Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Fakultät für Informatik und Wirtschaftsinformatik

Master-Thesis

Name:	Alexander Melde
Thema:	Evaluierung der Nutzung synthetischer Daten zur Verbesserung von Modellen für Maschinelles Lernen zur Erkennung menschlicher Handlungen
Arbeitsplatz:	EnBW Energie Baden-Württemberg AG, Karlsruhe
Referent:	Prof. Dr.-Ing. Laubenheimer
Korreferent:	Prof. Dr. Zirpins
Abgabetermin:	30.09.2020

Karlsruhe, 01.04.2020

Der Vorsitzende des Prüfungsausschusses

Prof. Dr. Heiko Körner

Eidesstattliche Erklärung

Ich, Alexander Melde, versichere hiermit, dass ich die vorliegende Arbeit mit dem Titel „Evaluierung der Nutzung synthetischer Daten zur Verbesserung von Modellen für Maschinelles Lernen zur Erkennung menschlicher Handlungen“ 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

Öffentliche Version

Die für das Prüfungsverfahren eingereichte Version dieser Arbeit enthält im Anhang vertrauliche Daten der EnBW Energie Baden-Württemberg AG.

Für diese öffentliche Version der Thesis wurden diese Anhänge entfernt.

Abstract

Human actions can be detected in videos using artificial intelligence (AI). To use these systems in public places, sample videos of these locations are required. Due to the rarity of individual actions and the relevance to data protection law in these places, there is a lack of suitable real videos.

In this thesis a 3D simulation is developed, which simulates actions on public places and can export these scenes as labeled videos. These videos will be added to real data in order to test whether the accuracy of the AI model can be increased by adding synthetic videos.

Furthermore, possible sources for real videos that can be used as a reference are researched in the context of this thesis. It turned out that no single dataset fulfills all criteria. Therefore, the videos were taken from different sources and a new dataset was aggregated and published in the course of this work.

The training of the action recognition model was tested in four different experiments. Unfortunately, the addition of synthetic data did not lead to a significant improvement in action recognition accuracy, but a lot of highly relevant information for a new test could be obtained.

The most important findings of this work are that in order to make reliable statements regarding the improvement of the models, many real videos are necessary and the scenes of the synthetic dataset should be very close to those of the real videos. In this thesis, the synthetic scenes fit closely to the use case, which unfortunately is not the case for the real videos. Finding more suitable real videos turns out to be very demanding, which is why the hypothesis should better be tested with another use case that provides more real videos. With more available computing capacity, better generalizing training methods could be used.

Other topics covered in this thesis are the creation of a comprehensive concept for the implementation of the above mentioned components as well as an explanation of the basics of the topics covered. Finally, instructions for the use of the developed components are given and possibilities for extensions are shown, a recommendation for action is given and ethical questions regarding the topic are discussed.

Keywords

synthetic data, human action recognition, 3D simulation, CCTV, privacy, anonymizing, anonymization

Kurzbeschreibung

Mithilfe künstlicher Intelligenz (KI) können menschliche Handlungen in Videos detektiert werden. Um diese Systeme auf öffentlichen Plätzen einzusetzen werden Beispielvideos dieser Orte benötigt. Aufgrund der Seltenheit einzelner Handlungen und der datenschutzrechtlichen Relevanz dieser Orte gibt es einen Mangel geeigneter realen Videos.

In dieser Arbeit wird eine 3D-Simulation entwickelt, die Handlungen auf öffentlichen Plätzen simuliert und als beschriftete Videos exportieren kann. Diese Videos werden zu den realen Daten hinzugegeben, um zu prüfen, ob durch Hinzunahme der synthetischen Videos die Genauigkeit des KI-Modells gesteigert werden kann.

Mögliche Quellen für als Referenz verwendbare Realvideos werden ebenfalls im Rahmen dieser Arbeit recherchiert. Hierbei hat sich herausgestellt, dass kein Datensatz alle Kriterien erfüllt, weshalb die Videos aus verschiedenen Quellen stammen und im Rahmen dieser Arbeit ein eigener Datensatz aggregiert und veröffentlicht wurde.

Das Trainieren der Handlungserkennung wurde in vier verschiedenen Experimenten probiert. Die Hinzunahme synthetischer Daten konnte dabei zu keiner deutlichen Verbesserung der Handlungserkennung führen, es konnten aber zahlreiche Informationen erlangt werden, die für einen erneuten Test von großer Relevanz sind.

Die wichtigsten Erkenntnisse dieser Arbeit sind, dass zum Treffen verlässlicher Aussagen hinsichtlich der Verbesserung der Modelle viele reale Videos notwendig und die Szenen des synthetischen Datensatzes sehr nah an denen der Realvideos sein sollten. In dieser Arbeit passen die synthetischen Szenen nah zum Anwendungsfall, was für die realen Videos leider nicht zutrifft. Das Auffinden passenderer Realvideos erweist sich als sehr anspruchsvoll, weshalb die These besser mit einem anderen Anwendungsfall, der mehr reale Videos bereitstellt, getestet werden sollte. Mit mehr zur Verfügung stehenden Rechenkapazität können zudem besser generalisierende Trainingsmethoden eingesetzt werden.

Weitere im Rahmen dieser Arbeit behandelten Themen sind die Erstellung eines umfassenden Konzepts zur Umsetzung der oben genannten Komponenten sowie die Erklärung von Grundlagen zu den behandelten Themen. Abschließend werden zudem Anleitungen zur Benutzung der Komponenten gegeben sowie Erweiterungsmöglichkeiten aufgezeigt, eine Handlungsempfehlung gegeben und ethische Fragestellungen hinsichtlich des Themas diskutiert.

Stichwörter

Synthetische Daten, Menschliche Handlungserkennung, 3D-Simulation, Unity, Videoüberwachung, Datenschutz, Anonymisierung

Inhaltsverzeichnis

Abkürzungsverzeichnis

2D Zweidimensional . . . 6
3D Dreidimensional . . . 6
AGI *Artificial General Intelligence* . . . 10
AI *Artificial Intelligence* . . . 10
ANN *Artificial Neural Network* . . . 11
AVA *Atomic Visual Actions* . . . 32
Abb. Abbildung . . . 1
Assets 3D-Objekte, Texturen, Charaktere, Animationen und Code . . . 38
CNN *Convolutional Neural Network* . . . 22
CSV *Comma-Separated Values* . . . 31
CV *Computer Vision* . . . 6
DL *Deep Learning* . . . 11
DNN *Deep Neural Network* . . . 11
ELU *Exponential Linear Unit* . . . 21
EMG Elektromyografie . . . 32
EnBW EnBW Energie Baden-Württemberg AG . . . 3
GAN *Generative Adversial Network* . . . 24
GB Gigabyte . . . 82
HMDB *Human Motion Database* . . . 31
IK *Inverse Kinematics* . . . 65
IoU *Intersection over Union* . . . 7
JSON *JavaScript Object Notation* . . . 7
KI Künstliche Intelligenz . . . 2
KNN Künstliches Neuronales Netz . . . 11
LSTM *Long Short-Term Memory* . . . 23
MB Megabyte . . . 83
MLP *Multi Layer Perceptron* . . . 11
ML Maschinelles Lernen . . . 2
NLP *Natural Language Processing* . . . 27
NLU *Natural Language Understanding* . . . 27
OCR *Optical Character Recognition* . . . 25
PReLU *Parametric ReLU* . . . 21
RGB Rot, Grün und Blau . . . 6
RNN *Recurrent Neural Network* . . . 23
ReLU *Rectified Linear Unit* . . . 21
S-SPHAR *Synthetic SPHAR* . . . 55
SGD *Stochastic Gradient Descent* . . . 19
SPHAR *Surveillance-Perspective Human Action Recognition* . . . 49
SVM *Support Vector Machine* . . . 13
tanh *Tangens Hyperbolicus* . . . 20

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1. Einleitung

Viele Personen haben den Wunsch nach mehr Sicherheit auf öffentlichen Plätzen. Einer Erhebung des Bundeskriminalamts zufolge stieg die Kriminalitätsfurcht zwischen 2012 und 2017 [24, S. 46], und so meiden beispielsweise 38,9% der 2017 befragten Frauen und 20,9% der befragten Männer bestimmte Straßen, Plätzen und Parks häufig oder immer. Weitere Personen gaben an, es zu vermeiden, allein im Dunkeln unterwegs zu sein oder für ihre Sicherheit sogar Umwege in Kauf zu nehmen. [24, S. 59–59]

> *„Auch von den Frauen, die sich in ihrer Wohngegend sehr sicher fühlen, vermeiden es mehr als die Hälfte, im Dunkeln allein unterwegs zu sein, ein Fünftel sogar häufig oder immer. Dies unterstreicht, dass Einschränkungen der Bewegungsfreiheit ein fester Bestandteil des Alltagslebens vieler Frauen sind.“*
>
> Bundeskriminalamt / Birkel u. a. [24, S. 60]

Um dieser Entwicklung entgegenzuwirken, suchen Städte und Kommunen nun nach Methoden, ihre Straßen, Plätze und Parks sicherer und freundlicher zu machen. In dieser Arbeit wird ein Ansatz erarbeitet, der sie dabei unterstützen soll. Mithilfe einer Analyse von Videobildern sollen Gefahren im öffentlichen Raum erkannt werden und bei Bedarf Hilfe in Form von Polizei oder Ordnungsamt herbeigerufen werden.

Dies könnte der Polizei dabei helfen, schneller vor Ort zu sein. Laut der Studie begründeteten 40% der mit dem Polizeikontakt unzufriedenen Opfer von Körperverletzungen ihre Unzufriedenheit mit dem späten Eintreffen der Polizei [24, S. 70].

Aus einer Umfrage des Meinungsforschungsinstituts IfD Allensbach geht hervor, dass 58% der befragten Bürger Sicherheit sogar wichtiger als Wohlstand ist - nur jeder vierte Bürger wünscht sich Wohlstand statt Sicherheit [76]. Auch sprechen sich zahlreiche Bürger als Konsequenz von Anschlägen oder sexuellen Übergriffen für eine Ausweitung von Videoüberwachung aus [77; 192].

Dennoch sollte die Wahrung der Privatsphäre nicht unberücksichtigt werden. Zur Gewährleistung des Datenschutzes soll bei der entwickelten Lösung keine Person Klarbilder oder Merkmale angezeigt bekommen, anhand derer eine andere Person ohne weitere Zusatzinformationen identifiziert werden könnte.

Eine rein textuelle Beschreibung von Gefahrensituationen, beispielsweise eine Kombination aus dem Typ der Gefahr (zum Beispiel Körperverletzung, Unfall oder Belästigung) und der Anzahl der beteiligten Personen würde bereits genügen, um eine Polizei-Streife vor Ort zu alarmieren und auf die Gefahr aufmerksam zu machen. Durch diese gezielte Einsatzplanung kann die Effektivität und das Abdeckungsgebiet von vor-Ort Streifen erhöht werden, ohne den personellen Aufwands zu vergrößern. Eine zusätzliche anonymisierte Darstellung des Kamerabilds bietet der Sicherheitsleitstelle die Möglichkeit, ungewöhnliche Situationen anhand der visuellen Konturen einzuschätzen und den Datenschutz im Vergleich zu herkömmlichen Videoüberwachungslösungen zu erhöhen.

Ein Beispiel, wie eine solche Software aussehen könnte, wird zur Veranschaulichung in Abbildung (Abb.) 1.1 gezeigt.

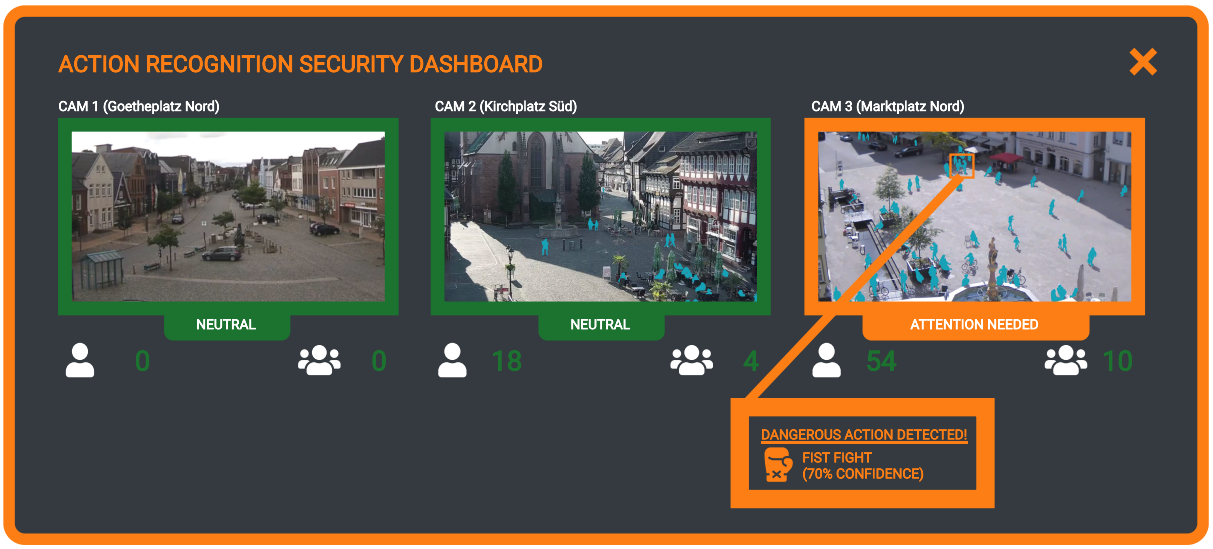


Abb. 1.1.: Prototyp einer Handlungserkennung im Anwendungsfall. Die Kamera-Bilder stammen aus öffentlichen Livestreams [83; 72; 168] und wurden für die Visualisierung der Anonymisierung digital nachbearbeitet.

Für die textuelle Beschreibung von Handlungen wird ein Algorithmus benötigt, der erkennen kann, welche Handlungen in einem Video-Ausschnitt ausgeführt werden. Eine solche Handlungserkennung kann mithilfe von Algorithmen für Maschinelles Lernen (ML) realisiert werden, eine Untereinheit des Forschungsbereichs Künstliche Intelligenz (KI). Diese Algorithmen lernen die Unterscheidung verschiedener Klassen von Handlungen anhand einer Vielzahl von Beispielvideos.

Eine Aufnahme realer Videos ist aufgrund von Datenschutzgesetzen und der Seltenheit einzelner Handlungen nur unter großem Aufwand möglich, beispielsweise mit Schauspielern. In dieser Arbeit soll daher überprüft werden, ob künstlich erzeugte (synthetische) Daten ebenfalls zum Lernen der Algorithmen genutzt werden können.

1.1. Umfeld der Arbeit

Bei dieser *Master-Thesis* handelt es sich um eine Abschlussarbeit des Masterstudiengangs Informatik an der Hochschule für Technik und Wirtschaft in Karlsruhe.

Der Autor dieser Arbeit, Alexander Melde, hat die Studiengangs-Vertiefung „Maschinelles Lernen“ gewählt und das Thema dieser Arbeit auf eigenen Wunsch der betreuenden Professorin sowie dem kooperierenden Unternehmen zur Prüfung vorgeschlagen. Er forscht seit 2018 an dem Thema Handlungserkennung. In seiner Bachelorarbeit hat er gezeigt, dass bereits durch die reine Auswertung von Körperhaltungen im zeitlichen Verlauf menschliche Handlungen unterschieden werden können [124].

Die betreuende Professorin Dr.-Ing. Astrid Laubenheimer ist Dozentin für Analysis, maschinelles Lernen, *Computer Vision* und Optimierung an der Hochschule für Technik und Wirtschaft in Karlsruhe. Zu ihren Forschungsgebieten zählen neben Maschinellem Lernen und *Computer Vision* auch Datenmodellierung und Industrie 4.0 [79]. Ihre Veröffentlichungen behandeln unter anderem automatisierte Überwachungssysteme [20] und Personenerkennung [191].

Diese Arbeit entstand in Kooperation mit der EnBW Energie Baden-Württemberg AG (EnBW). Die EnBW ist ein deutsches Energieunternehmen mit rund 5,5 Millionen Kunden [52, S. 39]. Sie ist überwiegend tätig in den Bereichen Vertrieb, Netze, erneuerbare Energien, Erzeugung und Handel [52, S. 32]. Die Kernkompetenz der EnBW besteht dabei aus dem sicheren und zuverlässigen Betrieb sowie dem Management kritischer Infrastruktur im Bereich Energie [52, S. 42]. Eine wichtige Strategie der EnBW für die nächsten Jahre ist der Aufbau eines umfangreichen Portfolios im Bereich Stadt- oder Quartiersentwicklung [52, S. 3] durch das Anbieten von Dienstleistungen für Kommunen mit Fokus auf Infrastruktur [52, S. 43]. Die Forschung und Entwicklung dieser Arbeit kann die Entwicklung und Weiterentwicklung aktueller und zukünftiger Produkte und Dienstleistungen der EnBW beeinflussen.

1.2. These

Die Genauigkeit eines Modells zur Erkennung menschlicher Handlungen kann durch die Hinzunahme von synthetisch erzeugten Daten verbessert werden.

1.3. Ziel der Arbeit

Ziel dieser Arbeit ist die Erarbeitung und Umsetzung eines Konzepts zur Evaluierung der Genauigkeitssteigerung eines Algorithmus zur Handlungserkennung bei Verwendung zusätzlicher synthetischer Daten als Beispielvideos.

1.4. Vorgehensweise

Zur Erarbeitung des Ziels dieser Arbeit werden zunächst einige Grundlagen erklärt, die für ein Verständnis von digitaler Bildverarbeitung, künstlicher Intelligenz, synthetischer Datenerzeugung und dem Erkennen von Handlungen notwendig sind.

Anschließend wird ein Konzept erarbeitet, dessen Umsetzung zur Evaluation der These führen soll. Hierzu werden verschiedene Methoden zur Datengenerierung sowie Datensätze und Algorithmen zur Handlungserkennung untersucht und miteinander verglichen.

Die Umsetzung des Konzepts wird in den drei darauffolgenden Kapiteln dokumentiert. Hierfür wird zunächst die Entwicklung der Simulation dokumentiert und die Generierung von Videos mit dieser beschrieben.

Anschließend wird die Implementierung der Handlungserkennung vorgestellt und der zentrale Test zur Evaluation der Verbesserung des Modells durch synthetische Daten beschrieben, wobei auf den Versuchsaufbaus, den Verlauf und die Ergebnisse eingegangen wird.

Im letzten Kapitel des Hauptteils werden Möglichkeiten zur Integration der Simulation und Handlungserkennung in Geschäftsprozesse geprüft.

Abschließend werden die gewonnenen Erkenntnisse und Ergebnisse der Experimente erneut kritisch geprüft und zusammenfassend reflektiert sowie ein Ausblick auf mögliche Erweiterungsmöglichkeiten des Projekts gegeben und ein Fazit gezogen.

Abb. 1.2.: Visualisierung der Vorgehensweise. Nach den Grundlagen wird in Unterabschnitten das Konzept erarbeitet und anschließend in drei Kapiteln umgesetzt. Darauf folgend werden einige Anleitungen zur Bedienung der Simulation und der Handlungserkennung gegeben. Abschließend werden Erweiterungsmöglichkeiten aufgezeigt und ein rückblickendes Fazit formuliert.

Diese Vorgehensweise wird zur Veranschaulichung in Abb. 1.2 als Diagramm gezeigt.

Im Anhang dieser Arbeit werden im Text referenzierte Quelltext-Dokumente aufgeführt.

Als Konvention zur Formatierung dieser Arbeit wird eine *kursive* Schriftart verwendet, wenn fremdsprachige Begriffe oder Block-Zitate genutzt werden und eine `monospace` Schriftart verwendet, wenn Programmcode, Dateinamen oder Web-Links dargestellt werden.

2. Grundlagen

Diese Arbeit erhebt den Anspruch, auch von Personen ohne tiefe Vorkenntnisse in den behandelten Bereichen grundlegend verstanden werden zu können. Um die teilweise sehr komplexen Themen dieser Arbeit zu verstehen, werden in diesem Kapitel die passenden Grundlagen erklärt.

Da in dieser Arbeit Videos ausgewertet werden sollen, wird zunächst ein Überblick über das Thema digitale Bildverarbeitung gegeben.

Anschließend folgt eine umfangreiche Beschreibung des Themenkomplexes Künstliche Intelligenz (KI). In diesem Zusammenhang werden auch die Begriffe maschinelles Lernen und neuronale Netze erläutert und voneinander abgegrenzt.

Zur Veranschaulichung der gezeigten Techniken, und um die über diese Arbeit hinausgehende Bedeutung von KI zu verdeutlichen, werden in diesem Zusammenhang auch einige beispielhafte Anwendungsfälle für KI-Software vorgestellt.

Wie bereits einleitend erwähnt, sollen in dieser Arbeit künstlich erzeugte „synthetische“ Daten genutzt werden, um ein Programm zur Erkennung menschlicher Handlungen in Videos zu verbessern. Im vorletzten Abschnitt des Grundlagen-Kapitels werden daher Vorteile von der Verwendung synthetischer Daten sowie mögliche Methoden zur Generierung dieser vorgestellt.

Abschließend werden einige Grundlagen rund um das zentrale Thema dieser Arbeit, die Handlungserkennung, beschrieben. Hierbei werden nicht nur grundlegende Methoden vorgestellt, sondern auch aktuelle Herausforderungen sowie der Stand der Forschung aufgezeigt.

2.1. Digitale Bildverarbeitung

Damit Computer Bilder und Videos verarbeiten können, müssen sie diese nicht nur in „Bits und Bytes“ abspeichern können, sondern auch verstehen.

Dieses „maschinelle Sehen“, auch bekannt unter dem englischen Begriff *Computer Vision* (*CV*), ist eine zentrale Komponente bei der Auswertung von Videodaten.

Um ein Verständnis zu ermöglichen, müssen nicht nur die Bilder als Datenstruktur verarbeitet werden, sondern es muss auch eine Möglichkeit geben, diese Bilder mit zusätzlichen Informationen zu beschriften, beispielsweise mit den Namen und Positionen der darin vorkommenden Objekten.

Der folgende Abschnitt erklärt zunächst, wie diese beiden Ziele erreicht werden können. Anschließend folgt eine kurze Einleitung in mögliche datenschutzkonforme Methoden zur Anonymisierung von Personen.

Digitale Repräsentation von Bildern und Videos Die meisten Fotos sind Rastergrafiken, die aus Pixeln bestehen [70, S. 75]. Jeder dieser Pixel hat eine Farbe, die sich aus Rot, Grün und Blau (RGB) - Anteilen zusammensetzt. Diese Unterteilung wird in Abb. 2.1 veranschaulicht [64, S. 251].

Der Farbwert eines Pixels kann also als Vektor mit drei Werten gesehen werden. Die Pixel selbst können als zweidimensionale (2D) Matrix betrachtet werden. Durch Kombination der 2D Matrix mit den Vektoren entsteht so eine dreidimensionale (3D) Matrix, die das Bild repräsentiert.

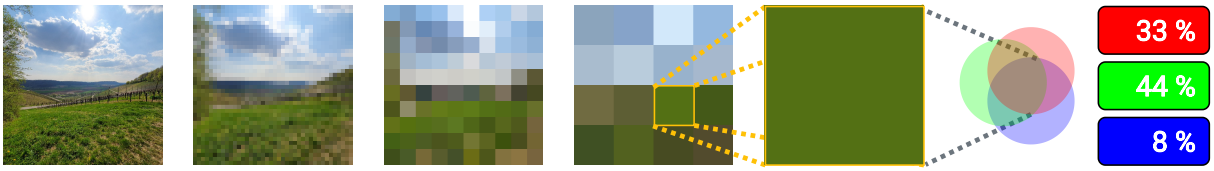


Abb. 2.1.: RGB Bild: Aufteilung der Rastergrafik in Pixel und Zerlegung des Pixel-Farbwerts in die drei Grundfarben Rot, Grün und Blau.

Neben Rastergrafiken gibt es auch sogenannte Vektorgrafiken, bei denen Programmcode-artige Befehle den Aufbau des Bildes definieren. Linien können dort beispielsweise durch Definiton eines Start- und Endpunkts gezeichnet werden, Boxen und andere geometrische Formen durch eine Position sowie eine Höhe und Breite. Vektorgrafiken haben den Vorteil, dass sie nie „pixelig“ werden, da sie stufenlos skaliert und für jede Auflösung gerendert, also aus den Rohdaten heraus erstellt, werden können.

Im Bereich des maschinellen Lernens wird überwiegend mit Rastergrafiken gearbeitet, weshalb im Folgenden der Begriff RGB-Bild immer diese bezeichnet.

Videos sind technisch gesehen eine Serie von Bildern. Diese sind hintereinander angeordnet, wodurch die oben erwähnte Matrixdarstellung zur Repräsentation von Videos einfach um eine weitere Dimension erweitert wird [162, S. 235].

Annotationen Für Methoden zur Handlungserkennung, die in den folgenden Abschnitten vorgestellt werden, ist es von großer Bedeutung, das neben den reinen Pixelinformationen zur Darstellung der Bilder und Videos auch Annotationen, also Informationen über den Inhalt dieser Bilder und Videos strukturiert gespeichert werden können.

Die einfachste Methode zur Klassifizierung von Bildern ist das Zuordnen von Klassen zu Bildern, beispielsweise mit einer Ordnerstruktur. Das Bild aus Abb. 2.2a wäre beispielsweise in einem Ordner namens *dog*, da es einen Hund zeigt.

Befinden sich mehrere relevante Objekte in einem Bild oder möchte man diese auch lokalisieren, so müssen weitere Informationen angegeben werden. Das bekannteste Verfahren zur Lokalisierung von Objekten in einem Bild ist das *Bounding Box* Verfahren. Hierbei wird zu jedem Objekt ein Rahmen an einer Pixel-Position (x, y) mit einer Breite w und Höhe h gezeichnet (siehe Abb. 2.2b) [64, S. 469].

(a) Originalbild (w=500, h=500)

(b) *Bounding Box* (x=35, y=201, w=402, h=256)

(c) Segmentierung in Form einer Maskierung

Abb. 2.2.: Möglichkeiten der Objekt-Annotation (nach [162, S. 22]).

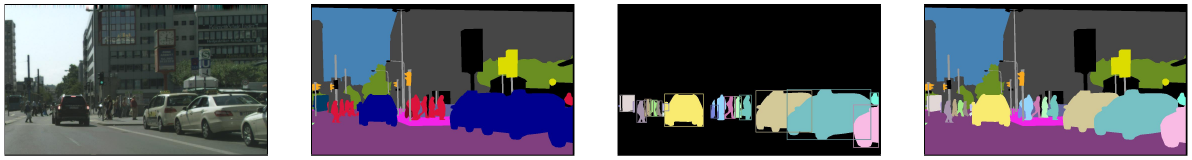
Eine *Bounding Box* Annotation des Hundes aus Abb. 2.2a könnte in *JavaScript Object Notation* (*JSON*) wie folgt repräsentiert werden:

```
{"objects": [{"class": "dog", "bbox": [35,201,402,256] }] }
```

Wird ein höherer Detailgrad gewünscht, so ist auch eine pixelweise Zuordnung von Klassen möglich. Dieses Verfahren wird Segmentierung genannt und in Abb. 2.2c beispielhaft in Form einer Bildmaske gezeigt. Um diese Maske numerisch darzustellen, können beispielsweise eine Konturlinie gezeichnet werden und alle Punkte (x_i, y_i) angegeben werden, die zum Zeichnen dieser Konturlinie miteinander verbunden werden müssen.

Es gibt drei verschiedene Arten von Segmentierung: Die semantische Segmentierung, die pro Pixel Objekt-Klassen unterscheidet (siehe Abb. 2.3b) [64, S. 251, 478; 93, S. 1], die Instanz-Segmentierung, die pro Objekt eine Maske und eine Klasse bestimmt (siehe Abb. 2.3c) [64, S. 251; 93, S. 1] und die panoptische Segmentierung, mit der sowohl Objekt-Klassen als auch Objekt-Instanzen auf Pixel-Ebene unterschieden werden können (siehe Abb. 2.3d) [93, S. 1].

Zur Messung der Ähnlichkeit zweier Segmentierungsmasken kann die sogenannte *Intersection over Union* (*IoU*) verwendet werden. Hierbei wird die Schnittmenge beider Masken durch die Vereinigung beider Masken geteilt, sodass ein Verhältnis der korrekt vorhergesagten Pixel im Vergleich zu der Gesamtzahl relevanter Pixel bestimmt werden kann. [64, S. 470–471]



(a) Originalbild (b) Semantische Segm. (c) Instanz-Segm. (d) Panoptische Segm.

Abb. 2.3.: Segmentierungsarten (aus [93, S. 1]).

Zur Beschriftung von Videos gibt es verschiedene Ansätze. In den meisten Fällen werden einfach Techniken zur Beschriftung von Bildern auf die Einzelbilder der Videos angewendet, beispielsweise also mehrere Segmentierungsmasken hintereinander.

Wenn über die Einzelbildinformationen hinweg auch Bewegungen erkannt werden sollen, so können andere Ansätze wie beispielsweise der optische Fluss (*Optical Flow*) verwendet werden. Dieser gibt für jeden Punkt im Bild deren Bewegungsvektoren an, sodass beispielsweise Bewegungsrichtungen vorhergesagt und Objekte über mehrere Einzelbilder hinweg verfolgt werden können.

Anonymisierung Um Personen im Bild unkenntlich zu machen, können Methoden zur Anonymisierung eingesetzt werden. Datenschutz ist hierbei nicht nur eine moralische Überzeugung, sondern je nach Anwendungsfall auch eine gesetzliche Vorgabe [102].

Eine Videoüberwachung ohne Personenbezug kann beispielsweise erreicht werden, indem eine ausreichend niedrige Auflösung verwendet wird. Die Norm DIN EN 62676-4 setzt Pixel und Meter in ein Verhältnis, um beispielsweise zu definieren, dass Personen erst unter einer Bildauflösung von 16 mm pro Pixel definitiv nicht mehr identifizierbar sind [101, S. 36]. Weitere in der Norm definierte Grenzwerte (siehe Abb. 2.4) geben an, ab wie vielen Pixeln eine Person zweifelsfrei zu identifizieren ist, ab wann diese Erkannt werden kann, wenn diese schon einmal gesehen wurde, ab wann das Vorkommen eines Menschen detektiert werden kann, und ab wann selbst Ansammlungen mehrerer Menschen nicht mehr als solche ersichtlich sind.

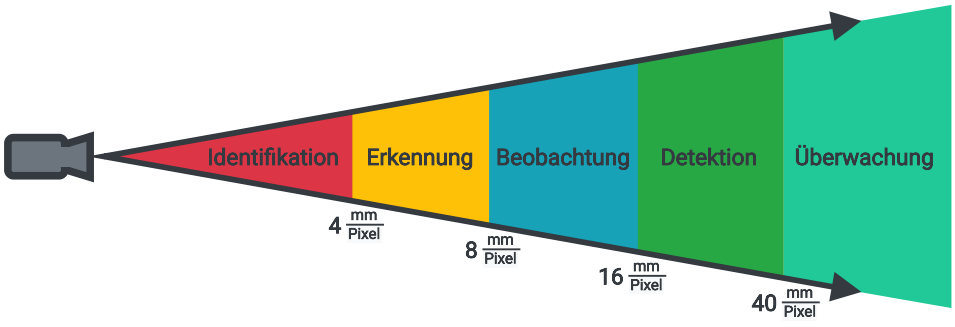
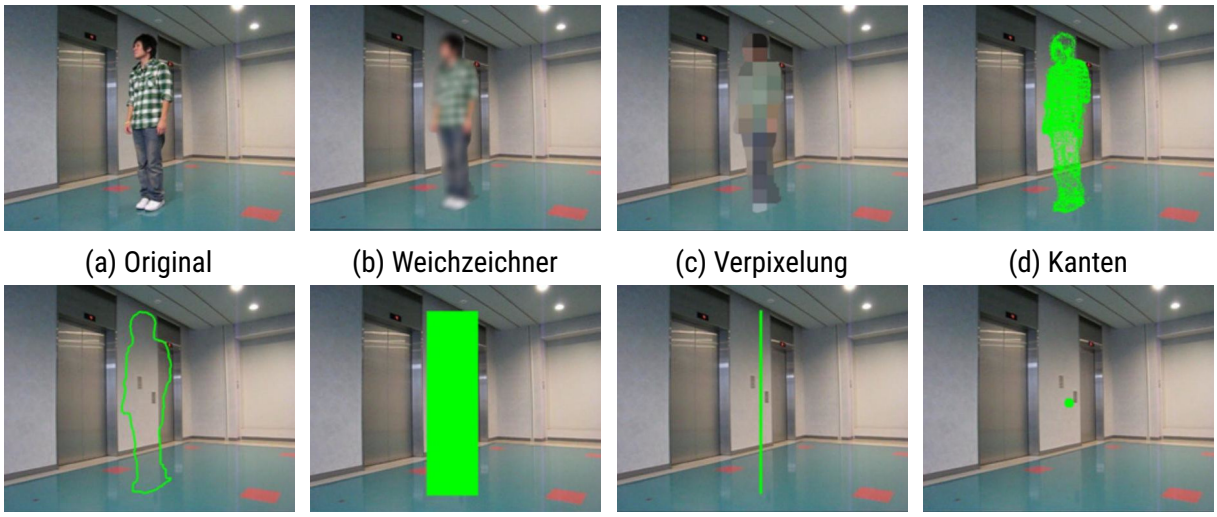


Abb. 2.4.: Anonymisierungszonen nach DIN EN 62676-4 (nach [101, S. 124; 81]).

Zur Verhinderung des Personenbezugs bei hoher Auflösung können Anonymisierungsmethoden angewandt werden. Diese können unterteilt werden in Methoden, die personenbeziehbare Bildbereiche unkenntlicher machen, wie beispielsweise die in Abb. 2.5b–c gezeigten Techniken, und Methoden, die Bewegungen, Objekte oder Personen erkennen und in abstrahierter Form vor einen statischen, nicht-personenbeziehbaren Hintergrund einzeichnen (siehe Abb. 2.5d–h) [202, S. 1674].

Anonymisierungs-Methoden, die Abstraktionen in das Bild einzeichnen, haben den Vorteil, dass bei Versagen der Methode keine personenbeziehbaren Teile des Original-Bilds sichtbar werden können.



(a) Original (b) Weichzeichner (c) Verpixelung (d) Kanten

(e) Rahmen (f) Box (g) Strich (h) Punkt

Abb. 2.5.: Verschiedene Methoden zur Anonymisierung von Personen [202, S. 1674].

Die in Abb. 2.5e gezeigte Anonymisierung basiert auf der im vorherigen Abschnitt vorgestellten semantischen Segmentierung. Hierzu wird zuerst ein Algorithmus zur Erkennung von Personen ausgeführt, um eine Maske zu erstellen, die alle Pixel die Personen enthalten markiert. Wird diese Maske über ein statisches Hintergrundbild der Szene ohne Personen gelegt, kann eine vollständige Anonymisierung wie im einleitend vorgestellten Prototyp (Abb. 1.1) erreicht werden.

Grundsätzlich gilt für diese Methoden: Je stärker die Anonymisierung, desto größer der mit ihr einhergehende 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.

Durch Einsatz einer Handlungserkennung könnten so auch die Handlungen der Personen beschrieben werden. In Kombination mit einer Anonymisierung wie in Abb. 2.5e können so zahlreiche Informationen über die beteiligten Personen gegeben werden, ohne deren Privatsphäre zu verletzen.

2.2. Künstliche Intelligenz (KI)

Der Begriff Künstliche Intelligenz (KI) ist heutzutage in aller Munde. Eine repräsentative Umfrage aus dem Jahr 2018 hat gezeigt, dass 74% der deutschen Bevölkerung den Begriff kennen und sich zutrauen, diesen zumindest grob erklären zu können. [29, S. 4]. In einer vergleichbaren Umfrage im Vorjahr gaben nur 33% der befragten Deutschen an, den Begriff erklären zu können [151, S. 4]. Es ist daher davon auszugehen, dass sich mittlerweile mehr Personen mit dem Thema KI auseinandersetzen.

Dennoch verstehen vermutlich nur die wenigsten, was sich genau dahinter verbirgt. Zahlreiche der im Detail verwendeten Techniken basieren auf mathematischen Methoden und Statistik, die erst in Studiengängen behandelt werden. Auch die Grundlagen zum Thema KI werden in Schulen nach Einschätzung der Schüler noch nicht ausreichend gut vermittelt [196].

Unter dem Begriff KI versteht man je nach Definition ein Programm, das fühlen, folgern, handeln und sich anpassen kann oder einfacher gesagt menschliches Verhalten nachbildet [11, S. 17]. Der Begriff KI kann synonym zu der englischen Bezeichnung *Artificial Intelligence* (*AI*) verwendet werden.

Grundsätzlich lassen sich zwei Typen von KI-Systemen unterscheiden. Expertensysteme versuchen die Entscheidungsfindung eines menschlichen Experten nachzubauen. Ein Experte kann eine Sache besonders gut, ist aber nicht in der Lage, alle anderen Expertengebiete auf diesem Niveau abzudecken.

Der zweite Typ *Artificial General Intelligence* (*AGI*) beschreibt die (hypothetische) Intelligenz einer Maschine, die in der Lage ist, alle Aufgaben zu verstehen oder zu erlernen, die ein menschliches Wesen verstehen und erlernen kann. Diese Art von KI ist die Grundlage für Bücher und Filme wie „2001: Odyssee im Weltraum“, in denen eine dystopische KI, die menschliche Intelligenz übertrifft und so zur Gefahr für die Menschen wird [98].

In dieser Arbeit werden ausschließlich Expertensysteme behandelt. In den folgenden Abschnitten werden die Techniken hinter KI sowie ihre Anwendungsfälle beschrieben.

2.2.1. Maschinelles Lernen

Die technische Grundlage für KI-Systeme ist maschinelles Lernen (ML), auch bekannt unter der englischen Bezeichnung *Machine Learning*. Hierunter versteht man Algorithmen, deren Leistung mit der von ihnen bekannten Anzahl Daten oder mit Erfahrung steigt. Sie haben die Fähigkeit, Dinge zu lernen, ohne explizit für diese programmiert worden zu sein. [66, S. 99]

Maschinelles Lernen kann in drei Typen untergliedert werden. Beim sogenannten überwachten Lernen (*supervised learning*) lernt der Algorithmus anhand von beschrifteten Beispieldaten (*Trainingsdaten*), die gewünschten Aufgaben wie Klassifikation oder Regression [66, S. 105; 64, S. 8]. Ein Beispiel für überwachtes Lernen ist ein E-Mail Spam-Filter, der anhand vieler als Spam klassifizierter Beispiel-E-Mails lernt, weitere E-Mails zu klassifizieren [64, S. 9].

Eine weitere typische Aufgabe ist das Vorhersagen eines numerischen Wertes wie beispielsweise dem Preis eines Autos anhand einer gegebenen Menge an Merkmalen wie Kilometerstand, Baujahr und Marke. Diese Aufgabe wird Regression genannt. Um das System zu trainieren, werden viele Beispiele von Autos benötigt, die jeweils sowohl ihre Merkmale als auch ihre Beschriftung (den Preis) enthalten [64, S. 9].

Beim unüberwachten Lernen (*unsupervised learning*) gruppiert der Algorithmus hingegen selbstständig verschiedene Cluster oder erkennt Anomalien ohne beschriftete Beispieldaten [64, S. 10; 66, S. 105].

Beim bestärkenden Lernen (*reinforcement learning*) lernt der Algorithmus iterativ in vielen Versuchen mit Hilfe von Belohnungen und Bestrafungen, was besonders gute Versuche waren (die die Belohnung maximieren) [15, S. 271].

