

Erkennung menschlicher Handlungen durch Auswertung der Körperhaltungen von Personen in einem Video mithilfe von Machine Learning und neuronalen Netzen

Bachelorarbeit

von Alexander Melde

Bearbeitungszeitraum	11.06.2018 – 31.08.2018
Abgabedatum	03.09.2018
Studiengang	Angewandte Informatik – Kommunikationsinformatik
Hochschule	Duale Hochschule Baden – Württemberg, Stuttgart
Matrikelnummer, Kurs	7939560, STG-TINF15K
Ausbildungsfirma	Netze BW GmbH, Stuttgart
Gutachter	Dr. M. Sieck, EnBW AG Prof. Dr. M. Babilon, DHBW Stuttgart

Eidesstattliche Erklärung

Ich, Alexander Melde, versichere hiermit, dass ich die vorliegende Arbeit mit dem Titel „Erkennung menschlicher Handlungen durch Auswertung der Körperhaltungen von Personen in einem Video mithilfe von Machine Learning und neuronalen Netzen“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Unterschrift

Abstract

Research in artificial intelligence and digital image processing has led to automatic person detection and human pose estimation.

This work proves if human pose estimation can be used to classify human actions.

Classifying videos or even actions in videos is still an unsolved and actively researched problem.

This work first explains basics of video surveillance, digital image processing and artificial intelligence to create an understanding of the context. Second, this work compares and rates previous methods and implementations of human pose estimation.

To train the artificial intelligence, videos labeled by action classes are required. This work shows some of the available datasets needed to train the artificial intelligence.

Learning from the results of the survey, multiple prototypes for human action recognition are developed as a proof of concept.

Finally, expansion and optimization methods for a productive system are presented.

At every step, optimization methods regarding the application "surveillance of public places" are evaluated, as the action recognition should be used to differentiate between violent and peaceful situations by classifying actions like punches and kicks as well as normal everyday activities.

Keywords

Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), pose estimation, action recognition, human action classification

Kurzbeschreibung

Dank bedeutender Forschungsergebnisse in den Bereichen der künstlichen Intelligenz und digitalen Bildverarbeitung ist es Computern mithilfe von künstlichen neuronalen Netzen möglich, Personen in Videos zu detektieren und deren Körperhaltungen abzuschätzen.

In dieser Arbeit wird geprüft, ob durch Auswertung dieser Körperhaltungen menschliche Handlungen erkannt werden können.

Die selbstständige Klassifikation von Videos oder gar Handlungen in Videos durch einen Computer ist ein noch nicht gelöstes Problem, zu dem noch viel geforscht wird. Um die komplexen Zusammenhänge, Lösungsvorschläge und Implementierungen zu verstehen, werden in dieser Arbeit zunächst einige Grundlagen aus den Bereichen Videoüberwachung, digitale Bildverarbeitung und künstliche Intelligenz erarbeitet. Anschließend werden zahlreiche Ansätze zur Handlungsklassifikation in Videos miteinander verglichen und bewertet.

Damit die künstliche Intelligenz angelernt werden kann, werden darüber hinaus zahlreiche Datensätze mit beschrifteten Handlungen aufgezeigt.

Ausgehend von den Ergebnissen dieser Untersuchung werden anschließend mehrere im Rahmen dieser Arbeit entwickelter Prototypen zur Handlungserkennung vorgestellt. Für einen produktiven Einsatz werden abschließend Erweiterungs- und Optimierungs-Möglichkeiten gezeigt.

In jedem Schritt werden darüber hinaus Optimierungen hinsichtlich des Anwendungsfalls „Überwachung von öffentlichen Plätzen“ geprüft. Durch die Haltungserkennung sollen Gewaltsituationen in Videoüberwachungs-Streams erkannt und Aktionen wie Schläge oder Tritte von normalen Alltags-handlungen unterschieden werden können.

Stichwörter

Neuronale Netze, Faltungsnetze, rekurrente neuronale Netze, CNN, RNN, LSTM, Haltungserkennung, Handlungserkennung, Klassifikation von Handlungen

Inhaltsverzeichnis

1 Einleitung	1
1.1 Umfeld der Arbeit	1
1.2 Motivation und Problemstellung	1
1.3 These	2
1.4 Ziele der Arbeit	2
1.5 Vorgehensweise	2
2 Grundlagen	3
2.1 Analyse von digitalen Videos	3
2.1.1 Geschichte der Videoüberwachung	3
2.1.2 Intelligente Analyse von Videos	5
2.1.3 Digitale Repräsentation von Videos	6
2.1.4 Anonymisierung	7
2.1.5 Haltungserkennung	9
2.2 Machine Learning (ML)	11
2.2.1 Lernmethoden	11
2.2.2 Klassifikation	12
2.2.3 Künstliche neuronale Netze	13
2.2.4 Convolutional Neural Networks (CNNs)	16
2.2.5 Recurrent Neural Networks (RNNs)	18
3 Handlungserkennung	20
3.1 Grundlagen	21
3.1.1 Interpretation von Einzelbildern (Raum)	21
3.1.2 Interpretation von Bewegungsdaten (Zeit)	22
3.2 Untersuchung bisheriger Implementierungen	25
3.2.1 Vergleich	28
3.2.2 Auswertung der Untersuchung	30
3.3 Handlungserkennung mithilfe von Körperhaltungen	31
3.3.1 Strukturierung der Körperhaltungs-Daten	32
3.4 Datensätze zur Handlungserkennung	34
3.4.1 Unbeschriftete Videos	35
3.4.2 Videos mit Handlungen	36
3.4.3 Videos mit Handlungen und Körperhaltungen	38
3.4.4 Sonstige Datensätze	38
3.4.5 Herausforderungen bei den Datensätzen	39
3.5 Anpassungen hinsichtlich des Anwendungsfalls	41
3.6 Herausforderungen der Handlungserkennung	42

4 Experimente	44
4.1 Projektplanung	44
4.1.1 Anforderungsdefinition	44
4.1.2 Lösungsansatz	45
4.1.3 Risikoanalyse	46
4.1.4 Vorangehender Test	48
4.2 Prototyp 1: Auswertung der räumlichen Dimension	49
4.2.1 Vorbereiten der Trainingsdaten	49
4.2.2 Training des Bildklassifikators	51
4.2.3 Test und Auswertung	52
4.2.4 Installation und Ausführung	54
4.3 Prototyp 2: Auswertung von Raum und Zeit	55
4.3.1 Implementierung	55
4.3.2 Ausführung	56
4.3.3 Test und Auswertung	58
4.4 Prototyp 3: Auswerten von Zeitreihen mittels LSTM	61
4.4.1 Darstellung des Videos als Zeitreihe	61
4.4.2 Handlungsklassifikation durch Zeitreihen	63
4.4.3 Test und Auswertung	64
4.5 Auswertung der Experimente	65
5 Ausblick	66
5.1 Erweiterungsmöglichkeiten	66
5.1.1 Optimierung der Erkennungsrate	66
5.1.2 Optimierung der Geschwindigkeit	67
5.1.3 Optimierungen hinsichtlich des Anwendungsfalls	68
5.1.4 Funktionserweiterung	68
5.2 Handlungsempfehlung	69
6 Fazit	70

Abkürzungsverzeichnis

2D	Zweidimensional	9
3D	Dreidimensional	9
Abb.	Abbildung	4
BPTT	Backpropagation through time	26
BRNN	Bidirectional RNN	18
CCTV	closed-circuit television	3
CNN	Convolutional Neural Network	9
CV	Computer Vision	4
EnBW	EnBW Energie Baden-Württemberg AG	1
HOF	Histogram of Optical Flow	22
HOG	Histograms of Oriented Gradients	23
HSV	Hue, Saturation, Value	6
iDT	Improved Dense Trajectories	24
JPEG	Joint Photographic Experts Group	6
JSON	JavaScript Object Notation	62
k. A.	keine Angabe	28
KI	Künstliche Intelligenz	11
LRCN	Long-term Recurrent Convolutional Network	26
LSTM	Long Short-Term Memory	19
MBH	Motion Boundary Histogram	24
MIL	Multi Instance Learning	12
ML	Machine Learning	11
PDF	Portable Document Format	62
ReLU	Rectified Linear Unit	15
RGB	Rot, Grün und Blau	6
RNN	Recurrent Neural Network	18
RTSP	Real-Time Streaming Protocol	68
ST-LSTM	Spatio-Temporal LSTM	33
SVM	Support Vector Machine	12
TSM	Temporal Segment Networks	26

Abbildungsverzeichnis

2.1	Anwendungsfälle für computergestützte Videoanalyse	4
2.2	Videokamera als Blackbox-Sensor	5
2.3	Aufbau einer RGB-Raster-Grafik	6
2.4	Anonymisierung mittels Bewegungsextraktion	7
2.5	Verschiedene Methoden zur Anonymisierung von Personen	8
2.6	Darstellungsformen Haltungserkennung	9
2.7	SVM zur Klassifikation von sportlichen Personen.	12
2.8	Minimales neuronales Netz	13
2.9	Aus drei Schichten bestehendes neuronales Netz	14
2.10	Verschiedene Aktivierungsfunktionen für künstliche neuronale Netze	15
2.11	Convolutional Neural Network (CNN)	16
2.12	Beispielhaftes CNN zur Bildklassifikation	17
2.13	Darstellung eines klassischen neuronalen Netzes als RNN	18
2.14	Verschiedene Typen von RNN	19
3.1	Vereinfachte Darstellung des Zwei-Kanal-Ansatzes	22
3.2	Visualisieren von Bewegungen mittels optischem Fluss	23
3.3	Verschiedene Stufen bei der Erstellung eins HOG	23
3.4	Verschiedene Ansätze zur Visualisierung von Bewegungsdaten	24
3.5	Zwei-Kanal-CNN	25
3.6	Skelett-Hierarchie	32
3.7	Beispielhafte Einzelbilder aus den Datensätzen	35
4.1	Screenshots des getesteten Handlungserkenners	48
4.2	Ordnerstruktur des ersten Prototyps	49
4.3	Architektur des ersten Prototyps	50
4.4	Netzarchitektur des Bildklassifikators	51
4.5	Wahrheitsmatrix des ersten Prototyps	52
4.6	Wahrheitsmatrizen des ersten Prototyps beim KTH Datensatz	53
4.7	Posen-Überlagerung im zweiten Prototyp	55
4.8	Grafische Ausgabe des zweiten Prototyps	57
4.9	Wahrheitsmatrix des zweiten Prototyps	58
4.10	Wahrheitsmatrizen des zweiten Prototyps beim KTH Datensatz	59
4.11	Genauigkeit des zweiten Prototyps	60
4.12	Darstellung der Körperhaltungen in einem Video durch Gelenk-Positionen	61
4.13	Gelenk-Beschleunigung im zeitlichen Verlauf	62
4.14	Genauigkeit und Fehlerrate beim dritten Prototyp	64

Tabellenverzeichnis

3.1	Übersicht über bisherige Implementierungen	27
3.2	Vergleich der untersuchten Implementierungen	29
3.3	Datensätze ohne Beschriftungen	35
3.4	Datensätze mit beschrifteten Handlungen	37
3.5	Datensätze mit Videos und Keypoints	38
4.1	Ergebnisse der Experimente	65

Quellcodeverzeichnis

2.1	Zugriff auf einzelne Farbwerte eines Videos	6
4.1	Bauen der C++ Bibliotheken für die Handlungserkennung	54
4.2	Aufruf der Handlungserkennung des ersten Prototyps	54
4.3	Aufruf der Handlungserkennung des zweiten Prototyps	56

1 Einleitung

In diesem Kapitel werden zur Orientierung das Projektumfeld, die Motivation sowie die Problemstellung beschrieben, eine These und verschiedene Ziele der Arbeit aufgestellt und die zum Erreichen dieser Ziele verwendete Vorgehensweise beschrieben.

1.1 Umfeld der Arbeit

Die EnBW Energie Baden-Württemberg AG (EnBW) ist ein deutsches Energieunternehmen mit rund 5,5 Millionen Kunden [37, S. 4]. Sie ist überwiegend tätig in den Bereichen Vertrieb, Netze, erneuerbare Energien, Erzeugung und Handel [38, S. 14]. Die Kernkompetenz der EnBW besteht dabei aus dem sicheren und zuverlässigen Betrieb sowie dem Management kritischer Infrastruktur im Bereich Energie [38, S. 26].

Diese Arbeit entstand in dem EnBW-Technologie-Labor „TechLab“, das neue und zukunftsweisende Technologien testet und mögliche Anwendungen dieser für den Konzern erarbeitet. Bei der Entwicklung von Prototypen wird eng mit potentiellen internen und externen Kunden zusammengearbeitet, um die Techniken direkt an den jeweiligen Einsatzorten testen zu können [38, S. 41].

1.2 Motivation und Problemstellung

Zur Umsetzung der Energiewende wurde die Konzern-Strategie EnBW 2020 entwickelt, die unter anderem einen Rückbau von Energieversorgungsanlagen beinhaltet [37, S. 4]. Ein langfristiges strategisches Ziel der EnBW ist daher das Erschließen neuer Wachstumschancen im Bereich der Kernkompetenzen, auch jenseits des Energiesektors, ab 2020 [38, S. 26].

Bereits heute beschäftigt sich die EnBW im Bereich Forschung und Entwicklung mit „der frühzeitigen Identifizierung wichtiger Trends und technologischer Entwicklungen sowie dem Aufbau von Kompetenzen für die spätere kommerzielle Nutzung in Pilot- und Demonstrationsprojekten“ [38, S. 37]. Ziel dieser Forschung ist eine Vergrößerung des Dienstleistungs-Portfolio der EnBW [37, S. 4–5].

Mithilfe einer Handlungserkennung lassen sich zahlreiche neue Dienstleistungen entwickeln. Der Haupt-Anwendungsfall, der in dieser Arbeit betrachtet werden soll, ist die anonymisierte Analyse von Handlungen an öffentlichen Orten, um ein sicheres Miteinander zu gewährleisten, indem bei gefährdenden Handlungen automatisch eine Hilfe gerufen wird. Weitere Anwendungsgebiete sind das Auswertung von Handlungen in Videos für statistische Zwecke, beispielsweise zur Analyse von Spieltechniken im Sport, und der Schutz kritischer Infrastruktur durch eine frühe Erkennung von Handlungen wie dem Überspringen eines Sicherheits-Zauns.

1.3 These

Durch Auswertung von Körperhaltungen in einem Video ist es möglich, Handlungen zu erkennen.

1.4 Ziele der Arbeit

In dieser Arbeit soll untersucht werden, in welchem Umfang eine Handlungserkennung durch Auswertung von Körperhaltungen nach aktuellem Stand der Wissenschaft theoretisch und auch praktisch möglich ist. Die Untersuchungen sollen durch die Entwicklung eines Prototypen, der durch Auswertung von Körperhaltungen in einem Video Handlungen erkennen kann, bestätigt werden.

1.5 Vorgehensweise

Am Anfang dieser Arbeit werden alle Grundlagen beschrieben, die für ein Verständnis der nachfolgenden Kapitel erforderlich sind. Es erfolgt eine Einleitung in die Bereiche „Digitale Videoüberwachung“, „Maschinelles Lernen“ und „Neuronale Netze“.

Im ersten Hauptteil dieser Arbeit werden zahlreiche Ansätze und Datensätze zur Erkennung von Handlungen vorgestellt und aktuelle Herausforderungen in diesem Bereich aufgezeigt. Am Ende des Kapitels wird geprüft, welche dieser Ansätze sich unter welchen Voraussetzungen für den Anwendungsfall eignen.

Der zweite Hauptteil dieser Arbeit beschreibt die Planung, Implementierung und Evaluierung verschiedener Prototypen zur Handlungserkennung. Hierfür erfolgt eine qualifizierte Auswahl eines geeigneten Algorithmus zur Erkennung von Körperhaltungen sowie eines passenden Klassifikators.

Abschließend werden einige Funktions-Erweiterungen und Optimierungs-Möglichkeiten erarbeitet sowie ein Resümee gezogen.

2 Grundlagen

Dieses Kapitel verfolgt das Ziel, alle für ein Verständnis der nachfolgenden Kapitel erforderlichen Grundlagen zu beschreiben, insbesondere die Entwicklung der digitalen Videoüberwachung, das Aufzeigen verschiedener Techniken im Bereich der digitalen Bildverarbeitung, die Erklärung des Begriffs Machine Learning sowie die Erklärung verschiedener neuronaler Netze. Das zentrale Thema der Handlungserkennung wird im Hauptteil dieser Arbeit erläutert.

2.1 Analyse von digitalen Videos

Die Ergebnisse dieser Arbeit finden Anwendung im Bereich der Videoanalyse. In diesem Abschnitt werden daher die Grundlagen und Ursprünge dieser Technik aufgezeigt.

2.1.1 Geschichte der Videoüberwachung

Der englische Begriff für Videoüberwachung lautet closed-circuit television (CCTV) – wörtlich ins Deutsche übersetzt, bedeutet das in etwa „Fernsehen für einen geschlossenen Personenkreis“. Diese interessante Formulierung beinhaltet bereits Hinweise auf den Ursprung der Videoüberwachung.

Das erste System zur Videoüberwachung wurde 1942 im deutschen Peenemünde aufgebaut, um den Start von nationalsozialistischen Raketen zu beobachten [34, S. 14], rund 16 Jahre nach der ersten öffentlichen Zurschaustellung eines Fernseh-Systems in 1926 [16, S. 6].

Seitdem hat sich im Bereich der Videoüberwachung viel verändert. Während anfangs eine dauerhafte Überwachung durch Mitarbeiter erforderlich war, ist es heutzutage möglich, Aufnahmen zu speichern und später noch einmal anzusehen. Mit der zunehmenden Leistungsfähigkeit von Computern wurden ab 1990 auch Überwachungsvideos mithilfe von Computern ausgewertet, beispielsweise für Zeitraffer- oder bewegungsaktivierte Aufnahmen. [105]

In naher Vergangenheit haben sich auch Systeme zur automatisierten Erkennung von Objekten und Personen sowie zur Re-Identifizierung dieser etabliert. Auch die Verbreitung von Videoüberwachung ist deutlich gestiegen. Die Anzahl der Überwachungskameras in London (Vereinigtes Königreich) wurde beispielsweise bereits 2002 auf 500.000 geschätzt. [86, S. 20] Genutzt werden die Kameras unter anderem zum Vergleich von Gesichtern im Kamera-Bild mit „gesuchten“ Gesichtern einer Verbrecher-Liste. Die britische Polizei soll hierfür bereits über 19 Millionen Bilder von Gesichtern verfügen. [81, S. 34]

Als größter Nutzer von Videoüberwachung gilt zurzeit die Volksrepublik China. Bis Ende 2017 wurden über das gesamte Land verteilt bereits über 170 Millionen Überwachungskameras installiert. Bis 2020 sollen 400 Millionen weitere Kameras dazukommen. [79]

Die hohe Dichte an Kameras und die wenigen Datenschutzgesetze ermöglichen dort nicht nur eine Identifikation der Gesichter in einzelnen Kameras, sondern auch eine landesweite Lokalisierung von Personen anhand deren Gesicht oder Name. In einem Test des britischen Fernsehsenders BBC wurde das Bild eines Reporters an die chinesische Polizei gegeben, um zu sehen, wie schnell dieser lokalisiert werden kann. Zwischen der Abgabe des Bilds und der Festnahme der gesuchten Person lagen nur sieben Minuten. [79]

Anwendung findet Videoüberwachung aber nicht nur auf staatlicher oder kommunaler Ebene. Auch zahlreiche Unternehmen und Privatpersonen nutzen Überwachungskameras, um Firmengelände, Grundstücke oder Wertsachen zu schützen.

Klassischerweise funktioniert Videoüberwachung, indem sich Wachpersonal die Bilder aller Kameras anschaut und prüft, ob darin Gefahren auftreten. Wenn das gleiche Personal nun aber immer mehr Kameras überblicken soll, so sinkt die Qualität der Überwachung, wodurch Situationen übersehen werden könnten. Um steigende Personalkosten zu verhindern und bestehendes Wachpersonal zu entlasten wurden computer-gestützte Lösungen entwickelt. Diese führen zur Beibehaltung oder Verbesserung der gewohnten Qualität bei steigender Anzahl von Kameras. Die Grundidee hierbei ist es, einige Statistiken selbstständig zu tätigen und darüber hinaus dem Wachpersonal nur genau die Kameras und Szenen zu zeigen, die ein Computer als verdächtig einschätzt, und „harmlose“ Szenen herauszufiltern. Einige Anwendungsfälle dieser Technik sind in Abbildung (Abb.) 2.1 gezeigt.



Abb. 2.1: Anwendungsfälle für computergestützte Videoanalyse. [56]

Mithilfe der Computer ist nicht nur, wie in China, eine viel größer vernetzte Auswertung möglich, sondern es ergeben sich auch neue Anwendungsfälle, die sich unauffällig in den Alltag integrieren lassen.

Bereits seit 2016 sind auch auf europäischen Straßen Systeme zur automatischen Kennzeichenerkennung im Einsatz, wodurch Zoll- und Grenzkontrollen vereinfacht werden [91]. In Parkhäusern werden Kameras genutzt um noch verfügbare Parkplätze zu ermitteln, und an zahlreichen Bahnhöfen und Flughäfen gehören Systeme Erkennung von verloren gegangenen oder vergessenen Gepäckstücken bereits zum Alltag [68].

Weitere Anwendungsfälle sind unter anderem (1) die Zählung von Personen oder Fahrzeugen, (2) die Detektion von Warteschlangen oder Verkehrstaus sowie (3) die Erkennung von sich (nicht) in eine vorgegebene Bewegungsrichtung bewegenden Objekten [68].

Die hinter diesen Produkten stehende Technik wird Computer Vision (CV) genannt. CV beschreibt die maschinelle Lösung von Aufgaben, die Menschen durch „sehen“ lösen würden. Neben der Video-Überwachung werden CV-Algorithmen auch in vielen anderen Bereichen eingesetzt, wie beispielsweise an Supermarktkassen, um Barcodes zu scannen, in Produktionsbetrieben, um Qualitätskontrollen durchzuführen, im medizinischen Bereich und in Multifunktionsdruckern, um Text in eingescannten Bildern zu erkennen.

2.1.2 Intelligente Analyse von Videos

Die eben beschriebenen Lösungen haben mehrere Nachteile: Sie sind meist teuer, speziell auf einen Anwendungsfall spezialisiert und unflexibel, verglichen mit den moderneren und zukünftigen Lösungen, die in diesem Abschnitt vorgestellt werden. In den letzten Jahren gab es deutliche Verbesserungen im Bereich der technischen Möglichkeiten. Die notwendige Hardware wird kleiner, leistungsfähiger und benötigt weniger Strom. Das ermöglicht es, Berechnungen direkt in der Kamera-Einheit oder einem Kamera-nahen Microcomputer durchzuführen, ganz ohne einen zeit- und ressourcenintensiven Transport in ein Rechenzentrum.

Auch auf Seiten der Software hat sich im Bereich der Videokodierung und der Objekt- und Gesichtserkennung in den letzten Jahren einiges getan. Grund hierfür ist unter anderem die Forschung im Bereich Machine Learning (siehe Abschnitt 2.2), die in dieser Arbeit eine zentrale Rolle spielen wird. Durch die großen Performance- und Genauigkeitsgewinne und zahlreiche neue Frameworks, die die Portabilität der Algorithmen deutlich erhöhten, konnten in den letzten Jahren zahlreiche Forschungsergebnisse erzielt werden, die im Bereich der Videoüberwachung angewendet zu neuen Möglichkeiten führen können.

Zahlreiche Produkte, die mit Hilfe von Machine Learning Videos analysieren, besitzen keine Anonymisierung. In Deutschland dürfen diese aufgrund der geltenden Datenschutz-Gesetze nicht überall oder nur nach Einverständnis aller gefilmten eingesetzt werden, da dass die Privatsphäre der dort lebenden Bürger sonst zu sehr einschränken würde [15].

Um zukunftssichere, ethisch und datenschutzrechtlich kompatible Produkte zu erarbeiten, wird der Ansatz verfolgt, Kameras als Sensoren zu betrachten.

Die Sensoren sollen zwar weiterhin Bilder aufnehmen, diese aber innerhalb einer Blackbox auswerten und nur nicht-personenbeziehbare Daten ausgeben. Ein Beispiel für diese Form einer sehr starken Anonymisierung wird in Abb. 2.2 gezeigt.



Abb. 2.2: Videokamera als Blackbox-Sensor. Die Kamera wertet das ursprüngliche Bild aus und gibt nur anonymisierte Daten zurück.

Sensoren können aber nicht nur Daten liefern, sondern auch Ereignisse signalisieren. Im Falle einer Videoüberwachung käme beispielsweise ein Bewegungsmelder, der zwischen Menschen und anderen Objekten unterscheiden kann, oder ein Objektdetektor, der bei Sicht eines Messers eine Warnung auslöst in Frage. Auch Ereignisse wie ausbrechende Feuer oder Prügeleien könnten zu einem Alarm führen.

Umfangreiche Überwachungssysteme kombinieren diese visuellen Sensoren mit akustischen Sensoren, um Vorfälle anhand ihrer akustischen Signatur zu erkennen (Audioanalyse), sodass beispielsweise bei einem Glasbruch oder beim Betätigen einer Spraydose ein Alarm ausgelöst werden kann.

2.1.3 Digitale Repräsentation von Videos

Nachdem in den letzten beiden Abschnitten zahlreiche Grundlagen zur Geschichte der Bildverarbeitung beschrieben wurden, folgen nun technische Details. Ein digitales Video besteht aus mehreren aufeinanderfolgenden Einzelbildern, ähnlich wie ein klassisches Daumen-Kino.

Während die Bilder des Daumen-Kinos noch entweder per Hand gezeichnet oder gedruckt wurden, werden die in dieser Arbeit verarbeiteten Bilder digital abgespeichert. Um spätere Zusammenhänge und Entscheidungen besser zu verstehen, ist ein grundlegendes Verständnis über den internen Aufbau eines digitalen Bilds erforderlich.

Bilder werden digital je nach Ursprung und Anwendungsfall auf verschiedene Arten gespeichert. Kamerabilder, wie beispielsweise die einer Überwachungskamera, werden in der Regel als Raster-Grafik gespeichert. Im Folgenden wird nur von diesem Fall ausgegangen.¹

Eine Raster-Grafik besteht aus mehreren einfarbigen Bildpunkten (Pixeln). Jede Farbe kann als Mischung der Farbwerte Rot, Grün und Blau (RGB) angegeben werden. Die einzelnen Bildpunkte und Farben werden in mehrdimensionalen Matrizen (Arrays) gespeichert, die mehrere Zeilen, Zellen und Farb-Mischwerte unterscheiden (siehe Abb. 2.3). [41, S. 585]

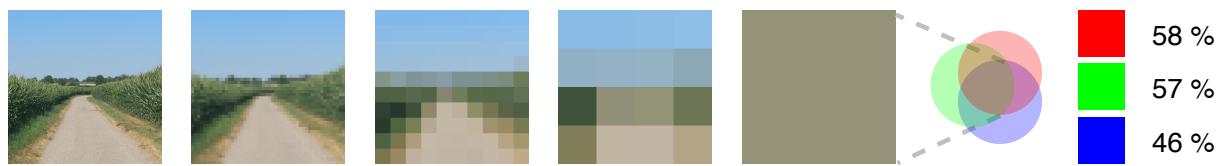


Abb. 2.3: Aufbau einer RGB-Raster-Grafik. Das Bild ist in mehrere Bildpunkte unterteilt, wobei jeder Bildpunkt durch eine Mischung der drei Grundfarben Rot, Grün und Blau (RGB) angegeben werden kann.

Kombiniert man nun also mehrere Bilder zu einem Video, so erhält man eine vierdimensionale Datenstruktur. Der Zugriff auf einen einzelnen Farbwert wird in Quellcode 2.1 dargestellt.

¹ `color_value = video[frame_nr][x_pos][y_pos][color_nr];`

Quellcode 2.1: Beispielhafter Zugriff auf den Farbwerts eines Pixels eines Einzelbilds in einem RGB-Video. Die Variable `color_nr` kann die Werte 0, 1 und 2 für rot, grün und blau annehmen.

Neben RGB kann zur Farb-Darstellung beispielsweise auch die Form Hue, Saturation, Value (HSV) [41, S. 587–593] verwendet werden, wobei Farbton, Sättigung und Helligkeit kodiert werden.

Hinweis: Die oben dargestellte Struktur wird oft erst beim Einlesen der Bilder generiert, die Bild-Dateien liegen meist in komprimierten Formaten wie JPEG [145] vor.

¹ Die bekannteste Alternative zu Rastergrafiken sind Vektorgrafiken. Diese sind anhand einer Reihe von Befehlen aufgebaut und können beliebig skaliert werden. Sie werden beispielsweise verwendet, um Logos oder Piktogramme zu beschreiben, die aus wenigen Grundelementen bestehen. Eine Anweisung wie „Zeichne einen roten Kreis, in dem, in weißer Arial-Schriftart mittig zentriert, der platz-füllende Text <Firma> steht“ kann von jedem Gerät, das den jeweiligen Bild-Standard unterstützt, gezeichnet werden. Für die Speicherung von umfangreichen Kamera-Bildern, die nicht mit wenigen Anweisungen aufgebaut werden können, sind Vektor-Grafiken daher ungeeignet.

2.1.4 Anonymisierung

Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say.

– Edward Snowden [101]

Das Thema Anonymisierung spielt im Bereich der Videoüberwachung eine große Rolle. Nicht nur aus gesellschaftlicher und moralischer Sicht ist es sinnvoll, eine „totale Überwachung“ zu vermeiden. Das Recht auf Privatsphäre wird von einigen Menschen als genauso wichtig erachtet, wie das Recht auf Meinungsfreiheit (siehe Zitat).

Die gesetzlichen Vorgaben fordern für bestimmte Anwendungsfälle unter anderem eine Anonymisierung, die sicherstellen soll, dass Personen im Video nicht identifiziert werden können.

Videos und Bilder können mit verschiedenen Methoden anonymisiert werden, unter anderem mit der im folgenden vorgestellten Bildarithmetik (Bildsubtraktion), bei der sämtliche Bewegungen im Vordergrund des Bilds anonymisiert werden (Bewegungsextraktion).

Alle Bewegungen, also Stellen im Bild, die nicht seit einem bestimmten Zeitraum unverändert sind, werden mit einer Farbe überdeckt, sodass nur noch Konturen zu erkennen sind (siehe Abb. 2.4).

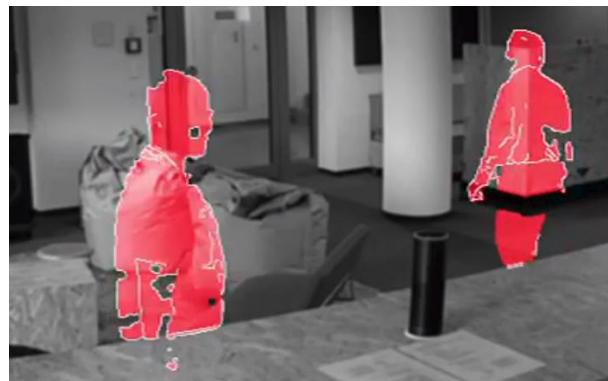


Abb. 2.4: Anonymisierung mittels Bewegungsextraktion: sich bewegende Objekte werden in ein Standbild eingezeichnet, sodass nur noch Konturen zu sehen sind. Das Bild ist ein Ausschnitt eines bei der EnBW zu Testzwecken entwickelten Video-Anonymisierers.

Ein Test während der Entwicklung des in Abb. 2.4 gezeigten Anonymisierers ergab, dass Bewegungen in einigen Fällen gut erkannt und maskiert werden, es aber Situationen gibt, in denen der Algorithmus unzuverlässig arbeitet. In nahen Aufnahmen und Situationen mit schlechten Kontrast-Verhältnissen, beispielsweise wenn die Kleidung von Personen im Vordergrund der Farbe des Hintergrunds sehr ähnlich war, wurden falsche oder weniger Bewegungen in das Video eingezeichnet.

Die Zahl dieser Fehler ließ sich deutlich verringern, indem vor der Bewegungsextraktion verschiedene Methoden der Bildverarbeitung angewendet werden, wie beispielsweise eine Konvertierung in Graustufen, das Anwenden eines Weichzeichners um Rauschen zu verringern, einer Kantenerkennung sowie dem Festlegen einer Mindestgröße für Bewegungen im Bild.

Wie in Abb. 2.4 gezeigt, können trotz der Anonymisierung detaillierte Konturen erkannt werden. Wenn eine stärkere Anonymisierung erreicht werden soll, ist es möglich, Kanten und Details der erkannten Silhouetten mit einem Weichzeichner zu entfernen. [5, S. 416]

Alternative Methoden der Anonymisierung sind das Verpixeln von erkannten Objekten wie Gesichtern und sich bewegenden Stellen im Bild, die ausschließliche Darstellung von Kanten (Kontrastübergängen) und die reduzierte Darstellung von ausschließlich äußeren Konturen der bewegten

Objekte. Für eine maximale Anonymisierung kommt auch eine Reduktion von Personen und Objekten auf Blöcke, Striche oder Punkte in Frage (siehe Abb. 2.5). [157, S. 1674]

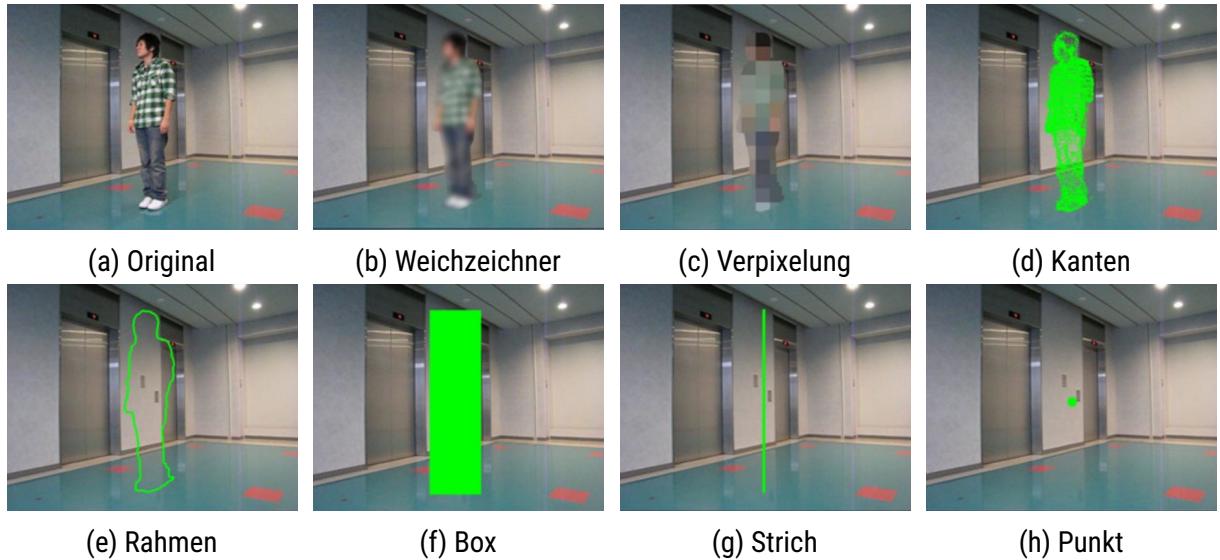


Abb. 2.5: Verschiedene Methoden zur Anonymisierung von Personen. [157, S. 1674]

Es kann unterschieden werden zwischen (1) Anonymisierern, die zu anonymisierende Teile durch Überdeckungen oder Weichzeichner unkenntlich machen (Abb. 2.5b–c) und (2) Anonymisierern, die Bewegungen, Objekte oder Personen erkennen und deren Vorhandensein durch neu generierte Abstraktionen darstellen (Abb. 2.5d–h). Letztere haben den Vorteil, dass bei Versagen der Methode keine Teile des Original-Videos sichtbar werden, was sonst zu einer De-Anonymisierung führen könnte.

Je stärker die Anonymisierung ist, desto größer ist in der Regel auch der mit ihr eingehende Informationsverlust.

Ein Ansatz, der stark anonymisiert und keine Rückschlüsse auf Individuen zulässt, aber dennoch wichtige Hinweise für die Überwachenden gibt, ist die Konvertierung von Video-Szenen in textuelle Situationsbeschreibungen. Statt einem Kamera-Bild könnte beispielsweise der Text „In dem Bild befinden sich drei schwarz gekleidete Männer zwischen 20 und 30 Jahren“ angezeigt werden.

