisch beschriebenen Modellarchitekturen in Kapitel 4.2 (vgl. Abbildung 4.5), kann eine aufgabenspezifische Vorhersage $\hat{\boldsymbol{y}}^t$ bzw. $\hat{\boldsymbol{y}}^{t+P}$ gelernt werden. Bei einer zweistufigen Architektur wird dieser schematische Aufbau in *zwei* Teilarchitekturen abgebildet und verwendet. Mit der *ersten* Teilarchitektur wird ein Modell $\mathcal{M}_1$ gelernt, welches die Aktionsklassenverteilung für die aktuelle Aktion lernt. Dieses Modell $\mathcal{M}_1$ wird mit *eingefrorenen* Gewichten in der zweiten Teilarchitektur verwendet, um die Aktionsklassenverteilung als weiteres Merkmal für die Bewegungsvorhersage und somit für das Modell $\mathcal{M}_2$ zu verwenden. Somit kann $\mathcal{M}_2$ auch als $\mathcal{M}_2 := f_{\mathcal{M}_2}(\mathcal{G}_S^t, f_{\mathcal{M}_1}(\mathcal{G}_S^t))$ durch Modell $\mathcal{M}_1$ als Bestandteil beschrieben werden, wobei beide Modelle den gleichen Eingangs-Szenengraphen $\mathcal{G}_S^t$ verabeiten, dem Modell $\mathcal{M}_2$ jedoch zusätzlich die Vorhersage von $\mathcal{M}_1$ als weiteres Merkmal zur Verfügung steht.

Die eigentliche Verwendung der Vorhersage von Modell $\mathcal{M}_1$, also der Aktionsklassenverteilung, wird nach dem Erlernen der Szenen-Embeddings durch

den *zweiten* Szenen-Encoder, durch Konkatenation (in der Abbildung 5.1 als *concat*) mit jedem Knoten-Embedding $\boldsymbol{h}_u \in \mathcal{V}$ realisiert. Somit ergibt sich für die Knoten-Embeddings $\boldsymbol{h}_u \in \mathbb{R}^{C+A}$, wobei C der Dimension der Knoten-Embeddings und A der Dimension der Aktionsklassenverteilung entsprechen. Wie bereits in Kapitel 4.2.3 angeführt, handelt es sich dabei um eine unnormalisierte Aktionsklassenverteilung. Grundsätzlich kann die Vorhersage der Aktionserkennung auf drei Arten als Merkmal verarbeitet werden: unnormalisiert, normalisiert oder als One-Hot-Kodierung.

Die Vorhersage von $\mathcal{M}_2$ entspricht dann der beschriebenen Bewegungsvorhersage in Kapitel 4.2.4.

Die beschriebene zweistufige Architektur zur Aktionserkennung und Bewegungsvorhersage ist eine lose gekoppelte Art und Weise, mehr als eine Lernaufgabe kombiniert zu lösen. Aus Sicht der Inferenz, also der eigentlichen Anwendung, kann sowohl auf die Vorhersage des Modells $\mathcal{M}_1$ als auch auf die des vollständig trainierten Modells (hier: Modell $\mathcal{M}_2$) zugegriffen werden. Dabei handelt es sich um die aktuell getätigte Aktion des Menschen sowie der Bewegungstrajektorie, abgeleitet aus den 3D-Positionen für einen oder mehrere Knoten.

Hinsichtlich der Modellarchitektur werden zwei Modelle sequentiell trainiert, welche redundante Komponenten in den Teilarchitekturen bzw. Stufen beinhalten, wie etwa dem Szenen-Encoder. Die lose Kopplung soll die einseitige Beeinflussung der gelernten Aktionsklassenverteilung auf die Bewegungsvorhersage motivieren. Dadurch, dass zwei separate Encoder verwendet werden, findet auch keine wechselseitige Einflussnahme auf die Gewichte dieser statt. Lediglich durch die Konkatenation und die damit gemeinsame Verarbeitung in der READOUT-Schicht der Bewegungsvorhersage (zweite Stufe) findet ein kombiniertes Lernen statt. Folglich wird in jeder Stufe eine unabhängige Verlustfunktion getrennt voneinander verwendet.

Unabhängig von den Auswirkungen auf den Lernprozess ist die Umsetzung mehrerer Lernaufgaben durch mehrstufige Architekturen in der Erweiterung um neue Aufgaben komplex. Die Redundanz nimmt zu und die sequentiell durchführbaren Schritte zum Trainieren von Modellen auf Basis solcher Architekturen können komplex und unüberschaubar werden.

Im Folgenden wird eine weitere Kombinationsmöglichkeit vorgestellt, die diese beschriebenen Probleme löst und Chancen zum wechselseitigen Lernen aufgreift.

5.2 Kombination durch gemeinsamen Encoder

Mit einer *kombinierten* Encoder-Architektur ist im Kontext dieser Arbeit eine Architektur aufbauend auf einem einzelnen GNN-Encoder gemeint, der als Grundgerüst für das Lernen einer oder mehrerer Aufgaben des maschinellen Lernens verwendet werden kann. Dabei werden die gelernten Embeddings durch unterschiedliche READOUT-Schichten auf die jeweilige Lernaufgabe zugeschnitten und die prädizierte Zielgröße ausgelesen. Diese Vorgehensweise ist der Thematik des *Multitask Learning* [Car97] zuzuordnen, da ein großer Teil der Modellarchitektur und somit der Netzparameter geteilt und spezifisch für mehrere Aufgaben separate (READOUT-)Schichten verwendet werden. Es wird auf die Literatur von [GBA16] für ein tiefergehendes Verständnis zum Multi-Task-Lernen verwiesen.

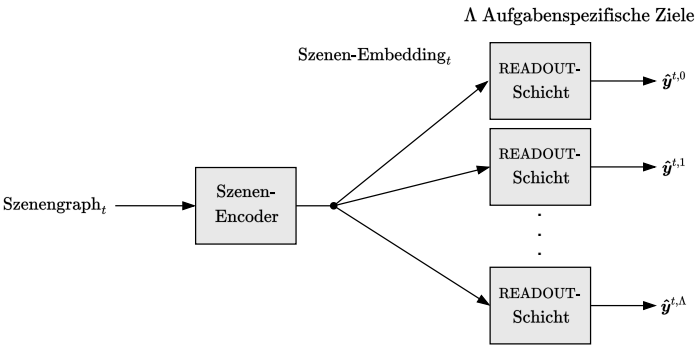


Abbildung 5.2: Generische Abbildung einer Architektur mit geteiltem Szenen-Encoder für Λ unterschiedliche Lernaufgaben durch einzelne READOUT-Schichten dargestellt.

Bei dieser Kombination, schematisch in Abbildung 5.2 dargestellt, bedarf es auch einer Kombination der jeweiligen Verlustfunktionen der einzelnen Lernaufgaben, um den Lernvorgang durch alle einzelnen Lernaufgaben zu bedingen.

Nachfolgend wird die kombinierte Encoder-Architektur für die Aktionserkennung, die Aktionsprädiktion und die Bewegungsvorhersage, veröffentlicht in der Arbeit [LSS+23b], näher beschrieben. Des Weiteren werden auf Basis dieser Kombination auch Möglichkeiten zur Kombination von Verlustfunktionen beschrieben, wobei auch auf die generelle Problematik von Λ-vielen zu kombinierenden Aufgaben eingegangen wird.

5.2.1 Kombinierte Aktionserkennung, Aktionsprädiktion und Bewegungsvorhersage

Diese Art der Kombination von Lernaufgaben in einer Architektur basiert, wie in [LSS+23b] beschrieben, auf der Kombination eines Szenen-Encoders und der resultierenden Verlustfunktionen der jeweiligen READOUT-Schichten. Die beschriebene Architektur kombiniert die Aktionserkennung, Aktionsprädiktion und Bewegungsvorhersage mit dem Ziel, die aktuelle Aktion zum Zeitpunkt t, die Bewegungstrajektorie der Hand, über einen Prädiktionshorizont $T_P := t + t_P$, wobei $t_P \in \{1, \ldots, P\}$ sowie die zukünftige Aktion zum statischen Prädiktionszeitpunkt P, vorherzusagen.

Die Architektur besteht aus einem Szenen-Encoder, der analog zu Kapitel 4.2.2 und Abbildung 4.7 beschrieben werden kann. Wie in Abbildung 5.3 dargestellt, wird aus der Szenen-Embedding-Matrix $\mathbf{Z}_S^t \in \mathbb{R}^{|\mathcal{V}| \times C}$ durch jeweils eine READOUT-Schicht die Aktionsklassenverteilung für die aktuelle und zukünftige Aktion sowie die 3D-Positionen aller Knoten $u \in \mathcal{V}$ ausgelesen.

Neben der Verwendung eines gemeinsamen Encoders bzw. von geteilten Netzparametern, ist die Gestaltung der Verlustfunktionen bzw. der Kombination dieser von besonderem Interesse. Im Kapitel 5.2.2 werden Strategien für eine kombinierte Verlustfunktion vorgestellt. Dabei wird auch auf die Verlustfunktion, die in [LSS+23b] verwendet wurde, eingegangen.

5.2.2 Kombination der Verlustfunktionen

Für die kombinierte Encoder-Architektur wird neben eines zwischen den READOUT-Schichten bzw. Lernaufgaben geteilten Encoders auch eine Strategie zur Kombination der Verlustfunktionen der einzelnen Vorhersagen und der

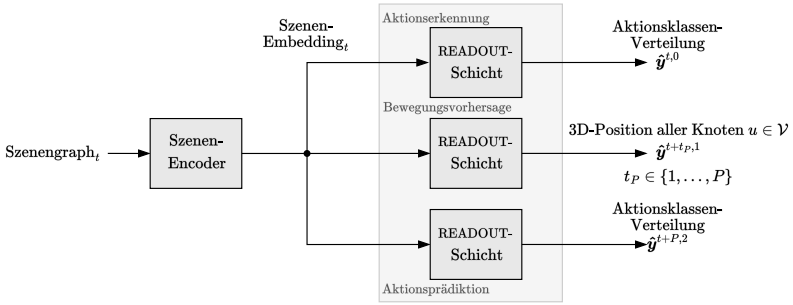


Abbildung 5.3: Schematische Abbildung der Architektur aus [LSS+23b] mit geteiltem Szenen-Encoder für die drei Lernaufgaben: Aktionserkennung, Aktionsprädiktion und Bewegungsvorhersage.

resultierenden Verlustterme benötigt. Die in der Arbeit dargestellten Verlustfunktionen basieren auf einer Klassifikation (vgl. Kapitel 4.2.3) oder einer Regression (vgl. Kapitel 4.2.4 und 4.2.5). In Kapitel 3.2.4 werden für die Knoten-Ebene zur Klassifikation die Kreuzentropie-Verlustfunktion (vgl. Gl. (3.48)) und für die Regression Verlustfunktionen auf Basis des MSE, MAE oder aber dem ADE (vgl. Gln. (3.49),(3.50) und (4.2)) vorgestellt.

Im Folgenden werden Möglichkeiten zur Kombination dieser Verluste, welche schematisch in Abbildung 5.4 dargestellt sind, beschrieben. Dabei ist die multiplikative Kombination nicht berücksichtigt worden, da diese nicht besonders geläufig ist. In der Literatur [Gir15], [RDGF16], [HGDG17], [KGC18], [LJD19], [GKGM20], [ZVVM21] und [DFD22a] gibt es gerade für die additive Kombination von Verlustfunktionen viele Beispiele, die mitunter die nachfolgende Kategorisierung von Kombinationen motivieren.

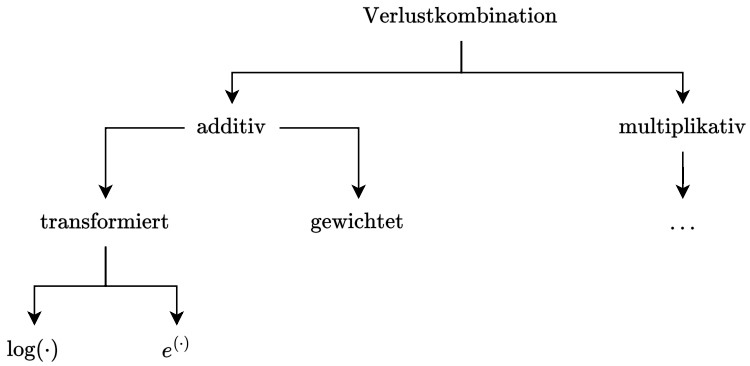


Abbildung 5.4: Übersicht einer möglichen Einteilung der Verlustfunktionen.

Kombination der Verlustfunktionen durch Transformation

Eine Art für eine sinnvolle Kombination von Verlustfunktionen durch Addition, ist die Kombination durch eine Transformation der Verlustfunktionen. Sollen beispielsweise Verlustfunktionen, die im euklidischen Raum oder auf einer logarithmischen Skala operieren, kombiniert werden, ist eine Transformation in einen gemeinsamen Kontext sinnvoll.

Am Beispiel der Addition der Verlustfunktionen für die Aktionserkennung $\mathcal{L}_{\text{AR}}$ und für die Bewegungsvorhersage $\mathcal{L}_{\text{MF}}$, kann beispielsweise eine kombinierte Verlustfunktion durch die Transformation der Kreuzentropie-Verlustfunktion $\mathcal{L}_{\text{AR}}$ in die euklidischen Raum durch

$$\mathcal{L}_{\text{eukl}} = e^{(\mathcal{L}_{\text{AR}})} + \mathcal{L}_{\text{MF}}, \tag{5.1}$$

bzw. der Transformation der Verlustfunktion $\mathcal{L}_{\text{MF}}$ in die logarithmische Skala durch

$$\mathcal{L}_{\log} = \mathcal{L}_{\text{AR}} + \log(\mathcal{L}_{\text{MF}}) \tag{5.2}$$

formuliert werden.

Dieser Ansatz ist durch eine sinnvolle Kombination der Verlustfunktionen, die in unterschiedlichen Kontexten operieren, motiviert. Es werden jedoch beispiels-

weise keine möglichen Verzerrungen durch Skalierungen berücksichtigt. Bei Positionsdaten kann es beispielsweise zu Skalierungsproblemen kommen, da die Position in Metern, Zentimetern oder Millimetern vorliegen kann, wohingegen die Kreuzentropie-Verlustfunktion bereits bei geringen Wahrscheinlichkeiten, relativ kleine Werte annimmt. Daher ist es sinnvoll, solche Skalierungsprobleme bei der Kombination von Verlustfunktionen zu berücksichtigen. Auch der Umgang mit Verlustwerten kleiner oder gleich Null stellt bei der Verwendung einer solchen Transformation in den logarithmischen Raum ein Problem dar, da diese dafür nicht definiert ist. Im Folgenden wird ein weiterer Ansatz vorgestellt, der diese Probleme zu löst.

Kombination der Verlustfunktionen mit Gewichtsvorfaktoren

Um die Probleme, die im bisherigen Ansatz zur Kombination von Verlustfunktionen angemerkt wurden, anzugehen, wird hier auf parametrisierbare Gewichtsvorfaktoren für die eigentliche Addition gesetzt. Diese Art der Kombination findet sich in der Literatur als gängige Methode wieder und wird auch in der Arbeit [LSS+23a] und [LSS+23b] angewandt.

Im Folgenden wird der allgemeine Fall sowie der raus resultierende vereinfachte Fall zur additiven Kombination von Verlusttermen mittels parametrisierbaren Gewichtsvorfaktoren beschrieben.

Allgemeiner Fall Der allgemeine Fall beschreibt die additive Kombination mehrerer Verlustfunktionen mittels Gewichtsvorfaktoren zu einem kombinierten Verlust $\mathcal{L}_{\text{gewichtet}}$. Dieser ergibt sich aus allen Verlusttermen, gegeben durch eine spezifische Verlustfunktion $\mathcal{L}_\lambda$ sowie deren Gewichtsvorfaktoren w_λ. Der kombinierte Verlust somit definiert als

$$\mathcal{L}_{\text{gewichtet}} = \sum_{\lambda=1}^{\Lambda} w_\lambda \mathcal{L}_\lambda, \qquad w_\lambda \geq 0. \tag{5.3}$$

Die Kombination dreier Verlustfunktionen $\mathcal{L}_{\text{MF}}$, $\mathcal{L}_{\text{AR}}$ und $\mathcal{L}_{\text{AP}}$ kann somit beispielsweise als

$$\mathcal{L}_{\text{gewichtet}} = w_1 \cdot \mathcal{L}_{\text{MF}} + w_2 \cdot \mathcal{L}_{\text{AR}} + w_3 \cdot \mathcal{L}_{\text{AP}} \tag{5.4}$$

beschrieben werden. Die Gewichte $w_\lambda, \lambda \in \{1, \ldots, \Lambda\}$ können zur Bestimmung von $\Lambda - 1$ Gewichten reduziert werden, da die verbleibenden Gewichte durch die Optimierung ausgeglichen werden, sodass die Formulierung in Gl. (5.4) zu

$$\mathcal{L}_{\text{gewichtet}} = \mathcal{L}_{\text{MF}} + w'_2 \cdot \mathcal{L}_{\text{AR}} + w'_3 \cdot \mathcal{L}_{\text{AP}} \tag{5.5}$$

vereinfacht werden kann, wobei dann für den Verlustterm $\mathcal{L}_{MF}$ der Gewichtsfaktor $w_1 = 1$ ist. Die Kombination durch unabhängige Gewichte kann für endlich viele Verlustfunktionen automatisiert erweitert werden.

Einfacher Fall Der einfache Fall beschreibt die additive Kombination mehrerer Verlustfunktionen, wie im allgemeinen Fall (vgl. Gl. (5.3)) mit $w_\lambda = 1, \lambda \in \{1, \ldots, \Lambda\}$ und somit

$$\mathcal{L}_{\text{einfach}} = \sum_{\lambda=1}^{\Lambda} \mathcal{L}_\lambda. \tag{5.6}$$

Es ergibt sich für die Addition von drei Verlustfunktionen, Aktionserkennung (AR) als $\mathcal{L}_{\text{AR}}$, Aktionsprädiktion (AP) als $\mathcal{L}_{\text{AP}}$ und Bewegungsvorhersage (MF) als $\mathcal{L}_{\text{MF}}$, eine additive Kombination durch

$$\mathcal{L}_{\text{einfach}} = \mathcal{L}_{\text{AR}} + \mathcal{L}_{\text{AP}} + \mathcal{L}_{\text{MF}}. \tag{5.7}$$

Kriterien zur Bestimmung der Gewichte Zur Bestimmung der Gewichte für eine Kombination von Verlustfunktionen müssen die Anzahl und der Wert der Gewichte festgelegt werden. Der Wert für die einzelnen Gewichte wird durch eine Suche für die Gewichte selbst wird durch eine Hyperparameteroptimierung realisiert, die in Kapitel 6.1 näher beschrieben wird. Die Anzahl der Gewichte folgt der Anzahl der zu kombinierenden Verlustterme und kann auf $\Lambda - 1$ viele Gewichte reduziert werden. Für die Suche der Gewichte durch Hyperparameteroptimierung bedeutet dies ebenfalls einen reduzierten Suchraum.

Zusammenfassung Für eine Kombination mehrerer Verlustfunktionen gibt es unterschiedliche Möglichkeiten. Bei der additiven Kombination können parametrisierbare Gewichtsvorfaktoren für die einzelnen Verlustterme oder aber eine Transformation dieser verwendet werden.

Die additive Kombination wie im einfachen Fall beschrieben ist initial hilfreich, um den kombinierten Trainingsvorgang zu untersuchen. Diese kann bei gleichartigen Verlustfunktionen, wie beispielsweise zur Kombination der Verluste von Aktionserkennung und Aktionsvorhersage, welche beide auf die Kreuzentropie-Verlustfunktion zurückzuführen sind, ausreichend sein. Die Verwendung einer additiven Kombination durch Transformation der Verlustfunktionen in den jeweiligen Abbildungsraum, um die zu kombinierenden Verlustterme in einen gemeinsamen Wertebereich abzubilden, kann ebenfalls hilfreich sein.

Bei der Verwendung parametrisierbarer Gewichte wie im allgemeinen Fall beschrieben kann datengetrieben ohne explizite Vorkenntnis über die Abbildungsräume der jeweiligen Verlustfunktionen die optimale Kombination erreicht werden. Die Festlegung der einzelnen Gewichte bedarf dabei eines Suchvorgangs, welcher abhängig von der Anzahl der Hyperparameter Λ ist. Zur Beschleunigung des Modellfindungsprozesses für eine optimale Kombination kann die Suche auch auf $\Lambda - 1$ Hyperparameter reduziert werden, indem ein Gewicht explizit definiert wird.

Im folgenden Kapitel 6 wird die Evaluation und somit auch der Modellfindungsprozess, welcher eine Hyperparameteroptimierung (vgl. Kapitel 6.1) beinhaltet, vorgestellt und erläutert.

6 Evaluation

In diesem Kapitel wird die Evaluation der vorliegenden Anwendung für ausgewählte Datensätze vorgestellt. Diese basiert dabei auf den Grundlagen zu GNNs aus Kapitel 3 sowie der in Kapiteln 4 und 5 beschriebenen Anwendung, den zu lösenden Lernaufgaben, Vorhersagemethoden und den weiterführenden Umsetzungen zur kombinierten Vorhersage dieser Aufgaben.

Im Rahmen dieser Arbeit und der erarbeiteten Methoden ist die Repräsentation von Daten als Graphstruktur (vgl. Kapitel 4.1) von besonderer Bedeutung. Die Repräsentation der Daten als Graphstruktur und das Implementieren von GNNs wurde mit Hilfe der Open-Source-Bibliothek PyTorch Geometric [FL19] umgesetzt.

6.1 Hyperparameteroptimierung

Wie bereits im Verlauf dieser Arbeit angemerkt, gibt es viele Stellen, an denen ohne eine datengetriebene Untersuchung keine Entscheidungsfindung für eine Auslegung der Architektur möglich ist. Genauer gesagt handelt es sich um die Detail-Entscheidung innerhalb der Modellierung eines Szenengraphen (vgl. Kapitel 4.1), bis hin zur finalen Form der optimalen Architektur, die auf Basis der Module, beschrieben in Kapitel 4, zusammengesetzt wird.

Zur Entscheidungsfindung wird eine im Arbeitsablauf des maschinellen Lernens weit verbreitete Methode verwendet - die sogenannte Hyperparameteroptimierung. Die generelle Beschreibung der Hyperparameteroptimierung ist an die Literatur von [GBA16] angelehnt, wobei im Speziellen die Arbeiten [BBBK11], [BB12] und [BYC13] herangezogen werden können.

Wie in [GBA16] beschrieben, werden drei Arten von Algorithmen zur Suche von Hyperparameter in Betracht gezogen. In der Raster- und Zufallssuche [BBBK11], [BB12] werden Modelle in voneinander unabhängigen Iterationen,

mit einer Wahl aus Hyperparametern aus einem definierten Raum, trainiert. Gegeben einer Validierungsmetrik werden dann iterativ die optimalen Parameter bestimmt. Weiterhin gibt es das sogenannte Modell-basierte Suchverfahren, auch sequentielle Modell-basierte Optimierung (engl. *Sequential Model-Based Optimization* (SMBO)) genannt, welche auf Bayes'scher Statistik basieren und sich über eine Zielfunktion bedingt, den optimalen Parametern annähern. Zu diesen SMBO-Ansätzen gehören beispielsweise der baumstrukturierte Parzen-Schätzer [BBBK11] (engl. *Tree-Structured Parzen Estimator* (TPE)) oder die sequentielle modellbasierte Algorithmuskonfiguration (engl. *Sequential Model-Based Optimization for General Algorithm Configuration* (SMAC)) [HHL11], welche sich in der Modellierung der statistischen Beziehung zwischen den Hyperparametern unterscheiden. SMBO-Methoden sind besonders geeignet, wenn der Hyperparameterraum groß ist und diese aus einer kontinuierlichen Menge von Zahlen ausgewählt werden . Da der für die vorliegende Arbeit definierte Hyperparameterraum diese beiden Eigenschaften vorweist, wird ein SMBO-Ansatz für die Optimierung konkret die TPE-Methode von [BBBK11] unter Verwendung der Open-Source-Bibliothek Optuna [ASY+19] verwendet.

Weiterhin ist es für das generelle Verständnis wichtig, dass eine Hyperparameteroptimierung aus einer bestimmten Anzahl an Versuchen V besteht. Jeder Versuch v evaluiert eine Hyperparameterkonfiguration $\mathcal{H}_v$, für welche ein Modell $\mathcal{M}$ auf einem Trainingsdatensatz $\mathcal{D}_{\text{train}}$ trainiert und auf einem Validierungsdatensatz $\mathcal{D}_{\text{val}}$ validiert wird. Das Ergebnis der Validierung wird über eine Metrik für jede Epoche des jeweiligen Versuches v dokumentiert. Iterativ werden Hyperparameterkonfigurationen für eine bestimmte Anzahl an Versuchen evaluiert. Der Versuch mit der besten dokumentierten Metrik wird als optimale Hyperparameterkonfiguration $\mathcal{H}^*$ verwendet.