Technisch basiert ML entweder auf einfachen statistischen Methoden oder künstlichen neuronalen Netzen KNN (englisch: *Artificial Neural Network* (*ANN*)). KNN sind dem menschlichen Gehirn nachempfundene mathematische Optimierungsverfahren. Neuronale Netze wie das mehrlagige Perzeptron (englisch: *Multi Layer Perceptron* (*MLP*)) bestehen aus mehreren Schichten, deren Aufbau in den Unterabschnitten näher beschrieben wird. Netze mit besonders vielen Schichten werden tiefe neuronale Netze (englisch: *Deep Neural Network* (*DNN*)) genannt und für das sogenannte *Deep Learning* (*DL*) verwendet. Netze mit wenigen Schichten werden auch *Shallow Neural Network* genannt, eine genaue Definition, ab wann ein Netz als tief gilt, gibt es aber nicht. [64, S. 286]

Im Folgenden werden verschiedene statistische Modelle sowie einige Typen von neuronalen Netzen grundlegend vorgestellt.

2.2.1.1. Statistische Modelle

Klassische statistische Methoden können bereits einige Anwendungsfälle von maschinellem Lernen abdecken. Nicht immer ist es notwendig, rechenaufwendigere künstliche neuronale Netze, die in Abschnitt 2.2.1.2 erklärt werden, einzusetzen.

In den folgenden Absätzen werden einige dieser Methoden vorgestellt.

Regression Regressionen können als Methoden des überwachten Lernens verwendet werden. Bei der linearen Regression wird anhand von Wertepaaren (x_i, y_i) als „Trainingsmaterial" eine Ausgleichsgerade gezeichnet (siehe Abb. 2.6), mit deren Hilfe nun auch für bisher unbekannte x passende y vorhergesagt werden [66, S. 107].

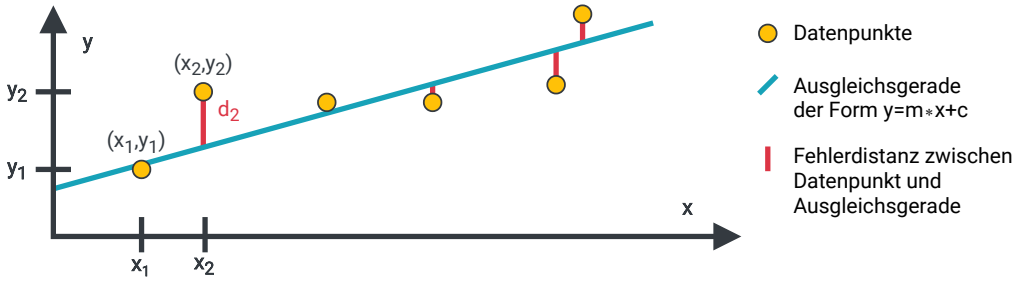


Abb. 2.6.: Beispielhafte lineare Regression (nach [190, S. 23; 131, S. 9; 64, S. 21]).

Die Regressionsgerade $f(x) = m * x + c$ wird so gewählt, das für alle n Punkte der durch die Annäherung entstehende Fehler $l = \sum_{i=1}^{n} (d_i)$ mit $d_i = |y_i - f(y_i)|$ minimal ist. Eine Optimierungsaufgabe wie $\min_{m,c} (l)$ ist ein einfaches mathematisches Problem aus dem Bereich der Analysis, wobei in diesem Fall die Geradensteigung m und der y-Achsenabschnitt c die zu bestimmenden Parameter sind. [190, S. 20–21; 66, S. 109]

Die zu minimierende Formel l wird auch Kostenfunktion, *loss function* oder *error function* genannt [190, S. 19].

Zur Messung des Fehlers können verschiedene Distanzmaße d_i und damit Kostenfunktionen verwendet werden. Diese unterscheiden sich vor allem dann, wenn ein mehrdimensionaler Merkmalsvektor X auf ein y abgebildet wird. Der oben genannte Fall entspricht dank dem eindimensionalem Eingabevektor sowohl der euklidische Distanz $d(p,q) = \sqrt{\sum_{i=1}^{k}(q_i - p_i)^2}$ als auch der City-Block Distanz $d(p,q) = \sum_{i=1}^{k} |q_i - p_i|$ [64, S. 44]. Weitere gängige Distanzfunktionen sind die für große Datenmengen besonders geeignete Hamming-Distanz [70, S. 119; 142, S. 8] und die gegenüber Skalenniveaus invariante Mahalanobis Distanz [131, S. 98; 123, S. 25]. Die euklidische Distanz wird im Bereich des maschinellen Lernens am häufigsten verwendet [131, S. 17–18; 64, S. 44–45].

Wird statt einer Gerade ein Polynom höheren Grades verwendet, so spricht man von einer Polynom-Regression [190, S. 115]. Auch eine Berücksichtigung anderer Kurvenverläufe wie exponentialem Anstieg sind durch Verwendung entsprechender Funktionen möglich, zum Beispiel mit einer exponentiellen Regression oder Sättigung mittels logistischer Regresseion [190, S. 76–77].

Eine Regression kann unter anderem zur Vorhersage von Zeitreihen (Werten in der Zukunft) oder zum Füllen fehlender Werte eingesetzt werden. So können beispielsweise die Anzahl verkaufter Produkte abgeschätzt werden oder die Anzahl der Kunden, die eine Website besucht haben und besuchen werden.

***Support Vector Machine* (*SVM*)** Eine *Support Vector Machine* (*SVM*) ist ein überwachtes Lernverfahren zur Trennung von Objekten in einem Merkmalsraum. Bildlich wird wie in Abb. 2.7 gezeigt eine Trennlinie zwischen die im Merkmalsraum angeordneten Objektcluster gezeichnet, sodass diese voneinander getrennt werden können [64, S. 149].

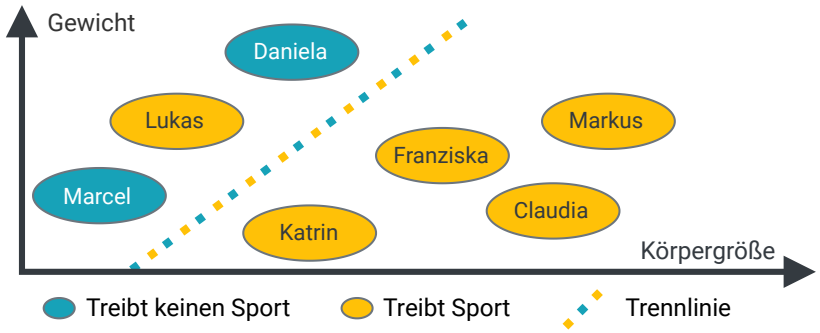


Abb. 2.7.: Beispiel *SVM* zur Einschätzung, ob Personen sportlich sind oder nicht, anhand der Merkmale Gewicht und Körpergröße.

Die berechnete Funktion kann zur Klassifikation genutzt werden. Hierfür wird das zu klassifizierende Objekt anhand seiner Merkmale (in Abb. 2.7 Gewicht und Körpergröße) als Punkt in den Graph eingezeichnet. Befindet sich der Punkt oberhalb der Funktion, so wird das Objekt als „Treibt keinen Sport“ eingeordnet, ansonsten als „Treibt Sport“.

Bei der gezeigten *SVM* handelt es sich um eine lineare *SVM* [64, S. 114]. Nichtlineare *SVM* verwenden statt einer Gerade Funktionen höherer Grade zur Trennung des Merkmalsraums.

Die Trennfunktion wird ähnlich wie bei der Regressionsgerade so gewählt, dass der durch sie entstehende Fehler, beispielsweise die Anzahl der falsch klassifizierten Objekte, minimal ist.

Weitere klassische Methoden für überwachtes Lernen Neben der vorgestellten Regression und *SVM* gibt es weitere Methoden, die für überwachtes Lernen eingesetzt werden können. Sie sind für ein Verständnis der weiteren Arbeit nicht erforderlich, weshalb sie nur kurz beschrieben werden.

Beim *k-nearest Neighbor* Klassifikator werden ähnlich wie bei der *SVM* Objekte im Merkmalsraum angeordnet, es gibt jedoch keine Trennlinie. Die Klassifikation eines neuen Objekts erfolgt durch Bestimmung der Klasse, die in den k nächsten Nachbar-Objekten am häufigsten auftritt. [131, S. 16–18]

Auch Entscheidungsbäume (englisch *Decision Trees*) können zur Klassifikation genutzt werden. Bei diesen wird mit Hilfe eines Binärbaumes an jedem Knoten eine Bedingung geprüft, was letztendlich zur Einordnung in Klassen führt [131, S. 544–550].

Ein Bayes Klassifikator basiert auf dem in Formel 2.1 gezeigtem Satz von Bayes, der die Berechnung bedingter Wahrscheinlichkeiten beschreibt.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} \tag{2.1}$$

Sehr vereinfacht formuliert benutzt der Bayes Klassifikator die bedingte Wahrscheinlichkeit $P(A \mid B)$, um unter gegebenen Merkmalen (Bedingung B) die Wahrscheinlichkeiten für jede Klasse A zu berechnen, sodass anschließend die wahrscheinlichste Klasse als Vorhersage ausgewählt werden kann [131, S. 82–87].

Klassische Methoden für unüberwachtes Learning Auch für unüberwachtes Lernen gibt es einige Methoden, denen klassische Statistik zu Grunde liegt. Einige Ansätze basieren auf Algorithmen, die selbstständig eine Menge von Objekten anhand derer Merkmale in Gruppen (*Cluster*) unterteilen, ganz ohne zusätzliche Informationen oder Trainingsdaten.

Zur Ermittlung der *Cluster* werden Ähnlichkeitsmaße verwendet, sodass Objekte, deren Ähnlichkeit besonders hoch ist, gruppiert werden können. Einige Ähnlichkeitsmaße basieren auf Distanzfunktionen, wie die bei der Regression genannten.

Darüber hinaus gibt es Ansätze, die zur Erkennung von Anomalien und Ausreißern, also einer binärem Zuordnung optimiert sind.

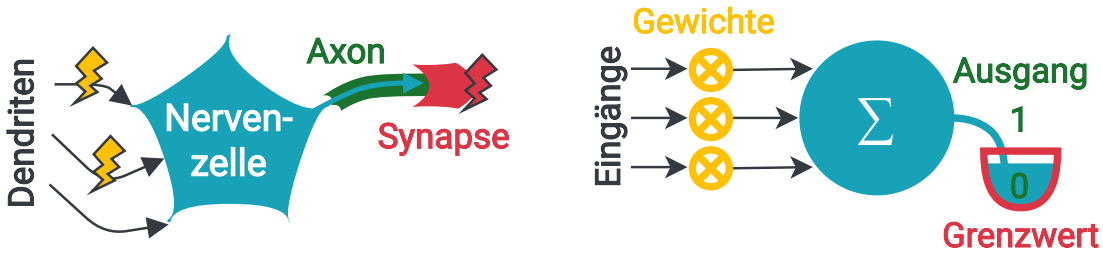
In dieser Arbeit wird kein unüberwachtes Lernen eingesetzt, weshalb nicht näher auf die genannten Methoden eingegangen wird.

2.2.1.2. Künstliche Neuronale Netze (KNN)

Viele technologische Erfindungen haben Inspiration durch die Natur erhalten. Auch im Bereich des maschinellen Lernens wurde sich an einem biologischen Vorbild orientiert. Die Struktur und Funktionsweise des menschlichen Gehirns sind Vorbilder für künstliche neuronale Netze.

Neuronale Netze im Nervensystem des menschlichen Gehirns bestehen aus verschiedenen Neuronen, die in aufeinanderfolgenden Schichten (*layers*) angeordnet sind [64, S. 280] und durch Synapsen verbunden sind [70, S. 13]. Ein solches biologisches Neuron empfängt Reize des Körpers als Eingabe, verarbeitet die Eingabe und sendet dann je nach Entscheidung ein Signal oder nicht (siehe Abb. 2.8a).

Künstliche Neuronen, 1943 vorgestellt von McCulloch und Pitts [122], basieren auf einem ähnlichen Prinzip und dienen als eine Art binäres Logikgatter [70, S. 15; 64, S. 281]. Sie „feuern“ ihr Signal beispielsweise immer dann ab, wenn mindestens eine bestimmte Anzahl von Eingängen aktiv sind [122]. Die Eingänge können vor der Summierung mit Gewichten multipliziert werden, um die gewünschten Ausgangswerte zu erhalten. Dieses Prinzip wird in Abb. 2.8b gezeigt und im Folgenden näher beschrieben.



(a) Biologisches Neuron (nach [70, S. 13; 64, S. 280]) (b) Künstliches Neuron (nach [70, S. 14; 64, S. 282])

Abb. 2.8.: Funktionsweise biologischer und künstlicher Neuronen. In beiden Fällen werden abhängig von Eingangswerten (je links) Ausgangswerte (je rechts) bestimmt.

Das künstliche Neuron mit n-dimensionalen Eingangswerten (*inputs*) $x \in \mathbb{R}^n$ und Gewichten (*weights*) $w \in \mathbb{R}^m$ kann durch die folgende Formel 2.2 repräsentiert werden [148, S. 469]:

$$f(x) = A(x * w^T) = A\left(\sum_{i=0}^{n-1}(x_i * w_i)\right) \text{ mit } A(x) = \left\{ \begin{array}{ll} 0 & \text{für } x < 0 \\ 1 & \text{sonst} \end{array} \right\} \tag{2.2}$$

Im Beispiel wird als Aktivierungsfunktion $A(x)$ die Stufenfunktion verwendet. Weitere Grenzwertfunktionen werden auf Seite 20 beschriebenen.

Ein solches künstliches Neuron kann bereits zur binären Klassifikation benutzt werden und damit Fragestellungen wie „Gehört eine x-Wert Kombination zu einer Klasse oder nicht?“ beantworten. Anhand einer Menge von Beispielwerten können beim „Training“ Werte für die Gewichte an den Eingängen des Neurons bestimmt werden, die später bei der „Inferenz“ dafür genutzt werden, um anhand dieses Modells vorherzusagen, ob bisher nicht-gesehene x-Wert-Kombinationen zu der Klasse gehören oder nicht.

Rechenbeispiel zu künstlichen Neuronen Zur Visualisierung der Funktionsweise des in Abb. 2.8b gezeigten künstlichen Neurons wird ein Rechenbeispiel durchgeführt.

Zur Vereinfachung wird in dem Rechenbeispiel angenommen, dass das Neuron nur zwei Eingänge besitzt. Es handelt sich also um eine binäre Klassifikation mit einem Ausgabewert $y \in \{0, 1\}$) anhand von $n = 2$ Merkmalen (Eingabewerten).

Als „Trainingsmaterial" liegen die $m = 3$ Wertepaare x^i als Matrix $X \in \mathbb{R}^{m \times n}$ vor (siehe Formel 2.3).

Zur Zuordnung von Klassen sollen mithilfe einer zu bestimmenden Vorhersagefunktion $f(x) = A(x * w)$ diese Eingabewerte auf die folgenden Ausgabewerte abgebildet werden:

$$\text{für } X = \begin{pmatrix} x^0 \\ x^1 \\ x^2 \end{pmatrix} = \begin{pmatrix} 4 & 1 \\ 3 & 2 \\ 5 & 6 \end{pmatrix} \text{ soll gelten: } \begin{pmatrix} f(x^0) \\ f(x^1) \\ f(x^2) \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \tag{2.3}$$

Der Ausgang soll also nur für das zweite und dritte Wertepaar aktiviert sein, das heißt nur das zweite und dritte Objekt sind Instanzen der zu prüfenden Klasse.

Zur Bestimmung der Vorhersagefunktion des Neurons müssen die zwei unbekannten Gewichte bestimmt werden. In der Praxis werden diese Gewichte mit speziellen Algorithmen wie *Back Propagation* oder anderen Optimierungsverfahren ermittelt. In diesem Rechenbeispiel wird nicht näher auf diese eingangen, sondern es wird davon ausgegangen, dass diese durch geschicktes Ausprobieren verschiedener Werte ermittelt wurden: $w = \begin{pmatrix} -1 & 2 \end{pmatrix}^T$.

Die mit diesen Gewichten zusammengesetzte Funktion sagt alle Trainingsdaten korrekt vorher, wenn zunächst mit $\sum_{rows}(X) = \begin{pmatrix} \sum_{i=0}^{n}(x_i^0) & \dots & \sum_{i=0}^{n}(x_i^m) \end{pmatrix}^T \forall X \in \mathbb{R}^{m \times n}$ die Elemente jeder Reihe summiert und anschließend die Aktivierungsfunktion elementweise angewendet wird ($A_V(x) = \begin{pmatrix} A(x_1) & \dots & A(x_m) \end{pmatrix}^T \forall x \in \mathbb{R}^m$). So ergibt sich die gewünschte Ausgabe:

$$\begin{aligned} \text{Probe: } \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \stackrel{!}{=} y &= A_V\left(\sum_{rows}\left(X * w^T\right)\right) = A_V\left(\sum_{rows}\left(\begin{pmatrix} 4 & 1 \\ 3 & 2 \\ 5 & 6 \end{pmatrix} * \begin{pmatrix} -1 & 2 \end{pmatrix}\right)\right) \\ &= A_V\left(\sum_{rows}\left(\begin{pmatrix} -4 & 2 \\ -3 & 4 \\ -5 & 12 \end{pmatrix}\right)\right) = A_V\left(\begin{pmatrix} -4+2 \\ -3+4 \\ -5+12 \end{pmatrix}\right) \\ &= A_V\left(\begin{pmatrix} -2 \\ 1 \\ 7 \end{pmatrix}\right) = \begin{pmatrix} A(-2) \\ A(1) \\ A(7) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \text{ q.e.d.} \end{aligned} \tag{2.4}$$

Die in Formel 2.4 gezeigte Probe ist erfolgreich, weshalb die Funktion mit diesen Gewichten nun zur Inferenz genutzt werden kann. Würde sie nicht exakt das gewünschte Ergebnis zeigen, so kann mithilfe eines Ähnlichkeitsmaßes oder einer Distanzfunktion, beispielsweise anhand der Anzahl der korrekt vorhergesagten Klassen, eine Genauigkeit (*accuracy*) und eine Fehlerrate der Funktion bestimmt werden, die dann unter 100% liegt.

Nach dem Training des Modells können die Gewichte gespeichert werden, sodass diese bei der späteren Anwendung des Modells, der sogenannten Inferenz, verwendet werden können.

Um beispielsweise zu prüfen, ob das Objekt $x^3 = \begin{pmatrix} 3 & 4 \end{pmatrix}$ der Klasse angehört, kann dieses in die mit Gewichten versehene Funktion f eingesetzt werden:

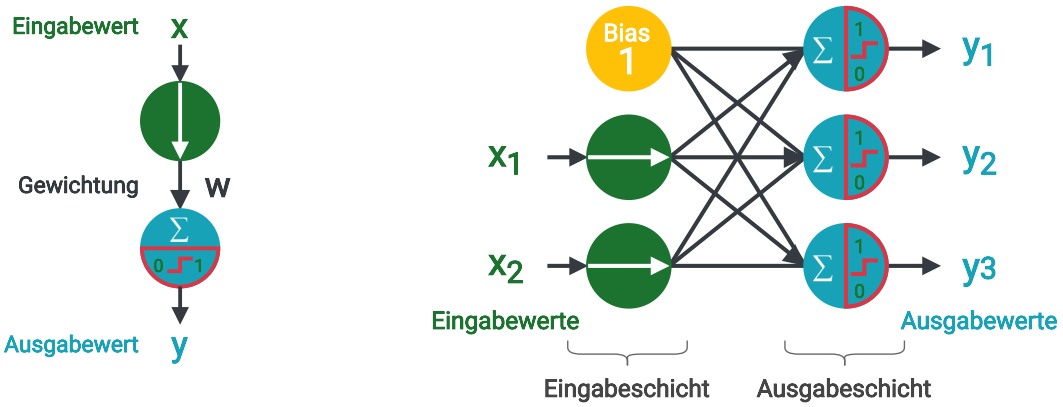
$$f\left(\begin{pmatrix} 3 & 4 \end{pmatrix}\right) = A(3 * -1 + 4 * 2) = A(-3 + 8) = A(5) = 1 \tag{2.5}$$

Das getestete Objekt x^3 gehört also laut dem Modell der Klasse an.

Durch geschickte Kombination der verschiedenen Eingänge können auf diese Weise UND, ODER und NICHT-Gatter realisiert werden [64, S. 281].

Perzeptron Das Perzeptron [154] ist eines der ältesten KNN. Es wurde 1957 von Frank Rosenblatt entwickelt.

In Abb. 2.9a wird das kleinste mögliche Perzeptron mit nur einem Neuron gezeigt. Ein Eingabewert wird auf einen Ausgabewert abgebildet.



(a) Perzeptron mit einem Input und einem Output

(b) Perzeptron mit zwei Inputs und drei Outputs (nach [64, S. 283; 70, S. 71]). Jeder Pfeil zwischen den Schichten hat ein Gewicht.

Abb. 2.9.: Beispiele für Perzeptrone.

In der zweiten Abb. 2.9b wird ein Perzeptron mit zwei Eingabewerten (*Inputs*) und drei Ausgabewerten (*Outputs*) gezeigt. Es enthält darüber hinaus ein Bias Merkmal (*feature*) $x_0 = 1$.

Diese Struktur und die dahinterliegende Mathematik ist sehr ähnlich zu dem eben vorgestellten künstlichen Neuron.

Jeder Knoten im Graph repräsentiert ein Neuron. Die Ausgabe aller Neuronen des Perzeptrons aus Abb. 2.9b kann wie folgt berechnet werden: $Y_{W,b}(X) = A(X * W_X + W_b)$ (mit der Gewichtsmatrix W_X, dem Eingabevektors X und den Bias-Gewichten W_b) [64, S. 283].

Die Gewichte werden zu Beginn zufällig gewählt und während des Trainings angepasst. A bezeichnet die verwendete Stufen-Aktivierungsfunktion, die den Wertebereich der Ausgaben je Neuron normiert.

Die Aktualisierung der Gewichte läuft in jedem Zeitschritt mit einer Funktion zur Gewichtsaktualisierung der Art $w_{i,j}^{t+1} = w_{i,j}^t + n(y_j - y_j^d)x_i$ [64, S. 284]. $w_{i,j}$ ist hierbei das Gewicht zwischen dem i-ten Input und dem j-ten Output Neuron. x_i ist der Wert des i-ten Input, und analog dazu y_j die Ausgabe des j-ten Output-Neurons des aktuellen Trainings. y_j^d ist der erzielte Ausgabewert des j-ten Ausgabeneurons, und n die Lernrate.

Weitere Details zu derartigen Funktionen werden im folgenden Abschnitt zu Optimierungsfunktionen beschrieben.

***Multi Layer Perceptron* (*MLP*)** Ein Perzeptron mit mehreren Schichten wird *Multi Layer Perceptron* (*MLP*) genannt. In Abb. 2.10 wird ein *MLP* mit einer versteckten Schicht und somit insgesamt zwei Schichten dargestellt (die Eingabeschicht wird nicht gezählt).

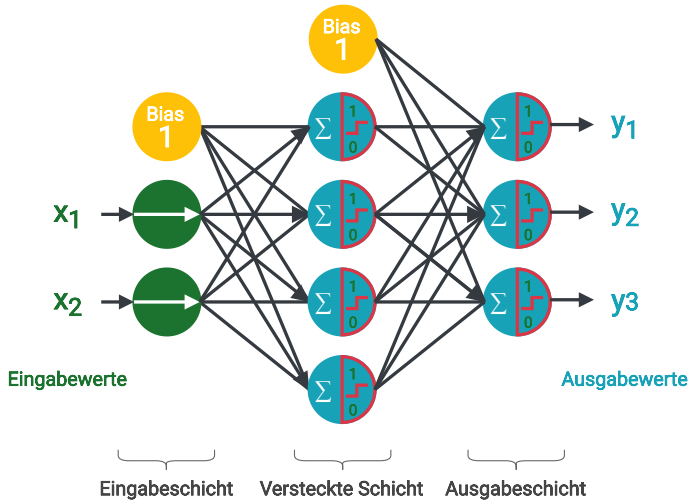


Abb. 2.10.: Zweischichtiges *MLP* (nach [64, S. 286; 148, S. 471]).

Mithilfe eines *Multi Layer Perceptrons* (*MLPs*) ist es nun auch möglich, ein exklusives ODER (XOR Gatter) abzubilden, wodurch es sich von linearen Klassifikationsmodellen wie der Klassifikation durch lineare Regression absetzt [64, S. 285].

Zum Training wird der *Backpropagation* Algorithmus [155] eingesetzt [148, S. 472–474]. Im vorigen Beispiel wurden die Gewichte „erraten“, in der Praxis funktioniert das aber nicht ganz zufällig, sondern nur mit zufälliger Initialisierung. Anschließend werden die Gewichtswerte in kleinen Schritten („Deltas“) erhöht oder verringert, um die Gewichte für einen minimalen Fehler abzuschätzen. Diese Optimierung kann also als Minimierungsproblem angesehen werden.

Eine gängige Methode zur Bestimmung von Minima ist Differenzierung. Da automatische Differenzierung immer noch mit hohem Rechenaufwand verbunden ist [147], wird in der Praxis beim *Backpropagation* Algorithmus [155] eine Annäherung im Gradientenabstiegsverfahren (englisch *Stochastic Gradient Descent* (*SGD*)) verwendet [15, S. 17–20; 66, S. 151–153, 294–300, 308].

Aktivierungsfunktionen Im vorangehenden Rechenbeispiel wurde als Aktivierungsfunktion eine Stufenfunktion verwendet, die bei allen negativen Werten 0 zurückgibt, und für alle andere Werte eine 1. Die hierfür benötigte Formel wird in Abb. 2.11a gezeigt, der zugehörige Graph in Abb. 2.12b. Je nach Anwendungsfall und Netzarchitektur kann es Sinn machen, eine andere Aktivierungsfunktion zu verwenden. In Abb. 2.12 werden daher einige alternative Aktivierungsfunktionen $f(x)$ gezeigt, die am Ende einer Schicht verwendet werden können, um abhängig von der Summe x der Werte der einzelnen Neuronen den „Aktivierungsstatus“ zu setzen (aktiviert $\iff f(x) > 1$).

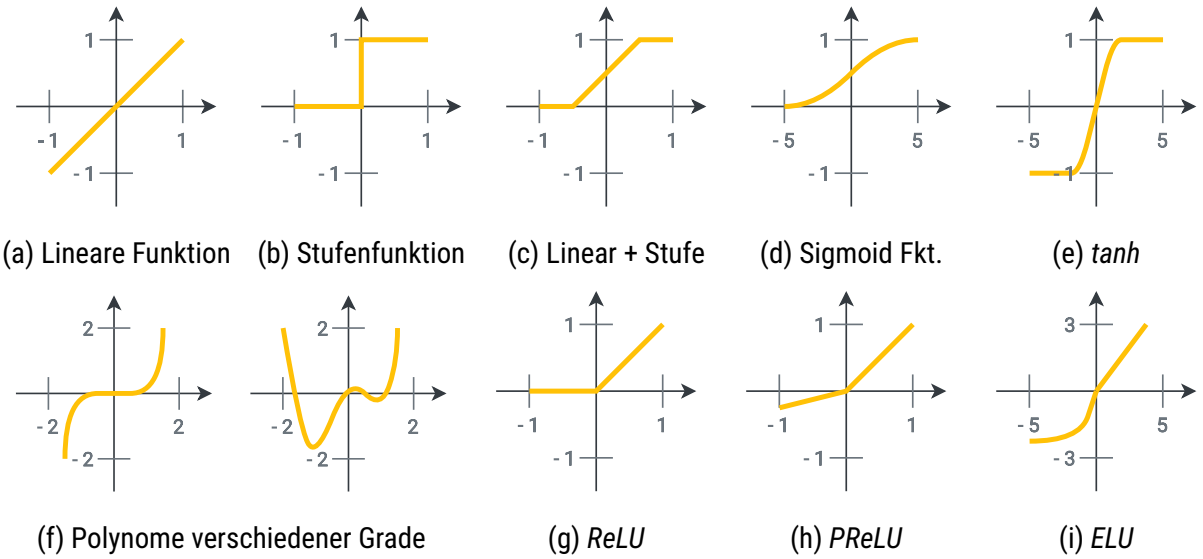
Die Stufenfunktion kann mit einer linearen Funktion der Art $f(x) = m * x$ kombiniert werden (siehe Abb. 2.12a–c), um die in in Abb. 2.11b mit $i = 0.5$ gezeigte Funktion zu schaffen. Diese Funktion approximiert die Sigmoid Funktion abschnittsweise [70, S. 36]. Die logistische Sigmoid Funktion $\frac{1}{(1+e^{-x})}$ (siehe Abb. 2.12d)wird ebenfalls als Aktivierungsfunktion verwendet [131, S. 21; 162, S. 8].

$$\left\{ \begin{array}{ll} 0 & \text{für } x < 0 \\ 1 & \text{sonst} \end{array} \right\} \qquad \left\{ \begin{array}{ll} 0 & \text{für } x < -i \\ x + i & \text{für } -i \leq x \leq i \\ 1 & \text{sonst} \end{array} \right\} \qquad \left\{ \begin{array}{ll} x & \text{für } x \geq 0 \\ \alpha(e^x - 1) & \text{sonst} \end{array} \right\}$$

(a) Stufenfunktion [64, S. 282] (b) Linear + Stufe [70, S. 36] (c) *ELU* [42, S. 5]

Abb. 2.11.: Formeln für abschnittsweise definierte Aktivierungsfunktionen. $f(x) = \ldots$

Auch andere klassische Funktionen wie *Tangens Hyperbolicus* $\tanh(x)$ (siehe Abb. 2.12e) [162, S. 9; 148, S. 469–470] oder Polynome verschiedener Grade (siehe Abb. 2.12f) können als Aktivierungsfunktionen verwendet werden [64, S. 289].



(a) Lineare Funktion (b) Stufenfunktion (c) Linear + Stufe (d) Sigmoid Fkt. (e) *tanh*

(f) Polynome verschiedener Grade (g) *ReLU* (h) *PReLU* (i) *ELU*

Abb. 2.12.: Aktivierungsfunktionen.

Ein wichtiges Problem, das beim Training von KNN auftreten kann, ist der sogenannte *Vanishing Gradient*. Er bezeichnet das Problem, dass die Ableitung (der *Gradient*) der Kurve im Verlaufe des Trainings gegen Null geht. Er kann durch Verwendung geeigneter Aktivierungsfunktionen vermieden werden.

Der *Vanishing Gradient* entsteht im folgenden Fall: Wenn eine Aktivierungsfunktion einen großen Eingabebereich auf einen kleinen Ausgabebereich abbildet, so verändern große Änderungen am Input der Aktivierungsfunktion deren Output nur wenig. Dadurch entsteht nur ein kleiner, meistens

nahe bei Null liegender Ableitungswert. Während des Gewicht-Updates, also dann, wenn mit dem *Backpropagation* Algorithmus [155] die Ableitungen des Netzwerks berechnet werden, werden die Gradienten jeder Schicht (aufgrund der Kettenregel) miteinander multipliziert.

Die Multiplikation mehrerer kleiner Werte nahe Null führt zu noch kleineren Werten. So gehen dann auch die Ableitungswerte im Verlauf der Backpropagation immer näher an 0, und die Gewichte sowie deren Neuronen werden gesättigt. Die Gewichte aktualisieren sich nicht mehr wie vorgesehen, woraufhin der gemessene Fehler (Loss) des Netzes nicht weiter sinkt und weiteres Training zu keiner weiteren Verbesserung führt. [45, S. 6]

Die *Rectified Linear Unit* (*ReLU*) Funktion $f(x) = \max(0, x)$ [132; 162, S. 10] (siehe Abb. 2.12g) verhindert den *Vanishing Gradient* für alle $x > 0$, für alle kleineren Werte tritt er aber noch auf.

Die verbesserte *Parametric ReLU* (*PReLU*) Funktion verhindert das Problem auch für kleine Werte, indem wie in Abb. 2.12h gezeigt eine Funktion mit leichter Steigung verwendet wird: $f(x) = \max(\alpha x, x), \alpha \leq 1$, wobei α lernbar ist und durch den *Backpropagation* Algorithmus [155] bestimmt wird. Wird hingegen ein vorbestimmtes $\alpha = 0.01$ verwendet, so wird die Funktion *Leaky ReLU* genannt [45, S. 11; 200, S. 2]. Bei $\alpha = 0$ entspricht die Funktion der *ReLU* Funktion.

Eine etwas rechenaufwendigere, aber dafür rausch-resistentere Variante der *PReLU* Funktion ist die *Exponential Linear Unit* (*ELU*) Funktion aus Abb. 2.11c, die in Abb. 2.12i gezeigt wird.

Netzwerke, die eine auf $\max()$ basierende Aktivierungsfunktion verwenden, werden auch *Maxout Networks* genannt. Die Generalisierung von *PReLU* und *ReLU* lautet daher auch *maxout* Funktion und kann als $max(w_1^T x + b_1, w_2^T x + b_2)$ geschrieben werden. Werden diese Parameter gelernt, so verdoppelt das allerdings die Anzahl der Parameter pro Neuron und damit den Rechenaufwand.

***Convolutional Neural Network* (*CNN*)** Sollen Bilder klassifiziert werden, so werden deren Pixel-Farbwerte (siehe Abschnitt 2.1) als Eingabewerte verwendet. Bei einem quadratischen Bild der Größe 500 Pixel gibt es also $500 * 500 * 3 = 750.000$ Eingabewerte. Wäre das Bild 100 weitere Pixel in jede Richtung größer, so würden $1.080.000$ Eingabewerte benötigt.

Dieser rapide Anstieg von Eingabewerten je Pixel wird auch *Curse of Dimensionality* genannt, übersetzt etwa ein „Fluch der Dimensionalität“, womit gemeint ist, dass der Rechenaufwand mit jedem Neuron steigt [131, S. 18–19; 64, S. 216–217].

CNN lösen dieses Problem durch Verwendung partiell verbundener Schichten und dem Teilen von Gewichten [64, S. 433]. Sie basieren auf dem biologischen Vorbild des visuellen Cortex, dessen Neuronen nur ein eingeschränktes rezeptives Feld haben. Hierdurch wird die Flut an Informationen je Neuron reduziert, weshalb Signale leichter verarbeitet und komplexe Muster erkannt werden können. [131, S. 565; 64, S. 432; 66, S. 365]

Die wichtigste Komponente eines *CNN* ist die Faltungsschicht (*convolutional layer*). Hierbei werden Neuronen der Eingabeschicht nicht mit jedem Pixel des Eingabebilds verbunden, sondern nur mit denen, die in ihrem rezeptiven Feld liegen.

Über mehrere Schichten hinweg bietet diese Architektur die Möglichkeit, in den ersten Schichten lokale Muster zu lernen und in den höheren Schichten diese zu größeren Gesamt-Mustern zusammenzusetzen [64, S. 434].

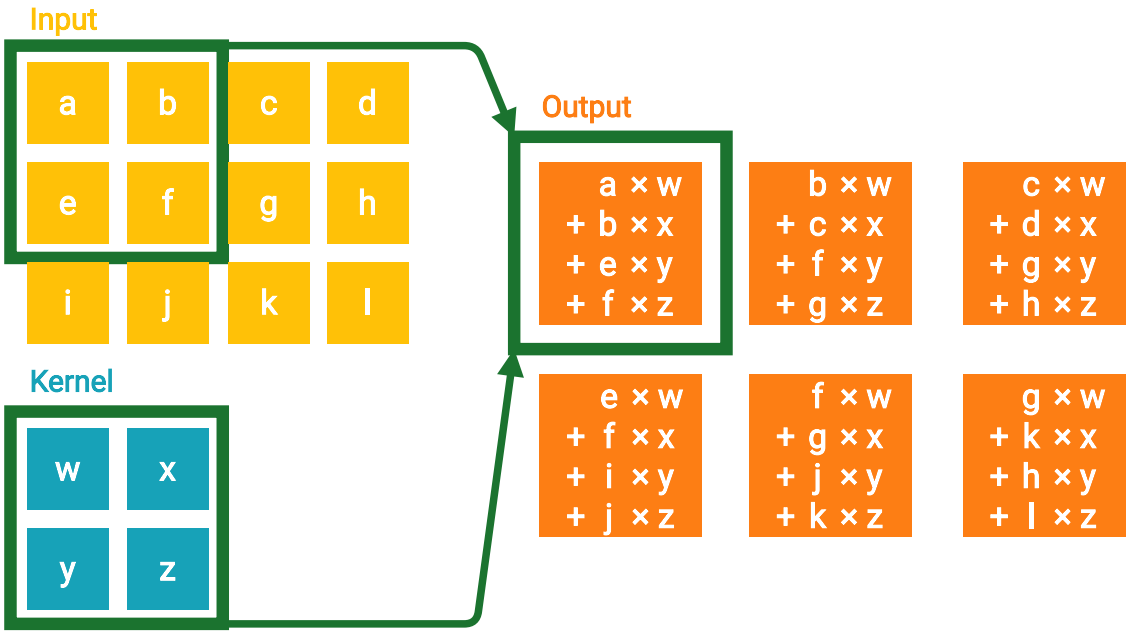
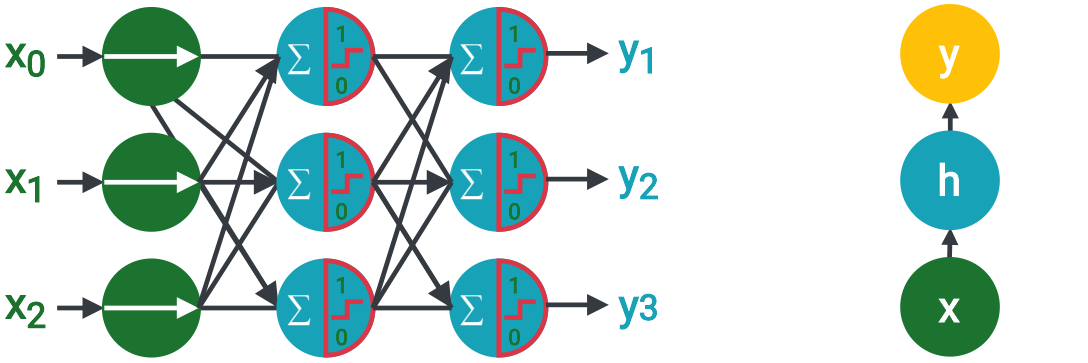


Abb. 2.13.: Anwendung eines *CNN*-Kernels (nach [66, S. 334]).

Die Faltung erfolgt mithilfe eines sogenannten Kernels. Dieser wird „über das Bild geschoben“, wobei für jede Position wie in Abb. 2.13 gezeigt die Werte des Kernels mit den Werten der Eingabe multipliziert werden.

RNN* und *LSTM Die Knoten der bisher vorgestellten Netze sind geradlinig angeordnet, weshalb sie *Feedforward Neural Networks* [131, S. 563; 64, S. 286] genannt werden. Netze, deren Knoten einen Zyklus formen heißen *Recurrent Neural Network* (*RNN*).