In dieser Arbeit soll dieser Ansatz erweitert werden, indem zusätzlich die Handlungen der erkannten Personen ausgegeben werden. Hierfür wird eine Reduktion von Personen im Bild auf deren Skelette eingesetzt. Dieser Ansatz wird im folgenden Abschnitt näher beschrieben.

2.1.5 Haltungserkennung

Dank moderner Technologien ist es Computern möglich selbstständig Personen in Videos zu detektieren und deren Körperhaltungen abzuschätzen. In den vergangenen Jahren gab es deutliche Fortschritte in der computergestützten Erkennung von Körperhaltungen. [155, S. 1]

Bei der sogenannten „Pose Estimation“ geht es nicht nur darum, die Positionen von Menschen in einem Bild zu erkennen, sondern es wird auch eine detailliertere Lokalisierung von einzelnen Gelenken (*Joints*) vorgenommen [88, S. 242].

Die wahrscheinlich bekannteste Anwendung für computergestützte Handlungserkennung ist das durch die Filmproduktion bekannt gewordenen „Motion Capturing“. Hierbei werden am Körper getragene Sensoren und Marker ausgewertet, um Körper-Bewegungen abzuspeichern. Diese Bewegungen und Skelett-Daten werden anschließend genutzt, um Charaktere in Filmen und Computerspielen zu animieren. [88, S. 232; 46, S. 1–2]

Da nicht in jedem Anwendungsfall Sensoren oder Marker am Körper der beobachteten Personen getragen werden, gibt es bereits zahlreiche Ansätze zur Erkennung von Körperhaltungen in RGB-Bildern [18; 67; 52; 137; 136; 98]. Zu den bekanntesten zählen OpenPose [18], DensePose [52], DeepPose [137] sowie der Ansatz von Tome u. a. [136]. [46, S. 5]

Mithilfe von Computer Vision oder Machine Learning Techniken (siehe Abschnitt 2.2) können, beispielsweise mittels eines Convolutional Neural Network (CNN) im gesamten Bild Muster gesucht werden, die für Menschen und deren einzelne Gelenke typisch sind. Wird ein solches Muster gefunden, wird die Position des Musters zur Lokalisierung des jeweiligen Gelenks genutzt. [46, S. 9]

Dargestellt werden können die erkannten Gelenke als Skelette im RGB-Bild, vor schwarzem Hintergrund oder als Datenstruktur (siehe Abb. 2.6).

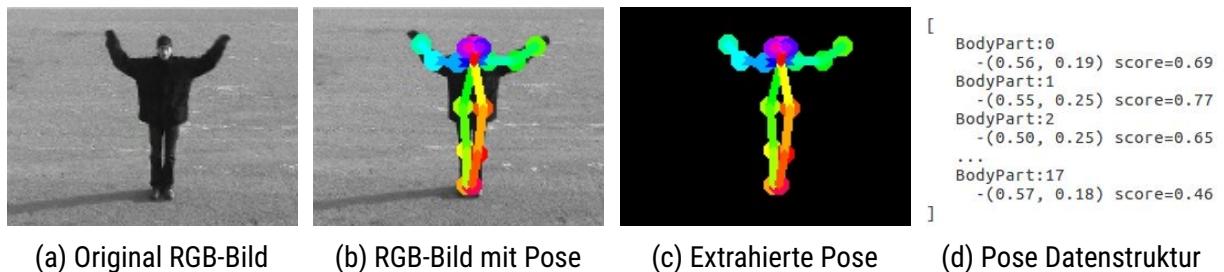


Abb. 2.6: Verschiedene Darstellungsformen für die Haltungserkennung. Das berechnete Skelett kann in das Original-Bild eingezeichnet werden (b), vor einen schwarzen Hintergrund gesetzt werden (c) oder in einer nicht bildlichen Datenstruktur gespeichert werden (d). Das Original-Bild stammt aus dem KTH-Datensatz [109], die Körperhaltung wurde mit dem Algorithmus von Kim [67] berechnet.

Es wird unterschieden zwischen 2D- [18; 137; 67; 98] und 3D- [52] Modellen sowie den Ansätzen modellfrei und modellbasiert [88, S. 243].

Modellbasierte Ansätze [18; 52; 137; 67; 98] wissen bereits, wie ein menschliches Skelett aussehen sollte, während modellfreie Ansätze versuchen, diese Zusammenhänge selbst zu erlernen. Das führt dazu, dass modellbasierte Ansätze auch Gelenke abschätzen können, die im Bild nicht mehr sichtbar sind, da bekannt ist, dass Proportionalitäten zwischen verschiedenen Knochen bestehen, und beispielsweise zwei Beine eines Menschen in der Regel gleich lang sind. [88, S. 243–245; 46, S. 10–12, 19–21]

Zweidimensionale Modelle, wie das aus Abb. 2.6, berechnen eine flache Repräsentation des eigentlich dreidimensionalen Körpers.

3D-Modelle hingegen ergänzen zweidimensionale Fotografien mit 3D-Informationen, in dem sie Wissen aus einem Modell (Modellbasierter Ansatz) oder Stereo-Kamera-Systemen nutzen. Bilder aus letzteren Systemen enthalten Tiefen-Informationen, die zur dreidimensionalen Einordnung eines jeden Pixels im Bild genutzt werden können. Eine alternative Methode zur Berechnung eines 3D-Modells ist das Aufnehmen von Handlungen aus mehreren Perspektiven mit herkömmlichen Kameras.

Das Berechnen von Posen in RGB-Videos ist rechenaufwändiger und fehleranfälliger, als das Verwenden von Tiefen-Informationen [4, S. 7]. In [166] und [35] werden die Körperhaltungen daher aus Bildern mit Tiefen-Informationen gewonnen.

Die Haltungserkennung kann eingesetzt werden, um unabhängig von vermeintlich irrelevanten Dingen wie Kleidung, Hintergrund und Objekten der Szene Handlungen, die aus Körperbewegungen bestehen, zu erkennen. Da die Haltungs- und Handlungserkennung eine Abstraktion von Menschen auf deren Skelette und Handlungen ermöglicht (siehe Abschnitt 3.3), eignet sich diese auch zur Anonymisierung von Videos.

In dieser Arbeit vorangehenden Versuchen wurden zu Forschungszwecken die Bibliotheken OpenPose [18] und DensePose [52] implementiert und hinsichtlich ihrer Performance getestet. In dem Test benötigte DensePose deutlich länger als OpenPose, gab dafür aber auch mehr Gelenke (*Key-points*) und ein damit detaillierteres Skelett zurück.

Ein neuer Ansatz, der an dieser Stelle erwähnt werden soll, ist PoseLab [98], das besonders schnell und robust sein soll, selbst bei in Pixeln gemessen kleinen Personen im Bild. Aus Zeitgründen konnte PoseLab im Rahmen dieser Arbeit nicht implementiert werden.

2.2 Machine Learning (ML)

Machine Learning (ML) ist ein Teilgebiet des Forschungsgebiet „Künstliche Intelligenz (KI)“, in dem versucht wird, Mechanismen zu entwerfen, mit denen Maschinen oder Computer intelligentes Verhalten entwickeln können [25, S. 556].

Der Begriff ML beschreibt eine Technik, bei der Computer-Algorithmen aus vorausgegangen Situationen lernen und Rückschlüsse für neue Situationen schließen [90, S. 1]. Anwendung findet diese Technik in immer mehr Bereichen und wird beispielsweise zur Prognose von Aktienkursen [99], zur Vorhersage von Produktionsausfällen [128] oder zur Klassifikation von auf Bildern dargestellten Objekten [90, S. 7; 30; 71] eingesetzt.

ML besteht aus einer Lern- und Anwendungsphase. In dem Lern- oder auch Trainingsphase genannten Schritt lernt das System aus annotierten (beschrifteten) Daten („Erfahrung“), die es bei der späteren Anwendung zur Nachbildung von kognitiven Fähigkeiten des Menschen einsetzt. [102, S. 1]

Konkret ermöglicht ML beispielsweise die Klassifizierung von Bildern [71] und Videos [66], nur anhand der darin dargestellten Objekte und Bewegungsmerkmale, ohne dass weitere Metadaten wie beispielsweise Schlagworte in der Bild- oder Video-Datei enthalten sein müssen.

Die Funktionsweise von ML wird im Folgenden näher beschrieben.

2.2.1 Lernmethoden

Der Lernalgorithmus bestimmt die Parameter, mit dem das Modell des neuronalen Netzes aufgebaut wird. Allgemein wird zwischen drei Lernmethoden unterschieden. [77, S. 6–7; 90, S. 2]

Beim überwachten Lernen (Englisch: *supervised*) nutzt das Training annotierte Beispieldaten, beispielsweise kategorisierte Bilder oder Videos, um später in der Anwendungsphase andere Bilder in die gelernten Kategorien einzufügen [77, S. 7; 90, S. 2–3].

Beim unüberwachten Lernen (Englisch: *unsupervised*) wird für das Training eine nicht-annotierte Menge von Daten genutzt, die durch den ML-Algorithmus selbstständig in verschiedene Gruppen unterteilt werden [77, S. 7; 90, S. 2, 9]. Dieser Ansatz wird auch benutzt, um Reihen fortzusetzen, beispielsweise werden in einem Versuch von Srivastava, Mansimov und Salakhudinov mehrere auf einen kurzen Video-Ausschnitt folgende Einzelbilder vorhergesagt [123].

Eine letzte Methode ist das sogenannte bestärkende Lernen (Englisch: *reinforcement learning*), bei dem ein System aus Belohnungen und Bestrafungen aufgebaut wird. Während der Anwendungsphase läuft der Trainings-Algorithmus weiter und wird bei guten Entscheidungen belohnt und bei schlechten Entscheidungen bestraft. Der stetige Versuch die Belohnung zu maximieren führt so zu immer besseren Ergebnissen. [90, S. 2]

Für die Erkennung von Handlungen scheint sich besonders das überwachte Lernen zu eignen, da zwischen verschiedenen bereits bekannten Klassen (Handlungen) unterschieden werden soll.

Das unüberwachte Lernen eignet sich hierfür nicht, da die durch den Algorithmus neu erstellte Gruppierung möglicherweise nicht verschiedene Handlungen unterscheiden, sondern beispielsweise dominante Farben oder Objekte gruppieren.

Auch Reinforcement Learning scheint für den Anwendungsfall Handlungs-Klassifikation nicht geeignet zu sein, da zur automatischen Verteilung der Belohnungen und Bestrafungen bereits ein fehlerfreier Handlungs-Klassifikator existieren müsste.

Eine Spezialform des überwachten Lernens ist Multi Instance Learning (MIL), bei dem es statt klassifizierten Gruppen von Beispieldaten Mengen gibt, die jeweils mehrere Klassen enthalten. Für jede Klasse ist bekannt, in welchen der Gruppen diese Klasse vorkommt. Bei der Handlungsklassifikation könnte MIL eingesetzt werden, um Videos, die mehr als eine Handlung enthalten, für das Training zu verwenden [84, S. 3, 5; 39]. MIL hilft zudem dabei, unbalancierte Trainingsdaten auszugleichen [84, S. 5].

2.2.2 Klassifikation

Zur Veranschaulichung, wie eine Klassifikation mithilfe von neuronalen Netzen funktioniert, wird zunächst die grundlegende Funktionalität einer Support Vector Machines (SVMs) beschrieben. SVM definieren eine Grenzfunktion, die zwei Klassen voneinander trennt. Liegt ein Wert über der Funktion, so gehört er zur ersten Klasse, liegt er darunter, so gehört er zur zweiten und nicht zur ersten Klasse. Die Achsen des Funktions-Graphen sind mit Merkmalswerten beschriftet, die zur Klassifikation genutzt werden. [14, S. 148]

In Abb. 2.7 wird zur Veranschaulichung eine beispielhafte SVM gezeigt.

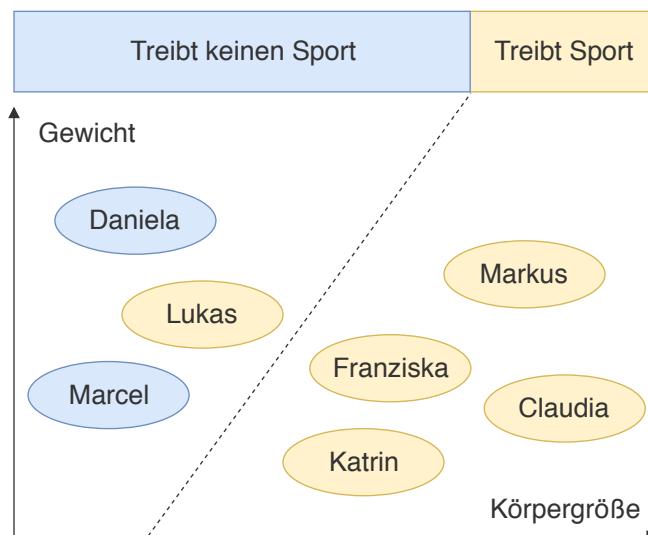


Abb. 2.7: Support Vector Machine (SVM) zur Klassifikation von Personen, die Sport treiben (gelb) und unsportlichen Personen (blau). Hierfür werden die Merkmale Gewicht und Größe verwendet. Durch den Klassifikator wird jeder, dessen Größe um einen bestimmten Wert oder Prozentsatz höher ist, als sein Gewicht, als Sportler klassifiziert. Während der Trainingsphase wird versucht, die Trennlinie (gepunktet) an die richtige Position zu setzen. Die Person Lukas ist Sportler, wird durch die aktuelle Trennlinie aber nicht als solcher klassifiziert. Dies ist ein Zeichen dafür, dass die gewählten Merkmale möglicherweise nicht zur Klassifikation geeignet oder nicht ausreichend sind.

Bei einer klassischen SVM werden die Merkmale zur Unterscheidung der verschiedenen Objekte manuell entworfen, weshalb diese, wie in Abb. 2.7 gezeigt, nicht immer sinnvoll sind.

Zur selbstständigen Gewichtung möglicher Merkmale werden künstliche neuronale Netze eingesetzt.

2.2.3 Künstliche neuronale Netze

Künstliche neuronale Netze sind eine Möglichkeit, Computern zu beschreiben, wie „lernen“ funktioniert.

Die Idee hierbei ist es, die Funktionsweise des menschlichen Gehirns nachzubilden. Neuronale Netze bestehen im Gehirn hauptsächlich aus Neuronen des Nervensystems und Synapsen, die diese miteinander verbinden. Die Neuronen sind in verschiedenen Schichten (*layers*) angeordnet und werden mithilfe von Reizen gesteuert. [102, S. 30, 36; 77, S. 4]

Auch die im Computer nachgebildeten, künstlichen neuronale Netze bestehen aus Neuronen und Synapsen. Während komplexe Netze aus mehreren Millionen Neuronen bestehen, genügen für ein einfaches Beispiel bereits zwei Neuronen (siehe Abb. 2.8). [102, S. 31]

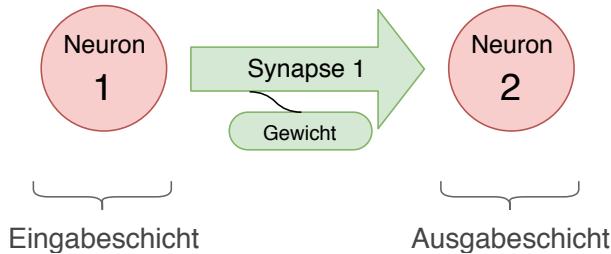


Abb. 2.8: Minimales neuronales Netz aus zwei Neuronen und einer Synapse.

In der in Abb. 2.8 gezeigten Eingabeschicht (*input layer*) liegt ein Neuron, das Eingangsdaten entgegen nimmt und diese an alle Synapsen weiterleitet. Jede Synapse hat ein bestimmtes Gewicht (*weight*), mit dem alle Eingangsdaten multipliziert werden, bevor sie an das Neuron in der Ausgabeschicht (*output layer*) weitergegeben werden. [77, S. 5]

Die Ausgabe kann also verändert werden, indem die Gewichte der Synapsen geändert werden. Machine Learning Algorithmen versuchen während des Trainings, die Gewichte so oft anzupassen, bis die Ausgabe für alle Eingabewerte zu den jeweils passenden Ausgabewerten führt.

Dieser Vorgang kann anhand des folgenden Beispiels veranschaulicht werden. Es wird angenommen, dass die Elemente der Menge der Eingabewerte $x = \{1, 2, 5\}$ auf die Ausgabewerte $f(x) = \{2, 4, 10\}$ abgebildet werden sollen. Da das in Abb. 2.8 gezeigte neuronale Netz aus nur einer Synapse besteht, gibt es nur ein Gewicht, das zum Erreichen der Ausgabewerte angepasst werden kann: $f(x) = S_1 * x$. Diese Formel lässt sich bereits mit mathematischen Grundkenntnissen auflösen, die Funktion ist für die gegebenen Beispielwerte genau dann wahr, wenn für das Synapsen-Gewicht $S_1 = 2$ gilt. [87, S. 22]

Der Machine Learning Algorithmus versucht, das Gewicht ohne diese mathematischen Kenntnisse zu bestimmen. Während der Trainings-Phase werden die Gewichte solange schrittweise um einen gewissen Wert (Lernrate, *step size*) erhöht oder verringert, bis die gewünschten Ausgabewerte erreicht werden.

In der Praxis bestehen neuronale Netze häufig aus mehreren Ein- und Ausgabe-Neuronen. Wie in Abschnitt 2.1.3 beschrieben wurde, besteht ein RGB-Bild aus einer bestimmten Anzahl Pixel, die jeweils aus drei Farbwerten zusammengesetzt sind. Soll ein solches Bild klassifiziert werden, gibt

es für jeden Pixel-Farbwert ein Eingabe-Neuron. Für jede zu unterscheidende Klasse gibt es ein Ausgabe-Neuron, das die Wahrscheinlichkeit für die jeweilige Klasse angibt.

Damit eine sinnvolle Verknüpfung der Ein- und Ausgabeneuronen auch bei umfangreicherer Anwendungsfällen möglich ist, werden fortgeschrittenere Netze mit weiteren Synapsen und Neuronen zwischen der Eingabe- und der Ausgabeschicht eingesetzt (siehe Abb. 2.9) [77, S. 10, 16].

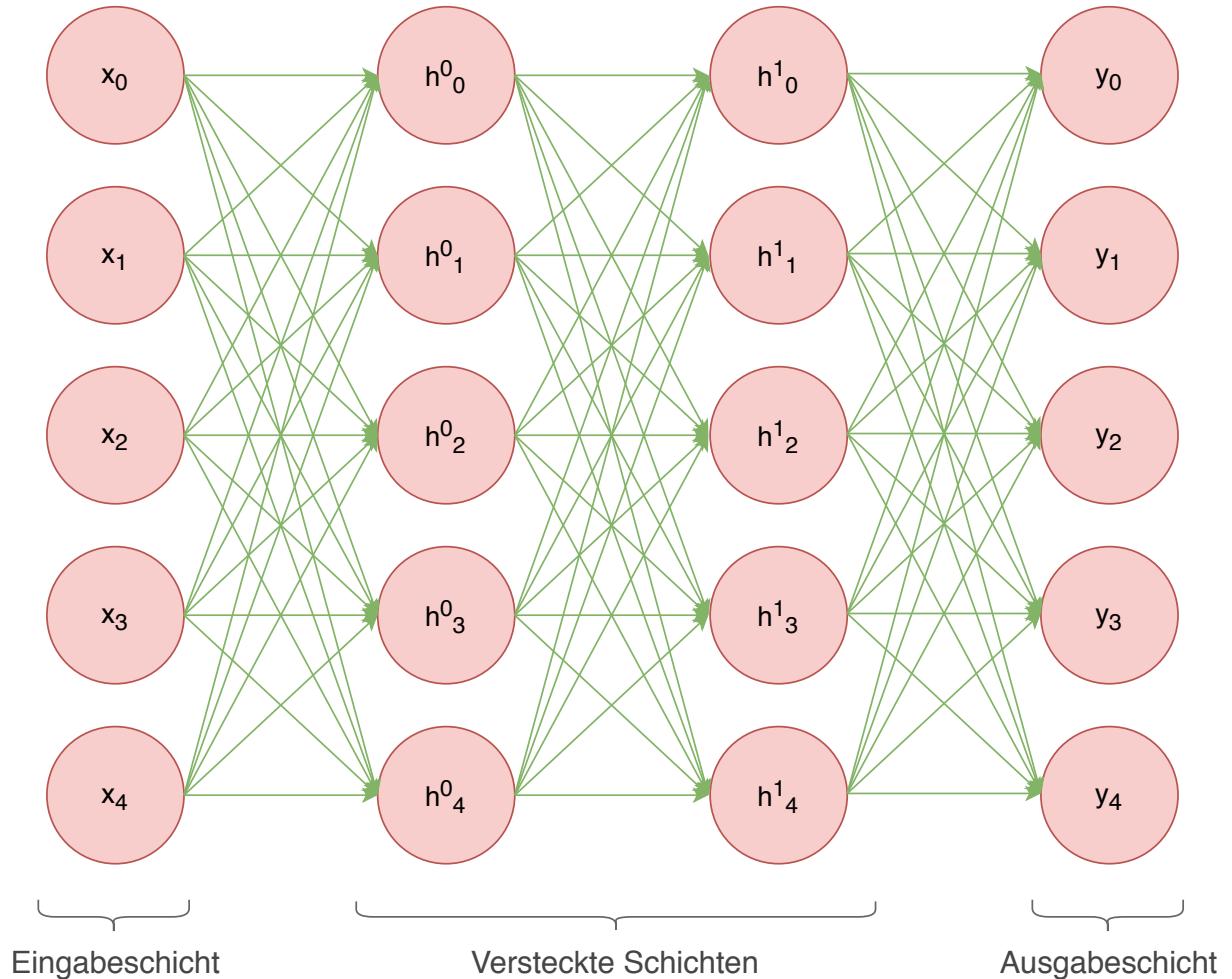


Abb. 2.9: Ein aus drei Schichten bestehendes neuronales Netz (der Eingangsvektor in der Eingabeschicht wird nicht gezählt) mit je fünf Ein- und Ausgabeneuronen. Jede Schicht ist vollständig mit den benachbarten Schichten verbunden (*fully connected layers*). [77, S. 16]

In solchen größeren Netzen gibt es deutlich mehr Synapsen und damit Gewichte, die berechnet werden können. Anfangs werden die einzelnen Gewichte meist mit Zufallswerten bestückt. Anschließend werden die Gewichte, wie im letzten Beispiel gezeigt, durch „ausprobieren“ so lange verändert, bis ein gutes Ergebnis erzielt wird. [102, S. 4–5]

In größeren neuronalen Netzen erhält ein Neuron also Eingabewerte von mehreren Synapsen. Die verschiedenen Werte werden miteinander verrechnet und anschließend mithilfe einer sogenannten „Aktivierungsfunktion“ normalisiert. Zur Diskretisierung werden die einzelnen Ausgaben der Vorgänger-Neuronen mit dem jeweiligen Synapsen-Gewicht multipliziert und anschließend summiert. Für das Neuron h_1^0 aus Abb. 2.9 wird beispielsweise mit dem Synapsen-Gewicht $w(a, b)$ zwischen zwei Neuronen a und b die folgende Summe berechnet: [102, S. 49]

$$h_1^0 = x_0 * w(x_0, h_1^0) + x_1 * w(x_1, h_1^0) + x_2 * w(x_2, h_1^0) + x_3 * w(x_3, h_1^0) + x_4 * w(x_4, h_1^0) \quad (2.1)$$

Um den Eingangswert des Neurons zu erhalten, wird nun diese Summe an die Aktivierungsfunktion übergeben. Je nach Art des neuronalen Netzes wird hierfür eine andere Funktion genutzt. Gängig ist es, eine lineare Funktion, eine binäre Stufenfunktion oder eine Sigmoid-Funktion zu verwenden (siehe Abb. 2.10). [102, S. 33–34]

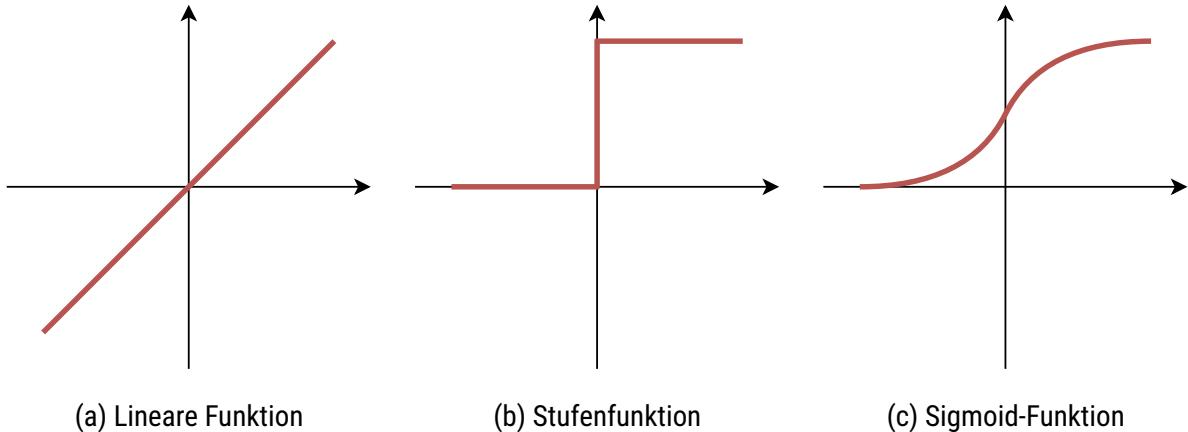


Abb. 2.10: Verschiedene Aktivierungsfunktionen für künstliche neuronale Netze. Die Skalierung der Achsen variiert je nach Implementation, oft wird der Wertebereich [0,1] gewählt. Die x -Achse (horizontal) beschreibt den Eingangswert, die y -Achse (vertikal) die Funktion $f(x)$ (rot) als Ausgangswert. [102, S. 33–34; 77, S. 5]

Die Idee hinter den Aktivierungsfunktionen entstammt der Biologie, da auch menschliche Neuronen von einem Schwellwert abhängig aktiviert werden [102, S. 31–32].

Um die eben beschriebenen Rechen-Operationen für ein komplettes neuronales Netz durchzuführen, genügt es, die Matrix aller Eingabewerte mit der Matrix aller Gewichte zu multiplizieren und die Aktivierungsfunktion anschließend auf die sich ergebende Matrix anzuwenden [102, S. 49–50].

In tiefen neuronalen Netzen, die aus sehr vielen Schichten bestehen, werden auch Rectified Linear Units (ReLUs) als Aktivierungsfunktion genutzt [2]. Eine ReLU-Funktion $f(x)$ steigt für alle $x \geq 0$ linear mit der Funktion $f(x) = x$. Für die Werte $x < 0$ wird je nach ReLU-Implementierung entweder $f(x) = 0$ gesetzt oder es wird eine deutlich schwächer steigende Funktion, wie $f(x) = 0,01 * x$, verwendet. [62]

2.2.4 Convolutional Neural Networks (CNNs)

CNN sind eine spezielle Form der eben vorgestellten neuronalen Netze. Bei CNN sind die Neuronen einer Schicht nicht nur eindimensional in einer Spalte, sondern in einer zweidimensionalen Matrix, auch Faltungsmatrix genannt, dargestellt. Die Synapsen werden in diesem Modell Kernel genannt und berücksichtigen nun mehrere, in der Matrix nebeneinander liegende Neuronen auf einmal (siehe Abb. 2.11). [64]

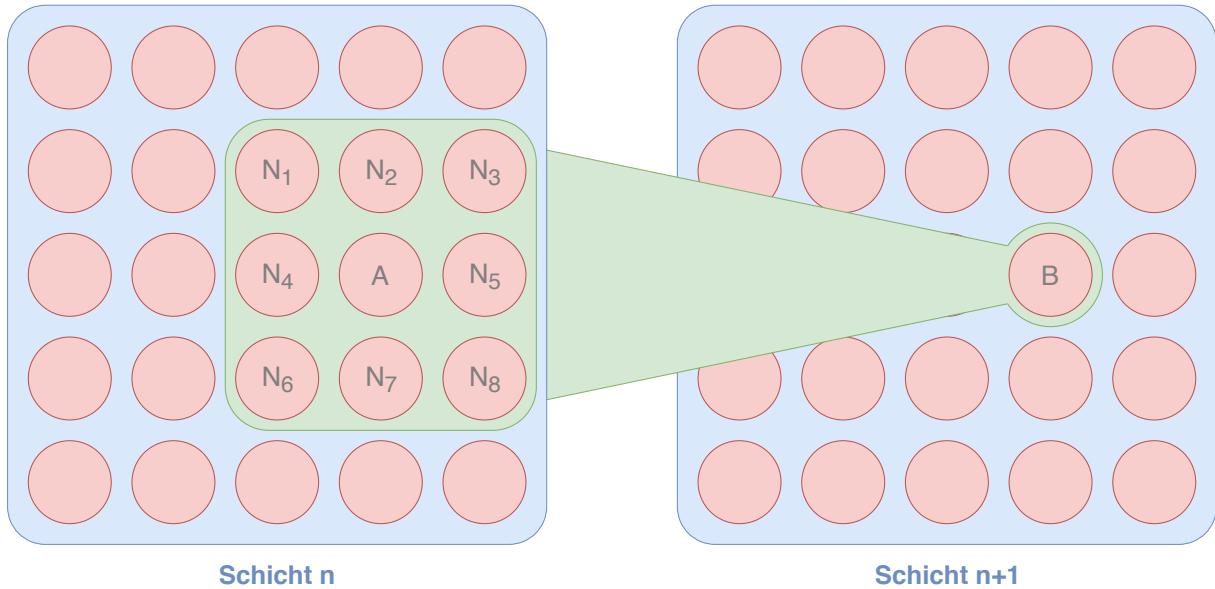


Abb. 2.11: Convolutional Neural Network (CNN). Zur Berechnung der Neuronen der Schicht $n + 1$ (zum Beispiel B) werden das jeweilige Vorgänger-Neuron (A) und seine Nachbar-Neuronen ($N_1 \dots N_8$) berücksichtigt. Diese Matrix (grün) wird Kernel genannt.

Zur Berechnung der Neuronen wurde im vorigen Abschnitt die Formel $f(x) = S_1 * x$ genutzt. Es wurden also das Synapsen-Gewicht S_1 mit dem Vorgängerwert x multipliziert. Bei CNN passt das gleiche, allerdings werden sowohl für das Gewicht als auch für den Vorgängerwert nun Matrizen verwendet, weshalb zur Diskretisierung noch das Ermitteln eines arithmetischen Mittels hinzukommt. Mit den Bezeichnungen aus Abb. 2.11 sowie dem Gewicht des Kernels G_K lässt sich die folgende Formel aufstellen:

$$B = \frac{\text{sum} \left(G_K * \begin{pmatrix} N_1 & N_2 & N_3 \\ N_4 & A & N_5 \\ N_6 & N_7 & N_8 \end{pmatrix} \right)}{9} \quad (2.2)$$

Um diese Formel zu veranschaulichen, werden Beispielwerte eingesetzt:

$$B = \frac{\text{sum} \left(\begin{pmatrix} 1 & 1 & 1 \\ 1 & 90 & 1 \\ 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{pmatrix} \right)}{9} = 94,4 \quad (2.3)$$

In diesem Beispiel ist das zentrale Neuron A im Gewicht des Kernels am stärksten gewichtet. Die diskretisierende Funktion, hier das arithmetische Mittel, wird *Pooling-Funktion* genannt und ist je nach Anwendungsfall unterschiedlich.

Die in der Praxis am häufigsten verwendete Pooling-Funktion verwendet statt dem arithmetischen Mittel (Mean-Pooling) das Maximum der Matrix (Max-Pooling). Eine weitere, vor allem für die Bildklassifikation verwendete Methode, ist Softmax-Pooling, bei dem die Softmax-Funktion (auch normalisierte Exponentialfunktion genannt) [10, S. 198] eingesetzt wird. [64]

CNN können effizient zur Klassifikation von Bildern eingesetzt werden [66, S. 1; 161, S. 2; 39, S. 1; 32, S. 1; 71]. Ein beispielhaftes CNN zur Klassifikation von Bildern ist in Abb. 2.12 gezeigt.

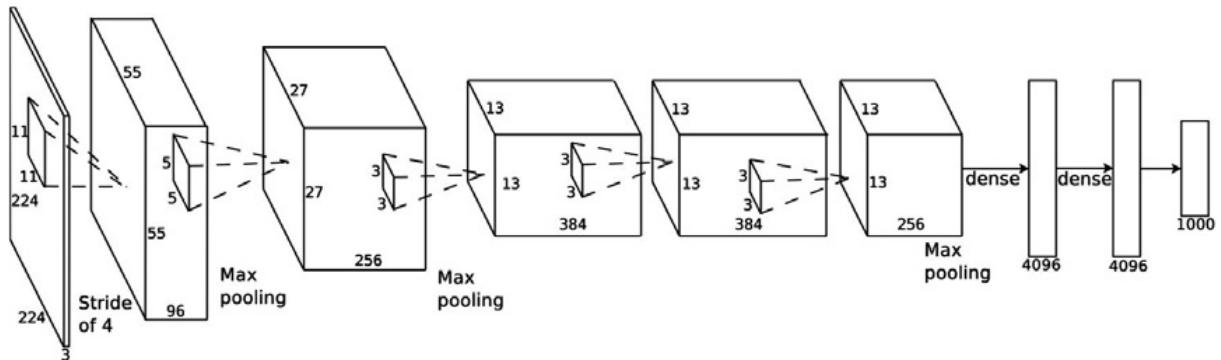


Abb. 2.12: Beispielhafte Architektur eines CNNs zur Bildklassifikation. Das CNN besitzt acht Schichten, zwischen denen mittels verschieden großer Kernel mehrere Neuronen zusammengefasst werden (aus [120, S. 91]).

Bilder können, wie bereits in Abschnitt 2.1.3 gezeigt, als Matrix repräsentiert werden. Um ein ganzes Bild zu verarbeiten, „fährt“ der Kernel nun über die gesamte Matrix, also über alle Neuronen der ersten Schicht und gibt die Ergebnisse an die nächste Schicht weiter. [64]

Die versteckten Schichten eines CNN werden genutzt, um mehrere Neuronen zusammenzufassen. Bei einem Bild kann man sich das als Zusammenfassen von Bildausschnitten vorstellen. [64]

Die Aufgabe des Trainings ist weiterhin das Bestimmen der Gewichte. [64]

3D-CNN

Die bisher vorgestellten CNN sind gut zur Klassifikation von Bildern geeignet, können aber nicht für Videos eingesetzt werden, da hierfür eine drei- statt zweidimensionale Kernel-Architektur benötigt würde [139, S. 1]. CNN, die dreidimensionale Kernel verwenden, werden 3D-CNN genannt [139, S. 1]. Mithilfe dieser ist es möglich, zeitliche Bezüge innerhalb des neuronalen Netzes zu berücksichtigen [61, S. 2; 139, S. 1].

In Abb. 2.11 wurde gezeigt, wie der Kernel einen Teil der zweidimensional angeordneten Neuronen zusammenfasst. Zur Visualisierung eines 3D-CNNs müssten in Abb. 2.11 für jede Schicht n_t weitere Schichten n_{t-1}, n_{t-2}, \dots hinter die Schicht eingezeichnet werden, wobei die weiteren Schichten für jedes Neuron die jeweiligen Vorgänger-Werte enthalten [61, S. 3].

Der 3D-Kernel berücksichtigt nun auch die Werte der nahen Vergangenheit, bei der Auswertung von lang andauernden Zusammenhängen stoßen aber auch 3D-CNN an ihre Grenzen [61, S. 3; 31, S. 1].

2.2.5 Recurrent Neural Networks (RNNs)

RNNs berücksichtigen im Gegensatz zu den bisher vorgestellten Netzen nicht nur die aktuelle Eingabe, sondern auch mehrere Eingabewerte, die zuvor durch das Netz geschickt wurden [36; 65].

Hierdurch können neue Aufgaben wie das Erstellen von Sätzen, Übersetzungen oder Trendanalysen gelöst werden. RNNs werden darüber hinaus zur Klassifikation von Videos und zum Verarbeiten menschlicher Sprache eingesetzt [63].