Hyperparameterraum Der Hyperparameterraum ergibt sich aus der Spezifikation der jeweiligen Architektur und ihrer Komponenten (vgl. Kapitel 5) sowie den spezifischen Parametern für den Trainingsprozess, wie beispielsweise der Batchgröße oder der Lernrate. Eine Übersicht des Hyperparameterraums ist in der Abbildung 6.1 exemplarisch dargestellt und beschrieben.

Zur Vereinfachung der Hyperparametersuche und da die verschiedenen Parameter teilweise eine unterschiedliche Wichtigkeit für die jeweilige GNN-Variante haben, wird für jede GNN-Variante eine separate Suche durchgeführt.

Hyperparameter	**Bereich**
GNN-Variante	[Varianten]
Embedding-Dimension	{32, 64, 96, 128, . . . , 1024}
# Blöcke	[1, 42]
Dropout-GNN	[0, 1]
Bias-Term	{true, false}
Aggregation	{mean, max, sum}
Graph-Pooling	{mean, max, sum}
Pre-Aktivierung	{true, false}
Aktivierung	{ReLU, PReLU}
Residualschicht	{true, false}
Pre-Normalisierung	{true, false}
Art der Normalisierung	{Batch, Layer}
Selbst-Schleifen	{true, false}
Kantengewichte	{true, false}
Kantenart	{gerichtet, ungerichtet}
# MLP-Schichten	{1, 2}
MLP-Dropout	[0, 1]
Lernrate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$
# Batchgröße	{32, 64, 128, . . . , 1024}
# Patience	[3, 20]
Gewichtsterme (Verlust-Kombination)	[0, 2]

Tabelle 6.1: Beschreibung des Hyperparameterraums: Darstellung der wichtigsten Parameter

Neben der Möglichkeit durch eine Hyperparameteroptimierung Parameter zu finden, die zu einem optimalen Modell führen, können des Weiteren auch Strategien direkt in den Trainingsprozess integriert werden. Die beiden Strategien, die im Rahmen dieser Arbeiten neben einer Hyperparameteroptimierung ergänzt genutzt werden, sind das Pruning, welches speziell in den Hyperparameteroptimierungsprozess integriert wird, und das Early-Stopping, wobei letztere Strategie sowohl in der Hyperparameteroptimierung, als auch im finalen Training eines Modells Verwendung findet. Nachfolgend wird die Funktionsweise kurz erläutert.

Pruning-Strategie Hierunter versteht man das vorzeitige Beenden eines Versuchs mit der gewählten Hyperparameterkonfiguration verstanden. Dabei wird der Verlauf (Ergebnisse der Validierung) der vorigen Versuche herangezogen, um eine Entscheidung zur vorzeitigen Beendigung eines Versuchs zu treffen. In dieser Arbeit wird die sogenannte Median-Pruning-Strategie, bereitgestellt in [ASY+19], verwendet. Es wird dabei der Median-Wert über die Validierungsmetrik der dokumentierten Versuche für jede Trainingsepoche mit der Validierungsmetrik des aktuellen Versuchs verglichen. Wenn für die Epoche des aktuellen Versuchs die Güte der Validierungsmetrik geringer ist, wird der Versuch vorzeitig gestoppt und eine neue Hyperparameterkonfiguration ermittelt und ein neuer Versuch mit ebendieser begonnen.

Early-Stopping Eine weitere Strategie, die sowohl in der Hyperparameteroptimierung als auch im finalen Training zum Lernen des finalen Modells verwendet wird, ist das sogenannte Early-Stopping. Die Parameter θ eines Modells $\mathcal{M}$ werden mit Hilfe der Early-Stopping-Strategie so bestimmt, dass sie den Gewichten der Epoche im Training mit der besten Validierungsmetrik entsprechen, wobei das Training durch ein Kriterium zum Stoppen terminiert wird. Dies ergibt sich durch das Erreichen einer maximalen Anzahl an Epochen im Training oder durch das Erreichen einer frühzeitigen Schranke an Epochen (Early-Stopping), für die sich die Validierungsmetrik nicht mehr verbessert [GBA16], auch Geduld (engl. *patience*) genannt.

Während der Hyperparameteroptimierung soll neben einer Pruning-, eine Early-Stopping-Strategie zusätzlich dabei helfen, den Hyperparameterraum schneller zu erforschen, um die optimalen Parameter zu finden. Während des

Trainings des finalen Modells soll zusätzlich durch das Early-Stopping eine Überanpassung verhindert werden.

Zusammenfassend gibt es jedoch keine Garantie für eine Verbesserung, weder durch den Einsatz einer Pruning- oder einer Early-Stopping-Strategie. Arbeiten, die sich speziell mit dem Verhalten trainierter Modelle mit *ungesehenen* Daten beschäftigen, wie beispielsweise [NKB+19] und [HY20], beschreiben, dass durch längeres Training über ein lokales Minimum des Validierungsfehlers hinaus, sich erneut eine Konvergenz des Validierungsfehlers beobachten lassen kann. Dies würde auf ein alternatives Kriterium zum Stoppen hinweisen. Da die Anwendung dieser Strategien in der vorliegenden Arbeit jedoch durch den großen Hyperparameterraum motiviert ist um hinreichend geeignete Hyperparameter zu finden, werden solche Phänomene nicht weiter berücksichtigt.

6.2 Durchgeführte Experimente

In diesem Kapitel werden die durchgeführten Experimente auf Basis der verwendeten Datensätze vorgestellt. Des Weiteren wird die Vorverarbeitung der Datensätze zur Nutzung mit GNNs sowie deren Unterteilung für das Training erläutert. Darüber hinaus werden die Baseline-Methoden, die zum Vergleich herangezogen und die Beschreibung der Experimente, die durchgeführt werden, aufgezeigt.

6.2.1 Beschreibung der Daten

Im Folgenden werden die Datensätze zur Evaluation der in den Kapiteln 4 und 5 vorgestellten Methoden beschrieben. Außerdem wird in diesem Zuge die Datenvorverarbeitung zur Generierung von Szenengraphen (vgl. Kapitel 4.1) erläutert. Abschließend wird die Unterteilung der vorverarbeiteten Datenpunkte für die Experimente mit GNNs beschrieben.

Bimanual-Actions-Datensatz

Der sogenannte *Bimanual Actions Dataset* (Bimacs) [DWA20] ist ein Datensatz, welcher im Kontext von Küchen- und Werkstattaufgaben, Daten für das Lernen von zweihändigen Aktionen bereitstellt. Er umfasst insgesamt neun verschiedene Aufgaben, die von sechs Menschen ausgeführt werden, wobei jeder Mensch die jeweilige Aufgabe zehnmal ausführt. Bereitgestellt werden die Daten in Form von RGB- und Tiefenbildern sowie als *abgeleitete* Daten, die für jedes Bild Kennzeichen zu Objekten, inklusive des Menschen, beinhalten. Zu diesen Kennzeichen gehören die 2D- und 3D-Positionen aller Objekte (Begrenzungsrahmen) sowie ein Klassen- und Klasseninstanzkennzeichen für jedes Objekt, aber auch das jeweilige Aktionskennzeichen (*Prädikat*), abgeleitet aus der Tätigkeit der linken und der rechten Hand. Weiterhin wird Information zur menschlichen Pose als 2D-Skelettrepräsentation sowie abgeleitete semantische Relationen zwischen sämtlichen Objekten, einschließlich den Händen des Menschen, bereitgestellt.

Collaborative-Action-Datensatz

Der CoAx-Datensatz [LSSD22] bildet speziell industrielle Montageaufgaben in Zusammenarbeit mit kollaborativen Roboterarmen ab. Er umfasst drei verschiedene Montageaufgaben, die von sechs Menschen ausgeführt werden, wobei jeder Mensch die jeweilige Aufgabe zehnmal ausführt. Die Aufgaben sind exemplarisch in Abbildung 6.1 dargestellt. Bereitgestellt werden die Daten in Form von RGB- und Tiefenbildern sowie über Kennzeichen zu Objekten, inklusive des Menschen und Roboters, mit deren 2D- und 3D-Position.

Abbildung 6.1: Ausgewählte Bilder aus dem CoAx-Datensatz [LSSD22]. Jede Reihe zeigt eine der drei beinhalteten Aufgaben. Die erste Reihe zeigt die Aufgabe *Valve Terminal Plug & Play*, welche den Zusammenbau und die Inbetriebnahme einer Ventilinsel beschreibt. Die zweite Aufgabe *Valve Assembly* zeigt das Zusammenbauen eines Ventils, aus dessen einzelnen Bestandteilen. In der dritten Aufgabe *Collaborative Soldering* wird eine Kollaboration zwischen Mensch und Roboter beschrieben, in welcher der Roboter als dritte Hand dieses Menschens agiert, indem der Roboter eine Lötplatine festhält, damit der Mensch einen Kondensator darauf verlöten kann.

Außerdem enthält jedes Frame einer Videosequenz ein Aktionskennzeichen, welches die vom Menschen ausgeführte Aktion für das jeweilige Frame kennzeichnet. Der CoAx-Datensatz verfügt dabei über Aktionskennzeichen, die auf die Tätigkeit der rechten Hand zurückzuführen sind. Neben der Aktion (*Prädikat*) selbst, liegt darüber hinaus noch eine weitere Information zum Objekt, welches manipuliert wird (*Prädikat+Objekt*), vor (vgl. Abbildung 6.2). Diese sind für die drei Aufgaben, die im CoAx-Datensatz enthalten sind, in der Abbildung 6.2 beschrieben.

Aufgabe 1	Aufgabe 2	Aufgabe 3
approach	approach	approach
grab_valve	grab_valve	grab_capacitor
plug_valve	join_valve	wait_for_robot
grab_screwdriver	grab_screws	plug_capacitor
screw_valve	plug_screws	wait_for_robot
release_screwdriver	grab_screws	grab_soldering_tin
grab_hose	plug_screws	grab_soldering_iron
plug_hose	grab_screwdriver	solder_capacitor
retreat	screw_valve	release_soldering_iron
	release_screwdriver	release_soldering_tin
	place_valve	retreat
	grab_membrane	
	place_membrane	
	retreat	

Abbildung 6.2: Aktionen im CoAx-Datensatz, abgebildet in der vorliegenden sequentiellen Folge für die drei enthaltenen Montageaufgaben. Die Aktionen sind hier in der expliziten Form als *Prädikat+Objekt* dargestellt und nach jeweiligem *Prädikat* koloriert.

Datenvorverarbeitung

Die Datenvorverabreitung dient als notwendiger Schritt, um Szenengraphen, wie in Kapitel 4.1 beschrieben, zu generieren. Außerdem müssen für die Experimente zur Aktionsklassifikation und Bewegungsvorhersage die notwendigen Kennzeichen vollständig sein, um einen Lernvorgang zu ermöglichen. Für eine Aktionsklassifikation werden also Aktionskennzeichen und für eine Bewegungsvorhersage die Positionsinformation für jeden Datenpunkt benötigt. Speziell für die Vorhersage der Bewegung der Hand des Menschen muss somit die Verfügbarkeit der Handposition gewährleistet sein.

Für die Vervollständigung fehlender Aktionskennzeichen gibt es keine systematische Regel. Daher werden Datenpunkte mit fehlenden Aktionskennzeichen ignoriert. Für die Bewegungsvorhersage kann jedoch beispielsweise durch Interpolation oder Auffüllen, aus den vorherigen oder nachfolgenden Posi-

tionsinformationen, bis zu einem gewissen Punkt, die fehlende Information wiederhergestellt werden. Nicht zuletzt kann generell durch die Interpolation fehlender Positionsinformation von Objekten für mehr Information bzw. vollständigere Szenengraphen gesorgt werden.

Das Resultat der Datenvorverarbeitung ist somit eine Menge an Graphdatenpunkten $\mathcal{D}$, in dieser Arbeit auch Graphdatensatz genannt, welcher folglich aus Szenengraphen $\mathcal{G}_S^t$ für jeden Zeitpunkt t besteht.

Datenunterteilung

Die Unterteilung der Daten bezieht sich primär auf den Trainingsvorgang und die anschließende Evaluation. Um die vorgeschlagene Unterteilung in Trainings-, Validierungs- und Testdatensatz durchzuführen, muss eine Strategie gefunden werden, die wie in [HTF09] und [GBA16] oder in anderer Literatur im Bereich des maschinellen Lernens verbreitet ist. Darüber hinaus wird für die überwachte Lernaufgabe zur Klassifikation der Aktionskennzeichen in den Datensätzen auch eine Klassenbalancierung herangezogen. Dies soll die Überrepräsentation bestimmter Aktionen im Kontext menschlicher Aktionen ausgleichen und somit einen überproportionalen Einfluss auf das Training und die Gewichte eines lernenden Netzes verhindern. Diese beiden Aspekte werden im Folgenden für die beiden Datensätze (vgl. Bimacs- und CoAx-Datensatz) näher beschrieben.

Trainings-, Validierungs- und Testdatensatz Die beschriebenen Datensätze weisen eine spezifische Struktur auf, die sich auf die einzelnen Versuchspersonen bezieht, welche die Aufgaben ausführen. Wie in den durchgeführten Experimenten von [DWA20] vorgeschlagen, kann diese Struktur zur Aufteilung der Daten genutzt werden. Somit kann der Datensatz $\mathcal{D}$ als k disjunkte Teilmengen $\mathcal{S}_k$ mit $\mathcal{D} = \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$, wobei $1 \leq k \leq 6$ interpretiert werden. So kann der Testdatensatz durch eine dieser disjunkten Teilmengen gewählt werden. Der Trainingsdatensatz $\mathcal{D}_{\text{train}}$ kann folglich durch die restlichen Aufzeichnungen der übrigen Versuchspersonen gebildet werden. Ferner kann aus dem Trainingsdatensatz $\mathcal{D}_{\text{train}}$ eine weitere Teilmenge zur Validierung aus den k Teilmengen gewählt werden. Da jede Teilmenge k aus $\mathcal{A}$ Aufgaben besteht, welche jeweils zehnmal wiederholt werden, können n Wiederholungen (hier:

$n = 1$) jeder Aufgabe $\mathcal{A}$, einer jeden Teilmenge k des Trainingsdatensatzes $\mathcal{D}_{\text{train}}$ gewählt werden, um den Validierungsdatensatz $\mathcal{D}_{\text{val}}$ zu bilden.

Die genaue Unterteilung für die beiden Datensätze ist exemplarisch in Tabelle 6.2 dargestellt.

Datensatz	**Beide**	**CoAx**		**Bimacs**	
	[%] Aufnahmen	# Aufnahmen	[%] Frames	# Aufnahmen	[%] Frames
Training	75.00	135.00	74.52	405.00	76.23
Validierung	8.33	15.00	8.34	45.00	8.15
Test	16.67	30.00	17.14	90.00	15.62

Tabelle 6.2: Datensatzunterteilung: Für die Berechnung der Verteilung der Datenpunkte wurde der Datensatz mit der Testteilmenge $\mathcal{S}_5$ und jeweils einer Wiederholung ausgewählt.

Klassenbalancierung Im Kontext der Aktionsklassifikation beschreibt die Klassenbalancierung den Umgang mit unter- und überrepräsentierten Aktionsklassenkennzeichen im Datensatz $\mathcal{D}$. Gerade bei Aufgaben, welche von Menschen ausgeführt werden, können einzelne Aktionen in *statische* und *dynamische* Aktionen aufgeteilt werden. Ob eine Aktion statisch oder dynamisch ist, bezieht sich hierbei auf die ganzheitliche, räumliche Bewegung der aktiven Hand des Menschen, während der jeweiligen ausgeführten Aktion. Dabei kann es sein, dass dadurch für einige dynamische Aktionen (z.B. das Greifen eines Schraubenziehers), die gesamte Ausführzeit kürzer ist, als für andere Aktionen, welche hingegen statisch sind und somit eine längere Zeit zur Ausführung in Anspruch nehmen (z.B. das Verschrauben eines Werkstücks). Für eine statisch oder dynamische Aktion resultiert dies quantitativ in *mehr* oder *weniger* Aktionsklassenkennzeichen. Diese beiden Fälle können im Trainingsvorgang somit dazu führen, dass das trainierte Modell durch überrepräsentierte Datenpunkte einer Aktion überangepasst oder durch unterrepräsentierte Datenpunkte unterangepasst ist. In der Abbildung 6.3 ist dies für die Aktionsklassen im CoAx-Datensatz dargestellt.

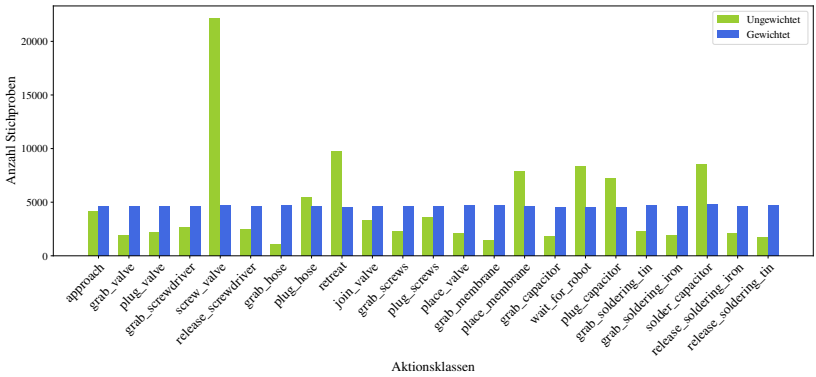


Abbildung 6.3: Klassenbalancierung am Beispiel des CoAx-Datensatzes. In *grün* ist die Häufigkeit vor der Balancierung und in *blau* nach der Balancierung der Klassen dargestellt.

Um dieser möglichen Problematik vorzubeugen, kann eine gewichtete Stichprobenentnahme verwendet werden, welcher bei der Auswahl von Datenpunkten aus dem Datensatz $\mathcal{D}$ das Aufkommen der jeweiligen Aktionsklassen berücksichtigt und somit für ein balanciertes Verhältnis dieser sorgt.

Der vorliegenden Arbeit dient die Klassenbalancierung als ein weiteres Werkzeug zur Optimierung des Trainingsprozesses. Dies ist vor allem für die Aktionsklassifikation wichtig, da diese wie beschrieben durch die Aktionsklassenkennzeichen während des Trainings hauptsächlich beeinflusst wird. Für den Trainingsvorgang der Bewegungsvorhersage sind beispielsweise besonders variable Positionsdaten der Szenenobjekte wichtig. Aus diesem Grund ist der Einfluss der Aktionsklassenkennzeichen hier zu vernachlässigen, um so viele unterschiedliche Positionsdaten wie möglich im Trainingsprozess zu berücksichtigen. Auch bei einer Kombination von Aktionsklassifikation und Bewegungsvorhersage wird zugunsten der Letzteren auf eine Klassenbalancierung verzichet.

6.2.2 Beschreibung der Baseline-Verfahren

Zur Evaluation der vorgeschlagenen Modellarchitektur (vgl. Kapitel 4.2) für die Lernaufgabe, Aktionserkennung und Bewegungsvorhersage werden Baseline-

Verfahren hinzugezogen. Diese unterscheiden sich jeweils für die Aktionserkennung und die Bewegungsvorhersage. Nachfolgend werden die verschiedenen Baseline-Verfahren kurz erläutert.

Aktionserkennung

Motiviert durch den aufgeführten Stand der Technik in Kapitel 2 mit den bisherigen Vorhersagemethoden und den Kriterien, die sich auch aus dem dieser Arbeit zugrundeliegenden Anwendungsszenario ergeben, wird die vorgeschlagene Methode von [DWA20] als Basis für die Experimente zur Aktionserkennung verwendet. Gegeben der in Kapitel 6.2.1 beschriebenen Datensätze wird die Methode von [DWA20] als Basis für die Aktionserkennung verwendet. Dabei wird die in [DWA20] vorgestellte Variante mit den besten Ergebnissen herangezogen.

Funktionsweise Zur Aktionserkennung wird ein Szenengraph (vgl. Kapitel 4.1) verwendet. Dieser Szenengraph verfügt über räumliche Relationen (vgl. [ZKTW18]) für alle Szenenobjekte und zeitliche Relationen für zeitlich zusammenhängende Szenenobjekte. Die Art der Beschreibung eines Szenengraphen bezüglich seiner räumlichen Relationen für einen Zeitschritt, entspricht dabei der eines vollständigen Graphens, was in dieser Arbeit auch als O2O referenziert wird (vgl. Abbildung 4.4). Zum Erlernen von Szenen-Embeddings wird nach [BHB+18] ein GNN verwendet. Dieses verwendet analog zur Beschreibung des generischen Message-Passings (vgl. Kapitel 3.2.5) für Knoten- und Kanten-Merkmale des Graphen sowie für das globale Graphattribut, separate neuronale Netze, um das der aktuellen Aktion entsprechende globale Graphattribut zu erlernen (vgl. Kapitel 3.2.5).

Bewegungsvorhersage

Für die Bewegungsvorhersage werden zwei etablierte Methoden zur Bewegungsvorhersage herangezogen. Beide beruhen auf physikalischen Annahmen bezüglich der zugrundeliegenden Geschwindigkeit der Bewegung.

Funktionsweise Die einfachste Art der Bewegungsvorhersage ist, gesetzt dem Fall, dass die Position x_t konstant bleibt, also keine Bewegung stattfindet und über einen Prädiktionshorizont T_P die Geschwindigkeit $\mathrm{v} = 0\mathrm{m/s}$ entspricht. Diese Methode ist in der Literatur auch als *Zero Velocity* (ZV) bekannt und wird beispielsweise in [MBR17] als Methode zum Vergleich der Bewegungsvorhersage herangezogen. Formulieren lässt sich dies durch

$$\mathrm{x}_{t+t_P} = const, t_P \in \{1, \ldots, P\}. \tag{6.1}$$

Eine weitere Methode zur Bewegungsvorhersage ergibt sich durch die Annahme konstanter Geschwindigkeit. In dieser Arbeit wird die von [LS17] vorgeschlagene Methode zur Bewegungsvorhersage, eingeführt als *Velocity-Based Position Projection* (VBPP), verwendet. Unter der Annahme, dass die momentane Geschwindigkeit aus vergangenen Positionsdaten geschätzt wird, kann die zukünftige Bewegung vorhergesagt werden. Dabei wird zur Bestimmung der momentanen Geschwindigkeit das Savitzky-Golay-Filter [SG64] auf die Positionsdaten angewandt, um diese für die Bestimmung der momentanen Geschwindigkeit zu glätten (vgl. Polynomialregression). Folglich wird aus einer Menge an beobachteten Positionsdaten die momentane Geschwindigkeit $\tilde{\mathrm{v}}_t$ geschätzt, mit der dann die zukünftige Bewegung, formuliert durch

$$\hat{\mathrm{x}}_{t+t_P} = \mathrm{x}_t + \tilde{\mathrm{v}}_t \cdot t_P, t_P \in \{1, \ldots, P\} \tag{6.2}$$

berechnet wird. Die Geschwindigkeit $\tilde{\mathrm{v}}_t$ ist somit durch eine Schätzung gegeben.

Bei der Bewegungsvorhersage im Kontext der MRK bei Montageaufgaben ist entscheidend, dass sich die Bewegungen des Menschen aus den sequenziell auszuführenden Aktionen am definierten Arbeitsplatz ergeben. Diese Aktionen dienen dazu, Werkstücke zu manipulieren und die Gesamtaufgabe zu erfüllen. Solche Bewegungen erfordern oft hohe Präzision und Aufmerksamkeit, wie beispielsweise beim Löten, und sind daher häufig statisch. Aus diesem Grund sind diese beiden Methoden als Vergleichsmethoden geeignet. Für einen generellen Vergleich bezüglich GNNs gegenüber anderen Ansätzen zur Bewegungsvorhersage, wird auf die Arbeiten [YXL18] [LYH+20] verwiesen, welche zeigen, wie GNNs im Vergleich zu CNNs, RNNs, TCNs oder weiteren Methoden abschneiden.

6.2.3 Beschreibung der Experimente

Die in dieser Arbeit vorgestellte Methode für Aktionsklassifikation und Bewegungsvorhersage wird experimentell untersucht, wobei die Aktionserkennung auf den beiden in Kapitel 6.2.1 vorgestellten Datensätzen ausgewertet wird. Die Bewegungsvorhersage und alle damit kombinierten Vorhersagen werden dagegen auf dem durch das Anwendungsszenario motivierten CoAx-Datensatz (vgl. S. 96) ausgewertet. Nachfolgend werden die einzelnen Experimente kurz beschrieben. In Kapitel 6.3 wird anschließend die Auswertung dieser Experimente vorgestellt.