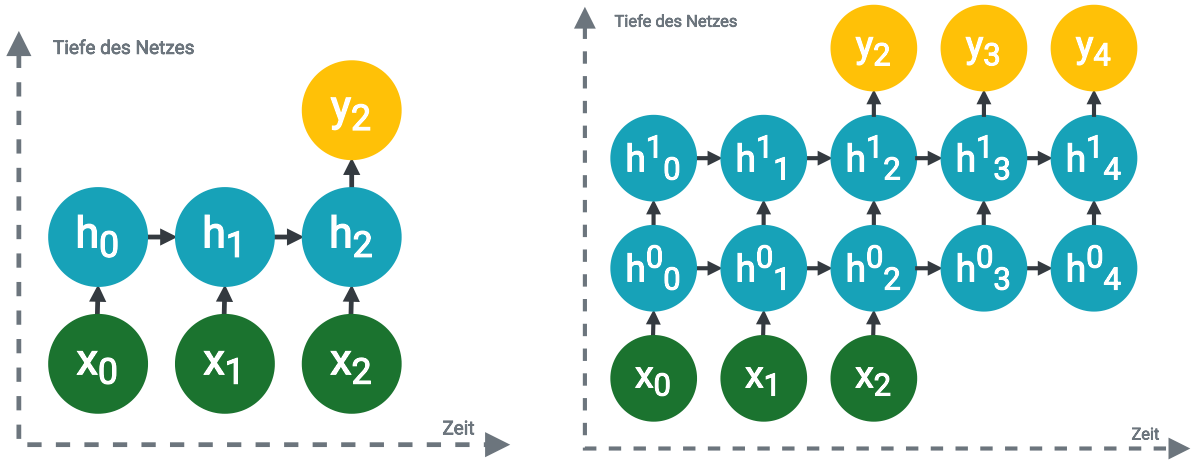
RNN können besonders gut zur Auswertung von Zeitreihen benutzt werden. Um mehrdimensionalen Ein- und Ausgabevektoren im Verlauf der Zeit auch in den Diagrammen darstellen zu können, wird im Folgenden eine Dimensionsreduktion in der Darstellung vorgenommen (siehe Abb. 2.14).



(a) Klassisches KNN (b) Darstellung als *RNN*

Abb. 2.14.: Dimensionsreduzierte Darstellung eines KNNs.

RNN können also genutzt werden, um Sequenzen von Eingangswerten auszuwerten. Sie können dabei entweder anhand einer Eingabesequenz eine einzelne Ausgabe generieren, zum Beispiel zur Klassifizierung (*many to one*, siehe Abb. 2.15a) oder anhand der Eingabesequenz eine Ausgabesequenz generieren (*many to many*, siehe Abb. 2.15b).



(a) *many to one RNN* (b) *many to many RNN*

Abb. 2.15.: Typen von *RNN*.

Für eine Handlungserkennung könnten beide Arten von *RNN* zum Einsatz kommen, da je nach Anwendungsfall entweder die Klassifikation eines abgeschlossenen Videos zu einer Klasse (*many to one*) oder die Auswertung eines Videostreams, und damit die Vorhersage mehrerer Handlungen hintereinander, relevant wäre (*many to many*).

Das im Abschnitt „Aktivierungsfunktionen“ bereits angesprochene *Vanishing Gradient* Problem sorgt bei *RNN* für eine Verkürzung der Zeit, die zurückgeschaut werden kann [51, S. 2].

Long Short-Term Memory (*LSTM*) Netze sind eine weiterentwickelte Spezialform von *RNN*. *LSTM* versuchen, lange zeitliche Verläufe (*long-term dependencies*) korrekt abzubilden, indem sie relevante Informationen aus der Vergangenheit für den aktuellen Output berücksichtigen [66, S. 408].

Die interne Struktur eines *LSTM* ist komplexer als die eines *RNN*. *LSTM* nutzen nicht nur den aktuellen *Input*, die vergangenen *Outputs* und versteckte Schichten, sondern sie führen darüber hinaus einen Zellen-Zustand (*cell state*) s_t ein, der Informationen von einer Zelle zur nächsten trägt [66, S. 408–412]. Mithilfe dieser Zustände können *LSTM* lernen, ab wann alte Informationen vergessen werden sollten [66, S. 409].

Generative Modelle Ein *Generative Adversial Network* (*GAN*) ist ein künstliches Neuronales Netz, das aus zwei als Gegenspieler agierenden Teilen besteht. In Abb. 2.16 werden die beiden Komponenten gezeigt.

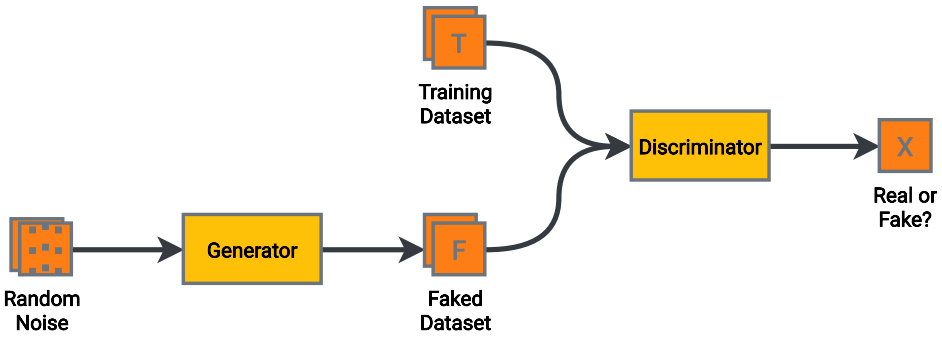


Abb. 2.16.: Komponenten eines *GAN* (nach [15, S. 101]).

Der Generator (*generative model*) kreiert kontinuierlich neue zufällige Daten aus zufälligem Rauschen heraus. Er wird mit der Zeit besser darin, realistisch aussehende Daten zu generieren, die ähnlich wie ein Trainingsdatensatz sind, da es Rückmeldung vom Diskriminator bekommt, wie gut der *Fake* war [15, S. 99–102].

Der Diskriminator (*discriminative model*) berechnet die Wahrscheinlichkeit dafür, dass ein Datensatz vom Generator kommt, statt aus dem Trainingssatz. Er wird mit der Zeit ein wenig besser darin, diese *Fakes* zu unterscheiden [15, S. 99–102].

Generative Modelle können zum Generieren von Bildern genutzt werden. Hierfür startet das generative Model G mit einer Menge von Zufallsvektoren sowie einer Ziel-Klasse, in die es trainiert wird oder werden soll. Anschließend versucht der Generator, Iteration für Iteration realistischere Bilder zu generieren, welche irgendwann idealerweise von dem Diskriminator nicht mehr von echten Bildern unterschieden werden können. [15, S. 99–102]

Hierbei ist es wichtig, dass der Diskriminator nicht zu schnell lernt, die *Fakes* von den echten Bildern zu unterscheiden. Wäre dies der Fall, würde der Diskriminator alle Bilder als *Fakes* erkennen, weshalb der Generator kein eindeutiges Feedback bekommt, ob seine Objekte besser werden. Damit dieser Fall nicht eintritt, werden Gewichte für die Lernfortschritte gesetzt. [15, S. 71–74]

Ein weiteres generatives Modell sind sogenannte *Variational Autoencoder* (*VAE*). Klassische Autoencoder nutzen neuronale Netze, um Datenmengen auf in der Regel kleinere (niedrig-dimensionalere) latente arbiträre Repräsentationen abzubilden (*Encoder*) und später wieder zurückzurechnen (*Decoder*) [15, S. 71]. *VAE*s verfolgen ein ähnliches Prinzip, erlernen aber latente Parameter auf Basis einer Zufallsverteilung. Dies ermöglicht das generieren neuer zufälliger *Samples* anhand eines Zufallsvektors. [15, S. 71–74]

2.2.2. Anwendungsgebiete von KI

Wie bereits einleitend erwähnt, kann KI in zahlreichen Gebieten angewendet werden. In diesem Abschnitt werden die verschiedenen Anwendungsgebiete näher beschrieben.

Die einzelnen Anwendungsfälle werden in diesem Abschnitt näher beschrieben. Sie sind gruppiert nach den Oberthemen digitale Bildverarbeitung, digitale Sprachverarbeitung, Autonome Systeme und Data Science.

Der Anwendungsfall dieser Arbeit lässt sich eindeutig dem Bereich digitale Bildverarbeitung zuordnen.

2.2.2.1. Digitale Bildverarbeitung

Wie bereits in Abschnitt 2.1 und den weiteren Grundlagen beschrieben, können Bilder mithilfe neuronaler Netze klassifiziert werden. Sie können aber auch für zahlreiche weitere Anwendungen im Bereich der Bild- und Videoanalyse eingesetzt werden.

Bildanalyse Im Bereich der Bildanalyse werden tiefe KNN am häufigsten für Objekt-Erkennung und Detektion eingesetzt [66, S. 453]. So kann beispielsweise geprüft werden, ob oder wo ein bestimmtes Objekt in einem Bild vorhanden ist und ein Rahmen gezeichnet oder eine Segmentierungsmaske erzeugt werden [66, S. 453]. Darüber hinaus ist es auch möglich, Körperhaltungen zu erkennen [198]. Eine Auswahl möglicher Methoden zur Erkennung von Objekten und Körperhaltungen wird in Abb. 2.17 gezeigt. Hierfür werden meist *CNN* eingesetzt [198].

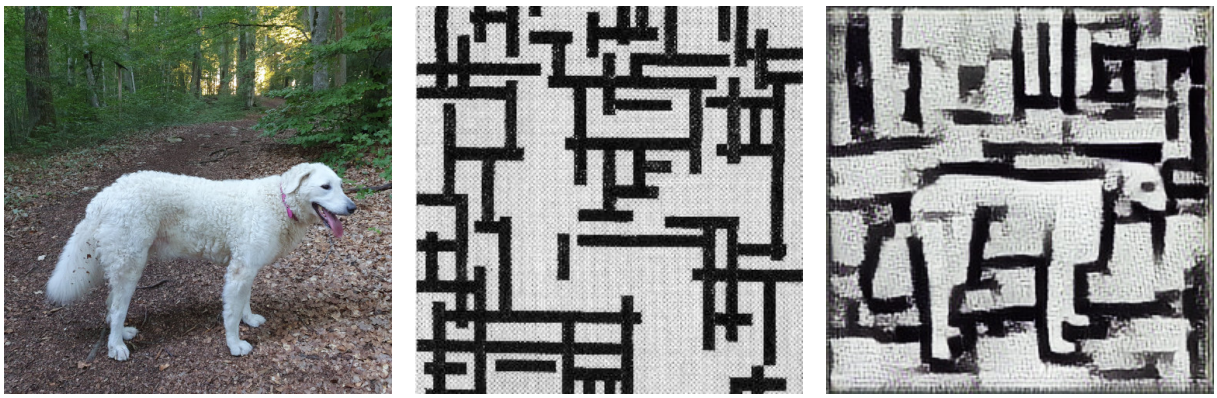
(a) Originalbild (b) Objekterkennung mit Rahmen (c) Instanz-Segmentierung

(d) Panoptische Segmentierung (e) Körperhaltung als 2D-Skelett (f) Körperhaltung als 3D-Mesh

Abb. 2.17.: Verschiede Methoden zur Bildanalyse [198].

Diese Art der Bildklassifikation ermöglicht beispielsweise das Kategorisieren und Indizieren von großen Bilddatenbanken, aber auch die Erkennung von handgeschriebenen Zeichen [105], bekannt unter dem Begriff *Optical Character Recognition* (*OCR*). Auch eine Erkennung von Hautkrebs anhand von Fotos wurde bereits realisiert [57].

Bildsynthese Mithilfe generativer Modelle ist es möglich, Bilder zu synthetisieren, also künstlich zu erzeugen. Abb. 2.18 zeigt beispielsweise die Möglichkeit der Übertragung künstlerischer Stile zwischen Bildern (*Neural Style Transfer*) [162, S. 203; 133].



(a) Originalbild (b) Künstlerischer Stil [50] (c) Kombination

Abb. 2.18.: Beispiel für Stiltransfer, erstellt mithilfe von [133].

Mit dieser Technik ist es zudem möglich, Gesichtsausdrücke zwischen Personen „auszutauschen" (*Facial Expression Transfer*), was die Grundlage für sogenannte *deep fakes* bietet. Darunter werden mit Hilfe tiefer KNN erzeugte Bilder und Videos verstanden, die beliebige Szenen zeigen [89; 90] und damit beispielsweise die Verbreitung von Falschmeldungen (*fake news*) unterstützen können [28, S. 49], aber auch ein großes Potential für die Filmindustrie bieten [12].

Die Bildsynthese kann auch nur auf bestimmte Zonen des Bilds angewendet werden, beispielsweise zum Füllen von Lücken in Bildern oder dem Entfernen störender Personen im Hintergrund eines Fotos, indem alternative Bildinhalte generiert werden (*image inpainting*) [109; 162, S. 209].

Weitere Anwendungsfälle sind das Erhöhen von Bildauflösungen (Skalierung mit intelligenter Interpolation, *super resolution*) [92; 162, S. 205] oder das Generieren fotorealistischer Bilder anhand einer einfachen Zeichnung [145; 162, S. 206].

Videoanalyse Die oben beschriebenen Methoden können statt auf voneinander unabhängigen Bildern auch auf alle Einzelbilder eines Videos angewendet werden. So können Videos zwar ausgewertet werden [162, S. 254], zeitliche Verläufe wie Bewegungsrichtungen und andere Zusammenhänge zwischen den Einzelbildern werden hiermit aber nicht berücksichtigt.

Zur Auswertung von zeitlichen Verläufen in Videos gibt es daher auch spezielle Ansätze auf Basis von *LSTM*s oder 3D-*CNN*s [162, S. 235–236, 253], die bei der Klassifikation eines Einzelbilds auch die Informationen über die vorangegangenen Einzelbilder berücksichtigen.

Eine derartige Videoanalyse, englisch *Video Content Analysis* (*VCA*), kann zur Klassifikation von Videos hilfreich sein, beispielsweise um automatisiert Videos in Genres einzuordnen [23] oder zur Realisierung eines Filters zur Erkennung von Kinderpornografie [63].

In anderen Anwendungsfällen wird die Erkennung von Bewegungen und die Verfolgung von Objekten benötigt [36] oder, wie in dieser Arbeit, die Detektion von menschlicher Handlungen [82; 33], die in Kapitel 2.4 detaillierter beschrieben wird.

2.2.2.2. Digitale Sprachverarbeitung

KI kann auch im Bereich der digitalen Sprachverarbeitung an vielen Punkten eingesetzt werden.

Grundsätzlich kann gesprochene Sprache wie jede Audiodatei digital in Form eines Signalverlaufs dargestellt werden [148, S. 37–55]. Dieser kann ebenso wie geschriebene Worte ausgewertet werden. Es gibt Anwendungen zum Verständnis und zur Verarbeitung von natürlicher gesprochener und geschriebener Sprache (*Natural Language Understanding* (*NLU*) und *Natural Language Processing* (*NLP*)).

Vielen Menschen bekannt ist die Technik der Spracherkennung. Hierunter versteht man Algorithmen, die gesprochene Sprache in Text umwandeln [148, S. 28, 331]. Auch die Umkehrfunktion, Sprachsynthese *Text-to-Speech* (*TTS*), die für einen gegebenen Text Audiodateien generiert wird häufig eingesetzt [148, S. 27, 194]. Digitale Assistenten wie Google Home, Alexa oder Siri nutzen diese beiden Techniken in Kombination mit Komponenten zum Verstehen von Text, um gestellte Fragen zu erkennen und auf diese zu antworten (*conversational AI*, *chat bots*, *reasoning & question answering*) [66, S. 428]..

Auch zur automatisierten Bearbeitung von beispielsweise E-Mails und Rechnungen können Algorithmen zum Textverständnis eingesetzt werden. Dieser Aufgabenbereich wird auch *back office automation* genannt. Dies ist auch hilfreich, um beispielsweise Stimmungen (positiv, negativ, ...) und Anfragetypen (Anfrage, Informationssuche, Kündigung, ...) aus Nutzerkommentaren/-nachrichten automatisch zu bestimmen [66, S. 428].

Weitere Forschungsfelder sind die Text-Generierung, mit der natürlichsprachliche Texte generiert werden können. Dies wird beispielsweise zur Autovervollständigung von Texten, zum Generieren von Text-Vorlagen zu einem bestimmten Thema oder zur Übersetzung zwischen verschiedenen Sprachen eingesetzt [27]. Hierfür werden *Encoder-Decoder* Modelle oder *Sequence-to-Sequence RNN* eingesetzt [184, S. 2–3].

Auch eine Klassifizierung von Stimmen, beispielsweise zur Zuordnung dieser zu Geschlechtern oder gewissen Personen ist möglich [148, S. 29].

Ähnlich wie die bereits bei der Bildverarbeitung vorgestellten Methode zum Übertragen von Mimik gibt es auch Ansätze, die Stimme einer Person zu klonen und nachzubilden. Dank *Text-to-Speech* sogar nur anhand eines Textes, ohne gesprochenes Vorbild. Dieser Vorgang wird *voice cloning* genannt [12].

2.2.2.3. Autonome Systeme

Ein weiteres wichtiges Anwendungsgebiet für KI sind intelligente autonome Systeme.

Autonome Systeme sind selbstständig lernend und handelnde Maschinen, Roboter oder Softwaresysteme. Sie können auch für komplexe Aufgaben eingesetzt werden und auf unvorhersehbare Ereignisse reagieren. [49]

Beispiele hierfür sind autonome Fahrzeuge oder den Menschen flexibel unterstützende interaktive Produktionsroboter. Sie können auch dort selbsttätig handeln, wo es für den Menschen zu gefährlich ist. [49]

Erprobt werden derartige Systeme heutzutage unter anderem in Spielen. Auf KI basierende Weiterentwicklungen des klassischen Schachcomputers können heutzutage in deutlich komplexeren Spielen wie Go [163] oder dynamischen Computerspielen wie „Dota 2“ [22] menschliche Weltmeister unter bestimmten Voraussetzungen besiegen. Systeme, die mehrere verschiedene Spiele beherrschen und ihr Wissen spielübergreifend nutzen, gehören der Kategorie *General Game Playing* an [163, S. 6]. Die dahinterliegenden Algorithmen nutzen Methoden des *reinforcement learning* [163, S. 1].

Auch in der Chemie-Branche werden derartige Algorithmen bereits eingesetzt, beispielsweise zur Erforschung neuer Impfstoffe [120].

2.2.2.4. Data Science

Auch im Bereich traditioneller Datenauswertungen kann der Einsatz von KI-Algorithmen sehr hilfreich sein. Anwendungsfälle finden sich in zahlreichen Bereichen.

In Büros können viele im Bereich *business intelligence* anfallende Aufgaben wie beispielsweise die Vorhersage von Zeitreihen mithilfe von KI gelöst werden. Im Marketing können ML-Algorithmen beispielsweise dafür eingesetzt werden, um den Erfolg einer Kampagne auszuwerten. Im Handel kann die Vorhersage von Aktienkursen und Handelspreisen oder das Abschätzen geeigneter Verkaufspreise realisiert werden. Neuronale Netze können hier die Grundlage für umfangreiche Vorhersagemodelle sein, die bessere Ergebnisse liefern können als viele klassische statistische Methoden, wie die lineare Regression. [64, S. 239]

Im industriellen Umfeld hat der Begriff *predictive maintenance* seit einigen Jahren an Bedeutung gewonnen. Mithilfe von KI ist es möglich, im Voraus vorherzusagen, wann eine Maschine gewartet werden muss, sodass Produktionsausfälle vermieden werden können [150, S. 594].

Auch als Privatperson kommt man mit KI-Algorithmen in Berührung, beispielsweise beim Einkauf auf Websites die Empfehlungssysteme einsetzen. Diese Systeme schlagen Kunden für diesen relevante Produkte oder beliebte Produkte anderer Kunden vor, um gezielt zu werben. Im Handel können ML-Algorithmen auch zur Erkennung von Betrugsfällen genutzt werden, beispielsweise anhand einer Anomaliedetektion. [66, S. 478; 64, S. 12]

2.3. Synthetische Daten

Wenn bisher in dieser Arbeit ein Bezug zu Daten hergestellt wurde, so war stets von realen Daten die Rede, die aufgrund echter Ereignisse, wie beispielsweise direkten Messungen oder Videokameras entstanden sind. Darüber hinaus gibt es auch künstliche, häufig durch Algorithmen erzeugte Daten, die im Folgenden als synthetische Daten bezeichnet werden.

Hierbei kann es sich nicht nur um Bilder handeln, sondern auch um Zahlenreihen und Wertetabellen. Synthetische Daten können wie reale Daten für ein Training von neuronalen Netzen eingesetzt werden, aber auch in anderen Bereichen beispielsweise als Testdaten für neue Produkte, Werkzeuge oder Modelle.

Vorteile synthetischer Daten Der große Vorteil von synthetischen Daten ist, dass Situationen nachgebildet werden können, die so nie oder nur selten in der benötigten Vielfalt in der Realität stattfinden [176, S. 1082].

Das kann deutlich günstiger und weniger zeitaufwendiger sein als das händische Sammeln und Beschriften von Daten, gerade dann, wenn die Beschriftung Expertenwissen erfordert oder viele kleine Datenpunkte beschriftet werden sollen. Beispiele hierfür sind die Beschriftung von Bildern mit Segmentierungsmasken auf Pixelebene oder das Annotieren von Gelenkpositionen für 3D-Körperhaltungen [176, S. 1082].

Auch hinsichtlich des Datenschutzes eignen sich synthetische Daten besonders, da im Gegensatz zu anonymisierten realen Daten auch bei Fehlschlag der Anonymisierungssoftware kein Bezug zu realen Personen hergestellt werden kann, weder in künstlichen Wertetabellen (zum Beispiel Kundendaten), noch in generierten Videos [21, S. 1–2]. Die Verwendung synthetischer Daten umgeht zudem das Problem der stetigen Konkurrenz von Software zur Anonymisierung und Deanonymisierung [21, S. 50].

Erstellung synthetischer Daten In dieser Arbeit soll eine 3D-Simulation zur Erzeugung synthetischer Daten zur Handlungserkennung entwickelt werden, wodurch beispielsweise die Aufnahme von Handlungen mittels Schauspielern und dem manuellen Beschriften dieser Handlungen eingespart wird.

Selbstverständlich könnten auch andere Werkzeuge zur Generierung der Daten eingesetzt werden, beispielsweise das Filmen von Puppen oder Figuren, beispielsweise mit der Stop-Motion-Technik.

In den letzten Jahren wurden bereits einige synthetische Datensätze veröffentlicht und erfolgreich für maschinelles Lernen eingesetzt [30; 71; 121; 152; 17; 118].

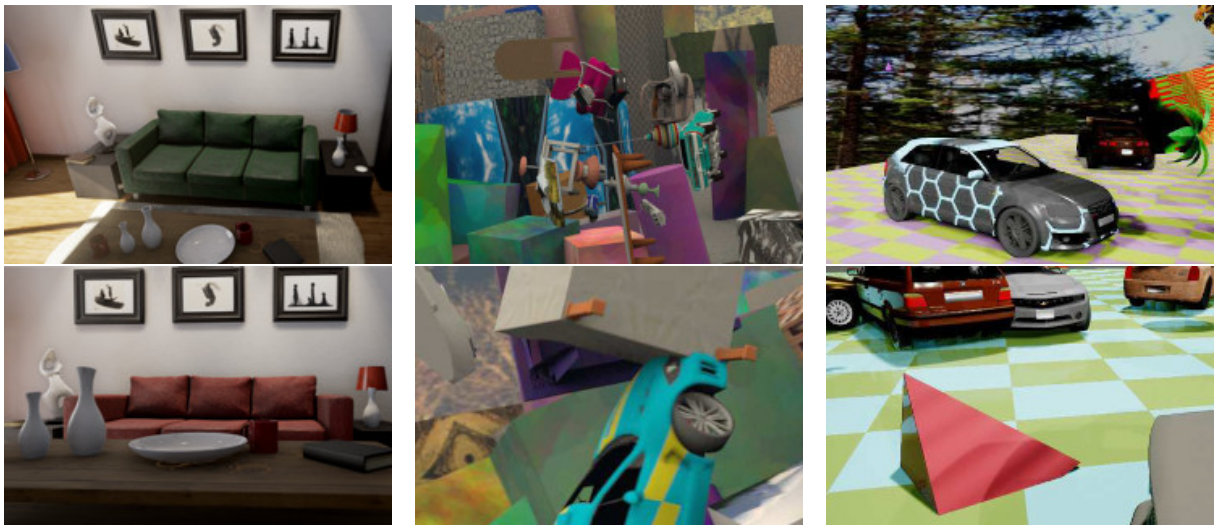
Unterschieden werden kann hier grundsätzlich zwischen voll-synthetischen und teil-synthetischen Ansätzen, also solchen, die rein mit synthetischen Daten trainieren und solchen, die einen Mix aus realen und synthetischen Daten für das Training verwenden.

Beim teil-synthetischen Ansatz handelt es sich um *Transfer Learning*, denn das Modell wird zunächst auf Basis synthetischer Daten gelernt und anschließend mithilfe von realen Daten auf den Anwendungsfall spezialisiert.

Eine Herausforderung hierbei ist das Bestimmen der „richtigen“ Menge synthetischer Daten zur Verhinderung von Überanpassung auf synthetische Daten statt realer Daten.

Realitätstreue synthetischer Daten Synthetische Daten können auf Basis verschiedener Ziele hinsichtlich der Genauigkeit und Realitätsnähe der Simulation generiert werden. Der naheliegendste Ansatz ist die Abbildung von fotorealistischen Szenen in der Simulation. Wie in Abb. 2.19a gezeigt können so Bilder generiert werden, die kaum von realen Fotos zu unterscheiden sind. Diese Methode hat den Nachteil, dass zur Erstellung der aufwendigen Szenen Künstler benötigt werden, die Zeit für die genaue Nachbildung der Realität benötigen [176, S. 1082].

Es gibt bereits einige Veröffentlichungen, in denen fotorealistische synthetische Daten erfolgreich für das Training von neuronalen Netzen genutzt wurden, beispielsweise für Netze zur Berechnung des optischen Flusses oder zur Erkennung von Körperhaltungen [176, S. 1082; 121, S. 4044].



(a) Ausgewählte Perspektiven und Texturen für eine fotorealistische Szene (aus [152, S. 5])

(b) Zufällig angeordnete Objekte schweben vor zufälligen Hintergründen (aus [121, S. 4043])

(c) Zufällig texturierte aber physikalisch realistisch platzierte Objekte (aus [176, S. 1085])

Abb. 2.19.: Beispiele synthetisch generierter Bilder zum Training einer Objektdetektion und -segmentierung.

Je realistischer die Szenen sein sollen, desto mehr Handarbeit fließt in deren Erstellung ein, was die Automatisierbarkeit und Skalierbarkeit der Simulatoren einschränkt.

Es wird daher auch ein sehr gegensätzlicher Ansatz zur fotorealistischen Nachbildung verfolgt: Bei der *Domain Randomization* Methode werden Objekte komplett zufällig positioniert, gedreht, skaliert und texturiert, sodass unendlich viele Möglichkeiten zur Datengenerierung gegeben sind [175; 176, S. 1084; 121, S. 4043]. In Abb. 2.19b werden beispielhafte Bilder einer solchen Simulation gezeigt, die dann je nach Verwendungszweck beispielsweise durch eine passende Segmentierungsmaske für jedes Einzelbild begleitet werden.

Auch Mischformen wie die in Abb. 2.19c gezeigte physikalisch realistische Platzierung von Objekten an zufälligen Positionen und Winkeln sowie mit zufälligen Modifikationen der Texturen sind möglich [176, S. 1085].

In einigen Fällen kann *Domain Randomization* zu einer deutlichen Verbesserung der Genauigkeit von ML-Modellen führen [175, S. 4–6]. Die Generierung der hohen Anzahl an Trainingsdaten und das anschließende Training benötigen allerdings eine leistungsfähigere Entwicklungsumgebung oder deutlich mehr Zeit als detailliertere Ansätze, die weniger, aber dafür hochwertigere Videos verwenden.

Die passende Wahl der Detailtreue ist daher eine anwendungsfallspezifische Herausforderung.

2.4. Handlungserkennung

Das Thema Handlungserkennung, auch bekannt unter dem englischen Begriff *action recognition*, ist ein aktives Forschungsgebiet im Bereich der KI.

Unter Handlungserkennung versteht man die zeitliche und räumliche Detektion und Lokalisierung menschlicher Handlungen oder die Klassifikation von Videos zu bestimmten Handlungen.

Methode Grundsätzlich basiert die Handlungserkennung daher auf der Klassifikation von Videos, wofür analog zu den Beispielbildern zur Bildklassifikation nun Beispielvideos als „Trainingsmaterial“ benötigt werden, die beispielsweise in der in Abb. 2.20 gezeigten Ordnerstruktur vorliegen.

Der Computer lernt auf Pixelebene die Videos der verschiedenen Klassen zu unterscheiden und je nach Detailgrad der Handlungserkennung zudem auch die Handlungen in langen Videos zeitlich (also von wann bis wann die Handlungen auftreten) sowie räumlich (also in welchem Bildbereich die Handlung stattfindet) voneinander zu trennen. Für diese Lokalisierung werden dann zusätzlich zu der gezeigten Ordnerstruktur noch Dateien, die diese Zusatzinformationen den Bildern zuordnen, verwendet.

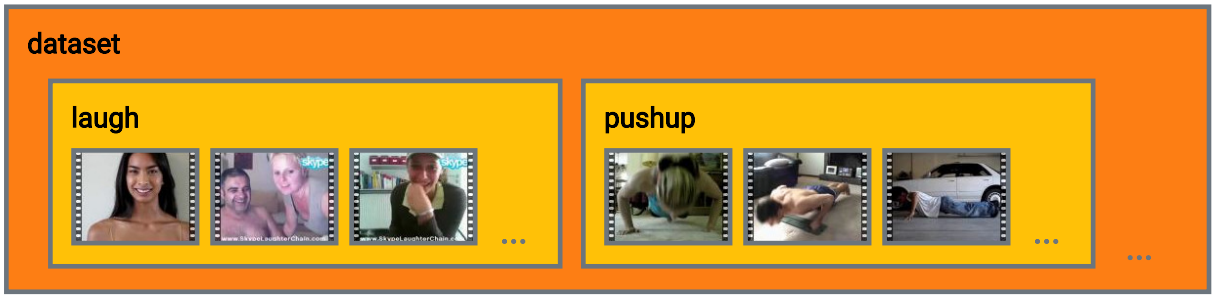


Abb. 2.20.: Ordnerstruktur eines Datensatzes zur Videoklassifizierung. Für das Beispiel wurden Videos und Klassen des *Human Motion Database* (*HMDB*) [99] Datensatzes verwendet.

Je nach gewünschtem Detailgrad kann die Annotation in verschiedenen Formaten vorliegen. Mithilfe der Ordnerstruktur sind Beschriftungen auf Video-Ebene realisierbar. Um hingegen die Existenz von Handlungen für jedes Einzelbild eines Videos zu annotieren, wird eine andere Methode benötigt.

Auch wenn mehrere Handlungen an verschiedenen Bildpositionen hintereinander oder zeitgleich ausgeführt werden, und diese korrekt voneinander getrennt werden sollen, wird ein komplexeres Format zur Annotation benötigt.

Dazu kommt, dass sich auch die Position einer Handlung im Bild im Verlauf der Zeit ändern kann, und auch die Grenzen zwischen Handlungen können sowohl zeitlich als auch räumlich fließend sein. Für letztere Fälle kann es Hilfreich sein, einen Leitfaden zur Annotation zu schreiben, der die korrekten Formulierungen für diese Spezialfälle definiert [78].

Feingranularere Datensätze setzen oft auf *JSON* oder *CSV*-Dateien zur Annotierung von Videos [78; 69]. Mit diesen Dateien können jedem Video oder sogar Einzelbild beliebig viele Informationen zugewiesen werden. Sollen beispielsweise Handlungen über mehrere Einzelbilder hinweg zugeordnet werden, so kommen teilweise relationale Datenbanken zum Einsatz [78].

Die Videos werden zudem aufgrund ihrer Dateigröße oft nicht direkt zum Herunterladen angeboten, sondern nur über Kennungen (*IDs*) der Videoplattform *YouTube* referenziert. Ein Beispiel für die Struktur der Annotationen eines solchen Datensatzes wird in Abb. 2.21 gezeigt.

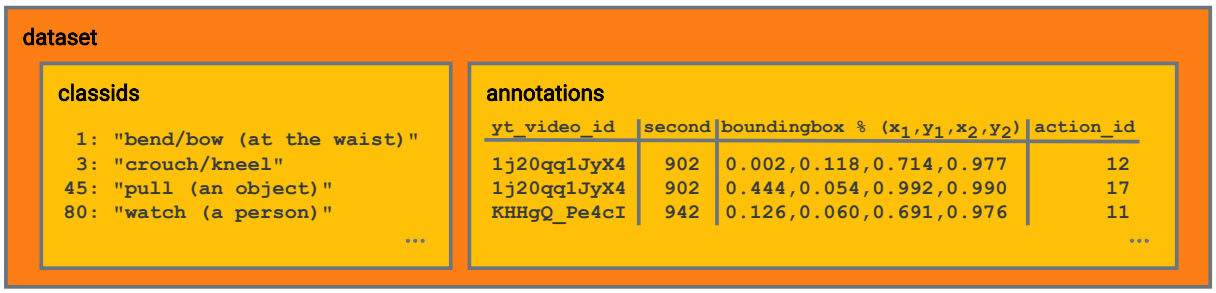


Abb. 2.21.: Struktur eines Datensatzes zur Lokalisierung von Handlungen. Im Beispiel werden die Annotationsdateien des *Atomic Visual Actions* (*AVA*) [69] Datensatzes gezeigt.

Eine Handlungserkennung kann nicht nur aufgrund visueller Sensoren wie einer Kamera erfolgen, sondern auch durch die Auswertung von am Körper getragener Sensoren [143] oder zum Beispiel einer Elektromyografie (EMG) Messung [204].

Im Folgenden werden nur die visuell-basierten Ansätze untersucht, da das Tragen von Sensoren für den in Abschnitt 1 beschriebenen Anwendungsfall der öffentlichen Plätze ungeeignet scheint.

Der Anwendungsfall dieser Arbeit erzeugt aufgrund der Kameraperspektive, der hohen Anzahl der Handlungen im Bild und der Art der Handlungen eine weitere Herausforderung. Viele bekannte Datensätze zur Erkennung von Handlungen zeigen die Handlungen aus frontaler Perspektive und bildfüllend. Einige verwenden auch Handlungs-Klassen, die für den Anwendungsfall irrelevant sind, beispielsweise in Innenbereichen aufgenommene Szenen, weshalb diese nicht ohne weiteres verwendet werden können.

Eine detaillierte Übersicht bestehender Datensätze wird in Abschnitt 3.2.1 erarbeitet.

Eine Methode zum isolierten Klassifizieren bildfüllender Handlungen ist das Zuschneiden des Videos in mehrere kleine Videos, die jeweils nur eine Person zeigen. Hierfür kann ein Algorithmus zur Personenerkennung verwendet werden, dessen *bounding box*-Ausgaben für den Zuschnitt verwendet werden können. So kann die Handlungserkennung anschließend auf diese zugeschnittenen Videos, die auch *bounding box tubes* oder *action tubes* genannt werden, angewendet werden [65].

Durch den Zuschnitt können also mehr Datensätze verwendet werden und es wird vermutlich auch weniger Rechenzeit zur Inferenz benötigt, da weniger Pixel ausgewertet werden müssen, allerdings gehen auch gegebenenfalls wichtige Zusatzinformationen wie die Reaktionen umstehender Personen verloren. Bei einer Schlägerei ist beispielsweise zu erwarten, das neben den sich prügelnden Personen auch zahlreiche Schaulustige oder verängstigte Personen ihr Verhalten ändern würden. Um den mangelnden Kontext-Informationen entgegenzuwirken, können statt einem exakten Zuschnitt auch jeweils eine bestimmte Anzahl Pixel pro Seite (ein *padding*) dem Zuschnitt hinzugefügt werden.

Ein Algorithmus, der das gesamte Bild auf einmal betrachtet, hat zudem den Vorteil, dass dessen Verarbeitungsdauer unabhängig von der Anzahl der Personen im Bild ist.

Die Auswahl eines geeigneten Zuschnitts sollte daher auf den Anwendungsfall abgestimmt werden.

Herausforderungen Neben der genannten Herausforderung hinsichtlich der Wahl eines geeigneten Zuschnitts gibt es zahlreiche weitere Herausforderungen, die die Handlungserkennung zu einem herausfordernden Forschungsgebiet machen.

Hierzu zählt beispielsweise, wie bei jedem Algorithmus, der mit Bildern von Menschen arbeitet, die Generalisierung hinsichtlich der anthropometrischen Diversität, also die Berücksichtigung verschiedener Körperbautypen und -proportionen. Darüber hinaus sollen Handlungen vor verschiedenen Hintergründen erkannt werden, auch wenn sich die Hintergründe dynamisch verändern oder überladen sind. Darüber hinaus kann es verschiedene Betrachtungswinkel und Videos mit schlechter Qualität geben, beispielsweise aufgrund einer geringen Auflösung oder Bildrate, Unschärfe oder Kompressions-Artefakten. [82, S. 2–8]

Darüber hinaus ist es bei Handlungen schwer, räumliche und zeitliche Grenzen zu definieren. Als Beispiel können verschiedene Fragestellungen für die Handlung „essen“ genannt werden:

- „Beginnt die Handlung ‚essen‘ erst beim Kauen der Nahrung oder schon bei Verwendung der Gabel?“
- „Sollte die Handlung granularer unterteilt werden, beispielsweise in ‚kauen‘ und ‚herunterschlucken‘?“
- „Sollten Gegenstände Teil der Handlung sein? Sind ‚eine Banane essen‘ und ‚einen Apfel essen‘ zwei verschiedene Handlungen?“

Aufgrund dieser Vielfalt innerhalb von Klassen und Ähnlichkeiten zwischen Klassen von Handlungen ist davon auszugehen, dass auch beschriftete Handlungen immer eine gewisse Ungenauigkeit besitzen. [82, S. 2–8]

Weitere Herausforderungen sind Verdeckungen, variierende Beleuchtungen, Schattenwürfe und Skalenvarianz, Bewegungen der Kamera, die Abbildung verschiedener Wetter-Zustände sowie oft unzureichende Datengrundlagen zum Trainieren der Modelle [82, S. 2–8].

Eine Lösung, um diesen Problemen gegenzuwirken ist die Verwendung größerer Datensätze, die genau diese Probleme beinhalten. So können beim Trainieren des Modells Merkmale gelernt werden, die robust gegenüber den oben genannten Herausforderungen sind.

In der später durchgeführten Datensatzrecherche werden auch einige Methoden beschrieben, mit denen die bestehenden Datensätzen mit diesen Herausforderungen umgehen.

3. Erarbeitung des Konzepts

Im Rahmen dieser Arbeit soll geprüft werden, ob die Hinzunahme synthetischer Daten zur Verbesserung eines Modells zur Handlungserkennung führen kann.

Um diese These zu prüfen, müssen daher synthetische Daten sowie eine Möglichkeit zur Handlungserkennung geschaffen werden. Darüber hinaus wird ein Referenz-Datensatz benötigt, der zum initialen Training der Handlungserkennung verwendet wird.

In diesem Kapitel werden für diese drei Teilaspekte mögliche Ansätze geprüft und miteinander verglichen, sodass für jeder am Ende des Kapitels für jeden der Aspekte eine passende Lösung vorliegt.

3.1. Auswahl einer Quelle für synthetische Daten

Synthetische Daten können, wie bereits in den Grundlagen in Abschnitt 2.3 erwähnt, auf verschiedene Arten erzeugt werden.

In diesem Abschnitt werden nun deutlich konkreter mögliche Ansätze zur Erstellung der synthetischen Daten für den Anwendungsfall dieser Arbeit vorgestellt und miteinander verglichen, sodass anschließend eine speziell für diese Arbeit geeignete Methode ausgewählt werden kann.

3.1.1. Vergleich kommerzieller Anbieter synthetischer Daten

Der Erhalt von synthetischen Daten ist nicht nur durch Generierung eigener synthetischer Daten möglich. In diesem Abschnitt werden daher für unseren Anwendungsfall relevante kommerzielle Anbieter zur Generierung von synthetischen Daten vorgestellt und miteinander verglichen.