Die Architektur von RNN ist dynamisch, weshalb im Unterschied zu den bereits vorgestellten neuronalen Netzen Ein- und Ausgaben verschiedener Größen mit dem selben Netz verarbeitet werden können. Angewendet wird dieser Vorteil unter anderem bei der Generierung und Verarbeitung von Sequenzen mit dynamischer Länge. [63]

In Abschnitt 2.2.3 wurde gezeigt, dass ein neuronales Netz aus mehreren Schichten besteht. Jede dieser Schicht besteht aus mehreren Neuronen, die je einen Skalar verarbeiten (siehe Abb. 2.13a). In RNN-Neuronen können Vektoren verarbeitet werden, weshalb ein Neuron in einem RNN die Funktionalität einer ganzen Schicht eines klassischen neuronalen Netzes abbilden kann (siehe Abb. 2.13b) [63].

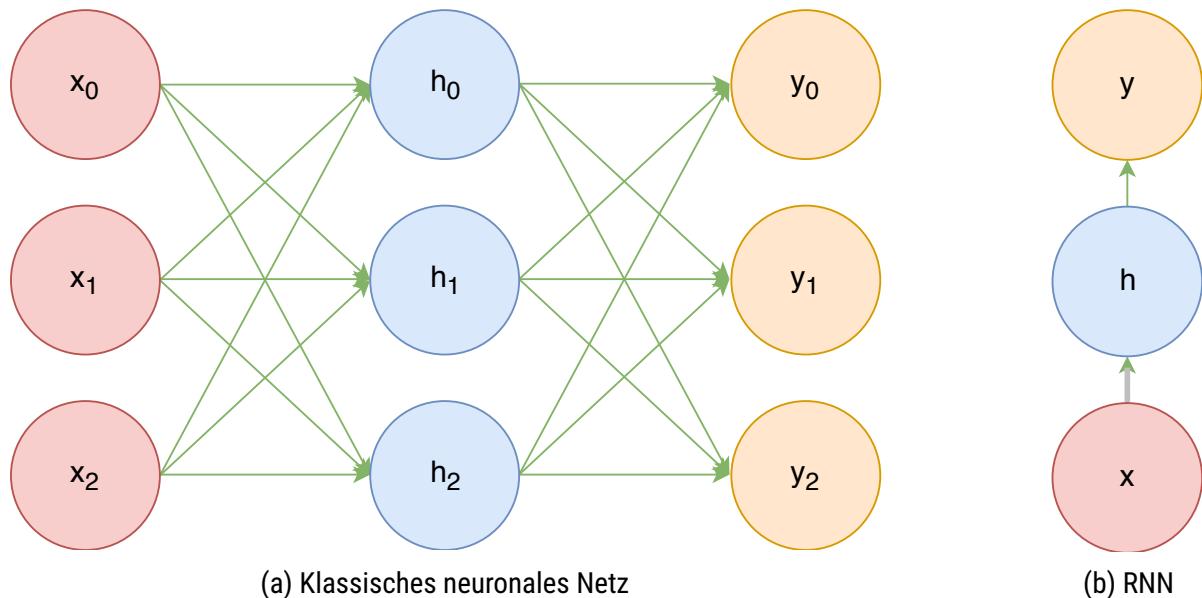


Abb. 2.13: Darstellung eines (a) klassischen neuronalen Netzes als (b) RNN. (nach [63])

RNN können zur Klassifikation von Sequenzen benutzt werden (*many to one*, siehe Abb. 2.14a), zur Generierung von Sequenzen anhand einer festen Eingabe (*one to many*) oder zur Generierung von Sequenzen anhand einer Sequenz (*many to many*, siehe Abb. 2.14b) [65; 63].

Zur Klassifikation von Handlungen könnte ein *many to one* RNN eingesetzt werden, da dieses anhand mehrerer Einzelbilder (*many*) eine Klasse (*one*) bestimmen kann. *Many to many* RNN können beispielsweise zum Übersetzen von Texten zwischen verschiedenen Sprachen oder zum Generieren einer textuellen Beschreibung für ein Video eingesetzt werden. [65; 63]

Eine Spezialform von RNNs sind die Bidirectional RNNs (BRNNs), die eine zusätzliche rückwärts laufende Schicht besitzen. Zur Visualisierung eines BRNN müssten in Abb. 2.14b alle von links nach rechts laufenden Pfeile der oberen versteckten Schicht (blau eingezzeichnet) umgekehrt werden.

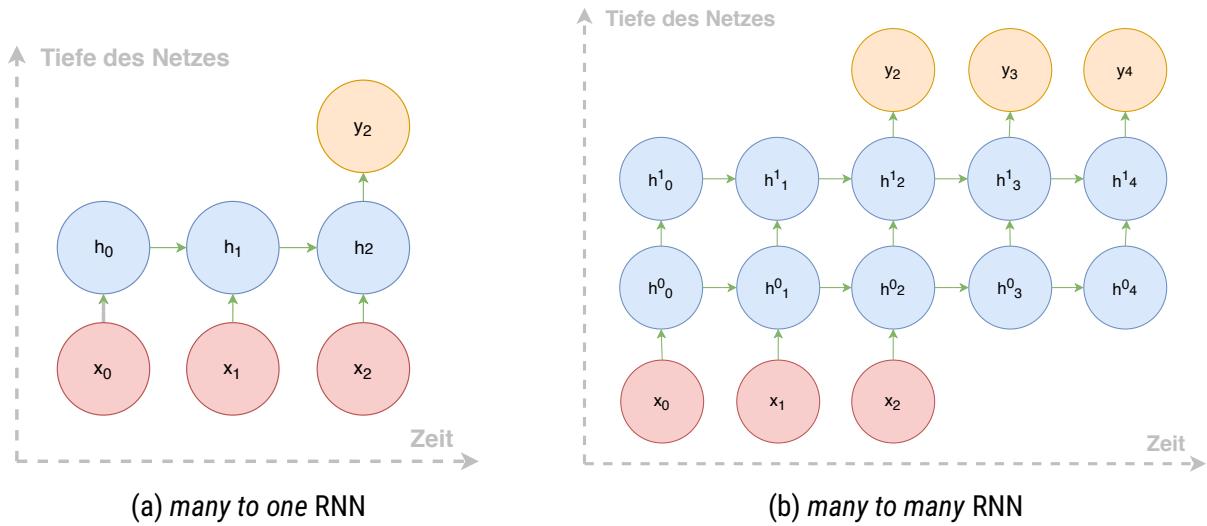


Abb. 2.14: Verschiedene Typen von RNN (nach [65; 63]).

Bei RNN tritt das sogenannte „Vanishing Gradient“ Problem auf, das den Zeitraum in den zurückgeschaut wird, einschränkt [32, S. 2].

Zur Berechnung der Gewichte eines RNN ist das Bestimmen von Minima einer Funktion in einem mehrdimensionalem Raum notwendig. Hierfür kommt ein „Gradientenverfahren“ zum Einsatz, dass Ableitungen und deren Steigungen berechnet. Mit einer steigenden Anzahl an Neuronen sind die einzelnen Steigungen weniger steil.

Um die kleinen Steigungen darzustellen, sind viele Nachkommastellen notwendig. Da in Computern die Größe von Zahlen sowohl nach oben (*overflow*) als auch nach unten (*underflow*) beschränkt ist, werden Steigungen ab dem Unterschreiten eines gewissen Grenzwertes fälschlicherweise als Null angegeben. Für ein RNN können dann keine Gewichte mehr berechnet werden, was ein Training verhindert. [32; 63; 65; 62]

Long Short-Term Memory (LSTM)

Eine Möglichkeit zur Vermeidung des „Vanishing Gradient“ Problems ist die Verwendung von sogenannten Long Short-Term Memory (LSTM) Zellen [54, S. 1; 96]. In den LSTM Zellen befinden sich verschiedene Gatter [54, S. 7; 32, S. 2], die es ermöglichen, Zustände weiterzugeben, diese zu modifizieren, zu aktualisieren, zurückzusetzen [32, S. 2] oder mit der Zeit zu summieren [123, S. 3].

LSTMs ermöglichen es daher, Zusammenhänge und Merkmale aus längeren Zeiträumen zu detektieren [123, S. 3].

Werden in einem RNN LSTM-Zellen verwendet, so spricht man auch von LSTM-Netzen.

LSTM eignen sich gut für Spracherkennung [50] und Sprachübersetzung [129; 23] sowie zur Auswertung und Prognose von Werten im zeitlichen Verlauf [96], beispielsweise zum Erkennen einer Dokumentstruktur oder Grammatik.

3 Handlungserkennung

Der Begriff Handlungserkennung (englisch: *human activity recognition, action recognition*) wird in dieser Arbeit als Überbegriff für das Detektieren und Klassifizieren von Handlungen verwendet.

Ziel dieser Arbeit ist es, zu prüfen, ob durch Auswertung von Körperhaltungen in einem Video Handlungen erkannt werden können.

In den letzten Jahren gab es deutliche Fortschritte im Bereich der Verarbeitung von statischen Bildern, weshalb Aufgaben wie Bild-Klassifizierung oder Objekt-Erkennung bereits bald gelöst sein könnten [116, S. 1; 149, S. 2].

Wenn es nun aber um die Erkennung von Videos geht, wird in der Forschung noch über grundlegende Fragen diskutiert: [116, S. 1]

- „*Was ist eine Handlung und wie sollen wir sie repräsentieren?*“
- „*Welche räumliche und zeitliche Grenzen haben Handlungen?*“
- „*Welche Rolle spielen Ziele und Absichten bei Definition und Verständnis von Handlungen?*“

Dieses Kapitel ist in mehrere Abschnitte unterteilt. Zu Beginn dieses Kapitels wird grundlegend beschrieben, wie maschinelles Lernen zur Klassifikation von Handlungen eingesetzt werden kann.

Anschließend werden einige der bisher veröffentlichten Implementierungen beschrieben und miteinander verglichen. Die Untersuchung der verschiedenen Ansätze soll einen Überblick über den aktuellen Stand der Technik schaffen und der Ermittlung eines Ansatzes, der sich als Grundlage für die praktische Implementierung eignet, dienen.

Wie bereits im Titel dieser Arbeit erkennbar, soll die in dieser Arbeit entwickelte Handlungserkennung auf einer Auswertung von Körperhaltungen basieren. Wie diese Körperhaltungen erstellt werden können wurde bereits in den Grundlagen auf Seite 9 beschrieben. In diesem Abschnitt soll gezeigt werden, wie diese zur Klassifikation von Handlungen genutzt werden können und welche verschiedenen Methoden es zur Strukturierung dieser Daten gibt.

In Abschnitt 3.4 werden zahlreiche Datensätze aufgezeigt, die zur Klassifikation von Handlungen genutzt werden können sowie aktuelle Herausforderungen hinsichtlich der Datensätze beschrieben.

Im darauf folgenden Abschnitt wird beschrieben, welche Schritte notwendig sind, um einen allgemeinen Algorithmus zur Handlungserkennung auf den einleitend beschriebenen Anwendungsfall anzupassen.

Wie sich herausstellen wird, sind die bereits entwickelten Ansätze noch nicht perfekt. Zum Abschluss dieses Kapitels werden daher die aktuellen Herausforderungen im Bereich der Handlungserkennung aufgezeigt.

3.1 Grundlagen

In diesem Abschnitt wird beschrieben, wie maschinelles Lernen zur Klassifikation von Handlungen eingesetzt werden kann.

3.1.1 Interpretation von Einzelbildern (Raum)

Zunächst wird die Interpretation von Einzelbildern ohne zeitlichen Kontext betrachtet. Hierfür kommen beispielsweise Bildklassifikatoren und Objekterkenner zum Einsatz. Im Bereich der Bildklassifikation gibt es schon seit einigen Jahren effiziente Modelle zur Klassifikation. [71]

Ein typischer Bildklassifikator, der auf maschinellem Lernen basiert, lernt in einem ersten Schritt anhand von klassifizierten Beispielen (Trainingsdaten) Merkmale, die zur Unterteilung der Bilder in die verschiedenen Klassen hilfreich sind. Im zweiten Schritt versucht der Klassifikator, ein bislang nicht klassifiziertes Bild, anhand dieser Merkmale, einer Klasse zuzuordnen.

Eine Möglichkeit zur Erkennung von Handlungen ist nun, in der Lernphase eine Klasse für jede Handlung zu erstellen und Beispieldaten für jede dieser Klassen zu sammeln. In der Anwendungsphase kann anschließend das RGB-Bild, das eine noch nicht klassifizierte Handlung enthält, an den Klassifikator gegeben werden.

Bei der Auswahl der Trainingsdaten sollte darauf geachtet werden, dass ausreichend viele und für die spätere Anwendung passende und vielfältige Bilder verwendet werden.

In dieser Arbeit soll eine Klassifikation von Handlungen durch Auswertung der Körperhaltungen von Personen in einem Video vorgenommen werden. Zu diesem Zwecke können (1) Bilder generiert werden, auf denen die Körperhaltung dargestellt ist, und (2) diese Bilder für das Training und die Anwendung des Klassifikators genutzt werden. Dieser Ansatz wird in Abschnitt 3.3 näher beschrieben.

Durch eine reine Auswertung der Einzelbilder kann vermutlich keine hohe Genauigkeit bei der Klassifikation von Handlungen erreicht werden, da Handlungen über einen gewissen Zeitraum ausgeführt werden und zeitliche Aspekte in diesem Abschnitt nicht berücksichtigt wurden.

3.1.2 Interpretation von Bewegungsdaten (Zeit)

Zeitliche Zusammenhänge spielen beim Erkennen von Handlungen eine große Rolle [4, S. 2]. Die zeitliche Entwicklung einer Szene kann durch den Einbezug von vorherigen Einzelbildern berücksichtigt werden. Hierfür können 3D-CNN, RNN und LSTM eingesetzt werden (siehe Abschnitt 2.2).

Um Bewegungsinformationen auch mithilfe von CNN zu klassifizieren, wurde der sogenannte Zwei-Kanal-Ansatz [118] entwickelt. Ein Video lässt sich in die zwei Dimensionen Raum (Bild) und Zeit (Verlauf) unterteilen. Dieser Ansatz basiert auf der biologischen Idee, dass auch der visuelle Cortex des Menschen in zwei Hälften unterteilt werden kann. Das ventrale visuelle System erkennt Objekte und der dorsale Fluss Bewegungen. [47]

Der hierfür verfolgte Ansatz ist eine Trennung der beiden Dimensionen Raum und Zeit in eigene neuronale Netze, die jeweils nur eindimensionale Klassifikationen vornehmen, und später miteinander kombiniert werden (siehe Abb. 3.1) [118; 167; 39, S. 2].



Abb. 3.1: Vereinfachte Darstellung des von Simonyan und Zisserman vorgeschlagen Zwei-Kanal-Ansatzes [118]. Das obere, räumliche, CNN ordnet RGB-Einzelbildern je eine Handlungs-Klasse zu. Das zeitliche Bild wird im unteren CNN für die zeitliche Dimension ebenfalls klassifiziert und auf eine Handlung abgebildet. Abschließend werden die beiden Beschriftungen kombiniert, um bei der finalen Klassifikation sowohl zeitliche als auch räumliche Informationen zu berücksichtigen.

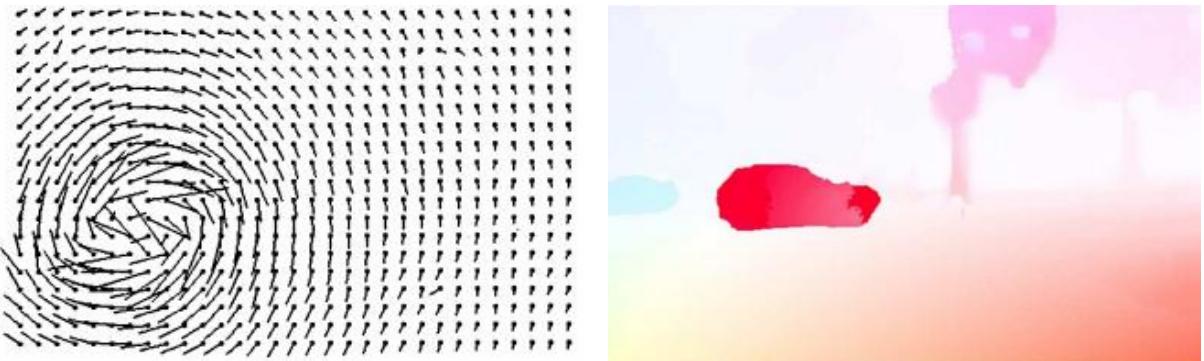
Damit bei der Klassifikation der zeitlichen Dimension bestehende Bildklassifikatoren eingesetzt werden können, wird versucht, ein statisches Bild zu generieren, das mithilfe von extrahierten Bewegungsinformationen den zeitlichen Verlauf des Videos beschreibt [118].

Das zweite CNN, dass diese Bewegungsinformationen anschließend interpretiert, ist weitgehend unabhängig von dem CNN, dass das RGB- oder Skelett-Bild klassifiziert [118].

Die klassifizierten Bewegungs-Informationen werden abschließend mit den klassifizierten Bild-Informationen kombiniert, um eine Raum- und Zeit-Klassifikation zu erhalten [118].

Zur Bestimmung der Merkmale, die zur Unterscheidung des zeitlichen Verlaufs von Handlungen geeignet sind, gibt es verschiedene Methoden, die im weiteren Verlauf dieses Abschnitts vorgestellt werden.

Optischer Fluss (HOF) Um zeitliche Bewegungen in einem statischen Bild zu visualisieren, kann die Technik des optischen Flusses verwendet werden. Hierbei werden für feste Punkte im Bild Vektoren (*Flußvektoren, Trajektorien* [146]) berechnet, die die Bewegungsrichtungen über einen kurzen Zeitraum repräsentieren [55] (siehe Abb. 3.2a). Diese Darstellung wird Histogram of Optical Flow (HOF) genannt [27].



(a) Berechnen von Vektoren für eine Rotationsbewegung (aus [55, S. 20]). Die Länge der Vektoren gibt die Geschwindigkeit der Bewegung an.
 (b) Dichte Berechnung des optischen Flusses und farbliche Darstellung der Bewegungen. Das Bild zeigt ein fahrendes Auto (aus [110, S. 3892]).

Abb. 3.2: Visualisieren von Bewegungen mittels optischem Fluss.

Die Vektoren können in das Originalbild eingezeichnet werden, um die Zusammenhänge zwischen räumlicher und zeitlicher Ebene hervorzuheben. Statt unterschiedliche Längen von Vektoren zu verwenden, können diese auch eingefärbt werden, um die Stärke der Bewegung zu verdeutlichen (siehe Abb. 3.2b).

Bei dem in Abb. 3.5 gezeigten Ansatz generiert das „MotionNet-CNN“ ein Bild, das den optischen Fluss beschreibt [167, S. 3].

In Abb. 3.4b wird der Handlungs-Klassifikator von Wang und Schmid [146] gezeigt, die zur Extraktion von zeitlichen Merkmalen den optischen Fluss verwendet.

Videos können in verschiedenen Formaten und Kodierungen gespeichert werden. In einigen Fällen benutzen die Kodierungen Bewegungsvektoren, die beschreiben, welches Verhalten die einzelnen Pixel in den nächsten Einzelbildern zeigen werden. Die Auswertung dieser Bewegungsvektoren kann zu einer schnelleren Berechnung des optischen Flusses führen. [161]

Bewegte Umrisse (HOG) Eine alternative Darstellungsform für derartige zeitliche Verläufe verwendet Kanten-Übergänge, die durch ein Histograms of Oriented Gradients (HOG) repräsentiert werden [26]. Durch diese Methode können Objekte im Vordergrund vom Hintergrund getrennt werden (siehe Abb. 3.3).



Abb. 3.3: Verschiedene Stufen bei der Erstellung eines HOG. In den Bildern auf der linken Seite werden interne Gewichte des zum Ermitteln des HOG genutzten SVM visualisiert. In den rechten drei Bildern sind verschiedene Varianten des zum mittigen Bild gehörenden HOG dargestellt (aus [26]).

Anschließend werden die sich bewegenden Objekte im zeitlichen Verlauf überlagert dargestellt, beispielsweise mit den von Gorelick u. a. vorgestellten „Space-Time Shapes“ [49]. Diese stellen die Umrisse von Objekten im Vordergrund im zeitlichen Verlauf dar. Durch die verschiedenen Silhouetten können alle Bewegungen, die für die jeweilige Handlung notwendig sind, in einem Bild visualisiert werden (siehe Abb. 3.4a).



Abb. 3.4: Verschiedene Ansätze zur Visualisierung von Bewegungsdaten. In (a) sind die Handlungen „Hampelmann“, „gehen“ und „rennen“ im zeitlichen Verlauf dargestellt, in (b)(1) wurden zwei aufeinanderfolgende Bilder überlagert, in (b)(2) wurde der optische Fluss zwischen den zwei Einzelbildern visualisiert und in (b)(3) wurden die zugehörigen Bewegungsflüsse (Trajektorien) eingezeichnet, wobei die Kamera-Bewegungen weiß markiert sind (3.4a aus [49, S. 2] und 3.4b aus [146, S. 1]).

Zur Extraktion der Objekte im Vordergrund kann auch die in Abschnitt 2.1.4 genutzte Bewegungs-Extraktion verwendet werden [49, S. 1].

Da sich die einzelnen Silhouetten überlagern, ist dieser Ansatz allerdings nur für kurze und weniger komplizierte Handlungen geeignet.

Kombination vom optischen Fluss und Umrissen (MBH) Bei den sogenannten Motion Boundary Histograms (MBHs) werden die beiden Ansätze HOF und HOG miteinander kombiniert, um Übergänge zwischen horizontalen und vertikalen Komponenten des optischen Flusses getrennt voneinander zu verarbeiten. Hierdurch wird die Performance verbessert [118, S. 2] und die Erkennungsrate durch eine Hintergrund-Extraktion bei Videos erhöht [27, S. 5, 9–12].

Mittels CNN gelernte Merkmale Eine Alternative zu den vorgestellten Methoden zur manuellen Extraktion von zeitlichen Merkmalen ist das Verwenden von neuronalen Netzen, die selbstständig zum Beschreiben der zeitlichen Verläufe relevante Merkmale lernen. Diese Methode wurden in den Ansätzen [139], [66], [125] und [9] verwendet. [167, S. 1]

Die Bestimmung der Merkmale mittels CNNs ist zunächst langsamer, als der manuelle Entwurf von Merkmalen, dafür aber dynamisch und daher für eine größere Zahl von Anwendungsfällen einsetzbar.

3.2 Untersuchung bisheriger Implementierungen

Es wurden bereits zahlreiche Methoden zur Handlungserkennung entwickelt. Die Methoden basieren auf verschiedenen Ansätzen und kombinieren eine große Auswahl an verschiedenen Techniken. Eine Einordnung und Bewertung ist daher nicht trivial und im Rahmen dieser Arbeit nur begrenzt möglich.

In den Grundlagen (Kapitel 2) wurden bereits verschiedene Typen von neuronalen Netzen vorgestellt. Die Methoden dieses Abschnitts lassen sich zunächst anhand des jeweils verwendeten Netzes in die Gruppen CNN, RNN und LSTM einordnen. Darüber hinaus gibt es zahlreiche Ansätze, die auf einer Kombination der verschiedenen Netztypen basieren.

Die meisten der untersuchten Implementierungen verwenden CNN und setzen auf den Zwei-Kanal-Ansatz (siehe 3.1.2), der 2014 von Simonyan und Zisserman [118] entwickelt wurde, unter anderem [21; 32; 45; 158; 123; 143; 151; 149; 39; 167].

Im Zwei-Kanal-Ansatz von Zhu u. a. [167] findet keine Vorberechnung von Merkmalen, die den zeitlichen Verlauf beschreiben, statt. Stattdessen werden die Bilder des optischen Flusses mithilfe eines CNN generiert (siehe Abb. 3.5).

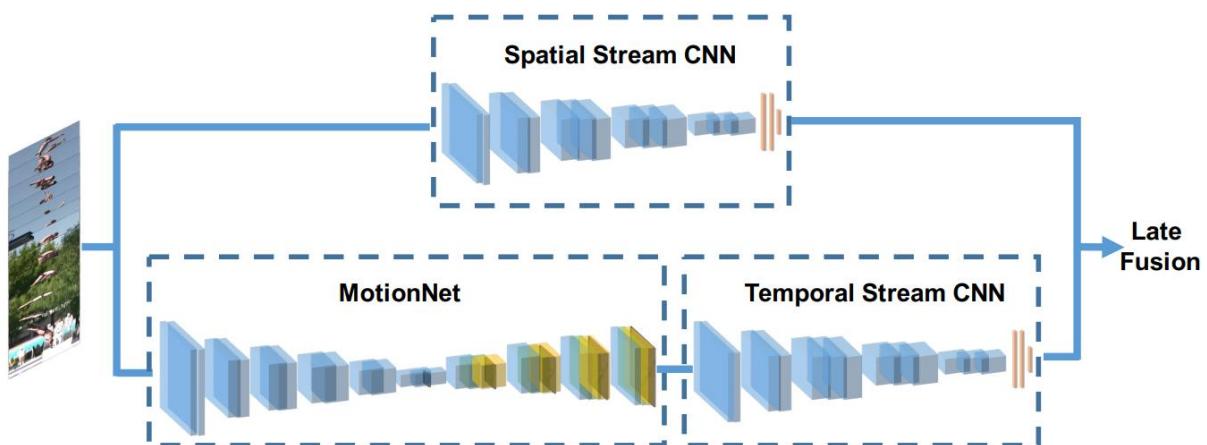


Abb. 3.5: Skizze des in [167] vorgeschlagen Zwei-Kanal-CNNs. Das obere räumliche CNN ordnet Einzelbildern je eine Handlungs-Klasse zu. Das MotionNet-CNN generiert aus jedem Video ein Bild, das den optischen Fluss des Videos beschreibt (in der Skizze gelb-braune Blöcke). Dieses Bild wird im darauf folgenden Kanal-CNN der zeitlichen Dimension ebenfalls auf eine Handlung abgebildet. Anschließend werden die beiden Beschriftungen (in der Skizze braune Stäbchen) kombiniert, um sowohl zeitliche als auch räumliche Informationen zu berücksichtigen. (aus [167, S. 1])

Bei der Untersuchung der verschiedenen Implementierungen des Zwei-Kanal-Ansatzes konnten zwei Nachteile identifiziert werden. Erstens ist es nicht möglich, zwischen räumlichen und zeitlichen Merkmalen Zusammenhänge auf Pixel-Ebene herzustellen, da getrennt klassifiziert wird und erst anschließend die beiden Klassifikations-Ergebnisse fusioniert werden. Zweitens kann das zeitliche CNN nicht für beliebig große Zeiträume verwendet werden, da die Größe des Stapels von zeitlich adjazenten optischen Fluss Bildern begrenzt ist [39, S. 2–3].

Um diesen Nachteil auszugleichen, wurden in [118] in regelmäßigen Abschnitten nur kurze Teile des Videos extrahiert. Für die Modellierung der zeitlichen Entwicklung von Handlungen ist das aber nicht ausreichend [39, S. 2–3].

Etablierte CNN-Ansätze wie [118] eignen sich daher gut für die Auswertung der räumlichen Dimension (des Aussehens) und Bewegungen der kurzen Vergangenheit, scheitern aber an der Berücksichtigung von längeren zeitlichen Strukturen [149, S. 2].

Seit 2015 gibt es verschiedene Ansätze längere zeitliche Strukturen mittels CNNs auszuwerten (zum Beispiel [142; 158; 32]). Die hierfür primär verwendete Methode versucht die zeitliche Dimension in kleine Stücke zu zerteilen (*dense temporal sampling*), wobei die Länge der Stücke im Voraus festgelegt wird [149, S. 2]. Die Methode hat den Nachteil, dass der Rechenaufwand bei längeren Videos sehr hoch ist und es besteht die Gefahr, dass Informationen, die länger als die maximale Videosequenz-Dauer sind, verloren gehen [149, S. 2]. Der Ansatz von Donahue u. a. [32] verwendet reine RGB-Bilder zur Klassifikation.

3D-CNN wurden im Bereich der Handlungserkennung erstmalig von Ji u. a. [61] eingesetzt. Zwei Jahre spätere wurde dieser Ansatz von Tran u. a. [139] weiterentwickelt. Durch den Einsatz von tieferen Netz-Architekturen konnten bessere Ergebnisse erzielt werden [139, S. 8].

Weitere Ansätze verwenden RNN mit LSTM-Zellen, um zeitliche Zusammenhänge besser zu berücksichtigen.

Eine Übersicht über einige der bisher veröffentlichten Implementierungen ist in Tabelle 3.1 gezeigt.

Jahr	Typ	Implementierung	Kommentar
2011	Mix	Sequential deep learning for ... [6]	3D-CNN zur Merkmalsextraktion und LSTM zur Klassifikation von Kurzvideos (15 Einzelbilder) mit je einer Handlung, händisch auf Person zugeschritten.
2013	3D-CNN	3D CNN for Human Action...[61]	Räumliche und zeitliche Auswertung, KTH [109] Datensatz, Zuschnitt auf Personen
2014	CNN	Large-scale Video Class...[66]	Getrennte Verarbeitung von niedrig- und hochauflösten Bildern sowie Überprüfung mehrerer Zeitpunkte, an denen zeitliche Informationen berücksichtigt werden können.
2014	CNN	Finding action tubes [45]	Finden von bewegungsreichen Zonen im Bild und Handlungs-Suche in diesen durch Klassifikation von RGB und opt. Fluss
2014	CNN	Two-Stream CNN for Action...[118]	Aufteilung des Videos in Zeit- und Raum-Kanal (opt. Fluss und RGB, siehe Abschnitt 3.1.2)
2015	CNN	P-CNN: Pose-based CNN...[21]	Bewegungs- und Aussehens-Daten von Körperteilen (Skelett-Stücken) interpretieren
2015	3D-CNN	Learning spatiotemporal...[139]	neue Architektur und Merkmale
2015	CNN	Learning to track for spatio-...[151]	Erkennt Objekte in Einzelbildern, klassifiziert diese zu Handlungen, verfolgt die wahrscheinlichsten Detektionen im Video und klassifiziert in Kombination mit einem Sliding Window für die zeitliche Ausmaße der Handlung
2015	RNN	Hierarchical RNN for...[35]	Skelettbasiert mit Unterteilung in Körperteile
2015	Mix	Beyond short snippets...[158]	(a) Verschiedene CNN-Methoden zur zeitlichen Fusion, (b) Handlungen mittels LSTM, das CNN-Ausgabe einliest.
2015	Mix	Sequence to sequence...[143]	Beschreibungen zu Videos, LSTM lernt zeitliche Strukturen und Satzbau anhand der Ausgaben eines CNN für RGB-Bilder und/oder optische Fluss Bilder
2016	CNN	Learning Models for Actions...[84]	Kombination der Konturen um Personen mit dem Einzelbild und anschließendes MIL
2016	CNN	Machine Learning for...[113]	Mehrkanal CNN mit ImageNet Basis
2016	CNN	Temporal Segment Net...[149]	Unterteilt Video in zeitliche Abschnitte um lange Zusammenhänge zu erkennen, nutzt Temporal Segment Networks (TSM)
2016	CNN	Convolutional Two-Stream...[39]	Verbesserter Zwei-Kanal-Ansatz mit anderer Vereinigungsstrategie MIL
2016	LSTM	Unsupervised Learning...[123]	LSTM-Netz, das (a) supervised Eingaben klassifiziert oder (b) unsupervised Videos rekonstruiert kann (encoder-decoder) und fortführen (vorhersagen) kann. Enthält Info über zeitlichen Aufwand.
2016	LSTM	Co-occurrence Feature ...[166]	Skelettbasiert mit neuem Dropout Algorithmus
2016	Mix	Long-term Recurrent CNN...[32]	LSTM mit einem CNN kombiniert: Long-term Recurrent Convolutional Network (LRCN)
2016	Mix	A Multi-Stream Bi-Direct...[119]	Mehrkanal CNN gefolgt durch eine bidirektionale LSTM Schicht
2016	Mix	Regularizing LSTM with 3D...[83]	LSTM mit CNN Basis, mit 3D Skelett Daten, Backpropagation through time (BPTT) und einem Encoder-LSTM
2017	CNN	Chained Multi-stream Net...[168]	Mehrkanal 3DCNN mit Markovketten, berücksichtigt Körperhaltung, Bewegungen und Aussehen
2017	CNN	Long-term Temporal Conv...[142]	Faltungen für längere Zeiträume ohne Aufteilen, dafür weniger räumliche Auflösung (LTC-CNN)
2017	CNN	Hidden Two-Stream CNN...[167]	Two-Stream-Ansatz ohne explizite opt. Fluss Berechnung (end to end) und ca. 10 mal schneller als [118]
2017	3D-CNN	Temporal 3D CNN: New...[31]	mit variabler zeitlicher Tiefe und Transfer-Learning aus 2D-Daten
2017	LSTM	Every Moment Counts...[156]	Mehrere Eingabe- und Ausgabe-Verbindungen
2018	CNN	Action Recognition with...[155]	Skelettbasierte Handlungserkennung mit Aufmerksamkeitssteuerung, GLAN und SSAN

Tabelle 3.1: Übersicht über bisherige Implementierungen.

3.2.1 Vergleich

Um die verschiedenen in dieser Arbeit untersuchten Implementierungen zu vergleichen, wurden mehrere Kriterien definiert und für die untersuchten Implementierungen geschätzt, wie gut diese die einzelnen Kriterien erfüllen.

Ein vollständiger Test mit vergleichbaren Benchmark-Tests konnte im Rahmen dieser Arbeit nicht durchgeführt werden, da das Implementieren der Netz-Architekturen sowie das Trainieren der neuronalen Netzen zu viel Zeit in Anspruch nehmen würde (siehe Abschnitt 3.6).

Zur Unterscheidung der Implementierungen wurden die Kriterien Präzision P (in einem aktuellen Datensatz), Geschwindigkeit G , Komplexität K und Funktionalität F gewählt. Für den in Abschnitt 1.2 beschriebenen Anwendungsfall sollten primär alle funktionale Anforderungen abgedeckt werden. Hierzu zählen eine Erkennung durch anonymisierte Körperhaltungen F_K , das Erkennen von mehreren Personen F_P und Handlungen F_H in einem Bild sowie die Analyse eines kontinuierlichen Livestreams F_L , wofür es notwendig ist, zeitlich aufeinanderfolgende Handlungen voneinander zu trennen.

Beim Berücksichtigen der nicht-funktionalen Anforderungen sollte in erster Linie eine hohe Präzision und hohe Geschwindigkeit erreicht werden. Die Komplexität der Implementierung sollte darüber hinaus möglichst gering sein. Die Eignung der jeweiligen Implementierung für den Anwendungsfall wird in einem Score S zusammengefasst, der sich wie in Formel 3.1 gezeigt zusammensetzt.

$$\begin{aligned} S = & F_K * 10\% + F_P * 10\% + F_H * 10\% + F_L * 10\% \\ & + P * 20\% + G * 20\% + (1 - K) * 20\% \end{aligned} \quad (3.1)$$

Bei Implementierungen, für die keine Geschwindigkeits-Werte geschätzt werden konnten, gilt $G = 50\%$. In der Tabelle sind diese mit der Abkürzung keine Angabe (k. A.) angegeben.

Die Ergebnisse des Tests werden in Tabelle 3.2 gezeigt.

Eine Auswertung der Testergebnisse findet in Abschnitt 3.2.2 statt.