In den grundlegenden Experimenten wird zur in Kapitel 4.2 vorgestellten Architektur zusätzlich eine Übersicht über die untersuchten Ablationen gegeben. Dabei handelt es sich um die GNN-Varianten, die innerhalb der entwickelten Architektur untersuchten wurden.

Variante	**Besonderheit**
GCN	Symmetrische Normalisierung
GAT	Aufmerksamkeitsmechanismus
GATv2	Aufmerksamkeitsmechanismus
RGCN	Relationsbasiert
Graph Transformer	Transformer-basiert
GRNN	RNN-Mechanismus
ResGGCN	Gate-Mechanismus

Tabelle 6.3: Beschreibung der GNN-Varianten mit der angeführten Besonderheit relativ zum Basis-GNN (vgl. Kapitel 3.2.2), welche im Kapitel 3.2.3 vorgestellt wurden.

Reine Aktionserkennung

In diesem Experiment wird einerseits ein Vergleich mit der in [DWA20] vorgestellten Baseline-Methode durchgeführt, was implizit auch einem Vergleich zwischen der GNN-Methodik in [BHB+18] und der GNN-Methodik in [KW16a] entspricht. Zum anderen werden unterschiedliche GNN-Varianten (vgl. Tabelle 6.3) für die in dieser Arbeit vorgestellten Methode (vgl. Kapitel 4.2) untersucht.

Analog zur Evaluationstrategie von [DWA20] wird bei den Experimenten zur reinen Aktionserkennung für beide Datensätze eine *Leave-One-Out*-Kreuzvalidierungsstrategie (vgl. S. 99) herangezogen. Das Ergebnis für einen Kreuzvalidierungsdurchlauf ergibt sich aus dem Mittel aller Kreuzvalidierungsschritte. Dieser Vorgang wird mehrfach durchgeführt und erneut über die Ergebnisse gemittelt, um zufällig positive bzw. negative Ergebnisse zu vermeiden.

So ergibt sich prozentual eine Aufteilung für die beiden Datensätze wie in Tabelle 6.2 dargestellt.

Bewegungsvorhersage und Kombinationen

Die Bewegungsvorhersage und alle weiteren Kombinationen von Vorhersagen mit dieser, werden experimentell auf dem CoAx-Datensatz ausgewertet. Auch hierbei werden unterschiedliche GNN-Varianten innerhalb der vorgeschlagenen Methode gegenübergestellt. Anders als im vorigen Abschnitt für die Experimente zur reinen Aktionserkennung beschrieben, wird für die Bewegungsvorhersage und alle weiteren Untersuchungen eine einzige Aufteilung für das Training, die Validierung und den Test verwendet. Zum Vergleich werden die Varianten der erarbeiteten Methode und für die Bewegungsvorhersage zusätzlich die beiden Basismethoden herangezogen. Für die Trainingsdaten werden die Datenteilmengen $\mathcal{S}_1 - \mathcal{S}_4, \mathcal{S}_6$ des Datensatzes (Versuchspersonen $1 - 4, 6$ vgl. Kapitel 6.2.1) verwendet und zum Testen die Datenteilmenge $\mathcal{S}_5$. Zur Validierung wird nach wie vor aus dem Trainingsdatensatz eine Wiederholung zurückgehalten.

Für diese Reihe an Experimenten auf dem CoAx-Datensatz ergibt sich zusätzlich, dass am Anfang und am Ende jeder Videosequenz unvollständige Szenengraphen exkludiert werden. Effektiv werden also die ersten Szenengraphen verworfen, die nicht über die vollständige Anzahl historischer Daten verfügen, da es zu Beginn eines Videos noch keine historische Information gibt. Analog dazu besteht auch das Problem, dass für das Ende der Videosequenz, für die es zwar historische Information gibt, um einen Szenengraph zu konstruieren, jedoch keine vollständige Information über den korrespondierenden zukünftigen Szenengraphen. Die jeweils verbleibenden Datenpunkte für die Trainings-, Validierungs- und Testdaten sind in der Tabelle 6.4 vergleichend dargestellt.

CoAx	[%] Aufnahmen	# Aufnahmen	[%] Frames angepasst	
Training	75.00	135.00	~~74.52~~	70.84
Validierung	8.33	15.00	~~8.34~~	7.93
Test	16.67	30.00	~~17.14~~	16.33

Tabelle 6.4: Datensatzunterteilung: Analog zu Tabelle 6.2 für den CoAx-Datensatz, jedoch mit Abzug der nicht verwendbaren Datenpunkte aufgrund der Bewegungsvorhersage.

Mit der in Kapitel 4.2 und Kapitel 5 beschriebenen Architektur werden für die Bewegungsvorhersage vier weitere Experimente durchgeführt. Dabei handelt es sich um die reine Bewegungsvorhersage und der Kombinationen mit anderen Vorhersageaufgaben.

Bei der Evaluation der Bewegungsvorhersage werden immer die Ergebnisse der Baseline-Methoden (ZV und VBPP) zum direkten Vergleich angeführt.

Reine Bewegungsvorhersage Analog zur beschriebenen Bewegungsvorhersage in Kapitel 4.2.4 wird hier die Architektur zur *reinen* Bewegungsvorhersage als Lernaufgabe auf dem CoAx-Datensatz ausgewertet. Für einen Vorhersagehorizont von einer Sekunde wird vergleichend mit den Baseline-Methoden die Bewegungsvorhersage (vgl. S. 102) ausgewertet. Außerdem dient die reine Bewegungsvorhersage als weitere Grundlage für einen Vergleich mit den kombinierten Varianten der vorgestellten Methode. Dadurch kann der Einfluss vom kombinierten Lernen auf die Bewegungsvorhersage untersucht werden

Bewegungsvorhersage mit Wissen über erkannte Aktion In diesem Experiment wird die in Kapitel 5.1 beschriebene Architektur ausgewertet. Diese verwendet das Wissen der erkannten Aktion, welche durch ein trainiertes Modell für die Aktionserkennung bereitgestellt wird. Damit steht für die Bewegungsvorhersage ein Szenengraph zur Verfügung, der an jedem Knoten des Szenengraphen die erkannte Aktion als weiteres Merkmal enthält. Ausgewertet wird die Bewegungsvorhersage für einen Vorhersagehorizont über eine Sekunde und mit den Baseline-Methoden für die Bewegungsvorhersage verglichen. Das verwendete Modell für die Aktionserkennung entspricht dabei dem in Kapitel

6.2.3 für die Experimente zur Aktionserkennung herangezogenen und muss somit nicht erneut betrachtet werden. Es dient lediglich als weiteres Merkmal für das hier verwendete Modell. Damit ist auch ein Vergleich zur reinen Bewegungsvorhersage möglich, welche *aktionsagnostisch* ist.

Kombinierte Bewegungsvorhersage und Aktionserkennung Bei diesem Experiment handelt es sich um die in Kapitel 5.2 beschriebene Architektur zur kombinierten Bewegungsvorhersage und Aktionserkennung. Vergleichend zu den bisherigen Experimenten wird untersucht, ob das gemeinsame Lernen der Bewegungsvorhersage und der Aktionserkennung möglich ist und ob es in der Modellgüte eine Verbesserung gegenüber dem einzelnen Lernen pro Aufgabe oder der mehrstufigen Architektur (vgl. Experiment in vorigem Paragraph) gibt. Auch hier werden die Baseline-Methoden als Vergleich für die Bewegungsvorhersage hinzugezogen.

Kombinierte Bewegungsvorhersage, Aktionserkennung und statische Aktionsprädiktion Analog zum Experiment mit der kombinierten Architektur zur Bewegungsvorhersage und Aktionserkennung wird hier die Kombination um die Vorhersage der zukünftigen Aktion (in einer Sekunde) erweitert. Bei dieser Variante der kombinierten Architektur (vgl. Kapitel 5.2.1), ist zum einen die Machbarkeit von drei gleichzeitigen Vorhersagen von Interesse, zum anderen der Vergleich der Bewegungsvorhersage- und der Aktionserkennungsgüte mit den vorigen Varianten.

Kombinierte Bewegungsvorhersage, Aktionserkennung, dynamische Aktionsprädiktion und Aktionswechselzeitpunkt Abschließend wird eine kombinierte Architektur zur zusätzlichen Vorhersage der *nächsten* Aktion und des Zeitpunkts des Aktionswechsels beschrieben. Anders als in den bisherigen Varianten, wird hier der Effekt einer zusätzlichen, ereignisbasierten Vorhersage untersucht. Die Vorhersage der nächsten Aktion entspricht dabei einer *dynamischen* Vorhersage, da der Zeitraum variabel durch das Eintreten der nächsten Aktion gegeben ist (vgl. Kapitel 4.2.5). Zusätzlich wird die Güte der Aktionsvorhersage in einer Sekunde untersucht, welche aus der Vorhersage der nächsten Aktion abgeleitet werden kann.

Die Aktion in einer Sekunde entspricht dann der

$$\begin{cases} \text{Aktionserkennung}, & \text{falls die Zeit zur nächsten Aktion} > 1\text{s oder der} \\ \text{nächsten Aktion}, & \text{falls die Zeit zur nächsten Aktion} \leq 1\text{s ist.} \end{cases} \tag{6.3}$$

Dabei wird für die Entscheidung über die Zeit des Aktionswechsels der vorhergesagte Aktionswechselzeitpunkt verwendet. Somit ist die Ableitung der vorhergesagten Aktion in einer Sekunde vollständig unabhängig von Informationen, welche während der Inferenz nicht verfügbar sind.

Zusammenfassung Eine Übersicht über die beschriebenen Experimente sind in der nachfolgenden Abbildung 6.4 dargestellt.

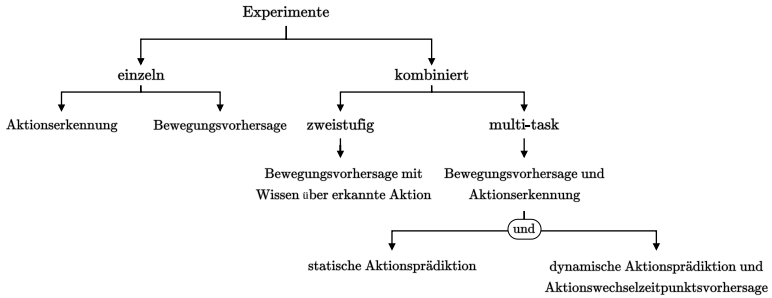


Abbildung 6.4: Übersicht der beschrieben Experimente, welche in Kapitel 6.3 ausgewertet werden.

Neben den hier beschriebenen Kombinationen können weitere Experimente mit der vorgestellten Architektur durchgeführt werden. Bei den Experimenten handelt es sich zum einen um grundlegende Untersuchungen und zum anderen um komplexer werdende Varianten der Architektur, die mehrere Vorhersagen auf Basis von Eingangsszenengraphen bereitstellen.

6.3 Auswertung der Experimente

In diesem Kapitel werden die durchgeführten Experimente, beschrieben in Kapitel 6.2.3, ausgewertet, dargestellt und interpretiert. Darüber hinaus wird ein qualitativer Überblick über die Ergebnisse der Experimente gegeben.

6.3.1 Ergebnisse

Zunächst werden die in Kapitel 6.2.3 beschriebenen Experimente quantitativ ausgewertet und dargestellt. Anschließend werden die Ergebnisse der Bewegungsvorhersage in diesem Kapitel ganzheitlich verglichen.

Für die Bewertung der Bewegungsvorhersage wird neben der quantitativen ADE-Metrik (vgl. Kapitel 4.2.4) zur Einordnung eine Fehlertoleranz eingeführt. Konkret wird die repräsentative Länge einer menschlichen Handfläche (11.1cm nach [WSH+18]) zusätzlich als Toleranz für eine akzeptable Bewegungsvorhersage der Handposition über den zeitlichen Verlauf einer Sekunde verwendet. Bei einem Vorhersagefehler im Toleranzbereich gilt diese Vorhersage als hinreichend gut.

Im Folgenden wird das F_1-Maß (engl. F_1-*score*) als Gütemaß für die Aktionsklassifikation verwendet. Ferner wird, sofern nicht anders spezifiziert, das Mikro-F_1-Maß als Korrektklassifikationsrate (engl. *accuracy*) genutzt, wobei *mikro* dabei das harmonische Mittel über alle Klassen hinweg für alle Vorhersagen bedeutet.

Reine Aktionserkennung

Die reine Aktionserkennung untersucht die Güte der vorgestellten Methode für die Aktionsklassifikation der aktuellen Aktion. Durch diese Untersuchung können grundlegende Erkenntnisse für ein direktes Erlernen der Fähigkeit, die aktuell vom Menschen ausgeführte Aktion zu klassifizieren, gewonnen werden.

Hyperparameteroptimierung Für die Aktionserkennung wurde die Hyperparameteroptimierung zur Modellauswahl auf beiden Datensätzen untersucht.

Dabei wurden für die verschiedenen GNN-Varianten (vgl. Tabelle 6.3) Hyperparameteroptimierungen für die jeweiligen Datensätze durchgeführt. Die besten Hyperparameter für jede Optimierung wurden anhand des Mikro-F_1-Maßes auf dem Validierungsdatensatz unter Durchführung einer sechsfach Kreuzvalidierungsstrategie (vgl. Kapitel 6.2.3) bestimmt. Anschließend wurde für die beste Hyperparameterkonfiguration jeder Optimierung ein unabhängiges Training auf den jeweiligen Datensätzen durchgeführt, welches dann durch die sechsfache Kreuzvalidierungsstrategie evaluiert wurde. Somit konnte für jede Hyperparameterkonfiguration aus jeder Optimierung eine Aussage über die Güte auf den beiden untersuchten Datensätzen (Bimacs- und CoAx-Datensatz) getroffen werden. Das finale Ergebnis für die jeweilige GNN-Variante ist dann durch die Hyperparameterkonfiguration gegeben, welche auf einem Datensatz optimiert und dann auf beiden Datensätzen trainiert und evaluiert wurde. Somit werden die Hyperparameter so gewählt, dass nicht für jeden Datensatz die optimalen Hyperparameter gesucht und verwendet werden, sondern nur auf einem Datensatz optimiert und daraufhin für beide untersuchten Datensätzen zur Evaluation verwendet werden. Dadurch wird die Fähigkeit des Modells zur Generalisierung bewertet.

Die Validierungsmodellgüte für die Hyperparameteroptimierungen auf dem CoAx-Datensatz ist in der Tabelle 6.5 für ausgewählte Hyperparametersuchen dargestellt.

GNN-Variante	$\boldsymbol{F_1}$**-Maß** [%] ↑
RGCN	94.87
GCN	93.54
Graph Transformer	88.14
GAT	82.57
GATv2	81.97

Tabelle 6.5: Darstellung der Validierungsmodellgüte für ausgewählte Hyperparametersuchen auf dem CoAx-Datensatz.

Anhand dieser Übersicht kann bereits beurteilt werden, welche GNN-Variante die erfolgversprechendste ist. Die Hyperparameterkonfiguration mit der besten

Validierungsmodellgüte wird dann als Modellkonfiguration verwendet und jeweils auf beiden Datensätzen trainiert und evaluiert.

Die beiden besten Hyperparameterkonfigurationen werden im nachfolgenden Paragraphen sowie in der Tabelle 6.6 anhand der resultierenden Modellgüte beschrieben. Tabelle 6.7 zeigt die Auswertung der besten Hyperparameterkonfiguration für die jeweilige GNN-Variante auf den Testdaten.

Hyperparameter Tabelle 6.6 zeigt die Hyperparameter-Konfigurationen der beiden besten Modelle: Die GNN-Variante *RGCN* wurde auf dem Bimacs-Datensatz und die GNN-Variante *GCN* auf dem CoAx-Datensatz durch jeweils eine Hyperparametersuche gefunden.

Hyperparameter	**RGCN**	**GCN**
# Parameter	3×10^6	9.7×10^5
Embedding-Dimension	352	416
# Blöcke	8	5
Dropout-GNN	0.119	0.2197
Bias-Term	true	true
Aggregation	mean	mean
Graph-Pooling	max	max
Pre-Aktivierung	false	false
Aktivierung	ReLU	ReLU
Residualschicht	true	true
Pre-Normalisierung	true	false
Normalisierung	Batch	Batch
Selbst-Schleifen	n.a.	true
Kantengewichte	n.a.	true
Kantenart	gerichtet	gerichtet
# MLP-Schichten	1	2
MLP-Dropout	0.0	0.3102
Lernrate	0.00071	0.0017
# Batchgröße	128	512
# Geduld	15	8

Tabelle 6.6: Beschreibung der zwei besten Hyperparameterkonfigurationen anhand wichtiger Parameter, jeweils mit Information zur Anzahl der Modellparameter.

Hervorzuheben ist zum einen, dass es sowohl mit als auch ohne Pre-Batch-Normalisierung die besten gefundenen Hyperparameter gibt. Zum anderen, dass die Anzahl der Parameter des RGCN-basierten Modells dreimal so groß ist wie die der GCN-Variante. Das RGCN-basierte Modell erzielt die besten Ergebnisse, das GCN-basierte Modell die zweitbesten. Die quantitativen Er-

gebnisse der trainierten Modelle auf dem CoAx- und Bimacs-Datensatz sind in den nachfolgenden Paragraphen in den Tabellen 6.7 und 6.8 beschrieben.

Quantitative Ergebnisse der Aktionserkennung auf dem Bimacs-Datensatz Die Ergebnisse in der Tabelle 6.7 zeigen die untersuchten GNN-Varianten der hier vorgestellten Methode zur Aktionserkennung auf dem Bimacs-Datensatz. Die vorgestellte Methode mit der H2O-Graph-Konfiguration (vgl. Abbildung 4.2) sowie der alternativen O2O-Graph-Konfiguration (vgl. Abbildung 4.4) für die GCN-basierte Variante und der Methode von [DWA20] wird dabei gegenübergestellt.

GNN-Variante	$\boldsymbol{F_1}$**-Maß** [%] ↑
RGCN	68.3 ± 0.6
GCN	65.6 ± 0.4
Graph Transformer	53.9 ± 0.6
GAT	62.7 ± 0.9
GATv2	56.7 ± 0.8
GCN-O2O	48.2 ± 0.4
GNN [DWA20]	64.0

Tabelle 6.7: Darstellung des gemittelten Mikro-F_1-Maßes der sechsfachen Kreuzvalidierungsstrategie auf den Testdaten des Bimacs-Datensatzes.

Dargestellt ist das jeweilige Mikro-F_1-Maß über alle Aktionsklassen, für die mit der vorgestellten Methode durchgeführten Experimente sowie der Fehlerbereich nach fünffacher Durchführung. Die Abweichung der Modellgüte ist für alle mit der vorgestellten Methode durchgeführten Experimente unter ±1%. Die RGCN-basierte Variante mit der H2O-Graph-Konfiguration erzielt dabei das beste und die O2O-Graph-Konfiguration das schlechteste Ergebnis.

Quantitative Ergebnisse der Aktionserkennung auf dem CoAx-Datensatz Die Ergebnisse der untersuchten GNN-Varianten der hier vorgestellten

Methode zur Aktionserkennung auf dem CoAx-Datensatz werden in der Tabelle 6.8 dargestellt, analog zu den Ergebnissen in Tabelle 6.7 und werden unter Anwendung der in [DWA20] beschriebenen Methode generiert.

GNN-Variante	$\boldsymbol{F_1}$**-Maß** [%] ↑
RGCN	90.2 ± 0.4
GCN	88.0 ± 2.3
Graph Transformer	84.9 ± 0.4
GAT	88.9 ± 0.3
GATv2	81.1 ± 0.5
GCN-O2O	87.8 ± 0.9
GNN [DWA20]	69.7

Tabelle 6.8: Darstellung des gemittelten Mikro-F_1-Maßes der sechsfachen Kreuzvalidierungsstrategie auf den Testdaten des CoAx-Datensatzes. Das Ergebnis mit der Methode von [DWA20] wurde dabei unter Anwendung Methode von [DWA20] erzeugt.

Auch hier ist das jeweilige Mikro-F_1-Maß über alle Aktionsklassen für die mit der vorgestellten Methode durchgeführten Experimente sowie der Fehlerbereich nach fünffacher Durchführung dargestellt. Die Abweichung der Modellgüte ist für alle mit der vorgestellten Methode durchgeführten Experimente unter ±1%, mit Ausnahme der GCN-basierten Variante. Die RGCN-basierte Variante mit der H2O-Graph-Konfiguration erzielt dabei erneut das beste und die O2O-Graph-Konfiguration das schlechteste Ergebnis.

Tabelle 6.9 zeigt exemplarisch das Klassifikationsergebnis für jede einzelne Aktionsklasse des besten Modells (RGCN-basiertes Modell mit H2O-Graph-Konfiguration dargestellt in Tabelle 6.8). Zusätzlich zur Korrektklassifikationsrate (Mikro-F_1-Maß) werden dabei die Metriken *Genauigkeit* (engl. *precision*) und *Trefferquote* (engl. *recall*) sowie die Anzahl der Datenpunkte (engl. *support*) für die jeweilige Aktionsklasse für einen der fünf Durchläufe der Auswertung dargestellt. Gerade weil Aktionsdatensätze oft Aktionen beinhalten, welche unterschiedlich lange ausgeführt werden, entsteht ein Ungleichgewicht. Dieses spiegelt sich im Support-Wert wider, worin die Anzahl der Testdaten pro Aktionsklasse über alle Kreuzvalidierungsschritte dargestellt ist.

Aktion	Genauigkeit ↑	Trefferquote ↑	F_1-Maß ↑	Support
approach	0.90	0.94	0.92	4175
grab_valve	0.81	0.91	0.86	1940
plug_valve	0.96	0.96	0.96	2162
grab_screwdriver	0.83	0.85	0.84	2670
screw_valve	0.99	0.97	0.98	22 194
release_screwdriver	0.78	0.93	0.85	2506
grab_hose	0.89	0.88	0.88	1088
plug_hose	0.98	0.93	0.95	5417
retreat	0.93	0.88	0.90	9745
join_valve	0.69	0.70	0.69	3313
grab_screws	0.89	0.80	0.84	2320
plug_screws	0.94	0.86	0.90	3599
place_valve	0.41	0.81	0.55	2059
grab_membrane	0.79	0.87	0.83	1407
place_membrane	0.92	0.73	0.81	7857
grab_capacitor	0.84	0.89	0.87	1817
wait_for_robot	0.96	0.86	0.91	8380
plug_capacitor	0.92	0.98	0.95	7269
grab_soldering_tin	0.84	0.89	0.86	2303
grab_soldering_iron	0.67	0.86	0.75	1886
solder_capacitor	0.96	0.97	0.97	8532
release_soldering_iron	0.90	0.80	0.84	2088
release_soldering_tin	0.95	0.97	0.96	1685
Mikro	0.90	0.90	0.90	106 412
Makro	0.86	0.88	0.86	106 412
Gewichtet	0.91	0.90	0.90	106 412

Tabelle 6.9: Darstellung der Klassifikationsübersicht des besten Modells (RGCN-basiertes Modell mit H2O-Graph-Konfiguration, dargestellt in Tabelle 6.8) für einen Evaluationsdurchlauf auf dem CoAx-Datensatz.

Fazit Die vorgestellte Methode kann erfolgreich zur Erkennung der aktuellen Aktion verwendet werden und zeigt eine Verbesserung der Klassifikationsgüte auf zwei Datensätzen. Die gefundene Hyperparameterkonfiguration wird unabhängig vom Datensatz zum Training des jeweiligen Modells verwendet. Die Evaluation zeigt beispielsweise für die beste Modellvariante (RGCN-basiert),

dass die Methode die Aufgabe der Aktionserkennung unabhängig vom Datensatz lernen kann.

Insgesamt deuten die Fehlerbereiche für die verschiedenen Modelle darauf hin, dass den Ergebnissen eine geringe Modellunsicherheit zugrunde liegt.

Reine Bewegungsvorhersage

Analog zur reinen Aktionserkennung, untersucht die reine Bewegungsvorhersage die Güte der vorgestellten Methode für die Bewegungsvorhersage für einen Zeithorizont von einer Sekunde. Durch diese Untersuchung können grundlegende Erkenntnisse für die Vorhersage der Bewegung des Menschen bzw. dessen Handbewegung gewonnen werden.

Hyperparameter Die Hyperparameter-Konfiguration des besten Modells mit der GNN-Variante *RGCN*, gefunden durch eine Hyperparametersuche auf dem CoAx-Datensatz, ist in der Tabelle 6.10 dargestellt.