Die Untersuchung des Markts ist ein wichtiger Teilaspekt der Methoden-Auswahl, da für ein wirtschaftliches Unternehmen vor der Entwicklung eigener Komponenten immer die *Make, Buy, Use* oder *Compose*?-Frage gestellt werden sollte: Für welche Komponenten macht es Sinn, diese einzukaufen, und für welche ist es effizienter, diese selber zu entwickeln [97, S. 156]?

Durch die Recherche im Rahmen dieser Arbeit konnten drei relevante Unternehmen bestimmt werden, die Produkte oder Dienstleistungen im Bereich der Erzeugung synthetischer Daten für Handlungserkennung anbieten. Im Folgenden werden jeweils ihr Angebot sowie verwendete Methoden und eventuelle Besonderheiten vorstellt.

SynCity SynCity ist laut Angabe des Herstellers CVIDIA ein „detailgetreuer Simulator für Machine Learning Projekte“, der genutzt werden kann, um *Computer Vision* Systeme anhand einer anpassbaren, fotorealistischen Simulation zu trainieren oder zu validieren. Die Umgebungen bestehen aus hochauflösenden Strukturen und unterstützen eine hohe Anzahl von Objekten, bieten ein großes Maß an Entropie sowie digitale Nachbearbeitung (*post processing*). Der Simulator kann eine Vielzahl von Annotationen ausgeben, darunter semantische Segmentierungsmasken, Rahmen (*bounding boxes*) in 2D oder 3D, Bilder des optischen Flusses oder Punktwolken sowie die Geschwindigkeit, Koordinaten oder andere Zustandsinformationen von Objekten. [43]

Um realistisch eine Vielzahl echter Kameras nachzubilden, wurden neben klassischen RGB Sensoren verschiedene andere wie thermale Sensoren, LiDAR, Infrarot oder Ultraschall Sensoren implementiert und auch Wetterbedingungen wie Wasser, Schnee oder Nebel auf der Linse werden ebenso wie andere Störungen miteinbezogen [43].

Aufbauend auf der Unity Engine (siehe Abschnitt 3.1.2) wurde eine eigene grafische Oberfläche entwickelt, mit der die Parameter der Simulation ohne tiefergehende technische Kenntniss verändert werden können [43].

Zur Erhöhung der Vielfalt der Umgebung werden Methoden zum Steigern der Unvorhersagbarkeit der Simulation eingesetzt, so gibt es beispielsweise einen Generator, der aus zufällig ausgewählten Komponenten 3D-Charaktere generiert, sodass Passanten verschiedene Körperformen und Haltungen besitzen. Für die Generierung der Umgebung wird ein API-basiertes System zur prozeduralen Platzierung von Objekten verwendet. [60]

ANYVERSE ANYVERSE ist ein Datengenerierungs-Dienst von NEXT LIMIT, einem Unternehmen für Simulationstechnik [137; 135]. Die synthetischen Daten sollen die Wahrnehmung autonomer Fahrzeuge verbessern, können aber auch für Industrieroboter oder smarte Kameras für Innenräume eingesetzt werden [135; 139]. Die Simulation generiert neben RGB Bildern mit Tiefeninformationen auch Rahmen (*bounding boxes*) in 2D sowie Instanz- und Semantische Segmentierungsmasken und bei Bedarf LiDAR Sensordaten sowie Zusatzinformationen über die Szene [138].

Auf Anfrage wurden von ANYVERSE 8.371 Einzelbilder und deren Annotationen zur Evaluierung der Qualität der Simulation bereitgestellt (siehe Abb. 3.1). Die Einzelbilder folgen nicht direkt aufeinander, sodass keine Evaluierung von der Qualität von Videos und beispielsweise Animationen möglich ist. Die Bildqualität scheint hochauflösend und fotorealistisch.

Abb. 3.1.: Beispielbilder aus der ANYVERSE Simulation, auf Anfrage zu Forschungszwecken bereitgestellt von ANYVERSE.

Ein Experiment von ANYVERSE hat gezeigt, das eine Objekterkennung in Bildern des Cityscapes Datensatzes mit einem auf 37.000 durch ANYVERSE generierte Bilder im Durchschnitt doppelt so hohe *IoU*-Werte (siehe Abschnitt 2.1) erreichen kann, als mit einem Modell, das mit den 270.000 Bildern des öffentlich verfügbaren SYNTHIA Datensatzes [153] trainiert ist [136].

Dies ist unter anderem auf die höhere Realitätstreue zurückzuführen, die beispielsweise Wetterbedingungen und Reflektionen berücksichtigt [136; 138]. Für die Bildgenerierung wird die hauseigene Software Maxwell Render verwendet.

AI.Reverie AI.Reverie stellt synthetische Daten zur Verbesserung und Beschleunigung von Deep Learning Algorithmen bereit. Mithilfe der Spieleentwicklungs-Plattform Unreal Engine werden virtuelle Welten erstellt, die zum Generieren neuer beschrifteter Bilddaten genutzt werden können. [7]

Hierzu können sowohl vorgefertigte Szenen von Innen- und Außenbereichen verwendet, eigene Szenen angefordert oder echte Orte anhand von Satellitenbildern und Höhenkarten nachgebildet werden [8].

Bei der Videogenerierung können verschiedene Wetterszenarien und Beleuchtungsmodi aktiviert werden. In den Szenen kommen animierte Personen, Tiere und Fahrzeuge vor, die sich anhand von Wegpunkten unter Berücksichtigung physikalischer Umstände bewegen und miteinander interagieren können [8].

Zur Aufnahme stehen neben RGB auch LiDAR- und Thermal-Sensoren zur Verfügung, wobei jeweils zwischen verschiedenen Perspektiven und Störfaktoren wie einem Bildrauschen gewählt werden kann [8]. Als Annotationsmethoden stehen 2D und 3D Bounding Boxes, Segmentierungsmasken, Tiefenmasken, Oberflächen-Normalen (Winkel von Objekten) und Posen (Skelettpositionen) zur Verfügung [8].

Ein integriertes Benchmarking Framework ermöglicht es, in Echtzeit die Verbesserung von Machine Learning Modellen durch die synthetischen Daten zu messen [8].

3.1.2. Vergleich von Simulatoren

Kommerzielle Lösungen haben oft den Nachteil, nicht so individuell anpassbar zu sein wie individuell neu entwickelte Produkte.

Um besondere Anforderungen hinsichtlich der Funktionalität oder des Anwendungsfalls exakt abzudecken, aber auch um Kosten zu sparen, können neben den kommerziell vermarkteten Lösungen auch direkt die unterliegenden Frameworks zur Simulation genutzt werden.

Zu den bekanntesten Frameworks zur Entwicklung von interaktiven 3D-Welten gehören die beiden *Game Engines* Unity und Unreal Engine [149; 41, S. 11]. Diese Laufzeit- und Entwicklungsumgebungen sind primär zur Entwicklung von Computerspielen gedacht, werden aber auch beispielsweise von Architekten zur Visualisierung von Gebäuden oder von Industrieunternehmen zur Simulation von Fabrikanlagen eingesetzt. [9]

Eine weitere Möglichkeit wäre die Verwendung und Modifizierung fertiger Computerspiele, wie beispielsweise Städtebau-Simulationen oder physikbasierter Sandbox-Spiele wie „Garry's Mod".

Im folgenden werden die verschiedenen Ansätze näher beschrieben.

Unity Unity ist eine 2005 vorgestellte Spiel-Engine. Sie hat die Branche nachhaltig geprägt, da sie als erste Plattform das Entwickeln für bis zu 15 Plattformen von einer Code-Basis ermöglichte, darunter verschiedene Betriebssysteme für PC und Smartphone sowie mobile und stationäre Konsolen. [13, S. 2]

Unity gilt als einsteigerfreundlich und hat eine große Community, die bereits zahlreiche Anleitungen veröffentlicht und Fragen beantwortet hat [41, S. 32–34]. Darüber hinaus gibt es für diese einen Marktplatz, auf dem 3D-Objekte, Texturen, Charaktere, Animationen und Code (im Folgenden Assets genannt) untereinander ausgetauscht oder zum Verkauf angeboten werden können. Durch das große Angebot an kostenlosen Assets können erste Prototypen schnell und ohne zusätzliche Kosten entwickelt werden.

Technisch gesehen deckt Unity mehrere Bereiche der 3D-Entwicklung ab. Es ist eine Grafik-Engine, eine Physik-Simulation und bietet umfangreiche *Scripting*-Möglichkeiten mit der Programmiersprache *C#*.

Eine Bearbeitung einzelner 3D-Objekte und derer Knotenpunkte und Kanten ist mit Unity nicht möglich, es gibt aber eine gute Schnittstelle zur Bearbeitung von 3D-Objekten mit Blender.

Die Kosten zur Lizensierung von Unity steigen mit dem durch die Spiele erzeugten Umsatz, weshalb sie sich hervorragend für Forschungsprojekte und Indie-Game Studios eignet.

Zu den populärsten mit Unity entwickelten Spielen gehören unter anderen „Angry Birds", „Call of Duty Mobile", „Pokémon Go", „Temple Run" und „Kerbal Space Program" [193].

Speziell für die Erzeugung von annotierten synthetischen Daten können das Perception Toolkit [181] und die *ML-Imagesynthesis* Bibliothek [178] verwendet werden.

Unreal Engine Die Unreal Engine wurde 1998 von Epic Games für die Spieleserie Unreal entwickelt und wird seitdem weiterentwickelt und in zahlreichen anderen Spielen verwendet, darunter die Shooter aus der „Tom Clancy's" Reihe, die Spiele der „BioShock", „Darksiders", „Borderlands" und „Batman Arkham" Serien, „Fortnite" sowie „PlayerUnknown's Battlegrounds (PUBG)" [194].

Vergleicht man die beiden Engines anhand ihrer Spiele, so fällt auf, dass die Unreal Engine weniger für *Indie Games* und Spiele für Mobilgeräte verwendet wird, sondern eher für grafisch aufwendige *Triple A* Spiele für PC und Spielekonsolen.

Die Unreal Engine bietet ebenfalls einen Marktplatz zum Handeln von Assets, dieser enthält laut Christopoulou und Xinogalos [41] aber weniger kostenlose Objekte.

Für die Erzeugung semantisch segmentierter synthetischer Daten kann das UnrealGT Framework eingesetzt werden [150].

Für Projekte mit Bruttoeinnahmen unter 1.000.000 US-Dollar kann die Engine kostenfrei genutzt werden. Nach überschreiten dieser Grenze fällt eine Gebühr von 5% des Umsatzes an. [54; 13, S. 3]

Test der Engines Um die Aussagen von Andrade [13], Christopoulou und Xinogalos [41] und Pluralsight [149] zu überprüfen, wurden beide Engines im Rahmen dieser Arbeit testweise installiert.

Unity bietet eine native Unterstützung und Installationsprogramme für Windows, Mac OS und Linux, während die Unreal Engine primär für Windows entwickelt wurde und unter Mac OS und Linux nur mithilfe eines Emulators wie *Wine* ausgeführt werden kann. Für Mac OS wird ein Installationsprogramm angeboten, dass die meisten Schritte für den Nutzer übernimmt, unter Linux muss der Quelltext aber manuell heruntergeladen und gebaut werden, was einige Konsolebefehle erfordert [53]. Die Installation der Unreal Engine unter Ubuntu war daher also deutlich umständlicher als die Installation von Unity. Auch während der Benutzung führte die Emulation zu sehr hohen Ladezeiten, Abstürzen und einem hohen Speicherplatzbedarf (etwa 80 GB, ohne zusätzliche Assets).

Der Marktplatz der Unreal Engine bietet jeden Monat mehrere sonst kostenpflichtige Assets für eine kurze Zeit kostenlos an. Der Download dieser erwies sich jedoch als sehr umständlich, da auch für das Programm zum Downloaden der Assets nur mithilfe eines Emulators installiert werden kann.

Die Unreal Engine verarbeitet sämtliche Assets im proprietären `.utasset` Format, während Unity ohne Umwandlung gängige Formate wie `.jpg`, `.fbx` und `.psd` unterstützt. Da `.utasset` Dateien ohne Export nicht in anderen Programmen geöffnet werden können und die Dateiendung allein keinen Aufschluss über den Inhalt der Datei gibt, ist die Verwaltung der Assets nur mithilfe des unter Ubuntu sehr langsamen Editors möglich. Aufgrund des hohen Zeitbedarfs jeder einzelnen Export-Aktion zur Überführung der einzelnen Assets im proprietären Format in gängige Formate, ist die Umwandlung ganzer Asset-Pakete des Marktplatzes, beispielsweise zur Verwendung dieser Assets in einer anderen Spiel-Engine, nicht effizient möglich und daher nur wenig zielführend.

Die Benutzeroberfläche des Unity-Editors wirkt etwas schlanker und moderner als die der Unreal Engine. Nach einer kurzen Einarbeitungszeit können aber beide Oberflächen ohne große Hürden genutzt werden.

Verwendung eines Computerspiels Statt die Simulation ähnlich wie ein neues Spiel zu entwickeln, können zur Erzeugung von Videos auch vorhandene Spiele genutzt und angepasst werden. Diese Methode hat den Vorteil, dass keine eigenen 3D-Welten, Charaktere und Animationen kreiert werden müssen, allerdings wird man auch in ebendieser Auswahl eingeschränkt. Der Anwendungsfall kann deshalb mit einem Computerspiel voraussichtlich nicht so exakt nachgebildet werden, wie mit einer eigenes entwickelten Simulation.

Einige wenige Spiele wie beispielsweise „Garry's Mod" bieten bewusst Schnittstellen für Modifikationen durch Entwickler an. Es gab bereits Versuche, dieses Spiel zur Datengenerierung zu nutzen [26, S. 279].

Bei einem Großteil der Spiele würde es aber nur schwer bis nicht möglich sein, Inhalte, Szenenwechsel und die Generierung von Annotationen zu automatisieren. In diesen Fällen würde der Versuch, Modifikationen am Spiel vorzunehmen eher einem aufwendigem „hacken" in das dafür nicht gedachte Spiele gleichen, als klassischer Programmierung. Es müsste auch aufgrund mangelhaft verfügbarer Dokumentation voraussichtlich viel durch Ausprobieren (*Try & Error*) ermittelt werden, was unter anderem aufgrund langer Ladezeiten durch viele Neustarts des Spiels sehr zeitintensiv werden könnte.

Eine Verwendung von Computerspielen als Simulator wird daher im Rahmen dieser Arbeit nicht weiter verfolgt.

3.1.3. Vergleich von Quellen für 3D-Modelle und Animationen

3D-Engines wie Unity und die Unreal Engine benötigen zum Bau der Szenen 3D-Modelle von beispielsweise Gebäuden, Charakteren und Fahrzeugen sowie Animationen der Handlungen. Diese *Assets* kann man mithilfe geeigneter Software selbst erstellen, in Online-Plattformen käuflich erwerben oder kostenlos herunterladen.

Die Wiederverwendung derartiger Komponenten hat gegenüber der Eigenentwicklung viele Vorteile wie beispielsweise weniger Entwicklungsaufwand und eine daher resultierende Zeitersparnis, weniger Einarbeitungszeit, bestehende Dokumentation sowie Support durch die Hersteller.

Die Auswahl geeigneter 3D-Modelle geschieht anhand von verschiedenen Kriterien wie Kosten, Lizenzen und der Wiederverwendbarkeit. Hochwertige universell nutzbare Objekte zeichnen sich unter anderem dadurch aus, dass diese modular aufgebaut und mit anderen Komponenten kompatibel sind.

Im Folgenden werden einige mögliche Quellen für sowohl statische 3D-Modelle wie Häuser, Fahrzeuge und dekorative Elemente (Brunnen, Bäume, ...) als auch animierte 3D-Modelle von Charakteren (Personen) beschrieben und bewertet.

Mixamo Mixamo [5] ist eine Plattform des Softwareherstellers Adobe. Auf ihr können 3D-Charaktere und zu diesen passende Animationen angezeigt, miteinander frei kombiniert und in verschiedenen Formaten heruntergeladen werden.

Die 121 angebotenen 3D-Charaktere haben eine hohe Qualität und Diversität, so gibt es passende Charaktere für realistische Zwecke, Cartoons, Fantasy oder Sci-Fi. Jeder Charakter ist vollständig texturiert und mit Gelenken versehen, so dass die Animationen direkt auf diesen genutzt werden können [5].

Die 2.484 Animationen wurden mithilfe von *Motion Capturing* und professionellen Schauspielern aufgenommen, sodass jeweils der komplette Körper berücksichtigt und animiert wurde [5].

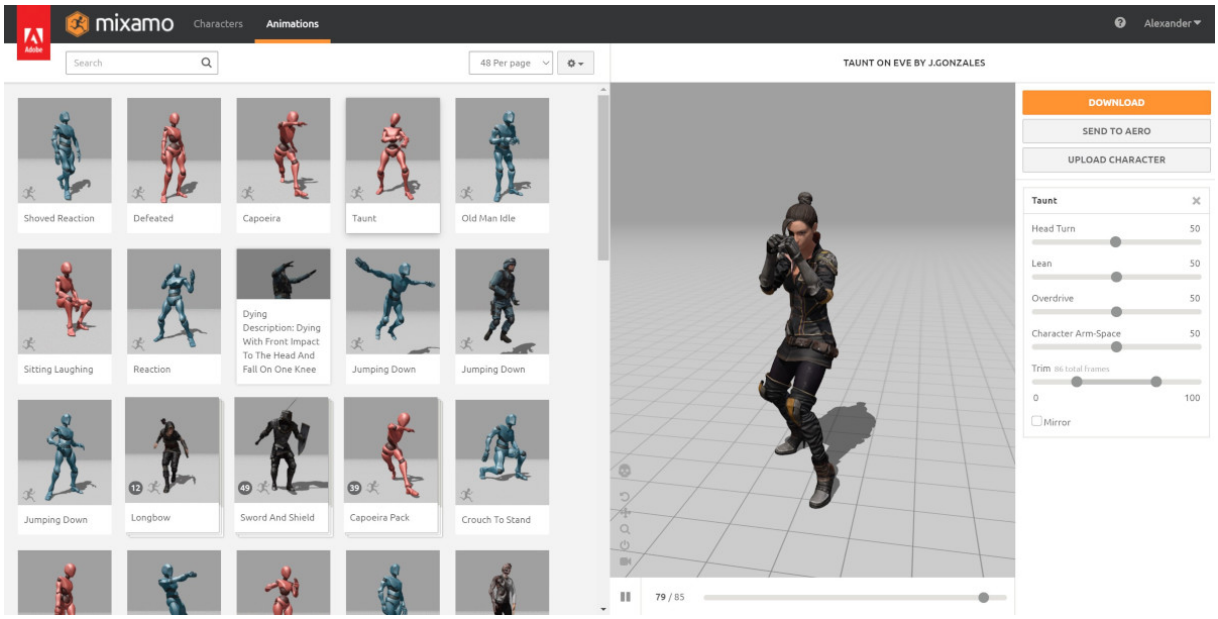


Abb. 3.2.: Benutzeroberfläche von Mixamo.

Zur Nutzung der Plattform ist eine Anmeldung mit einem kostenfreien Adobe-Konto erforderlich. In Abb. 3.2 wird die Oberfläche der Plattform nach der Anmeldung gezeigt. In der linken Hälfte wird eine Liste von entweder Charakteren oder Animationen angezeigt, rechts eine Live-Vorschau sowie die Möglichkeiten zur Anpassung und zum Herunterladen von Animation und Charakter. Jedes Vorschaubild in der Liste ist animiert und zeigt bereits die Bewegungen, beim Überfahren mit der Maus wird eine kurze Beschreibung der Animation eingeblendet. Die Suchfunktion bietet zudem die Möglichkeit zur Filterung der Animationen nach Genres.

Die Charaktere und Animationen von Mixamo können lizenzfrei für persönliche, kommerzielle und gemeinnützige Projekte verwendet werden [6].

Unity Asset Store Der Unity Asset Store ist ein Marktplatz für Pakete von 3D-Objekten, Materialien, Texturen und Skripten sowie Erweiterungen für den Editor.

Die Lizenzierung erfolgt je nach Asset entweder pro Person oder Projekt, wobei stets auch die kommerzielle Nutzung erlaubt ist [179]. Der Store umfasste am 06.09.2020 63.057 Elemente, darunter 6.248 kostenlose.

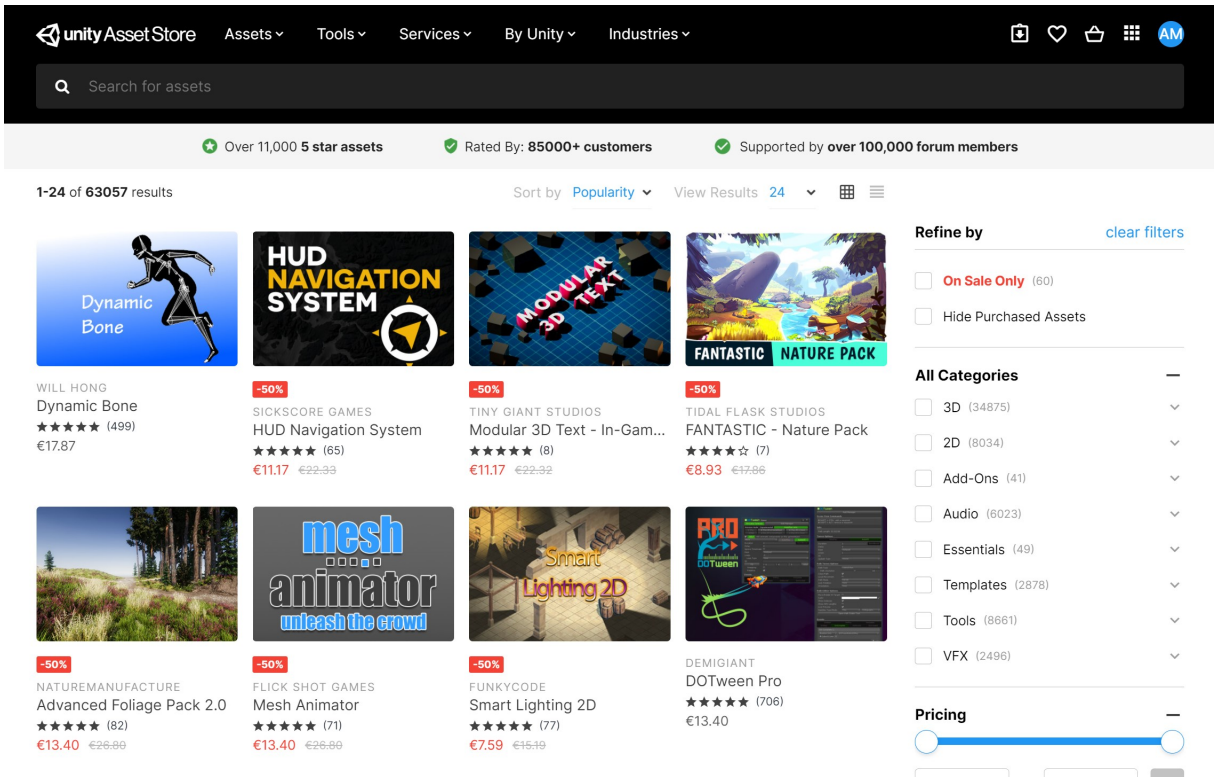


Abb. 3.3.: Website des Unity Asset Stores.

Wie in Abb. 3.3 gezeigt werden nicht nur 2D- und 3D-Objekte angeboten, sondern auch zahlreiche Skripte, die als Werkzeuge die Funktionalität der Unity Engine erweitern.

Unreal Engine Marketplace Auch der Marktplatz der Unreal Engine bietet verschiedene Assets zum Herunterladen an. Auf den ersten Blick fällt auf, dass es im Vergleich zum Unity Asset Store weniger dauerhaft kostenlose Produkte gibt, dafür werden aber im monatlichen Wechsel ausgewählte Produkte temporär kostenfrei angeboten.

Der Store umfasste am 06.09.2020 13.336 Elemente, darunter 420 kostenlose.

Der Download der Assets erfolgt im `.utasset`-Format, das mithilfe der Unreal Engine in auch in anderen Programmen nutzbare Formate umgewandelt werden kann. Dieser Vorgang ist vergleichsweise langsam und erfordert oft händische Korrekturarbeit zur Wiederherstellung von Verlinkungen, zum Beispiel zwischen 3D-Objekt und Textur. Die Verwendung der Assets der Unreal Engine in anderen Programmen ist daher nur umständlich möglich.

Für jedes Asset gibt es die gleiche Lizenz, die ein lebenslanges Nutzungsrecht für alle Produkte gewährt, die das Asset als Komponente nutzen.

3.1.4. Ergebnis: Unity und Mixamo

Die durchgeführten Vergleiche dienen der Erarbeitung eines Konzepts zum Erhalt von synthetischen Daten. Da die kommerziellen Lösungen nicht genügend Möglichkeiten zur Individualisierung bieten und Kosten gespart werden sollten, wird ein Ansatz zur Datengenerierung ausgewählt.

Aufgrund der oben genannten Vorteile von Unity bei der Asset-Verwaltung, der Installation, der Geschwindigkeit und der Oberfläche sowie der Vorerfahrung des Autors mit der Unity Engine wird entschieden, die Unity Engine zu verwenden.

Um für die Engine benötigte 3D-Objekte zu erhalten, sollen kostenlose Objekte aus dem „Unity Asset Store“ heruntergeladen werden. Für Animationen und Charaktere wird die Mixamo-Plattform genutzt.

Weitere Details zu den ausgewählten Lösungen werden im Rahmen der Implementierung beschrieben.

3.2. Auswahl eines Referenzdatensatzes

In diesem Abschnitt soll der in dieser Arbeit verwendete Datensatz zur Referenz-Messung ermittelt werden. Hierfür werden zahlreiche bestehende Datensätze analysiert und geprüft, ob die Aufnahme oder Zusammenstellung eines eigenen Datensatzes erforderlich ist.

3.2.1. Vergleich von Datensätzen

Um die Forschung im Bereich der Videoklassifikation und der Erkennung von Handlungen voranzutreiben, wurden in den letzten Jahren zahlreiche Datensätze veröffentlicht.

Da das Aufnehmen von Videos im Anwendungsfall aufgrund von Datenschutzbestimmungen besonders schwer ist und einige Handlungen, wie zum Beispiel Gewalt in öffentlich verfügbaren Videos wie Livestreams, besonders selten in der Realität auftreten, wird eine sehr ausführliche Datensatzrecherche durchgeführt.

So soll sichergestellt werden, dass für den Test der Verbesserung auch geeignete Datensätze als Referenz verwendet werden können.

In diesem Abschnitt werden einige der veröffentlichten Datensätze vorgestellt und miteinander verglichen. Grundsätzlich kann unterschieden werden zwischen mit Annotationen versehenen beschrifteten Datensätzen und unbeschrifteten Datensätzen.

3.2.1.1. Beschriftete Datensätze

Da im Rahmen dieser Arbeit Handlungen unterschieden werden sollen, ist ein Datensatz hierzu passend beschriftet, wenn die Beschriftung Informationen über die in den Videos vorkommenden Handlungen enthält.

Anhand einer Recherche konnten zahlreiche geeignete Datensätze ermittelt werden. In Tabelle 3.1 wird eine Übersicht über die verschiedenen passend beschrifteten Datensätze gegeben.

Um die Tabelle übersichtlich zu gestalten wurden einige Abkürzungen verwendet. Diese und die allgemeine Struktur der Tabelle werden im Folgenden erläutert.

Die Spalte „Datensatz“ enthält den Namen des jeweiligen Datensatzes sowie in Klammern die Quellenangabe, die zu weiteren Informationen und in vielen Fällen auch zur Website des Datensatzes führt. Datensätze mit fettgedruckten Namen werden in dieser Arbeit auch in anderen Kapiteln erwähnt.

Die Spalte „Persp.“ gibt an, aus welcher Perspektive die jeweiligen Videos aufgenommen wurden. Mögliche Optionen sind „Obersi.“, was Aufnahmen aus Obersicht bezeichnet. Ein Spezialfall dieser Perspektive ist der *Top-Shot*, bei dem aus einem exaktem 90°-Winkel von oben herab aufgenommen wird. Eine weitere gängige Perspektive ist die Seitenperspektive, die Personen ungefähr auf Kopfhöhe zeigt. Die Ego-Perspektive (*First Person*) kommt nur selten vor. Enthält ein Datensatz Videos verschiedener Perspektiven, so steht bei diesem der Vermerk „Versch.“.

Die Spalte „Videotyp“ bezeichnet die Art der Kanäle je Video. Während es sich bei den meisten Videos um RGB-Videos handelt, so gibt es auch Videos in Graustufen sowie Videos mit zusätzlichen Tiefeninformationen (*depth*, *+D*).

Datensatz	Persp.	Videotyp	Label	Datengrundlage	Jahr	# Kla.	# Videos
KTH [159]	Seite	Grau	V	Eigene Aufnahmen	2004	6	600
CAVIAR [61]	Obersi.	RGB	E,B	Eigene Aufnahmen	2005	18	40
IXMAS [188]	Seite	Grau	V,B	Eigene Aufnahmen	2006	11	1.148
CASIA Action [38]	Obersi.	RGB	V	Eigene Aufnahmen	2007	15	1.446
ETISEO [140]	Obersi.	RGB	E,B	Eigene Aufnahmen	2007	40	85
Weizmann [67]	Seite	RGB	V	Eigene Aufnahmen	2007	9	81
Kodak [114]	Versch.	RGB	E	Hobbyfilme	2007	25	1.358
Hollywood [104]	Versch.	RGB	V	Spielfilme	2008	8	430
UCF-ARG [39]	Obersi.	RGB	V	Eigene Aufnahmen	2008	10	1.442
Hollywood2 [119]	Versch.	RGB	V	Spielfilme	2009	12	1.787
MCG-WEBV [32]	Versch.	RGB	V	Hobbyfilme	2009	6	234.414
UCF-Aerial [40]	Obersi.	RGB	E,B	Eigene Aufnahmen	2009	9	7
UT-Interaction [157]	Obersi.	RGB	E,B,G	Eigene Aufnahmen	2010	6	20
BEHAVE [25]	Obersi.	RGB	E,B,G	Eigene Aufnahmen	2010	4	9
Olympic Sports [141]	Versch.	RGB	V	Sport- & Hobbyfilme	2010	16	800
VIRAT Ground [144]	Obersi.	RGB	E,B	Eigene Aufnahmen	2011	23	329
HMDB51 [99]	Versch.	RGB	V	Hobby- & Spielfilme	2011	51	6.849
CCV [85]	Versch.	RGB	V	Hobbyfilme	2011	20	9.317
ASLAN [94]	Versch.	RGB	E	Gem. Webvideos	2011	432	1.571
CAD-60 [171]	Seite	RGB+D	V,P	Eigene Aufnahmen	2012	12	60
UCF-101 [167]	Versch.	RGB	V	Hobbyfilme	2012	101	13.320
BIT-Interaction [95]	Seite	RGB	V	Eigene Aufnahmen	2012	8	400
CAD-120 [96]	Seite	RGB+D	V,P	Eigene Aufnahmen	2013	20*	120
JHMDB [84]	Versch.	RGB	V,P,O,S	HMDB51 [99]	2013	21	928
THUMOS [86; 75]	Versch.	RGB	V	Hobbyfilme	2014	101	18.394
LIRIS D2 [195]	Seite	RGB	E,B	Eigene Aufnahmen	2014	10	167
Sports-1M [88]	Versch.	RGB	V	Sport- & Hobbyfilme	2014	487	1.133.158
MPII Human Pose [14]	Versch.	RGB	T,Y	Sport- & Hobbyfilme	2014	491	3.913
HuPBA8K+ [158; 56]	Seite	RGB	E,P	Eigene Aufnahmen	2014	11	235
ActivityNet [58]	Versch.	RGB	E	Gem. Webvideos	2015	203	27.901
EventNet [201]	Versch.	RGB	E	Hobbyfilme	2015	500	95.321
FCVID [87]	Versch.	RGB	V	Gem. Webvideos	2015	239	91.223
MOBISERV-AIIA [80]	Seite	RGB	V	Eigene Aufnahmen	2015	13	96
NTU RGB+D [161]	Versch.	RGB+D	V,J	Eigene Aufnahmen	2016	60	56.880
PHAV [46]	Versch.	RGB	V	Comp.-3D-Simulat.	2016	35	39.982
DALY [189]	Versch.	RGB	E,B,P	Gem. Webvideos	2016	10	510
MERL Shopping [164]	Top-Sh.	RGB	E,B	Eigene Aufnahmen	2016	5	106
YouTube-8M [3]	Versch.	RGB	V	Gem. Webvideos	2016	4.800	8.000.000
Kinetics-400 [91]	Versch.	RGB	V	Gem. Webvideos	2017	400	306.245
Live Videos (LV) [106]	Versch.	RGB	V,B	Überw.-Kameras	2017	14	28
UCF-101-24 [167; 75; 165]	Versch.	RGB	E,B	Hobbyfilme	2017	24	3.207
Okutama Action [19]	Obersi.	RGB	E,B	Eigene Aufnahmen	2017	12	33
AVA [69]	Versch.	RGB	E,B	Gem. Webvideos	2018	80	430
Something-Sth. V2 [117; 68]	Versch.	RGB	V	Eigene Aufnahmen	2018	174	220.847
Kinetics-600 [34]	Versch.	RGB	V	Gem. Webvideos	2018	600	495.547
EGTEA Gaze+ [108]	Ego	RGB	E,G,L	Eigene Aufnahmen	2018	200*	86
Moments in Time [128]	Versch.	RGB	V	Gem. Webvideos	2019	339	1.020.000
Multi-Moments i.T. [129]	Versch.	RGB	V*,B*	Gem. Webvideos	2019	313	1.020.000
NTU RGB+D 120 [110]	Versch.	RGB+D	V,J	Eigene Aufnahmen	2019	120	114.480
Kinetics-700 [35]	Versch.	RGB	V	Gem. Webvideos	2019	700	650.317
Oops [55]	Versch.	RGB	V,T	Hobbyfilme	2019	2	20.723
HACS Clips [205]	Versch.	RGB	V	Gem. Webvideos	2019	200	1.500.000
HACS Segments [205]	Versch.	RGB	E	Gem. Webvideos	2019	200	50.000
MEVA [78]	Obersi.	RGB	E,B	Eigene Aufnahmen	2019	37	266
AVA-Kinetics [107]	Versch.	RGB	E,B	Kinetics-700 [35]	2020	80	238.906
SPHAR	Obersi.	RGB	V	siehe Tabelle 5.3	2020	15	7.759
S-SPHAR	Obersi.	RGB	E,S	Comp.-3D-Simulat.	2020	10	6.901

Tabelle 3.1.: Datensätze für Videoklassifikation und Handlungserkennung [87, S. 2; 166, S. 1111–1112; 124, S. 37; 25, S. 2; 195, S. 4].

Die Spalte „*Label*“ gibt die Genauigkeit der Annotationen an. *E* und *V* stehen für eine Beschriftung der Handlung auf Einzelbild- und respektive Videoebene. Der Spezialfall *V** kommt vor, wenn einem Video mehrere Handlungen zugeordnet sind, aber nicht angegeben ist, in welchen Einzelbildern diese jeweils auftreten.

Die Buchstaben *B* und *S* stehen in der Liste, wenn *bounding box* oder Segmentierungsmasken in jedem Einzelbild der Videos beschriftet sind. Der Spezialfall *B** gibt an, dass die *bounding boxes* nur für einen Teil der Einzelbilder annotiert wurden.

In seltenen Fällen wurden darüber hinaus weitere Daten annotiert, die in dieser Arbeit keine größere Rolle spielen. Dazu zählen die Körperhaltungen von vorkommenden Personen in jedem Einzelbild (*pose*, *P*) oder nur dem ersten Einzelbild der Aktion (*Y*), teilweise in 3D (*J*), der optische Fluss dieser in jedem Einzelbild (*O*), eine Beschriftung von Handlungen auf Videoebene mit Zeitstempel des Aktionsbeginns (*T*), die Blickrichtung der die Handlung ausführenden Person (*look*, *L*), eine Maske, die in jedem Bild die Hand der Person zeigt (*H*), eine Zuordnung von Personen-Gruppen (*G*).

Die Videos der verschiedenen Datensätze kommen aus verschiedenen Quellen, die in der Spalte „Datengrundlage“ angegeben werden. Bei einigen Videos handelt es sich um eigens für den Datensatz aufgenommene Videos, einige verwenden Videos professioneller Produktionen für Film und Fernsehen (Spielfilme) und wieder andere nutzen Amateuraufnahmen (Hobbyfilme) oder Rohbilder von Überwachungskameras. Viele der großen Datensätze setzen als Datengrundlage auf Videoplattformen wie YouTube. Diese „Gemischte Web-Videos“ können Sport-, Hobby- und Spielfilme sein. Wie im vorherigen Abschnitt beschrieben können Datensätze auch mithilfe einer durch Computer simulierten 3D-Umgebung aufgenommen worden sein. Auch diese sind entsprechend gekennzeichnet.

Die Spalte „Jahr“ gibt das Veröffentlichungsdatum der Datensätze oder derer begleitender Literatur an.

In der Spalte „# Kla.“ wird je Datensatz die Anzahl der verschiedenen unterschiedenen Klassen angegeben. Hierbei gibt es zwei mit Stern markierte Sonderfälle, die einer genaueren Beschreibung bedürfen: Im CAD-120 Datensatz [96] wurden zehn Aktivitäten wie beispielsweise *Essen zu sich nehmen* mit zehn Handlungen wie „etwas greifen“ oder „etwas trinken“ genauer unterteilt. Im EGTEA Gaze+ Datensatz [108] wurden 200 feingranulare Handlungen zusammengesetzt aus 19 Verben wie „nehmen“ und „waschen“ und 53 Nomen wie „Teller“ und „Pfanne“.

Die Datensätze in den untersten beiden Zeilen werden im weiteren Verlauf dieser Arbeit erstellt und erst später beschrieben. Zur besseren Vergleichbarkeit mit den anderen Datensätzen wurden sie dennoch bereits in die Tabelle aufgenommen.

Zusammenfassend lässt sich erkennen, dass nur ein kleiner Teil der Datensätze ausschließlich Videos aus der für den Anwendungsfall passenden Obersicht enthält. Die meisten Datensätze sind auf Video-Ebene beschriftet und bestehen aus RGB Videos. Hinsichtlich der Anzahl der Klassen und Videos gibt es sowohl Datensätze mit nur zwei Klassen oder sieben Videos als auch solche mit 4.800 Klassen und 8 Millionen Videos. Hiermit einhergehend gibt es auch große Unterschiede bei der Länge der Videos und der Qualität der Annotationen.

Die großen Datensätze wie YouTube-8M [3] verwenden Videos großer Internet-Plattformen, auf denen Nutzer Videos aller Arten hochladen können, während die meisten kleinen Datensätze die Videos in Laboren selbst aufzeichnen und so eine besonders hohe Qualität der Annotationen und Videos sicherstellen können.

3.2.1.2. Unbeschriftete Datensätze

Unabhängig von den bereits vorgestellten Datensätzen können auch für unseren Anwendungsfall zu schwach, unpassend oder gar nicht beschriftete Video-Quellen bei Bedarf selber annotiert werden. In Tabelle 3.2 werden geeignete Videoquellen zur Erhebung neuer selbst-beschrifteter Datensätze gezeigt.