Implementierung	F_K	F_P	F_H	F_L	P	G	K	S
Unsupervised Learning...[123]	0%	0%	0%	0%	60%	k. A.	70%	28%
Sequential deep learning for...[6]	0%	0%	0%	0%	32%	k. A.	20%	32%
Sequence to sequence-...[143]	0%	30%	0%	0%	50%	k. A.	50%	33%
Machine Learning for...[113]	0%	0%	0%	0%	60%	k. A.	45%	33%
Learning Models for Actions...[84]	0%	30%	40%	0%	50%	k. A.	60%	35%
Beyond short snippets...[158]	0%	0%	0%	0%	63%	k. A.	30%	37%
Long-term Recurrent CNN...[32]	0%	0%	0%	30%	70%	k. A.	50%	37%
3D CNN for Human Action...[61]	10%	70%	30%	0%	30%	k. A.	50%	37%
Temporal 3D CNN: New...[31]	0%	0%	0%	10%	74%	k. A.	40%	38%
Large-scale Video Class...[66]	0%	0%	0%	0%	63%	k. A.	20%	39%
Convolutional Two-Stream...[39]	0%	0%	0%	0%	73%	k. A.	30%	39%
Two-Stream CNN for Action...[118]	0%	0%	0%	0%	65%	k. A.	20%	39%
Finding action tubes [45]	0%	30%	30%	0%	60%	k. A.	40%	40%
Learning spatiotemporal...[139]	0%	0%	0%	0%	50%	100%	40%	42%
Regularizing LSTM with 3D...[83]	100%	30%	0%	0%	65%	k. A.	60%	44%
Temporal Segment Net...[149]	0%	0%	70%	50%	75%	k. A.	60%	45%
Learning to track for spatio-...[151]	30%	30%	30%	70%	51%	k. A.	50%	46%
Long-term Temporal Conv...[142]	0%	0%	0%	50%	73%	100%	60%	48%
P-CNN: Pose-based CNN...[21]	100%	30%	0%	0%	65%	k. A.	40%	48%
Hidden Two-Stream CNN...[167]	0%	0%	0%	0%	74%	100%	30%	49%
Co-occurrence Feature...[166]	100%	30%	0%	0%	80%	k. A.	50%	49%
A Multi-Stream Bi-Direct...[119]	0%	30%	100%	30%	70%	k. A.	50%	50%
Action Recognition with...[155]	100%	70%	70%	0%	50%	k. A.	65%	51%
Every Moment Counts...[156]	0%	30%	100%	30%	80%	k. A.	50%	52%
Hierarchical RNN for...[35]	100%	30%	0%	0%	80%	100%	50%	59%
Chained Multi-stream Net...[168]	100%	30%	30%	0%	65%	100%	30%	63%

Tabelle 3.2: Vergleich der untersuchten Implementierungen. Der Score S ist ein gewichteter Durchschnitt der Erfüllungsgrade der einzelnen Testkriterien Präzision P , Geschwindigkeit G , Komplexität K und Funktionalität (Auswertung mittels Körperhaltung F_K , Berücksichtigung mehrerer Personen F_P , Berücksichtigung mehrerer Handlungen F_H , Anwendung in Livestreams F_L). Eine detaillierte Bedeutung der Kürzel sowie die Berechnung des Scores kann dem Text in Abschnitt 3.2.1 entnommen werden.

3.2.2 Auswertung der Untersuchung

In diesem Abschnitt werden die, bei der Untersuchung der verschiedenen Ansätze, gewonnenen Kenntnisse zusammengefasst und aktuelle Tendenzen bei der Entwicklung von Algorithmen zur Handlungserkennung beschrieben.

Bei der Forschung im Bereich der Handlungserkennung werden weiterhin verschiedene Netz-Architekturen verwendet. Die, anhand des Score S gemessen, besten drei der in Tabelle 3.2 gezeigten Ansätze basieren auf CNN [168], RNN [35] und LSTM [156]. Aber auch an Ansätzen mit 3D-CNN (beispielsweise [31]) oder hybriden Ansätzen (beispielsweise [83]) wird weiterhin entwickelt.

Die besten Geschwindigkeiten wurden mit 3D-CNN und CNN erreicht. Mit modernen Grafikkarten wurden teilweise über 4.000 Einzelbilder pro Sekunde verarbeitet [142, S. 7].

Die höchste Präzision bei der Klassifikation wurde mit LSTM-Zellen erreicht [166, S. 5].

Der Ansatz von Yang u. a. [167] deckt die meisten der funktionalen Anforderungen ab. Neben der Erkennung und Auswertung von Körperhaltungen arbeitet diese Implementierung auch mit sogenannten *attention maps*, mit denen es möglich ist, für die Klassifikation relevante Bereiche räumlich und zeitlich zu lokalisieren. Die *attention maps* werden durch ein CNN generiert, die Handlungsklassifikation erfolgt über eine Spezialform eines RNN mit LSTM-Zellen. [155, S. 3–4]

Sowohl für den, am Score gemessenen, besten Ansatz, als auch für den Ansatz, der die meiste Funktionalität abdeckt, wurde kein Quellcode zur Implementierung der vorgestellten Architektur veröffentlicht. Eine Implementierung dieser Architekturen im Rahmen dieser Arbeit wäre sehr zeitaufwendig, weshalb in Kapitel 4 ein eigener Ansatz entwickelt wird, der verschiedene Aspekte der untersuchten Ansätze miteinander kombiniert.

3.3 Handlungserkennung mithilfe von Körperhaltungen

Menschliche Handlungen erfordern immer eine gewisse Bewegung des Körpers. Diese Körperbewegungen sind Veränderungen der menschlichen Haltung im Lauf der Zeit. In dieser Arbeit wird untersucht, ob eine Abstraktion einer Handlung auf die hinter dieser stehenden Körperhaltungen zur Klassifikation von Handlungen ausreicht.

Ein naheliegender Vorteil, der durch diese Abstraktion entsteht, ist, dass sich während des maschinellen Lernens auf den „wesentlichen“ Teil des Videos konzentriert werden kann, und nicht erst gelernt werden muss, welche Objekte im Bild von der Handlung unabhängig sind. So spart man nicht nur Zeit beim Training des neuronalen Netzes, sondern es werden auch weniger Datensätze benötigt, da weniger Hintergrund-Szenarien abgedeckt werden müssen.

Wenn die Hintergründe in die Klassifikation einfließen, kommt es bei gleicher Datensatz-Größe schneller zu Überanpassung (*overfitting*). Ein Algorithmus, der Sportarten klassifiziert, könnte aufgrund der gelernten Videos beispielsweise davon ausgehen, dass Basketball immer in einer Halle gespielt werden muss und Fußball immer auf einem Feld im Freien. Bei der Klassifikation eines Videos, das ein in einer Halle stattfindendes Fußballspiel zeigt, würde der Algorithmus bei einer Überanpassung inkorrekt die Handlung Basketball klassifizieren. Das kann durch eine Abstraktion auf Körperhaltungen verhindert werden.

Der zeitliche Gewinn durch die kürzeren Trainingszeiten wird durch den Zeitaufwand, der für das Ermitteln der Skelette erforderlich ist geschwächt. Eine manuelle Detektion der Körperhaltungen wäre für die Auswertung mehrerer langer Videos zu aufwendig und automatische Methoden waren lange Zeit rechenintensiv und ungenau. Mittlerweile gibt es aber immer bessere Ansätze um Skelette abzuschätzen, beispielsweise OpenPose [18] und DensePose [52] (siehe Abschnitt 2.1.5).

Wenn nur die Körperhaltung ausgewertet ist, so ist es unter Umständen schwer, Handlungen wie „sich an einen Tisch setzen“ und „sich auf einen Stuhl setzen“ voneinander zu unterscheiden. Hierfür ist ein Einbezug der Objekte, mit denen interagiert wurde, notwendig. In einigen weiteren Fällen sind im Bild vorkommende Objekte zudem starke Indizien dafür, dass eine bestimmte Handlung ausgeführt wird. Das Vorhandensein eines Fußballs erhöht die Wahrscheinlichkeit, dass in dem Video die Handlung „Fußball spielen“ vorhanden ist.

Für diese Fälle kommt eine Kombination der Skelett-Erkennung mit den Kontext-Informationen, die durch das Kamerabild gewonnen werden, in Frage. Wenn sowohl das Kamerabild, als auch die Körperhaltung klassifiziert werden, beispielsweise mit einem Zwei-Kanal-Ansatz (siehe Abschnitt 3.1.2), so lässt sich der Vorteil der Abstraktion mit der Flexibilität des Original-Bilds kombinieren.

Auch eine Kombination einer Haltungserkennung mit einer dedizierten Objekterkennung wäre denkbar.

3.3.1 Strukturierung der Körperhaltungs-Daten

In diesem Abschnitt soll beschrieben werden, welche Darstellungsformen zur Strukturierung von Skelett-Daten verwendet werden können. In Abschnitt 2.1.5 wurde bereits in Abb. 2.6 gezeigt, dass für jede Gelenk-Position eine x- und eine y-Koordinate angegeben werden kann. Mithilfe dieser Positionen ist es möglich, die Gelenke in das Bild einzulezeichnen.

Um auch Verbindungslien zwischen den Gelenken einzulezeichnen, ist ein Vorwissen über die menschliche Anatomie erforderlich, etwa in Form einer Liste von zu verbindenden Gelenk-Paaren oder in Form einer Hierarchie, die zeigt, welche Gelenke und Körperteile miteinander verbunden sind.

Eine anatomische Hierarchie kann auch bei der Klassifikation von Handlungen berücksichtigt werden. Du, Wang und Wang unterteilen die Gelenke in ihrem Ansatz [35] in fünf Gruppen, die für verschiedene Körperteile stehen. Anschließend werden die Körperteile mittels mehrerer neuronaler Netze, im konkreten Fall BRNNs, interpretiert und zusammengesetzt (siehe Abb. 3.6).

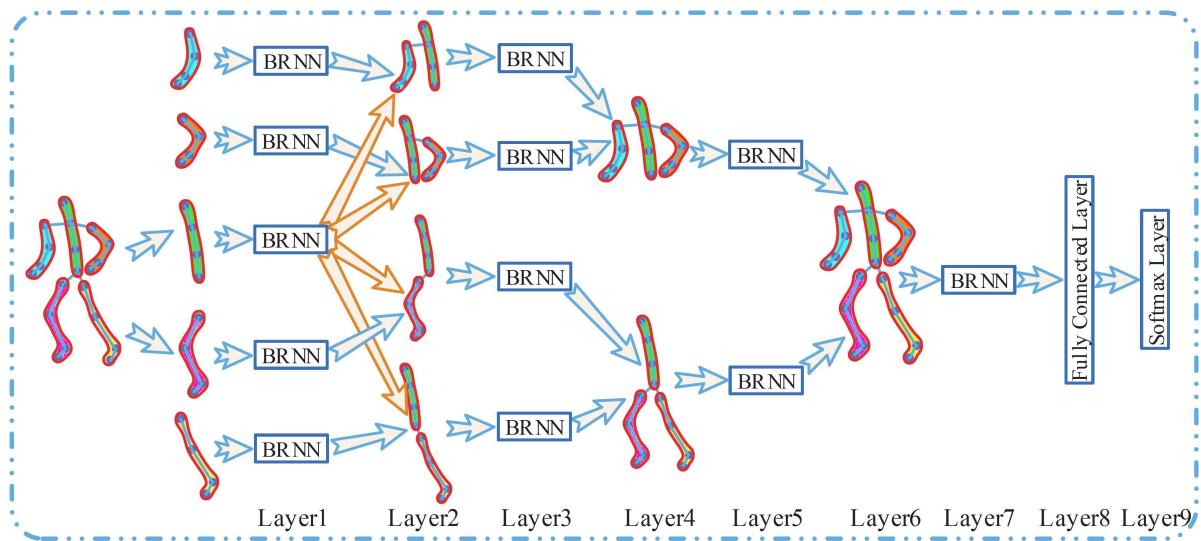


Abb. 3.6: Skizze der in [35] vorgeschlagenen Skelett-Hierarchie. Das Skelett wird in fünf Gruppen unterteilt, die zunächst in je einem BRNN verarbeitet werden. Die BRNNs der folgenden Schichten kombinieren je zwei Ergebnisse der vorherigen Schichten, wobei die Schichten bewusst so kombiniert werden, dass anatomisch nahe Gruppen zuerst vereint werden. Abschließend werden zur Klassifikation der Handlung eine voll-verknüpfende Schicht sowie eine auf der Softmax-Funktion basierende Schicht eingesetzt. (nach [35, S. 1110])

Eine Gruppe des Skeletts aus Abb. 3.6 besteht je nach benutztem Datensatz aus vier bis acht Gelenken, die jeweils als relative Vektoren zur Position der Hüfte angegeben werden [35, S. 1115-1117].

Die Hierarchie in Abb. 3.6 ist statisch und manuell definiert. Für den menschlichen Körper, dessen individuelle Ausprägungen sich hinsichtlich der Anordnung von Gelenken nur geringfügig unterscheiden, scheint eine manuell definierte Hierarchie auszureichen.

Für ein universelles Modell, dass neben diesem idealisierten Menschen auch Kinder, Kleinwüchsige, Menschen mit körperlichen Behinderungen oder sogar Tiere berücksichtigt, ist eine manuelle Definition der Gelenke aber nicht zielführend. In [80] wird daher ein Netzwerk vorgeschlagen, das die anatomischen Beziehungen zwischen den Gelenken mittels maschinellem Lernen selbstständig

erlernt. Das beschriebene 2D LSTM Netzwerk, auch Spatio-Temporal LSTM (ST-LSTM) Netzwerk genannt, lernt während der Trainings-Phase sowohl zeitliche als auch räumliche Beziehungen.

Das berechnete Skelett kann je nach Anwendungsfall entweder in das von der Kamera stammende Original-Bild oder auf einen einfarbigen Hintergrund eingezeichnet werden. Wenn es abzusehen ist, dass sich die Handlungs-Klassifikation durch Einbezug des Hintergrunds vereinfachen lässt, so erscheint ein Einbezug des Hintergrunds sinnvoll.

Der Einbezug des Hintergrunds ist sinnvoll, wenn die zu klassifizierenden Handlungen meist in einer bestimmten Umgebung oder mit einer bestimmten Kleidung ausgeführt werden, beispielsweise bei der Unterscheidung von Sportarten.

Die Abstraktion auf Skelett-Daten und eine Subtraktion des Hintergrunds hilft insbesondere dann, wenn das trainierte Modell in einer hohen Anzahl von Szenen eingesetzt werden soll, für das Training aber nicht genügend Trainingsdaten für jede dieser Szenen zur Verfügung stehen.

In beiden Fällen wird neben den relativen Gelenk-Positionen auch die Position des Körpers in der Szene berücksichtigt. Ist dies nicht gewünscht, so kann das Bild auch auf den Bereich des Skeletts zugeschnitten werden.

Die Skelett-Daten müssen aber nicht auf ein Bild eingezeichnet werden. Es ist auch möglich, die jeweiligen Pixel-Koordinaten als Rohdaten zu interpretieren. Bei der Klassifikation könnten die Rohdaten in Form einer Zeitreihe interpretiert werden.

Werden wie in dem oben beschriebenen Fall extrahierte Merkmale wie Skelett-Daten vor der Interpretation durch das neuronale Netz mit dem RGB-Bild zusammengefügt, so spricht man von einer „early fusion“. Im Gegensatz hierzu spricht man bei einer Kombination der Ausgaben verschiedener neuronaler Netze von einer „late fusion“. Auch eine Kombination verschiedener Merkmale innerhalb des Netzes ist möglich („slow“ oder „middle fusion“), beispielsweise in dem die in einem Netz berechneten Gewichte auch in parallel genutzten Netzen verwendet werden. [4, S. 3; 66, S. 3]

3.4 Datensätze zur Handlungserkennung

Für das Training des neuronalen Netzes eines Machine Learning Algorithmus werden annotierte Beispieldaten benötigt. Im Fall einer Handlungserkennung in Videos sind das Beispielvideos, die verschiedene Handlungen zeigen.

Da diese Arbeit auf der Auswertung von Körperhaltungen basiert, werden neben den beschrifteten Videos mit Handlungen zudem auch beschriftete Skelett-Positionen für die Körperhaltung benötigt.

Ein bedeutendes Problem war in den letzten Jahren der Mangel an guten Datensätzen für die Erkennung und Klassifikation von Handlungen [116, S. 1]. Bei der Objekt-Erkennung haben vor allem generische Datensätze wie beispielsweise ImageNet [30] zu großen Fortschritten geführt [71].

In diesem Abschnitt sollen daher einige der bisher veröffentlichten Datensätze im Bereich der Video-Klassifikation und Handlungserkennung vorgestellt werden.

Grundsätzlich gilt, dass die Erkennungsrate mit der Anzahl der verwendeten geeigneten Trainingsdaten steigt [149, S. 2]. Es gibt daher verschiedene Ansätze, mit denen die Anzahl der Trainingsdaten erhöht werden kann:

- In einigen Forschungsprojekten werden mehrere ähnliche Datensätze miteinander kombiniert, z.B. UCF101 [122] und HMDB51 [72] („Multitask Learning“) [39, S. 2].
- Datensätze, die Handlungen aber keine Körperhaltungen enthalten, können mithilfe eines Algorithmus zur Erkennung von Körperhaltungen um diese Informationen ergänzt werden. In dieser Arbeit werden daher auch Datensätze ohne Körperhaltungen berücksichtigt.
- Für diese Arbeit wurden auch Datensätze zu Themen recherchiert, die zum Anwendungsfall passen, aber noch keine beschrifteten Handlungen enthalten. Die Handlungen müssen bei diesen noch manuell annotiert werden.
- Die Videos beziehungsweise die darin enthaltenden Skelette können unter bestimmten Umständen kopiert und an der vertikalen Achse gespiegelt, verschoben oder skaliert werden, um dem Algorithmus beizubringen, dass die jeweilige Handlungsklasse unabhängig von der Richtung, Größe oder Position der Skelette ist [149, S. 9].

Zur Recherche der Datensätze waren vor allem Übersichten (Surveys) wie [4, S. 4], [163] und [42, S. 11–15] hilfreich. In [163] werden beispielsweise die wichtigsten bis Ende 2015 veröffentlichten Datensätze vorgestellt, die neben RGB-Bildern auch Tiefen-Informationen enthalten.

Da für diese Arbeit Tiefen-Informationen nicht zwingend erforderlich sind, werden im Folgenden auch weitere Datensätze, die lediglich RGB-Bilder enthalten berücksichtigt. Zur weiteren Recherche waren auch [113, S. 226], [146, S. 3556], [117, S. 4], [163], [60, S. 3], [107, S. 2], [72, S. 2], und [153, S. 4] hilfreich. Datensätze aus der Zeit vor 2012 werden übersichtlich in [20] dargestellt und miteinander verglichen.

Im Folgenden wird eine Auswahl von geeigneten Datensätzen kategorisiert und anhand einiger Kennzahlen untereinander verglichen. In Abb. 3.7 werden beispielhafte Einzelbilder der Datensätze gezeigt, um die hohe Diversität an Handlungen und Methoden zur Aufnahme zu verdeutlichen. Während sich einige Datensätze [109; 49; 111] auf einfache Körperbewegungen aus statischer Perspektive beschränken, enthalten andere [48] komplexe Aktionen, wie das Auftragen eines Lippenstifts, aufgenommen aus verschiedenen Kamera-Perspektiven mit Schnitten und Kamerafahrten.

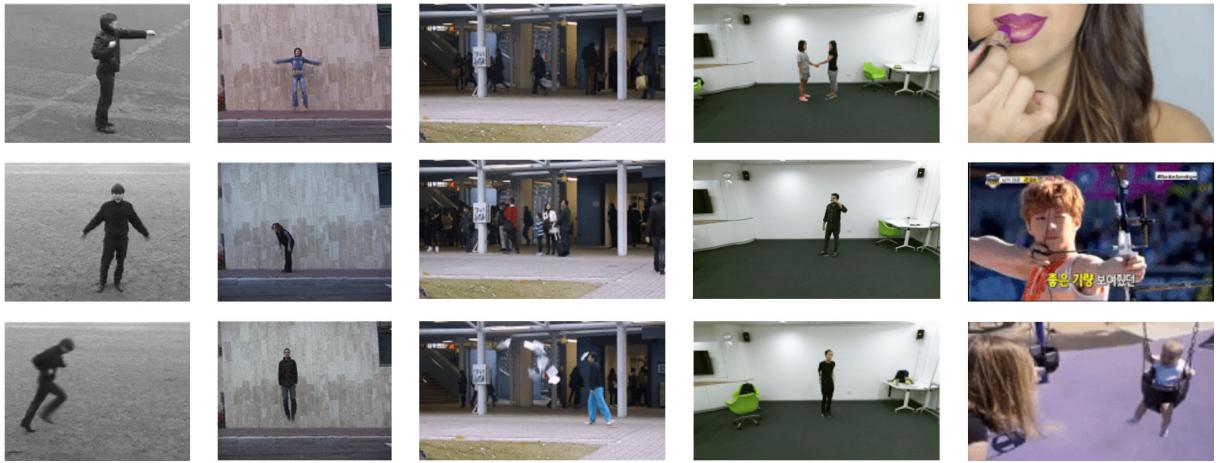


Abb. 3.7: Beispielhafte Einzelbilder aus den Datensätzen KTH [109], Weizmann [49], Avenue Anomaly [82], NTU RGB+D [111] und THUMOS [48] (von links nach rechts).

3.4.1 Unbeschriftete Videos

In diesem Abschnitt werden Datensätze gezeigt, die Videos zu je einem Thema enthalten, das gut zu dem Anwendungsfall der Video-Überwachung von öffentlichen Plätzen passt, ohne eine genauere Kategorisierung von Handlungen vorzunehmen.

Zu den in Tabelle 3.3 gezeigten Szenen gehören daher primär Video-Überwachungs-Aufnahmen, beispielsweise von zurückgelassenen Gepäckstücken, Treppenhäusern oder öffentlichen Plätzen.

Datensatz	Jahr	Thema und Umfang
PETS [135]	2006	Vergessene Gepäckstücke, 7 Szenarien mit je 4 Kameras
UCF Crowds [3]	2005	24 Videos von Menschenansammlungen
Anomalous Behaviour [160]	2010	8 Videos mit abschnittsweise ungewöhnlichem Verhalten
VIRAT [95]	2011	8 Stunden verschiedener Video-Überwachungs-Szenen
3DPeS [7]	2012	500 gewaltfreie Überwachungs-Videos mit Personen
Avenue Anomaly [82]	2013	37 Videos mit abschnittsw. ungewöhnlichem Verhalten
Violent Scenes [29]	2015	Kategorisierte Gewaltszenen aus 32 Spielfilmen
Surveillance Videos [131]	2016	Treppenhaus, 7*24 Stunden Videos aus einer Perspektive
Live Videos (LV) [74]	2017	ca. 4 Stunden Videos mit 34 ungewöhnlichen Ereignissen
Marktplatz Bredstedt [59]	2018	Livestream des Marktplatzes aus einer Perspektive
Marktplatz Einbeck [53]	2018	Livestream des Marktplatzes aus einer Perspektive
Marktplatz Biberach [124]	2018	Livestream des Marktplatzes aus einer Perspektive

Tabelle 3.3: Datensätze ohne Beschriftungen, zu dem Anwendungsfall nahen Themen.

Einige Datensätze wie 3DPeS [7], [131] und auch die Livestreams [59; 53; 124] eignen sich gut zur Definition eines neutralen Zustands. Andere Datensätze wie PETS [135], Anomalous Behaviour [160] und Violent Scenes [29] spezialisieren sich hingegen auf Szenen, die ein Gefahrenpotential bieten.

Für das Trainieren eines Machine Learning Klassifikators scheint daher eine Kombination mehrerer Datensätze sinnvoll.

3.4.2 Videos mit Handlungen

Für das Lernen eines Video-Klassifikators ist es erforderlich, dass jedes Video passend beschriftet ist. Hierfür wurden in den letzten Jahren zahlreiche Datensätze veröffentlicht (siehe Tabelle 3.4).

Frühe Datensätze, wie der 2004 veröffentlichte KTH-Datensatz [109], enthalten einfache Handlungen wie Springen, Winken oder Schlagen (Typ „Körper-Bewegungen“ in Tabelle 3.4).

Später wurden auch Videos, die komplexere Handlungen des Alltags darstellen, Videos die verschiedene Sport-Arten zeigen, Videos die verschiedene Handlungen beim Kochen und in der Küche zeigen und Videos, die verschiedene Interaktionen enthalten veröffentlicht.

Die in den Videos gezeigten Interaktionen können unterteilt werden in (1) Interaktionen von einer Person mit Objekten, (2) Interaktionen zwischen zwei Menschen und (3) Interaktionen in größeren Gruppen.

Datensätze, die aus Perspektive einer Überwachungskamera aufgenommen wurden, beispielsweise in einem Einkaufszentrum oder Marktplatz, wurden für diese Arbeit mit dem Typ „Öffentlicher Raum“ gekennzeichnet, da diese Perspektive für den einleitend beschriebenen Anwendungsfall von besonderer Relevanz ist.

Seit der Einführung von günstigen Stereo-Vision Kameras, wie beispielsweise der Microsoft Kinect-Kamera in 2009, gibt es auch Datensätze, die neben den RGB-Videos zusätzliche Tiefen-Informationen enthalten [163, S. 1]. Da in dieser Arbeit Handlungen auch ohne das Vorhanden-sein von Tiefen-Informationen erkannt werden sollen, sind diese Datensätze in Tabelle 3.4 nicht gesondert aufgeführt.

Im Folgenden werden einige Auffälligkeiten der Tabelle genauer beschrieben:

- Der AVA-Datensatz [153] enthält insgesamt 1.580.000 einzelne Handlungen, obwohl er mit 430 Videos deutlich weniger Videos als der Charades-Datensatz [117] mit 66.500 Handlungen und 9848 Videos enthält. Das liegt daran, dass die AVA-Videos im Durchschnitt länger sind und jeweils mehr Handlungen je Video enthalten.
- Einige Datensätze (zum Beispiel UCF Sports [106; 121] oder AVA [51]) enthalten neben der zeitlichen Einordnung der Handlungen auch einen räumlichen Rahmen, der die Position der Handlung im Bild beschreibt.
- Einige Datensätze (zum Beispiel Charades [117]) beschriften auch die in den Szenen gezeigte Objekte. Der Einbezug von Objekten kann je nach Anwendungsfall und Implementierung dazu führen, dass Handlungen besser unterschieden werden können. Ein Basketball ist beispielsweise vor allem in Szenen, in denen Basketball gespielt wird, vorhanden.
- Der Datensatz Youtube8M [1] wurde ursprünglich für eine thematische Klassifikation von Videos entwickelt, weshalb nur ein kleiner Teil der Klassen für eine Handlungserkennung relevant sind.
- Der Something-Something-Datensatz enthält für jedes Video eine oder mehrere ausformulierte Beschriftungen mit einem Prädikat und einem oder mehreren Objekten, z.B. „Holding <something> behind <something>“.

Für diese Arbeit eignen sich besonders die Datensätze der Typen Körper-Bewegungen, Öffentlicher Raum und Menschen-Interaktionen.

Datensatz	Jahr	Videos	Klassen	Typ	Kamerap.
KTH [109]	2004	600	6	Körper-Bewegungen	1 (stat.)
Weizmann [49]	2005	81	9	Körper-Bewegungen	1 (stat.)
CAVIAR [40]	2005	40	18	Öffentlicher Raum	1–2 (st.)
IXMAS [150]	2006	363	11	Körper-Bewegungen	5 (stat.)
ETISEO [92]	2007	85	40	Öffentlicher Raum	1–2 (st.)
Hollywood [73]	2008	2391	8	Alltag	1 (dyn.)
UCF Sports [106; 121]	2009	150	9	Sport	1 (dyn.)
UIUC1 [138]	2008	532	14	Körper-Bewegungen	1 (stat.)
Hollywood2 [85]	2009	1707	12	Alltag	1 (dyn.)
UCF11 YouTube [78]	2009	1100	11	Alltag	1 (dyn.)
Collective Activity [24]	2009	44*	5	Gruppen-Verhalten	1 (stat.)
Olympic Sports [93]	2010	800	16	Sport	1 (dyn.)
MSR Actions 3D [75]	2010	567	20	Körper-Bewegungen	1 (dyn.)
UCF50 [103]	2010	6618	50	Alltag	1 (dyn.)
UT-Interaction [108]	2010	120	6	Menschen-Interaktionen	1 (stat.)
TV Human Inter. [100]	2010	300	4	Menschen-Interaktionen	1 (dyn.)
CAD-60 [126]	2011	60	12	Alltag	1 (stat.)
HMDB51 [72]	2011	6849	51	Alltag	1 (dyn.)
CAD-120 [127]	2012	120	20	Alltag & Interaktionen	1 (stat.)
MSR Gesture 3D [148]	2012	336	12	Zeichensprache	1 (stat.)
MSR Daily Act. 3D [147]	2012	320	16	Alltag	1 (stat.)
UCF101 [122]	2012	13320	101	Alltag	1 (dyn.)
ASLAN [69]	2012	3697	432	Alltag	1 (dyn.)
BIT-Interaction [70]	2012	400	8	Menschen-Interaktionen	1 (stat.)
LIRIS [153]	2012	167*	10	Alltag & Interaktionen	1 (stat.)
MSR 3D Action Pairs [97]	2013	360	6	Objekt-Interaktionen	1 (stat.)
ActivityNet [17]	2015	2000	200	Alltag	1 (dyn.)
THUMOS [48]	2015	15T	101	Alltag & Sport	1 (dyn.)
DALY [152]	2016	510*	10	Alltag	1 (dyn.)
NTU RGB+D [111]	2016	56T	60	Alltag	1 (stat.)
Charades [117]	2016	9848*	157	Alltag	1 (dyn.)
UOW RGB-D comb. 3D [162]	2016	k. A.	94	Alltag	1 (dyn.)
AVA [51]	2017	430*	80	Alltag	1 (dyn.)
SLAC [164]	2017	520T*	1.7M	Alltag & Sport	1 (dyn.)
Youtube8M [1]	2018	6.1M*	3862	Alltag & Sport	1 (dyn.)
Moments in Time [89]	2018	1M	339	Alltag	1 (dyn.)
Something-Somet. V2 [141]	2018	220T*	318T	Objekt-Interaktionen	1 (dyn.)

Tabelle 3.4: Datensätze mit beschrifteten Handlungen. Aus mehreren Kamera-Perspektiven aufgenommene Handlungen wurden nicht zur Anzahl der Videos gezählt. Während sich bei statischen Aufnahmen (stat.) lediglich Objekte in der Szene bewegen, ändert sich bei dynamischen Aufnahmen (dyn.) auch die Kamera-Perspektive. Es findet immer genau eine Handlung je Video statt, außer bei den mit einem * markierten Datensätzen, die mehrere mit Zeitstempeln angegebene Handlungen in einem Video enthalten. Weitere Abkürzungen: keine Angabe (k. A.), für den jeweiligen Wert des Datensatzes konnte keine Information gefunden werden, T=Tausend, M=Millionen.

3.4.3 Videos mit Handlungen und Körperhaltungen

Es gibt auch Datensätze, die neben verschiedenen Handlungen auch die Skelett-Daten der gezeigten Personen enthalten. Bei einer Erkennung von Handlungen anhand von Körperhaltungen kann das Verwenden dieser Datensätze eine große Zeittersparnis bedeuten, da die Skelett-Daten der Videos nicht mehr selber berechnet werden müssen.

Einige der meistgenutzten Datensätze dieser Art werden in Tabelle 3.5 gezeigt.

Datensatz	Jahr	Videos	Klassen	Kameras	Typ	Zusätzliche Daten
CMU-MMAC [28]	2008	215	5	6 + 3D	Küche	Beschleunigung, Ton
TUM kitchen [132]	2009	20	10	4	Küche	-
HumanEva II [114]	2010	56	6	4 + 3D	Alltag	-
MP II Cooking [107]	2012	44*	65	1	Küche	-
UT-Kinect [154]	2012	200	10	1 + 3D	Körper	-
G3D [13; 12]	2012	210*	20	1 + 3D	Gaming	-
DHA [76]	2012		23	1 + 3D	Körper	-
2P Interaction [159]	2012	300	8	1 + 3D	Körper	-
Berkeley MHAD [94]	2013	660	11	4 + 3D	Alltag	Beschleunigung, Ton
J-HMDB [60]	2013	928	21	1	Alltag	-
G3Di [11; 12]	2014	90*	17	1 + 3D	Gaming	-
Human 3.6M [57; 19]	2014	165	15	4 + 3D	Alltag	-

Tabelle 3.5: Datensätze mit Videos und Keypoints. Aus mehreren Kamera-Perspektiven aufgenommenen Handlungen wurden nicht zur Anzahl der Videos gezählt. Der Zusatz *+ 3D* in der Spalte Kameras bedeutet, dass ein Motion-Capture- oder Stereo-Vision-System eingesetzt wurden und Tiefen-Informationen vorliegen. Mit einem * markierte Datensätze enthalten mehrere mit Zeitstempeln angegebene Handlungen in einem Video.

Datensätze, bei denen ein Motion Capture System zum Einsatz kam, eignen sich für ein Training eines rein auf Körperhaltungen basierenden Netzes besonders gut, da die Daten zum Lernen von 2D-Systemen beliebig gedreht werden können, wodurch eine Unabhängigkeit vom Kamera-Winkel entsteht.

Datensätze des Typs „Gaming“ enthalten Szenen, die in Computer-Spielen vorkommen könnten, wie beispielsweise Rennen, das Zielen mit einer Waffe, Tritte, Schläge oder auch Sportarten wie Golf [13].

3.4.4 Sonstige Datensätze

Neben den bisher gezeigten Datensätzen zur Handlungserkennung gibt es noch weitere Datensätze, die in einem Teil der Quellen referenziert, an dieser Stelle aber nicht näher beschrieben werden. Hierzu gehören Datensätze zur Gesten-Erkennung, die je nach Anwendungsfall und Implementierung zu einer Verbesserung der Handlungserkennung führen können. Da diese Arbeit mit Körperhaltungen arbeiten soll, und die Videos mit Gesten oft nur Ausschnitte des Körpers zeigen, eignen sich diese Videos weniger gut für die skelettbasierte Handlungs-Klassifikation.

3.4.5 Herausforderungen bei den Datensätzen

Bei der Auswahl sowie beim Erstellen von geeigneten Datensätzen wurden die folgenden Herausforderungen identifiziert:

Vorurteile Einige Datensätze repräsentieren nicht die Bedingungen und Häufigkeits-Verteilungen der echten Welt. Werden zum Erstellen von Datensätzen beispielsweise öffentliche Video-Archive wie YouTube oder Ausschnitte aus Spielfilmen und Serien verwendet, so sind tagtägliche, tendenziell langweilige Tätigkeiten oft unterrepräsentiert. [113, S. 225]

Je nach verwendetem Trainings-Algorithmus kann es sein, dass Handlungen mit wenigen Beispielvideos nicht oder verhältnismäßig weniger häufig in den Klassifikations-Ergebnissen erscheinen. Darüber hinaus erschwert diese ungleiche Verteilung von bereits verfügbaren Videos das Zusammenstellen neuer Datensätze, die bereits aufgenommene Videos beschriften.

Szenen-Vielfalt Handlungen werden in der echten Welt vor sehr vielen verschiedenen Hintergründen durchgeführt. Besonders in den früheren Datensätzen wurde oft nur eine kleine Anzahl von Hintergründen verwendet, da Videos selbst aufgezeichnet wurden, beispielsweise auf dem Gelände des jeweiligen Forschungs-Instituts oder in speziellen Studios. [113, S. 225]

Wenn ein Machine-Learning Klassifikator mit einseitigen Hintergründen trainiert, die möglicherweise von Klasse zu Klasse unterschiedlich sind, so ist es möglich, dass der Hintergrund als Merkmal für zu unterscheidende Handlungen zu stark gewertet wird.

Da der in dieser Arbeit untersuchte Ansatz auf der Auswertung von Körperhaltungen basiert, die vom Hintergrund getrennt werden können, spielt diese Herausforderung in diesem Fall keine so große Rolle.

Zeitliche und räumliche Begrenzung Bei der Klassifikation von Objekten kann zu jedem Zeitpunkt eindeutig bestimmt werden, ob ein Objekt im Bild ist, und wenn ja, in welchem Bereich des Bildes. Es ist rein physikalisch einfach, die Ränder des Objekts zu bestimmen. Handlungen haben im Gegensatz hierzu keine festen Grenzen und es gibt verschiedene Meinungen zur räumlichen und zeitlichen Eingrenzung von Handlungen. [116, S. 3–4]

Als Beispiel kann man sich die folgende Situation vorstellen: Eine Person hält ein Glas mit Wasser in der Hand und führt dieses zum Mund. Es ist nicht eindeutig, ob es sich hier bereits um die Handlung „trinken“ handelt, und es kann auch nicht klar definiert werden, wann diese beginnen würde. Verschiedene Personen würden den Start der Handlung beispielsweise erst beim Übergang des Wassers in den Mund sehen, oder gar erst beim Herunterschlucken des Wassers.