Hyperparameter	**RGCN**
# Parameter	13.9×10^6
Embedding-Dimension	480
# Blöcke	30
Dropout-GNN	0.111
Bias-Term	true
Aggregation	mean
Graph-Pooling	n.a.
Pre-Aktivierung	true
Aktivierung	ReLU
Residualschicht	true
Pre-Normalisierung	true
Normalisierung	Layer
Selbst-Schleifen	n.a.
Kantengewichte	n.a.
Kantenart	gerichtet
# MLP-Schichten	1
MLP-Dropout	0.0
Lernrate	0.00005
# Batchgröße	256
# Geduld	15

Tabelle 6.10: Beschreibung der besten Hyperparameterkonfiguration der reinen Bewegungsvorhersage, anhand wichtiger Parameter mit Information zur Anzahl der Modellparameter.

Anders als bei den Modellen für die reine Aktionserkennung, ist hier beim *besten* Modell eine deutliche Zunahme der Anzahl der Blöcke ($\{5, 8\} \rightarrow 30$) und somit auch der insgesamten Anzahl der Modellparameter ($\{5 \times 10^5, 3 \times 10^6\} \rightarrow 13.9 \times 10^6$) festzustellen.

Quantitative Ergebnisse der Bewegungsvorhersage auf dem CoAx-Datensatz In der nachfolgenden Tabelle 6.11 sind die Ergebnisse des besten Modells und der Ablationen auf Grundlage der verschiedenen GNN-Varianten zusammen mit den Ergebnissen der Baseline-Methoden zur Bewegungsvorhersage (vgl. S. 102) dargelegt. Der dargestellte Fehler ist über einen Prädiktionshorizont von einer Sekunde abgebildet und beschreibt den ADE-Fehler (vgl. Kapitel 4.2.4) auf Basis der wahren und vorhergesagten 3D-Handposition in Zentimetern.

Zeit [ms]	67	133	200	267	333	400	467	533	600	667	733	800	867	933	1000
ZV	2.51	3.58	4.56	5.48	6.32	7.11	7.86	8.56	9.23	9.86	10.46	11.02	11.56	12.07	12.55
VBPP	2.68	4.01	5.40	6.82	8.25	9.69	11.13	12.56	13.98	15.38	16.76	18.13	19.49	20.85	22.20
RGCN	2.13	2.60	3.16	3.77	4.39	5.01	5.61	6.19	6.73	7.26	7.76	8.25	8.73	9.20	9.69
GCN	2.23	3.19	4.05	4.42	4.95	5.47	6.04	6.59	7.14	7.68	8.25	8.79	9.31	9.81	10.31
GAT	2.29	3.32	3.90	4.43	5.07	5.75	6.37	6.94	7.46	7.91	8.30	8.66	9.01	9.46	10.17
GRNN	2.14	2.77	3.53	4.32	5.05	5.82	6.51	7.15	7.77	8.51	9.24	10.03	10.82	11.59	12.30
ResGGCN	2.16	2.87	5.19	5.91	6.72	7.54	8.38	9.23	10.08	10.89	11.65	12.35	13.00	13.62	14.20

Tabelle 6.11: Quantitative Ergebnisse der Bewegungsvorhersage der Handposition. Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die ADE-Metrik (vgl. Gl. (4.3)) in Zentimetern.

Quantitative Ergebnisse der Bewegungsvorhersage aktionsspezifisch Die in der Tabelle 6.11 dargestellten Ergebnisse, sind in Tabelle 6.12 unterteilt in statische und dynamische Aktionen abgebildet. Damit soll eine gezielte Bewertung der Vorhersageergebnisse, anhand der charakteristischen Bewegungen des Menschen ermöglicht werden. Neben dem gemittelten ADE-Fehler im Verlauf einer Sekunde, ist der FDE-Fehler (vgl. 4.2.4) ebenfalls dargestellt.

Der ADE- und FDE-Fehler für die Menge „alle“ entspricht dabei dem gemittelten ADE-Fehler über einer Sekunde und der FDE-Fehler, dem ADE-Fehler bei einer Sekunde. Die Einteilung der Testdatenpunkte in die Untermengen $\{\text{statisch}, \text{dynamisch}\}$ ist nach der Aktion gewählt und soll primär eine Aussage über die Güte der Vorhersage von Testdatenpunkten (Szenengraphen, die den zeitliche Verlauf über eine Sekunde widerspiegeln) für statische und dynamische Bewegungen liefern.

Methode	mittlerer ADE			FDE		
	alle	dynamisch	statisch	alle	dynamisch	statisch
ZV	8.18	16.30	4.62	12.55	24.40	7.18
VBPP	12.49	24.13	7.42	22.20	41.99	13.23
RGCN	6.03	10.00	4.23	9.69	15.80	6.94
GCN	6.55	11.19	4.44	10.31	17.22	7.17
GAT	6.60	11.27	4.48	10.17	16.85	7.14
GRNN	7.17	12.77	4.63	12.30	22.42	7.72
ResGGCN	8.92	17.52	5.02	14.20	27.15	8.33

Tabelle 6.12: Quantitative Ergebnisse der Bewegungsvorhersage, aufgeteilt nach vorliegender Aktionscharakteristik (statisch oder dynamisch), welche auf eine limitierte bzw. moderate Bewegung der Hand des Menschen zurückgeht. Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die mittlere ADE-Metrik (vgl. Gl. (4.3)) im Verlauf einer Sekunde und die FDE-Metrik (vgl. Gl. (4.5)) in Zentimetern.

Fazit Die vorgestellte Methode kann erfolgreich zur Vorhersage der Handbewegung des Menschen in 3D verwendet werden und zeigt eine Verbesserung gegenüber den Baseline-Methoden. Außerdem wird ersichtlich, dass die GNN-Varianten, wie etwa die RGCN-, GCN-, und GAT-basierte Variante einen deutlich geringeren Prädiktionsfehler über den gesamten Prädiktionshorizont haben als die Baseline-Methoden. Der initiale Fehler bei der Prädiktion bei 67 Millisekunden ist zwar für alle GNN-Varianten im Bereich von 2 cm, nimmt jedoch für die beiden GNN-Varianten GRNN und ResGGCN bei zunehmendem Prädiktionshorizont stärker zu.

In der Tabelle 6.12 wird deutlich, für welche charakteristischen Bewegungen des Menschen (statisch oder dynamisch) die Baseline-Methoden sowie die hier vorgestellte Methode besonders gut geeignet sind. Die beiden Baseline-Methoden, die auf Annahmen bezüglich der Geschwindigkeit bei der Bewegungsvorhersage basieren, sind bei der Prädiktion (gemittelt über den Prädiktionshorizont von einer Sekunde) für statische Bewegungen deutlich besser als für dynamische Bewegungen geeignet. Der Fehler wird bei dynamischen Bewegungen wesentlich größer, da die Annahmen einer konstanten Geschwindigkeit sowie Richtung oder einer Geschwindigkeit von Null nicht mehr gültig sind. Die

GNN-basierten Varianten (besonders RGCN, GCN und GAT) zeigen sowohl für statische Bewegungen einen kleineren Fehler als die Baseline-Methoden, als auch für die dynamischen Bewegungen.

Die Ursache für den insgesamt geringen Prädiktionsfehler der Baseline-Methode ZV ergibt sich aus dem Verhältnis von statischen und dynamischen Bewegungen im CoAx-Datensatz, welches etwa 70:30 (statisch:dynamisch) beträgt.

Insgesamt zeigt sich, dass der Fehler für die vorgestellte GNN-basierte Methode mit voranschreitendem Prädiktionshorizont deutlich langsamer ansteigt als bei den Baseline-Methoden.

Bewegungsvorhersage mit Wissen über erkannte Aktion

Bei diesem Experiment handelt es sich um die erste in dieser Arbeit vorgestellte Art, die Aktions- und Bewegungsvorhersage gemeinsam zu kombinieren. Es handelt sich hierbei um die in Kapitel 5.1 präsentierte zweistufige Modellarchitektur, die eine Bewegungsvorhersage analog zum vorigen Experiment (vgl. S. 116) mit zusätzlicher Anreicherung über die erkannte Aktion, durchführt. Diese Aktion wird durch ein vortrainiertes Modell als weiteres Merkmal bereitgestellt. Das vortrainierte Modell entspricht der RGCN-basierten Variante aus dem Experiment zur reinen Aktionserkennung auf dem CoAx-Datensatz (vgl. S. 109) und wurde für die Verwendung mit den gleichen Trainingsdaten vortrainiert, um kein Vorwissen über den Testdatensatz zu haben. Untersucht wird hier ebenfalls die Güte bzw. der Prädiktionsfehler für einen Zeithorizont von einer Sekunde. Grundsätzlich ist erneut die Vorhersage der Bewegung des Menschen bzw. dessen Handbewegung von primärem Interesse.

Hyperparameter Die Hyperparameterkonfiguration des besten Modells ist grundsätzlich ähnlich wie die des besten Modells zur reinen Bewegungsvorhersage (vgl. Tabelle 6.10). Hervorzuheben ist jedoch die verwendete Anzahl an GNN-Blöcken und die damit nochmals gestiegene Gesamtanzahl an Modellparametern. Die Anzahl der Blöcke steigt um sechs ($30 \rightarrow 36$) und die insgesamte Anzahl der Modellparameter um etwa 7×10^6 ($13.9 \times 10^6 \rightarrow 21 \times 10^6$).

Hyperparameter	**RGCN**
# Parameter	21×10^6
Embedding-Dimension	432
# Blöcke	36
Dropout-GNN	0.116
Bias-Term	true
Aggregation	mean
Graph-Pooling	n.a.
Pre-Aktivierung	true
Aktivierung	ReLU
Residualschicht	true
Pre-Normalisierung	true
Normalisierung	Layer
Selbst-Schleifen	n.a.
Kantengewichte	n.a.
Kantenart	gerichtet
# MLP-Schichten	1
MLP-Dropout	0.0
Lernrate	0.00005
# Batchgröße	32
# Geduld	10

Tabelle 6.13: Beschreibung der besten Hyperparameterkonfiguration der Bewegungsvorhersage mit dem Wissen über die erkannte Aktion, anhand wichtiger Parameter mit Information zur Anzahl der Modellparameter.

Quantitative Ergebnisse der Bewegungsvorhersage auf dem CoAx-Datensatz Nachfolgend sind die Ergebnisse der Methode zur Bewegungsvorhersage mit dem Wissen über die erkannte Aktion, durch ein zweistufiges Modell in Tabelle 6.14 im direkten Vergleich mit den Baseline-Methoden dargestellt.

Zeit [ms]	67	133	200	267	333	400	467	533	600	667	733	800	867	933	1000
ZV	2.51	3.58	4.56	5.48	6.32	7.11	7.86	8.56	9.23	9.86	10.46	11.02	11.56	12.07	12.55
VBPP	2.68	4.01	5.40	6.82	8.25	9.69	11.13	12.56	13.98	15.38	16.76	18.13	19.49	20.85	22.20
RGCN	2.42	2.87	3.41	4.01	4.59	5.14	5.68	6.19	6.69	7.16	7.61	8.08	8.56	9.05	9.56
GCN	2.76	3.33	4.09	4.43	4.88	5.35	5.87	6.42	6.95	7.48	8.00	8.50	9.00	9.48	9.95
GAT	3.27	3.81	4.13	4.63	5.15	5.69	6.19	6.67	7.12	7.53	7.88	8.24	8.60	9.02	9.54
GRNN	2.92	3.50	3.82	4.35	4.89	5.47	6.05	6.61	7.12	7.63	8.12	8.62	9.15	9.70	10.27
ResGGCN	4.25	4.43	6.24	6.40	6.69	7.07	7.53	8.04	8.57	9.10	9.61	10.09	10.55	10.99	11.42

Tabelle 6.14: Quantitative Ergebnisse der Bewegungsvorhersage der Handposition. Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die ADE-Metrik (vgl. Gl. (4.3)) in Zentimetern.

Quantitative Ergebnisse der Bewegungsvorhersage aktionsspezifisch
Die aktionsspezifische Auswertung für dieses Experiment ist nachfolgend in der Tabelle 6.15 im Vergleich zu den Baseline-Methoden dargestellt.

Methode	mittlerer ADE			FDE		
	alle	dynamisch	statisch	alle	dynamisch	statisch
ZV	8.18	16.30	4.62	12.55	24.40	7.18
VBPP	12.49	24.13	7.42	22.20	41.99	13.23
RGCN	6.07	9.74	4.40	9.56	14.66	7.24
GCN	6.43	10.78	4.46	9.95	16.16	7.14
GAT	6.50	10.90	4.50	9.54	15.50	6.83
GRNN	6.55	10.88	4.58	10.27	16.90	7.27
ResGGCN	8.07	14.54	5.13	11.42	19.75	7.65

Tabelle 6.15: Quantitative Ergebnisse der Bewegungsvorhersage, aufgeteilt nach vorliegender Aktionscharakteristik (statisch oder dynamisch). Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die mittlere ADE-Metrik (vgl. Gl. (4.3)) im Verlauf einer Sekunde und die FDE-Metrik (vgl. Gl. (4.5)) in Zentimetern.

Fazit Ähnlich wie bei der reinen Bewegungsvorhersage führen die Varianten der hier vorgestellten Methode zu einem geringeren Prädiktionsfehler als die

Baseline-Methoden. Dies gilt auch für die aktionsabhängige Untersuchung der Ergebnisse, unterteilt in statisch und dynamisch.

Bewegungsvorhersage und Aktionserkennung

In diesem Experiment wird die kombinierte Aktionserkennung und Bewegungsvorhersage untersucht. Es handelt es sich dabei um die Kombination aus einem gemeinsamen GNN-basierten Encoder (vgl. Kapitel 5.2), der für die Erkennung der aktuellen Aktion und zur Vorhersage der Handbewegung über einen Prädiktionshorizont von einer Sekunde verwendet wird.

Es werden im Folgenden die Ergebnisse der Bewegungsvorhersage über den Prädiktionshorizont von einer Sekunde sowie die aktionsabhängige Unterteilung der Evaluation beschrieben. Zusätzlich wird das Ergebnis der Aktionserkennung dargestellt, da bei dieser Art der Kombination über einen gemeinsamen Encoder die Aktionserkennung als weitere Ausgabe des Modells zur Verfügung steht.

Die beste Hyperparameterkonfiguration entspricht der des zweistufigen Modells (vgl. S. 120, sowie Tabelle 6.13) für die Bewegungsvorhersage und wird daher nicht erneut aufgeführt.

Die Kombination der beiden Verlustfunktionen für die Bewegungsvorhersage $\mathcal{L}_{\text{MF}}$ und die Aktionserkennung $\mathcal{L}_{\text{AR}}$ wird mit $w_{\text{MF}} = 0.8$, $w_{\text{AR}} = 0.2$ (vgl. Gl. (5.3)) gewichtet.

Quantitative Ergebnisse der Bewegungsvorhersage auf dem CoAx-Datensatz Tabelle 6.16 zeigt die quantitativen Ergebnisse der Bewegungsvorhersage für die kombinierte Variante mit Aktionserkennung.

Zeit [ms]	67	133	200	267	333	400	467	533	600	667	733	800	867	933	1000
ZV	2.51	3.58	4.56	5.48	6.32	7.11	7.86	8.56	9.23	9.86	10.46	11.02	11.56	12.07	12.55
VBPP	2.68	4.01	5.40	6.82	8.25	9.69	11.13	12.56	13.98	15.38	16.76	18.13	19.49	20.85	22.20
RGCN	2.21	2.65	3.21	3.83	4.41	4.98	5.54	6.11	6.67	7.22	7.75	8.26	8.76	9.24	9.70

Tabelle 6.16: Quantitative Ergebnisse der Bewegungsvorhersage der Handposition. Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die ADE-Metrik (vgl. Gl. (4.3)) in Zentimetern.

Quantitative Ergebnisse der Bewegungsvorhersage aktionsspezifisch
Die aktionsspezifische Evaluation ist in Tabelle 6.17 dargestellt.

Methode	mittlerer ADE			FDE		
	alle	dynamisch	statisch	alle	dynamisch	statisch
ZV	8.18	16.30	4.62	12.55	24.40	7.18
VBPP	12.49	24.13	7.42	22.20	41.99	13.23
RGCN	6.04	9.97	4.25	9.70	15.35	7.14

Tabelle 6.17: Quantitative Ergebnisse der Bewegungsvorhersage, aufgeteilt nach vorliegender Aktionscharakteristik (statisch oder dynamisch). Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die mittlere ADE-Metrik (vgl. Gl. (4.3)) im Verlauf einer Sekunde und die FDE-Metrik (vgl. Gl. (4.5)) in Zentimetern.

Quantitative Ergebnisse der Aktionserkennung In Tabelle 6.18 ist das Ergebnis der Aktionserkennung auf dem CoAx-Datensatz, mit der Teilmenge $\mathcal{S}_5$ als Testdaten, dargestellt. Beschrieben wird die Modellgüte durch das Mikro-, Makro- und gewichtete F_1-Maß, wobei das Mikro-F_1-Maß als die repräsentative Modellgüte interpretiert wird.

Methode	AR		
	Mikro	Makro	Gewichtet
RGCN	0.91	0.87	0.91

Tabelle 6.18: Quantitatives Ergebnis der Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage und Aktionserkennung mit der GNN-Variante *RGCN*. Die Modellgüte ist durch das Mikro-, Makro- und gewichtete F_1-Maß dargestellt.

Die dazugehörige normalisierte Konfusionsmatrix für die Aktionsklassen aus der Tabelle 6.18 ist in der Abbildung 6.5 dargestellt.

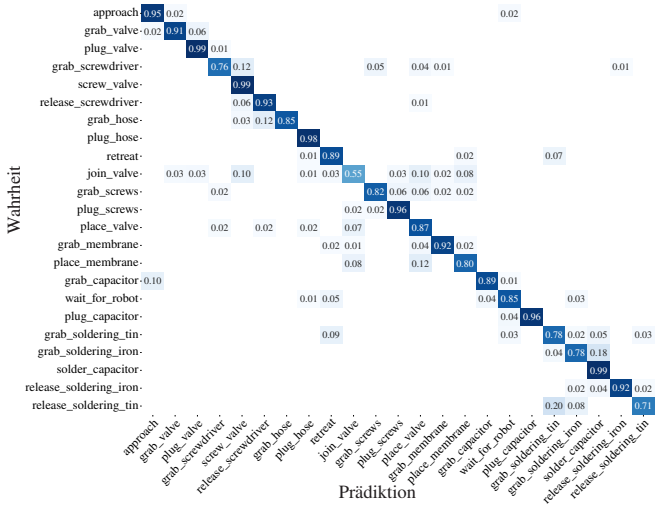


Abbildung 6.5: Exemplarische Darstellung der normalisierten Konfusionsmatrix zur Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage und Aktionserkennung mit der GNN-Variante *RGCN*. Auf der vertikalen Achse ist die tatsächliche Aktionsklasse und auf der horizontalen Achse die vorhergesagte Aktionsklasse aufgetragen.

Insgesamt wird der Großteil der Aktionsklassen mit einer Korrektklassifikationsrate von über 85% erkannt. Bei der Aktionsklasse „join_valve“ wird jedoch auch ersichtlich, dass in jeweils 10% der Testdatenpunkte für die Aktion

„screw_valve“ bzw. „place_valve“, eine irrtümliche Vorhersage getroffen wird. Diese Art von Fehlern kann dadurch bedingt sein, dass die Szene, einschließlich aller Objekte und dem Menschen, sowie deren Abstände und Relationen zueinander, sehr ähnlich sind und eine irrtümliche Vorhersage wahrscheinlich ist.

Fazit Auch bei der Kombination von Bewegungsvorhersage und Aktionserkennung durch einen gemeinsamen Encoder und einer gewichteten Verlustkombination, bleibt die bisherige Güte für die Bewegungsvorhersage und die Aktionserkennung erhalten. Die Ergebnisse der Bewegungsvorhersage sind besser als die der Baseline-Methoden und auch bei der Aktionserkennung liegt die Korrektklassifikationsrate bei über 90%. Der insgesamte Vergleich aller Varianten der vorgestellten Methode für die Bewegungsvorhersage wird auf S. 142 ganzheitlich gegenübergestellt.

Kombinierte Bewegungsvorhersage, Aktionserkennung und statische Aktionsprädiktion

Analog zum vorherigen Experiment (vgl. S. 123) wird in diesem Experiment zusätzlich die Aktionsprädiktion in Kombination mit der Aktionserkennung und Bewegungsvorhersage untersucht. Die Kombination geht auf die beschriebene Methode durch einen gemeinsamen GNN-basierten Encoder (vgl. Kapitel 5.2) zurück.

Es werden im Folgenden die Ergebnisse der Bewegungsvorhersage über den Prädiktionshorizont von einer Sekunde sowie der aktionsabhängigen Unterteilung der Evaluation beschrieben. Zusätzlich wird das Ergebnis der Aktionserkennung sowie die statische Aktionsprädiktion, also die Klassifikation der Aktion in einer Sekunde, dargestellt.

Im Folgenden wird auf die detaillierte Beschreibung der jeweiligen Hyperparameterkonfiguration verzichtet, da die wichtigsten Parameter weitgehend unverändert sind.

Die Gewichtung der Verlustfunktionen ist ähnlich der Kombination von Bewegungsvorhersage und Aktionserkennung (vgl. S. 123), sodass die Verlustfunktion der Bewegungsvorhersage deutlich stärker gewichtet wird als die der Klassifi-

kationen (Aktionserkennung und Aktionsprädiktion). Die Wahl der Gewichte[1] mit $w_{\mathrm{MF}} = 0.8$, $w_{\mathrm{AR}} = 0.06$ und $w_{\mathrm{AP}} = 0.14$ für die Kombination der Verlustterme der Bewegungsvorhersage $\mathcal{L}_{\mathrm{MF}}$, der Aktionserkennung $\mathcal{L}_{\mathrm{AR}}$ und der Aktionsprädiktion $\mathcal{L}_{\mathrm{AP}}$ geht dabei auf die Arbeit [LSS+23b] zurück.

Quantitative Ergebnisse der Bewegungsvorhersage auf dem CoAx-Datensatz In Tabelle 6.19 sind die Ergebnisse der Experimente für die Kombination von Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion durch einen gemeinsamen Encoder und einer kombinierten Verlustfunktion dargestellt.

Bei diesem Experiment wurden verschiedene GNN-Varianten, wie auch im Experiment mit dem zweistufigen Modell (vgl. S. 120), untersucht.

Zeit [ms]	67	133	200	267	333	400	467	533	600	667	733	800	867	933	1000
ZV	2.51	3.58	4.56	5.48	6.32	7.11	7.86	8.56	9.23	9.86	10.46	11.02	11.56	12.07	12.55
VBPP	2.68	4.01	5.40	6.82	8.25	9.69	11.13	12.56	13.98	15.38	16.76	18.13	19.49	20.85	22.20
RGCN	2.17	2.71	3.31	3.99	4.64	5.27	5.87	6.43	6.95	7.44	7.90	8.36	8.81	9.28	9.78
GCN	2.49	3.38	4.19	4.65	5.12	5.73	6.32	6.95	7.60	8.21	8.75	9.26	9.75	10.25	10.77
GAT	2.34	3.37	3.84	4.57	5.25	5.86	6.48	7.11	7.73	8.28	8.76	9.20	9.65	10.14	10.72
GRNN	2.23	2.89	3.67	4.47	5.24	6.00	6.86	7.87	8.39	9.00	9.70	10.44	11.16	11.81	12.44
ResGGCN	2.15	2.81	5.57	6.05	6.70	7.41	8.19	9.01	9.85	10.66	11.43	12.14	12.81	13.43	14.02

Tabelle 6.19: Quantitative Ergebnisse der Bewegungsvorhersage der Handposition. Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die ADE-Metrik (vgl. Gl. (4.3)) in Zentimetern.

Analog zu den vorigen Experimenten sind die untersuchten GNN-Varianten der vorgestellten Methode besser (bis auf ResGGCN) als die Baseline-Methoden, wobei die RGCN-, GCN-, und GAT-Variante die besten Ergebnisse bezüglich des Vorhersagefehlers (ADE) erzielen.

[1] Ausgedrückt durch die eingeführte Formulierung des allgemeinen Falls für eine additive Kombination durch Gewichtsfaktoren (vgl. Gl. (5.3)).

Quantitative Ergebnisse der Bewegungsvorhersage aktionsspezifisch Die aktionsspezifische Ergebnisübersicht der Evaluation, beschrieben in Tabelle 6.20, stellt dar, wie geeignet die GNN-Varianten für dynamische bzw. statische Bewegungen sind und wie diese im Vergleich dazu zu den Baseline-Methoden stehen. Die ResGGCN-Variante ist dabei die schlechteste von den hier untersuchten GNN-Varianten.