Datensatz	Persp.	Jahr	Datengrundlage	Umfang
PETS Left-Luggage [174]	Obersi.	2006	Eigene Aufnahmen	25 Videos
UCF-Crowds [10]	Obersi.	2007	TV-Beiträge & SV	38 Videos
Dam Platz Amsterdam [182]	Versch.	2010-heute	Eigene Aufnahmen	Kontin. Livestream
3DPeS [18]	Obersi.	2011	Eigene Aufnahmen	8 Szenen, 500 Videos
Jufferenstraat Elburg [183]	Versch.	2012-heute	Eigene Aufnahmen	Kontin. Livestream
Marktplatz Bredstedt [83]	Obersi.	2013-heute	Eigene Aufnahmen	Kontin. Livestream
Avenue Anomaly [115]	Versch.	2013	Eigene Aufnahmen	37 Videos
Brunnen Bad Wildungen [16]	Versch.	2014-heute	Eigene Aufnahmen	Kontin. Livestream
Violent Scenes [48]	Versch.	2014	Spielfilme	Gewaltszenen aus 32 Spielfilmen
Piazza Comune Assisi [177]	Obersi.	2014-heute	Eigene Aufnahmen	Kontin. Livestream
Marktplatz Biberach [168]	Obersi.	2015-heute	Eigene Aufnahmen	Kontin. Livestream
Surveillance Videos [173]	Obersi.	2016	Eigene Aufnahmen	7*24 Stunden, ein Treppenhaus
Zakopane Krupówki [111]	Obersi.	2017-heute	Eigene Aufnahmen	Kontin. Livestream
Schlossplatz Warschau [112]	Obersi.	2018-heute	Eigene Aufnahmen	Kontin. Livestream
Marktplatz Einbeck [72]	Obersi.	2018-heute	Eigene Aufnahmen	Kontin. Livestream
Ruzzini Palace Venedig [156]	Obersi.	2019-heute	Eigene Aufnahmen	Kontin. Livestream

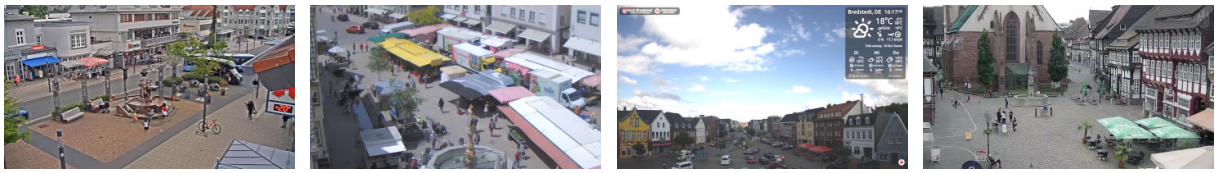
Tabelle 3.2.: Unbeschriftete Videos für den Anwendungsfall [124, S. 35]. Es handelt sich in allen Fällen um RGB-Videos. Kontin. = Kontinuierlicher, SV = Stock-Videos

Während es sich bei einigen Datensätzen um abgeschlossene Pakete von Videos handelt, die jederzeit heruntergeladen werden können, so gibt es auch zahlreiche, kontinuierlich neue Video-Daten generierende Livestreams.

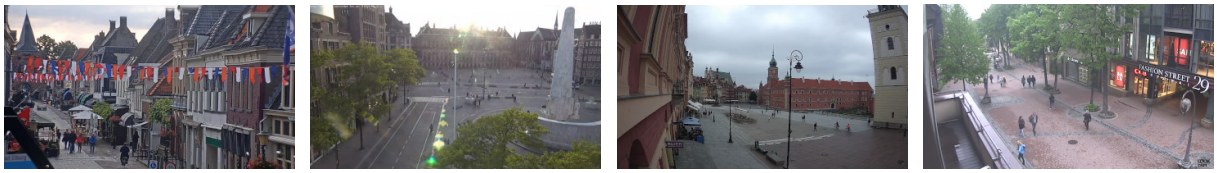
Die Livestreams haben die Vorteile, dass sie rund um die Uhr neues Datenmaterial produzieren und das Videomaterial sehr aktuell ist, und daher auch tagesaktuelle Änderungen in der Gesellschaft, wie beispielsweise die durch *COVID-19* hervorgerufenen erhöhten Abstände zwischen Personen oder das Tragen von Mund- und Nasenbedeckungen berücksichtigt sind [103, S. 2]. Durch die kontinuierliche Aufnahme werden zudem auch Anomalien wie ein Wochenmarkt oder sonstige Veranstaltungen aufgezeichnet.

Ein Nachteil eines Teils mancher Livestreams ist allerdings, dass je nach Stream keine historischen Daten abgerufen werden können, weshalb ein Download dort nur in Form eines Mitschnitts möglich ist, und daher die Dauer des gewünschten Videomaterials der in Anspruch genommenen Aufnahmezeit entspricht.

Zur bildlichen Veranschaulichung dieser unbeschrifteten Datensätze wird für diese in Abb. 3.4 je ein Vorschaubild gezeigt.



(a) Bad Wildungen [16] (b) Biberach [168] (c) Bredstedt [83] (d) Einbeck [72]



(e) Elburg [183] (f) Amsterdam [182] (g) Warschau [112] (h) Zakopane [111]

(i) Surv. Videos [173] (j) Violent Scenes [48] (k) Avenue Anomaly [115] (l) 3DPeS [18]

(m) PETS [174] (n) Assisi [177] (o) Venedig [156] (p) UCF-Crowds [10]

Abb. 3.4.: Beispielbilder der unbeschrifteten Datensätze.

3.2.2. Ergebnis: *HMDB* und *SPHAR* Datensatz

Der *HMDB* Datensatz [99] hat eine gute Auswahl an Klassen und eine gute Größe, da die Anzahl der Videos für ein Training ausreicht, aber dennoch nicht zu viel Speicherplatz oder Rechenkapazität benötigen. Die Kamera-Perspektive passt hingegen nicht zum Anwendungsfall.

Die beschrifteten Datensätze, die gut zu dem Anwendungsfall passen, haben je nur wenige passende Klassen oder Videos. Daher soll im Rahmen dieser Arbeit zudem ein neuer Datensatz mit relevanten Klassen und Videos aus passenden Perspektiven zusammengestellt werden. Im Folgenden wird dieser *Surveillance-Perspective Human Action Recognition* (*SPHAR*) Datensatz genannt. Hierzu werden keine eigenen Videos aufgenommen, sondern Videos aus den bereits existierenden Datensätzen ausgewählt, in ein einheitliches Format konvertiert und Annotationen vereinheitlich.

Bei der Vorbereitung der Handlungserkennung in Abschnitt 5.1.3 werden weitere Details zu den ausgewählten Datensätzen vorgestellt sowie eine detailliertere Begründung der Auswahl der Datensätze vorgenommen.

Es wurde keiner der unbeschrifteten Datensätze ausgewählt, da diese nicht ausreichend viele relevante Handlungen enthalten und die Beschriftung dieser im Rahmen dieser Arbeit sehr aufwendig wäre und zu viel Zeit in Anspruch nehmen würde.

3.3. Auswahl einer Handlungserkennung

Als Werkzeug zur Messung der Genauigkeitssteigerung durch die Hinzunahme von synthetischen Daten wird ein Algorithmus zur Erkennung von Handlungen eingesetzt. Dieser kann entweder auf Basis eines Open Source Frameworks selber programmiert werden oder es kann eine passende kommerzielle Lösungen eingesetzt werden.

In diesem Abschnitt werden beide Varianten vorgestellt und die verschiedenen Optionen miteinander verglichen. Abschließend wird ein Ansatz ausgewählt, der im Folgenden eingesetzt wird.

3.3.1. Vergleich kommerzieller Anbieter zur Handlungserkennung

In diesem Abschnitt werden die kommerziellen Lösungen zur Handlungserkennung untersucht.

Scylla Scylla ist ein System zur Erkennung von Waffen, Ladendiebstählen, Einbrechern und anderen Gefahren. Die eingesetzte Software zur Verhaltensanalyse kann zudem auch Kämpfe, Taschendiebe, Vandalismus und Aufstände erkennen [160]. Das ursprünglich in Deutschland gegründete Start-up ist vor allem in den USA sehr beliebt, da es dort zu mehr Vorfällen mit Waffen kommt [116].

Das System wurde mithilfe zahlreicher Videos von Anomalie-Ereignissen trainiert, sodass kein vor-Ort-Training notwendig ist. Die Inferenz verläuft laut Hersteller so schnell, dass mithilfe einer einzelnen Grafikkarte mehrere Video-Streams in Echtzeit ausgewertet werden können. [160]

Die aufgenommen Videos werden hierbei vollständig lokal verarbeitet, weshalb der Hersteller auch mit Datenschutzkonformität wirbt [160].

Viisights Viisights ist ein israelisches Start-up, das sich auf Systeme zur Verhaltens-Analyse in Echtzeit-Videos spezialisiert hat. Der Hersteller betont, das zur Erkennung von Handlungen Sequenzen ausgewertet werden, und nicht nur Objekte in Einzelbildern erkannt und gedeutet werden. [185]

Dieses Produkt *viisights wise* kann in einer privaten Cloud oder lokal installiert werden und Menschenmassen, Gruppen oder Gewalt analysieren, verdächtige Aktivitäten erkennen, Verkehr überwachen und Statistiken erheben.

Die Alarmmeldungen zu den erkannten Ereignissen sowie kurze Videoausschnitte der Ereignisse können mithilfe der Oberfläche *viisights true* übersichtlich auf einer Zeitachse angesehen werden. Durch eine Klassifikation von Fehlalarmen können diese automatisch ausgefiltert werden.

Weitere Anbieter Im Rahmen der Recherche konnten zahlreiche weitere Anbieter gefunden werden, die der Erkennung von Gefahren dienen. Diese werden nicht näher beschrieben, da sie auf reiner Objekt-Detektion in Einzelbildern basieren und daher keine vollwertige Alternative zur Handlungserkennung auf Basis von Videostreams sind.

Die Lösungen von Trueface [1], Camect [31], CVEDIA [44], Cawamo [37] und Defendry [47] sowie die Lösungen von Vintra [186], Monitoreal [130] und ZeroEyes [203] erkennen in Einzelbildern von

Videos das Vorhandensein von Personen, Waffen oder Masken, die als Indiz für eine gefährliche Handlung gesehen werden können.

Neuromation [134] und die Video Expert Group [187] bieten universelle Plattformen zur Auswertung von Bilddaten mithilfe von KI an. Sogenannte *AI-Powered Cameras* sollen Objekte detektieren und klassifizieren. Es werden die für KI-basierte Bildverarbeitung typischen Anwendungsfälle genannt (siehe Abschnitt 2.2.2.1), allerdings wird nicht weiter auf die konkrete Produkte und die Genauigkeiten eingegangen.

3.3.2. Vergleich von Open Source Frameworks zur Handlungserkennung

Wie bereits einleitend erwähnt, kann auch ein eigener Algorithmus zur Handlungserkennung programmiert werden. Um nicht ganz „bei Null“ anfangen zu müssen können Frameworks eingesetzt werden, die gängige Funktionen bereitstellen und so die Programmierung vereinfachen und beschleunigen können, sowie die Entstehung gängiger Fehlern verhindern können [127].

Hierbei kann unterschieden werden zwischen Frameworks, die allgemein für Programmierung im Bereich ML entwickelt wurden, und solcher, die speziell die Klassifikation von Videos abdecken. Während die allgemeineren Frameworks dabei unterstützen, neuronale Netze durch Code abzubilden, so bauen die spezielleren in der Regel auf den allgemeineren Frameworks auf und bieten beispielsweise die Möglichkeit, verschiedene vorgefertigte Netzarchitekturen direkt zu verwenden und bringen bereits Funktionen mit, wie beispielsweise das Speichern von Zwischenständen beim Trainieren von Modellen oder die Verwendung von Datensatz-Splits.

TensorFlow TensorFlow [2] ist ein 2015 von Google veröffentlichtes Framework zur Formulierung, Implementierung und Ausführung von Algorithmen für maschinelles Lernen. Es bietet die Möglichkeit zur Entwicklung von Software für ein breites Spektrum an Anwendungsfällen, da der Fokus einer Architektur nicht nur auf handelsübliche Desktop-PCs gesetzt werden kann, sondern auch auf große Rechenzentren-Cluster oder Mobilgeräte. [2, S. 1]

TensorFlow repräsentiert die neuronalen Netze durch Konstruktion eines gerichteten Datenstroms, der auf Teilmengen der Daten wiederholt angewendet wird. Dieser Ansatz bietet eine gute Sichtbarkeit der Berechnungen, die Definition von Netzen als Graph erfordert jedoch etwas Einarbeitung [2, S. 2; 146, S. 2].

PyTorch PyTorch [146] ist eine 2016 von Facebook veröffentlichte Python-Bibliothek auf Basis des in der Sprache Lua geschriebenen Torch Frameworks. PyTorch verfolgt das Ziel, im Vergleich zu beispielsweise TensorFlow eine ebenso hohe Geschwindigkeit, aber höhere Benutzerfreundlichkeit zu bieten. [146, S. 1]

Basis Zahlreicher Operationen in PyTorch ist die besonders effiziente und benutzerfreundliche automatische Berechnung von Ableitungen [147].

SlowFast SlowFast [59] ist ein 2020 vom „Facebook AI Research Lab“ veröffentlichtes quelloffenes Framework zur Klassifikation und Detektion von Videos auf Basis von *PyTorch*.

Es stellt eine performante und leichtgewichtige Schnittstelle für Implementationen verschiedener Methoden und Netz-Architekturen bereit und wurde mit dem Ziel entwickelt, die Entwicklung von Prototypen zum Test neuer Ideen zu unterstützen.

ClassyVision Ebenfalls von Facebook entwickelt wurde das ClassyVision [4] Framework. Es basiert auch auf PyTorch und dient der Klassifikation von Bildern und Videos. Die Klassifikation von Videos erfolgt allerdings nur anhand der Berechnung des Durchschnitts der Ergebnisse von Bildklassifikationen aller Einzelbilder [4]. Zeitliche Verläufe und Bewegungsrichtungen können so nicht vollständig erfasst werden, weshalb das Framework für den Anwendungsfall dieser Arbeit als ungeeignet erachtet wird.

ViP Das Framework ViP [62] der Universität von Michigan ermöglicht die Erkennung von Handlungen und die Detektion von Objekten in Videos. Es basiert auf PyTorch und bietet verschiedene Modelle und Vorkonfigurationen für verschiedene Datensätze an.

Ebenfalls untersucht wurde das auf Tensorflow basierende Framework M-PACT [74] derselben Autoren, dieses nutzt allerdings Python 2 und eine alte TensorFlow-Version, die auch nur mit veralteten Grafikkarten-Treibern funktioniert, weshalb es nicht näher betrachtet wird.

3.3.3. Ergebnis: SlowFast Framework

TensorFlow [2] und PyTorch [146] bieten in Vergleich zu den anderen Frameworks höhere Individualisierungs- und Performance-Möglichkeiten, die Programmierung einer eigenen Handlungserkennung auf Basis dieser Frameworks ist aber deutlich aufwendiger als die Verwendung eines der Frameworks wie SlowFast [59], M-PACT [74], ViP [62] oder ClassyVision [4].

Die kommerziellen Lösungen klingen teilweise sehr vielversprechend, gerade Viisights [185] und Scylla [160] können laut Angabe des Herstellers auch Handlungen im gesuchten Anwendungsfall klassifizieren. Leider geben beide Hersteller keine Preise an Außerdem ist unklar, ob zum Training eigener Modelle eigene Trainingsdaten verwendet werden können oder ob lediglich die Modelle des Herstellers verwendet werden können.

Für die Evaluierung im Rahmen dieser Arbeit macht es daher Sinn, eine Lösung mit ausreichend Freiräumen und der Möglichkeit zur Verwendung eigener Trainingsdaten einzusetzen. Es wurde entschieden, das SlowFast Framework [59] zu verwenden, da dieses für den Anwendungsfall gute Architekturen und Beispieldatensätze anbietet, einfach erweiterbar ist und aktiv weiterentwickelt wird.

Eine detaillierte Beschreibung des SlowFast Framewoks [59] erfolgt im Rahmen der Implementierung in Abschnitt 5.1.2.

3.4. Erarbeitetes Konzept

Anhand der in den vorangehenden Abschnitten durchgeführten Recherche wurde das in der folgenden Abb. 3.5 gezeigte Gesamtkonzept erarbeitet.

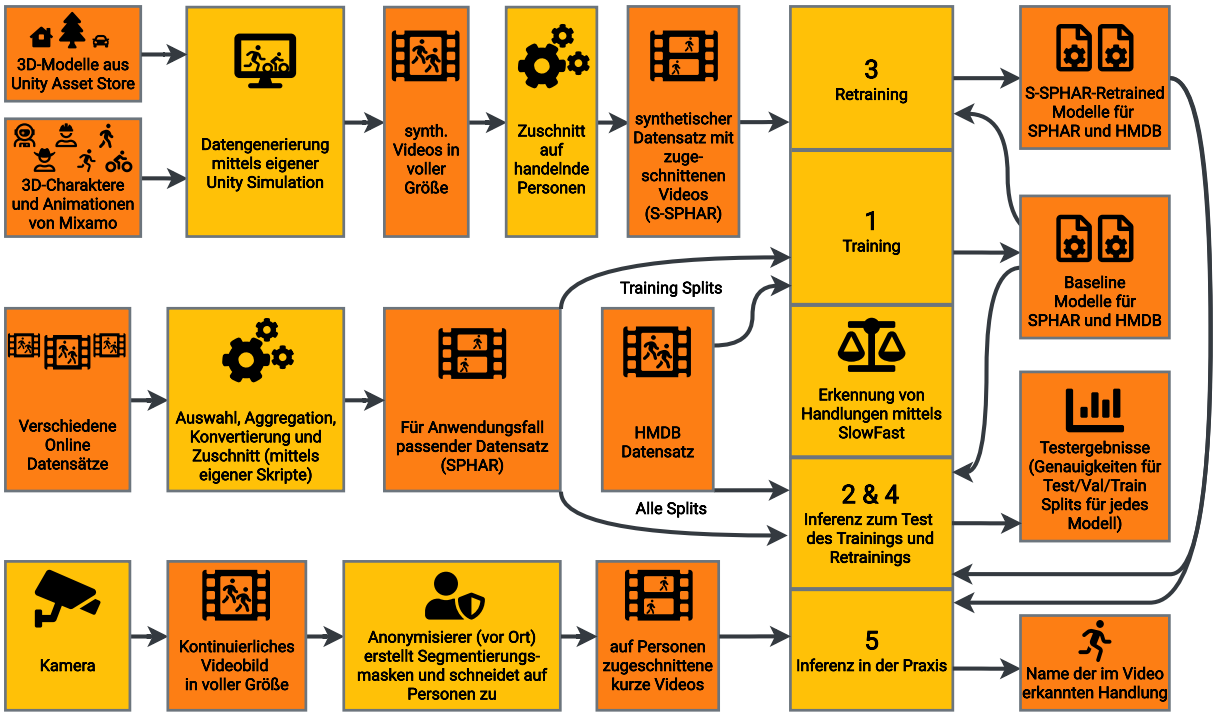


Abb. 3.5.: Erarbeitetes Konzept zur Überprüfung der These.

Als Datengenerierungsmethode wird eine eigens in Unity entwickelte Simulation verwendet, wobei Charaktere und Animationen von Mixamo sowie 3D-Modelle aus dem Unity Asset Store genutzt werden. Als Referenzdatensatz wird sowohl *HMDB* [99] als auch der im Rahmen dieser Arbeit bereits erarbeitete Datensatz *SPHAR* [126] genutzt. Als Framework wird SlowFast [59] eingesetzt. Sämtliche Videos werden vor der Klassifikation zugeschnitten, sodass eine Klassifizierung auf Video-Ebene erfolgen kann.

In den folgenden Kapiteln wird die Implementierung der Simulation, der Handlungserkennung und der erwähnten zusätzlichen Helfer-Skripte dokumentiert, sodass anschließend Modelle trainiert („1“ und „3“ in Abb. 3.5) und vor und nach dem Retraining getestet werden können („2“ und „4“ in Abb. 3.5).

Der in Abb. 3.5 gezeigte unterste Pfad „5“ : „Inferenz in der Praxis“ dient lediglich der Visualisierung des Anwendungsfalls und ist kein Teil dieser Arbeit.

4. Implementierung der Simulation

In diesem Kapitel wird die Erstellung und Programmierung der Simulation mit Unity beschrieben.

Die Simulation soll zur Erstellung von Videos von Handlungen auf öffentlichen Plätzen genutzt werden können. Es soll zudem die Möglichkeit geschaffen werden, so generierte Videos automatisch anhand der darin vorkommenden Handlungen zu beschriften.

Um eine hohe Qualität der generierten Videos zu gewährleisten, sollen Qualitätsparameter der Engine wie Auflösung und Schattenqualität frei eingestellt werden können und zahlreiche Möglichkeiten zur Erhöhung der Realitätsnähe und Diversität der Simulation implementiert werden.

Dazu kann das Verwenden verschiedener Modelle und Texturen für Personen, Fahrzeuge und Gebäude gehören, aber auch die Implementierung von Wetterbedingungen, Tieren, sonstige Verdeckungen der Sicht oder unvorhergesehene Ereignisse.

Der am Ende der Implementierung erzeugte synthetische Datensatz wird in Anlehnung an den ebenfalls im Rahmen dieser Arbeit erstellten *SPHAR* Datensatz *S-SPHAR* Datensatz (*Synthetic Surveillance-Perspective Human Action Recognition Dataset*) genannt.

In den folgenden Abschnitten wird die konkrete Umsetzung des Konzepts zur Implementierung der Simulation beschrieben. Hierfür werden zunächst die modellierten Szenen und anschließend die Programmierung beziehungsweise Umsetzung der einzelnen Komponenten und Funktionen der Simulation beschrieben. Anschließend werden die speziell zur Erhöherung der Diversität der Szenen eingesetzten Methoden vorgestellt. Zum Abschluss dieses Kapitels werden die zur Beschriftung der Videos verwendeten Methoden und Skripte beschrieben.

4.1. Grundlegende Überlegungen

In dieser Arbeit wird ein teil-synthetischer Ansatz verfolgt, das Modell soll also zunächst auf Basis synthetischer Daten gelernt und anschließend mithilfe von realen Daten auf den Anwendungsfall spezialisiert werden.

Die synthetischen Daten sollen nicht nach dem *Domain Randomization* Verfahren in hoher Anzahl und Zufälligkeit erzeugt werden (siehe Abschnitt 2.3), sondern mit vergleichsweise fotorealistischen Details, da die für die hohe Stückzahl benötigte Rechenpower und Zeit, die für das Generieren und Trainieren benötigt wird, im Rahmen dieser Arbeit nicht zur Verfügung steht.

4.2. Einarbeitung in Unity

Im Folgenden werden einige Details zum Unity Editor beschrieben, um ein grundlegendes Verständnis über dessen Funktionalität zu schaffen.

Oberfläche des Unity Editors Das Fenster des Unity Editors kann in der Standardeinstellung in vier Bereiche unterteilt werden (siehe Abb. 4.1). Im linken Bereich des Editors befindet sich eine hierarchisch angeordnete Liste aller Objekte in der Szene. Diese Objekte werden *Game Objects* genannt und können beispielsweise 3D-Modelle, Lichtquellen oder Partikelsysteme sein. Auch die virtuelle Kamera sowie Elemente der grafischen Benutzeroberflächen im Spiel (beispielsweise für Menüs) werden hier angezeigt.

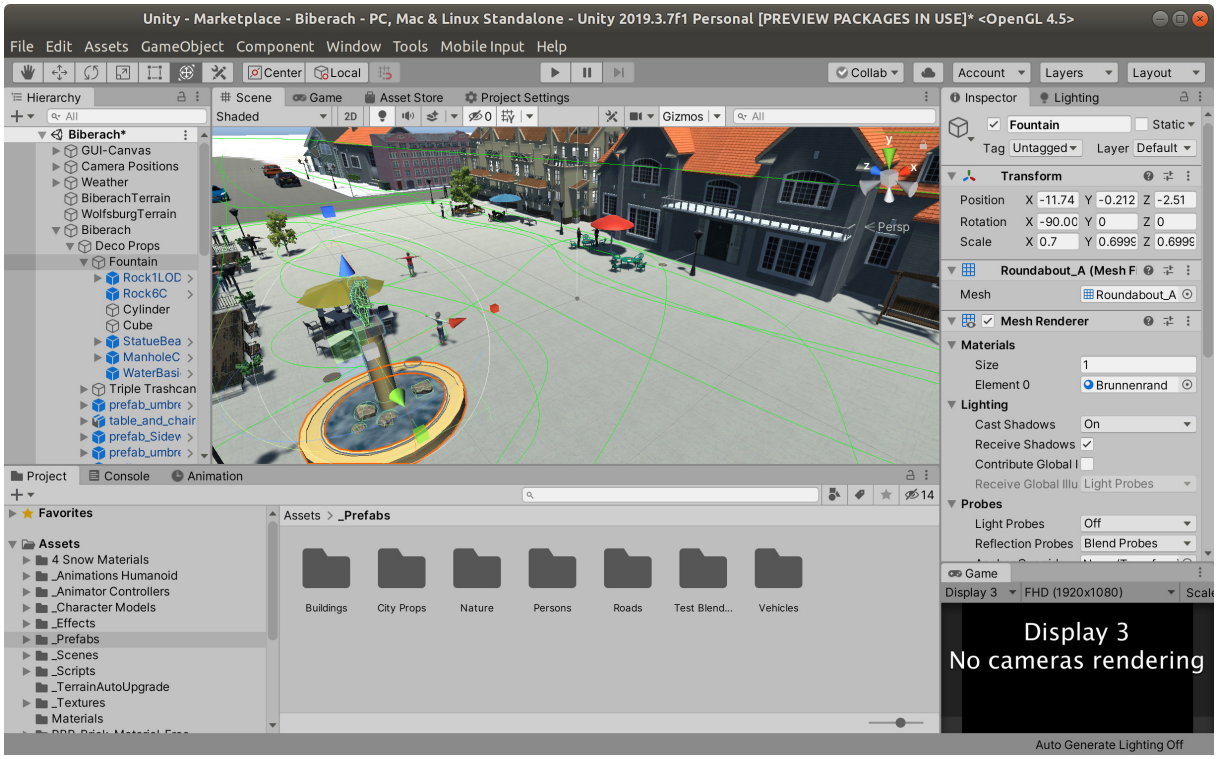


Abb. 4.1.: Benutzeroberfläche des Unity Editors.

In der Mitte des Editors befindet sich die Szenen-Vorschau. Wird ein Objekt in der Hierarchie ausgewählt, so wird dieses in der Szenen-Vorschau hervorgehoben und mit einem Werkzeug zum Verschieben und Skalieren des Objekts ausgestattet, sodass dieses und alle untergeordneten Objekte auch visuell verschoben werden können.

In der rechten Hälfte des Editors befindet sich der Inspektor. Dieser zeigt für jedes Objekt dessen Position, Größe und Rotation in der Szene an und ermöglicht es, das Objekt umzubenennen und mit weiteren Metadaten zu versehen. Darüber hinaus können alle weiteren Eigenschaften des Objekts in diesem Fenster angepasst werden, beispielsweise die Sichtweite einer ausgewählten Kamera oder die Helligkeit einer Lichtquelle. Auch öffentliche Variablen benutzerdefinierter Skripte können hiermit angepasst werden, sodass diese auf bekannte Art und Weise angepasst werden können.

Im unteren Bereich des Editors befindet sich eine Ordneransicht des Projekts, die alle relevanten Dateien enthält. Dazu zählen *Assets* wie 3D-Objekte und *Prefabs*, das sind vorkonfigurierte Unity *Game Objects*, sowie Skripte, Animationen, Texturen und gespeicherte Szenen.

In der oberen Toolbar befindet sich mittig eine *Play*-Schaltfläche, mit der in den Spielmodus gewechselt werden kann. Dieser aktiviert die Spielphysik und benutzerdefinierte Skripte und ermöglicht so ein Testen des entwickelten Spiels. Änderungen in dieser Ansicht werden in der Regel nicht automatisch auf die Szene angewendet. Dennoch ist es weiterhin möglich, Objekte über die Szenen-Vorschau sowie den Inspektor zu modifizieren, so dass dieser Modus auch zum Experimentieren mit den Einstellungen genutzt werden kann. Beim Beenden des Spielmodus wird der Ausgangszustand der Szene wiederhergestellt.

Skripte Für die Programmierung des über die Physik-Simulation hinaus gehenden Verhaltens der 3D-Objekten können Skripte eingesetzt werden. Diese sind in der Sprache `C#` geschrieben und immer einem *GameObject* zugeordnet. Die zwei zentralen Komponenten der Skripte sind die `Start()`-Funktion, die beim Beginn des Spiels aufgerufen wird und die `Update()`-Funktion, die je Einzelbild einmal aufgerufen wird.

Die Skripte erben von einer Unity-Klasse, die einen einfachen Zugriff auf das verbundene sowie auf per Inspektor verknüpfte *GameObjects* ermöglicht.

Detaillierte Beispielskripte mit weiteren Erklärungen werden in den weiteren Abschnitten dieses Kapitels vorgestellt.

4.3. Szenen der Simulation

In der Simulation wurden vier verschiedene Szenen modelliert, die je verschiedene Anwendungsszenarien repräsentieren. Die Szenen sind nach Städten benannt, die genaue Nachbildung deren Marktplätze war aber kein Ziel dieser Arbeit, sie dienten nur als Inspiration und Orientierung.

Biberach Die Stadt Biberach bietet online einen öffentlichen Livestream des Biberacher Marktplatzes an [168]. Da dieser im Rahmen der Recherche als besonders nah am Anwendungsfall bewertet wurde, wurde beschlossen, die erste Szene der Simulation möglichst ähnlich diesem nachzubauen.

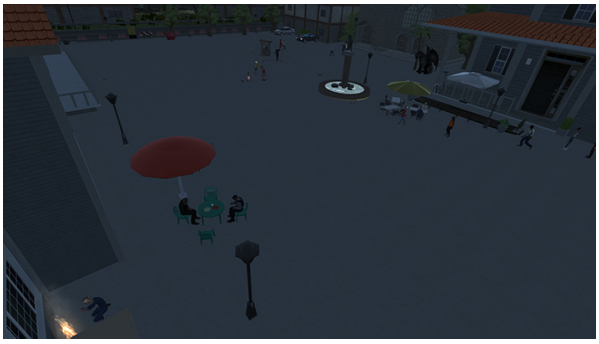
Während einige zentrale Punkte wie der Brunnen und die oben angrenzende Straße sowie das allgemeine Layout des Marktplatzes mit Geschäften und Bäumen vor allem in der Perspektive „Süd“ (siehe Abb. 4.2a) dem Kamerabild des Livestreams (siehe Abb. 3.4b auf Seite 49) stark ähneln, so sind die Details wie Häuserfassaden und Tische auf dem Platz relativ frei interpretiert und darauf ausgelegt, auch aus anderen Perspektiven interessante Inhalte (und Handlungen) zu bieten und auch die Möglichkeit für Spezialfälle wie durch die Fußgängerzone fahrende Fahrzeuge zu geben.

Die Kameras in dieser klassischen Marktplatz-Szene sind in verschiedenen Höhen und Winkeln positioniert und zeigen daher Personen verschiedener Größe. Neben den statischen Kameras gibt es zudem eine Drone, die auf einem definierten Pfad durch die Szene schwebt und dabei das Geschehen des Platzes fokussiert, sowie die Möglichkeit eine freie Kamera zu aktivieren, mit der Details aus nähster Nähe oder nach freiem Belieben gefilmt werden können. Hierfür wurde eine Steuerung mittels Maus und Tastatur implementiert, die einen „freien Flug“ ermöglicht.

(a) Perspektive „Süd“

(b) Perspektive „Nord“

(c) Perspektive „Marktplatzsstraße“

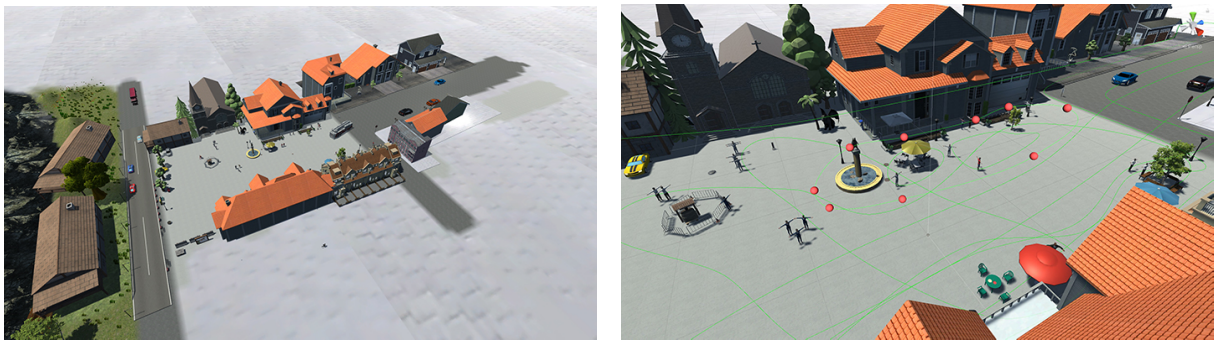


(d) Ansicht der Drone bei Nacht

Abb. 4.2.: Kamera-Perspektiven der Simulationsszene „Biberach“

In Abb. 4.3 werden beispielhaft einige Bilder der verschiedenen Kameraperspektiven gezeigt. Bei der Auswahl der Bilder wurde darauf geachtet, verschiedene Features der Simulation abzubilden, wie beispielsweise die durch den Tag- und Nachtzyklus entstehenden verschiedenen Schattenwürfe, die sich bewegenden Fahrzeuge sowie verschiedene Handlungen wie Schlägereien und die Brandstiftung mit Simulation von Feuer und Rauch.

In Abb. 4.3a wird die gesamte Szene gezeigt. Es ist zu erkennen, dass nicht sichtbare Bereiche nur angedeutet wurden, um Zeit beim Modellieren einzusparen und die Performance des Simulators zu erhöhen.



(a) Aufbau (b) Bewegungspfade

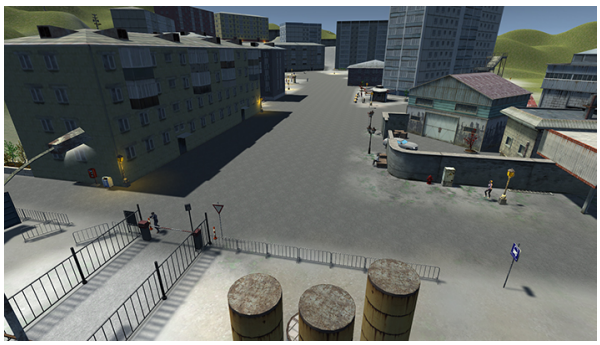
Abb. 4.3.: Aufbau und Bewegungspfade der Simulationsszene „Biberach“

Einige Handlungen wie „gehen“ und „rennen“ können während einer Bewegung eingesetzt werden. Für sich fortbewegende Personen sowie die Fahrzeuge wurden verschiedene Bewegungspfade definiert, an denen sich die computergesteuerten Personen orientieren können. In Abb. 4.3b ist beispielhaft einer der Pfade mithilfe roter Punkte hervorgehoben. Im Editor kann der Pfad durch bewegen der roten Punkte angepasst werden. Auch die Geschwindigkeit sowie die Blickrichtung der Personen kann angepasst werden. Die vollständigen Pfade sind im Editor grün dargestellt.

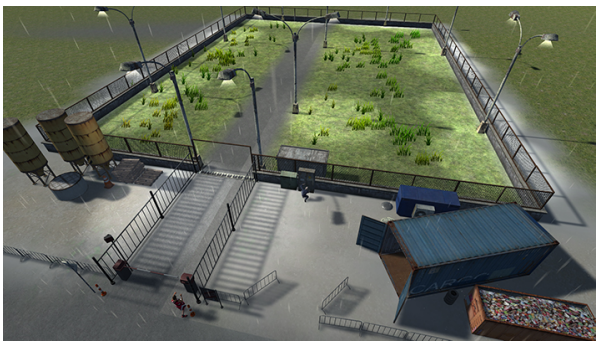
Für jedes Objekt kann eine „Spawn“-Wahrscheinlichkeit angegeben werden. Diese bestimmt die Wahrscheinlichkeit, mit der das Objekt bei Beginn der Simulation in die Szene gesetzt wird. Mithilfe dieser Technik können Events wie das Fahren des Feuerwehrautos oder einzelne Handlungen zufallsgesteuert aktiviert werden. Kombiniert mit den zufällig ausgewählten Charaktermodellen kann die Varietät der Szene so deutlich gesteigert und ein nicht-deterministisches Verhalten erzielt werden.

Zur weiteren Erhöhung des Realismus der Szene sowie zur Implementierung von zufallsgesteuerten Verdeckungen wurde ein Skript eingebunden, das automatisiert in zufälligen Mustern Vögel durch die Szene fliegen lässt. Darüber hinaus wurde eine sich bewegende Katze in die Szene gesetzt. Die detailliert modellierten Bäume bewegen sich zudem abhängig von der Windrichtung, um ein einheitliches und realistisches Aussehen zu erzielen.

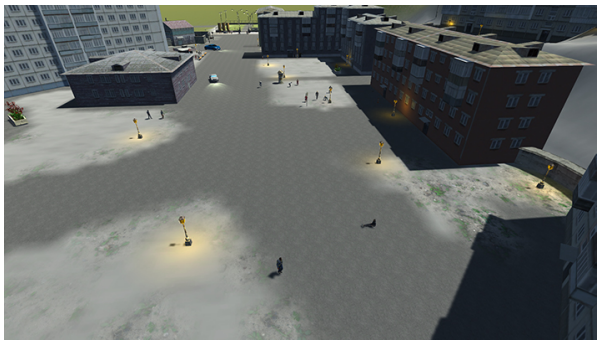
Wolfsburg Die industriell geprägte Szene „Wolfsburg“ beinhaltet verschiedene Wohnkomplexe, eine Einfahrt zu einem Firmengelände und einen sich im Aufbau befindenden Solarpark (siehe Abb. 4.4).



(a) Perspektive „Straßenschlucht“



(b) Perspektive „Solarpark“



(c) Perspektive „Innenhof“ bei Tag

(d) Perspektive „Innenhof“ bei Nacht

Abb. 4.4.: Kamera-Perspektiven der Simulationsszene „Wolfsburg“

Zu den gezeigten Handlungen zählen neutrale Handlungen sowie „gehen“, „rennen“, „hinfallen“ und „sitzen“.

In dieser Szene wurde besonders auf die Beleuchtung bei Nacht geachtet sowie darauf geachtet, dass Bewegungspfade von Passanten und Fahrzeugen nicht im sichtbaren Bereich enden.

In dem in Abb. 4.4b Beispielbild ist auch der simulierte Regen zu erkennen.

Mondsee Im Mittelpunkt dieser naturbetonten Szene steht ein kleiner See, dessen physikbasiertes Wasser und das animierte Gras am Ufer die Szene sehr gemütlich und realistisch wirken lassen. Wie in Abb. 4.5 gezeigt, gibt es zudem zahlreiche Bäume, einen Steg sowie zwei Holzhütten mit kleinem Garten und einem Arbeitsbereich.

Die Grasfläche kann zur Generierung von teilverdeckten Handlungen genutzt werden, da Gras und Bäume je nach Positionierung in den Weg gestellt werden können.