Auch räumlich lassen sich Handlungen oft nicht eindeutig lokalisieren. Beispielsweise ist für die Handlung „Tennis spielen“ nicht definiert, ob neben den spielenden Personen auch der Schläger, der Ball oder sogar das gesamte Spielfeld Teil der Handlung sind.

Sigurdsson, Russakovsky und Gupta [116] untersuchten 2017, ob die Auswertung von zeitlichen Grenzen einer Handlung zu besseren Ergebnissen bei der Klassifikation von Handlungen führt. Sie fassen zusammen, dass „zeitliche Grenzen von Handlungen zwar mehrdeutig, aber dennoch bedeutungstragend sind“ [116, S. 4].

Handlungs-Kategorisierung Auch das Benennen von Handlungen stellt beim Erstellen eines Datensatzes eine Herausforderung dar. Handlungen können sowohl sehr breit gefasst und allgemein gültig formuliert sein, beispielsweise „trinken“, es wäre aber auch möglich, diese sehr spezifisch zu definieren, beispielsweise „im Wohnzimmer aus einem Becher Wasser trinken“. [116, S. 3]

Für dieses Beispiel scheint eine Verallgemeinerung hilfreich, um für jede Handlungs-Klasse ausreichend Daten sammeln zu können [116, S. 5]. In anderen Fällen ist eine Verallgemeinerung aber nicht zielführend. Die Verben „nehmen“ oder „legen“ sind sehr mehrdeutig und ohne ein beschreibendes Nomen („Schuhe nehmen“) oder sogar zugehörige Präpositionen („Mantel ablegen“) für eine Unterscheidung von Handlungen nur wenig geeignet. [116, S. 3]

Darüber hinaus lässt sich beobachten, dass einige Handlungen andere Handlungen beinhalten oder voraussetzen, beispielsweise impliziert die Handlung „ein Auto fahren“ in den meisten Fällen die Handlung „sitzen“.

In den Experimenten von Sigurdsson, Russakovsky und Gupta [116] wurde außerdem der Frage nachgegangen, welche Kategorien sich für die Klassifikation von Handlungen eignen. Hierfür wurde eine Studie durchgeführt, in der die Probanden Videos mit einer vorgegebenen Menge von Handlungs-Kategorien beschriften sollten. Teilweise wurden auch entweder das die Handlung beschreibende Verb oder das an der Handlung beteiligte Objekt vorgegeben. Bei der Studie gab es weniger Verwechslungen bei Objekten mit vorgegebenen Verben als Verwechslungen bei Verben mit vorgegebenen Objekten. Demnach fällt es den Befragten leichter, Objekte statt Verben zu bestimmen. [116, S. 3].

Der Charades-Datensatz [117] besteht sowohl aus Handlungen mit Verben und Nomen als Beschriftung als auch solche, die nur Verben verwenden [117, S. 6–7]. Ein Test der oben genannten Autoren ergab, dass die Kombination von Nomen mit Verben zu unterscheidbareren Klassen führt [116, S. 3].

Eine weitere Herausforderung bei der Beschriftung von Handlungen ist die Berücksichtigung der Absichten hinter einer Handlung. Es ist in einem Video nicht unbedingt erkennbar, ob eine Person gerade rennt, weil diese gerne Sport treibt, weil sie verfolgt wird oder ob die Person jemanden verfolgt. [116] Ob eine Berücksichtigung der Absichten sinnvoll ist, wurde in keiner der Quellen untersucht.

Label Noise Beim Erstellen großer Datensätze (beispielsweise Youtube8M [1]) werden teils Methoden zur automatischen Generierung von Beschriftungen genutzt. Diese durchsuchen zum Finden von Videos, die eine bestimmte Handlung enthalten, Video-Plattformen, wie beispielsweise YouTube, anhand von Meta-Daten, die Nutzer zu ihren Videos angegeben haben.

Hierbei können Videos falsch zugeordnet werden, was zu einem „Label Noise“ genanntem Rauschen führt. Dieses Rauschen kann auch bei der manuellen Beschriftung von Videos passieren, wenn den beschriftenden Personen Fehler unterlaufen. Bei großen Datensätzen mit mehreren tausend Videos sind solche Fehler zu erwarten. [39, S. 2] Auch bei der Klassifikation von Videos, die mehr als eine Aktion zeigen oder lang andauernde Aktionen und Pausen enthalten, können diese Fehler entstehen [167, S. 2].

Die Größe des „Label Noise“ Rauschens antikorreliert mit der Genauigkeit des trainierten Modells [39, S. 2].

3.5 Anpassungen hinsichtlich des Anwendungsfalls

Einleitend wurde bereits ein Anwendungsfall für die Handlungserkennung beschrieben: „die anonymisierte Analyse von Handlungen an öffentlichen Orten, um ein sicheres Miteinander zu gewährleisten, indem bei gefährdenden Handlungen automatisch eine Hilfe gerufen wird“ (siehe Abschnitt 1.2).

Für eine solches System ist die Verarbeitung eines kontinuierlichen Videos in Echtzeit erforderlich, idealerweise mit einer geringen Verarbeitungsdauer, damit die Verzögerung bis zur Hilfe minimal ist.

In den in diesem Kapitel untersuchten Ansätzen wurden meist nur kurze, auf je eine Handlung zugeschnittene Videos klassifiziert. Die Unterteilung von Videos in für die Klassifikation relevante zeitliche und räumliche Abschnitte ist ein Forschungsgebiet, das unter anderen in [165; 161; 44; 49] untersucht wurde. In [165] wurde ein Framework vorgestellt, dass Videos in „Schlüsselvolumen“ unterteilt, also die zeitlichen und räumlichen Grenzen von Handlungen erkennt, ohne bereits eine Klassifikation vorzunehmen. So sollen irrelevante Bereiche nicht klassifiziert und Rechenzeit gespart werden.

Bei der Auswertung von Echtzeit-Videos muss für die Klassifikation eine passende Intervall-Länge gewählt werden. Diese sollte lang genug sein, um eine ganze Handlung zu enthalten, aber nicht zu lang, da sonst eine hohe zeitliche Verzögerung zwischen der Handlung und der Klassifikation entsteht.

Darüber hinaus muss darauf geachtet werden, dass bei zahlreichen Handlungen (Interaktionen) mehr als eine Person beteiligt ist und daher mehrere Skelette in den Videos erkannt und daraufhin klassifiziert werden müssen. Bei der Definition der unterscheidbaren Handlungen macht unter Umständen eine Trennung der Klassen nach den daran beteiligten Personen Sinn.

Eine weitere Schwierigkeit, die in den bisherigen Ansätzen noch nicht untersucht wurde, ist die Detektion von Personen im Hintergrund, die nicht an einer Handlung beteiligt, aber ebenfalls im Bild sichtbar sind. Damit der Klassifikator diese zuverlässig als irrelevant betrachtet, ist ein großer Datensatz notwendig.

Noch eine Stufe schwieriger wird es, wenn mehrere Personen im Bild mehrere Handlungen parallel ausführen. In diesem Fall lässt sich nicht mehr eine für das ganze Video zu dieser Zeit gültige Handlung bestimmen, sondern es gäbe für verschiedene Bereiche des Bilds verschiedene Handlungen.

Die für den Anwendungsfall eingesetzte Kamera wird vermutlich eine Totale Einstellung und keine Nahaufnahme als Bildausschnitt zeigen. Für die Detektion der, in Pixeln gemessen, kleineren Personen muss ein geeigneter Pose Estimator zur Erkennung von Haltungen eingesetzt werden. Auch der zum Training verwendete Datensatz sollte zur Kameraperspektive passen oder groß genug sein, um auch einige Aufnahmen aus dieser Perspektive zu enthalten.

In dieser Arbeit können aufgrund der hohen Komplexität nicht alle Anforderungen des Anwendungsfalls berücksichtigt werden. Es wird aber versucht, so weit im Rahmen dieser Arbeit sinnvoll, Entscheidungen zugunsten des Anwendungsfalls zu treffen.

3.6 Herausforderungen der Handlungserkennung

Bei der Untersuchung der verschiedenen bisherigen Ansätze konnten mehrere Herausforderungen identifiziert werden. In diesem Abschnitt werden die Herausforderungen aufgeführt, die bei der Entwicklung eines Handlungserkenners auftreten können. Weitere Herausforderungen, die bei Auswahl und Erstellung von Datensätzen zur Handlungserkennung auftreten, wurden bereits in Abschnitt 3.4.5 genannt.

Rechenaufwand Eine der größten Herausforderungen, die die Entwicklung und den Test von Algorithmen zur Handlungserkennung erschwert, ist die hohe Rechenkapazität, die zum Trainieren der neuronalen Netze erforderlich ist. Auch wenn die Anwendung des Klassifikators später in Echtzeit erfolgen kann, nimmt das maschinelle Lernen viel Zeit in Anspruch.

Ein einfaches 2D-CNN benötigt zur Klassifikation von 101 Klassen rund 5 Millionen Parameter. Ein 3D-CNN benötigt für dieselbe Aufgabe rund 33 Millionen Parameter, weshalb mehr als sechsmal so viel Zeit eingeplant werden sollte. [43] Das Training eines 3D CNN mit den über 13.000 Videos des UCF101 Datensatzes [122] benötigt rund drei bis vier Tage. Für den, mit einer Million Videos, deutlich größeren Sports-1M Datensatz [66] werden ungefähr zwei Monate benötigt. [140, S. 1]

Der hohe Rechenaufwand erschwert das Suchen einer geeigneten Architektur [140, S. 1] und bietet eine Grundlage für Overfitting, da häufiger kleine Datensätze eingesetzt werden [43].

Einbezug der zeitlichen Zusammenhänge In den verschiedenen Ansätzen wurden verschiedene Methoden genutzt, um zeitliche Informationen, über mehrere Einzelbilder hinweg, in die Klassifikation einzubeziehen. Im zeitlichen Verlauf verändern sich aber oft nicht nur die, die Handlung ausführenden, Personen im Bild, sondern je nach Datensatz auch die Kamera-Perspektive [43].

Bei der Klassifikation von Handlungen in Spielfilmen gibt es beispielsweise zahlreiche Kamera-Bewegungen und Schnitte, die bei der Klassifikation bewusst ignoriert werden sollten. Es gibt verschiedene Ansätze, um Schnitte in Filmen zu erkennen [144] und verschiedene Objekterkennungs- und Tracking-Algorithmen, die fokussierte Objekte über mehrere Einzelbilder hinweg verfolgen können. [43]

Ein einfaches Aufreihen der Techniken, die zur Klassifikation von Einzelbildern genutzt werden, ist nicht immer optimal [140, S. 1].

Vielseitige Netzarchitekturen Wie bereits aus Tabelle 3.1 ersichtlich, basieren Ansätze zur Handlungserkennung auf verschiedenen neuronalen Netzen und unterschiedlichen Architekturen. Durch diese Vielfalt ist die Auswahl einer für den eigenen Anwendungsfall geeigneten Netzarchitektur nicht trivial. Die Netzarchitekturen unterscheiden sich hinsichtlich (1) des genutzten Netzes, (2) ihrer Vorverarbeitung der Eingabedaten, (3) der Anzahl der genutzten Schichten und (4) der Fusions-Strategie, mit der die zeitlichen Informationen miteinbezogen werden [140, S. 1].

Auch die Merkmalsextraktion kann entweder Teil eines „Ende-zu-Ende“-Ansatzes sein oder separat stattfinden [43].

Zahlreiche Datensätze erschweren Vergleichbarkeit Für die Handlungserkennung existieren zahlreiche Datensätze (siehe Abschnitt 3.4). Unpraktischerweise ist keiner der Datensätze ideal, da es auch in diesem Bereich zahlreiche Herausforderungen gibt (siehe Abschnitt 3.4.5). Bei der Entwicklung der verschiedenen Ansätze wurden daher jeweils verschiedene Datensätze zum Test verwendet, was die Vergleichbarkeit der Ansätze erschwert [43].

Es hat sich noch kein Standard-Datensatz für Benchmarks durchgesetzt. Die kleineren Datensätze enthalten nicht genügend Videos, und große Datensätze sind aufgrund des Rechenaufwands zur schnellen Evaluation von Ansätzen ungeeignet. Sports-1M [66] ist für das Lernen eines grundlegenden Verständnisses von Handlungen hilfreich, aber auf Sportarten beschränkt. In Experimenten wurde bereits gezeigt, dass ein mehrstufiges Training mit verschiedenen Datensätzen (*finetuning*) die Genauigkeit bei der Klassifikation erhöhen kann, weshalb ein Festlegen auf einen einzigen Datensatz bisher noch nicht sinnvoll ist. [140, S. 1]

Geringe Größe der Datensätze Eine große Herausforderung bei der Entwicklung von Handlungserkennern war bis 2017, dass die verfügbaren Datensätze zu klein und nicht divers genug waren [149, S. 2]. Tiefe CNN sind daher von Overfitting bedroht, es konnte also eine Überanpassung an für die Klassifikation nicht geeignete Merkmale stattfinden [149, S. 2]. Verglichen mit den heute verfügbaren Datensätzen für Bildklassifizierung, wie beispielsweise ImageNet [30], sind die verfügbaren Video-Datensätze immer noch klein.

Darüber hinaus erschweren auch die in Abschnitt 3.4.5 genannten Herausforderungen hinsichtlich der Kategorisierung von Handlungen das Erstellen größerer Datensätze.

4 Experimente

Die in dieser Arbeit beschriebenen theoretischen Ansätze können auf ihre praktische Umsetzbarkeit untersucht werden, indem diese implementiert, getestet und miteinander verglichen werden.

Leider wurde der für die Ansätze verwendete Quelltext nur in wenigen Fällen veröffentlicht. Da eine Implementierung der komplexen Ansätze den Rahmen dieser Arbeit übersteigen würden, sollen in den Experimenten Teile der bisherigen Ansätze implementiert und mit neuen, selbst entwickelten Ansätzen, kombiniert werden.

4.1 Projektplanung

4.1.1 Anforderungsdefinition

Im Rahmen dieser Arbeit soll ein Prototyp entwickelt werden, der Handlungen in Videos durch das Auswerten von Körperhaltungen klassifiziert.

Zur Haltungserkennung und Klassifikation von Handlungen sollen neuronale Netze eingesetzt werden. Während für die Haltungserkennung auf bestehende Bibliotheken zurückgegriffen werden soll, sollen bei der Entwicklung der Handlungserkennung neue Ansätze verfolgt werden.

Ein wichtiges Kriterium zur Messung des Erfolgs ist die Klassifikations-Genauigkeit. Zur Messung dieser wird der Trainings-Datensatz in drei Teile aufgeteilt, die für Training, Validierung und Test des neuronalen Netzes verwendet werden. Anschließend werden die Test-Daten klassifiziert und die vorhergesagten Klassen mit den tatsächlichen Klassen verglichen, um die Genauigkeit des Klassifikators zu berechnen.

Da diese Genauigkeit stark vom verwendeten Datensatz abhängt, sich bisher aber (1) noch kein Datensatz als fester Standard etabliert hat (siehe Abschnitt 3.6), (2) das Training eines modernen Datensatzes mehrere Tage in Anspruch nehmen kann und (3) die mit den Testdaten gemessene Genauigkeit nicht mit der Genauigkeit im produktiven Betrieb übereinstimmt, werden für diese Arbeit keine genauen Anforderungen hinsichtlich der Klassifikations-Genauigkeit definiert.

Ein Erfolg ist gegeben, wenn für einen beliebigen Datensatz über die Hälfte der Testdaten korrekt klassifiziert werden. Die entwickelten Prototypen sollen als Proof of Concept für eine Handlungserkennung eingesetzt werden.

4.1.2 Lösungsansatz

Zur Entwicklung der praktischen Implementierung wurde ein Notebook mit der folgenden Hardware-Ausstattung verwendet:

- Prozessor: Intel Core i7-6820HK @ 4 GHz
- Grafikkarte: NVIDIA GeForce GTX 1070 mit 8 GB GDDR5 RAM
- Arbeitsspeicher: 16 GB DDR4
- Massenspeicher: 512 GB NVMe-SSD

Softwareseitig wurde das Betriebssystem Windows 10 Home eingesetzt. Um eine bessere Grafikkarten-Unterstützung zu erhalten, wurde das Framework NVIDIA CUDA in Version 9.0 sowie die Bibliothek NVIDIA CuDNN in Version 7 verwendet.

Um schrittweise auf die komplexeren Ansätze hinzuarbeiten werden mehrere Prototypen mit steigender Komplexität erstellt.

Der erste Prototyp soll überprüfen, ob ein auf Körperhaltungen trainierter Bildklassifikator zur Erkennung von Handlungen in den Einzelbildern eines Videos genutzt werden kann.

Anschließend soll ein zweiter Prototyp diesen Ansatz erweitern und neben der räumlichen Dimension auch zeitliche Zusammenhänge berücksichtigen.

Da der Zeitraum dieser Arbeit beschränkt ist, wurden die Experimente parallel zum Verfassen der Texte durchgeführt. Falls während der Entwicklung weitere, vielversprechende, Ansätze gefunden werden, so können auch diese in Form von Prototypen implementiert werden.

4.1.3 Risikoanalyse

Um ein Scheitern des Projektes zu verhindern, ist es sinnvoll, die Projekt-Risiken im Vorfeld zu erarbeiten. Eine Definition von Gegenmaßnahmen ist nur dann möglich, wenn alle Probleme und deren geschätzte Auswirkungen bekannt sind. [87, S. 29]

Die Herausforderungen und Probleme bisheriger Ansätze wurden bereits in Abschnitt 3.6 erarbeitet. In dieser Risikoanalyse werden die Verursacher der damaligen Probleme mit dem Lösungsansatz dieser Arbeit verglichen.

Klassische Risiken, die bei jedem Projekt auftreten können, wie beispielsweise Krankheit oder Naturkatastrophen, werden an dieser Stelle bewusst nicht genannt.

Risiken bei der Erkennung von Körperhaltungen

Zu kleinen Personen Ein digitales Bild besteht, wie einleitend bereits erwähnt, aus mehreren Pixeln. Wenn das Bild eine geringe Auflösung hat oder Personen im Bildausschnitt klein sind, so enthält deren digitale Repräsentation nur wenige Pixel. Das kann dazu führen, dass die Personen nur schwer erkannt werden können und Skelette aufgrund der ungenauen Bild-Informationen über die Person nicht berechnet werden können.

Wenn geeignete Datensätze gewählt werden, die in Pixeln gemessen große Personen enthalten, und auch bei Anwendung mit eigenen Videos eine Perspektive gewählt wird, die Menschen vollständig, aber nicht zu klein darstellt und eine Kamera mit ausreichender Auflösung eingesetzt wird, sollte dieses Risiko nicht auftreten.

Fehlende Körperteile, partielle oder vollständige Überdeckung Sowohl in den Datensätzen, als auch in selbst aufgenommenen Videos, kann es passieren, dass Teile der Personen, deren Körperhaltungen erkannt werden sollen, von anderen Objekten oder Personen partiell oder vollständig verdeckt werden. Der verwendete Algorithmus zur Erkennung von Körperhaltungen sollte in der Lage sein, sichtbare Gelenke auch in teilweise verdeckten Körpern zu erkennen. Darüber hinaus könnte der Algorithmus versuchen, die Position von nicht sichtbaren Gelenken anhand der menschlichen Anatomie vorherzusagen.

Fehlerkennungen im Bildhintergrund Im Hintergrund der zu analysierenden Videos können Objekte vorhanden sein, die menschlichen Körpern ähneln. Wenn für diese Objekte Skelette in das Bild eingezeichnet werden, können diese sich mit dem Original-Bild überlagern oder es wirkt so, als wären mehrere Personen an der aufgenommenen Handlung beteiligt, wodurch die Genauigkeit des Klassifikators sinkt.

Es gibt zwei Ansätze, um dieses Problem zu umgehen. Wenn es möglich ist, die Hintergründe der Trainings- und Anwendungs-Videos neutral zu gestalten, so scheint das eine schnell umsetzbare Lösung. Die Alternative hierzu ist, falls die Bewegungen und Objekte im Hintergrund des Anwendungs-Videos nicht entfernt werden können, bereits während des Trainings bewusst falsch erkannte Skelette im Hintergrund zu platzieren, damit der Machine Learning Klassifikator die Möglichkeit hat, das Rauschen zu erkennen und dieses von den eigentlichen Skeletten zu trennen. Hierfür sind deutlich mehr Trainings-Videos erforderlich.

Risiken bei der Klassifikation von Handlungen

Auswahl ungeeigneter Trainingsdaten Bei der Untersuchung der Datensätze hat sich gezeigt, dass diese zahlreiche verschiedene Handlungen, Szenen und Perspektiven enthalten. Wenn in dem für das Training verwendeten Datensatz keine Körperhaltungen erkannt werden können oder die Perspektive grundlegend anders ist, als im Anwendungsfall, so werden bei der Klassifikation vermutlich weniger gute Genauigkeiten erreicht.

Überanpassung Wenn für das Training einer Klasse viele ähnliche Videos klassifiziert werden, so kann es dazu kommen, dass der Algorithmus Gemeinsamkeiten dieser Videos erlernt, die für eine Unterscheidung von Handlungen zwar innerhalb des Trainingsdatensatzes Sinn machen, aber nicht in der Anwendungsphase. Dieses Verhalten wird Überanpassung (*overfitting*) genannt.

In einem unsorgfältig zusammengestellten Datensatz könnten beispielsweise alle Handlungen des Typs „springen“ von einer Person mit einem roten T-Shirt ausgeführt werden und alle Handlungen des Typs „gehen“ von einer Person mit einem blauen T-Shirt. Bei der Suche nach Merkmalen zur Unterscheidung dieser beiden Klassen wird der Algorithmus vermutlich auch die T-Shirt Farbe berücksichtigen, da diese zu einer eindeutigen Unterscheidung führen kann. In der Anwendungsphase, wenn die T-Shirt Farbe nicht mehr mit der Handlung korreliert, werden zahlreiche Fehlklassifikationen erfolgen.

Eine Methode zur Vermeidung von Überanpassung ist das Verwenden einer rein auf Skeletten basierten Handlungserkennung. Da verschiedene Personen Handlungen unterschiedlich ausführen, sollte ein großer Datensatz verwendet werden, der mehrere Personen aus verschiedenen Perspektiven und Entfernung zeigt, damit Handlungen nicht anhand von Eigenschaften wie der Körpergröße oder der Position im Bild unterschieden werden.

Nicht ausreichende Rechenkapazität Das Training eines neuronalen Netzes kann bei großen Datensätzen mehrere Monate in Anspruch nehmen (siehe Abschnitt 3.6). Im Rahmen dieser Arbeit können daher nur kleinere Datensätze oder eine Kombination mehrerer kleinerer Datensätze eingesetzt werden. Grundsätzlich sinkt die für das Training benötigte Zeit mit der Rechenleistung des eingesetzten Computers. Da Computer mit einer hohen Rechenkapazität aber teurer und, beispielsweise beim Mieten eines Hochleistungs-Servers, aufwendiger zu bedienen sind, wurde in dieser Arbeit ein hochwertiges Gerät für Privatkunden verwendet (siehe Abschnitt 4.1.2).

4.1.4 Vorangehender Test

Um einen ersten praktischen Eindruck von den bisher entwickelten Implementierungen im Bereich der Handlungserkennung zu bekommen, wurde der Quelltext von Sigurdsson [115] ausgeführt. Sigurdsson ist Teil der Gruppe, die den Charades Datensatzes [117] zusammengestellt hat.

Dieser Python-Code analysiert mithilfe der OpenCV-Bibliothek das Bild einer an den Computer angeschlossenen Webcam. Mithilfe von TensorFlow werden die Handlungen des Charades Datensatzes [117] verglichen. Die Klassifikation der Handlungen erfolgt in Echtzeit: für eine Klassifikation werden circa 0,04 Sekunden benötigt (25 Einzelbilder pro Sekunde).

Die Handlungs-Klassen des Charades Datensatzes sind aus Verben und Objekten zusammengesetzt, beispielsweise „Holding a cup/glass/bottle of something“ [117, S. 7].

In Abb. 4.1 werden zwei Screenshots des Klassifikators gezeigt. Aus diesen ist erkennbar, dass für jedes untersuchte Bild die drei wahrscheinlichsten Klassifikationen angezeigt werden.

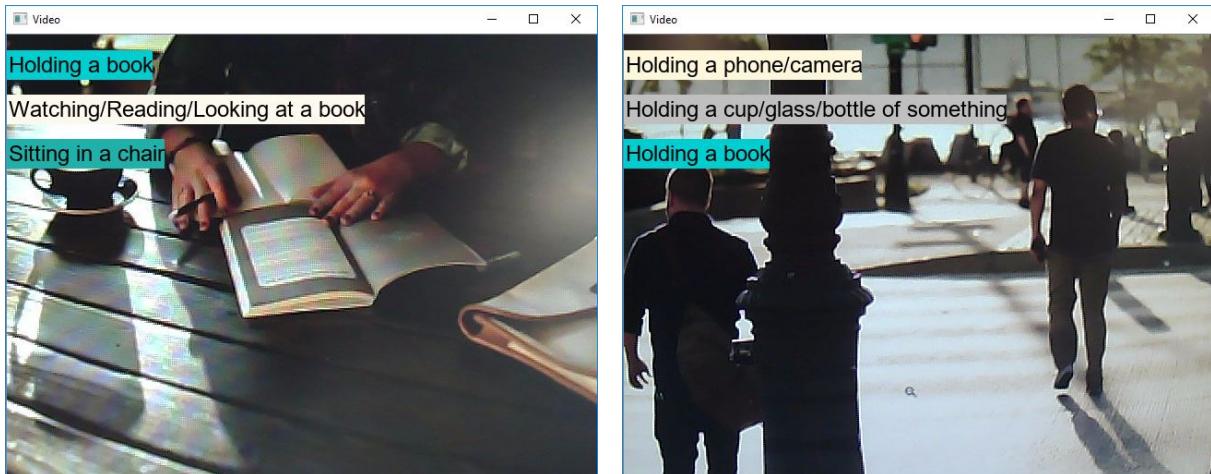


Abb. 4.1: Screenshots des getesteten Handlungserkenners. Es wurden Ausschnitte der beiden Bilder [104] und [58] klassifiziert.

Während in Abb. 4.1 links alle vorhergesagten Klassen korrekt sind, stimmt im rechten Bild keine einzige der Klassen, obwohl im Datensatz auch die Klassen „Someone is running somewhere“ und „Walking through a doorway“ vorhanden sind [117, S. 11].

In dieser Arbeit wird keine Weiterentwicklung dieser Implementierung angestrebt, da diese nicht auf Körperhaltungen basiert und ein starker Fokus auf die Erkennung von Objekten gelegt wurde, die bei der Erkennung von Körperhaltungen nicht sichtbar sind.

Weitere Ansätze wurden nicht getestet, da oft kein Quelltext vorhanden war oder die Implementierungen umfangreiche Installationen erfordern würden, die im Rahmen dieser Arbeit zu viel Zeit in Anspruch nehmen würden.

4.2 Prototyp 1: Auswertung der räumlichen Dimension

Als erste praktische Implementierung wurde ein Prototyp entwickelt, der die grundlegende Funktionalität einer skelettbasierten Handlungserkennung bietet.

Die Grundidee des Prototypen ist es, bestehende Datensätze mit RGB-Videos von einfachen Handlungen in einzelne Bilder aufzuteilen und in diesen Skelette zu erkennen (siehe Abb. 4.2). Die auf einen schwarzen Hintergrund gezeichneten Skelette werden anschließend zum Trainieren eines Machine Learning Klassifikators genutzt.

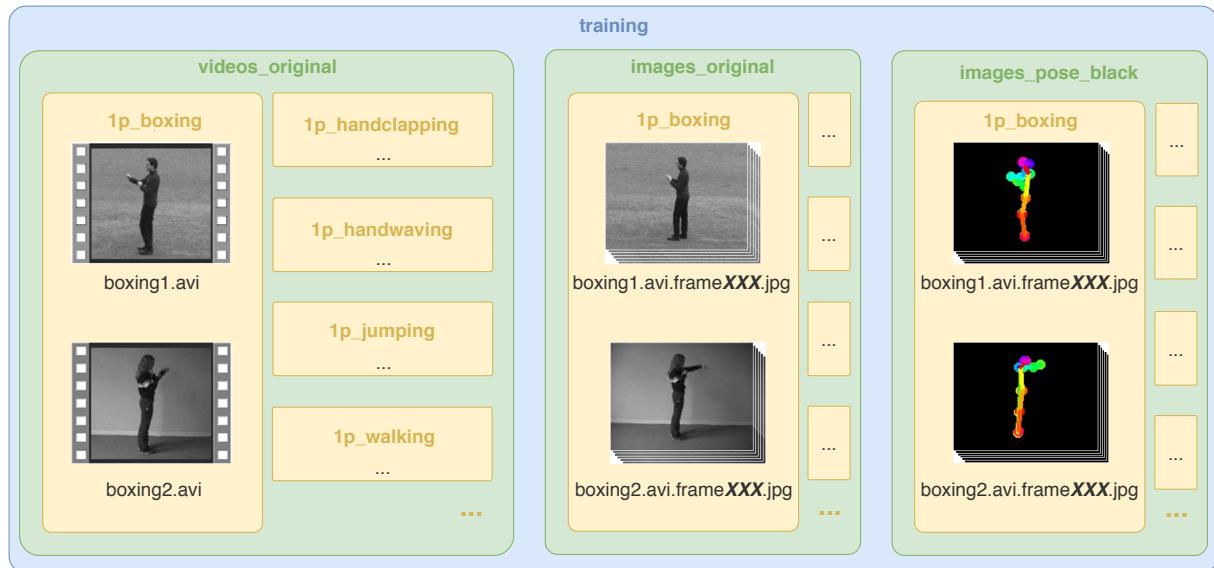


Abb. 4.2: Die für das Training genutzte Ordnerstruktur des ersten Prototyps.

In der auf das Training folgenden Anwendungsphase werden Webcam-Bilder oder Bilder aus einem Ordner eingelesen und ebenfalls Skelette in diesen erkannt. Ein klassischer Bild-Klassifikator gibt anschließend zurück, zu welcher Handlung das eingelesene Bild am wahrscheinlichsten passt.

Das Programm besteht aus mehreren Komponenten, die je nach Bedarf und Anwendungsfall entweder der Reihe nach oder einzeln gestartet werden können (siehe Abb. 4.3).

Im Folgenden finden sich weitere Details zur Umsetzung des Prototypen, eine Installations- und Bedienungs-Anleitung sowie erste Ergebnisse von Performance-Tests.

4.2.1 Vorbereiten der Trainingsdaten

Bei der Analyse bisheriger Datensätze (siehe Kapitel 3.4) wurde gezeigt, dass der Großteil der verfügbaren, mit Handlungen beschrifteten Videos keine bereits berechneten Körperhaltungen enthalten, und die Datensätze, die Skelett-Daten enthalten, diese in verschiedenen Formaten und Detailstufen angeben.

Um beliebige mit Handlungen beschriftete Videos für das posenbasierte Training nutzen zu können, wurde ein Programm zur einheitlichen Erkennung von Körperhaltungen in den Prototypen integriert.

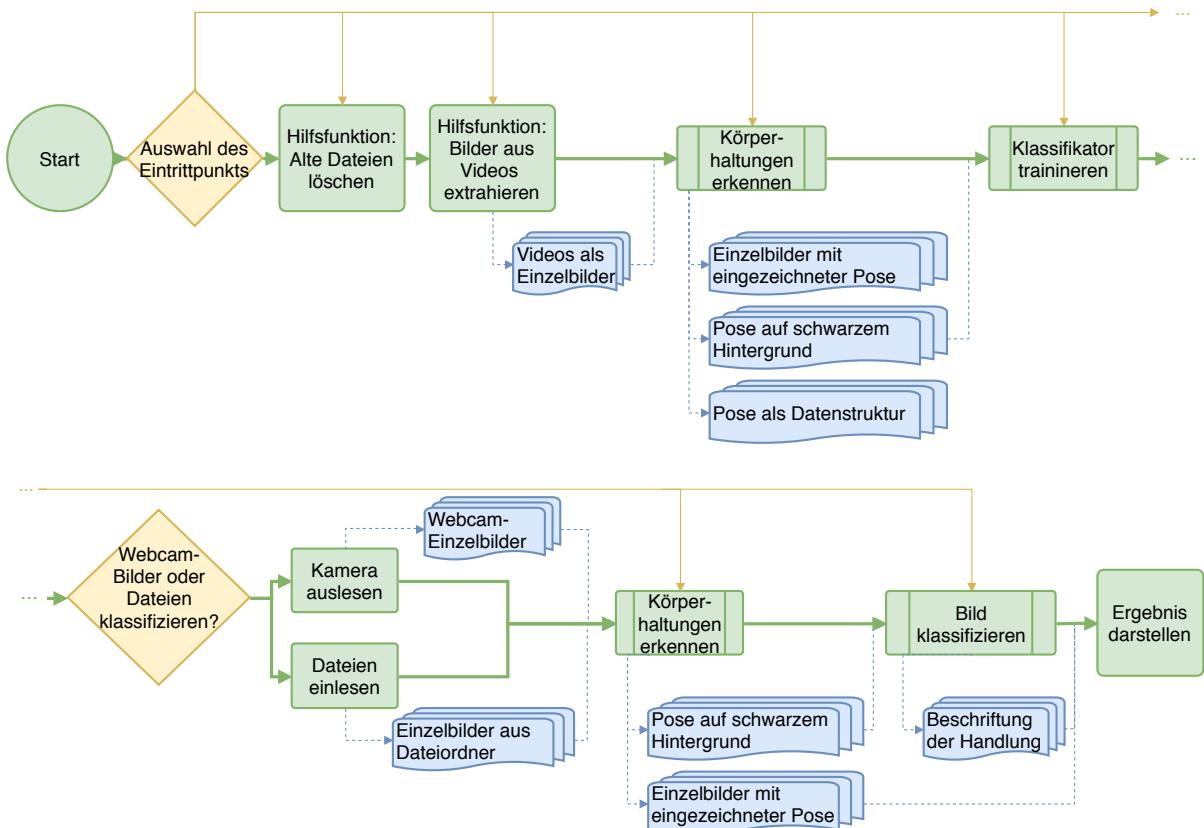


Abb. 4.3: Architektur des ersten Prototyps.

Da die Körperhaltungs-Erkennung im Rahmen dieser Arbeit mit weiteren Algorithmen kombiniert werden soll, wurde für diesen Prototypen die schnellere (siehe Abschnitt 2.1.5) Bibliothek OpenPose [18] ausgewählt. Der Original-Quelltext von OpenPose ist für die Programmiersprache C++ geschrieben. Da sonstiger Quelltext in dieser Arbeit auf Python und TensorFlow basiert, wurde als Grundlage für die Erkennung von Körperhaltungen die ebenfalls auf Python und Tensorflow basierende OpenPose-Modifikation von Ildoo Kim [67] verwendet.

Als Trainingsdaten wurde eine Kombination mehrerer Datensätze verwendet, so dass 17 verschiedene Klassen von Handlungen, an denen eine Person (1p) oder zwei Personen (2p) beteiligt sind, unterschieden werden können: *1p_boxing*, *1p_handclapping*, *1p_handwaving*, *1p_jogging*, *1p_jumping*, *1p_pointing*, *1p_running*, *1p_walking*, *2p_approaching*, *2p_departing*, *2p_exchanging*, *2p_highfive*, *2p_hugging*, *2p_kicking*, *2p_punching*, *2p_pushing*, *2p_shakinghands*.

Es wurden 1.197 Videos aus den Datensätzen KTH [109], Weizmann [49], BIT-Interaction [70], 2P Interaction [159], TV Human Interaction [100] und UT-Interaction [108] in insgesamt 387.159 Einzelbilder aufgeteilt. In 242.606 dieser Bilder wurden Körperhaltungen erkannt, die zum Trainieren des Klassifikators genutzt werden können.

4.2.2 Training des Bildklassifikators

Zur Unterscheidung von Handlungen werden Skelett-Bilder an einen Bildklassifikator übergeben, der überprüft, zu welcher Handlungs-Klasse das jeweilige Skelett-Bild am wahrscheinlichsten passt.