Methode	mittlerer ADE			FDE		
	alle	dynamisch	statisch	alle	dynamisch	statisch
ZV	8.18	16.30	4.62	12.55	24.40	7.18
VBPP	12.49	24.13	7.42	22.20	41.99	13.23
RGCN	6.19	10.47	4.26	9.78	16.09	6.92
GCN	6.89	11.82	4.66	10.77	18.01	7.24
GAT	6.89	11.52	4.78	10.72	17.38	7.70
GRNN	7.48	13.53	4.73	12.44	22.61	7.82
ResGGCN	8.82	17.14	5.04	14.02	26.94	8.16

Tabelle 6.20: Quantitative Ergebnisse der Bewegungsvorhersage, aufgeteilt nach vorliegender Aktionscharakteristik (statisch oder dynamisch). Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die mittlere ADE-Metrik (vgl. Gl. (4.3)) im Verlauf einer Sekunde und die FDE-Metrik (vgl. Gl. (4.5)) in Zentimetern.

Quantitative Ergebnisse der Aktionserkennung Die Ergebnisse der Aktionserkennung aller GNN-Varianten sind in der Tabelle 6.21 dargestellt. Des Weiteren sind in den Abbildungen 6.6 und 6.7 die Konfusionsmatrizen der beiden GNN-Varianten (RGCN und GAT) mit der höchsten Korrektklassifikationsrate dargestellt.

Methode	AR		
	Mikro	Makro	Gewichtet
RGCN	0.91	0.87	0.91
GCN	0.90	0.86	0.90
GAT	0.91	0.86	0.91
GRNN	0.90	0.85	0.90
ResGGCN	0.89	0.84	0.89

Tabelle 6.21: Quantitatives Ergebnis der Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion mit den untersuchten GNN-Varianten. Die Modellgüte ist durch das Mikro-, das Makro- und das gewichtete F_1-Maß dargestellt.

Die Modellgüte unterscheidet sich nur minimal bezüglich des Mikro- und des gewichteten F_1-Maßes. Das Makro-F_1-Maß ist bei den Varianten GRNN und ResGGCN am schlechtesten.

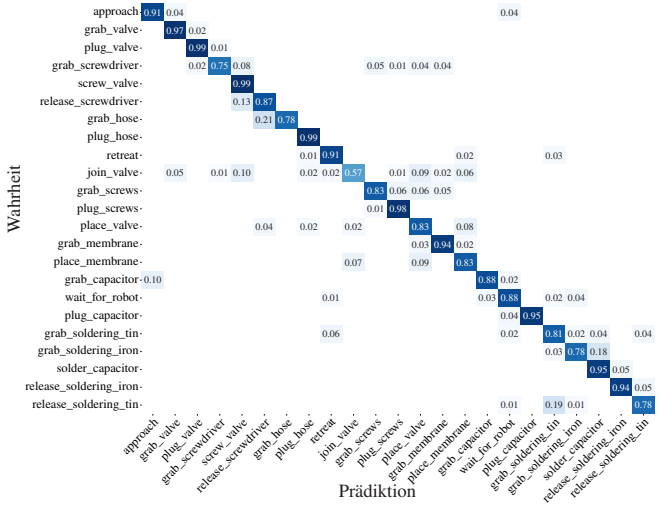


Abbildung 6.6: Exemplarische Darstellung der normalisierten Konfusionsmatrix der Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion mit der GNN-Variante *RGCN*.

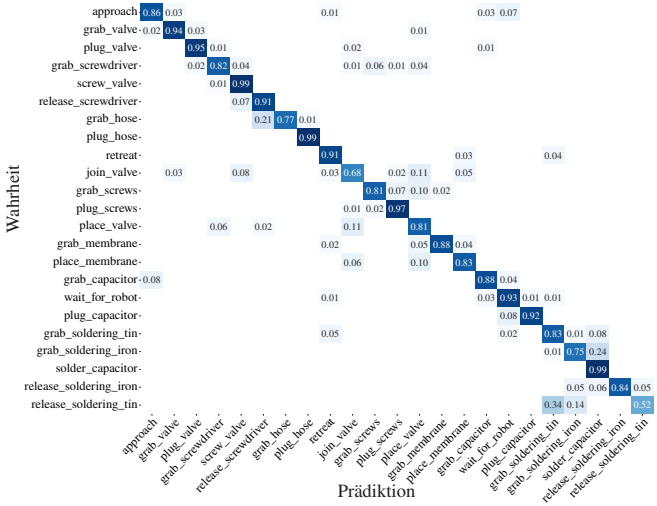


Abbildung 6.7: Exemplarische Darstellung der normalisierten Konfusionsmatrix der Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion mit der GNN-Variante *GAT*.

Quantitative Ergebnisse der Aktionsprädiktion (statisch) Die Ergebnisse der statischen Aktionsprädiktion aller GNN-Varianten sind in der Tabelle 6.22 aufgeführt. Des Weiteren sind analog zu den besten Modellvarianten aus der Aktionserkennung (vgl. Abbildungen 6.6 und 6.7) die Konfusionsmatrizen der GNN-Varianten RGCN und GAT für die Aktionsprädiktion in den Abbildungen 6.8 und 6.9 dargestellt.

Methode	AP		
	Mikro	Makro	Gewichtet
RGCN	0.80	0.67	0.79
GCN	0.81	0.66	0.79
GAT	0.79	0.63	0.78
GRNN	0.81	0.66	0.80
ResGGCN	0.79	0.63	0.78

Tabelle 6.22: Quantitatives Ergebnis der statischen Aktionsprädiktion des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion mit den untersuchten GNN-Varianten. Die Modellgüte ist durch das Mikro-, das Makro- und das gewichtete F_1-Maß dargestellt.

Die Korrektklassifikationsrate der Aktionsprädiktion ist bei allen GNN-Varianten um etwa 10% schlechter als die der Aktionserkennung. Wie in den Konfusionsmatrizen (Abbildungen 6.8 und 6.9) für die Aktionsprädiktion ersichtlich, liegt dies zum einen daran, dass für die jeweils *erste* Aktion einer jeden Aufgabe, „approach“, das Modell daran scheitert, diese korrekt zu klassifizieren und zu prädizieren. Zum anderen hat das Modell offenbar ebenfalls Schwierigkeiten bei Aktionen, welche sequentiell zusammenhängen (vgl. Abbildung 6.2) jeweils rechtzeitig die nächste Aktion zu klassifizieren. Visuell kann dies auch in den dargestellten Abbildungen (vgl. S. 146) nachvollzogen werden, in denen die wahre Aktionsklasse gegenüber der prädizierten Aktionsklasse, exemplarisch für eine gesamte Aufnahme einer Aufgabe dargestellt ist.

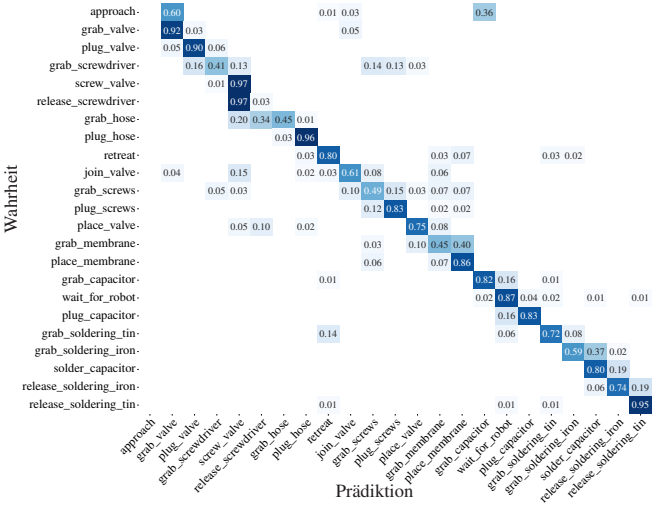


Abbildung 6.8: Exemplarische Darstellung der normalisierten Konfusionsmatrix, der statischen Aktionsprädiktion des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion mit der GNN-Variante *RGCN*.

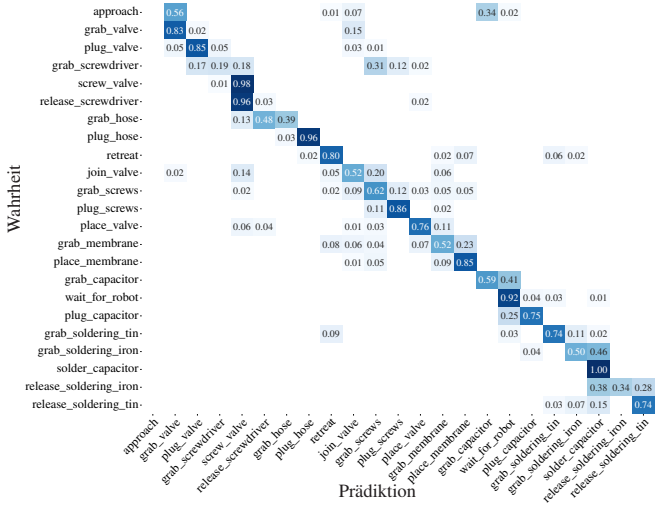


Abbildung 6.9: Exemplarische Darstellung der normalisierten Konfusionsmatrix, der statischen Aktionsprädiktion des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion mit der GNN-Variante *GAT*.

Fazit Analog zur Kombination von Bewegungsvorhersage und Aktionserkennung durch einen gemeinsamen Encoder und einer gewichteten Verlustkombination, ist die Erweiterung um die Kombination mit der statischen Aktionsprädiktion bezüglich der Güte von Bewegungsvorhersage und Aktionserkennung ohne signifikante Änderung möglich. Die Ergebnisse der Bewegungsvorhersage sind weiterhin besser als die der Baseline-Methoden und auch bei der Aktionserkennung ist die Korrektklassifikationsrate bei über 90%. Die Erweiterung um die statische Aktionsprädiktion liefert eine weitere Aussage, die aus ein und demselben trainierten Modell gewonnen werden kann. Diese ist zwar um etwa 10% schlechter als die Modellgüte bei der Aktionserkennung, kann aber als weitere Information verwendet werden, um während der Inferenz die Unsicherheit durch Plausibilitätsprüfungen zu verringern .

Kombinierte Bewegungsvorhersage, Aktionserkennung, dynamische Aktionsprädiktion und Aktionswechselzeitpunkt

Das abschließende Experiment beschäftigt sich analog zum vorigen Experiment (vgl. S. 126) nicht nur mit der Bewegungsvorhersage und Aktionserkennung, sondern auch mit der Vorhersage der zukünftigen Aktion. Wie bereits beschrieben, handelte es sich bisher um eine rein statische Aktionsprädiktion zu einem *festgelegten*, also statischen Prädiktionszeitpunkt, der analog zum Prädiktionshorizont für die Bewegungsvorhersage ist. Verschiedene Gründe, wie die unterschiedliche Dauer der vom Menschen ausgeführten Aktionen oder der häufige Wechsel von Aktionen, können zu einer Verschlechterung der Vorhersagegüte bei Aktionswechseln führen. Das Modell hat dabei insbesondere Schwierigkeiten, Aktionen von kurzer Dauer präzise zu erlernen. Aus diesem Grund wird in diesem Experiment anstelle eines statischen Zeitpunkts für die Aktionsprädiktion ein dynamischer Zeitpunkt verwendet, wodurch die zu klassifizierende Aktion der *nächsten* Aktion entspricht. Diese Art von Betrachtung kann auch als ereignisbasiert betrachtet werden. Zusätzlich wird die Kombination von Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion, um eine Vorhersage des Aktionswechselzeitpunkts erweitert. Als Anregung für diese Untersuchung wird auf die Arbeit von [WKS+21] verwiesen, in der im Kontext des automatisierten Fahrens zur Entscheidungsfindung die Zeitpunktsvorhersage beim Spurwechsel untersucht wird.

Auch für dieses Experiment wird auf die Beschreibung der Hyperparameterkonfiguration verzichtet, da nur eine GNN-Variante (RGCN) untersucht wurde, und für diese die bisher besten Parameter als Basis verwendet wurden. Auf Basis dieser Parameter wurden daraufhin die Gewichte für die Verlustkombination der vier Verlustfunktionen optimiert. Die Gewichte für die einzelnen Verlustfunktionen zeigt Tabelle 6.23.

Lernziel	Lernaufgabe	Verlustfunktion	Gewicht
Bewegungsvorhersage	Regression	ADE	1.84
Aktionserkennung	Klassifikation	Kreuzentropie	0.34
dyn. Aktionsprädiktion	Klassifikation	Kreuzentropie	0.92
Aktionswechselzeitpunkts-vorhersage	Regression	RMSE	0.30

Tabelle 6.23: Gewichte für die Kombination von vier Verlustfunktionen mit der zugehörigen Beschreibung der Lernaufgabe und der zugehörigen Verlustfunktion.

Bei den Gewichten handelt es sich hierbei um eine Kombination der Verlustfunktionen durch parametrisierbare Gewichte (vgl. Gl. (5.3)).

Quantitative Ergebnisse der Bewegungsvorhersage auf dem CoAx-Datensatz Das Ergebnis für die Bewegungsvorhersage mit der RGCN-Variante ist in der Tabelle 6.24, vergleichend zu den Ergebnissen der Baseline-Methoden dargestellt.

Zeit [ms]	67	133	200	267	333	400	467	533	600	667	733	800	867	933	1000
ZV	2.51	3.58	4.56	5.48	6.32	7.11	7.86	8.56	9.23	9.86	10.46	11.02	11.56	12.07	12.55
VBPP	2.68	4.01	5.40	6.82	8.25	9.69	11.13	12.56	13.98	15.38	16.76	18.13	19.49	20.85	22.20
RGCN	1.94	2.48	3.08	3.76	4.42	5.09	5.76	6.42	7.05	7.65	8.20	8.72	9.20	9.66	10.12

Tabelle 6.24: Quantitative Ergebnisse der Bewegungsvorhersage der Handposition. Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die ADE-Metrik (vgl. Gl. (4.3)) in Zentimetern.

Die Güte der Bewegungsvorhersage ist weiterhin konstant gut und besser als die der Baseline-Methoden, obwohl gleichzeitig vier Lernaufgaben durch einen geteilten Encoder gelernt werden.

Quantitative Ergebnisse der Bewegungsvorhersage aktionsspezifisch Die aktionsspezifische Darstellung der Evaluation der Bewegungsvorhersage wird in der Tabelle 6.25 dargestellt. Analog zu den bisherigen Untersuchungen mit der vorgestellten Methode wird eine ähnliche Güte, sowohl für statische, als auch für dynamische Bewegungen in der Vorhersage erzielt.

Methode	mittlerer ADE			FDE		
	alle	dynamisch	statisch	alle	dynamisch	statisch
ZV	8.18	16.30	4.62	12.55	24.40	7.18
VBPP	12.49	24.13	7.42	22.20	41.99	13.23
RGCN	6.24	10.02	4.50	10.12	15.42	7.72

Tabelle 6.25: Quantitative Ergebnisse der Bewegungsvorhersage, aufgeteilt nach vorliegender Aktionscharakteristik (statisch oder dynamisch). Vergleich der vorgestellten Methode mit Baseline-Methoden und Ablationen verschiedener GNN-Varianten mit dem CoAx-Datensatz. Der Vorhersagehorizont für die Bewegung entspricht einer Sekunde. Abgebildet wird die mittlere ADE-Metrik (vgl. Gl. (4.3)) im Verlauf einer Sekunde und die FDE-Metrik (vgl. Gl. (4.5)) in Zentimetern.

Quantitative Ergebnisse der Aktionserkennung Die quantitativen Ergebnisse für die Aktionserkennung sind in der Tabelle 6.26 und durch die Konfusionsmatrix in der Abbildung 6.11 dargestellt.

Methode	AR		
	Mikro	Makro	Gewichtet
RGCN	0.91	0.88	0.91

Tabelle 6.26: Quantitatives Ergebnis der Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion mit den untersuchten GNN-Varianten. Die Modellgüte ist durch das Mikro-, Makro- und gewichtete F_1-Maß dargestellt.

Analog zu den bisherigen Ergebnissen zur Aktionserkennung ist die Modellgüte im selben Bereich von etwa 91% (Mikro-F_1-Maß). Die dazugehörige norma-

lisierte Konfusionsmatrix (vgl. Abbildung 6.10) zeigt ebenfalls eine ähnliche Ausprägung für die einzelnen Klassen.

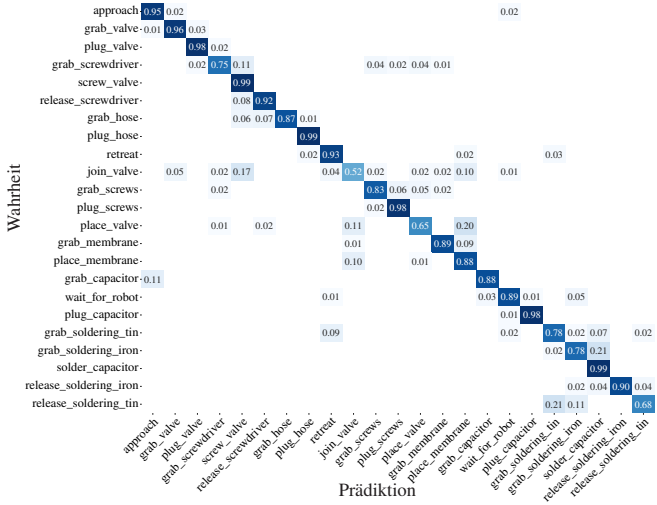


Abbildung 6.10: Exemplarische Darstellung der normalisierten Konfusionsmatrix der Aktionserkennung des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion mit der GNN-Variante *RGCN*.

Quantitative Ergebnisse der Aktionsprädiktion (dynamisch) Bei den quantitativen Ergebnissen der dynamischen Aktionsprädiktion, welche in der Tabelle 6.27 und durch die Konfusionsmatrix in der Abbildung 6.11 dargestellt sind, zeigt sich, anders als bei der statischen Aktionsprädiktion, eine höhere Modellgüte in der Klassifikation der zukünftigen Aktion. Die Modellgüte bei der Vorhersage der *nächsten* Aktion beträgt 91% und ist somit im selben Bereich wie die bisherigen Aktionserkennungsergebnisse und um etwa 10% besser als bei der statischen Aktionsprädiktion.

Methode	AP		
	Mikro	Makro	Gewichtet
RGCN	0.91	0.84	0.91

Tabelle 6.27: Quantitatives Ergebnis der dynamischen Aktionsprädiktion des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion mit den untersuchten GNN-Varianten. Die Modellgüte ist durch das Mikro-, Makro- und gewichtete F_1-Maß dargestellt.

Wie in der Abbildung 6.11 zu sehen ist, fehlt die Aktionsklasse *approach*, welche sowohl in den Konfusionsmatrizen für die Aktionserkennung, als auch in der statischen Aktionsprädiktion vertreten war. Der Grund dafür ist, dass bei der Vorhersage der nächsten Aktion per se, nie die Aktion *approach* als nächste Aktion und somit als Klassifikationsziel auftreten kann. Somit kann die Problematik, welche die bisherigen Modelle bei der statischen Aktionsprädiktion beim Lernen der Aktionsklasse *approach* hatten, nicht auftreten. Eine mögliche Ursache für die hohe Modellgüte kann dadurch begründet werden. Ein weiterer Aspekt ist die ereignisbasierte Natur beim Lernen der *nächsten* Aktion. Es ist nicht notwendig, sich bei der Vorhersage der Aktion um den Aktionswechsel zu kümmern, da für einen dynamischen Zeitpunkt ohne Vorwissen die nächste Aktion klassifiziert wird.

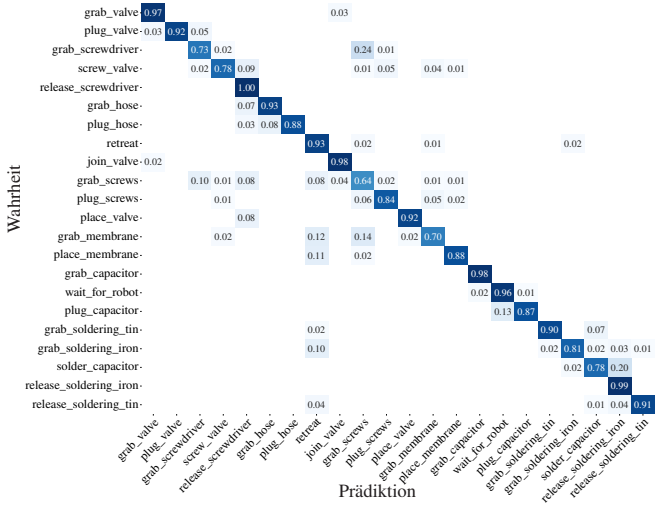


Abbildung 6.11: Exemplarische Darstellung der normalisierten Konfusionsmatrix der dynamischen Aktionsprädiktion des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion mit der GNN-Variante *RGCN*

Quantitative Ergebnisse der Aktionswechselzeitpunktsvorhersage In der Abbildung 6.12 ist das Ergebnis der Aktionswechselzeitpunktsvorhersage dargestellt. Dem gegenübergestellt ist der Fehler (RMSE) der Vorhersage, der Aktionswechselzeitpunkte für definierte Intervalle (engl. *bins*).

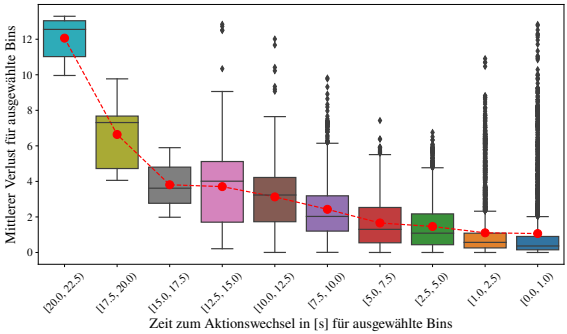


Abbildung 6.12: Gegenüberstellung der Fehler der Vorhersage, der Aktionswechselzeitpunkte und der in Bins eingeteilten Zeitpunkte zum Aktionswechsel.

Der Fehler über alle Bins beträgt im Mittel 2.3 Sekunden und der maximale Fehler 13.3 Sekunden, bei der Vorhersage der Aktionswechselzeitpunkte.

Quantitative Ergebnisse der Aktionsprädiktion (statisch, abgeleitet)
Eine weitere Aussage, die aus der Kombination der vier beschriebenen Vorhersagen abgeleitet werden kann, ist die Aktionsprädiktion für den statischen Prädiktionszeitpunkt von einer Sekunde (vgl. Gl. (6.3)). Dabei wird die Vorhersage über die aktuelle Aktion, die nächste Aktion und der Zeit bis zum Aktionswechsel herangezogen, um die Aktion zum Zeitpunkt in einer Sekunde zu bestimmen (vgl. Gl. (6.3)). Die abgeleiteten Ergebnisse für die Aktionsprädiktion sind in der Tabelle 6.28 und in der Konfusionsmatrix in der Abbildung 6.13 dargestellt.

Methode	AP		
	Mikro	Makro	Gewichtet
RGCN	0.79	0.64	0.78

Tabelle 6.28: Quantitatives Ergebnis der statischen Aktionsprädiktion (abgeleitet) des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion mit den untersuchten GNN-Varianten. Die Modellgüte ist durch das Mikro-, Makro- und gewichtete F_1-Maß dargestellt.

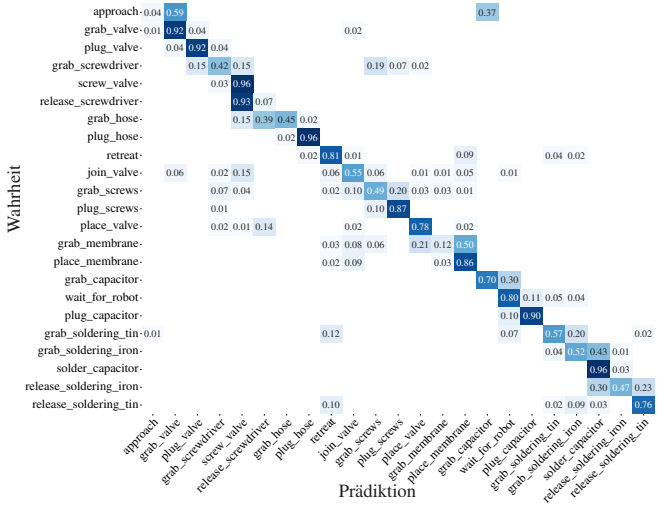


Abbildung 6.13: Exemplarische Darstellung der normalisierten Konfusionsmatrix, der statischen Aktionsprädiktion des Experiments mit der kombinierten Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsprädiktion mit der GNN-Variante *RGCN*.

Wie in der Gl. (6.3) beschrieben, hat dabei die Zeit bis zum tatsächlichen Aktionswechsel eine primäre Bedeutung für die erfolgreiche Ableitung der Aktion in einer Sekunde. So ist aktuell, wie in Abbildung 6.12 dargestellt, die Vorhersage des Aktionswechselzeitpunkts noch nicht ausreichend gut, um tatsächlich eine verbesserte Prädiktion der Aktion für einen Zeitpunkt, in einer Sekunde zu bestimmen. Somit wird lediglich eine Modellgüte von etwa 79% erreicht, was zumindest in etwa der Modellgüte der bisherigen Experimente zur statischen Aktionsprädiktion entspricht.