(a) Übersicht über die Szene

(b) Seitenansicht

(c) Drone

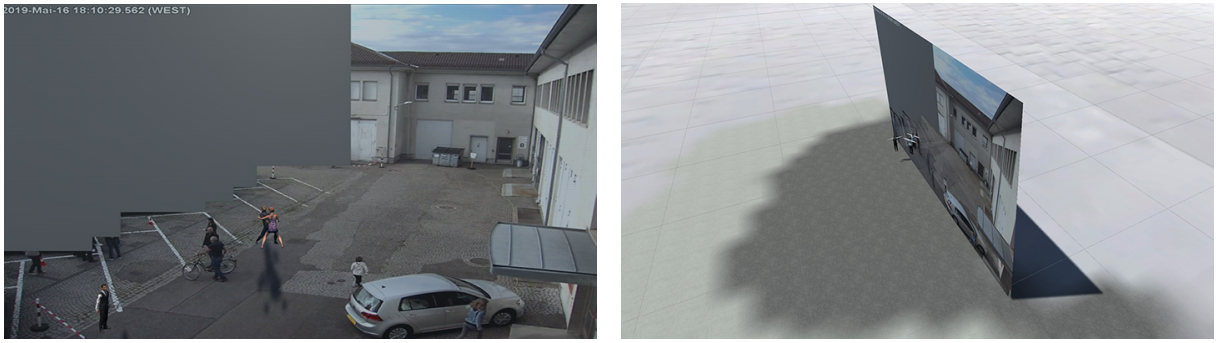
(d) Steg am See

Abb. 4.5.: Kamera-Perspektiven der Simulationsszene „Mondsee“

In den für den S-SPHAR-Datensatz generierten Videos treiben auf der Wiese verschiedene Personen Sport und führen weitere neutrale Handlungen aus. Auf dem Wanderweg wird die Aktivität „gehen“ ausgeführt. Hinter den Holzhütten wird durch Verwendung der Geiselnahme-Animation ein Mordfall simuliert.

Karlsruhe Für die Szene „Karlsruhe“ wurde ein besonderes Konzept erarbeitet und implementiert, mit dem der Hintergrund der Szene ohne Kenntnisse in der 3D-Modellierung ausgetauscht werden kann.

Wie in Abb. 4.6 gezeigt, wird als Hintergrund ein Foto verwendet, das aus einer realen Aufnahme stammt. Auf diesem Foto sollten idealerweise keine Personen zu sehen sein, da diese in der Simulation als 3D-Charaktere kurz vor das Bild gesetzt werden.



(a) Kameraperspektive (b) Bühnenartiger Aufbau in der 3D-Engine

Abb. 4.6.: Aufbau und Perspektive der Simulationsszene „Karlsruhe“

Die Animationen der 3D-Charaktere sowie deren Schattenwürfe werden dynamisch berechnet, womit die Szene weniger statisch wirkt. Auch die globalen Effekte wie Sonnenstand und Niederschlag werden bei dieser Szene angewendet.

Der einfache Austausch des Hintergrunds ermöglicht nicht nur die Generierung von großen Mengen an Trainingsmaterial vor verschiedenen Hintergründen, sondern erlaubt auch die Feinanpassung von Modellen für bestimmte Kundeninstallationen anhand von Bildern, ohne die Szene aufwendig dreidimensional nachzubilden.

Diese Methode kann auch genutzt werden, um bei neuen Kunden eine schnelle Qualitätseinschätzung der Modelle vorzunehmen, in dem eine relative Messung der Genauigkeit durchgeführt wird.

4.4. Animationen und Bewegungen

Grundsätzlich können 3D-Objekte (*Game Objects*) in einer Szene auf verschiedene Arten bewegt werden. Neue Objekte sind in Unity standardmäßig statisch und bewegungslos. Ein gängiges Verfahren zur Erstellung von interaktiven 3D-Welten ist die Verwendung von simulierter Physik. Wird den Objekten eine Physik-Komponente hinzugefügt, so fallen diese beispielsweise durch die Luft, bis sie auf ein festes Objekt stoßen, wobei sie auch beispielsweise an anderen fliegenden Objekten abprallen können.

Eine weitere Methode zur Bewegung von Objekten ist das Verändern der Objektposition durch Skripte oder Animationen. Skripte werden vor allem für interaktive Objekte mit vielen Funktionen, wie beispielsweise einem spielbaren Charakter, eingesetzt. Animationen sind vor allem dafür gedacht, untergeordnete Objekte in einem zusammengesetzten Objekt zu bewegen, beispielsweise die Finger einer Person oder die Reifen eines Fahrzeugs.

Im Rahmen dieser Arbeit wurden primär Personen mit Animationen versehen, aber auch einzelne Fahrzeuge oder rein dekorative Objekte, wie beispielsweise eine Schranke wurden zur Erhöhung der Realitätstreue mit Animationen versehen.

In den folgenden Unterabschnitten wird beschrieben, wie Charaktere mit Animationen versehen werden können, wie diese Animationen gesteuert werden können und wie sich die Personen in der Szene entlang eines Pfades bewegen können.

Importieren von Mixamo-Animationen und Charakteren Wie in Abschnitt 3.1.3 beschrieben bietet Mixamo Animationen und Charaktere zur kostenlosen Verwendung an. Um diese in Unity einzubinden, müssen spezielle Parameter beim Download und Import gesetzt werden.

Für die folgenden Schritte ist die Reihenfolge von großer Bedeutung, da die verschiedenen Einstellungen in die nächsten Schritte übernommen werden.

Der Download von Animationen erfolgt auf der Website *mixamo.com* immer in Kombination mit einem Charakter, weshalb zunächst beide ausgewählt werden müssen. Innerhalb Unity können die Animationen und Charaktere dann dennoch unabhängig genutzt werden. Für den Download von Charakteren empfiehlt es sich, diese ohne Animation herunterzuladen, da sie dann die sogenannten *T-Pose* Haltung haben, die eine automatische Zuordnung von Gelenkpunkten ermöglicht, dazu später mehr.

Nach Auswahl einer Animation können die animierten Charaktere heruntergeladen werden. Hierbei ist darauf zu achten, dass das Format „FBX for Unity“ ausgewählt ist. Für Animationen, die später zur Fortbewegung von Objekten eingesetzt werden (zum Beispiel Geh-Animationen), sollte zudem die Option „in place“ aktiviert werden.

Für jeden Charakter empfiehlt es sich, im Unity Projekt einen eigenen Ordner anzulegen, in den alle `.fbx`-Dateien dieses Charakters heruntergeladen werden. Bevor die Charaktere in die Szene gesetzt werden, sollten einige Import-Einstellungen vorgenommen werden.

Als erste Maßnahme sollte je Charakter einmal in den Einstellungen die Schaltfläche „Extract Textures“ betätigt werden. Diese speichert alle relevanten Texturen des Charakters in einen Unterordner. Falls hierbei eine Fehlermeldung zur „Korrektur von Normalmaps“ kommt, kann ohne Bedenken die Option „Fix now“ ausgewählt werden.

Einige Charaktere werden von Mixamo fehlerhafterweise in doppelter Größe ausgeliefert. Bei diesen muss der „Scale factor“ auf 0.5 gesetzt werden.

Damit die Animationen auf den Charakteren funktionieren, muss Unity das Skelett der Charaktere erkennen. Bei Charakteren in der „T-Pose“-Haltung können die Gelenke automatisch erkannt werden, wenn in den Import-Einstellungen im Tab „Humanoid“ die Schaltfläche „Apply“ betätigt wird. Ansonsten können die Gelenke über die Funktionen „Configure“ und „Bones“ angepasst werden.

Nach diesen Einstellungen kann die `.fbx` Datei wie jedes andere Asset per *drag & drop* in die Szene gezogen werde.

Bei einzelnen Modellen kann es zu Fehlern in der Anzeige der Textur kommen. Falls dies der Fall ist, sollte bei jedem generierten Material im Texturen-Ordner der „Legacy Shader/Diffuse“ gesetzt werden und anschließend die `.fbx`-Datei erneut in die Szene gesetzt werden.

Die Animationen müssen vor ihrer Benutzung aus der `.fbx`-Datei extrahiert werden. Mithilfe des kleinen Pfeils in der Dateiansicht können die Komponenten der Datei gesehen werden. Wird wie in Abb. 4.7 gezeigt die untergeordnete Animation ausgewählt, kann diese durch Druck der Steuerungstaste und „D“ aus der Datei heraus in den aktuellen Ordner dupliziert werden.

Abb. 4.7.: Komponenten einer .fbx Datei, darunter 3D-Objekte für einzelne Körperteile, Materialien mit Texturen, Skelett-Informationen und Animationen.

Nach dem Duplizieren ist die Animation nicht mehr schreibgeschützt und es können Einstellungen an ihr vorgenommen werden. Je nach gewünschtem Einsatzzweck, ist es wichtig, die Optionen „Loop Time“ (bei allen Animationen, die endlos abgespielt werden sollen) und „Bake Into Pose“ (bei allen Animationen, die den Charakter nicht bewegen sollen) zu aktivieren.

Anschließend kann die `.anim`-Datei via *drag & drop* einem beliebigen (!) Charakter-GameObject oder Animator Controller zugewiesen werden.

Nutzen und Steuern von Animationen In Unity gibt es zahlreiche Komponenten rund um das Thema Animation. Das hierfür verwendete Vokabular klingt für Neueinsteiger zunächst sehr ähnlich, weshalb in diesem Abschnitt eine kleine Übersicht über die verschiedenen Begriffe und deren Aufgaben gegeben wird.

Die sogenannten *Animation Clips* haben die Dateiendung `.anim` und enthalten die Bewegungsinformationen, welches Gelenk sich wann wohin bewegen soll. Um diese aufzurufen, werden *Animator Controller* eingesetzt. Diese bestimmen die Übergänge zwischen Animationen und bestimmen, wann welche Animationen abgespielt werden.

Animator Controller sind grundsätzlich nicht *GameObjekt*-spezifisch. Um diese einem bestimmten *GameObject*, also beispielsweise einem konkreten Charakter zuzuordnen, benötigt das *GameObject* eine *Animator* Komponente. Diese verknüpft das Gameobject und dessen Skelettdaten mit dem *Animation Controller*, und so dem *Animation Clip*.

Animation Controller sind ähnlich wie Zustandsautomaten aufgebaut und können über eine grafische Oberfläche konfiguriert werden (siehe Abb. 4.8). Die einzelnen Zustände heißen *Animation States* und beinhalten die Verweise auf *Animation Clips*. Die Zustandsübergänge werden *Animator Transition* genannt und bestimmen die Bedingungen zum und das Verhalten beim Wechsel von Animationszuständen.

Für Animationen, die den Charakter nicht bewegen sollen, sollte im *Animation State* die Option *Foot IK* aktiviert werden. Diese Einstellung verhindert ungewollte Bewegungen, sofern im *Base Layer* des *Animation Controllers* die Funktion *IK Pass* aktiviert ist. *IK* steht für *inverse kinematics* und ist eine bessere Alternative zur voreingestellten *forward kinematics* Methode.

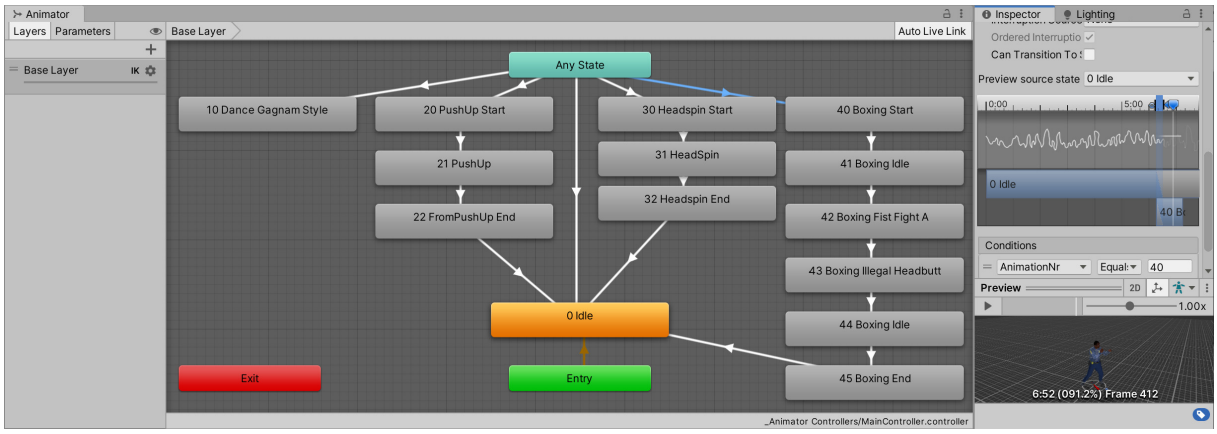


Abb. 4.8.: Steuerung von Animationen in Unity. Gezeigt wird die Konfiguration eines *Animation Controller* im *Animator*. In Blau ausgewählt ist der Übergang von einem beliebigen Zustand zur Animationsreihe *Boxing*, die aus mehreren sequenziell angeordneten Animationen besteht. In der rechten Hälfte des Fensters befinden sich verschiedene Optionen für den ausgewählten Zustandsübergang oder Zustand sowie eine Live-Vorschau der Animationsübergänge.

Um den Umgang mit den verschiedenen Animationen, Übergängen und Import-Optionen zu erlernen, wurde eine Animations-Testszene entwickelt, in der mit dem in Abb. 4.8 gezeigtem zentralen *Animation Controller* Animationen auf mehreren Charakteren gleichzeitig ausgeführt und synchron per Tastendruck gewechselt werden können (siehe Abb. 4.9).

Die Kommunikation zwischen grafischer Oberfläche und Animation erfolgt durch ein Skript, das eine für alle derartigen *Animation Controller* Instanzen zugängliche Variable setzt.

Abb. 4.9.: Szene zum Test von Animationen. Die Lücken in der Formation entstehen durch ein Skript, das bewusst Charaktere ausblendet, um die Zufälligkeit zu erhöhen. Die Auswahl einer Animation kann jederzeit über ein Menü (oben rechts im Bild) erfolgen, die Übergänge zwischen beliebigen Haltungen werden automatisch berechnet. Im Bild wird die Animation „push ups“ gezeigt.

Ergänzend zur Steuerung der *Animation Controller* mit der grafischen Oberfläche des *Animators*, können diese auch über Skripte angesprochen werden. Um beispielsweise nach Abschluss einer Animation die nächste abzuspielen, kann ein Skript wie in Quellcode 4.1 gezeigt eingesetzt werden.

```
1  using UnityEngine;
2  public class ChangeAnimNrBehaviourScript : StateMachineBehaviour {
3    public int NewAnimationNrOnExit = -1;
4    override public void OnStateExit (Animator animator,
5                                      AnimatorStateInfo stateInfo,
6                                      int layerIndex) {
7      if (NewAnimationNrOnExit != -1) {
8        animator.SetInteger ("AnimationNr", NewAnimationNrOnExit);
9      }
10   }
11 }
```

Quellcode 4.1: Beispiel für programmatische Animation. Im gezeigten Code wird beim Beenden einer Animation eine Variable auf einen bestimmten Wert gesetzt. Im *Animator* ist diese Variable die Grundlage für Übergänge zu verschiedenen Animationen, wobei der Wert der Variable den nächsten Zustand bestimmt.

Austausch von Charakter-Modellen Das im Rahmen dieser Arbeit erstellte C#-Skript für Unity `RandomModel.cs` kann auf ein leeres GameObject angewendet werden und erstellt automatisch einen Charakter mit einem zufälligen Charakter-Modell (aus einer konfigurierbaren Liste) als untergeordnetes Objekt. Darüber hinaus wendet es einen im Inspektor des Skripts definierten *Animator Controller* sowie die Schicht des leeren GameObjects auf diesen neu instanziierten Charakter an, sodass Animationen und die Beschriftung korrekt übereinstimmen.

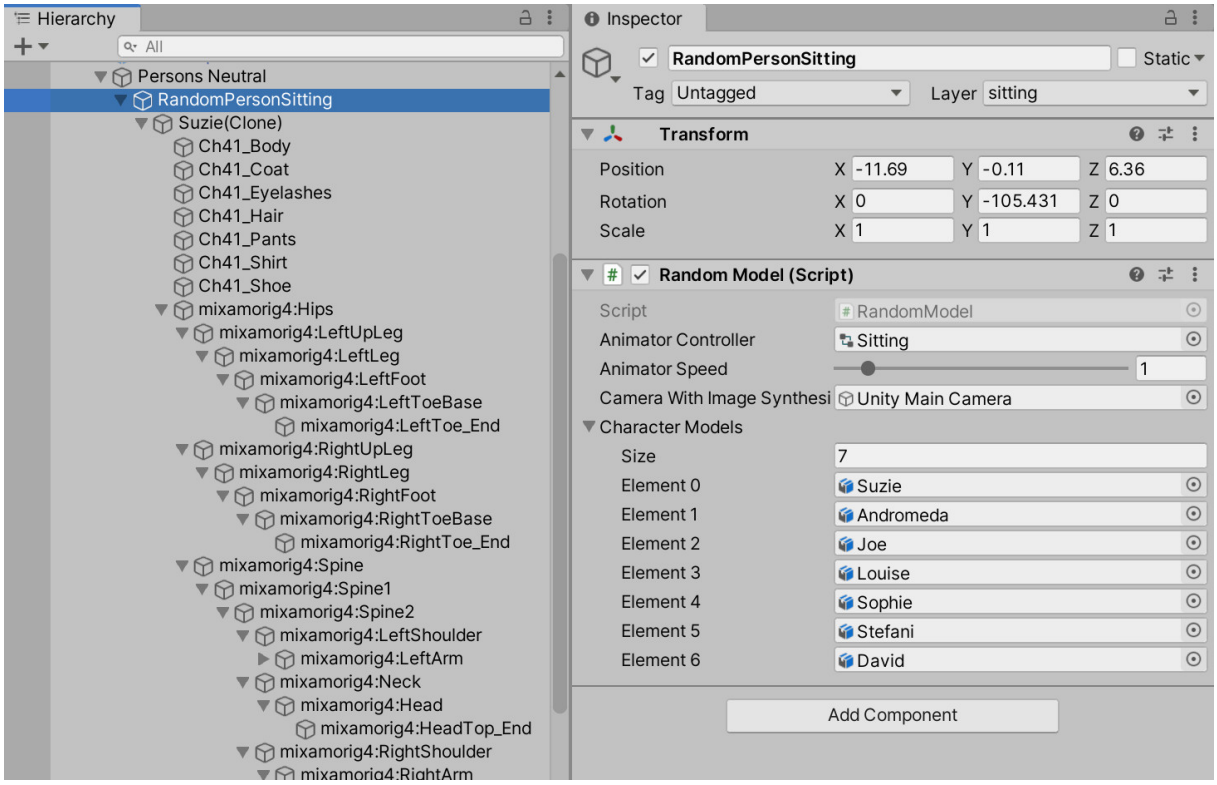


Abb. 4.10.: Konfiguration des Skripts zum Austausch von Charakter-Modellen. Links ist ein Ausschnitt aus der hierarchischen Ansicht aller Objekte in der Szene zu sehen, rechts die Komponenten des ausgewählten GameObjects „RandomPersonSitting“, darunter auch das eigene Skript `RandomModel.cs`, dessen globale Variablen automatisch zur Bearbeitung angeboten werden.

In Abb. 4.10 wird die Konfiguration des Skripts über den Inspektor gezeigt. Das Skript instanziiert im Editor-Modus bei jeder Änderung automatisch das erste Modell der Liste, sodass in der 3D-Ansicht stets ein definierbares Modell als Vorschau zu sehen ist. Beim Start des Simulators wird dann ein zufälliges Modell aus der Liste ausgewählt.

Jedes Modell hat die gleiche Chance, ausgewählt zu werden. Soll diese Wahrscheinlichkeit erhöht werden, so ist es möglich, dass gleiche Objekt zweimal zur Liste hinzuzufügen.

Bewegen von Objekten entlang eines Pfads Um ein 3D-Objekt, wie beispielsweise eine animierte Person oder ein Fahrzeug, durch die Szene zu bewegen, bietet es sich an, dessen Position mithilfe eines Skripts zu verändern.

Da es zum Bewegen von Objekten anhand eines definierten Pfads bereits vorgefertigte Skripte gibt, die einen höheren Komfort als selbstentwickelte Lösung bieten, wurde auch in dieser Arbeit ein bestehendes Skript eingebunden.

Der „Bézier Path Creator“ [100] von Sebastian Lague kann kostenlos aus dem Unity Asset Store heruntergeladen werden und dank der MIT Lizenz ohne Einschränkungen verwendet werden.

Nach dem Import des Pakets in das Unity Projekt kann ein neues „*Empty GameObject*“ erstellt und mit einer *Path Creator* Komponente versehen werden. Mithilfe der angezeigten Punkte kann der Pfad per *drag & drop* oder dem Inspektor angepasst werden. Ein Klick bei gedrückter Umschalt-Taste erstellt einen neuen Wegpunkt, ein Klick bei gedrückter STRG-Taste löscht einen bestehenden Wegpunkt.

Damit sich ein Objekt anhand eines dieser Pfade bewegt, benötigt es ein neues C#-Skript als Komponente. Dieses Skript wurde in der Simulation `Follower.cs` genannt und enthält den in Quellcode 4.2 gezeigten Code.

```
using UnityEngine;
using PathCreation;

public class Follower : MonoBehaviour
{
  public PathCreator pathCreator;
  public float speed = 5;
  float distanceTravelled;
  Quaternion newRotation;

  void Update()
  {
    distanceTravelled += speed * Time.deltaTime;
    transform.position = pathCreator.path.GetPointAtDistance(distanceTravelled);
    newRotation = pathCreator.path.GetRotationAtDistance(distanceTravelled);
    transform.eulerAngles = new Vector3(0, newRotation.eulerAngles.y , 0);
  }
}
```

Quellcode 4.2: Skript zur Bewegung eines 3D-Objekt anhand eines Pfads. Als Besonderheit dieses Skripts wird nur die Position sowie die Rotation um die y-Achse aktualisiert, da es sonst bei leicht falsch gesetzten 3D-Pfaden zu weiteren für Personen unnatürliche Rotationen kommt.

Über den Inspektor kann dem Skript nun der erstellte Pfad zugewiesen werden. Der Vorteil dieser Trennung von Pfad und Pfad-*Follower* ist, dass Pfade für mehrere *GameObjects* wiederverwendet werden können.

4.5. Grafische Benutzeroberfläche der Simulation

Damit die Simulation ohne Kenntnisse des Unity Editors bedient werden kann, wurde ein Menü programmiert, mit dem sich die zentralen Skripte der Simulation steuern lassen. Es ist zu jeder Zeit in einer Ecke des Bildschirms sichtbar, wird aber nicht in die exportierten Kamerabilder mit aufgenommen.

Die in Abb. 4.11 gezeigte Oberfläche dient sowohl zur Steuerung als auch zur Information. So werden neben der aktuellen Bildwiederholrate auch die aktuell gewählten Einstellungen, der Aufnahmestatus sowie die Hotkeys der Simulation gezeigt, denn einige Zustände der Oberfläche lassen sich auch mit einem passenden Tastendruck aktivieren oder deaktivieren.

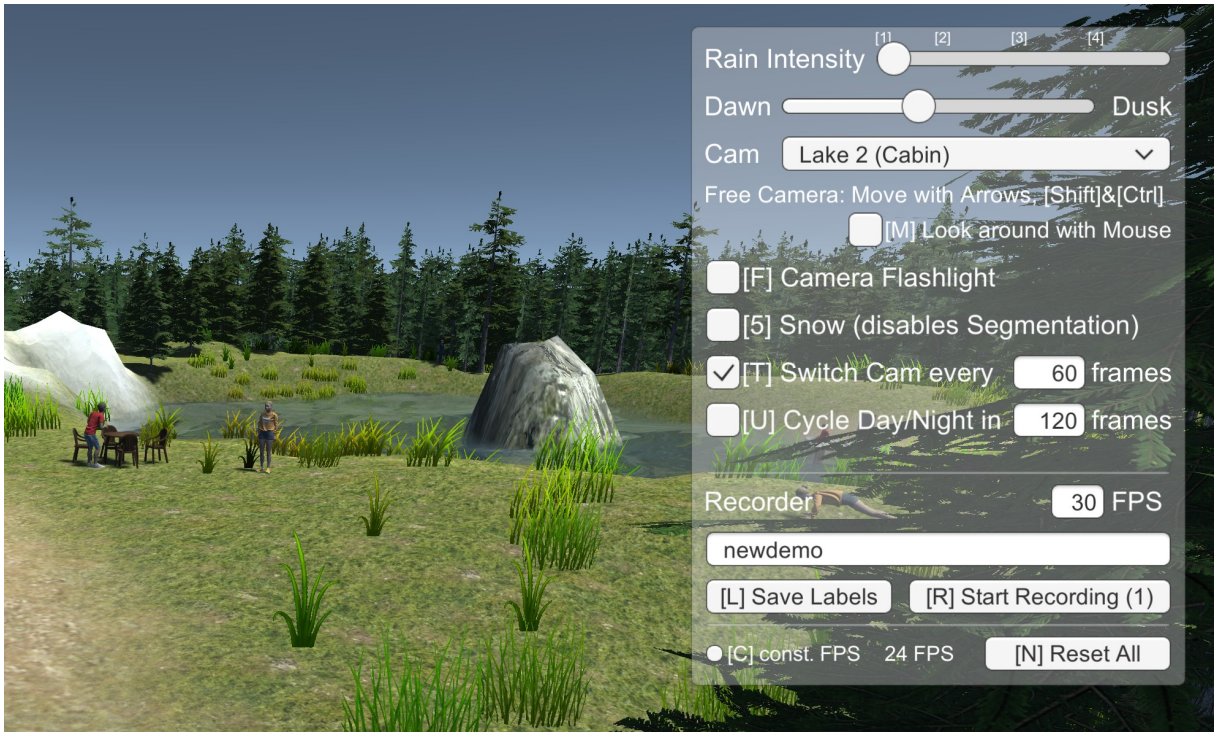


Abb. 4.11.: Grafische Bedienungsoberfläche der Simulation

Zum Starten der Aufnahme von Original- und Segmentierungsbildern genügt ein Klick auf die „Start Recording“ Schaltfläche. Die Schaltfläche wechselt daraufhin den angezeigten Text zu „Stop Recording“. Die Zahl in Klammern gibt die Anzahl der bisher aufgenommenen Einzelbilder an.

Über die gezeigte Oberfläche hinaus können die einzelnen Werte natürlich auch über den Unity Inspektor angepasst werden. Für die in dieser Arbeit verwendete Grundfunktionalität des Simulators ist dies aber nicht notwendig.

4.6. Maßnahmen zur Erhöhung der Diversität

Bei der Erstellung des synthetischen Datensatzes wird in dieser Arbeit nicht das Ziel verfolgt, den Herausforderungen der realen Welt zu entgehen, sondern diese realistisch nachzubilden. Das Modell zur Handlungserkennung kann situationsübergreifend lernen, wenn dieses wie in Abschnitt 2.4 beschrieben bereits im Training mit den Herausforderungen konfrontiert werden.

Die entwickelte Simulation verfolgt das Ziel, für sämtliche der ermittelten Herausforderungen passende Lösungen zu bieten. In Tabelle 4.1 werden die jeweils getätigten Maßnahmen gezeigt.

Herausforderung	Maßnahme
antropometrische Diversität	Verwendung verschiedener 3D-Charaktere & Animationen.
Betrachtungswinkel	Verschiedene Kamera-Positionen und Winkel, freie Kamera möglich.
Hintergrundbilder	Realistischer Nachbau mehrerer städtischer und ländlicher Szenen in 3D mit bewegten Passanten, Fahrzeugen und Tieren mit verschiedenen Texturen im Vorder- und Hintergrund, Implementierung einer Szene mit freiem Hintergrundbild.
Klassen-Vielfalt/Ähnlichkeit	Verwendung möglichst gut distanzierbarer und für den Anwendungsfall sinnvoller Klassen, immer genau eine Klasse pro Person. Anlehnung an *HMDB* und *SPHAR* Klassen.
Videos schlechter Qualität	Simulation kann in beliebiger Auflösung und Bildrate gerendert werden, Pixelgenaues Labeling, 3D-Objekte basieren auf skalierbare Vektoren, Texturen sind hochauflösend gewählt. Keine Kompressions-Artefakte per-se, aber bei Bedarf simulierbar im Nachhinein durch Umwandlung der Videos mit einem verlustbehafteten (*lossy*) Codec, beispielsweise `.mp4` statt `.avi` .
Verdeckungen	Verwendung von statischen und sich nach verschiedenen Methoden bewegenden 3D-Objekten wie Bäumen, Häusern, Fahrzeuge, Personen und Tiere.
Beleuchtung & Schatten	Simulation basiert auf Physik-Engine mit realistisch gerenderter Szene- Beleuchtung. Beliebig einstellbarer Sonnenstand (Tag- und Nachtzyklus möglich) sowie die Möglichkeit zur Aktivierung einer auf Kamerahöhe montierter Lampe.
Skalenvarianz	verschiedene Größen der Handlungen durch verschiedene Kamera-Positionen und Platzierungen der Handlungen in der 3D-Szene.
Kamera-Bewegungen	Simulation enthält statische Kameras, Kamerafahrten sowie eine freie Kamera, ungewünschte Kamera-Bewegungen kommen daher nur auf Wunsch vor.
unzureichende Trainingsdaten	Simulation enthält Zufallsfaktoren wie beispielsweise Vögel-Routen, Erscheinungs-Wahrscheinlichkeiten von Charakteren, zufällige Texturen und Modelle oder zufällige Start-Positionen von Charakteren und kann zudem jederzeit manuell verändert werden, sodass viele Möglichkeiten zur Generierung neuer Trainingsdaten gegeben sind.
Wetterzustände	Verschiedene Regen-Intensitäten und Schneefall möglich.

Tabelle 4.1.: Behandlung der Herausforderungen für Handlungserkennung durch die Simulation.

Zusammenfassend konnte also zu jeder der in Abschnitt 2.4 ermittelten Herausforderungen eine geeignete Maßnahme ermittelt und in die Simulation implementiert werden. So kann eine hohe Diversität der Szene sichergestellt werden, wodurch sich die synthetischen Daten besonders gut zum Trainieren eines Modells zur Handlungserkennung eignen sollten.

4.7. Erhebung und Zuschnitt der synthetischen Videos

In diesem Abschnitt wird beschrieben, wie die in der Simulation erzeugten Videos automatisch mit Segmentierungsmasken zu den jeweiligen Handlungen beschriftet und als für das Training passende Videos abgespeichert werden.

Speichern der Klarbilder und Segmentierungsmasken Jedes Objekt der Simulation ist im Unity-Editor einer bestimmten Schicht (*Layer*) zugeordnet. Für jede der zu unterscheidendem Handlungsklassen wurde eine gleichnamige Schicht erstellt und die Personen, die entsprechende Handlungen ausführen, wurden dieser Schicht zugeordnet (siehe Abb. 4.12).

Die entwickelte „Recording"-Funktion des Simulators baut auf der internen Bildschirmaufnahme (*Screenshot*) Funktion der Unity Engine auf. Diese Funktion nimmt keine Bedienelemente der Grafischen Oberfläche, sondern nur die gerenderte 3D-Welt auf.

Unity unterstützt die Bildausgabe auf mehreren und auch virtuellen Bildschirmen. Die Simulation bedient zu jedem Zeitpunkt zwei Bildschirme: Den Hauptbildschirm, der das Klarbild und die Benutzeroberfläche enthält, und einen virtuellen Zweitbildschirm, auf dem für jedes Einzelbild die passende Segmentierungsmaske angezeigt wird (siehe Abb. 4.12).

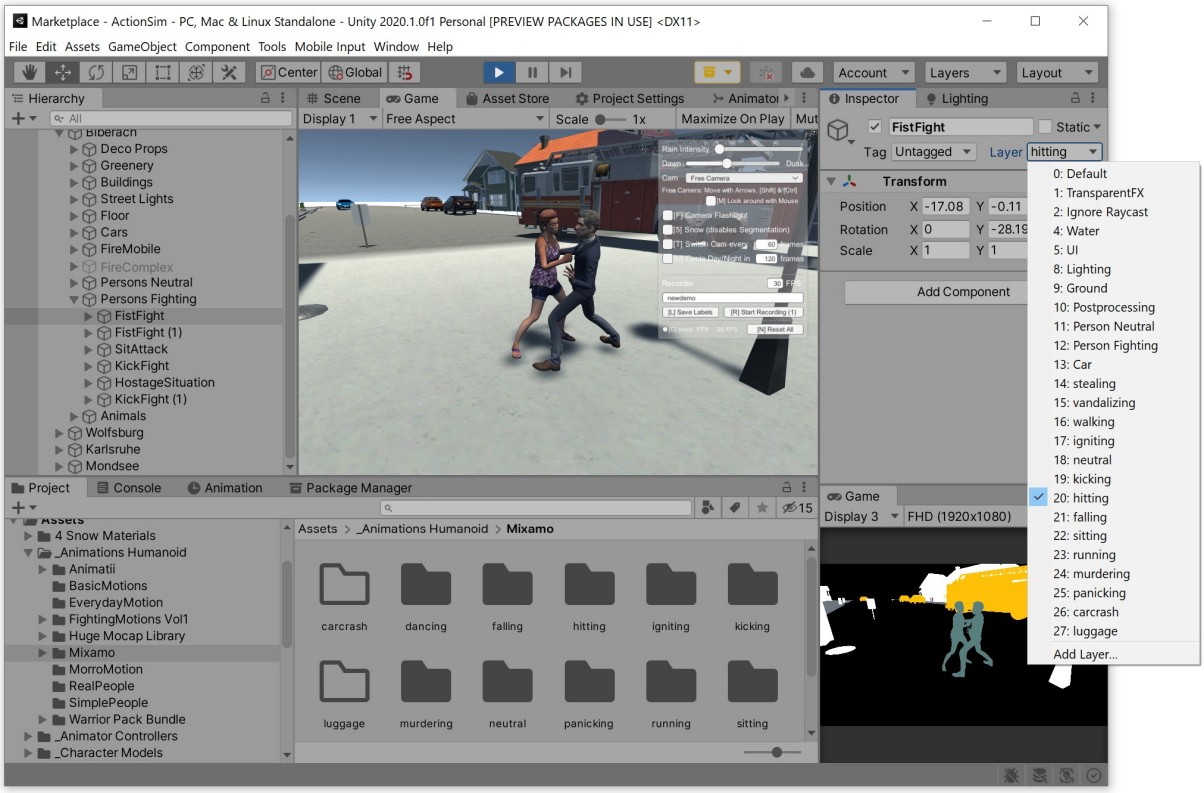


Abb. 4.12.: Auswahl und Vorschau der Segmentierung im Unity Editor.

Die Berechnung der Maske erfolgt durch Verwendung eines alternativen *Shaders*, also einer alternativen Definition zur Berechnung der finalen Bildausgabe. Auf der *Shader*-Ebene ist es möglich, für jeden Pixel im Bild die Schicht des gezeigten Objekts zu bestimmen. Abhängig von der Schicht erhält der Pixel in der Segmentierungs-Kamera eine andere Farbe. Als Basis für diesen *Replacement Shader* wurde der Code aus der Bibliothek *ml-imagesynthesis* [178] verwendet.

Im Rahmen dieser Arbeit wurde auch das jüngere „Unity Perception Toolkit“ [181] getestet, dieses konnte aber nicht in einem angemessenen Zeitrahmen funktionsfähig in das Projekt eingebunden werden.

Die Berechnung der beiden Kameras erfolgt je nach Szene und Rechenleistung des PCs in Echtzeit (rund 30 Einzelbilder pro Sekunde). Sobald die Aufnahmefunktion aktiviert wird, werden nach der Berechnung jedes Einzelbildes je Kamera ein *Screenshot* gespeichert. Der Schreibvorgang auf die Festplatte ist vergleichsweise langsam, weshalb die Bildrate während der Aufnahme auf rund 3 bis 4 Einzelbilder pro Sekunde sinkt, abhängig von der gewünschten Auflösung der Bilder.

Für die Aufnahme wird im Ordner des Unity Projekts ein Unterordner „rec“ angelegt. Dieser enthält für jeden Aufnahmevorgang einen Unterordner nach dem folgenden Namensschema:

```
[AufnahmeName]_[SimulationsName]_[Bildrate]fps
```

In diesem befinden sich zunächst zwei Ordner: `_img` mit den Klarbildern, sowie `_layer` mit den Segmentierungsmasken. Die Bilder liegen im `.png`-Format vor und sind anhand einer aufsteigenden Nummer benannt.

Die Zuordnung von Farben in der Segmentierungsmaske zu den Namen der Schichten und damit Klassen erfolgt in der Datei `layers.json`. Diese wird generiert, sobald in der Simulation die Schaltfläche „Save Labels“ oder die Taste L gedrückt wird. Die Datei ist wie folgt aufgebaut:

```
{"labels":[{"name":"hitting", "color": "598080"}, ...]}
```

Die Farbwerte sind hexadezimal kodierte RGB-Farbcodes. Diese können aufgeteilt werden in die Paare RRGGBB, wobei jedes Zahlenpaar der Intensität der jeweiligen Farbe auf einer Skala von 0 bis 255 entspricht.

Die so entstehenden Videos und Beschriftungen können für die räumliche und zeitliche Lokalisierung von Handlungen verwendet werden.

Zuschnitt der Videos Für den Anwendungsfall dieser Arbeit sollen in Echtzeit-Videos Handlungen erkannt werden. Die hierfür eingesetzte Handlungserkennung geht, wie in Abschnitt 3.4 beschrieben, davon aus, dass je Video genau eine Handlung stattfindet. Zur Anwendung erhält dieser daher ausschließlich bereits auf Personen zugeschnittene Videos. Es macht daher Sinn, auch während des Trainings derart zugeschnittene Videos zu verwenden.

Die Konvertierung der durch die Simulation erzeugten Bilder zu einem Video sowie der Zuschnitt dieses Videos in mehrere räumlich und zeitlich zugeschnittene Videos erfolgt mithilfe des selbst geschriebenen Skripts `generate_video.py` [125].

Das Skript nimmt den Ordner der Aufnahme mitsamt seiner Unterordner mit den Originalbildern und Segmentierungsmasken sowie einer Zuordnung von Maskenfarben zu Handlungsklassen als Eingabe und generiert eine Ausgabe in Form eines Ordner mit allen zugeschnittenen Videos (siehe Abb. 4.13b) sowie ein hochauflösendes Video, dass vier Varianten der Eingabe und derer verarbeiteten Formen darstellt (siehe Abb. 4.13a). In der Konsole werden darüber hinaus auch die Bounding-Box-Koordinaten und Klassen im *JSON*-Format ausgegeben.

Für den räumlichen Zuschnitt auf Personen über mehrere Einzelbilder hinweg musste ein Tracking implementiert werden. Dies ordnet jeder Instanz einer Handlung eine eindeutige ID zu (siehe unten rechts in Abb. 4.13b).

(a) Multiview (b) Tubes

Abb. 4.13.: Ausgabe des Skripts zum Zuschnitt der Simulationsvideos. Generiert werden ein *Side-by-Side*-Video und *bounding box tubes* der enthaltenen Handlungen. Im Bild werden Screenshots eines Teils dieser Videos gezeigt.

Das Skript kann also neben der Generierung der Zuschnitte auch zur Generierung eines *Side-by-Side*-Videos genutzt werden, beispielsweise zur synchronen Darstellung von Klarbildern und Segmentierungsmasken, für eine überlappenden Darstellung dieser beiden Videos, zur extrahierten Darstellung einer bestimmten Klasse von Handlungen oder zur Darstellung von Rahmen (*Bounding Boxes*) um die Handlungen (siehe Abb. 4.13a).

Die entstandenen Videos werden im Folgenden als *S-SPHAR* Datensatz bezeichnet.

***S-SPHAR* Datensatz** Als *Synthetic SPHAR* Datensatz [125] werden in dieser Arbeit alle durch die eigene Simulation synthetisch erzeugten Videos bezeichnet. Der Name ist eine Anlehnung an den später im Rahmen der Implementierung der Handlungserkennung (siehe Abschnitt 5.1.3.2) zusammengestellten Referenzdatensatz *SPHAR*.