Für den Bildklassifikator wurde das Inception-v3 Netz [130] um eine zusätzliche selbsttrainierte Schicht erweitert, die auf die Unterscheidung der Skelett-Bilder spezialisiert ist („Transfer Learning“) [33]. In [149, S. 6] wurde ebenfalls eine Modifikation des Inception Netzes gewählt. Die Architektur des neuronalen Netzes dieses Prototypen ist in Abb. 4.4 dargestellt.

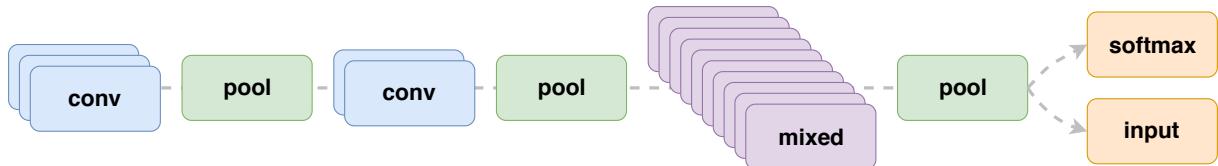


Abb. 4.4: Netzarchitektur des Bildklassifikators. Bei den mit *conv* beschrifteten Schichten handelt es sich um Convolutional Layer, bei den *pool*-Schichten um diese zusammenfassende Pooling-Layer. Die *mixed*-Schichten bestehen aus mehreren, in besonderen Strukturen (siehe [130]) angeordneten, Convolutional Layers. Als Aktivierungsfunktion wird eine *softmax*-Funktion verwendet (siehe Abschnitt 2.2.4). Die *input*-Schicht fasst die Eingabedaten des Trainings zusammen. Bei Tensorflow gehören hierzu die sogenannten „Bottleneck“-Dateien, die Merkmalsvektoren der Bilder enthalten [133], sowie die Beschriftungen der Trainingsdaten, die auch „Ground Truth“ genannt werden.

Dieser Ansatz ermöglicht Klassifikationen mit weniger Trainingsdaten bei kürzeren Trainingszeiten. [33, S. 8]

Zur Definition des neuronalen Netzes und zum Training wurde das Skript `retrain.py` [134] der TensorFlow-Autoren verwendet und leicht angepasst.

Das Skript erwartet, dass die Bilder nach Handlung unterschieden in einzelnen Ordnern liegen. Der in Abb. 4.2 rechts gezeigte Ordner `images_pose_black` enthält für jede Handlung einen nach der Handlung benannten Unterordner (gelbe Bereiche in Abb. 4.2), der nur Skelett-Bilder eben dieser Handlung enthält.

Der Dateiname der Skelett-Bilder spielt bei der Klassifikation keine Rolle. In diesem Prototypen wird der Dateiname automatisch generiert, wenn die Videos in Einzelbilder aufgeteilt werden und in diesen Einzelbildern Körperhaltungen erkannt werden.

Der Prototyp bietet neben dem Speichern der Skelette auf schwarzem Hintergrund auch die Möglichkeit, Skelette in das Originalbild einzuleichen und die Skelett-Daten als Datenstruktur in Textdateien abzuspeichern.

Der trainierte Klassifikator kann zur Handlungserkennung in beliebigen Videos und Einzelbildern genutzt werden. Hierfür wurde ein Skript geschrieben, das mit Hilfe von [67] Körperhaltungen in Einzelbildern erkennt und die dabei entstehenden Skelette an den Klassifikator weitergibt.

Der vollständige kommentierte Quelltext kann wie im folgendem Abschnitt 4.2.4 beschrieben angesesehen werden.

4.2.3 Test und Auswertung

Bei der Klassifikation des vollständigen, in Abschnitt 4.2.1 beschriebenen, Datensatzes wurde nach 5.000 Schritten (Epochen) eine Genauigkeit von 63,9% erreicht. Während der Anwendungsphase erzielt der Algorithmus beim Klassifizieren von Bildern einer lokalen Webcam eine Geschwindigkeit von 0,8 Einzelbildern pro Sekunde. Für ein Einzelbild werden also 1,25 Sekunden benötigt.

Im Bereich der Wahrscheinlichkeitstheorie hat sich eine Methode zur Darstellung der Qualität von Klassifikatoren und Trainingsdaten etabliert. Aus einer sogenannten Wahrheits- oder Konfusionsmatrix (*confusion matrix*) lässt sich ablesen, wie viele Elemente einer bestimmten Klasse beim Test des Klassifikators korrekt klassifiziert wurden und wie viele fälschlicherweise in eine bestimmte andere Klasse eingeordnet wurden (siehe Abb. 4.5).

Eine ideale Wahrheitsmatrix besteht außerhalb der Hauptdiagonale ausschließlich aus dem Wert 0 (es gibt also keine falschen Klassifikationen) und die Werte der Hauptdiagonale sollten ungefähr gleich groß sein (es gab also einen balancierten Testdatensatz).

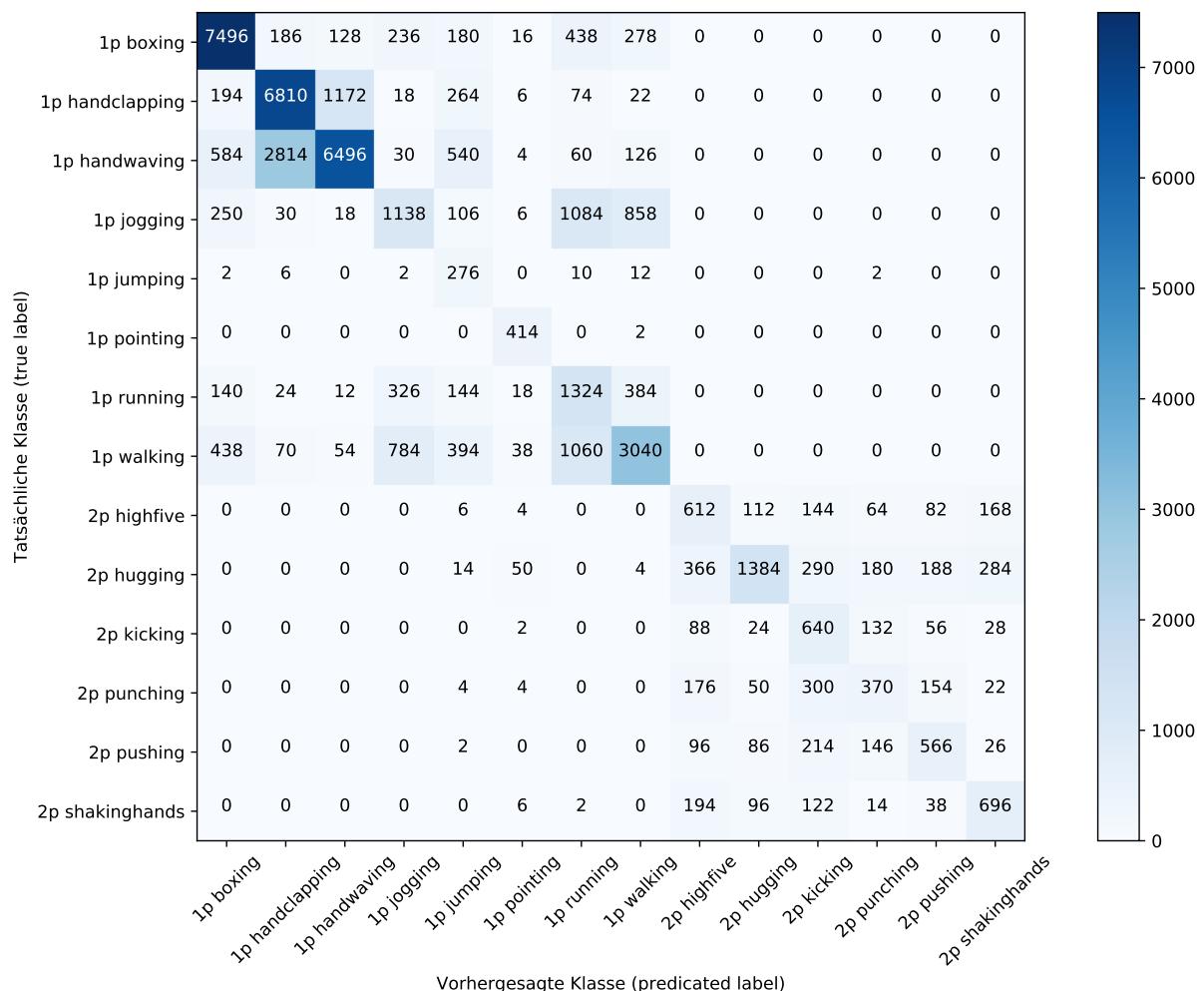


Abb. 4.5: Wahrheitsmatrix des Klassifikators des ersten Prototyps. Die Matrix zeigt, in welche Klassen die Videos des für das Training verwendeten Datensatzes klassifiziert werden.

Aus der in Abb. 4.5 gezeigten Wahrheitsmatrix lässt sich ablesen, dass beim Test des Klassifikators des ersten Prototyps zahlreiche Fehlklassifikationen vorkamen. Außerdem ist eine klare Trennung

zwischen den Klassen mit einer (1p) und zwei (2p) beteiligten Personen zu erkennen. Es kommt nur sehr selten vor, dass die Anzahl der beteiligten Personen falsch eingeschätzt wird.

Das schlechte Klassifikations-Ergebnis lässt sich unter anderem durch die Vielfalt des Datensatzes erklären. In diesem wurden zahlreiche Datensätze mit verschiedenen Kamerawinkeln, unterschiedlichen Kamerabewegungen und mehreren Szenen verwendet, und beim Zusammenstellen des Datensatzes wurde nicht auf eine ausgewogene Verteilung der Videos zu jeder Klasse geachtet.

Darüber hinaus variiert die Qualität und Auflösung der Videos zwischen und innerhalb der Klassen stark. Es wurde daher getestet, ob ein kleinerer Datensatz, dessen Videos manuell auf die Qualität der darin erkannten Posen überprüft werden können, zu besseren Ergebnissen führt.

Der KTH [109] Datensatz enthält rund 600 Videos für sechs verschiedene Handlungsklassen. Aus diesen Videos lassen sich insgesamt 194.292 Einzelbilder mit Körperhaltungen generieren, die zur Klassifikation genutzt werden können.

Nach einem Training mit 5.000 Epochen wurde eine Genauigkeit von 69,9% erreicht, genau sechs Prozentpunkte mehr als beim Test des großen Datensatzes. Eine Wahrheitsmatrix dieses Versuchs wird in Abb. 4.6a gezeigt.

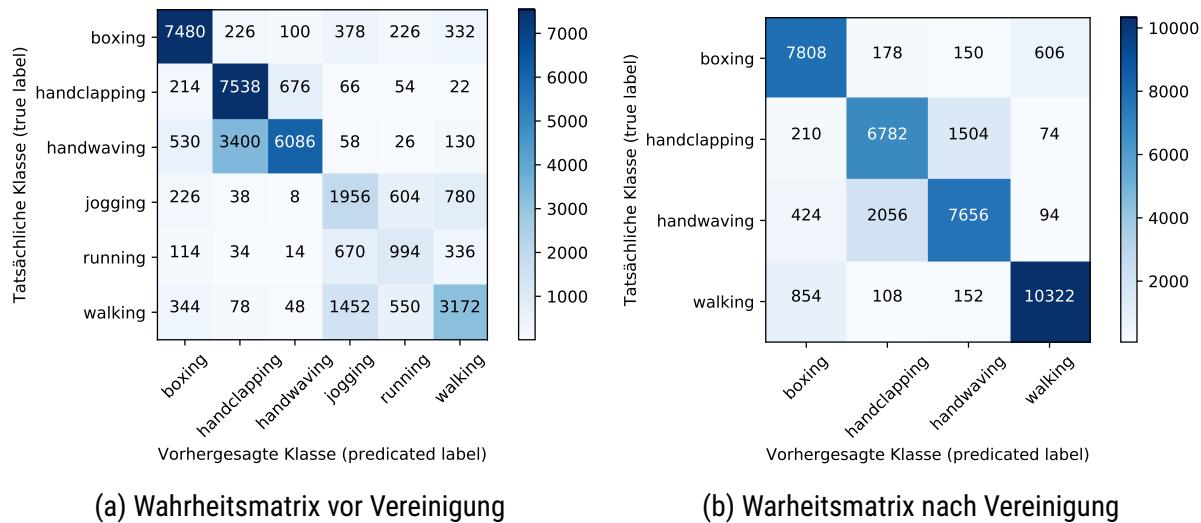


Abb. 4.6: Wahrheitsmatrizen des ersten Prototyps beim KTH Datensatz. In (a) gibt es zahlreiche Verwechslungen zwischen den Klassen *jogging*, *running* und *walking*. In (b) wird die Matrix nach Vereinigung der drei Klassen in die Klasse *walking* gezeigt.

Aus der Wahrheitsmatrix in Abb. 4.6a kann abgelesen werden, dass es einige Verwechslungen zwischen den Klassen *jogging*, *running* und *walking* gab.

Um zu überprüfen, inwieweit diese drei Klassen die Genauigkeit des Klassifikators beeinflussen, wurden diese ähnlichen Klassen miteinander vereinigt. Die neue Wahrheitsmatrix ist in Abb. 4.6b dargestellt. Nach Vereinigung der Klassen wurde nach ebenfalls 5.000 Schritten eine Genauigkeit von 83,6% erreicht. Die Genauigkeit ist also um rund 13 Prozentpunkte gestiegen.

Unabhängig vom verwendeten Datensatz berücksichtigt der erste Prototyp nur die räumliche Dimension des Videos (Einzelbilder), ohne den zeitlichen Zusammenhang zu den Vorgänger-Bildern des Videos herzustellen.

Im nächsten Prototyp soll daher auch geprüft werden, ob ein Einbezug der letzten Einzelbilder zu einer Verbesserung der Genauigkeit führt.

4.2.4 Installation und Ausführung

Der Quelltext für alle Prototypen findet sich auf der dieser Arbeit beiliegenden DVD-ROM sowie unter der folgenden Adresse im Internet:

<https://github.com/AlexanderMelde/Handlungserkennung>

Zur Installation des Prototyps sollte zunächst eine virtuelle Python-Umgebung mit TensorFlow, OpenCV und allen in der Datei requirements.txt genannten Modulen eingerichtet werden. Anschließend müssen der Compiler SWIG [8] sowie die Visual C++ Build Tools von Microsoft installiert werden, um für die Handlungserkennung benötigte C++ Bibliotheken zu komplizieren. Hierfür sollten die in Quellcode 4.1 gezeigten Befehle in der virtuellen Umgebung ausgeführt werden. [67]

```

1 cd tf_pose/pafprocess
2 swig -python -c++ pafprocess.i
3 python setup.py build_ext --inplace

```

Quellcode 4.1: Bauen der C++ Bibliotheken für die Handlungserkennung. [67]

Das Skript main.py kann zum Start aller Prototypen dieser Arbeit verwendet werden. Um die Funktionen des ersten Prototyps aufzurufen, können die in Quellcode 4.2 gezeigten Parameter gesetzt werden. Die Ordnerpfade und die numerischen Werte können beliebig angepasst werden. Um einzelne Schritte des Algorithmus zu überspringen, können die jeweiligen Parameter weggelassen werden.

```

1 python main.py \
2   --folder_train_vid_original="training/videos_original" \
3   --folder_train_img_original="training/images_original" \
4   --folder_train_img_pose_rgb="training/images_pose_rgb" \
5   --folder_train_img_pose_black="training/images_pose_black" \
6   --folder_train_data_pose="training/data_pose" \
7   --folder_classification_files="training/tf_files" \
8   --clean --delete_black_images \
9   --convert_videos_to_images --convert_each_nth_frame=10 \
10  --generate_poses \
11  --retrain --training_steps=5000 \
12  --classify_webcam --webcam_video_source=0

```

Quellcode 4.2: Aufruf der Handlungserkennung des ersten Prototyps.

Der Aufruf aus Quellcode 4.2 säubert das aktuelle Arbeitsverzeichnis, liest Videos ein, konvertiert diese in Einzelbilder und erkennt Körperhaltungen in den Einzelbildern. Anschließend werden die Bilder mit Körperhaltungen zum Trainieren eines neuronalen Netzes genutzt. Nach dem Training werden Handlungen im Bild einer an den Computer angeschlossenen Webcam klassifiziert.

Weitere Parameter, die zum Experimentieren mit den Prototypen gesetzt werden können, lassen sich mithilfe des Befehls `python main.py -h anzeigen`.

4.3 Prototyp 2: Auswertung von Raum und Zeit

Der zweite Prototyp berücksichtigt bei der Handlungs-Klassifikation nicht nur das aktuelle Standbild, sondern auch eine konfigurierbare Anzahl an vorangehenden Bildern.

In den bisherigen Ansätzen wurden zur Berücksichtigung von zeitlichen Informationen verschiedene Methoden eingesetzt, wie beispielsweise „Space-Time Shapes“ und der optische Fluss (siehe Abschnitt 3.1.2). Diese Ansätze gehen davon aus, dass es sich bei den darzustellenden Videos um Kamerabilder handelt.

In dieser Arbeit wird aber, wie bereits beim ersten Prototypen, nur anhand von Körperhaltungen klassifiziert. In diesem Prototyp soll daher geprüft werden, ob es für die skelettbasierte Handlungserkennung eine performantere Alternative gibt, die vergleichbare Ergebnisse erzeugt.

4.3.1 Implementierung

Für diesen Prototyp wurde ein Algorithmus entwickelt, der zunächst für eine bestimmte Anzahl von aufeinander folgenden Einzelbildern eines Videos Posen generiert (mit dem Algorithmus aus [67]) und diese anschließend überlagert.

Die hierbei entstehenden Bilder beschreiben damit Handlungen über mehrere Sekunden hinweg, und nicht mehr nur einzelne Momentaufnahmen (siehe Abb. 4.7).

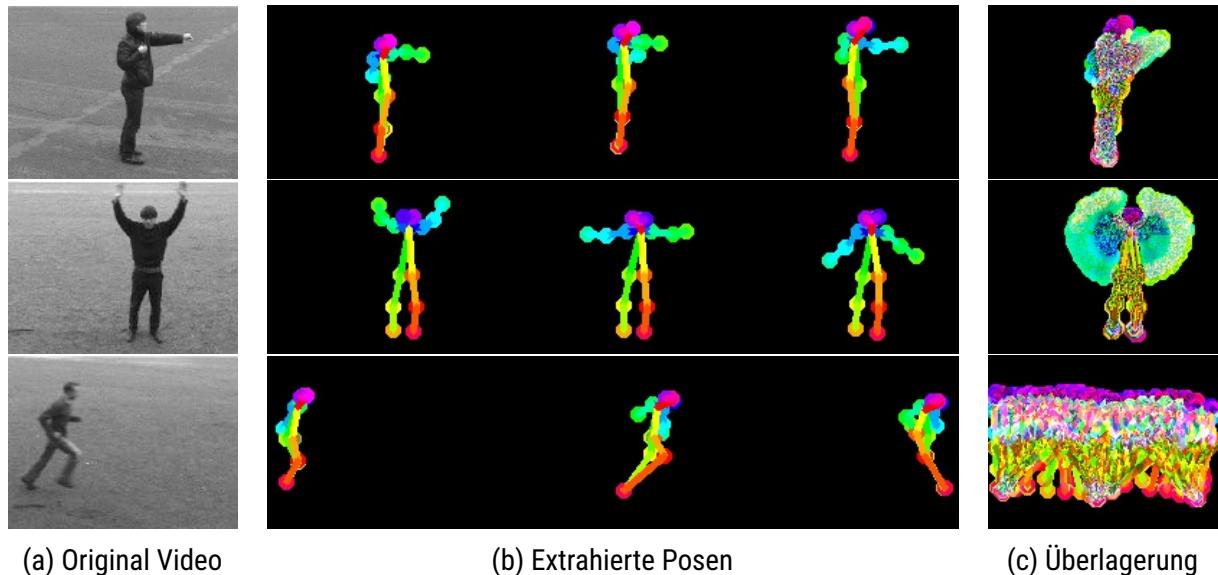


Abb. 4.7: Posen-Überlagerung im zweiten Prototyp. Das Original-Bild stammt aus dem KTH-Datensatz [109], die Körperhaltung wurde mit dem Algorithmus aus [67] berechnet, die Überlagerung mit der in diesem Abschnitt beschriebenen Technik.

Im Vergleich zum ersten Prototyp kann mit diesem Ansatz zum ersten Mal auch die Geschwindigkeit der ausgeführten Handlung berücksichtigt werden, was beispielsweise zur Unterscheidung von Aktivitäten wie „gehen“ und „schnelles gehen“ genutzt werden kann. Diese Unterscheidung ist hinsichtlich des angestrebten Anwendungsfalls essenziell, da in gefährlichen Situationen vermutlich schnellere Handlungen ausgeführt werden, als in harmlosen.

Die entwickelte Methode zur Überlagerung der Skelette hat den Nachteil, dass die Information, in welcher Reihenfolge die vorangegangenen Einzelbilder aufgetreten sind, in den überlagerten Bildern nicht gespeichert wird.

Werden lange Zeitabschnitte überlagert, so können Handlungen wie beispielsweise „Hand heben“ und „Hand senken“ nicht unterschieden werden, da die Information über die Richtung der Handbewegung fehlt.

Bei Handlungen, die, beispielsweise mit dem ersten Prototypen, bereits in Standbildern erkannt werden können, oder sich nur anhand ihrer Geschwindigkeit unterscheiden, tritt dieser Effekt nicht auf.

4.3.2 Ausführung

Der zweite Prototyp kann wie der erste ebenfalls über das Skript `main.py` gestartet werden. Ein beispielhafter Aufruf des zweiten Prototypen ist in Quellcode 4.3 gezeigt.

```

1 python main.py \
2   --folder_train_vid_original="training/videos_original" \
3   --folder_train_vid_as_img="training/videos_as_images" \
4   --folder_classification_files="training/tf_files" \
5   --convertVideosToOneImage \
6     --convert_each_nth_frame=10 \
7     --max_frames_per_vid=30 \
8     --split_videos \
9     --do_not_save_too_short_videos \
10    --retrain \
11      --training_steps=5000 \
12      --train_with_vid_as_img \
13      --classify_webcam --webcam_video_source=0

```

Quellcode 4.3: Aufruf der Handlungserkennung des zweiten Prototyps.

Der Aufruf aus Quellcode 4.3 liest Videos ein und überlagert alle in diesen Videos gefundenen Skelette in einem Bild. Mithilfe der Parameter `max_frames_per_vid` und `split_videos` kann der Algorithmus die Original-Videos in kleinere Videos mit einer überschaubaren Anzahl von Einzelbildern aufteilen. Anschließend werden die Bilder mit den Körperhaltungen zum Trainieren eines neuronalen Netzes genutzt. Nach dem Training werden wieder, wie im ersten Prototypen, Handlungen im Bild einer an den Computer angeschlossenen Webcam klassifiziert.

Eine Beschreibung der einzelnen Parameter kann weiterhin mit `python main.py -h` angezeigt werden.

Neben den aus dem ersten Prototypen bekannten Rückgaben wird dem Nutzer bei der Klassifikation von Videos und Webcam-Streams zudem das aktuelle und das zuletzt zur Klassifikation genutzte überlagerte Posen-Bild angezeigt (siehe Abb. 4.8).

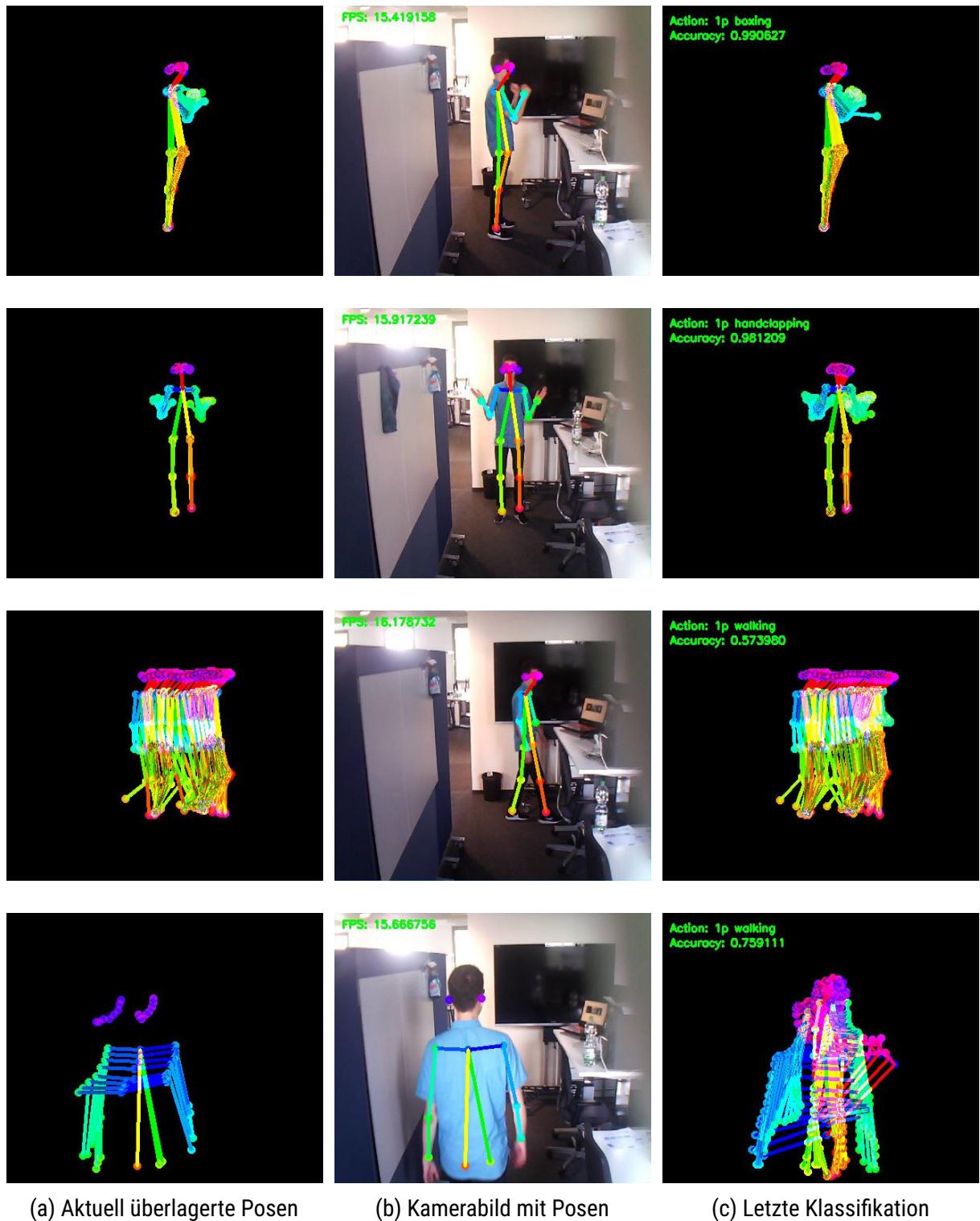


Abb. 4.8: Grafische Ausgabe des zweiten Prototyps. Bei der Klassifikation von Handlungen in einem Webcam-Stream werden neben den (a) bisher überlagerten Frames des aktuellen Zyklus auch (b) die aktuelle Pose im Kamerabild sowie (c) die zur Klassifikation genutzten überlagerten Posen der in diesem Fall letzten 30 Einzelbildern mit dem Klassifikations-Ergebnis angezeigt.

4.3.3 Test und Auswertung

Nach einem 5.000 Epochen langem Training des Klassifikators mit dem in Abschnitt 4.2.1 beschriebenen Datensatz wurde eine Genauigkeit von 85,2% erreicht. Das entspricht einer Steigerung von über 22 Prozentpunkte im Vergleich zum ersten Prototyp (siehe Abb. 4.5 auf Seite 52). Die besseren Klassifikations-Ergebnisse lassen sich auch auf den ersten Blick in der Wahrheitsmatrix in Abb. 4.9 erkennen. Es finden deutlich weniger Verwechslungen statt.

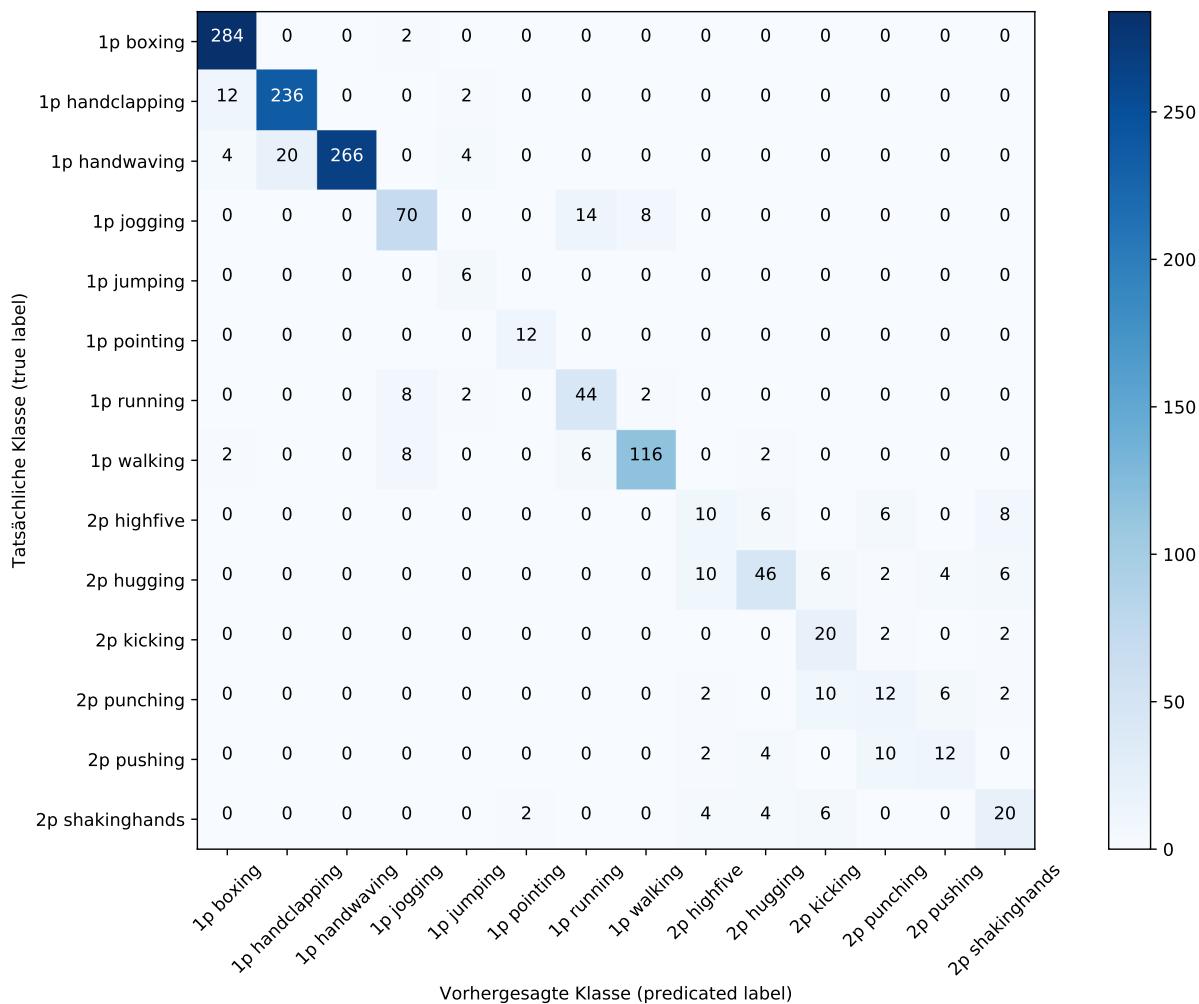


Abb. 4.9: Wahrheitsmatrix des Klassifikators des zweiten Prototyps. Die Matrix zeigt, dass im Vergleich zum ersten Prototyp deutlich weniger Verwechslungen stattfinden und weiterhin eine klare Trennung zwischen den Handlungen mit einer und mit zwei beteiligten Personen besteht.

Bei der genaueren Untersuchung der Wahrheitsmatrix fällt auf, dass die Anzahl der für jede Handlung klassifizierten Objekte deutlich geringer ist als beim vorherigen Prototyp. In erster Linie liegt das daran, dass während der erste Algorithmus direkt Einzelbilder aus den Videos zur Klassifikation genutzt hat, der zweite Prototyp mehrere Einzelbilder aus Videos extrahiert und diese überlagert.

Für den Test wurden die Parameter des Algorithmus zur Überlagerung so gewählt, dass nur jedes zehnte Einzelbild betrachtet wird und pro generiertem Bild mindestens 30 Skelette überlagert werden sollen. Das führt zu einer Mindestlänge von 300 Einzelbildern je Video und dazu, dass nicht alle Trainings-Videos in mehrere Überlagerungen aufgeteilt werden.

Aus den 1.197 Videos des Datensatzes aus Abschnitt 4.2.1 wurden 7.260 Überlagerungen mit Körperhaltungen generiert und zum Trainieren des Klassifikators genutzt. Der für den Test verwendete Teil, der in der Wahrheitsmatrix dargestellt wurde, besteht aus 1.354 Überlagerungen (rund 18% der Gesamtmenge).

Die Anzahl der Bilder für den Klassifikator ist im zweiten Prototyp deutlich geringer. Statt den 242.606 Einzelbildern im ersten Prototyp gibt es nun 7.260 Überlagerungen (circa 3%). Hieraus entsteht der Vorteil, dass das Trainieren des Klassifikators um ein Vielfaches schneller ist, und das, obwohl die Genauigkeit gestiegen ist.

Bei der Auswertung der Ergebnisse des ersten Prototyps wurde die Vermutung aufgestellt, dass ein kleinerer Datensatz zu besseren Ergebnissen führen könnte. Der zweite Prototyp wurde daher ebenfalls mit den rund 600 Videos der sechs Klassen des KTH-Datensatzes [109] trainiert, wobei nach 5.000 Epochen nun eine Genauigkeit von 85,7% erreicht wurde. Das sind über 15 Prozentpunkte mehr als bei der Klassifikation des gleichen Datensatzes mit dem ersten Prototyp. Die zugehörige Wahrheitsmatrix wird in Abb. 4.10a gezeigt.

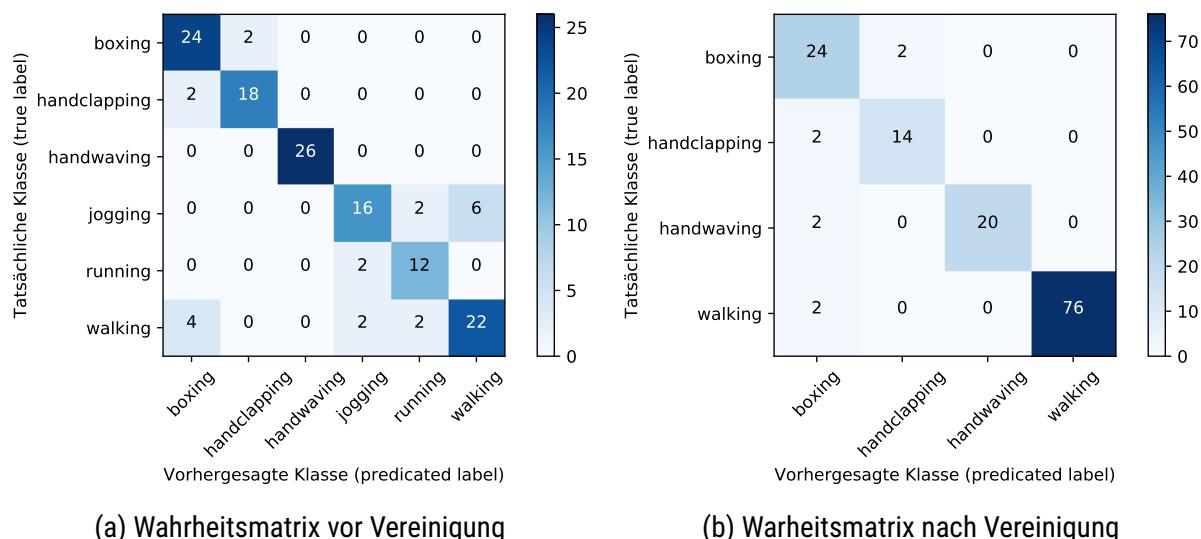


Abb. 4.10: Wahrheitsmatrizen des zweiten Prototyps beim KTH Datensatz. Es ist erkennbar, dass deutlich weniger Verwechslungen stattfinden, da nur wenige der klassifizierten Bilder außerhalb der Hauptdiagonale liegen.