Fazit Analog zu den bisherigen Varianten der Kombination von Bewegungsvorhersage und verschiedenen Aktionsklassifikationen durch einen gemeinsamen Encoder und einer gewichteten Verlustkombination, ist die Erweiterung um die Kombination mit dynamischer Aktionsprädiktion und der Vorhersage der Aktionswechselzeitpunkte, bezüglich der Güte von Bewegungsvorhersage, Aktionserkennung und Aktionsprädiktion, ohne signifikante Änderung.

Zusätzlich kann bei dieser Kombination direkt die nächste Aktion vorhergesagt werden und in Kombination mit der Vorhersage des Zeitpunkts zum Aktionswechsel und der Aktionserkennung, die statische Aktionsprädiktion für den Zeitpunkt in einer Sekunde abgeleitet werden.

Bewegungsvorhersage gesamt

In diesem Abschnitt werden die Ergebnisse zur Bewegungsvorhersage auf dem CoAx-Datensatz aller bisher vorgestellten Experimente ganzheitlich verglichen. In der Tabelle 6.29 sind alle bisherigen Experimente zur Bewegungsvorhersage über den Prädiktionshorizont von einer Sekunde und in der Tabelle 6.30 aufgeteilt nach statischer und dynamischer Bewegung beschrieben. Dabei handelt es sich um die reine Bewegungsvorhersage (vgl. S. 116), der Bewegungsvorhersage mit Wissen über die erkannte Aktion (vgl. S. 120), der kombinierten Bewegungsvorhersage mit Aktionserkennung (vgl. S. 123), der kombinierten Bewegungsvorhersage mit Aktionserkennung und statischer Aktionsprädiktion (vgl. S. 126), sowie der kombinierten Bewegungsvorhersage mit Aktionserkennung und dynamischer Aktionsprädiktion, inklusive Aktionswechselzeitpunktsvorhersage (vgl. S. 134).

Zeit [ms]	67	133	200	267	333	400	467	533	600	667	733	800	867	933	1000
ZV	2.51	3.58	4.56	5.48	6.32	7.11	7.86	8.56	9.23	9.86	10.46	11.02	11.56	12.07	12.55
VBPP	2.68	4.01	5.40	6.82	8.25	9.69	11.13	12.56	13.98	15.38	16.76	18.13	19.49	20.85	22.20
RGCN*	2.13	2.60	3.16	3.77	4.39	5.01	5.61	6.19	6.73	7.26	7.76	8.25	8.73	9.20	9.69
GCN*	2.23	3.19	4.05	4.42	4.95	5.47	6.04	6.59	7.14	7.68	8.25	8.79	9.31	9.81	10.31
GAT*	2.29	3.32	3.90	4.43	5.07	5.75	6.37	6.94	7.46	7.91	8.30	8.66	9.01	9.46	10.17
GRNN*	2.14	2.77	3.53	4.32	5.05	5.82	6.51	7.15	7.77	8.51	9.24	10.03	10.82	11.59	12.30
ResGGCN*	2.16	2.87	5.19	5.91	6.72	7.54	8.38	9.23	10.08	10.89	11.65	12.35	13.00	13.62	14.20
RGCN†	2.42	2.87	3.41	4.01	4.59	5.14	5.68	6.19	6.69	7.16	7.61	8.08	8.56	9.05	9.56
GCN†	2.76	3.33	4.09	4.43	4.88	5.35	5.87	6.42	6.95	7.48	8.00	8.50	9.00	9.48	9.95
GAT†	3.27	3.81	4.13	4.63	5.15	5.69	6.19	6.67	7.12	7.53	7.88	8.24	8.60	9.02	9.54
GRNN†	2.92	3.50	3.82	4.35	4.89	5.47	6.05	6.61	7.12	7.63	8.12	8.62	9.15	9.70	10.27
ResGGCN†	4.25	4.43	6.24	6.40	6.69	7.07	7.53	8.04	8.57	9.10	9.61	10.09	10.55	10.99	11.42
RGCN‡	2.21	2.65	3.21	3.83	4.41	4.98	5.54	6.11	6.67	7.22	7.75	8.26	8.76	9.24	9.70
RGCN**	2.17	2.71	3.31	3.99	4.64	5.27	5.87	6.43	6.95	7.44	7.90	8.36	8.81	9.28	9.78
GCN**	2.49	3.38	4.19	4.65	5.12	5.73	6.32	6.95	7.60	8.21	8.75	9.26	9.75	10.25	10.77
GAT**	2.34	3.37	3.84	4.57	5.25	5.86	6.48	7.11	7.73	8.28	8.76	9.20	9.65	10.14	10.72
GRNN**	2.23	2.89	3.67	4.47	5.24	6.00	6.86	7.87	8.39	9.00	9.70	10.44	11.16	11.81	12.44
ResGGCN**	2.15	2.81	5.57	6.05	6.70	7.41	8.19	9.01	9.85	10.66	11.43	12.14	12.81	13.43	14.02
RGCN††	1.94	2.48	3.08	3.76	4.42	5.09	5.76	6.42	7.05	7.65	8.20	8.72	9.20	9.66	10.12

Tabelle 6.29: Quantitative Ergebnisse aller Varianten zur Bewegungsvorhersage.
* : reine Bewegungsvorhersage.
† : Bewegungsvorhersage mit Aktion als weiteres Merkmal im zweistufigen Modell.
‡ : kombinierte Architektur für Bewegungsvorhersage und Aktionserkennung.
** : kombinierte Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsvorhersage.
†† : kombinierte Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsvorhersage.

Die Tabelle 6.29 zeigt, dass alle durchgeführten Experimente mit der vorgestellten Methode, eine insgesamt bessere Modellgüte und einen geringeren ADE-Fehler über den Verlauf einer Sekunde bei der 3D-Bewegungsvorhersage der Handposition des Menschen erreichen. Bereits bei einem Prädiktionshorizont von über 300 ms zeigt sich der Vorteil der vorgestellten GNN-basierten Methode gegenüber den Baseline-Methoden. Der Prädiktionsfehler der Baseline-Methoden nimmt mit zunehmendem Prädiktionshorizont zu, da die Annahmen über die Geschwindigkeit der beiden Methoden nicht mehr eingehalten werden. Bei der GNN-basierten Methode ist der Prädiktionsfehler zwar im direkten Vergleich zur ZV-Methode (vgl. S. 102 bzw. Gl. (6.1)) relativ ähnlich, jedoch durchweg geringer und nimmt weniger stark zu als bei dieser.

Methode	mittlerer ADE			FDE		
	alle	dyn.	stat.	alle	dyn.	stat.
ZV	8.18	16.30	4.62	12.55	24.40	7.18
VBPP	12.49	24.13	7.42	22.20	41.99	13.23
RGCN*	6.03	10.00	4.23	9.69	15.80	6.94
GCN*	6.55	11.19	4.44	10.31	17.22	7.17
GAT*	6.60	11.27	4.48	10.17	16.85	7.14
GRNN*	7.17	12.77	4.63	12.30	22.42	7.72
ResGGCN*	8.92	17.52	5.02	14.20	27.15	8.33
RGCN†	6.07	9.74	4.40	9.56	14.66	7.24
GCN†	6.43	10.78	4.46	9.95	16.16	7.14
GAT†	6.50	10.90	4.50	9.54	15.50	6.83
GRNN†	6.55	10.88	4.58	10.27	16.90	7.27
ResGGCN†	8.07	14.54	5.13	11.42	19.75	7.65
RGCN‡	6.04	9.97	4.25	9.70	15.35	7.14
RGCN**	6.19	10.47	4.26	9.78	16.09	6.92
GCN**	6.89	11.82	4.66	10.77	18.01	7.24
GAT**	6.89	11.52	4.78	10.72	17.38	7.70
GRNN**	7.48	13.53	4.73	12.44	22.61	7.82
ResGGCN**	8.82	17.14	5.04	14.02	26.94	8.16
RGCN††	6.24	10.02	4.50	10.12	15.42	7.72

Tabelle 6.30: Quantitative Ergebnisse aller Varianten zur Bewegungsvorhersage, aktionsspezifisch.
* : reine Bewegungsvorhersage.
† : Bewegungsvorhersage mit Aktion als weiteres Merkmal im zweistufigen Modell.
‡ : kombinierte Architektur für Bewegungsvorhersage und Aktionserkennung.
** : kombinierte Architektur für Bewegungsvorhersage, Aktionserkennung und statischer Aktionsvorhersage.
†† : kombinierte Architektur für Bewegungsvorhersage, Aktionserkennung und dynamischer Aktionsvorhersage.

Weiterhin kann genauer durch die Aufteilung der Daten in ihre charakteristische, zugrundeliegende Bewegung (statisch oder dynamisch) gesehen werden, dass gerade bei den dynamischen Bewegungen der Prädiktionsfehler der Modelle mit den besten GNN-Varianten um bis zu 6.5 cm geringer ist, siehe Tabelle 6.30. Somit liegt gerade in den dynamischen Bewegungen das Potenzial dieser

GNN-basierten Methode. Außerdem zeigt sich ebenfalls, dass auch für statische Bewegungen der Prädiktionsfehler geringer ist, auch wenn dieser Unterschied nicht so deutlich ist wie bei den dynamischen Bewegungen.

6.3.2 Interpretation der quantitativen Ergebnisse

Insgesamt zeigen die quantitativen Ergebnisse sowohl für die Aktionsklassifikation (Erkennung und Vorhersage) als auch für die Bewegungsvorhersage, dass die vorgestellte Methode auf Basis von GNNs für den vorliegenden Anwendungsfall geeignet ist. Dabei zeigt sich zum einen, dass die Lernaufgaben Aktionsklassifikation und Bewegungsvorhersage mit der Art und Weise wie der Szenengraph modelliert wird (vgl. Kapitel 4.1) und wie die Modellarchitektur aufgebaut wird (vgl. Kapitel 4.2), einzeln erfolgreich sind. Zum anderen zeigt sich auch, dass sich die Modellgüte (Korrektklassifikationsrate und Regressionsfehler bei der Bewegungsvorhersage) durch den kombinierten Lernvorgang teilweise verbessert bzw. gleich bleibt und sich somit nicht verschlechtert, wobei gleichzeitig mehrere relevante Vorhersagen für den in dieser Arbeit vorliegenden Anwendungsfall erfolgreich bereitgestellt werden können.

Die beiden einfachen GNN-Varianten GCN und RGCN, welche in der Theorie auf die Arbeit von [KW16a] bzw. [SKB+18] (Erweiterung des GCNs zur Berücksichtigung von Relationen, Kantentypen zwischen Knoten) zurückgehen, erzielen durchweg die besten Ergebnisse.

Die GNN-Varianten GRNN und ResGGCN schneiden im Vergleich zu den untersuchten GNN-Varianten am schlechtesten ab. Diese Methoden sind repräsentativ zur Verwendung von RNN-basierten Methoden (GRNN) und Methoden mit reinem Gate-Mechanismus (ResGGCN) im Kontext von neuronalen Netzen.

6.3.3 Qualitative Ergebnisse

In diesem Kapitel werden für die Aktionsklassifikation und Bewegungsvorhersage exemplarisch qualitative Ergebnisse der vorgestellten GNN-basierten Methode dargestellt.

Aktionsklassifikation

Bei den qualitativen Ergebnissen der Experimente für die Aktionsklassifikation handelt es sich um die sequentiell geordnete Abfolge der prädizierten Aktionsklasse einer Aufnahme einer Aufgabe, welche der wahren Abfolge der Aktionsklassen gegenübergestellt wird. Für die Interpretation der qualitativen Ergebnisse der Aktionsklassifikation wird die zugehörige Farbkodierung für die vorhandenen Aktionen des CoAx-Datensatzes, welche in der Abbildung 6.14 dargestellt ist, benötigt.

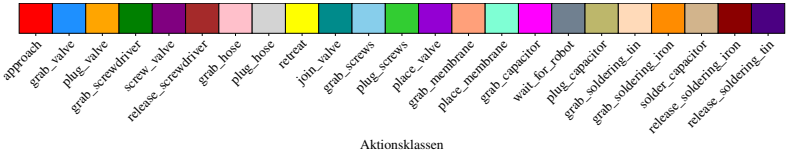


Abbildung 6.14: Übersicht der Aktionsklassen, welche im CoAx-Datensatz enthalten sind, und der zugehörigen Farbkodierung für die Interpretation der qualitativen Ergebnisse der Aktionsklassifikationsexperimente.

Mittels dieser Farbkodierung wird immer der Verlauf für eine gesamte Aufgabe dargestellt, wobei zuerst der Verlauf der prädizierten Aktionsklassen und anschließend die dazugehörigen, wahren Aktionsklassen abgebildet werden.

Exemplarisch werden diese qualitativen Ergebnisse der Aktionserkennung und die statische Aktionsprädiktion der Experimentenreihe (vgl. S. 126) sowie die qualitativen Ergebnisse der dynamischen Aktionsprädiktion (vgl. S. 134) dargestellt. Somit kann für die verschiedenen Lernaufgaben ein Eindruck über die qualitative Aussagekraft der Modelle für eine gesamte Aufnahme gewonnen werden. Es wird jeweils eine Aufnahme für jede Aufgabe aufgeführt, um die unterschiedlichen Aufgaben und ihre zugrundeliegende Aktionsabfolge (vgl. Abbildung 6.2) zu visualisieren.

Qualitative Ergebnisse der Aktionserkennung auf dem CoAx-Datensatz

Die nachfolgenden Abbildungen 6.15, 6.16 und 6.17 beziehen sich auf die Ergebnisse der Aktionserkennung für jeweils eine Aufnahme der drei Aufgaben,

der Testmenge $\mathcal{S}_5$ des CoAx-Datensatzes, mit der vorgestellten Methode auf S. 126.

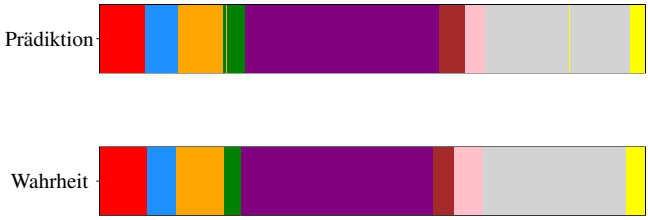


Abbildung 6.15: Qualitative Ergebnisse der Aktionserkennung einer Aufnahme der Aufgabe 1 aus dem Testdatensatz $\mathcal{S}_5$.

In der Abbildung 6.15 ist eine sehr gute Übereinstimmung der wahren und prädizierten Aktionsklassen einer Aufnahme der Aufgabe 1 ersichtlich. Teilweise zeigt sich, dass die Prädiktion bei der Erkennung einer neuen Aktion etwas verzögert ist. Dies wird gerade bei der *lila* bzw. *braun* eingefärbten Aktion (*screw_valve* bzw. *release_screwdriver*), bei welcher die korrespondierende wahre Aktionsklasse bereits gewechselt hat, die prädizierte Aktionsklasse jedoch nach wie vor die vorherige Aktionsklasse vorhersagt, sichtbar.

Abbildung 6.16: Qualitative Ergebnisse der Aktionserkennung einer Aufnahme der Aufgabe 2 aus dem Testdatensatz $\mathcal{S}_5$.

Ein ähnliches Verhalten zeigt sich in Abbildung 6.16, welche die qualitativen Ergebnisse einer Aufnahme der Aufgabe 2 darstellt. Teilweise ist auch zu beobachten, dass für kürzere Aktionssequenzen die falsche Aktionsklasse vorhergesagt wird. Während der Aktion *grab_screws* wird beispielsweise die Aktion *place_valve* erkannt, oder während der Aktion *place_membrane* die Aktion *join_valve*. Dies zeigt sich auch in den dargestellten Konfusionsmatrizen (vgl. Abbildungen 6.6 und 6.7), in denen für manche Aktionen oftmals die falsche Aktion prädiziert wird.

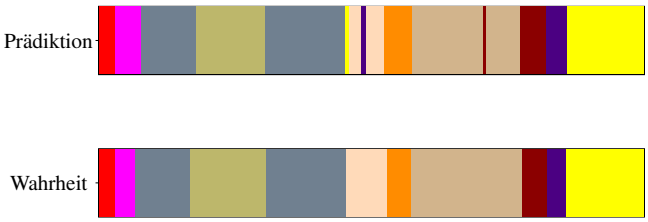


Abbildung 6.17: Qualitative Ergebnisse der Aktionserkennung einer Aufnahme der Aufgabe 3 aus dem Testdatensatz $\mathcal{S}_5$.

Die Abbildung 6.17 zeigt ein vergleichbares Verhalten des Modells bei der Vorhersage der Aktionsklassen wie zuvor beschrieben. Während der Aktion *grab_soldering_tin* wird kurzzeitig die Aktion *retreat* oder *release_soldering_tin* erkannt sowie für die Aktion *solder_capacitor* irrtümlicher Weise die Aktion *release_soldering_iron*.

Qualitative Ergebnisse der statischen Aktionsprädiktion auf dem CoAx-Datensatz Die Ergebnisse der statischen Aktionsprädiktion sind in den Abbildungen 6.18, 6.19 und 6.20 dargestellt und beziehen sich auf die Ergebnisse der statischen Aktionsprädiktion für jeweils eine Aufnahme der drei Aufgaben, der Testmenge $\mathcal{S}_5$ des CoAx-Datensatzes mit der vorgestellten Methode auf S. 126.

Abbildung 6.18: Qualitative Ergebnisse der statischen Aktionsprädiktion einer Aufnahme der Aufgabe 1 aus dem Testdatensatz $\mathcal{S}_5$.

Für die statische Aktionsprädiktion wird die zukünftige Aktion, welche in einer Sekunde stattfindet, als wahre Aktion verwendet. Somit entspricht beispielsweise die dargestellte *Wahrheit* in Abbildung 6.18, der Wahrheit von Abbildung 6.15, jedoch mit einer Verschiebung um eine Sekunde. Dies ist sehr gut an der ersten Aktion zu erkennen, die in *rot* eingefärbt ist und um eine Sekunde oder um 15 Zeitschritte (bei einer FPS-Rate von 15) nach links verschoben ist. Das Vorhersageverhalten bei der Aktionsprädiktion ähnelt dem der Aktionserkennung. Es scheint jedoch, dass das Modell den Aktionswechsel verzögert vorhersagt.

Besonders auffällig ist jedoch, dass das Modell bei der Vorhersage der ersten Aktion (*approach*, vgl. Abbildung 6.2) in allen drei Aufgaben scheitert. Dies wurde bereits in der quantitativen Beschreibung der Experimente bei der Diskussion der Konfusionsmatrizen angemerkt (vgl. beispielsweise Abbildung 6.8). Die Dauer der Aktion *approach* ist in den Aufzeichnungen des CoAx-Datensatzes relativ kurz (oftmals nur wenige Sekunden), was zur Folge hat, dass sich Trainingsdatenpunkte mit dem Aktionsklassenkennzeichen „approach“ markant reduzieren. Dies ist eine mögliche Erklärung für das Scheitern des Modells in der Vorhersage dieser Aktion.

Abbildung 6.19: Qualitative Ergebnisse der statischen Aktionsprädiktion einer Aufnahme der Aufgabe 2 aus dem Testdatensatz $\mathcal{S}_5$.

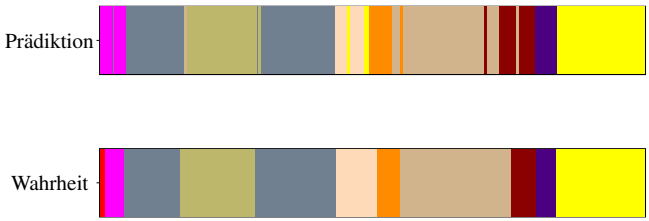


Abbildung 6.20: Qualitative Ergebnisse der statischen Aktionsprädiktion einer Aufnahme der Aufgabe 3 aus dem Testdatensatz $\mathcal{S}_5$.

In den Abbildungen 6.19 und 6.20 wird ein vergleichbares Verhalten des Modells bei der Vorhersage der Aktionsklassen, wie in Abbildung 6.18, gezeigt.

Qualitative Ergebnisse der dynamischen Aktionsprädiktion auf dem CoAx-Datensatz In den Abbildungen 6.21, 6.22 und 6.23 die Ergebnisse der dynamischen Aktionsprädiktion für jeweils eine Aufnahme der drei Aufgaben, der Testmenge $\mathcal{S}_5$ des CoAx-Datensatzes, mit der vorgestellten Methode auf S. 134 dargestellt.

Besonders interessant ist, dass hierbei nun der zeitliche Aspekt für die *Prädiktion* der zukünftigen Aktion zu vernachlässigen ist, da dieser entkoppelt

wurde. Es wird zeitunabhängig die *nächste* Aktion vorhergesagt. Dadurch kann auch die eben thematisierte Problematik mit der mangelnden Anzahl an Trainingsdatenpunkten für die immer erste Aktion *approach* vernachlässigt werden, da trivialerweise die erste Aktion ohne erneutes Ausführen, nie die nächste Aktion sein kann.

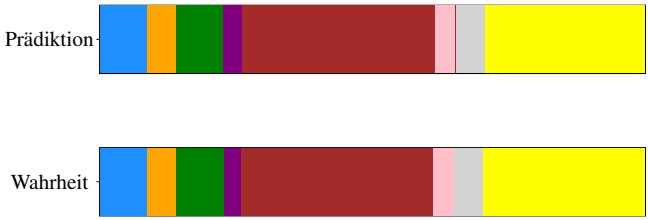


Abbildung 6.21: Qualitative Ergebnisse der dynamischen Aktionsprädiktion einer Aufnahme der Aufgabe 1 aus dem Testdatensatz $\mathcal{S}_5$.

Abbildung 6.22: Qualitative Ergebnisse der dynamischen Aktionsprädiktion einer Aufnahme der Aufgabe 2 aus dem Testdatensatz $\mathcal{S}_5$.

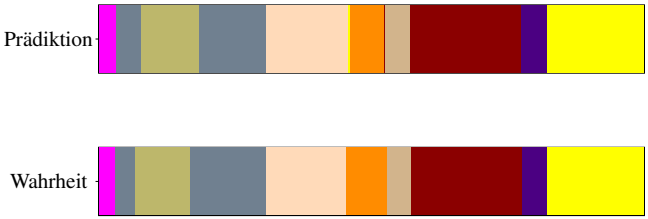


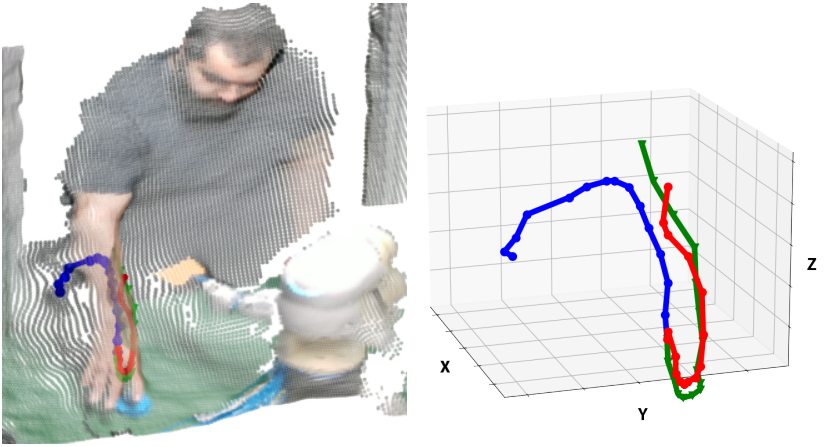
Abbildung 6.23: Qualitative Ergebnisse der dynamischen Aktionsprädiktion einer Aufnahme der Aufgabe 3 aus dem Testdatensatz $\mathcal{S}_5$.

Die Abbildungen 6.21, 6.22 und 6.23 zeigen ein sehr gutes Verhalten des Modells bei der Vorhersage der Aktionsklassen. Anders als bei der statischen Aktionsprädiktion, deuten die qualitativen Ergebnisse auf eine gute Übereinstimmung zwischen wahrer und prädizierter Aktionsklasse hin. Dieser Eindruck deckt sich ebenfalls mit den zuvor vorgestellten quantitativen Ergebnissen zur dynamischen Aktionsprädiktion (vgl. S. 134).

Bewegungsvorhersage

Für ein besseres Gesamtverständnis der Bewegungsvorhersage und des Nutzens im vorliegenden Anwendungsfall, wird die für diesen interessante Information durch die qualitativen Ergebnisse beispielhaft dargestellt. Die wichtigste Information bei der Bewegungsvorhersage ist das Wissen über die 3D-Handposition des Menschen und über die resultierende zukünftige Bewegung. Aus den trainierten Modellen kann die zukünftige Trajektorie für diese Handposition extrahiert werden. In der folgenden Abbildung 6.24 und Tabelle 6.31 sind zur Vereinfachung jeweils die verkürzten Trajektorien dargestellt.