Da während dieser Arbeit Simulation und Handlungserkennung parallel entwickelt wurden und immer wieder prototypische Tests durchgeführt wurden, existieren verschiedene Versionen des synthetischen Datensatzes. In Tabelle 4.2 wird ein Überblick über die verschiedenen Versionen des Datensatzes gegeben.

Version	# Videos	# Klassen	Videos pro Klasse	Speicherbedarf	Länge des Originalvideos	Speicherbedarf des Originalvideos
1	260	9 (HMDB)	8 - 92	40 MB	02:36 min	860 MB
2	696	10 (SPHAR)	3 - 236	168 MB	09:04 min	4,01 GB
3	6.901	10 (SPHAR)	42 - 2328	1,03 GB	48:22 min	12,9 GB

Tabelle 4.2.: Übersicht über den *S-SPHAR* Datensatz.

Die erste Version enthält 260 räumlich und zeitlich zugeschnittene Videos, die anhand der Klassen des *HMDB* [99] Datensatzes beschriftet sind. Die zweite Version verwendet die Klassennamen des *SPHAR* [126] Datensatzes, um die aus einem circa 9 Minuten langen Video der Simulation entstandenen Zuschnitte zu beschriften. Um die Anzahl der zugeschnittenen Videos weiter zu erhöhen und eine weitere Methode des Nachtrainings auszuprobieren, wurde eine dritte Version des Datensatzes erhoben. Für Version 3 wurden rund 48 Minuten Video produziert, das in 6.901 Videos mit Handlungen aufgeteilt wurde.

Die Zuschnitte und Original-Videos lassen sich von der folgenden Website herunterladen:

`https://github.com/AlexanderMelde/S-SPHAR-Dataset`

Die zum Zuschnitt der Videos notwendigen Skripte sind aus rechtlichen Gründen nicht veröffentlicht, werden dem Prüfungsausschuss aber über die im Anhang beschriebene DVD zur Verfügung gestellt.

Die Simulation wird in Full-HD-Auflösung (1920x1080 Pixel) aufgenommen, sodass die 4-in-1 Videos in 4K-Auflösung (3840x2160 Pixel) vorliegen. Eine Stichprobe hat ergeben, dass die meisten Zuschnitte circa 170 Pixel² groß sind. Diese Auflösung ist bewusst vergleichsweise gering gewählt, da auch die Bilder im Anwendungsfall keine höhere Auflösung besitzen, beispielsweise aus Datenschutzgründen oder um Internet-Bandbreite zu sparen. Das Rendern der Simulation wäre problemlos in einer höheren Auflösung möglich, aufgrund der damit verbundenen höheren Rechenzeiten wurde im Rahmen dieser Arbeit aber auf Experimente mit höherer Auflösung verzichtet.

4.8. Übersicht über die Simulation

In diesem Abschnitt wird ein Überblick über alle im Rahmen der Simulation entwickelten Komponenten gegeben.

Für die Erstellung synthetischer Daten und den Zuschnitt der synthetischen Daten in passende Videos wurden ein Unity Projekt und ein Python Skript erstellt.

Das Unity Projekt enthält zwei Unity-Szenen: Eine Unity-Szene zum Test von Animationen und eine Unity-Szene, die die eigentliche Simulation an öffentlichen Plätzen erlaubt. Letztere enthält die vier in Abschnitt 4.3 vorgestellten ortsbezogenen Szenen. An diesen Orten gibt es insgesamt 13 verschiedene Kameraperspektiven, darunter zwei automatisch bewegte Kameras und eine im „freien Flug“ steuerbare Kamera.

Wie bereits beschrieben wurden zahlreiche Maßnahmen zur Erhöhung der Diversität der Szene integriert. Hierzu zählen verschiedene Wetterbedingungen und die Berechnung von Schatten abhängig vom variablen Sonnenstand, eine zufällige Auswahl aus über 30 Charakteren, optionale Bewegungspfade, Fahrzeuge und rund 100 verschiedene für den Anwendungsfall relevante und frei kombinierbare Animationen.

In Form der verschiedenen *S-SPHAR* Datensatz-Versionen konnten über 7.000 Videos zugeschnittener und beschrifteter Handlungen auf öffentlichen Plätzen generiert werden.

Im nächsten Kapitel werden diese synthetischen Videos zur Erkennung von Handlungen eingesetzt.

5. Implementierung und Test der Handlungserkennung

In diesem Kapitel wird die Implementierung der Handlungserkennung sowie die Durchführung der Tests zur Evaluierung der These beschrieben.

Konkret wird hierbei auf die Programmierung der zur Benutzung des Frameworks notwendigen Skripte eingegangen, die Auswahl der Netzarchitektur beschrieben, die Auswahl und die Vorbereitung der Datensätze vorgenommen (inklusive Erstellung des *SPHAR* Datensatzes) und schlussendlich die Durchführung der Tests mit und ohne synthetischen Daten beschrieben sowie ausgewertet.

5.1. Vorbereitung des SlowFast Frameworks

Das SlowFast Framework [59] kann standardmäßig für eine Vielzahl von Anwendungsfällen eingesetzt werden, da es zahlreiche Optionen bietet. In diesem Abschnitt wird eine konkrete Methode zur Handlungserkennung bestimmt und das Framework auf diese eingerichtet.

Auch die in Abschnitt 3.2.2 ausgewählten Datensätze *HMDB* [99] und *SPHAR* [126] müssen vor der Verwendung im Framework noch passend angeordnet werden. Da der *SPHAR* Datensatz im Rahmen dieser Arbeit erstellt wird, wird auch dessen Erstellung in diesem Abschnitt beschrieben.

Abschließend werden einige Details zur Parametrisierung des Trainings des ML-Modells beschrieben, darunter die Netzarchitektur und die verwendete Lernrate.

5.1.1. Erklärung des Zuschnitts

Wie bereits in Abschnitt 2.4 beschrieben können Handlungen mit verschiedenen Detailgraden (Klassifikation oder räumliche und zeitliche Lokalisierung) und Methoden (zum Beispiel videobasiert oder durch Körpersensoren) erkannt werden.

In dem in Abschnitt 1 beschriebenen Anwendungsfall entstehen voraussichtlich zeitlich kontinuierliche Videos (*livestreams*), die zeitgleich mehrere Personen zeigen.

Die bereits vorgestellten Ansätze können nur zeitlich begrenzte Videos verarbeiten, weshalb das Live-Video in der Praxis in mehrere kürzere Videos unterteilt wird, die kontinuierlich hintereinander zur Handlungserkennung gegeben werden.

Um den zu untersuchenden Zeitraum einzuschränken, wird also immer nur eine bestimmte Anzahl der letzten Einzelbilder (*frames*) untersucht. Zum Zeitpunkt t werden beispielsweise die letzten 30 Einzelbilder (also im Intervall $[t-30, t]$) extrahiert (zeitlicher Zuschnitt) und analysiert.

Da in diesen kurzen Videos allerdings immer noch mehrere Personen zeitgleich vorkommen können, wird mithilfe einer Personenerkennung das große Video auch räumlich zugeschnitten (wie bereits im Konzept in Abb. 3.5 auf Seite 53 gezeigt). Der Zuschnitt auf einzelne Personen hat zudem den Vorteil, dass die in Abschnitt 2.1 genannten Anforderungen hinsichtlich der Auflösung zur Anonymisierung umgesetzt werden können.

Dies ist das gleiche Konzept, das auch bei der Erstellung des *S-SPHAR*-Datensatz angewendet wurde. Hier wurde das Skript `generate_video.py` [125] verwendet, um anhand der Segmentierungsmasken *bounding boxes* zu berechnen und die Videos auf diese zuzuschneiden [65].

Diese kleinen räumlich und zeitlich zugeschnittenen Videos werden auch *action tubes* genannt und anschließend mithilfe des Video-Klassifikators ausgewertet. Aufgrund der vorbereiteten Zuschnitte muss der Algorithmus zur Handlungserkennung selber also nur noch klassifizieren und nicht lokalisieren.

5.1.2. Installation und Test des SlowFast Frameworks

Um die Programmierung des Handlungserkenners zu vereinfachen, wird ein Framework zur Klassifikation von Videos verwendet. Anhand des Vergleichs aus Abschnitt 3.3.2 wurde das *SlowFast* Framework ausgewählt, da es vergleichsweise schnell zu erlernen war, noch aktiv entwickelt wird und eine verständliche Struktur für die Datensätze verwendet.

Damit dieses Framework in diesem Projekt zur Erkennung von Handlungen eingesetzt werden kann, sind einige Anpassungen notwendig. Zunächst wurde für die Containerisierungssoftware Docker eine Konfigurationsdatei (*Dockerfile*) geschrieben, die eine plattformübergreifende Nutzung des Frameworks ermöglicht.

Da die offizielle Installationsanleitung und einzelne Skripte Lücken aufgewiesen haben, wurden im Rahmen dieser Arbeit Änderungsvorschläge (*Pull Requests*) für SlowFast eingereicht und in das Original-Projekt integriert. Eine detaillierte Anleitung zur Installation des Frameworks wird in Abschnitt 6.2.1 beschrieben.

Um die Funktionalität des Programms zu testen, wurde mithilfe eines vortrainierten (*pretrained*) Modells, das mit SlowFast mitgeliefert wurde, eine Inferenz auf einem Mini-Datensatz ausgeführt. Der Mini-Datensatz wurde aus wenigen Videos auf Basis des *AVA* Datensatzes [69] erstellt. *AVA* wurde als Basis genutzt, weil SlowFast für diesen Datensatz bereits vordefinierte Konfigurationsdateien mitliefert und so das Finden geeignete Parameter für den ersten Test des Programms vereinfacht wurde.

Die Inferenz mit dem Mini-Datensatz war erfolgreich und hat zur Einarbeitung in das Framework beigetragen.

Eine detaillierte Anleitung zur Installation des SlowFast Frameworks erfolgt im nächsten Kapitel in Abschnitt 6.2.1.

Da nach jeder Epoche ein Zwischenstand des Modells (*Checkpoint*) gespeichert wird, kann das Training zu jedem Zeitpunkt unterbrochen und später fortgesetzt werden. Hierfür wird automatisch je der aktuellste Checkpoint verwendet.

5.1.3. Vorbereitung der Datensätze

Um die Effektivitätssteigerung unabhängig zu messen, werden wie bereits erwähnt verschiedene Referenz-Datensätze (sowohl *HMDB* [99] als auch *SPHAR* [126]) als Grundlage verwendet.

Hierbei ist jeweils darauf zu achten, dass der Datensatz genügend Videos für eine ausreichende Trainings- und Testgrundlage bietet, aber nicht zu groß ist, um auf dem im Rahmen dieser Arbeit verwendeten Computer gespeichert zu werden. Außerdem sollten die Klassen für den Anwendungsfall relevant sein.

In den folgenden Unterabschnitten werden eine detailliertere Beschreibung der beiden Datensätze gegeben und die zur Vorbereitung und gegebenenfalls Erstellung der Datensätze notwendigen Schritte beschrieben.

5.1.3.1. *HMDB* Datensatz

Als erster Datensatz wird *HMDB* [99] verwendet, da dieser in zahlreichen Veröffentlichungen als Referenz verwendet wurde, die Anzahl seiner Videos (6.849) gut ist und in den 51 Klassen auch für den in Abschnitt 1 beschriebenen Anwendungsfall relevante Klassen enthalten sind: *fall on the floor, run, walk, hit something, kick ball, push something, shoot ball, shoot bow, shoot gun, swing baseball bat* und *throw*. Hierbei ist zu beachten, dass es in den Klassen *hit, kick* und *push* zu keiner Gewalt gegenüber anderen Personen kommt, sondern lediglich sportliche Aktivitäten ausgeführt werden, beispielsweise wenn ein Ball beim Baseball geschlagen wird.

Der *HMDB* Datensatz [99] besteht aus 6.894 Videos, wobei jedes Videos einer der in Tabelle 5.1 gezeigten 51 Klassen zugeordnet ist.

Mimik		Körperbewegung		
allein	mit Objekt	allein	mit Objekt	mit Person
smile, laugh, chew, talk	*smoke, eat, drink*	*run, cartwheel, clap hands, climb, climb stairs, dive, fall on the floor, backflip, handstand, jump, pull up, push up, sit down, sit up, somersault, stand up, turn, walk, wave*	*brush hair, catch, draw sword, dribble, golf, hit, kick ball, pick, pour, push, ride bike, ride horse, shoot ball, shoot bow, shoot gun, swing baseball bat, sword exercise, throw*	*hug, punch, kick, shake hands, kiss, sword fight, fencing*

Tabelle 5.1.: Klassen des HMDB51 Datensatzes, gruppiert nach Typ und Interaktionsgrad [99, S. 3].

Die gezeigten Handlungen sind vor allem im Alltag, Zuhause oder beim Sport anzutreffen. Die Videos des Datensatzes stammen von Videoportalen aus dem Internet sowie aus Spielfilmen.

Die Betrachtung der einzelnen Videos des Datensatzes zeigt, dass sich die Kameraperspektiven, Kamerabewegungen und Bildausschnitte sowie die Bildqualität stark zwischen den einzelnen Videos unterscheiden (siehe Abb. 5.1). Dies war laut Aussage der Autoren beabsichtigt, um eine möglichst große Vielfalt von Szenen zu repräsentieren [99, S. 3].

Für die Detektion von Handlungen auf öffentlichen Plätzen sind aufgrund dieser Vielfalt einige der Videos und auch Klassen des HMDB51 Datensatzes weniger gut geeignet, da dort vor allem eine Obersicht-Kameraperspektive und keine Kamera-Bewegungen verwendet werden und nur eine Untermenge der 51 Handlungsklassen für den Anwendungsfall relevant sind.

(a) *kiss* (b) *run*

(c) *hit* (d) *fall on the floor*

Abb. 5.1.: Beispielbilder des HMDB51 Datensatzes [99].

Aufgrund der Popularität des Datensatzes sowie zur besseren Vergleichbarkeit mit anderen Forschungsprojekten soll der HMDB51 Datensatz im Folgenden dennoch zur Evaluation eingesetzt werden, zusätzlich zu dem selbst-erstellten *SPHAR* Datensatz.

Der Datensatz wurde im Rahmen dieser Arbeit in die folgenden *splits* unterteilt: Training (*train*) 60.98% (4.126 Videos), Validierung (*val*) 19.89% (1.346 Videos) und Test 19.13% (1.294 Videos).

Eine solche Einteilung ist beim Test von ML-Modellen üblich, um den Fehler bei neuen, noch nicht beim Training gesehenen Daten zu messen [64, S. 31].

Die Einteilung der Videos erfolgte durch Generierung einer Zufallszahl je Video im Intervall [0.0, 1.0), wobei dieser Wertebereich gewichtet unterteilt wurde, um die *split*-Zugehörigkeit zu bestimmen. Das ist Teil des selbstgeschriebenen Skripts `generate_csv.py`, das zudem auch eine tabellenartige Videoliste mit Dateinamen- und Label-ID Zuordnung generiert. Diese ist für die Verwendung des Datensatzes im *SlowFast* Framework erforderlich.

5.1.3.2. *SPHAR* Datensatz

Der *Surveillance-Perspective Human Action Recognition* (*SPHAR*) Datensatz [126] ist ein im Rahmen dieser Arbeit erstellter Video-Datensatz für die Erkennung menschlicher Handlungen. Sein Hauptzweck ist die Unterstützung der Forschung im Anwendungsbereich der Analyse von Aktivitäten an öffentlichen Orten.

In diesem Forschungsfeld haben die meisten Kameras einen ähnlichen Montagewinkel und eine ähnliche Perspektive, die namensgebend auch als *Surveillance-Perspective* bezeichnet wird. Alle Videos des *SPHAR*-Datensatzes sind aus dieser oder einer ähnlichen Perspektive aufgenommen.

Die Videos wurden aus mehreren Quellen aggregiert, in einen konsistenten Dateityp (H265 HEVC .mp4) konvertiert, geschnitten (zeitlich) und zugeschnitten (räumlich), um jeweils nur eine Aktion zu enthalten, und schlussendlich in 14 Handlungsklassen untergliedert.

Alle Videos des *SPHAR*-Datensatzes sowie die zur Erstellung des Datensatzes genutzten Skripte können aus dem folgenden GitHub Repository heruntergeladen werden:

`https://github.com/AlexanderMelde/SPHAR-Dataset` [126]

Im Folgenden wird die Erstellung dieses Datensatzes beschrieben.

Auswahl geeigneter Videos Für den in Abschnitt 1 vorgestellten Anwendungsfall der öffentlichen Plätze eignen sich einige der in Tabelle 3.1 vorgestellten Datensätze besonders gut, da sich in deren Videos beispielsweise die Perspektive und die vorkommenden Handlungen besonders gut mit denen des Anwendungsfalls decken.

Zunächst wurden vor allem die Datensätze betrachtet, deren Perspektive einer Obersicht entspricht. Anschließend wurde geprüft, welche Klassen von Handlungen in den Datensätzen vorkommen und beschriftet sind.

Aufgrund dieser händischen Recherche konnten 11 Datensätze ermittelt werden, die sich besonders gut für den Anwendungsfall eignen. In Abb. 5.2 werden zur bildlichen Veranschaulichung einige Momentaufnahmen derer Videos gezeigt.

Bei der Sichtung der Videos fiel auf, dass die BIT-Interaction Videos [95] im Gegensatz zu den Videos der anderen Datensätze etwas seitlicher gefilmt wurden, aufgrund der beinhalteten Klassen wie *boxing*, *kick* und *push* wurde der Datensatz aber dennoch als besonders relevant erachtet.

Die Kamera ist bei den meisten Videos statisch, lediglich bei Okutama Action [19] und UCF-Aerial Action [40] wurden Drohnen eingesetzt, wodurch das Kamerabild etwas schwankt.

(a) CAVIAR [61] (b) Live Videos (LV) [106] (c) VIRAT [144]

(d) MEVA [78] (e) CASIA Action [38] (f) UT Interaction [157] (g) Okutama Action [19]

(h) BIT Interaction [95] (i) UCF-Aerial Action [40] (j) UCF-ARG [39] (k) UCF-Crime [170]

Abb. 5.2.: Beispielhafte Bilder aus den Videos der für den Anwendungsfall besonders geeigneten Datensätze.

Download und Konvertierung der Videos Zur Erstellung des *Surveillance-Perspective Human Action Recognition* (*SPHAR*) Datensatzes wurden zunächst alle in Abb. 5.2 gezeigten Datensätzen heruntergeladen. Die Datensätze haben eine Gesamtgröße von 829,4 Gigabyte (GB) und bestehen aus insgesamt 100.907 Dateien, darunter 9.187 Videos. Bei den anderen Dateien handelt es sich um Annotationen, Zusatzinformationen sowie bei einzelnen Datensätzen um redundante Ordner mit Einzelbildern zu den Videos.

Um diese große Datenmenge effizient weiterzuverarbeiten, wurden alle Videodateien in ein einheitliches `.mp4` Format konvertiert. Hierfür wurde der *High Efficiency Video Codec H.265* verwendet, da dieser eine besonders gute Komprimierung bietet. Aufgrund der Größe des MEVA [78] Datensatzes (4.129 Videos, 587,3 GB) wurde dieser vor der Konvertierung gefiltert, um Videos, die keine relevanten Klassen enthalten, gar nicht erst zu konvertieren. Hierzu wurde das Skript `filter_meva.py` eingesetzt [126], das 182 Videos (27,1 GB) mit relevanten Klassen finden konnte.

Durch diese Konvertierung konnte der Speicherplatzbedarf der Videos auf nur noch 45,6 GB reduziert werden. Die Konvertierung nahm bei Verwendung eines Intel i5-7300HQ Prozessors rund 120 Stunden in Anspruch.

Anschließend sollten alle Annotationen in ein einheitliches Format überführt werden. Hierbei bestand die Schwierigkeit, dass die Annotationen nicht nur in verschiedenen Formaten (XML, txt, Ordnerstruktur, ...) vorhanden waren, sondern auch unterschiedliche Genauigkeiten besitzen. In einigen Datensätzen wurden ganze Videos eine Klasse zugewiesen, während bei anderen feingranularer jedes Einzelbild, oder sogar verschiedene Bereiche innerhalb des Einzelbilds unterschieden werden.

Da für die später verwendete Methode Annotationen auf Video-Ebene benötigt werden, sollen die Videos anhand ihrer Ordnerstruktur annotiert werden. Um die Zuordnung einzelner Videos zu ihren Original-Datensätzen zu ermöglichen, wurde mithilfe des Skripts `rename_videos.py` [126] der jeweilige Datensatz-Name in die Dateinamen der Videos eingefügt.

Zuschnitt der Videos Lange Videos mit mehreren Handlungen (MEVA, UT-Interaction, Okutama, VIRAT, UCF Aerial, Live Videos) sollten vor der Einordnung zu einer Handlung in kurze Abschnitte unterteilt und zugeschnitten werden. Hierfür wurde eine Funktion geschrieben, die anhand einer bestimmten Datenstruktur Videos zeitlich und räumlich zuschneiden kann, sowie mehrere als Adapter dienende Skripte, die die verschiedenen Annotationsformate einlesen und an die Funktion weitergeben.

Der MEVA [78] Datensatz wurde beispielsweise mithilfe der beiden Skripte `cut_meva.py` und `crop_meva.py` [126] und der Annotationsdateien des Datensatzes zunächst zeitlich zugeschnitten. So wurden aus den 182 gefilterten und konvertierten MEVA-Videos (18,5 GB) 661 kurze Videos (1 GB). Informationen darüber, in welchen Bildbereichen Handlungen stattfinden, gab es leider nur deutlich weniger. Aus dem MEVA Datensatz konnten letztendlich 25 kurze zugeschnittene für den Anwendungsfall relevante Videos mit einer Dateigröße von circa 1 Megabyte (MB) extrahiert werden. Die geringe Anzahl Videos lässt sich damit erklären, dass die Annotierung des Original-Datensatzes immer noch im Gange ist und noch nicht vollständig ist.

Für den UCF Aerial [40] Datensatz wurde das Skript `crop_ucfaerial.py` [126] geschrieben, das die originalen, teils über mehrere Dateien gestreckten Annotationen im „*VIPER XML*“ Format in ein besser lesbares *JSON* Format umwandelt. Diese Umwandlung war aufgrund der nicht fehlerfreien Original-Annotationen nur nach mehreren Versuchen und der Programmierung von Regeln für Ausnahmefällen möglich, hat schlussendlich aber ermöglicht, die Videos automatisiert sowohl zeitlich als auch räumlich zuzuschneiden.

Aus den ursprünglichen 7 Videodateien (2,6 GB) wurden durch Zuschnitt und Konvertierung 223 kurze Videos (0,14 GB), zunächst gruppiert in die 14 Klassen des UCF Aerial [40] Datensatzes. Bei der Sichtung der generierten Videos ist aufgefallen, dass sich die Dronenvideos oft nicht als Ersatz zur statischen Kamera eignen, da die Kamera so sehr schwankt, dass sich bei den Zuschnitten oft der Bildausschnitt mehr bewegt, als die handelnden Personen. Leider hat sich auch die Qualität der *bounding box*-Annotationen als ungenau herausgestellt. Durch die Kombination dieser beiden Nachteile zeigen viele der kurzen Videos einen zu großen Bildausschnitt und mehrere Personen und Handlungen gleichzeitig.

Die Videos des UT Interaction [157] Datensatzes können nicht nur vollständig, sondern auch bereits zugeschnitten heruntergeladen werden. Für die Eingruppierung war daher kein großes Skript notwendig, sondern lediglich ein Umbenennen der sechs Klassen-IDs zu Klassen-Namen mithilfe simpler Konsolen-Befehle.

Die Konvertierung des VIRAT [144] Datensatzes war vergleichsweise einfach. Das selbst geschriebene Skript `crop_virat.py` [126] extrahiert alle relevanten Daten aus den Event-Annotationen, speichert diese als JSON Datei und schneidet die 329 Original-Videos (6,3 GB) für jede Aktion zeitlich und räumlich zu, sodass eine nach Handlung sortierte Ordnerstruktur mit 1.555 Videos (50 MB) entsteht.

Beim Zuschnitt des aus 43 Videos (19,3 GB) bestehenden Okutama [19] Datensatzes konnten 2.903 kurze und kleine Videos mit zusammen 3,2 GB gewonnen werden. Dabei ist aufgefallen, dass die *Bounding Box* Annotationen nur in wenigen Videos genau waren, weshalb händisch einige fehlerhaften Videos aussortiert wurden.

Eine detaillierte Beschreibung der verschiedenen Skripte und derer Parameter befindet sich in der Readme-Datei im Skripte-Ordner des Code-Repositories [126], das auch auf der DVD im Anhang beigefügt ist.

Bei der Sichtung der von den Herausgebern eingruppierten kürzeren Videos wurde festgestellt, dass diese oft nicht auf die Handlung zugeschnitten sind. Dieses Problem trat vor allem bei den Videos des CAVIAR [61] und UCF Crime [170] Datensatzes auf. Da hier außer den Klassen keine Informationen diesbezüglich vorhanden sind, müsste der Zuschnitt händisch erfolgen, was aufgrund des hohen Aufwands nicht im Rahmen dieser Arbeit gemacht wurde.

Durch diese Ungenauigkeit in den Original-Annotationen sind nun nur ein Teil der Videos des *SPHAR*-Datensatzes auf Einzelpersonen und deren Handlungen zugeschnitten, während andere Videos mehrere Personen und zum Teil mehrere Handlungen zeigen. Darüber hinaus ist aufgefallen, dass die Videos des UCF Crime [170] Datensatzes teilweise Intros oder Wasserzeichen beinhalten, die zu einem ungewollten Overfitting führen könnten. Um dem entgegenzuwirken, muss darauf geachtet werden, dass in jeder *SPHAR*-Klasse Videos verschiedener Original-Datensätze vorkommen.

Wird einmal für eine Klasse kein Video gefunden, so ist es möglich, dass in den Videos zwar weitere Handlungen vorkommen, aber nicht gelabelt sind.

Ergebnis Die für den *SPHAR* Datensatz verwendeten Klassen sowie die finale Zusammensetzung des Datensatzes sind in Tabelle 5.2 dargestellt.

Klasse		Beschreibung	# Videos	Datengrundlage
	hitting	eine Person schlagen	401	[95; 38; 61; 39; 170; 157]
	kicking	eine Person treten	120	[95; 170; 157]
	falling	hinfallen, stolpern	123	[38; 61; 106; 19]
	vandalizing	vandalieren	209	[38; 106; 39; 170]
	panicking	in Panik ausbrechen	52	[38; 106]
	sitting	sitzen	621	[61; 78; 19; 144]
	walking	gehen	2800	[38; 61; 19; 40; 39; 144]
	running	rennen	516	[38; 19; 40; 39; 144]
	neutral	neutral/untätig sein	2166	[95; 61; 19; 40; 39; 157; 144]
	luggage	ein Gepäckstück zurücklassen	8	[61; 78]
	stealing	einen Diebstahl begehen	418	[38; 106; 170]
	murdering	einen Mord begehen / Schusswaffe verwenden	52	[106; 170]
	carcrash	an einem Autounfall beteiligt sein	157	[106; 170]
	igniting	ein Feuer entzünden	101	[170]
SPHAR			7759	

Tabelle 5.2.: Übersicht über die Klassen des *SPHAR* Datensatzes.

Zahlreiche Klassen, die nicht in der Tabelle sind, wurden zu den genannten vereint, wenn die Unterschiede gering waren oder für unseren Fall als eine Klasse betrachtet werden konnten. So wurden beispielsweise die Klassen „crouch“ (in der Hocke laufen oder in die Hocke gehen) und „follow person“ (einer Person folgen) des CAVIAR Datensatzes zu der Klasse „walking“ (gehen) hinzugezählt, da in den Videos gegangen wird, nur eben beispielsweise mit einer kurzen Pause zum Schuhe binden. Für den Anwendungsfall kann das daher als eine Variation der „walking“ Klasse gesehen und daher vereint werden.

Es war zudem schwer, Videos von Klassen der Handlung „fight“, die echte Kämpfe zeigt, in die Klassen „hit“ oder „kick“ zu unterteilen, da in einigen Kämpfen beides vorkommt. Die Grenzen verschmelzen hier also ein wenig, was für den Anwendungsfall aber nicht so schlimm ist.

Datensatz		# Videos mit relevanten Handlungen														Lizenz
CAVIAR	[61]	4	0	3	0	0	1	61	0	5	5	0	0	0	0	Public
CASIA	[38]	12	0	36	14	48	0	204	96	0	0	11	0	0	0	Z
UCF-Aerial	[40]	0	0	0	0	0	0	71	8	64	0	0	0	0	0	NC
UCF-Crime	[170]	100	50	0	50	0	0	0	0	0	0	400	50	150	100	NC
UT-Interact.	[157]	40	20	0	0	0	0	0	0	60	0	0	0	0	0	MIT
BIT-Interact.	[95]	100	50	0	0	0	0	0	0	250	0	0	0	0	0	NC
Live Videos	[106]	1	0	1	1	4	0	0	0	1	0	7	2	7	1	CC-BY-NC
UCF-ARG	[39]	144	0	0	144	0	0	288	288	432	0	0	0	0	0	NC
VIRAT	[144]	0	0	0	0	0	208	1111	22	214	0	0	0	0	0	C
MEVA	[78]	0	0	0	0	0	22	0	0	0	3	1	0	0	0	CC-BY-4
Okutama	[19]	0	0	83	0	0	390	1064	102	1170	0	0	0	0	0	CC-BY-NC-3
SPHAR	[126]	401	120	123	209	52	621	2800	516	2166	8	418	52	157	101	mehrere

Tabelle 5.3.: Verteilung der Videos des *SPHAR* Datensatzes anhand Klasse und Quelle. Eine Erklärung der Piktogramme befindet sich in Tabelle 5.2 auf Seite 84. C:Kommerzielle Nutzung erlaubt, NC:nicht-kommerziell/nur Forschung erlaubt, Z:Nutzung nur nach expliziter Zustimmung des Herausgebers erlaubt.

In Tabelle 5.3 wird die Anzahl Videos diese Datensätze je Klasse näher miteinander verglichen sowie die Lizenzen zu den jeweiligen Original-Datensätzen genannt.

MEVA wird fortlaufend weiter annotiert, die Zahlen können also höher sein, wenn mit zusätzlichen Annotationen gearbeitet wird.

Der *SPHAR* Datensatz besteht also aus 14 Klassen und 7.759 Videos und kann unter dem folgenden Link von GitHub heruntergeladen werden:

`https://github.com/AlexanderMelde/SPHAR-Dataset` [126]

Die Videos stammen aus 11 verschiedenen Datensätzen und haben eine Gesamtgröße von 6,2 GB. Je Klasse existieren zwischen 8 und 2800 Videos.

Damit der Datensatzes im SlowFast Framework verwendet werden kann, wurde dieser mit dem `generate_csv.py` Skript ebenfalls in drei Teile (*splits*) unterteilt. Der 60.41% große Trainings-Split enthält 4.678 Videos und wird zum Trainieren des Modells verwendet.

Um während des Trainings die Genauigkeit zu prüfen, wird ein 19.69% (1.525 Videos) Validierungs-Split verwendet. Für einen finalen Test wird ein weiterer 19.90% (1.541 Videos) großer Test-Split verwendet.

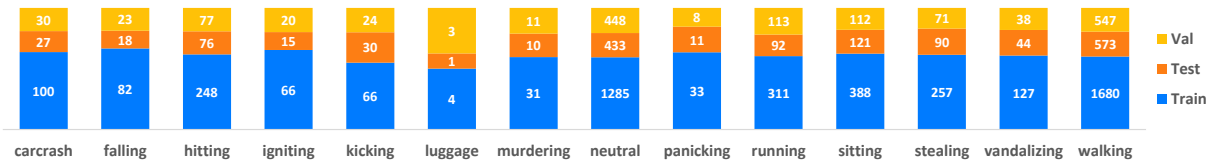


Abb. 5.3.: Verteilung der Handlungsklassen in den Splits des *SPHAR*-Datensatzes. Gezeigt wird jeweils die Anzahl der Videos pro Split und Handlungsklasse.

In Abb. 5.3 ist erkennbar, dass sich die Verteilung der einzelnen Handlungsklassen in jedem Split ähnlich ist. Dies ist eine wichtige Voraussetzung für verlässliche Aussagen beim Validieren.

5.1.4. Wahl einer Netzarchitektur und Lernrate

Die genauen Parameter des Trainings und der Inferenz werden bei SlowFast in einer Konfigurationsdatei definiert. Eine der im Rahmen dieser Arbeit genutzten Konfigurationsdateien wird im Anhang gezeigt.

Bei der Entwicklung eines Modells für maschinelles Lernen kann jeder einzelne Parameter für den Erfolg eines Modells ausschlaggebend sein. In dieser Arbeit geht es vor allem um die Messung der relativen Verbesserung mit und ohne synthetischen Daten, weshalb nicht zu viel Zeit in das Ermitteln idealer Parameter investiert wird.

Dennoch wurden die Parameter selbstverständlich mit Bedacht gewählt. Zur Optimierung wird das einleitend beschriebene *Stochastic Gradient Descent* (*SGD*) Verfahren angewendet.

Exemplarisch werden die Wahl der Netzarchitektur sowie der Lernrate im Folgenden genauer beschrieben.

Netzarchitektur Wie in den Grundlagen beschrieben bestehen moderne tiefe neuronale Netze aus vielen hintereinander gereihten Schichten verschiedener Typen und Größen. Eine konkrete Auswahl und Anordnung dieser Schichten wird Netzarchitektur genannt. Zahlreiche Veröffentlichungen betrachten ausschließlich die Erforschung neuer Netzarchitekturen für spezielle Anwendungsfälle [33; 172].

Einleitend wurden *LSTM*s als prädestiniert für die Auswertung von Zeitverläufen vorgestellt. Carreira und Zisserman haben 2017 eine *CNN* Netzarchitektur entwickelt, die die Genauigkeiten von *LSTM*s dieser Zeit überbieten konnte [33].

In dieser Arbeit wird genau diese Architektur als Grundlage für die Handlungserkennung verwendet. Neben der auf Basis dieser Tests hohen Genauigkeit ist einer der Gründe hierfür, dass diese Netzarchitektur bereits in das SlowFast Framework integriert ist und daher weniger Aufwand zur Neuentwicklung erforderlich ist [59].

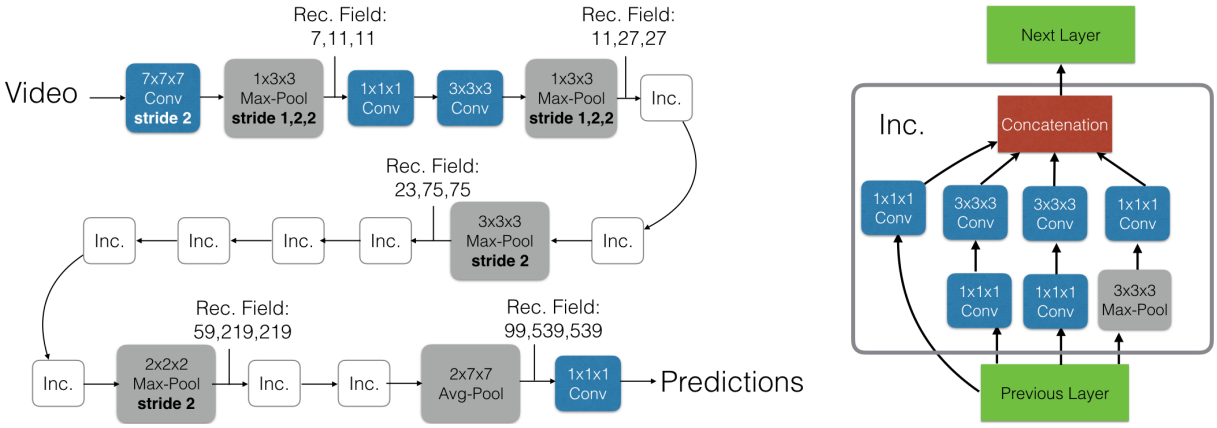
Diese „i3D“ genannte Architektur basiert auf der populären Inception [172] Netzarchitektur, die als Meilenstein in der Entwicklung von *CNN* gilt. Während frühere Ansätze höhere Genauigkeiten durch Hinzunahme immer weiterer Faltungsschichten realisierten, wurden bei Inception [172] stark optimierte und komplexe „Inception Module“ eingesetzt, die neben den Faltungsschichten auch Schichten zur Filterung und Abstraktion (*Pooling*) einsetzen.

In Quellcode 5.4 werden sowohl die gesamte Inception-Architektur als auch der Aufbau eines der Submodule (*Inc.*) gezeigt.

Das in SlowFast konkret verwendete Model wird im Anhang gezeigt.

Ursprünglich wurde bei i3D [33] der Kinetics [91] Datensatz genutzt. Dieser besteht aus bis zu 650.000 Videos und 700 Klassen [35]. Er ist für die Verwendung in dieser Arbeit aufgrund der Größe, der Perspektiven und der Auswahl der Klassen nicht geeignet.

Wie bereits beschrieben, werden stattdessen *HMDB* [99] und *SPHAR* [126] eingesetzt. Da das SlowFast Framework [59] diese standardmäßig nicht kennt, wurde jeweils ein Skript zum Einlesen der Datensätze geschrieben und in das Framework eingebunden.



(a) Inflated Inception V1. (b) Inception Modul (*Inc.*).

Abb. 5.4.: Architektur des i3D-Ansatzes (aus [33, S. 5]).

Der in SlowFast integrierte Kinetics Datensatz hat eine diesen Datensätzen ähnliche Struktur. Daher genügt es, im Ordner `slowfast/datasets` die Datei `kinetics.py` zu duplizieren und sämtliche Vorkommen von Kinetics in der neuen Datei und in deren Dateinamen durch *HMDB* beziehungsweise *SPHAR* zu ersetzen. Anschließend müssen die neuen Dateien noch in das Projekt eingebunden werden. Hierfür sollten in der Datei `slowfast/datasets/__init__.py` die beiden Import-Befehle ergänzt werden, beispielsweise mit `from .sphar import` SPHAR.

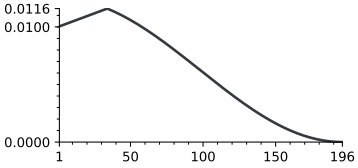
Nach diesen kurzen Anpassungen können die Datensätze aus der Konfigurationsdatei heraus referenziert werden, sodass beliebig mit verschiedenen in der passenden Struktur vorliegenden Datensätzen experimentiert werden kann.

Lernrate Die Lernrate eines neuronalen Netzes bestimmt die Schrittweite, die beim Gradientenabstiegsverfahren benutzt wird, um Gewichte anzupassen (siehe Abschnitt 2.2.1.2). Je kleiner die Lernrate, desto genauer trifft man das zu ermittelnde Minimum, allerdings sind auch viel mehr Iterationen notwendig, um dieses zu erreichen.

Zum Training wurde eine Basis-Learningrate $b = 0.0125$ gewählt und wie in Abb. 5.5a gezeigt mit einem *Warmup*- und *Decay*-Verfahren kombiniert. Das *Warmup*-Verfahren erhöht die Lernrate während der ersten Epochen linear. Das *Warmup* (deutsch: aufwärmen) findet in den Epochen 1 bis $w = 34$ statt und beginnt bei der Lernrate $s = 0.01$. Zum Abschluss des *Warmup* wird der Startwert des nun folgenden *Decay*-Verfahrens erreicht. Dieses lässt die Lernrate anhand einer Cosinus-Funktion abfallen. Eine derartige *decay policy* (auch *learning rate schedule* genannt) kann zu einer Verringerung der Fehlerrate beim Training führen [113].

$$LR(e) = \begin{cases} \frac{e*LR(w)-s}{w} + s & \text{für } e < w \\ 0.5 * b * \left(1.0 + \cos\left(\frac{\pi * e}{e_{max}}\right)\right) & \text{sonst} \end{cases}$$

(a) Berechnung der Lernrate in Epoche e (von insgesamt e_{max} Epochen) bei der Basis-Lernrate b und einer Warm-Up Phase beginnend mit der Lernrate s für die Epochen 1 bis w.