In Abb. 4.11 wird gezeigt, wie sich die Genauigkeit der Klassifikation während des Trainings verändert. Es ist erkennbar, dass die Genauigkeit ab 1500 Epochen nur noch weniger stark zunimmt, und auch weniger Trainings-Schritte zu einem vergleichbaren Ergebnis führen würden.

Durch die Vereinigung der Klassen *jogging*, *running* und *walking* konnte die Genauigkeit, wie bereits im ersten Prototypen, um über 10 Prozentpunkte gesteigert werden: Nach einem 5.000 Schritte umfassendem Training konnte nun eine Genauigkeit von 96,3% erreicht werden.

Bei der Klassifikation von selbst-aufgenommenen Webcam-Videos, die nicht Teil des zusammengestellten Datensatzes sind, führte die Verwendung des auf den KTH [109] Datensatz trainierten Netzes zu weniger Fehlklassifikationen, als das Verwenden des größeren, zusammengesetzten Datensatzes. Das lässt sich unter anderem damit begründen, dass einige Videos des großen Datensatzes Kamerafahrten enthielten, die bei den zum Test verwendeten selbst-aufgenommenen Videos aus einer festen Perspektive nicht vorkamen.

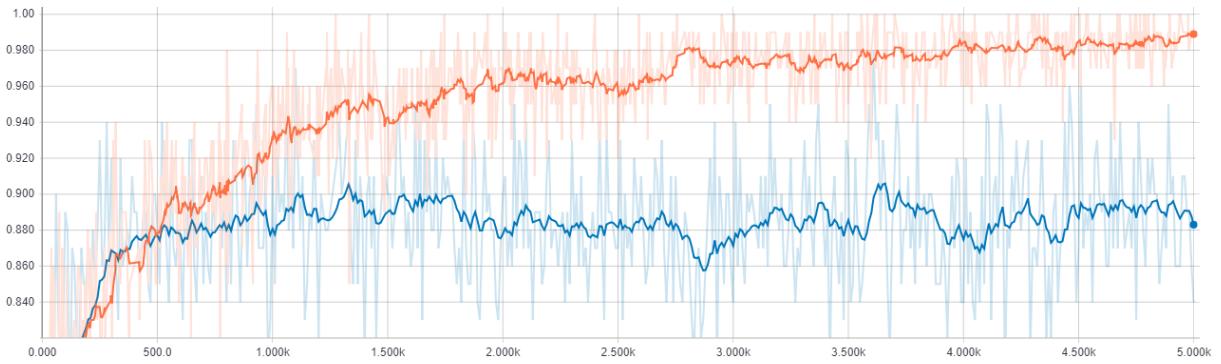


Abb. 4.11: Genauigkeit des zweiten Prototyps (vertikale Achse) bei zunehmender Anzahl an Trainings-Schritten (horizontale Achse). Dargestellt werden die geglätteten Genauigkeiten beim Test der trainierten Daten (orange) und beim Test mit nicht zum Training verwendeten Daten desselben Datensatzes (blau). In blassen Farben werden die jeweils ungeglätteten Daten dargestellt.

Das Überlagern der Einzelbilder ist sehr performant, da die aufwendigste Rechen-Operation eine Matrizen-Addition ist. In diesem Punkt ist die hier vorgestellte Technik der Überlagerung daher deutlich schneller als die bisherigen Ansätze. Bei Verwendung einer Webcam werden mit einer Geschwindigkeit von über 20 Bildern pro Sekunde Skelette für Körperhaltungen erkannt. Nur nach jeder vollständigen Überlagerung, je nach Einstellung also beispielsweise alle 300 Einzelbilder gibt es eine ein bis zwei Sekunden lange Wartezeit aufgrund der Klassifikation.

4.4 Prototyp 3: Auswerten von Zeitreihen mittels LSTM

Ein Video ist durch seine mehrdimensionale Struktur schwieriger auszuwerten als eine Zeitreihe numerischer Vektoren [4, S. 6].

In diesem Prototyp soll getestet werden, ob sich (1) Videos durch eine Abstraktion auf die darin vorkommenden Gelenk-Positionen als Zeitreihe mit weniger Dimensionen darstellen lassen und ob sich (2) diese Zeitreihen zur Klassifikation von Handlungen eignen.

4.4.1 Darstellung des Videos als Zeitreihe

Für diesen Prototyp wird die Annahme getroffen, dass die Kamera statisch ist und jedes Video genau eine Handlung enthält.

Wie in den vorherigen Prototypen wird zunächst die Körperhaltung der dargestellten Person berechnet. Diese kann als Kombination mehrerer Gelenk-Positionen angegeben werden (siehe Abb. 4.12).

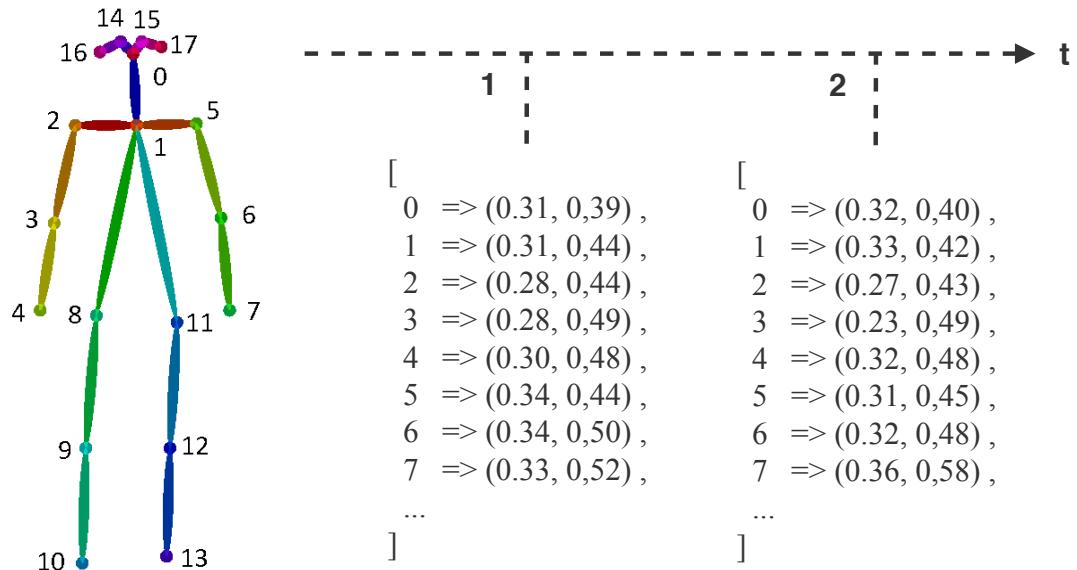


Abb. 4.12: Darstellung der Körperhaltungen in einem Video durch Gelenk-Positionen. Zu jedem Zeitpunkt t (also in jedem Einzelbild) existiert eine Liste, die die Nummer des jeweiligen Gelenks [18] zu einer relativen oder absoluten Position (x,y) im Bild zuordnet.

Die Gelenk-Positionen bestehen aus zwei Werten für die horizontale (x) und vertikale (y) Achse des Bilds. Bei 18 Gelenken ergeben sich daher für jeden Zeitpunkt 36 Werte. Ein Beispiel für einen konkreten dieser Werte wäre:

$$\text{relative X-Position des rechten Handgelenks im dritten Einzelbild} = 0.34 \quad (4.1)$$

In der Literatur finden sich Ansätze zur Klassifikation von Handlungen anhand von am Körper getragenen Beschleunigungssensoren [22]. Die Beschleunigung (acc) eines Gelenks ($joint$) eines Einzelbilds ($frame$) in eine Richtung ($axis$) lässt sich berechnen als die Differenz zwischen seiner aktuellen Position (pos) und der Position im vorangegangenen Einzelbild:

$$acc \left(\begin{array}{l} frame = t, \\ joint = 4, \\ axis = x \end{array} \right) = pos \left(\begin{array}{l} frame = t, \\ joint = 4, \\ axis = x \end{array} \right) - pos \left(\begin{array}{l} frame = (t - 1), \\ joint = 4, \\ axis = x \end{array} \right) \quad (4.2)$$

Im Rahmen dieser Arbeit wurde ein Skript entwickelt, das beliebige Videos als Eingabe nimmt, in diesen Videos Körperhaltungen erkennt und die hierbei berechneten Gelenk-Positionen als Zeitreihe darstellt.

Es werden zwei Typen von Signalen generiert:

1. Zeitlicher Verlauf der absoluten Positionen der einzelnen Gelenke im Bild.
2. Zeitlicher Verlauf der relativen Änderungen der absoluten Positionen der einzelnen Gelenke im Vergleich zum vorangegangenen Bild (Beschleunigung).

Die Darstellung von Gelenk-Positionen als zur Körpermitte relativ wurde als nicht sinnvoll erachtet, da die Körpermitte nicht in jedem Frame sichtbar ist.

Ein beispielhafter Signalverlauf eines Gelenks in einem Video ist in Abb. 4.13 gezeigt.

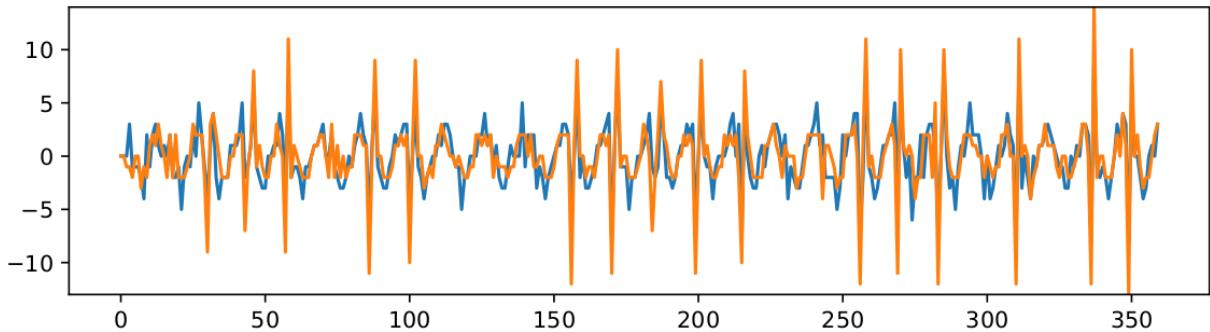


Abb. 4.13: Gelenk-Beschleunigung (vertikale Achse) des Gelenks mit der Nummer 6 im zeitlichen Verlauf (horizontale Achse) im Video *person01_boxing_d1_uncomp.avi* des KTH Datensatzes [109]. Die Beschleunigung in x-Achse ist in blau dargestellt, die Beschleunigung in y-Achse in orange.

Für jedes Video wurden der Anzahl der Gelenke, räumlichen Achsen und Signaltypen entsprechend $18 \cdot 2 \cdot 2 = 72$ Signalverläufe generiert und in JavaScript Object Notation (JSON) sowie in grafischer Darstellung als PDF-Datei auf der Festplatte gespeichert.

4.4.2 Handlungsklassifikation durch Zeitreihen

In diesem Abschnitt soll geprüft werden, ob die berechneten Signalverläufe zur Klassifikation von Handlungen geeignet sind.

Hierfür wurde der Handlungsklassifikator von Chevalier [22] verwendet. Dieser setzt ein LSTM-Netz ein, das auf einen Datensatz trainiert wurde, der mithilfe von Smartphones aufgezeichnete Beschleunigungsdaten enthält.

Dieser Handlungsklassifikator wurde zunächst installiert und mithilfe des erwähnten Smartphone-Datensatzes trainiert, wobei die von Chevalier [22] angegebenen Performance-Ergebnisse bestätigt werden konnten.

Anschließend wurde ein Skript geschrieben, das die aus den Videos berechneten Zeitreihen in das für diesen Algorithmus benötigte Format konvertiert und der Code von Chevalier [22] so angepasst, dass er statt den mitgelieferten Datensatz nun diese Daten verwendet.

Eine weitere Quelltext-Modifikation war notwendig, da das Signal aus 4.4.1 zweidimensionale Gelenkpositionen (x,y) besitzt und nicht durch Beschleunigungssensoren gewonnene dreidimensionale Daten (x,y,z). Auch die Anzahl der Signale musste angepasst werden. Statt zwei Sensoren mit je drei Achsen müssen für diesen Ansatz nun 18 Gelenke mit je zwei Achsen ausgewertet, also 36 statt sechs Signale.

Während des Versuchs, mit den konvertierten Datensätzen das neuronale Netz zu trainieren, wurde bemerkt, dass der gewählte Klassifikator von Chevalier [22] nur Signale gleicher Länge klassifizieren kann.

Da die Anpassung des Klassifikators für variable Signallängen voraussichtlich einige Zeit in Anspruch nehmen würde, wurde entschieden, dass für jedes Signal nur die ersten 100 Zeitwerte interpretiert werden. Diese sind für jedes der generierten Signale vorhanden, weshalb mit dieser Methode ein einheitlich langer Datensatz emuliert werden konnte. Es ist zu vermuten, dass durch diese Entscheidung die Qualität der Klassifikations-Ergebnisse abnimmt.

4.4.3 Test und Auswertung

Für einen Test des dritten Prototyps wurde zunächst der vereinigte KTH [109] Datensatz in Signale umgewandelt. Anschließend wurden diese Signale so strukturiert, das der modifizierte Klassifikator diese entgegennehmen kann. 30% des Datensatzes wurden für die Validierung verwendet. Bei einer Lernrate von 0,0025, einer Stapel-Größe (*batch size*) von 1.500 und 30.000 Iterationen konnte eine Genauigkeit von 97,2% erreicht werden.

Die Genauigkeit der Klassifizierung ist im zeitlichen Verlauf in Abb. 4.14 dargestellt. Es lässt sich erkennen, dass die Genauigkeit bei Klassifikation der Trainingsdaten bereits ab ca. 60.000 Iterationen bei 100% liegt. Die bei der Klassifikation der Testdaten erreichte Genauigkeit stagniert ab ungefähr 90.000 Iterationen.

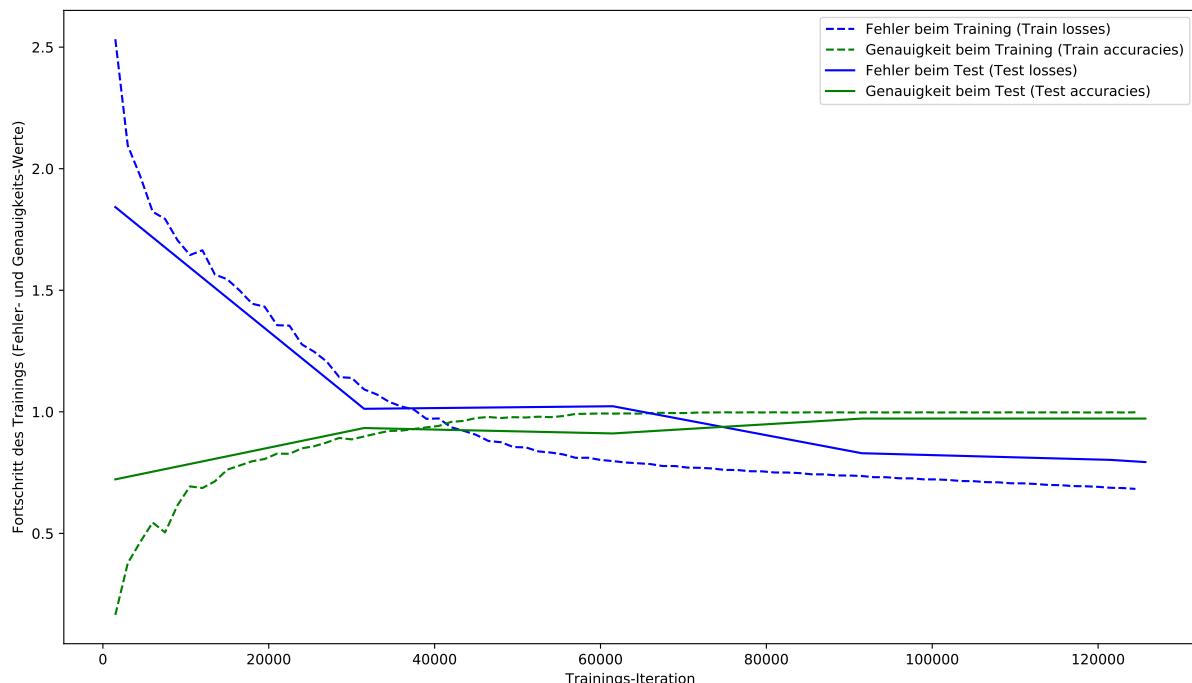


Abb. 4.14: Genauigkeit und Fehlerrate bei Training und Test des dritten Prototyps.

Bei der mehrmaligen Ausführung dieses Versuchs schwankte das Ergebnis in einem Bereich von 3% Prozent, und bei der genaueren Untersuchung des neuronalen Netzes kam es zu einigen Unklarheiten, die vor einem produktiven Betrieb des Prototyps genauer untersucht werden sollten. Es könnte sein, dass bei der Umstrukturierung der Signale nicht alle Trainingsdaten den korrekten Klassen zugeordnet wurden, wodurch das Ergebnis gefälscht werden würde.

Theoretisch sollte dieser Ansatz auch nach einer Prüfung möglicher Fehlerursachen sowie dem Beheben dieser Fehler bessere Ergebnisse liefern als die bisherigen Prototypen, da sich LSTMs, wie bereits in den Grundlagen erwähnt, gut zur Interpretation zeitlicher Verläufe eignen.

4.5 Auswertung der Experimente

Die besten Ergebnisse wurden mit dem zweiten Prototyp erzielt (siehe Abschnitt 4.3.3). Dieser erreichte eine Genauigkeit von bis zu 96,3%.

Die Ergebnisse der durchgeföhrten Experimente sind in Tabelle 4.1 dargestellt.

Experiment	Datensatz	Epochen	Genauigkeit
Prototyp 1 (S. 49)	Kombination (siehe S. 50)	5.000	63,9%
Prototyp 1 (S. 49)	KTH Datensatz [109]	5.000	69,9%
Prototyp 1 (S. 49)	KTH [109] mit vereinigten Klassen (S. 52)	5.000	83,6%
Prototyp 2 (S. 55)	Kombination (siehe S. 50)	5.000	85,2%
Prototyp 2 (S. 55)	KTH Datensatz [109]	5.000	85,7%
Prototyp 2 (S. 55)	KTH [109] mit vereinigten Klassen (S. 58)	5.000	96,3%
Prototyp 3 (S. 61)	KTH [109] mit vereinigten Klassen (S. 58)	120.000	97,2%

Tabelle 4.1: Ergebnisse der Experimente.

Aus Tabelle 4.1 kann abgelesen werden, dass die Genauigkeit in jedem Experiment gestiegen ist. Die Vermutungen hinsichtlich weiterer Genauigkeits-Verbesserungen haben sich jeweils bestätigt.

Der zweite Prototyp, der statt einzelnen Bildern die Überlagerungen mehrerer aufeinander folgender Bilder interpretiert, erreichte in allen Test-Szenarien bessere Ergebnisse als der erste Prototyp. Die Interpretation des zeitlichen Verlaufs von Handlungen scheint daher die Genauigkeit der Klassifikation zu erhöhen.

Es wird vermutet, dass mit dem später untersuchten dritten Prototypen noch deutlich bessere Ergebnisse erzielt werden können, da das dort verwendete LSTM zeitliche Zusammenhänge besser repräsentieren können soll als CNNs (siehe Abschnitt 2.2.5). In seiner aktuellen Implementation sollte der dritte Prototyp dennoch mit Vorsicht eingesetzt werden, da die Ursachen für die starken Schwankungen bei der Klassifikation noch nicht untersucht wurden.

5 Ausblick

In diesem Kapitel werden einige Erweiterungsmöglichkeiten zur Optimierung der Prototypen vorgestellt, unterteilt in Optimierungen der Erkennungsrate, der Geschwindigkeit, Funktionserweiterungen und Optimierungen hinsichtlich des Anwendungsfalls.

Anschließend werden einige Empfehlungen für das weitere Handeln des Unternehmens formuliert, die vor einer Marktreife des Produkts beachtet werden sollten.

5.1 Erweiterungsmöglichkeiten

Im Rahmen dieser Arbeit wurde mit dem ersten Prototyp eine funktionsfähige Handlungserkennung programmiert. In den späteren Prototypen wurde diese bereits in mehrfacher Hinsicht verbessert.

Dennoch wurde noch keine perfekte Erkennungsrate erreicht, und auch eine Echtzeit-Verarbeitung mit mehr als 20 Bildern pro Sekunde stellt noch eine Herausforderung dar.

Es wurden daher einige Erweiterungs- und Optimierungsmöglichkeiten erarbeitet, die im Folgenden vorgestellt werden.

5.1.1 Optimierung der Erkennungsrate

Verbesserung der Qualität der Eingangsdaten Bei hochauflösenden Eingabedaten in der Anwendungsphase, idealerweise mit wenig Bewegungsunschärfe, können kleinere Personen und deren Skelette besser erkannt werden als in kleinen und unscharfen Bildern.

Berücksichtigung der Häufigkeit von Handlungen Die Häufigkeitsverteilung der einzelnen Handlungen in der realen Welt ist nicht gleichmäßig. Es gibt in der Regel mehr ungefährliche Alltagshandlungen als beispielsweise gewalttätige Handlungen. Wenn zwei Handlungs-Klassen also eine ähnliche per TensorFlow berechnete Wahrscheinlichkeit haben (nur wenige Prozentpunkte zwischen zwei Klassen), dann sollte bei Berücksichtigung der Häufigkeit die im (deutschen) Raum häufiger vorkommende Handlung als Ergebnis angezeigt werden. Das Setzen eines solchen Schwellenwerts könnte im Anwendungsfall zudem zahlreiche Fehl-Alarmierungen verhindern. [87, S. 54]

Erhöhen der Trainingsdaten Wie bereits beschrieben gilt für auf maschinellem Lernen basierende Klassifikatoren grundsätzlich, dass diese mit steigender Anzahl an Trainingsdaten bessere Klassifikations-Ergebnisse generieren. Um mehr Trainingsdaten zu erhalten, können beispielsweise richtungsunabhängige Videos gespiegelt werden oder Skelette im Bild vergrößert, verkleinert oder verschoben werden, wenn die Handlung unabhängig von der Position im Bild ausgeführt werden kann.

5.1.2 Optimierung der Geschwindigkeit

Einsatz schnellerer Hardware Der Einsatz von leistungsstärkeren Hardware-Komponenten verbessert nicht die entwickelten Prototypen, kann aber zu einem besseren Nutzer-Erlebnis und höheren Geschwindigkeiten führen [87, S. 53]. Die entwickelten Prototypen wurden auf einem für Computerspieler ausgelegten „Gaming“-Notebook getestet. Speziell für Deep Learning entwickelte Grafikprozessoren wie die NVIDIA Tesla K40 und für den Hochleistungsbetrieb optimierte Prozessoren, beispielsweise Intel Xeon Prozessoren, sollten die Verarbeitungsgeschwindigkeit deutlich steigern können. Essenziell ist auch der Einsatz von ausreichend dimensioniertem und schnellem Arbeitsspeicher. [112]

Parallelisierung von Prozessor-Kernen Moderne Prozessoren bestehen meist aus mehreren Kernen, die unabhängig voneinander verschiedene Aufgaben abarbeiten können. Durch gezieltes Aufteilen der verschiedenen Verarbeitungsschritte sollte eine bessere Auslastung des Prozessors erzielt werden können.

Parallelisierung von Grafikkarten Für die entwickelten Prototypen wurde neben dem Prozessor auch eine Grafikkarte genutzt. Der entwickelte Code kann dank dem genutzten CUDA-Framework ohne weitere Anpassungen auch auf mehreren Grafikkarten ausgeführt werden.

Einsatz eines leichtgewichtigeren Frameworks Für diese Arbeit wurde das Framework TensorFlow gewählt, da es als einsteigerfreundlich und sehr umfangreich gilt. Mit dem Framework können alle gängigen Typen von neuronalen Netzen umgesetzt werden. Für eine Spezialisierung auf einen bestimmten Anwendungsfall bietet sich die Möglichkeit an, das umfangreiche TensorFlow-Framework durch eine leichtgewichtigere Alternative zu ersetzen, wie beispielsweise TensorFlow Lite oder Torch.

Einsatz eines leichtgewichtigeren Klassifikations-Modells Bei der Klassifikation der Einzelbilder und der überlagerten Skelett-Bilder wurde ein neuronales Netz verwendet, das zwar eine sehr genaue Klassifikationen ermöglicht, dafür aber etwas mehr Zeit in Anspruch nimmt. Für den Einsatz in Echtzeit-Systemen ist der Einsatz eines schnelleren Modells empfehlenswert. Im Skript `retrain_pose_classifier.py` kann hierfür der Parameter `--architecture` verändert werden. Mögliche Modelle, die verwendet werden können, sind in den Kommentaren im Quellcode angegeben.

5.1.3 Optimierungen hinsichtlich des Anwendungsfalls

Anpassen der Trainingsdaten an den Anwendungsfall Der verwendete KTH [109] Datensatz beinhaltet aus frontaler Perspektive aufgenommene Videos von je einer Person, die genau eine Handlung ausführt. Um Bilder aus Vogelperspektive zu klassifizieren, bietet es sich an, aus der gleichen Perspektive aufgenommene Videos zum Training zu verwenden. Falls bekannt ist, welche Bereiche im Bild wie genutzt werden, so ist auch eine Optimierung auf diese Bereiche möglich. Die Verarbeitungsgeschwindigkeit kann deutlich erhöht werden, wenn für die Klassifikation irrelevante Bereiche, die beispielsweise durch eine Hauswand verdeckt sind, nicht berücksichtigt werden.

Einsatz von kontinuierlichem Lernen Bisher wurde davon ausgegangen, dass überwachtes Lernen zum Trainieren des Klassifikators eingesetzt wird. Wenn die erkannten Handlungen durch eine menschliche Stelle verifiziert werden, beispielsweise um Fehlalarme während der ersten Monate zu vermeiden, so wäre auch eine Bestätigung von korrekten Handlungen und Korrektur von falsch klassifizierten Handlungen möglich.

5.1.4 Funktionserweiterung

Grenzwert für Alarmierung Einleitend wurde beschrieben, dass bei gefährdenden Handlungen automatisch um Hilfe gerufen werden könnte. Um Fehlalarme zu vermeiden, ist es sinnvoll, Rettungskräfte oder einen Sicherheitsdienst erst dann zu benachrichtigen, wenn die Handlung über einen längeren Zeitraum hinweg detektiert wurde. Auch eine Kombination dieses Sensors (Handlungserkennung) mit weiteren Sensoren, wie beispielsweise einer Audioanalyse, kann sinnvoll sein, um das Gefahrenpotential genauer einzuschätzen.

Anwendung in einem Rechenzentrum oder direkt in der Kamera Die in dieser Arbeit entwickelten Prototypen wurden auf einem Laptop getestet, der als Eingabe das Kamerabild einer Webcam verwendet. Für ein Anbieten der Handlungserkennung als Dienstleistung bietet sich eine zentrale Verarbeitung in einem Rechenzentrum an. Die entwickelten Prototypen verwenden die Webcam-Schnittstelle der Bibliothek OpenCV, weshalb eine Anpassung des Eingabe-Bilds von der Webcam hin zu einem Livestream, beispielsweise im Real-Time Streaming Protocol (RTSP)-Format, mit dem Ändern eines einzigen Parameters (- -webcam_video_source) erfolgen kann. Auch eine Integration der für die Klassifikation notwendigen Software in die Kamera oder einen an diese montierten Mini-PC wäre denkbar.

Erweiterung des zweiten Prototyps zur Auswertung der zeitlichen Reihenfolge Der zweite Prototyp berücksichtigt die Zeit bereits insofern, dass mehrere hintereinanderfolgende Bilder überlagert werden. Aus den Überlagerungen ist aber nicht mehr erkennbar, in welcher Reihenfolge diese überlagert wurden. In einer Modifikation des Prototypen könnte getestet werden, wie sich die Klassifikations-Genauigkeit verändert, wenn die Skelette statt übereinander nebeneinander angeordnet werden. Auch eine Kodierung der Reihenfolge mithilfe verschiedener Transparenz- oder Helligkeits-Werte wäre möglich. Eine weitere Methode wäre das Einfärben der nacheinanderfolgenden Skeletten in verschiedene Farben eines Farbverlaufs.

Unterscheidung von Gefahren-Situationen Die bereits vorgestellten Prototypen dienten zur Unterscheidung von einfachen Handlungen wie Winken, Springen, Boxen und Rennen. In einer Modifikation der Prototypen kann getestet werden, wie sich die Genauigkeit der Klassifikation verändert, wenn statt mehrerer detaillierter Klassen nur binär zwischen ruhigen und gefährlichen Situationen unterschieden wird.

5.2 Handlungsempfehlung

In diesem Abschnitt werden die für das Unternehmen wichtigsten Erkenntnisse ausgewertet, um eine Empfehlung für weiteres Handeln zu formulieren.

Mit dem zweiten Prototyp konnten beim Test eines kleinen Datensatzes eine Genauigkeit von über 96% erreicht werden. Es ist zu beachten, dass diese Genauigkeit beim Test mit einem Benchmark-Datensatz entstanden ist. Bei der Anwendung des Prototyps in realen Umgebungen sind zunächst deutlich schlechtere Genauigkeiten zu erwarten.

Die Genauigkeit sollte deutlich verbessert werden können, indem (1) mehr klassifizierte Beispielvideos verwendet werden, (2) die zum Training verwendeten Videos skaliert, verschoben oder gespiegelt werden und (3) durch farbliche Markierung der überlagerten Skelette auch die Reihenfolge der Überlagerungen berücksichtigt wird. Bei der (4) Auswahl der Trainingsdaten sollten die Position und die Perspektive der Kamera im Anwendungsfall berücksichtigt werden. Darüber hinaus sollten (5) die Klassen dem Anwendungsfall entsprechend gewählt werden, für die Hilferuf-Funktion wäre beispielsweise eine Klassifikation von stürzenden Menschen sinnvoll.

Bei der Anwendung des Prototyps auf Webcam-Bilder wurde gezeigt, dass das Programm Körperhaltungen in Echtzeit berechnet und Handlungen nach einer Verzögerung von circa einer Sekunde klassifizieren kann. Mit einer (6) Verbesserung der zur Verfügung stehenden Hardware-Ressourcen sowie einer (7) Optimierung durch Parallelisierung des Programmcodes sollte eine Echtzeit-Geschwindigkeit erreicht werden können.

Hinsichtlich des Anwendungsfalls ist zu beachten, dass für diesen mehrere Personen in einem kontinuierlichen Video berücksichtigt werden müssen, wobei jede Person mit anderen Personen interagieren kann und mehrere Handlungen parallel und hintereinander ausgeführt werden können.

Hierfür ist eine Erweiterung des entwickelten Prototyps notwendig. Im einfachsten Fall kann zur Detektion von parallelen Handlungen ein (8) Zuschnitt auf einzelne Personen erfolgen. Das ist anhand der bereits berechneten Körperhaltungen einfach möglich. Anschließend können die (9) zugeschnittenen Videos getrennt und parallelisiert klassifiziert werden, sodass (10) den verschiedenen Bereichen im Bild verschiedene Handlungen zugeordnet werden können.

Zur zeitlichen Unterteilung von Handlungen sollte (11) die, bereits jetzt zurückgegebene Genauigkeit einer Klassifikation als Grenzwert genutzt werden. Nur wenn ein aufgenommener Video-Zuschnitt zu beispielsweise über 70% einer bestimmten Handlung entspricht, sollte die jeweilige Handlung in das Video eingezeichnet werden.

Nach diesen Anpassungen sollte ein gut vorzeigbarer *Proof of Concept* entstehen, der Handlungen in Videos zuverlässig erkennt. Details zur Umsetzung dieser Anpassungsmöglichkeiten sind in Abschnitt 5.1 beschrieben.

6 Fazit

In den theoretischen Grundlagen wurden einige Anwendungsfälle im Bereich der Videoanalyse und die Themen digitale Bildverarbeitung, Anonymisierung, Erkennung von Körperhaltungen und maschinelles Lernen einführend beschrieben. Hierbei wurden auch einige Spezialformen neuronaler Netze gezeigt. Ein Verständnis dieser Themen ist eine wichtige Voraussetzung für das Nachvollziehen der folgenden Argumentationen.

Im ersten Hauptteil dieser Arbeit wurden zunächst 26 verschiedene Ansätze zur Handlungserkennung aufgezeigt und die dahinter liegenden Techniken erklärt und miteinander verglichen. Es hat sich herausgestellt, dass die moderneren Ansätze auf Convolutional Neural Networks (CNNs) oder Kombinationen von CNNs mit Long Short-Term Memory (LSTMs) Netzen basieren.

Anschließend wurde erläutert, wie eine Handlungserkennung mithilfe von Körperhaltungen funktionieren kann, und welche Vor- und Nachteile diese Methode hat. Die Grundidee hierbei ist es, statt dem Original-Bild der Kamera auch oder ausschließlich die anonymisierten Skelett-Daten zu klassifizieren.

Der größte Vorteil dieser Methode ist, dass die Modelle bereits nach einem kurzen Training einen großen Bereich an Anwendungsgebieten abdecken, da der Hintergrund bei der Klassifikation keine oder eine kleinere Rolle spielt. Durch die Abstraktion auf die Körperhaltung gehen allerdings auch Daten verloren, die nur bei einer Kombination mit dem Original-Bild berücksichtigt werden können.

Für das Training eines Handlungs-Klassifikators werden Beispielvideos benötigt. In der Mitte des ersten Hauptteils wurden daher 61 Datensätze zur Handlungserkennung miteinander verglichen. Eine große Herausforderung bei der Erstellung eines Datensatzes zur Handlungs-Klassifikation ist die richtige Kategorisierung der Handlungen. Es gibt kein standardisiertes Schema für die Benennung von Handlungen, und auch die Detailstufen, mit denen Handlungen kategorisiert wurden, sind in den verschiedenen Datensätzen unterschiedlich.

Beim Vergleich der Datensätze hat sich darüber hinaus herausgestellt, dass die Anzahl der jeweils enthaltenen Videos in den letzten drei Jahren deutlich gestiegen ist. Aufgrund der zeitgleich steigenden Dauer, die für das Training dieser Datensätze erforderlich ist, gibt es bisher aber noch keinen Datensatz, der sich als Standard-Benchmark für die Handlungserkennung durchgesetzt hat. Der bisher größte Datensatz ist Youtube8M [1]. Dieser enthält 6,1 Millionen Videos von 3.862 verschiedenen Handlungen.

Anschließend wurde geprüft, worauf geachtet werden muss, wenn ein allgemeiner Handlungs-Klassifikator für den einleitend beschriebenen Anwendungsfall angepasst werden soll. Einige große Datensätze eignen sich beispielsweise nicht für eine Erkennung mithilfe von Körperhaltungen aus Vogelperspektive, da deren Videos Nahaufnahmen sind, die nicht den gesamten Körper zeigen. Bei der Untersuchung der bisherigen Ansätze fiel zudem auf, dass diese meist nur Videos, die je eine Handlung zeigen, klassifizieren können und sowohl zeitlich aufeinander folgende als auch parallel in einem Bild stattfindende Handlungen nicht klassifizieren können.

Im letzten Abschnitt des ersten Hauptteils wurde zusammengefasst, welche Herausforderungen es im Bereich der Handlungserkennung noch gibt. Neben zu kleinen Datensätzen und einem hohen Rechenaufwand für das Training des Klassifikators sind vor allem die Vielseitigkeit der Netzarchitekturen, das Fehlen eines standardisierten Benchmark-Datensatzes sowie der Einbezug der zeitlichen Zusammenhänge Themen, die die Forschung im Bereich der Handlungserkennung verlangsamen.

Im zweiten Hauptteil wurden nach einer Projektplanung drei Prototypen zur Handlungserkennung entwickelt. Die ersten beiden Prototypen basieren auf einem Bildklassifikator, der entweder reine Skelett-Daten oder ein in dieser Arbeit entwickeltes Format mit überlagerten Skeletten, die einen zeitlichen Verlauf mit einbeziehen klassifiziert. Der dritte Prototyp konvertiert Videos in Signale, die die Positionen der Skelett-Gelenke im Verlauf der Zeit darstellen und verfolgt anschließend den Ansatz, diese Signale zu klassifizieren.