In der Abbildung 6.24 wird dieser Sachverhalt für eine Momentaufnahme der Szene aus dem Testdatensatz $\mathcal{S}_5$ visualisiert. Das hier verwendete Modell entspricht der Architektur zur Kombination der Bewegungsvorhersage, der Aktionserkennung und der statischen Aktionsprädiktion.



(a) (b)

Abbildung 6.24: Dargestellt wird die vorhergesagte 3D-Trajektorie der Hand: (a) 3D-Punktwolke der aufgezeichneten Szene mit: ● historischer, ● vorhergesagter und ● wahrer Handtrajektorie. (b) 3D-Plot mit isolierten 3D-Trajektorien der Hand.

Die beiden Teilabbildungen der Abbildung 6.24 zeigen jeweils die gleichen Informationen: die historische, vorhergesagte und wahre 3D-Handtrajektorie die direkt aus dem Modell extrahiert wird. Mit dem Wissen über die zukünftige Position könnte der Roboter zum Beispiel proaktiv planen, um eine Kollision mit dem Menschen zu vermeiden, oder um dem Menschen zu assistieren.

Da es sich bei den beschriebenen Experimenten primär um eine Kombination mehrerer Aufgaben handelt, würde in einem solchen beschriebenen Anwendungsfall, die zusätzliche Information über die vom Menschen ausgeführte und zukünftig auszuführende Aktion ergänzend genutzt werden, um dem Roboter weitere Informationen zur Entscheidungsfindung zur Verfügung zu stellen.

In der nachfolgenden Tabelle 6.31 werden die qualitativen Ergebnisse der Bewegungsvorhersage des Modells zur kombinierten Architektur mit Bewegungsvorhersage, Aktionserkennung und statischer Aktionsprädiktion dargestellt. Dabei wird die vorhergesagte Bewegungstrajektorie der Hand als 2D-Projektion der tatsächlich vorhergesagten 3D-Punkte, auf dem korrespondierenden 2D-Bild dargestellt. Darüber hinaus werden die zugehörigen Vorhersagen der Aktionserkennung (hier: **AR**) und der statischen Aktionsprädiktion (hier: **AP**) mit

der wahren (hier: **W**) und der prädizierten (hier: **P**) Aktionsklasse dargestellt. Dadurch kann neben der Bewegungstrajektorie auch die zugehörige erkannte oder vorhergesagte Aktionsklasse zur Interpretation kombiniert werden.

Die Bewegungsvorhersage ist für die verschiedenen Bewegungen und die auftretenden Aktionen korrekt. In der ersten Zeile der qualitativen Ergebnisse der Tabelle 6.31 wird ersichtlich, dass die vorhergesagte Richtung und Position, in die sich die Hand bewegt, mit der wahren Position übereinstimmt. Die Positionsvorhersage des Modells ist teilweise stark fehlerbehaftet, wie beispielsweise in der Szene, die in der zweiten Zeile und zweiten Spalte (Aktion: *release_soldering_iron*) dargestellt ist, aber auch in diesem Fall ist die vorhergesagte Bewegungsrichtung korrekt.

In der zweiten Zeile und ersten Spalte wird die Szene beim Ausführen der Aktion *solder_capacitor* dargestellt. Hierbei handelt es sich beispielsweise um eine „statische“ Aktion, die vom Menschen ausgeführt wird und auch hier stimmt die Prädiktion mit der Wahrheit überein und es wird eine statische Bewegung vorhergesagt.

Die in der Tabelle 6.31 durch die Farbe *grau* eingefärbten Aktionsklassifikationen zeigen falsch klassifizierte Aktionen in der Erkennung oder Prädiktion. Gerade bei Aktionen wie *wait_for_robot* oder *release_soldering_tin* handelt es sich um die Aktion, die anschließend vom Menschen ausgeführt wird. Daher kommt es manchmal zu einer falschen Klassifikation. In der zweiten Zeile zeigen die Teilabbildungen in Spalte zwei und drei, den Unterschied von einem Frame bzw. 66 ms, in welchen aus einer richtigen Aktionsklassifikation, eine falsche wird. Die sichtbare Bewegung in der Teilabbildung (in der zweiten Zeile und dritten Spalte) zeigt bereits, dass als nächstes der Lötzinn abgelegt wird.

Ein weiterer möglicher Fehler in der Aktionsklassifikation zeigt sich auch bei der Aktionsklassifikation der wahren Aktion *release_soldering_tin*. Der Szenengraph, der als Eingangsszenengraph für das visualisierte Ergebnis in der Teilabbildung in Zeile zwei und Spalte vier verwendet wird, hat eine sehr große Ähnlichkeit mit dem Szenengraph der Aktion *grab_soldering_tin*, daher ist auch hier eine Verwechslung bei der Klassifikation möglich.

Die Tabelle 6.31 verdeutlicht den vollständigen Umfang der Aussagen, die durch die Methode dieser Arbeit ermöglicht werden, und dient somit als Basis für eine intelligente MRK. Diese Informationen können beispielsweise an einen Planer (vgl. Abbildung 2.1) übergeben werden, um die Bewegungssteuerung auf Basis

AR	P:	grab_capacitor	grab_capacitor	grab_soldering_tin	grab_soldering_tin	grab_soldering_iron
	W:	grab_capacitor	grab_capacitor	grab_soldering_tin	grab_soldering_tin	grab_soldering_iron
AP	P:	grab_capacitor	wait_for_robot	grab_soldering_tin	grab_soldering_tin	grab_soldering_iron
	W:	grab_capacitor	grab_capacitor	grab_soldering_tin	grab_soldering_tin	grab_soldering_iron
AR	P:	solder_capacitor	release_soldering_iron	release_soldering_tin	grab_soldering_tin	retreat
	W:	solder_capacitor	release_soldering_iron	release_soldering_iron	release_soldering_tin	retreat
AP	P:	solder_capacitor	release_soldering_iron	release_soldering_iron	grab_soldering_tin	retreat
	W:	solder_capacitor	release_soldering_iron	release_soldering_iron	retreat	retreat

Tabelle 6.31: Visualisierung der qualitativen Ergebnisse der Bewegungsvorhersage (projezierte 3D-Punkte auf dem 2D-Bild; zur Übersichtlichkeit wurde eine reduzierte Anzahl der Punkte dargestellt) zusammen mit der wahren und der prädizierten Aktionsklasse, der Aktionserkennung, sowie der statischen Aktionsprädiktion auf dem Testdatensatz $\mathcal{S}_5$ des CoAx-Datensatzes. Eingezeichnet sind die ● vorhergesagte und die ● wahre projezierte 2D-Handtrajektorie. Darüber hinaus sind Vorhersagen der Aktionserkennung (hier: **AR**) und der statischen Aktionsprädiktion (hier: **AP**) mit der wahren (hier: **W**) und der prädizierten (hier: **P**) Aktionsklasse dargestellt.

dieser Information proaktiv zu planen. Das Wissen über mögliche erkannte und vorhergesagte Aktionen könnte beispielweise durch einen Verhaltensbaum weiter verarbeitet werden, um den Roboter auf Aktionswissen proaktiv reagieren zu lassen.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde neben der Aufbereitung der Grundlagen und dem Stand der Technik die entwickelte Methode mit Bezug auf die untersuchte Anwendung und der resultierenden Architektur erarbeitet und beschrieben.

Es wurde dazu eine neuartige Strukturierung der wahrgenommenen Szeneninformation in einer MRK-Arbeitsumgebung durch die Modellierung eines menschenzentrierten Graphen vorgestellt. Dieser Graph beinhaltet dabei die räumliche Szeneninformation durch die Modellierung der detektierten Szenenobjekte inklusive des Menschen als Knoten und der zugrundeliegenden Relation zwischen diesen als Kanteninformation. Die Kanteninformation kann dabei lediglich das Vorhandensein einer Kante zwischen eines Knotenpaars oder aber der räumlichen Distanz zwischen zwei Objekten oder der Art der Kante, ob es sich um eine räumliche Verbindung oder um eine zeitliche Verbindung des selben Objekts über mehrere Zeitpunkte hinweg handelt, was die zeitliche Szeneninformation darstellt. Besonders ist zudem die räumliche Strukturierung der Knoten, welche um den Knoten, der die Hand einbettet angeordnet sind, da die Hand als Ziel der Bewegungsvorhersage somit als zentrales Element eines räumlichen und folglich wiederkehrend eines zeitlichen Graphen ist.

Für die Umsetzung einer intelligenten MRK wurde in der vorliegenden Arbeit auf Basis der Graphenmodellierung eine GNN-basierte Methode für die Aktions- und Bewegungsvorhersage erarbeitet, wobei diese Vorhersagen unabhängig oder kombiniert gelernt werden können. Dabei wurde ein mehrstufiger Ansatz nach der Theorie des Transferlernens und ein geteilter Encoder-Ansatz nach der Theorie des Multi-Task-Lernens entwickelt und ausgewertet. In letzterem wurde das effiziente Lernen mehrerer Lernaufgaben ohne Verschlechterung der Vorhersagegüte gezeigt. Zudem wurde die Erweiterbarkeit durch eine datengetriebene Optimierung der Verlustkombination mehrerer Lernaufgaben demonstriert. Konkret wurden die Bewegungsvorhersage der Hand des Menschen, die Erkennung und Vorhersage der von diesem Menschen getätigten

Aktion sowie die Vorhersage der nächsten Aktion und des Zeitpunkts des Aktionswechsels erarbeitet.

Für den untersuchten Anwendungsfall des kollaborativen Lösens von Montageaufgaben in der Industrie wurde darüber hinaus ein neuartiger Datensatz zum Lernen von Aktionen und Bewegungen eines Menschen in einer solchen MRK-Umgebung erstellt und veröffentlicht.

Dabei zeigt der erarbeitete Ansatz eine Verbesserung der Modellgüte bei der Aktionserkennung im Vergleich zu anderen Ansätzen, auf dem anwendungsnahen CoAx-Datensatz sogar um 20%. Darüber hinaus ist der Vorhersagefehler für die Bewegungsvorhersage mit dem vorgestellten Ansatz besser als die Baselines und erreicht für die beste Konfiguration (RGCN) einen FDE von weniger als 10cm bei einem Vorhersagezeitpunkt von einer Sekunde.

Obgleich die durchgeführten Experimente zeigen, dass die erarbeitete Methode gute Ergebnisse liefert und Potenzial für den Einsatz in der MRK für Montageaufgaben in der Industrie hat, sind noch einige Verbesserungen vorstellbar:

- Die vorgestellte Strukturierung der Szeneninformation als homogener Graph kann zukünftig durch Heterogenität in der Modellierung des Graphen selbst erweitert werden, um die in der Szene vorhandene semantische Information noch genauer abzubilden. Teilweise wurde dies in dieser Arbeit durch die Modellierung zeitlicher und räumlicher Relationen als Kantentypen für das RGCN umgesetzt. Dies kann weiter durch die Modellierung von Knotentypen erweitertet werden. Beispielsweise könnte explizit durch die konkrete Definition von Knotentypen für Mensch, Roboter und Werkstücke die Aussagekraft und somit die Modellierung verbessert werden sowie zugleich dann auch durch spezifische Kanten zwischen Knoten bestimmter Knotentypen detailliertere Relationen modelliert werden. Eine offene Herausforderung ist dann der Umgang mit der damit steigenden Komplexität in der Modellierung und muss abgewogen werden.
- Auch die Einbringung von *a priori* Wissen für die Lernaufgaben kann untersucht werden. Konkret ist damit die Verwendung von bereits vorhandenem Wissen gemeint, wie es zum Beispiel bei Montageanleitungen für Möbel oder eben auch bei Montageanleitungen für Montageaufgaben in der Industrie (vgl. Aktionsablauf von Montageaufgaben in Abbildung

6.2) vorhanden ist. Dadurch kann zum einen direkt der Lernprozess unterstützt werden, indem die Anleitung als weiteres Merkmal integriert wird. Zum anderen kann das a prior Wissen auch für eine bedingte Vorhersage genutzt werden, um während der Inferenz unplausible Aktions- oder Bewegungsvorhersagen, die aus der Montageanleitung bereits nicht hervorgehen, auszuschließen.

- Unter Berücksichtigung der neuesten Entwicklungen im aktuellen Stand der Forschung im Bereich des maschinellen Lernens, die über den Rahmen der vorliegenden Arbeit hinausgehen, sollte auch die Verwendung von *Large Language Models* (LLM) zur Lösung des untersuchten Problems in Erwägung gezogen werden. Insbesondere multimodale LLMs werden im Kontext des automatisierten Fahrens sowie in der mobilen und stationären Robotik zum Lösen mehrerer Lernaufgaben genutzt und bedürfen weiterer Untersuchung. Zu den multimodalen LLMs gehören beispielsweise auch die sogenannten Vision-LLMs, die direkt mit Bilddaten als Eingabe verarbeiten können. Für einen Überblick zu LLMs wird auf die Übersichtsarbeit [FTT+23] verwiesen.

Literaturverzeichnis

[AER+21] V. Adeli, M. Ehsanpour, I. Reid, J. C. Niebles, S. Savarese, E. Adeli, and H. Rezatofighi, TRiPOD: Human Trajectory and Pose Dynamics Forecasting in the Wild, in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Montreal, QC, Canada, October 2021, pp. 13370–13380. https://doi.org/10.1109/ICCV48922.2021.01314.

[ASY+19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, Optuna: A Next-generation Hyperparameter Optimization Framework, http://arxiv.org/abs/1907.10902, July 2019.

[AAD+11] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, Learning the semantics of object–action relations by observation, *The International Journal of Robotics Research* **30** no. 10 (2011), 1229–1249. https://doi.org/10.1177/0278364911410459.

[ATW15] E. E. Aksoy, M. Tamosiunaite, and F. Wörgötter, Model-free incremental learning of the semantics of manipulation actions, *Robotics and Autonomous Systems* **71** (2015), 118–133. https://doi.org/10.1016/j.robot.2014.11.003.

[ASA21] G. Akyol, S. Sariel, and E. E. Aksoy, A Variational Graph Autoencoder for Manipulation Action Recognition and Prediction, in *2021 20th International Conference on Advanced Robotics (ICAR)*, IEEE, 2021, pp. 968–973.

[APRB22] N. Azizi, H. Possegger, E. Rodolà, and H. Bischof, 3D Human Pose Estimation Using Möbius Graph Convolutional Networks, in *Computer Vision – ECCV 2022* (S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds.), *Lecture Notes in Computer Science*, Springer Nature Switzerland, Cham, 2022, pp. 160–178. https://doi.org/10.1007/978-3-031-19769-7_10.

[BKH16] J. L. Ba, J. R. Kiros, and G. E. Hinton, Layer Normalization, July 2016. https://doi.org/10.48550/arXiv.1607.06450.

[BCB16] D. Bahdanau, K. Cho, and Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, http://arxiv.org/abs/1409.0473, May 2016, Accepted at ICLR 2015 as oral presentation.

[BG09] J. Bang-Jensen and G. Z. Gutin, *Digraphs*, *Springer Monographs in Mathematics*, Springer London, London, 2009. https://doi.org/10.1007/978-1-84800-998-1.

[BHB+18] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, Relational inductive biases, deep learning, and graph networks, *arXiv:1806.01261 [cs, stat]* (2018), http://arxiv.org/abs/1806.01261.

[BWB08] A. Bauer, D. Wollherr, and M. Buss, Human–robot collaboration: A survey, *International Journal of Humanoid Robotics* **05** no. 01 (2008), 47–66. https://doi.org/10.1142/S0219843608001303.

[BBB+16] W. Bauer, M. Bender, M. Braun, P. Rally, and O. Scholtz, Lightweight robots in manual assembly–best to start simply!, *Frauenhofer-Institut für Arbeitswirtschaft und Organisation IAO, Stuttgart* **1** (2016). https://doi.org/10.24406/publica-r-298032.

[BBBK11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, Algorithms for Hyper-Parameter Optimization, in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), **24**, Curran Associates, Inc., 2011.

[BB12] J. Bergstra and Y. Bengio, Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research* **13** no. 10 (2012), 281–305.

[BYC13] J. Bergstra, D. Yamins, and D. Cox, Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, in *Proceedings of the 30th International Conference on Machine Learning*, PMLR, February 2013, pp. 115–123.

[BSCS21] N. Bodla, G. Shrivastava, R. Chellappa, and A. Shrivastava, Hierarchical Video Prediction using Relational Layouts for Human-Object Interactions, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Nashville, TN, USA, June 2021, pp. 12141–12150. https://doi.org/10.1109/CVPR46437.2021.01197.

[BCCI+21] A. Bonci, P. D. Cen Cheng, M. Indri, G. Nabissi, and F. Sibona, Human-Robot Perception in Industrial Environments: A Survey, *Sensors* **21** no. 5 (2021), 1571. https://doi.org/10.3390/s21051571.

[BL18] X. Bresson and T. Laurent, Residual Gated Graph ConvNets, April 2018. https://doi.org/10.48550/arXiv.1711.07553.

[BGLL17] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, Massive Exploration of Neural Machine Translation Architectures, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (M. Palmer, R. Hwa, and S. Riedel, eds.), Association for Computational Linguistics, Copenhagen, Denmark, September 2017, pp. 1442–1451. https://doi.org/10.18653/v1/D17-1151.

[BAY22] S. Brody, U. Alon, and E. Yahav, How Attentive are Graph Attention Networks?, in *International Conference on Learning Representations*, 2022.

[BSCH19] D. Busbridge, D. Sherburn, P. Cavallo, and N. Y. Hammerla, Relational Graph Attention Networks, http://arxiv.org/abs/1904.05811, April 2019.

[COK+18] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy, and C. Ryan, Sensor Technology in Autonomous Vehicles : A review, in *2018 29th Irish Signals and Systems Conference (ISSC)*, June 2018, pp. 1–4. https://doi.org/10.1109/ISSC.2018.8585340.

[CLMT21] D. Cao, J. Li, H. Ma, and M. Tomizuka, Spectral Temporal Graph Neural Network for Trajectory Prediction, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 1839–1845. https://doi.org/10.1109/ICRA48506.2021.9561461.

[CGM+20] Z. Cao, H. Gao, K. Mangalam, Q.-Z. Cai, M. Vo, and J. Malik, Long-Term Human Motion Prediction with Scene Context, in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2020, pp. 387–404. https://doi.org/10.1007/978-3-030-58452-8_23.

[CHS+21] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43** no. 1 (2021), 172–186. https://doi.org/10.1109/TPAMI.2019.2929257.

[CZ17] J. Carreira and A. Zisserman, Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[Car97] R. Caruana, Multitask Learning, *Machine Learning* **28** no. 1 (1997), 41–75. https://doi.org/10.1023/A:1007379606734.

[CHN+20] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, Relational Graph Learning for Crowd Navigation, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2020, pp. 10007–10013. https://doi.org/10.1109/IROS45743.2020.9340705.

[CTH20] Y. Chen, Y. Tian, and M. He, Monocular human pose estimation: A survey of deep learning-based methods, *Computer Vision and Image Understanding* **192** (2020), 102897.

[CSLT20] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka, Towards Efficient Human-Robot Collaboration With Robust Plan Recognition and Trajectory Prediction, *IEEE Robotics and Automation Letters*

5 no. 2 (2020), 2602–2609. https://doi.org/10.1109/LRA.2020.2972874.

[CZLT19] Y. Cheng, W. Zhao, C. Liu, and M. Tomizuka, Human Motion Prediction using Semi-adaptable Neural Networks, in *2019 American Control Conference (ACC)*, July 2019, pp. 4884–4890. https://doi.org/10.23919/ACC.2019.8814980.

[CvBB14] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, On the Properties of Neural Machine Translation: Encoder–Decoder Approaches, in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 103–111. https://doi.org/10.3115/v1/W14-4012.

[CvG+14] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734. https://doi.org/10.3115/v1/D14-1179.

[CUH16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, Fast and accurate deep network learning by exponential linear units (ELUs), in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[CP99] J. E. Colgate and M. A. Peshkin, Cobots, September 1999, https://patents.google.com/patent/US5952796A/en.

[CPAM20] E. Corona, A. Pumarola, G. Alenya, and F. Moreno-Noguer, Context-Aware Human Motion Prediction, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Seattle, WA, USA, June 2020, pp. 6990–6999. https://doi.org/10.1109/CVPR42600.2020.00702.

[CSY20] Q. Cui, H. Sun, and F. Yang, Learning Dynamic Relationships for 3D Human Motion Prediction, in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CV-*

PR), June 2020, pp. 6518–6526. https://doi.org/10.1109/CVPR42600.2020.00655.

[DBV16] M. Defferrard, X. Bresson, and P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, *NIPS'16*, Curran Associates Inc., Red Hook, NY, USA, December 2016, pp. 3844–3852.

[DOL20] P. Dendorfer, A. Ošep, and L. Leal-Taixé, Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation, http://arxiv.org/abs/2010.01114, October 2020.

[Die17] R. Diestel, *Graph Theory*, *Graduate Texts in Mathematics* **173**, Springer Berlin Heidelberg, Berlin, Heidelberg, 2017. https://doi.org/10.1007/978-3-662-53622-3.

[DFD22a] C. Diller, T. Funkhouser, and A. Dai, Forecasting Actions and Characteristic 3D Poses, http://arxiv.org/abs/2211.14309v1, November 2022.

[DFD22b] C. Diller, T. Funkhouser, and A. Dai, Forecasting Characteristic 3D Poses of Human Actions, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, New Orleans, LA, USA, June 2022, pp. 15893–15902. https://doi.org/10.1109/CVPR52688.2022.01545.

[DWA20] C. R. G. Dreher, M. Wächter, and T. Asfour, Learning Object-Action Relations from Bimanual Human Demonstration Using Graph Networks, *IEEE Robotics and Automation Letters* **5** no. 1 (2020), 187–194. https://doi.org/10.1109/LRA.2019.2949221.

[FL19] M. Fey and J. E. Lenssen, Fast Graph Representation Learning with PyTorch Geometric, *arXiv:1903.02428 [cs, stat]* (2019), http://arxiv.org/abs/1903.02428.

[FTT+23] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu, C. Lu, and M. Schwager, Foundation Models in Robotics: Applications, Challenges, and the Future, December 2023. https://doi.org/10.48550/arXiv.2312.07843.

[FLFM15] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, Recurrent Network Models for Human Dynamics, in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4346–4354.

[FYD+23] J. Fu, F. Yang, Y. Dang, X. Liu, and J. Yin, Learning Constrained Dynamic Correlations in Spatiotemporal Graphs for Motion Prediction, *IEEE Transactions on Neural Networks and Learning Systems* (2023), 1–15. https://doi.org/10.1109/TNNLS.2023.3277476.

[Fuk69] K. Fukushima, Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements, *IEEE Transactions on Systems Science and Cybernetics* **5** no. 4 (1969), 322–333. https://doi.org/10.1109/TSSC.1969.300225.

[GSZ+20] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11525–11533.

[Gav99] D. Gavrila, The Visual Analysis of Human Movement: A Survey, *Computer Vision and Image Understanding* **73** no. 1 (1999), 82–98. https://doi.org/10.1006/cviu.1998.0716.

[GYDD20] P. Ghosh, Y. Yao, L. Davis, and A. Divakaran, Stacked Spatio-Temporal Graph Convolutional Networks for Action Segmentation, in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 576–585.

[GSR+17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, Neural Message Passing for Quantum Chemistry, in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, July 2017, pp. 1263–1272.

[GCDZ19] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, Video Action Transformer Network, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 244–253.

[Gir15] R. Girshick, Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[GGDH18] G. Gkioxari, R. Girshick, P. Dollár, and K. He, Detecting and Recognizing Human-Object Interactions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8359–8367.

[GBB11] X. Glorot, A. Bordes, and Y. Bengio, Deep Sparse Rectifier Neural Networks, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, June 2011, pp. 315–323.

[GBA16] I. Goodfellow, Y. Bengio, and Aaron Courville, *Deep Learning*, *Adaptive Computation and Machine Learning*, MIT Press, 2016, https://www.deeplearningbook.org.

[GPM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative Adversarial Nets, in *Advances in Neural Information Processing Systems*, **27**, Curran Associates, Inc., 2014.

[Goo23] Google, Hand landmarks detection guide | Mediapipe, https://developers.google.com/mediapipe/solutions/vision/hand_landmarker, November 2023.

[GMS05] M. Gori, G. Monfardini, and F. Scarselli, A new model for learning in graph domains, in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, **2**, July 2005, pp. 729–734 vol. 2. https://doi.org/10.1109/IJCNN.2005.1555942.

[GZBA22] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, Understanding Pooling in Graph Neural Networks, *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–11. https://doi.org/10.1109/TNNLS.2022.3190922.