Beim Test des KTH [109] Datensatzes konnte mit dem zweiten Prototyp eine Genauigkeit von 85,7% erreicht werden, beim Test einer modifizierten Version dieses Datensatzes sogar eine Genauigkeit von 96,3%. Der für die Prototypen geschriebene Quelltext liegt dieser Arbeit bei und kann für eine kontinuierliche Klassifikation von Webcam-Bildern verwendet werden.

Bei der Implementierung des dritten Prototyps mussten aufgrund einer Einschränkung des gewählten Klassifikators mehrere Kompromisse eingegangen werden, die teilweise zu einer inkonsistenten und ungenauen Klassifikation führen. Bei der Implementierung der Prototypen kam es ansonsten dank der guten Voruntersuchung und Projektplanung zu keinen weiteren Schwierigkeiten.

Im letzten Teil dieser Arbeit wurden einige Methoden zur Verbesserung der Erkennungsrate und der Verarbeitungsgeschwindigkeit aufgezeigt und Ideen zur Erweiterung der Funktion und zur Anpassung an den Anwendungsfall beschrieben. Mit diesen Ansätzen sollte es möglich sein, die erreichten Ergebnisse noch weiter zu verbessern.

Die Untersuchung der bisherigen Ansätze sowie die Genauigkeiten der entwickelten Prototypen sind ein starker Beweis für die anfangs aufgestellte These, durch eine Auswertung von Körperhaltungen sei eine Erkennung von Handlungen möglich.

Die untersuchten Ansätze, Datensätze und Herausforderungen sowie die entwickelten Prototypen bieten zudem eine umfangreiche Grundlage für die Entwicklung von Dienstleistungen im Bereich der Analyse von Handlungen.

Literaturverzeichnis

- [1] Sami Abu-El-Haija u. a. „Youtube-8M: A large-scale video classification benchmark“. In: *arXiv preprint arXiv:1609.08675* (2016) (ref. auf S. 36, 37, 40, 70).
- [2] Abien Fred Agarap. „Deep Learning using Rectified Linear Units (ReLU)“. In: *arXiv preprint arXiv:1803.08375* (2018) (ref. auf S. 15).
- [3] Saad Ali und Mubarak Shah. „A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis“. In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE. 2007, S. 1–6 (ref. auf S. 35).
- [4] Maryam Asadi-Aghbolaghi u. a. „A survey on deep learning based approaches for action and gesture recognition in image sequences“. In: *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on.* IEEE. 2017, 476–483 (ref. auf S. 10, 22, 33, 34, 61).
- [5] I. Askoxylakis u. a. *Computer Security – ESORICS 2016.* LNCS Part 2. Springer International Publishing, 2016. ISBN: 9783319457413 (ref. auf S. 7).
- [6] Moez Baccouche u. a. „Sequential deep learning for human action recognition“. In: *International Workshop on Human Behavior Understanding.* Springer. 2011, S. 29–39 (ref. auf S. 27, 29).
- [7] Davide Baltieri, Roberto Vezzani und Rita Cucchiara. „3DPeS: 3d people dataset for surveillance and forensics“. In: *Proceedings of the 2011 joint ACM workshop on Human gesture and behavior understanding.* ACM. 2011, S. 59–64 (ref. auf S. 35).
- [8] David M Beazley u. a. „SWIG: An Easy to Use Tool for Integrating Scripting Languages with C and C++.“ In: *Tcl/Tk Workshop.* 1996 (ref. auf S. 54).
- [9] Hakan Bilen u. a. „Dynamic image networks for action recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, S. 3034–3042 (ref. auf S. 24).
- [10] C.M. Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. Springer New York, 2016. ISBN: 9781493938438 (ref. auf S. 17).
- [11] Victoria Bloom, Vasileios Argyriou und Dimitrios Makris. „G3Di: A gaming interaction dataset with a real time detection and evaluation framework“. In: *Workshop at the European Conference on Computer Vision.* Springer. 2014, S. 698–712 (ref. auf S. 38).
- [12] Victoria Bloom, Vasileios Argyriou und Dimitrios Makris. „Hierarchical transfer learning for online recognition of compound actions“. In: *Computer Vision and Image Understanding* 144 (2016), S. 62–72 (ref. auf S. 38).
- [13] Victoria Bloom, Dimitrios Makris und Vasileios Argyriou. „G3D: A gaming action dataset and real time action recognition evaluation framework“. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on.* IEEE. 2012, S. 7–12 (ref. auf S. 38).
- [14] Bernhard E Boser, Isabelle M Guyon und Vladimir N Vapnik. „A training algorithm for optimal margin classifiers“. In: *Proceedings of the fifth annual workshop on Computational learning theory.* ACM. 1992, S. 144–152 (ref. auf S. 12).

- [15] *Bundesdatenschutzgesetz (BDSG) §5 vom 30. Juni 2017 (BGBl. I S. 2097)* (ref. auf S. 5).
- [16] R.W. Burns und Institution of Electrical Engineers and Science Museum (Great Britain). *Television: An International History of the Formative Years. History and Management of Technology Series.* Institution of Electrical Engineers, 1998. ISBN: 9780852969144 (ref. auf S. 3).
- [17] Fabian Caba Heilbron u. a. „ActivityNet: A large-scale video benchmark for human activity understanding“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, S. 961–970 (ref. auf S. 37).
- [18] Zhe Cao u. a. „Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields“. In: *arXiv preprint arXiv:1611.08050* (2016) (ref. auf S. 9, 10, 31, 50, 61).
- [19] Cristian Sminchisescu Catalin Ionescu Fuxin Li. „Latent Structured Models for Human Pose Estimation“. In: *International Conference on Computer Vision*. 2011 (ref. auf S. 38).
- [20] Jose M Chaquet, Enrique J Carmona und Antonio Fernández-Caballero. „A survey of video datasets for human action and activity recognition“. In: *Computer Vision and Image Understanding* 117.6 (2013), S. 633–659 (ref. auf S. 34).
- [21] Guilhem Chéron, Ivan Laptev und Cordelia Schmid. „P-CNN: Pose-based CNN features for action recognition“. In: *Proceedings of the IEEE international conference on computer vision*. 2015, S. 3218–3226 (ref. auf S. 25, 27, 29).
- [22] Guillaume Chevalier. *LSTMs for Human Activity Recognition*. Version 53132c8. 2016. URL: <https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition> (besucht am 02.07.2018) (ref. auf S. 62, 63).
- [23] Kyunghyun Cho u. a. „On the properties of neural machine translation: Encoder-decoder approaches“. In: *arXiv preprint arXiv:1409.1259* (2014) (ref. auf S. 19).
- [24] Wongun Choi, Khuram Shahid und Silvio Savarese. „What are they doing?: Collective activity classification using spatio-temporal relationship among people“. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE. 2009, S. 1282–1289 (ref. auf S. 37).
- [25] Andreas Cyffka. *PONS Kompaktwörterbuch: Deutsch als Fremdsprache*. Pons, 2007 (ref. auf S. 11).
- [26] Navneet Dalal und Bill Triggs. „Histograms of oriented gradients for human detection“. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Bd. 1. IEEE. 2005, S. 886–893 (ref. auf S. 23).
- [27] Navneet Dalal, Bill Triggs und Cordelia Schmid. „Human detection using oriented histograms of flow and appearance“. In: *European conference on computer vision*. Springer. 2006, S. 428–441 (ref. auf S. 22, 24).
- [28] Fernando De la Torre u. a. „Guide to the carnegie mellon university multimodal activity (cmu-mmact) database“. In: *Robotics Institute* (2008), S. 135 (ref. auf S. 38).
- [29] Claire-Hélène Demarty u. a. „VSD, a public dataset for the detection of violent scenes in movies: design, annotation, analysis and evaluation“. In: *Multimedia Tools and Applications* 74.17 (2015), S. 7379–7404 (ref. auf S. 35).
- [30] Jia Deng u. a. „ImageNet: A large-scale hierarchical image database“. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, S. 248–255 (ref. auf S. 11, 34, 43).

- [31] Ali Diba u. a. „Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification“. In: *arXiv preprint arXiv:1711.08200* (2017) (ref. auf S. 17, 27, 29, 30).
- [32] Jeffrey Donahue u. a. „Long-term recurrent convolutional networks for visual recognition and description“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, 2625–2634 (ref. auf S. 17, 19, 25–27, 29).
- [33] Jeff Donahue u. a. „Decaf: A deep convolutional activation feature for generic visual recognition“. In: *International conference on machine learning*. 2014, S. 647–655 (ref. auf S. 51).
- [34] Walter Dornberger. V-2. The Bantam war book series. New York: Ballantine Books, 1954. ISBN: 9780553126600 (ref. auf S. 3).
- [35] Yong Du, Wei Wang und Liang Wang. „Hierarchical recurrent neural network for skeleton based action recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 1110–1118 (ref. auf S. 10, 27, 29, 30, 32).
- [36] Jeffrey L Elman. „Finding structure in time“. In: *Cognitive science* 14.2 (1990), S. 179–211 (ref. auf S. 18).
- [37] EnBW Energie Baden-Württemberg AG. „EnBW-Halbjahresfinanzbericht Januar bis Juni 2018“. In: (Juli 2018). URL: <https://enbw.com/finanzberichte> (besucht am 13.08.2018) (ref. auf S. 1).
- [38] EnBW Energie Baden-Württemberg AG. „EnBW-Integrierter Geschäftsbericht 2017“. In: (März 2018). URL: <https://enbw.com/bericht2017> (besucht am 13.08.2018) (ref. auf S. 1).
- [39] Christoph Feichtenhofer, Axel Pinz und Andrew Zisserman. „Convolutional two-stream network fusion for video action recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 1933–1941 (ref. auf S. 12, 17, 22, 25, 27, 29, 34, 40).
- [40] Robert Fisher, Jose Santos-Victor und James Crowley. *CAVIAR: Context Aware Vision using Image-based Active Recognition*. Sep. 2005. URL: <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/> (besucht am 04.07.2018) (ref. auf S. 37).
- [41] J.D. Foley u. a. *Computer Graphics: Principles and Practice*. Addison-Wesley systems programming series. Addison-Wesley, 1996. ISBN: 9780201848403 (ref. auf S. 6).
- [42] Yun Fu. *Human Activity Recognition and Prediction*. Springer International Publishing, 2015. ISBN: 9783319270043 (ref. auf S. 34).
- [43] Rohit Ghosh. „Deep Learning for Videos: A 2018 Guide to Action Recognition“. In: *Qure.ai Blog* (Juni 2018). URL: <http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review> (besucht am 08.08.2018) (ref. auf S. 42, 43).
- [44] Rohit Girdhar und Deva Ramanan. „Attentional pooling for action recognition“. In: *Advances in Neural Information Processing Systems*. 2017, S. 33–44 (ref. auf S. 41).
- [45] Georgia Gkioxari und Jitendra Malik. „Finding action tubes“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 759–768 (ref. auf S. 25, 27, 29).
- [46] Wenjuan Gong u. a. „Human pose estimation from monocular images: a comprehensive survey“. In: *Sensors* 16.12 (2016), S. 1966 (ref. auf S. 9).
- [47] Melvyn A Goodale und A David Milner. „Separate visual pathways for perception and action“. In: *Trends in neurosciences* 15.1 (1992), S. 20–25 (ref. auf S. 22).

- [48] A. Gorban u. a. *THUMOS Challenge: Action Recognition with a Large Number of Classes*. <http://www.thumos.info/>. 2015 (ref. auf S. 34, 35, 37).
- [49] Lena Gorelick u. a. „Actions as space-time shapes“. In: *IEEE transactions on pattern analysis and machine intelligence* 29.12 (2007), S. 2247–2253 (ref. auf S. 24, 34, 35, 37, 41, 50).
- [50] Alex Graves und Navdeep Jaitly. „Towards end-to-end speech recognition with recurrent neural networks“. In: *International Conference on Machine Learning*. 2014, S. 1764–1772 (ref. auf S. 19).
- [51] Chunhui Gu u. a. „AVA: A video dataset of spatio-temporally localized atomic visual actions“. In: *CoRR, abs/1705.08421* 4 (2017) (ref. auf S. 36, 37).
- [52] Rıza Alp Güler, Natalia Neverova und Iasonas Kokkinos. „DensePose: Dense human pose estimation in the wild“. In: *arXiv preprint arXiv:1802.00434* (2018) (ref. auf S. 9, 10, 31).
- [53] Heinrich Rüttgerdt GmbH & Co KG. *Webcam Marktplatz Einbeck*. URL: <https://www.einbecker-morgenpost.de/webcam-einbeck.html> (besucht am 06.08.2018) (ref. auf S. 35).
- [54] Sepp Hochreiter und Jürgen Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997), S. 1735–1780 (ref. auf S. 19).
- [55] Berthold KP Horn und Brian G Schunck. „Determining optical flow“. In: *Artificial intelligence* 17.1-3 (1981), S. 185–203 (ref. auf S. 22, 23).
- [56] IntelliVision. *IntelliVision Intelligent Video Analytics for Smart Cameras*. URL: <https://youtu.be/Lr70N0ATgT8> (besucht am 06.08.2018) (ref. auf S. 4).
- [57] Catalin Ionescu u. a. „Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014) (ref. auf S. 38).
- [58] David Iskander. *Person, coffee, hands and book HD photo*. März 2018. URL: <https://unsplash.com/photos/8hFiT80X-6o> (besucht am 21.08.2018) (ref. auf S. 48).
- [59] Dirk Jensen. *Livestream Webcam am Markt*. URL: <https://www.bredstedt.cam/webcam-markt.php> (besucht am 06.08.2018) (ref. auf S. 35).
- [60] Hueihan Jhuang u. a. „Towards understanding action recognition“. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, S. 3192–3199 (ref. auf S. 34, 38).
- [61] Shuiwang Ji u. a. „3D convolutional neural networks for human action recognition“. In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2013), S. 221–231 (ref. auf S. 17, 26, 27, 29).
- [62] Rohan Kapur. „Rohan #4: The vanishing gradient problem“. In: *A Year of Artificial Intelligence* (März 2013). URL: <https://ayearofai.com/ec68f76ffb9b> (besucht am 23.08.2018) (ref. auf S. 15, 19).
- [63] Rohan Kapur und Lenny Khazan. „Rohan & Lenny #3: Recurrent Neural Networks & LSTMs“. In: *A Year of Artificial Intelligence* (Apr. 2017). URL: <https://ayearofai.com/10300100899b> (besucht am 22.08.2018) (ref. auf S. 18, 19).
- [64] Ujjwal Karn. „An Intuitive Explanation of Convolutional Neural Networks“. In: *the data science blog* (Aug. 2016). URL: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/> (besucht am 19.06.2018) (ref. auf S. 16, 17).
- [65] Andrej Karpathy. „The Unreasonable Effectiveness of Recurrent Neural Networks“. In: *Andrej Karpathy blog* (Mai 2015). URL: <http://karpathy.github.io/2015/05/21/rnn-effectiveness> (besucht am 19.06.2018) (ref. auf S. 18, 19).

- [66] Andrej Karpathy u. a. „Large-scale video classification with convolutional neural networks“. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, S. 1725–1732 (ref. auf S. 11, 17, 24, 27, 29, 33, 42, 43).
- [67] Ildoo Kim. *Deep Pose Estimation implemented using Tensorflow with Custom Architectures for fast inference*. Version d884cd1. Juli 2018. URL: <https://github.com/ildoonet/tf-pose-estimation> (besucht am 02.07.2018) (ref. auf S. II, 9, 50, 51, 54, 55).
- [68] KiwiSecurity. *Parking Space Analyzer*. URL: <https://www.kiwisecurity.com/parking-space-analyzer/> (besucht am 21.07.2018) (ref. auf S. 4).
- [69] Orit Kliper-Gross, Tal Hassner und Lior Wolf. „The action similarity labeling challenge“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (2012), S. 615–621 (ref. auf S. 37).
- [70] Yu Kong, Yunde Jia und Yun Fu. „Learning human interaction by interactive phrases“. In: *European conference on computer vision*. Springer. 2012, S. 300–313 (ref. auf S. 37, 50).
- [71] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems* 25. Hrsg. von F. Pereira u. a. Curran Associates, Inc., 2012, S. 1097–1105 (ref. auf S. 11, 17, 21, 34).
- [72] Hildegard Kuehne u. a. „HMDB: a large video database for human motion recognition“. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, S. 2556–2563 (ref. auf S. 34, 37).
- [73] Ivan Laptev u. a. „Learning realistic human actions from movies“. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, S. 1–8 (ref. auf S. 37).
- [74] Roberto Leyva, Victor Sanchez und Chang-Tsun Li. „The LV dataset: A realistic surveillance video dataset for abnormal event detection“. In: *Biometrics and Forensics (IWBF), 2017 5th International Workshop on*. IEEE. 2017, S. 1–6 (ref. auf S. 35).
- [75] Wanqing Li, Zhengyou Zhang und Zicheng Liu. „Action recognition based on a bag of 3d points“. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE. 2010, S. 9–14 (ref. auf S. 37).
- [76] Yan-Ching Lin u. a. „Human action recognition and retrieval using sole depth information“. In: *Proceedings of the 20th ACM international conference on Multimedia*. ACM. 2012, S. 1053–1056 (ref. auf S. 38).
- [77] Richard Lippmann. „An introduction to computing with neural nets“. In: *IEEE Assp magazine* 4.2 (1987), S. 4–22 (ref. auf S. 11, 13–15).
- [78] Jingwen Liu, Jiebo Luo und Mubarak Shah. „Recognizing realistic actions from videos “in the wild”“. In: *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*. IEEE. 2009, S. 1996–2003 (ref. auf S. 37).
- [79] Joyce Liu und Wang Xiqing. *In Your Face: China's all-seeing state*. Dez. 2017. URL: <https://www.bbc.com/news/av/world-asia-china-42248056/> (besucht am 21.07.2018) (ref. auf S. 3, 4).
- [80] Jun Liu u. a. „Spatio-temporal lstm with trust gates for 3D human action recognition“. In: *European Conference on Computer Vision*. Springer. 2016, S. 816–833 (ref. auf S. 32).
- [81] Josh Loeb. „The face of future surveillance“. In: *Engineering & Technology* (Nov. 2017), S. 32–35. ISSN: 17509637 (ref. auf S. 3).

- [82] Cewu Lu, Jianping Shi und Jiaya Jia. „Abnormal event detection at 150 fps in matlab“. In: *Proceedings of the IEEE international conference on computer vision*. 2013, S. 2720–2727 (ref. auf S. 35).
- [83] Behrooz Mahasseni und Sinisa Todorovic. „Regularizing long short term memory with 3D human-skeleton sequences for action recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 3054–3062 (ref. auf S. 27, 29, 30).
- [84] Arun Mallya und Svetlana Lazebnik. „Learning models for actions and person-object interactions with transfer to question answering“. In: *European Conference on Computer Vision*. Springer. 2016, S. 414–428 (ref. auf S. 12, 27, 29).
- [85] Marcin Marszalek, Ivan Laptev und Cordelia Schmid. „Actions in context“. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, S. 2929–2936 (ref. auf S. 37).
- [86] Michael McCahill und Clive Norris. „CCTV in London“. In: *Report deliverable of UrbanEye project*. The Urbaneye Working Papers Series (2002) (ref. auf S. 3).
- [87] Alexander Melde und Stefan Schneider. *Verkehrszeichenerkennung*. Version 77e029f. Juni 2018. URL: <https://github.com/AlexanderMelde/Verkehrszeichenerkennung> (besucht am 14. 07. 2018) (ref. auf S. 13, 46, 66, 67).
- [88] Thomas B Moeslund und Erik Granum. „A survey of computer vision-based human motion capture“. In: *Computer vision and image understanding* 81.3 (2001), S. 231–268 (ref. auf S. 9).
- [89] Mathew Monfort u. a. „Moments in Time Dataset: one million videos for event understanding“. In: *arXiv preprint arXiv:1801.03150* (2018) (ref. auf S. 37).
- [90] Kevin Murphy. *Machine learning: a probabilistic perspective*. Cambridge: MIT Press, 2012 (ref. auf S. 11).
- [91] National Police Chiefs' Council. *Automatic Number Plate Recognition (ANPR)*. 2016. URL: <http://www.npcc.police.uk/FreedomofInformation/ANPR.aspx> (besucht am 21. 07. 2018) (ref. auf S. 4).
- [92] Anh T Nghiem u. a. „ETISEO, performance evaluation for video surveillance systems“. In: *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*. IEEE. 2007, S. 476–481 (ref. auf S. 37).
- [93] Juan Carlos Niebles, Chih-Wei Chen und Li Fei-Fei. „Modeling temporal structure of decomposable motion segments for activity classification“. In: *European conference on computer vision*. Springer. 2010, S. 392–405 (ref. auf S. 37).
- [94] Ferda Ofli u. a. „Berkeley MHAD: A comprehensive multimodal human action database“. In: *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*. IEEE. 2013, S. 53–60 (ref. auf S. 38).
- [95] Sangmin Oh u. a. „A large-scale benchmark dataset for event recognition in surveillance video“. In: *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*. IEEE. 2011, S. 3153–3160 (ref. auf S. 35).
- [96] Christopher Olah. „Understanding LSTM Networks“. In: *colah's blog* (Aug. 2015). URL: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> (besucht am 19. 06. 2018) (ref. auf S. 19).
- [97] Omar Oreifej und Zicheng Liu. „Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, S. 716–723 (ref. auf S. 37).

- [98] George Papandreou u. a. „PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model“. In: *arXiv preprint arXiv:1803.08225* (2018) (ref. auf S. 9, 10).
- [99] Jigar Patel u. a. „Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques“. In: *Expert Systems with Applications* 42.1 (2015), S. 259–268 (ref. auf S. 11).
- [100] Alonso Patron-Perez u. a. „High Five: Recognising human interactions in TV shows.“ In: *British Machine Vision Conference (BMVC). Proceedings of the*. Bd. 1. 2010, S. 2 (ref. auf S. 37, 50).
- [101] Hossein Rahmani und Ajmal Mian. „3D action recognition from novel viewpoints“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 1506–1515 (ref. auf S. 7).
- [102] Tariq Rashid. *Neuronale Netze selbst programmieren: Ein verständlicher Einstieg mit Python*. O'Reilly, 2017 (ref. auf S. 11, 13–15).
- [103] Kishore K Reddy und Mubarak Shah. „Recognizing 50 human action categories of web videos“. In: *Machine Vision and Applications* 24.5 (2013), S. 971–981 (ref. auf S. 37).
- [104] Maxwell Ridgeway. *Tow Pound Ped Crossing*. Mai 2018. URL: <https://unplash.com/photos/jgmeGFg-3d4> (besucht am 27.07.2018) (ref. auf S. II, 48).
- [105] Lucy P. Roberts. *History of Video Surveillance and CCTV*. URL: <http://www.wecosurveillance.com/cctvhistory> (besucht am 06.08.2018) (ref. auf S. 3).
- [106] Mikel D Rodriguez, Javed Ahmed und Mubarak Shah. „Action MACH: a spatio-temporal maximum average correlation height filter for action recognition“. In: *Computer Vision and Pattern Recognition*, 2008. *CVPR 2008. IEEE Conference on*. IEEE. 2008, S. 1–8 (ref. auf S. 36, 37).
- [107] Marcus Rohrbach u. a. „A database for fine grained activity detection of cooking activities“. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, S. 1194–1201 (ref. auf S. 34, 38).
- [108] Michael S Ryoo und JK Aggarwal. „UT-interaction dataset, ICPR contest on semantic description of human activities (SDHA)“. In: *IEEE International Conference on Pattern Recognition Workshops*. Bd. 2. 2010, S. 4 (ref. auf S. 37, 50).
- [109] Christian Schuldt, Ivan Laptev und Barbara Caputo. „Recognizing human actions: a local SVM approach“. In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Bd. 3. IEEE. 2004, S. 32–36 (ref. auf S. 9, 27, 34–37, 50, 53, 55, 59, 62, 64, 65, 68, 71).
- [110] Laura Sevilla-Lara u. a. „Optical flow with semantic segmentation and localized layers“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 3889–3898 (ref. auf S. 23).
- [111] Amir Shahroudy u. a. „NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis“. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juni 2016 (ref. auf S. 34, 35, 37).
- [112] Shayan Shams u. a. „Evaluation of deep learning frameworks over different HPC architectures“. In: *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE. 2017, S. 1389–1396 (ref. auf S. 67).
- [113] Shikhar Shrestha. „Machine Learning for Human Activity Recognition from Video“. In: (Dez. 2016). URL: <https://pdfs.semanticscholar.org/e11a/5d5aa25e1bb39925f42cb1c118fc160752af.pdf> (besucht

- am 19. 06. 2018) (ref. auf S. 27, 29, 34, 39).
- [114] L. Sigal, A. Balan und M. J. Black. „HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion“. In: *International Journal of Computer Vision* 87.1 (März 2010), S. 4–27 (ref. auf S. 38).
- [115] Gunnar A Sigurdsson. *Charades Webcam*. Version b6482f9. Apr. 2018. URL: <https://github.com/gsig/charades-webcam> (besucht am 21. 08. 2018) (ref. auf S. 48).
- [116] Gunnar A Sigurdsson, Olga Russakovsky und Abhinav Gupta. „What Actions are Needed for Understanding Human Actions in Videos?“ In: *arXiv preprint arXiv:1708.02696* (2017) (ref. auf S. 20, 34, 39, 40).
- [117] Gunnar A Sigurdsson u. a. „Hollywood in Homes: Crowdsourcing data collection for activity understanding“. In: *European Conference on Computer Vision*. Springer. 2016, S. 510–526 (ref. auf S. 34, 36, 37, 40, 48).
- [118] Karen Simonyan und Andrew Zisserman. „Two-stream convolutional networks for action recognition in videos“. In: *Advances in neural information processing systems*. 2014, S. 568–576 (ref. auf S. 22, 24–27, 29).
- [119] Bharat Singh u. a. „A multi-stream bi-directional recurrent neural network for fine-grained action detection“. In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE. 2016, S. 1961–1970 (ref. auf S. 27, 29).
- [120] Evgeny A Smirnov, Denis M Timoshenko und Serge N Andrianov. „Comparison of regularization methods for imagenet classification with deep convolutional neural networks“. In: *Aasri Procedia* 6 (2014), S. 89–94 (ref. auf S. 17).
- [121] Khurram Soomro und Amir R Zamir. „Action recognition in realistic sports videos“. In: *Computer vision in sports*. Springer, 2014, S. 181–208 (ref. auf S. 36, 37).
- [122] Khurram Soomro, Amir Roshan Zamir und Mubarak Shah. „UCF101: A dataset of 101 human actions classes from videos in the wild“. In: *arXiv preprint arXiv:1212.0402* (2012) (ref. auf S. 34, 37, 42).
- [123] Nitish Srivastava, Elman Mansimov und Ruslan Salakhudinov. „Unsupervised learning of video representations using LSTMs“. In: *International conference on machine learning*. 2015, S. 843–852 (ref. auf S. 11, 19, 25, 27, 29).
- [124] Stadt Biberach. *Marktplatzcam*. URL: <http://www.marktplatzcam.mybiberach.de/> (besucht am 30. 07. 2018) (ref. auf S. 35).
- [125] Lin Sun u. a. „Human action recognition using factorized spatio-temporal convolutional networks“. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, S. 4597–4605 (ref. auf S. 24).
- [126] Jaeyong Sung u. a. „Human Activity Detection from RGBD Images.“ In: *plan, activity, and intent recognition* 64 (2011) (ref. auf S. 37).
- [127] Jaeyong Sung u. a. „Unstructured human activity detection from rgbd images“. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, S. 842–849 (ref. auf S. 37).
- [128] Gian Antonio Susto u. a. „Machine learning for predictive maintenance: A multiple classifier approach“. In: *IEEE Transactions on Industrial Informatics* 11.3 (2015), S. 812–820 (ref. auf S. 11).
- [129] Ilya Sutskever, Oriol Vinyals und Quoc V Le. „Sequence to sequence learning with neural networks“. In: *Advances in neural information processing systems*. 2014, 3 104–3112 (ref. auf S. 19).

- [130] Christian Szegedy u. a. „Rethinking the inception architecture for computer vision“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 2818–2826 (ref. auf S. 51).
- [131] Islam ATF Taj-Eddin u. a. „A new compression technique for surveillance videos: Evaluation using new dataset“. In: *Digital Information and Communication Technology and its Applications (DICTAP), 2016 Sixth International Conference on*. IEEE. 2016, S. 159–164 (ref. auf S. 35).
- [132] Moritz Tenorth, Jan Bandouch und Michael Beetz. „The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition“. In: *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV2009*. 2009 (ref. auf S. 38).
- [133] The TensorFlow Authors. „How to Retrain an Image Classifier for New Categories“. In: *TensorFlow Hub* (Aug. 2018). URL: https://www.tensorflow.org/hub/tutorials/image_retraining (besucht am 15.08.2018) (ref. auf S. 51).
- [134] The TensorFlow Authors. *Image Retraining Script retrain.py for use with TensorFlow Hub*. Version 52d5066. Apr. 2018. URL: https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py (besucht am 13.07.2018) (ref. auf S. 51).
- [135] David Thirde, Longzhen Li und F Ferryman. „Overview of the PETS2006 challenge“. In: *Proc. 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006)*. 2006, S. 47–50 (ref. auf S. 35).
- [136] Denis Tome, Christopher Russell und Lourdes Agapito. „Lifting from the deep: Convolutional 3d pose estimation from a single image“. In: *CVPR 2017 Proceedings* (2017), S. 2500–2509 (ref. auf S. 9).
- [137] Alexander Toshev und Christian Szegedy. „DeepPose: Human pose estimation via deep neural networks“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, S. 1653–1660 (ref. auf S. 9).
- [138] Du Tran und Alexander Sorokin. „Human activity recognition with metric learning“. In: *European conference on computer vision*. Springer. 2008, S. 548–561 (ref. auf S. 37).
- [139] Du Tran u. a. „Learning spatiotemporal features with 3D convolutional networks“. In: *Proceedings of the IEEE international conference on computer vision*. 2015, 4489–4497 (ref. auf S. 17, 24, 26, 27, 29).
- [140] Du Tran u. a. „Convnet architecture search for spatiotemporal feature learning“. In: *arXiv preprint arXiv:1708.05038* (2017) (ref. auf S. 42, 43).
- [141] Twenty Billion Neurons GmbH. *The 20BN-Something-Something* dataset. URL: <https://20bn.com/datasets/something-something> (besucht am 06.08.2018) (ref. auf S. 37).
- [142] GÜl Varol, Ivan Laptev und Cordelia Schmid. „Long-term temporal convolutions for action recognition“. In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2018), S. 1510–1517 (ref. auf S. 26, 27, 29, 30).
- [143] Subhashini Venugopalan u. a. „Sequence to sequence-video to text“. In: *Proceedings of the IEEE international conference on computer vision*. 2015, S. 4534–4542 (ref. auf S. 25, 27, 29).
- [144] Chirag Vora, Biswajit Kumar Yadav und Subhabrata Sengupta. „Comprehensive survey on shot boundary detection techniques“. In: *International Journal of Computer Applications* 140.11 (2016) (ref. auf S. 42).

- [145] Gregory K Wallace. „The JPEG still picture compression standard“. In: *Communications of the ACM* 34.4 (1991), S. 30–44 (ref. auf S. 6).
- [146] Heng Wang und Cordelia Schmid. „Action recognition with improved trajectories“. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, 3551–3558 (ref. auf S. 22–24, 34).
- [147] Jiang Wang u. a. „Mining actionlet ensemble for action recognition with depth cameras“. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, S. 1290–1297 (ref. auf S. 37).
- [148] Jiang Wang u. a. „Robust 3d action recognition with random occupancy patterns“. In: *Computer vision-ECCV 2012*. Springer, 2012, S. 872–885 (ref. auf S. 37).
- [149] Limin Wang u. a. „Temporal segment networks: Towards good practices for deep action recognition“. In: *European Conference on Computer Vision*. Springer. 2016, S. 20–36 (ref. auf S. 20, 25–27, 29, 34, 43, 51).
- [150] Daniel Weinland, Remi Ronfard und Edmond Boyer. „Free viewpoint action recognition using motion history volumes“. In: *Computer vision and image understanding* 104.2-3 (2006), S. 249–257 (ref. auf S. 37).
- [151] Philippe Weinzaepfel, Zaid Harchaoui und Cordelia Schmid. „Learning to track for spatio-temporal action localization“. In: *Proceedings of the IEEE international conference on computer vision*. 2015, S. 3164–3172 (ref. auf S. 25, 27, 29).
- [152] Philippe Weinzaepfel, Xavier Martin und Cordelia Schmid. „Human Action Localization with Sparse Spatial Supervision“. In: *arXiv preprint arXiv:1605.05197* (2016) (ref. auf S. 37).
- [153] Christian Wolf u. a. „Evaluation of video activity localizations integrating quality and quantity measurements“. In: *Computer Vision and Image Understanding* 127 (2014), S. 14–30 (ref. auf S. 34, 36, 37).
- [154] Lu Xia, Chia-Chih Chen und Jake K Aggarwal. „View invariant human action recognition using histograms of 3D joints“. In: *Computer vision and pattern recognition workshops (CVPRW), 2012 IEEE computer society conference on*. IEEE. 2012, S. 20–27 (ref. auf S. 38).
- [155] Zhengyuan Yang u. a. „Action Recognition with Spatio-Temporal Visual Attention on Skeleton Image Sequences“. In: *arXiv preprint arXiv:1801.10304* (2018) (ref. auf S. 9, 27, 29, 30).
- [156] Serena Yeung u. a. „Every moment counts: Dense detailed labeling of actions in complex videos“. In: *International Journal of Computer Vision* 126.2-4 (2018), S. 375–389 (ref. auf S. 27, 29, 30).
- [157] Xiaoyi Yu u. a. „Privacy protecting visual processing for secure video surveillance“. In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE. 2008, S. 1672–1675 (ref. auf S. 8).
- [158] Joe Yue-Hei Ng u. a. „Beyond short snippets: Deep networks for video classification“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 4694–4702 (ref. auf S. 25–27, 29).
- [159] Kiwon Yun u. a. „Two-person interaction detection using body-pose features and multiple instance learning“. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE. 2012, S. 28–35 (ref. auf S. 38, 50).
- [160] Andrei Zaharescu und Richard Wildes. „Anomalous behaviour detection using spatiotemporal oriented energies, subset inclusion histogram comparison

and event-driven processing". In: *European Conference on Computer Vision*. Springer. 2010, S. 563–576 (ref. auf S. 35).

- [161] Bowen Zhang u. a. „Real-time action recognition with enhanced motion vector CNNs“. In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE. 2016, S. 2718–2726 (ref. auf S. 17, 23, 41).
- [162] Jing Zhang u. a. „A large scale rgb-d dataset for action recognition“. In: *International Workshop on Understanding Human Activities through 3D Sensors*. Springer. 2016, S. 101–114 (ref. auf S. 37).
- [163] Jing Zhang u. a. „RGB-D-based action recognition datasets: A survey“. In: *Pattern Recognition* 60 (2016), S. 86–105 (ref. auf S. 34, 36).
- [164] Hang Zhao u. a. „SLAC: A Sparsely Labeled Dataset for Action Classification and Localization“. In: *arXiv preprint arXiv:1712.09374* (2017) (ref. auf S. 37).
- [165] Wangjiang Zhu u. a. „A key volume mining deep framework for action recognition“. In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE. 2016, S. 1991–1999 (ref. auf S. 41).
- [166] Wentao Zhu u. a. „Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks.“ In: *AAAI*. Bd. 2. 2016, S. 8 (ref. auf S. 10, 27, 29, 30).
- [167] Yi Zhu u. a. „Hidden two-stream convolutional networks for action recognition“. In: *arXiv preprint arXiv:1704.00389* (2017) (ref. auf S. 22–25, 27, 29, 30, 40).
- [168] Mohammadreza Zolfaghari u. a. „Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection“. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE. 2017, S. 2923–2932 (ref. auf S. 27, 29, 30).

Acknowledgement

I would like to thank Dr. Michael Sieck, Tarek Iraiki and Matthias Stumpp for helpful comments and discussion. Portions of the research in this paper used the NTU RGB+D Action Recognition Dataset made available by the ROSE Lab at the Nanyang Technological University, Singapore.