[GKGM20] R. Groenendijk, S. Karaoglu, T. Gevers, and T. Mensink, Multi-Loss Weighting with Coefficient of Variations, http://arxiv.org/abs/2009.01717, November 2020.

[GL16] A. Grover and J. Leskovec, Node2vec: Scalable Feature Learning for Networks, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Francisco California USA, August 2016, pp. 855–864. https://doi.org/10.1145/2939672.2939754.

[GZW+18] L.-Y. Gui, K. Zhang, Y.-X. Wang, X. Liang, J. M. F. Moura, and M. Veloso, Teaching Robots to Predict Human Motion, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018, pp. 562–567. https://doi.org/10.1109/IROS.2018.8594452.

[GDS+23] W. Guo, Y. Du, X. Shen, V. Lepetit, X. Alameda-Pineda, and F. Moreno-Noguer, Back to MLP: A Simple Baseline for Human Motion Prediction, in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 4809–4819.

[Ham20] W. L. Hamilton, *Graph Representation Learning*, *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Springer International Publishing, Cham, 2020. https://doi.org/10.1007/978-3-031-01588-5.

[HYL17] W. L. Hamilton, R. Ying, and J. Leskovec, Representation Learning on Graphs: Methods and Applications, *IEEE Data Eng. Bull.* **40** no. 3 (2017), 52–74. https://doi.org/10.48550/arXiv.1709.05584.

[HYL18] W. L. Hamilton, R. Ying, and J. Leskovec, Inductive Representation Learning on Large Graphs, September 2018. https://doi.org/10.48550/arXiv.1706.02216.

[HVG11] D. K. Hammond, P. Vandergheynst, and R. Gribonval, Wavelets on graphs via spectral graph theory, *Applied and Computational Harmonic Analysis* **30** no. 2 (2011), 129–150. https://doi.org/10.1016/j.acha.2010.04.005.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, *Springer Series in Statistics*, Springer, New York, NY, 2009. https://doi.org/10.1007/978-0-387-84858-7.

[HGDG17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, Mask R-CNN, in *2017 IEEE International Conference on Computer Vision (ICCV)*, October 2017, pp. 2980–2988. https://doi.org/10.1109/ICCV.2017.322.

[HZRS15] K. He, X. Zhang, S. Ren, and J. Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Clas-

sification, in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[HZRS16a] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Las Vegas, NV, USA, June 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.

[HZRS16b] K. He, X. Zhang, S. Ren, and J. Sun, Identity Mappings in Deep Residual Networks, in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2016, pp. 630–645. https://doi.org/10.1007/978-3-319-46493-0_38.

[HY20] R. Heckel and F. F. Yilmaz, Early Stopping in Deep Networks: Double Descent and How to Eliminate it, in *International Conference on Learning Representations*, October 2020.

[HHP17] S. Herath, M. Harandi, and F. Porikli, Going deeper into action recognition: A survey, *Image and vision computing* **60** (2017), 4–21.

[HS97] S. Hochreiter and J. Schmidhuber, Long Short-Term Memory, *Neural Computation* **9** no. 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[HCZG20] Y. Hu, S. Chen, Y. Zhang, and X. Gu, Collaborative Motion Prediction via Neural Motion Message Passing, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6319–6328.

[HBL+19] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6272–6281.

[HHL11] F. Hutter, H. H. Hoos, and K. Leyton-Brown, Sequential Model-Based Optimization for General Algorithm Configuration, in *Learning and Intelligent Optimization* (C. A. C. Coello, ed.), *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2011, pp. 507–523. https://doi.org/10.1007/978-3-642-25566-3_40.

[IS15] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, June 2015, pp. 448–456.

[IPOS14] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36** no. 7 (2014), 1325–1339. https://doi.org/10.1109/TPAMI.2013.248.

[JZL+19] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, A Survey of Deep Learning-Based Object Detection, *IEEE Access* **7** (2019), 128837–128868. https://doi.org/10.1109/ACCESS.2019.2939201.

[KCS+17] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, The Kinetics Human Action Video Dataset, May 2017. https://doi.org/10.48550/arXiv.1705.06950.

[KBA+17] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, A New Representation of Skeleton Sequences for 3D Action Recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3288–3297.

[KGC18] A. Kendall, Y. Gal, and R. Cipolla, Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[KW13] D. P. Kingma and M. Welling, Auto-Encoding Variational Bayes, December 2013. https://doi.org/10.48550/arXiv.1312.6114.

[KFW+18] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, Neural Relational Inference for Interacting Systems, in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, July 2018, pp. 2688–2697.

[KW16a] T. N. Kipf and M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, in *International Conference on Learning Representations*, November 2016.

[KW16b] T. N. Kipf and M. Welling, Variational graph auto-encoders, *NIPS Workshop on Bayesian Deep Learning* (2016).

[KHG+19] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, Panoptic Segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9404–9413.

[KS13a] H. Koppula and A. Saxena, Anticipating Human Activities using Object Affordances for Reactive Robotic Response, in *Robotics: Science and Systems IX*, **09**, June 2013.

[KS13b] H. Koppula and A. Saxena, Learning Spatio-Temporal Structure from RGB-D Videos for Human Activity Detection and Anticipation, in *Proceedings of the 30th International Conference on Machine Learning*, PMLR, May 2013, pp. 792–800.

[KS16] H. S. Koppula and A. Saxena, Anticipating Human Activities Using Object Affordances for Reactive Robotic Response, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38** no. 1 (2016), 14–29. https://doi.org/10.1109/TPAMI.2015.2430335.

[KGS13] H. S. Koppula, R. Gupta, and A. Saxena, Learning human activities and object affordances from RGB-D videos, *The International Journal of Robotics Research* **32** no. 8 (2013), 951–970. https://doi.org/10.1177/0278364913478446.

[KSM+19] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks, in *Advances in Neural Information Processing Systems*, **32**, Curran Associates, Inc., 2019.

[LSSD22] D. Lagamtzis, F. Schmidt, J. Seyler, and T. Dang, CoAx: Collaborative Action Dataset for Human Motion Forecasting in an Industrial Workspace, in *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, SciTePress, 2022, pp. 98–105. https://doi.org/10.5220/0010775600003116.

[LSS+23a] D. Lagamtzis, F. Schmidt, J. Seyler, T. Dang, and S. Schober, Exploiting Spatio-Temporal Human-Object Relations Using Graph Neural Networks for Human Action Recognition and

3D Motion Forecasting, in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Huntington Place in Detroit, Michigan, USA, October 2023, pp. 7832–7838. https://doi.org/10.1109/IROS55552.2023.10342491.

[LSS+23b] D. Lagamtzis, F. Schmidt, J. Seyler, T. Dang, and S. Schober, Graph Neural Networks for Joint Action Recognition, Prediction and Motion Forecasting for Industrial Human-Robot Collaboration, in *ISR Europe 2023; 56th International Symposium on Robotics*, VDE, Stuttgart, Germany, September 2023, pp. 30–37.

[LFS17] P. A. Lasota, T. Fong, and J. A. Shah, A Survey of Methods for Safe Human-Robot Interaction, *Foundations and Trends in Robotics* **5** no. 3 (2017), 261–349. https://doi.org/10.1561/2300000052.

[LS17] P. A. Lasota and J. A. Shah, A multiple-predictor approach to human motion prediction, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2300–2307. https://doi.org/10.1109/ICRA.2017.7989265.

[LFV+17] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, Temporal Convolutional Networks for Action Segmentation and Detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[LZLL18] C. Li, Z. Zhang, W. S. Lee, and G. H. Lee, Convolutional Sequence to Sequence Model for Human Dynamics, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5226–5234.

[LXTG20] G. Li, C. Xiong, A. Thabet, and B. Ghanem, DeeperGCN: All You Need to Train Deeper GCNs, http://arxiv.org/abs/2006.07739, June 2020.

[LMZT20] J. Li, H. Ma, Z. Zhang, and M. Tomizuka, Social-WaGDAT: Interaction-aware Trajectory Prediction via Wasserstein Graph Double-Attention Network, February 2020. https://doi.org/10.48550/arXiv.2002.06241.

[LYTC20] J. Li, F. Yang, M. Tomizuka, and C. Choi, EvolveGraph: Multi-Agent Trajectory Prediction with Dynamic Relational Reasoning, in *Advances in Neural Information Processing Systems*, **33**, Curran Associates, Inc., 2020, pp. 19783–19794.

[LES+21] K. Li, S. Eiffert, M. Shan, F. Gomez-Donoso, S. Worrall, and E. Nebot, Attentional-GCNN: Adaptive Pedestrian Trajectory Prediction towards Generic Autonomous Vehicle Use Cases, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 14241–14247. https://doi.org/10.1109/ICRA48506.2021.9561480.

[LCPW21] Q. Li, G. Chalvatzaki, J. Peters, and Y. Wang, Directed Acyclic Graph Neural Network for Human Motion Prediction, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 3197–3204. https://doi.org/10.1109/ICRA48506.2021.9561540.

[LYFG21] S. Li, J. Yi, Y. A. Farha, and J. Gall, Pose Refinement Graph Convolutional Network for Skeleton-Based Action Recognition, *IEEE Robotics and Automation Letters* **6** no. 2 (2021), 1028–1035. https://doi.org/10.1109/LRA.2021.3056361.

[LZL+23] S. Li, P. Zheng, S. Liu, Z. Wang, X. V. Wang, L. Zheng, and L. Wang, Proactive human–robot collaboration: Mutual-cognitive, predictable, and self-organising perspectives, *Robotics and Computer-Integrated Manufacturing* **81** (2023), 102510. https://doi.org/10.1016/j.rcim.2022.102510.

[LZW+22] S. Li, P. Zheng, Z. Wang, J. Fan, and L. Wang, Dynamic Scene Graph for Mutual-Cognition Generation in Proactive Human-Robot Collaboration, *Procedia CIRP* **107** (2022), 943–948. https://doi.org/10.1016/j.procir.2022.05.089.

[LTBZ16] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, Gated Graph Sequence Neural Networks, in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[LYH+20] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, Learning Lane Graph Representations for Motion Forecasting, in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2020, pp. 541–556. https://doi.org/10.1007/978-3-030-58536-5_32.

[LZW+23] C. Liu, Y. Zhan, J. Wu, C. Li, B. Du, W. Hu, T. Liu, and D. Tao, Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities, http://arxiv.org/abs/2204.07321, June 2023.

[LSP+20] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42** no. 10 (2020), 2684–2701. https://doi.org/10.1109/TPAMI.2019.2916873.

[LSXW16] J. Liu, A. Shahroudy, D. Xu, and G. Wang, Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition, in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2016, pp. 816–833. https://doi.org/10.1007/978-3-319-46487-9_50.

[LTMW22] S. Liu, S. Tripathi, S. Majumdar, and X. Wang, Joint Hand Motion and Interaction Hotspots Prediction from Egocentric Videos, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 3272–3282. https://doi.org/10.1109/CVPR52688.2022.00328.

[LJD19] S. Liu, E. Johns, and A. J. Davison, End-To-End Multi-Task Learning With Attention, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1871–1880.

[LZBC15] Z. Liu, J. Zhu, J. Bu, and C. Chen, A survey of human pose estimation: The body parts parsing based methods, *Journal of Visual Communication and Image Representation* **32** (2015), 10–19. https://doi.org/10.1016/j.jvcir.2015.06.013.

[LSD15] J. Long, E. Shelhamer, and T. Darrell, Fully Convolutional Networks for Semantic Segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[LTN+19] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, MediaPipe: A

Framework for Building Perception Pipelines, *arXiv:1906.08172 [cs]* (2019), http://arxiv.org/abs/1906.08172.

[LM19] R. Luo and L. Mai, Human Intention Inference and On-Line Human Hand Motion Prediction for Human-Robot Collaboration, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2019, pp. 5958–5964. https://doi.org/10.1109/IROS40897.2019.8968192.

[MHN+13] A. L. Maas, A. Y. Hannun, A. Y. Ng, and others, Rectifier nonlinearities improve neural network acoustic models, in *Proc. Icml*, **30**, Atlanta, GA, 2013, p. 3.

[MLSL19] W. Mao, M. Liu, M. Salzmann, and H. Li, Learning Trajectory Dependencies for Human Motion Prediction, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9489–9497.

[MBR17] J. Martinez, M. J. Black, and J. Romero, On Human Motion Prediction Using Recurrent Neural Networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2891–2900.

[MHRL17] J. Martinez, R. Hossain, J. Romero, and J. J. Little, A Simple yet Effective Baseline for 3D Human Pose Estimation, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2640–2649.

[MMZ+19] E. Matheson, R. Minto, E. G. G. Zampieri, M. Faccio, and G. Rosati, Human–Robot Collaboration in Manufacturing Applications: A Review, (2019), 25.

[ML05] C. Merkwirth and T. Lengauer, Automatic Generation of Complementary Descriptors with Molecular Graph Networks, *Journal of Chemical Information and Modeling* **45** no. 5 (2005), 1159–1168. https://doi.org/10.1021/ci049613b.

[MQEC20] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14424–14432.

[MLVT21] R. Morais, V. Le, S. Venkatesh, and T. Tran, Learning Asynchronous and Sparse Human-Object Interaction in Videos, in

2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Nashville, TN, USA, June 2021, pp. 16036–16045. https://doi.org/10.1109/CVPR46437.2021.01578.

[MRF+19] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks, *Proceedings of the AAAI Conference on Artificial Intelligence* **33** (2019), 4602–4609. https://doi.org/10.1609/aaai.v33i01.33014602.

[MJW+20] T. L. Munea, Y. Z. Jembre, H. T. Weldegebriel, L. Chen, C. Huang, and C. Yang, The Progress of Human Pose Estimation: A Survey and Taxonomy of Models Applied in 2D Human Pose Estimation, *IEEE Access* **8** (2020), 133330–133348. https://doi.org/10.1109/ACCESS.2020.3010248.

[MSRR18] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs, in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, September 2018.

[NH10] V. Nair and G. E. Hinton, Rectified Linear Units Improve Restricted Boltzmann Machines, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, *ICML'10*, Omnipress, Madison, WI, USA, 2010, pp. 807–814.

[NKB+19] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, Deep Double Descent: Where Bigger Models and More Data Hurt, in *International Conference on Learning Representations*, September 2019.

[OS20] K. Oono and T. Suzuki, Graph neural networks exponentially lose expressive power for node classification, in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[vdODZ+16] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, WaveNet: A Generative Model for Raw Audio,

September 2016. https://doi.org/10.48550/arXiv.1609.03499.

[PY10] S. J. Pan and Q. Yang, A Survey on Transfer Learning, *IEEE Transactions on Knowledge and Data Engineering* **22** no. 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191.

[PS23] C. Patsch and E. Steinbach, Self-Attention Based Action Segmentation Using Intra-And Inter-Segment Representations, in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2023, pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096960.

[PESVG09] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, You'll never walk alone: Modeling social behavior for multi-target tracking, in *2009 IEEE 12th International Conference on Computer Vision*, IEEE, Kyoto, September 2009, pp. 261–268. https://doi.org/10.1109/ICCV.2009.5459260.

[PS15] C. Perez-D'Arpino and J. A. Shah, Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Seattle, WA, USA, May 2015, pp. 6175–6182. https://doi.org/10.1109/ICRA.2015.7140066.

[RLL+23] S. Raab, I. Leibovitch, P. Li, K. Aberman, O. Sorkine-Hornung, and D. Cohen-Or, MoDi: Unconditional Motion Synthesis From Diverse Data, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13873–13883.

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.

[RPH+20] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, Human motion trajectory prediction: A survey, *The International Journal of Robotics Rese-*

arch **39** no. 8 (2020), 895–935. https://doi.org/10.1177/0278364920917446.

[SICP20] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data, in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2020, pp. 683–700. https://doi.org/10.1007/978-3-030-58523-5_40.

[SdA+22] A. Sampieri, G. M. D. di Melendugno, A. Avogaro, F. Cunico, F. Setti, G. Skenderi, M. Cristani, and F. Galasso, Pose Forecasting in Industrial Human-Robot Collaboration, in *Computer Vision – ECCV 2022* (S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds.), *Lecture Notes in Computer Science*, Springer Nature Switzerland, Cham, 2022, pp. 51–69. https://doi.org/10.1007/978-3-031-19839-7_4.

[SBIK16] N. Sarafianos, B. Boteanu, B. Ionescu, and I. A. Kakadiaris, 3D Human pose estimation: A review of the literature and analysis of covariates, *Computer Vision and Image Understanding* **152** (2016), 1–20. https://doi.org/10.1016/j.cviu.2016.09.002.

[SG64] A. Savitzky and M. J. E. Golay, Smoothing and Differentiation of Data by Simplified Least Squares Pro,cedures, *ANALYTICAL CHEMISTRY* **36** no. 8 (1964), 13.

[SGT+09] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, The Graph Neural Network Model, *IEEE Transactions on Neural Networks* **20** no. 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605.

[SKB+18] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, Modeling Relational Data with Graph Convolutional Networks, in *The Semantic Web* (A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2018, pp. 593–607. https://doi.org/10.1007/978-3-319-93417-4_38.

[SHSW05] O. Schrempf, U. Hanebeck, A. Schmid, and H. Worn, A novel approach to proactive human-robot cooperation, in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, IEEE, Nashville, TN, USA, 2005, pp. 555–560. https://doi.org/10.1109/ROMAN.2005.1513838.

[SLNW16] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1010–1019.

[SHF+21] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification, in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, Montreal, Canada, August 2021, pp. 1548–1554. https://doi.org/10.24963/ijcai.2021/214.

[SJMS17] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, Hand Keypoint Detection in Single Images Using Multiview Bootstrapping, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Honolulu, HI, July 2017, pp. 4645–4653. https://doi.org/10.1109/CVPR.2017.494.

[SLY15] K. Sohn, H. Lee, and X. Yan, Learning Structured Output Representation using Deep Conditional Generative Models, in *Advances in Neural Information Processing Systems*, **28**, Curran Associates, Inc., 2015.

[SKR17] T. Soo Kim and A. Reiter, Interpretable 3D Human Action Analysis With Temporal Convolutional Networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 20–28.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research* **15** no. 56 (2014), 1929–1958.

[SVL14] I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to Sequence Learning with Neural Networks, in *Proceedings of the 27th International Conference on Neural Information Processing*

Systems - Volume 2, *NIPS'14*, MIT Press, Cambridge, MA, USA, 2014, pp. 3104–3112.

[TXM+21] J. Tao, L. Xu, X. Ma, J. Yan, and K. Mei, Scene-Perception Graph Convolutional Networks for Human Action Prediction, in *2021 International Joint Conference on Neural Networks (IJCNN)*, July 2021, pp. 1–8. https://doi.org/10.1109/IJCNN52387.2021.9534132.

[ULT+18] V. V. Unhelkar, P. A. Lasota, Q. Tyroller, R.-D. Buhai, L. Marceau, B. Deml, and J. A. Shah, Human-Aware Robotic Assistant for Collaborative Assembly: Integrating Human Motion Prediction With Planning in Time, *IEEE Robotics and Automation Letters* **3** no. 3 (2018), 2394–2401. https://doi.org/10.1109/LRA.2018.2812906.

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is All you Need, in *Advances in Neural Information Processing Systems*, **30**, Curran Associates, Inc., 2017.

[VCC+18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, Graph Attention Networks, in *International Conference on Learning Representations*, February 2018.

[WZL+23] N. Wang, G. Zhu, H. Li, M. Feng, X. Zhao, L. Ni, P. Shen, L. Mei, and L. Zhang, Exploring Spatio-Temporal Graph Convolution for Video-based Human-Object Interaction Recognition, *IEEE Transactions on Circuits and Systems for Video Technology* (2023), 1–1. https://doi.org/10.1109/TCSVT.2023.3259430.

[WG18] X. Wang and A. Gupta, Videos as Space-Time Region Graphs, in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 399–417.

[WWLK17] Z. Wang, B. Wang, H. Liu, and Z. Kong, Recurrent convolutional networks based intention recognition for human-robot collaboration tasks, in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, Banff, AB, October 2017, pp. 1675–1680. https://doi.org/10.1109/SMC.2017.8122856.

[WSH+18] P. Weiner, J. Starke, F. Hundhausen, J. Beil, and T. Asfour, The KIT Prosthetic Hand: Design and Control, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2018, pp. 3328–3334. https://doi.org/10.1109/IROS.2018.8593851.

[WL68] B. Weisfeiler and A. Leman, The reduction of a graph to canonical form and the algebra which appears therein, *NTI, Series* **2** no. 9 (1968), 12–16.

[WKS+21] F. Wirthmüller, M. Klimke, J. Schlechtriemen, J. Hipp, and M. Reichert, Predicting the Time Until a Vehicle Changes the Lane Using LSTM-Based Recurrent Neural Networks, *IEEE Robotics and Automation Letters* **6** no. 2 (2021), 2357–2364. https://doi.org/10.1109/LRA.2021.3058930.

[XB22a] H. Xing and D. Burschka, Skeletal Human Action Recognition using Hybrid Attention based Graph Convolutional Network, in *2022 26th International Conference on Pattern Recognition (ICPR)*, August 2022, pp. 3333–3340. https://doi.org/10.1109/ICPR56361.2022.9956672.

[XB22b] H. Xing and D. Burschka, Understanding Spatio-Temporal Relations in Human-Object Interaction using Pyramid Graph Convolutional Network, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2022, pp. 5195–5201. https://doi.org/10.1109/IROS47612.2022.9981771.

[XLT+18] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, Representation Learning on Graphs with Jumping Knowledge Networks, in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, July 2018, pp. 5453–5462.

[XZJK21] K. Xu, M. Zhang, S. Jegelka, and K. Kawaguchi, Optimization of Graph Neural Networks: Implicit Acceleration by Skip Connections and More Depth, in *Proceedings of the 38th International Conference on Machine Learning*, PMLR, July 2021, pp. 11592–11602.

[YXL18] S. Yan, Y. Xiong, and D. Lin, Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition,

in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, *AAAI'18/IAAI'18/EAAI'18*, AAAI Press, New Orleans, Louisiana, USA, 2018.

[YMR+20] C. Yu, X. Ma, J. Ren, H. Zhao, and S. Yi, Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction, in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), *Lecture Notes in Computer Science*, Springer International Publishing, Cham, 2020, pp. 507–523. https://doi.org/10.1007/978-3-030-58610-2_30.

[Zac77] W. W. Zachary, An Information Flow Model for Conflict and Fission in Small Groups, *Journal of Anthropological Research* **33** no. 4 (1977), 452–473.

[ZKR+17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, Deep Sets, in *Advances in Neural Information Processing Systems*, **30**, Curran Associates, Inc., 2017.

[ZR17] A. M. Zanchettin and P. Rocco, Probabilistic inference of human arm reaching target for effective human-robot collaboration, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Vancouver, BC, September 2017, pp. 6595–6600. https://doi.org/10.1109/IROS.2017.8206572.

[ZVVM21] D. Zhang, N. A. Vien, M. Van, and S. McLoone, Non-local Graph Convolutional Network for joint Activity Recognition and Motion Prediction, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2021, pp. 2970–2977. https://doi.org/10.1109/IROS51168.2021.9636107.

[ZBV+20] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, MediaPipe Hands: On-device Real-time Hand Tracking, *arXiv:2006.10214 [cs]* (2020), http://arxiv.org/abs/2006.10214.

[ZC18] M. Zhang and Y. Chen, Link Prediction Based on Graph Neural Networks, in *Advances in Neural Information Processing Systems*, **31**, Curran Associates, Inc., 2018.

[ZKTW18] F. Ziaeetabar, T. Kulvicius, M. Tamosiunaite, and F. Wörgötter, Recognition and prediction of manipulation actions using Enriched Semantic Event Chains, *Robotics and Autonomous Systems* **110** (2018), 173–188. https://doi.org/10.1016/j.robot.2018.10.005.

Eigene Veröffentlichungen

[1] Dimitrios Lagamtzis, Fabian Schmidt, Jan Seyler, and Thao Dang. CoAx: Collaborative Action Dataset for Human Motion Forecasting in an Industrial Workspace. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 98–105. SciTePress, 2022.

[2] Dimitrios Lagamtzis, Fabian Schmidt, Jan Seyler, Thao Dang, and Steffen Schober. Exploiting Spatio-Temporal Human-Object Relations Using Graph Neural Networks for Human Action Recognition and 3D Motion Forecasting. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7832–7838, Huntington Place in Detroit, Michigan, USA, October 2023.

[3] Dimitrios Lagamtzis, Fabian Schmidt, Jan Seyler, Thao Dang, and Steffen Schober. Graph Neural Networks for Joint Action Recognition, Prediction and Motion Forecasting for Industrial Human-Robot Collaboration. In *ISR Europe 2023; 56th International Symposium on Robotics*, pages 30–37, Stuttgart, Germany, September 2023. VDE.