(b) Während der Trainings-Epochen verwendete Lernrate als Graph.

Abb. 5.5.: Zum Training verwendete Lernrate.

5.2. Training und Test der Handlungserkennung

Auf Basis von während dieser Arbeit gewonnenen Erkenntnissen werden mehrere Tests mit verschiedenen Datensätzen durchgeführt.

In diesem Abschnitt werden zunächst der Test mit dem *HMDB*-Datensatz und anschließend die Tests mit dem *SPHAR*-Datensatzes beschrieben.

5.2.1. Der Test mit *HMDB*

Der im Rahmen dieser Arbeit zeitlich als erstes ausgeführte Test verwendet den *HMDB*-Datensatz [99]. Dieser wird in zahlreichen Veröffentlichungen verwendet und muss, im Gegensatz zum *SPHAR*-Datensatz, nicht erst selbst erstellt werden, weshalb er sich für einen ersten Test gut eignet.

Es soll getestet werden, ob die Hinzunahme von synthetischen Daten ein auf *HMDB* Videos trainiertes Modell zur Erkennung von Handlungen verbessern kann. Hierfür wird zunächst ein Modell rein auf echten Videos trainiert und dessen Genauigkeit gemessen. Dieses Modell wird *baseline* genannt. Anschließend wird aufbauend auf diesem Modell weiter trainiert, wobei sich dann diesmal zusätzliche synthetisch erzeugte Daten in den Trainingsvideos befinden.

5.2.1.1. *Baseline* für *HMDB*

Zum Trainieren des *baseline* Modells werden keine synthetische Daten eingesetzt, sondern nur die Videos des *HMDB* Datensatzes. Das Training wurde unter Verwendung einer NVIDIA GeForce GTX 1070 Grafikkarte mit 8GB Speicher durchgeführt und nahm etwa 20 Stunden in Anspruch.

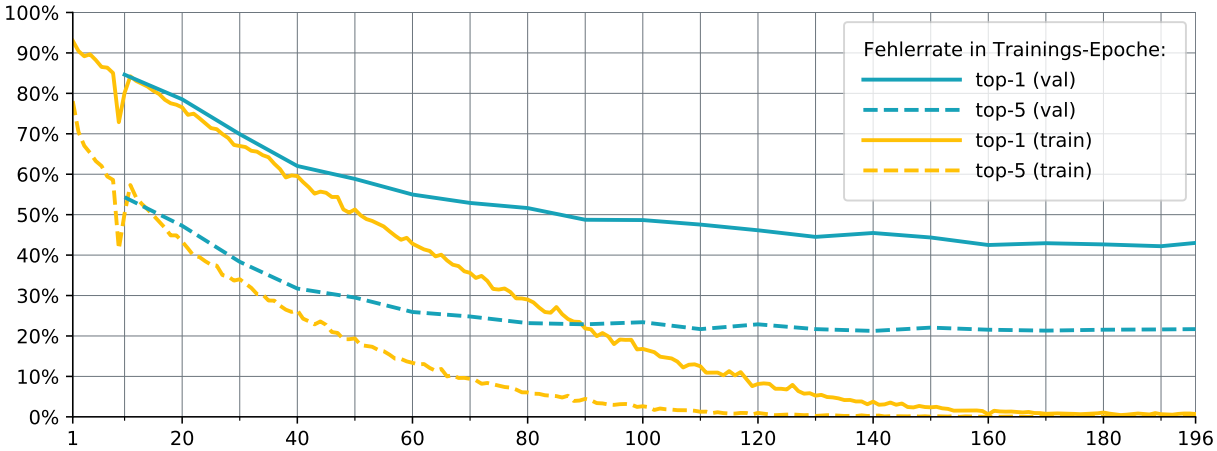


Abb. 5.6.: Fehlerraten beim initialen Training mit *HMDB*. *Top-n* bedeutet die Klasse des Videos war nicht unter den *n*-wahrscheinlichsten Vorhersagen. Es wurde mit Mini-Batches des Trainings-Split (train) sowie des des Validierungs-Split (val) getestet.

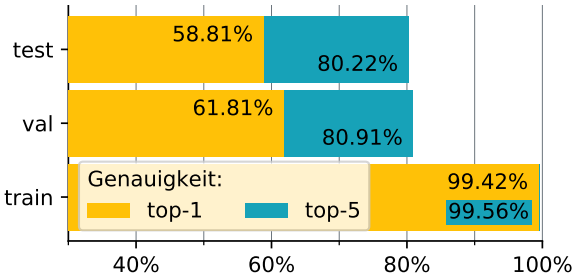
Während des Trainings wurde alle 10 Epochen eine Inferenz auf einem Mini-Batch des Validierungs-Split ausgeführt, um die Genauigkeit und Fehlerrate des bis zu dieser Epoche trainierten Modells festzustellen. Ein Mini-Batch ist eine kleine zufällig gewählte Untermenge eines Datensatzes. Darüber hinaus wurde in jeder Epoche die Fehlerrate bei Inferenz mit einem Mini-Batch des Trainings-Splits gemessen. Der Verlauf der Fehlerraten kann zur Evaluation der Effektivität des Trainings

genutzt werden und ist ein wichtiger Hinweis darauf, wie hilfreich das Trainieren weiterer Epochen wäre.

Wie in Abb. 5.6 dargestellt, sind alle Fehlerraten während des Trainings, von kleineren Schwankungen abgesehen, kontinuierlich gesunken. Dies deutet darauf hin, dass das Training effektiv war.

Es ist wichtig, dass die Validierungs-Fehlerraten mit zunehmender Anzahl von Epochen und sinkender Trainings-Fehlerrate nicht wieder steigen, da das Modell sonst überangepasst auf die Trainingsdaten wäre (*Overfitting*). Das ist hier nicht der Fall.

Um zu prüfen, ob die Inferenz mit Mini-Batches aussagekräftig ist, wurden nach dem Training Tests mit allen Videos der verschiedenen Splits (siehe Abschnitt 5.1.3.1) durchgeführt. So wurden die in Abb. 5.7 gezeigt finalen Genauigkeiten des Modells ermittelt.



(a) Genauigkeit (*Accuracy*).

$$Error = 1 - Genauigkeit$$

Datensatz	Top1-Error	Top5-Error
test split	41.19%	19.78%
val split	38.19%	19.09%
train split	0.58%	0.44%

(b) Fehlerrate (*Error*).

Abb. 5.7.: Ergebnisse des Tests des auf HDMB trainierten Modells unter Verwendung verschiedener Splits als Testvideos. *Top-n* Genauigkeit bedeutet, die korrekte Klasse des Videos muss unter den *n*-wahrscheinlichsten Vorhersagen des Modells enthalten sein, damit die Vorhersage als Erfolg gezählt wird.

Die Ergebnisse des Tests des Train- und Val-Splits sind, wie zu erwarten war, den jeweiligen Genauigkeiten der Mini-Batch Tests der letzten Epochen aus Abb. 5.6 sehr ähnlich. Das weist darauf hin, dass auch die anderen Mini-Batches als repräsentativ für den ganzen Split angesehen werden können.

Die hohe Genauigkeit bei Verwendung des Trainingsdatensatzes kommt daher, dass der Algorithmus diese Videos bereits beim Training gesehen hat und dementsprechend besonders leicht einordnen kann.

Diese Genauigkeiten dienen in dieser Arbeit nun als Referenzwerte. Im Folgenden werden nach dem Hinzufügen der synthetischen Daten zum Modell erneut Genauigkeiten berechnet. Sind diese höher, so hat sich die Hinzunahme synthetischer Daten gelohnt.

5.2.1.2. Nachtrainieren mit *S-SPHAR*-1

Im zweiten Teil des Tests mit dem *HMDB* Datensatz wird die erste Version des synthetischen *S-SPHAR* Datensatzes [125] zu den Trainingsvideos hinzugegeben und erneut von vorne trainiert.

Die Hinzunahme der synthetischen Videos aus *S-SPHAR*-1 erfolgt technisch gesehen über die selben Skripte, die auch den *HMDB*-Datensatz für das Framework nutzbar gemacht haben. Das Skript `generate_csv.py` wurde so konfiguriert, dass es aus *S-SPHAR* nur einen einzigen Split erstellt und die Klassen-IDs des *HMDB*-Datensatzes verwendet. So können anschließend die Inhalte beider Split-Textdateien untereinander kopiert werden, um eine große Split-Textdatei zu erstellen.

Die zusätzlichen synthetischen Videos können in die selben Ordner kopiert werden, die auch die *HMDB*-Videos enthalten. Aufgrund des Namenschemas der Videos kommt es zu keinen doppelten Dateinamen. Der Trainings-Split besteht in diesem Fall nun aus 168 weiteren Videos und damit aus insgesamt 4.294 Videos. Die Test- und Validierungs-Splits sind gleich geblieben.

Die zuvor generierten Ausgabedateien wie zum Beispiel Modell-Checkpoints sollten aus dem Projektordner heraus an einen Ort zur späteren Verwendung verschoben werden.

Um das neue Training zu beginnen, ist kein neu-bauen des Dockerfiles notwendig. Es genügend ein einfacher Start, wie beim vorherigen Versuch.

In Abb. 5.8 ist die Kurve des Trainingsverlaufs mit zusätzlichen synthetischen Daten zu sehen. Sie ist sehr ähnlich wie der erste Trainingsverlauf.

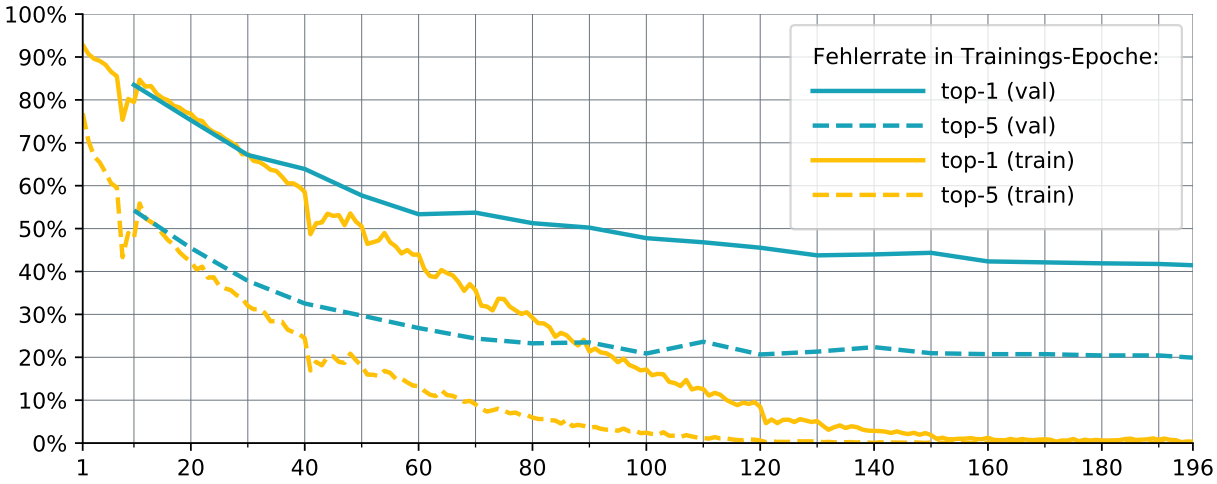
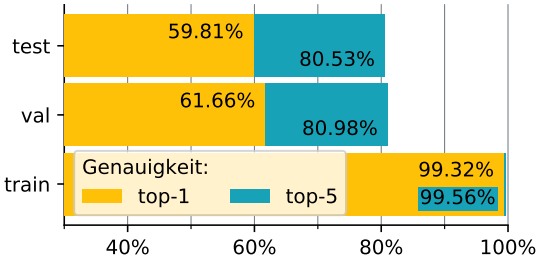


Abb. 5.8.: Fehlerraten beim Retraining von *HMDB* mit *S-SPHAR*-1. Die Kurven haben einen sehr ähnlichen Verlauf wie die in Abb. 5.6 gezeigten.

Auch die mit verschiedenen Splits getesteten Genauigkeiten (siehe Abb. 5.9) ähneln denen der *baseline* Tests.

Eine Auswertung dieser Werte erfolgt im folgenden Abschnitt.



(a) Genauigkeit (Accuracy).

$$Error = 1 - Genauigkeit$$

Datensatz	Top1-Error	Top5-Error
test split	40.19% (-1.00%)	19.47% (-0.31%)
val split	38.34% (+0.15%)	19.02% (-0.05%)
train split	0.68% (+0.10%)	0.44% (=)

(b) Fehlerrate (Error) (im Vergleich zum Originalen Training).

Abb. 5.9.: Test-Ergebnisse des mit *S-SPHAR*-1 nachtrainierten *HMDB* Modells.

5.2.1.3. Auswertung des Test mit *HMDB*

Mit diesem Experiment sollte getestet werden, ob die Hinzunahme von *S-SPHAR*-1 eine Genauigkeitssteigerung auf dem *HMDB*-Validierungssplits bewirkt, verglichen mit einem reinen *HMDB* Training.

Das Experiment hat gezeigt, dass sich die Genauigkeiten nur um wenige Promille ändern und damit je nur wenig gestiegen oder gesunken sind.

Der Grund für diese nur geringe Änderungen wird in der Art der Datensätze vermutet. Während *HMDB* aus verschiedenen Perspektiven gefilmt wurde, die für Spielfilme oder YouTube-Videos geeignet sind, wurden die Videos des *S-SPHAR*-1 aus Obersicht aufgenommen. Darüber hinaus enthält *HMDB* zahlreiche Klassen, die durch *S-SPHAR*-1 nicht nachgebildet werden konnten, beispielsweise Handlungen, die in Innenräumen ausgeführt werden.

Im nächsten Abschnitt wird geprüft, ob der *S-SPHAR* Datensatz bei einem anderen Datensatz, der näher an dem Anwendungsfall dieser Arbeit liegt, zu einer Verbesserung der Genauigkeit führen kann.

5.2.2. Der Test mit *SPHAR*

Da der *HMDB*-Datensatz [99] zwar bekannt ist, aber nicht zu unserem Anwendungsfall passt, werden die gleichen Tests im Folgenden auch für den *SPHAR*-Datensatz [126] durchgeführt.

5.2.2.1. *Baseline* für *SPHAR*

Analog zum Vorgehen für den *HMDB* Datensatz wurde auch für *SPHAR* zunächst ohne jegliche synthetische Daten ein Modell trainiert und getestet. Hierfür wurden die in Abschnitt 5.1.3.2 beschriebenen Splits verwendet. Das Training nahm etwa 6 Tage in Anspruch.

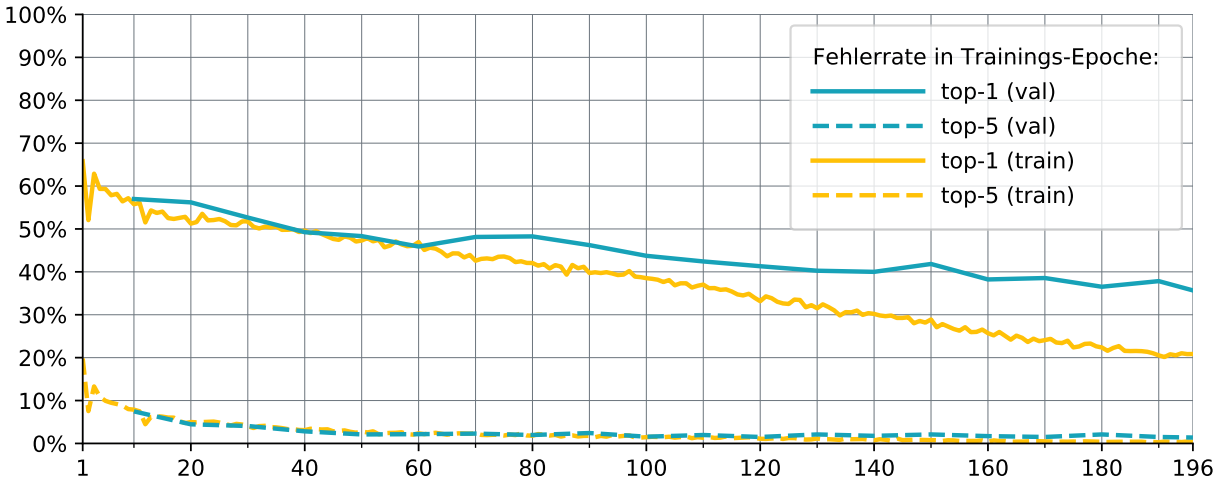
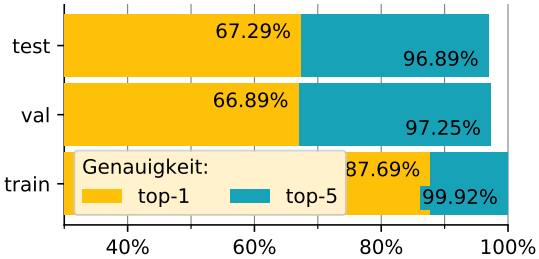


Abb. 5.10.: Fehlerraten beim initialen Training mit *SPHAR*. *Top-n* bedeutet, die Klasse des Videos war nicht unter den *n*-wahrscheinlichsten Vorhersagen. Es wurde mit Mini-Batches des Trainings-Splits (train) sowie des Validierungs-Splits (val) getestet.

In Abb. 5.10 sind die Fehlerraten während des Trainings dargestellt, in Abb. 5.11 die finalen Testergebnisse bei Inferenz der einzelnen Splits.



(a) Genauigkeit (*Accuracy*).

Datensatz	Top1-Error	Top5-Error
test split	32.71%	3.11%
val split	33.11%	2.75%
train split	12.31%	0.08%

(b) Fehlerrate (*Error*).

Abb. 5.11.: Test-Ergebnisse des auf *SPHAR* trainierten Modells unter Verwendung verschiedener Splits als Testvideos.

Im Vergleich zum Training und Test des *HMDB* Datensatzes fällt auf, dass die Top-1 Genauigkeit des Trainings-Datensatz einen weniger hohen Wert erreicht. Dafür sind die Genauigkeiten bei den Test und Train Splits höher, was darauf hinweist, das die Splits gut gewählt wurden und es zu keinem *Overfitting* kam.

Die berechneten Genauigkeiten werden im Folgenden verwendet, um die Effektivität der verschiedenen Methoden zum Nachtrainieren zu messen.

5.2.2.2. Nachtrainieren mit *S-SPHAR*-2

Für das Trainieren mit zusätzlichen synthetischen Daten werden diesmal zwei Methoden getestet. Neben der für *HMDB* getesteten Methode, bei der das komplette Training von Epoche 1 beginnend neu ausgeführt wird, wird diesmal ein Retraining „on top“ ausgeführt. Bei dieser Methode werden ab der letzten Episode des *baseline*-Trainings synthetische Daten hinzugefügt und bis zu einer höheren Epochen weiter trainiert. Diese Methode spart Zeit, da die ersten Epochen nicht erneut trainiert werden müssen.

Als synthetische Daten werden die Videos von *S-SPHAR*-2 verwendet.

Methode 1 (*on top*) Der in Abb. 5.12 gezeigte Trainingsverlauf streckt sich im Gegensatz zu den bisherigen Experimenten über 250 statt 196 Epochen. Der nach Epoche 196 entstehende vorübergehende Anstieg der Fehlerrate im Trainingssplit entsteht durch die Hinzunahme der bisher unbekannten synthetischen Daten in den Trainingssplit.

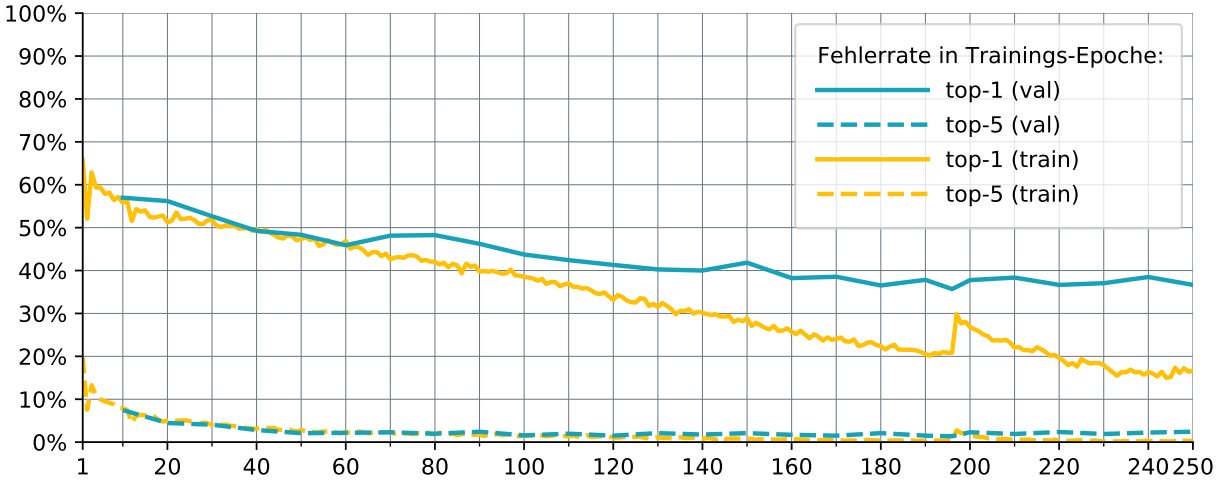
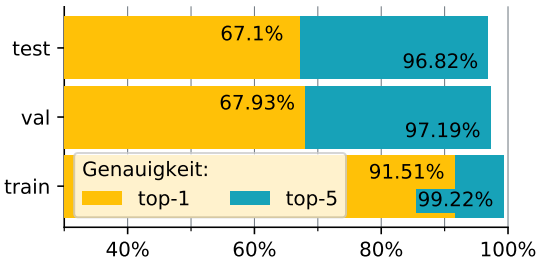


Abb. 5.12.: Fehlerraten beim Retraining mit *S-SPHAR*-2 und Methode 1 (*on top*). Bis Epoche 196 entspricht der Graph dem Verlauf von Abb. 5.10 und anschließend wurden ab Epoche 197 die synthetischen Daten zum Trainingsdatensatz hinzugefügt.

Die in Epoche 250 des Trainings gemessenen Genauigkeiten werden in Abb. 5.13 gezeigt.



(a) Genauigkeit (*Accuracy*).

Datensatz	Top1-Error	Top5-Error
test split	32.90% (+0.19%)	3.18% (-0.07%)
val split	32.07% (-1.04%)	2.81% (+0.06%)
train split	8.49% (-3.82%)	0.78% (+0.70%)

(b) Fehlerrate (*Error*) (im Vergleich zum Originalen Training).

Abb. 5.13.: Test-Ergebnisse des mit *S-SPHAR*-2 *on top* nachtrainierten Modells.

Es ist zu erkennen, dass mit dieser Methode die Fehlerrate der Handlungserkennung nicht signifikant gesenkt werden konnte. Im nächsten Abschnitt wird daher ein weiterer Versuch unternommen, in der die gleiche Methode wie beim Nachtrainieren des *HMDB* Datensatzes verwendet wird.

Methode 2 (*full*) In diesem Abschnitt erfolgt das Hinzufügen von synthetischen Daten wieder in Episode 1. Während der rund einwöchigen Trainingszeit wurden wie in den Vorgänger-Experimenten regelmäßig die Fehlerraten gemessen (siehe Abb. 5.14)

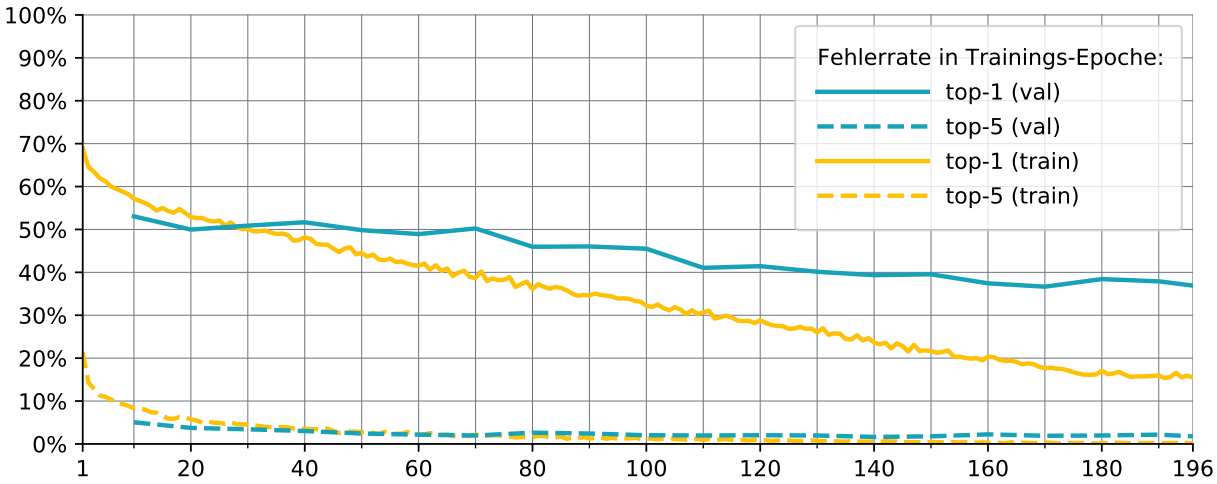
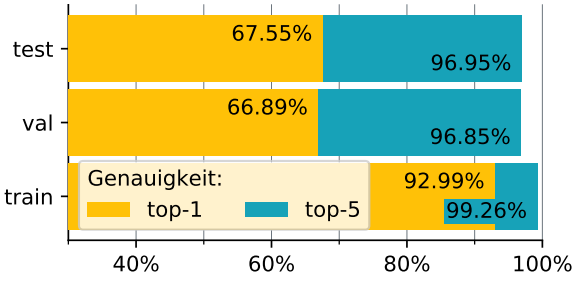


Abb. 5.14.: Fehlerraten beim Retraining mit *S-SPHAR-2* und Methode 2 (*full*). Die Kurven haben einen sehr ähnlichen Verlauf wie die in Abb. 5.10 gezeigten.

Die Fehlerrate ähnelt sehr dem Verlauf beim Trainieren des *SPHAR baseline* Modells. Die synthetischen Daten scheinen keinen großen Einfluss auf das Training des Modells zu haben.

Diese Vermutung wird auch durch die in Abb. 5.15 gezeigten Ergebnisse bestätigt, da sich die Genauigkeiten auch mit dieser Methode kaum verändert haben.



(a) Genauigkeit (Accuracy).

$$Error = 1 - Genauigkeit$$

Datensatz	Top1-Error	Top5-Error
test split	32.45% (-0.26%)	3.05% (-0.06%)
val split	33.11% (-0%)	3.15% (+0.4%)
train split	7.01% (-5.30%)	0.74% (+0.66%)

(b) Fehlerrate (Error) (im Vergleich zum Originalen Training).

Abb. 5.15.: Test-Ergebnisse des mit *S-SPHAR-2 full* nachtrainierten Modells.

In diesem sowie den vorangehenden Versuchen wurden die synthetischen Daten stets mit den realen Daten vermischt. Im nächsten Abschnitt wird geprüft, ob das getrennte Trainieren von zunächst rein synthetischen Daten und anschließenden Realdaten zum Erfolg führt.

5.2.2.3. Nachtrainieren mit *S-SPHAR*-3

S-SPHAR-3 ist der größte im Rahmen dieser Arbeit verwendete synthetische Datensatz. Dieser soll in Kombination mit einer Technik eingesetzt werden, bei der zuerst ausschließlich der synthetische Datensatz fürs Training verwendet wird. Anschließend soll dieses Modell dann mit realen Daten spezialisiert werden (*fine tuning*).

In diesem Experiment wird mit *S-SPHAR*-3 bis Epoche 100 trainiert. Anschließend werden alle synthetischen Videos aus dem Trainings-Split entfernt und durch die Trainings-Videos des *SPHAR* Datensatzes ersetzt. Als Validierungs- und Testvideos werden zu jedem Zeitpunkt die Videos der jeweiligen *SPHAR* Splits verwendet.

Die Kurve der Fehlerrate bei diesem Training wird in Abb. 5.16 gezeigt.

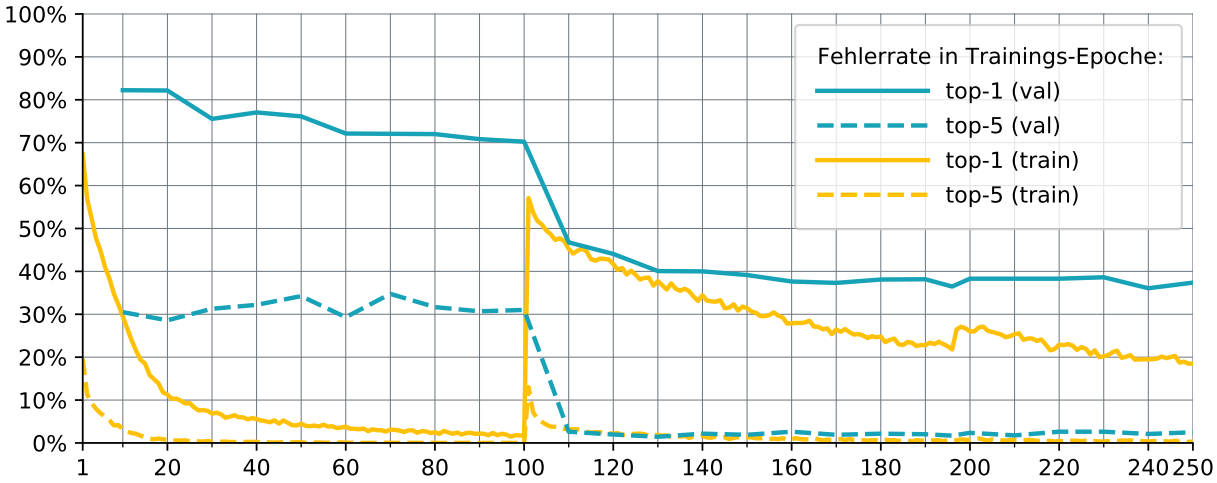


Abb. 5.16.: Fehlerraten beim Training mit *S-SPHAR*-3 und *SPHAR*. Bis Epoche 100 wurde rein auf synthetischen Daten trainiert, danach ausschließlich mit den realen Daten.

Es ist zu erkennen, dass die Validierungs-Genauigkeit mittels Mini-Batches bei rein synthetischen Daten bei rund 30% liegt. Nach der Verwendung der realen Daten steigt diese Genauigkeit innerhalb 30 Epochen auf circa 60% und behält diesen Wert anschließend bei. Dieser Wert ist ein wenig niedriger als die *baseline* Genauigkeit und sehr ähnlich zu den vorhergegangenen Versuchen. Ab Episode 160 verringert sich nur noch die Trainings-Genauigkeit signifikant, was auf eine Überanpassung (*overfitting*) hindeutet. Für einen produktiven Einsatz würde es also Sinn machen, das Training vorzeitig zu beenden.

Die Genauigkeit scheint sich auch mit dieser Methode und dem größeren synthetischen Datensatz nicht steigern zu lassen. Die Reihenfolge des Nachtrainierens scheint für die resultierenden Genauigkeiten nicht besonders relevant zu sein.

Auf eine Berechnung der Genauigkeiten mithilfe der vollständigen Splits wird in diesem Experiment verzichtet, da die Genauigkeiten der Mini-Batches diesen Tests stets sehr ähnlich waren und so Zeit eingespart werden kann. Stattdessen wird noch geprüft, ob statt bei Epoche 100 besser zu einem anderen Zeitpunkt der Trainings-Datensatz gewechselt werden sollte.

Hierfür wird für 196 Epochen rein auf synthetischen Daten trainiert.

Wie in Abb. 5.17 gezeigt, sinkt die Fehlerrate auch nach Episode 100 nicht mehr deutlich. Das durchgeführte Experiment kann durch Veränderung dieses Parameters also nicht mehr verbessert werden.

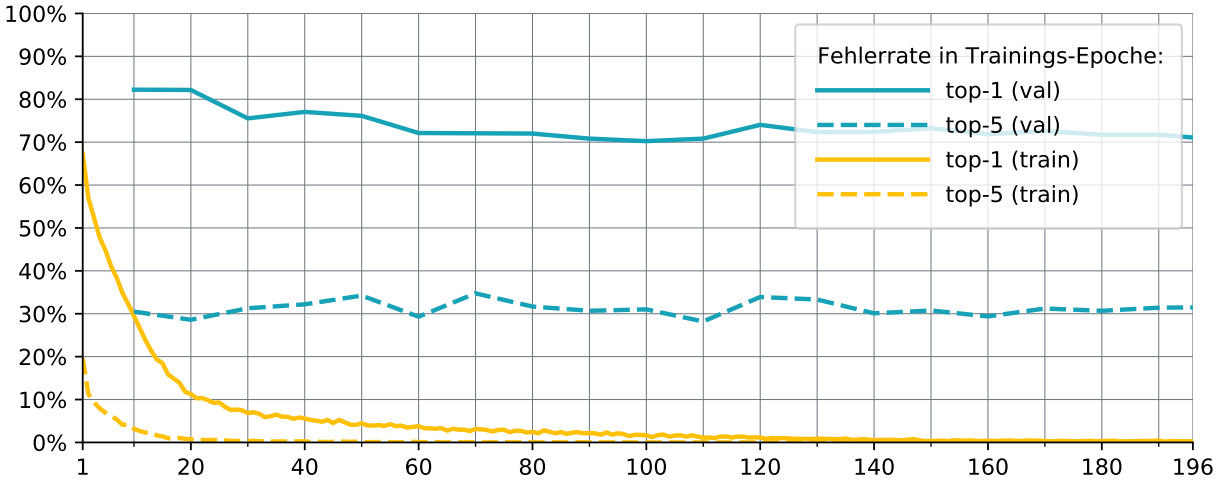


Abb. 5.17.: Fehlerraten beim rein synthetischen Training mit *S-SPHAR*-3.

5.2.2.4. Auswertung des Test mit *SPHAR*

Auch bei den Experimenten, die *SPHAR* als Grundlage für reale Videos verwendet haben, konnte keine deutliche Verbesserung der Genauigkeiten durch Hinzunahme von synthetischen Daten erreicht werden.

Obwohl der *SPHAR* Datensatz näher am *S-SPHAR* Datensatz ist, als der *HMDB* Datensatz, so kann weiterhin vermutet werden, dass sich die beiden Datensätze nicht ausreichend ähnlich sind.

Da bei der Erstellung von *SPHAR* bereits darauf geachtet wurde, den Anwendungsfall dieser Arbeit möglichst genau nachzubilden, ist es im Rahmen dieser Arbeit nicht möglich, noch bessere Videos für den Anwendungsfall zu finden.

Um die allgemein formulierte These zu beweisen oder zu widerlegen, sollte also ein anderer Anwendungsfall gewählt werden, für den ausreichend reale Daten zur Evaluierung zur Verfügung stehen. Dann könnte die Simulation entsprechend diesem Anwendungsfall entwickelt werden, und die drei Domänen „synthetische Daten“, „Anwendungsfall“ und „Reale Trainingsdaten“ würden übereinstimmen.

5.3. Ergebnisse der Experimente

Die Ergebnisse der vorangehenden Tests mit verschiedenen Methoden und unterschiedlichen Kombinationen von Datensätzen werden in Tabelle 5.4 noch einmal zusammengefasst gezeigt.

Methode	Top1-Genauigkeit Train	Val	Test
HMDB Baseline	99.42%	61.81%	58.81%
HMDB und *S-SPHAR*-1	99.32%	61.66%	59.81%
SPHAR Baseline	87.69%	66.89%	67.29%
S-SPHAR-3 Training mit *SPHAR* Val & Test Split	≈100%	≈30%	nicht getestet
SPHAR und *S-SPHAR*-2 *on top*	91.51%	**67.93**%	67.10%
SPHAR und *S-SPHAR*-2 *full*	92.99%	66.89%	**67.55**%
S-SPHAR-3 und dann *SPHAR*	≈81%	≈62%	nicht getestet

Tabelle 5.4.: Genauigkeiten der Handlungserkennung beim Training mit *SPHAR* und verschiedenen Methoden.

Die höchsten Genauigkeitssteigerung beim Test konnten bei den beiden Nachtrainings-Versuchen mit *SPHAR* und *S-SPHAR*-2 erreicht werden. Da die Genauigkeitssteigerung aber nur wenige Promille beträgt, werden diese als nicht aussagekräftig betrachtet.

Abhängig von der Auswahl der Videos in den Splits entstehen Schwankungen und Messungenauigkeiten, die für eine generalisierte Aussage berücksichtigt werden sollten.

Mit den Experimenten dieser Arbeit konnte keine ausreichende Genauigkeitssteigerung gemessen werden, um die These zu beantworten.

Um herauszufinden, warum die getesteten Datensätze nicht für diesem Nachweis geeignet sind, werden nun einige Analysen angestellt.

Zunächst einmal wird die Verteilung der Klassen in den jeweiligen Datensätzen geprüft. Hiermit kann sichergestellt werden, dass ausreichend Videos für jede Klasse vorhanden sind.

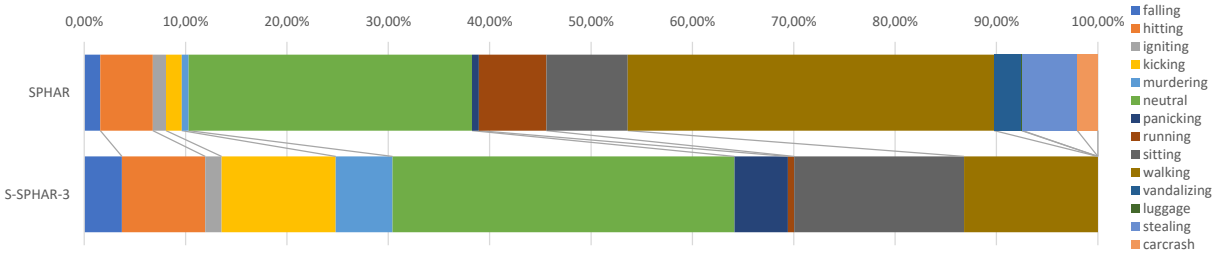


Abb. 5.18.: Verteilung der Klassen von *SPHAR* und *S-SPHAR*-3, relativ zur Gesamtgröße der jeweiligen Datensätze.

Wie in Abb. 5.18 gezeigt, sind vier Klassen des *SPHAR* Datensatzes nicht in *S-SPHAR*-3 vertreten. Für diese Klassen ist durch die Hinzunahme der synthetischen Daten also nur eine sehr geringe Genauigkeitssteigerung aufgrund des Ausschlussprinzips zu erwarten.

Die anderen Klassen scheinen recht gut vertreten, lediglich die Klasse *running* scheint etwas unterrepräsentiert. So kann die Anzahl der Videos je Klasse als Grund für das wenig verändernde Nachtrainieren ausgeschlossen werden.

Über die Qualität der Videos sagt die vorangehende Analyse nichts aus, weshalb hierfür eine Sichtprüfung der verschiedenen Videos vorgenommen wurde. Hierbei wurde festgestellt, dass der *SPHAR* Datensatz deutlich mehr Varietät zwischen den einzelnen Videos enthält. Vor allem zwischen Videos verschiedener Original-Datensätze gibt es große Unterschiede (siehe Abb. 5.19).

Abb. 5.19.: Visuelle Unterscheidung von *SPHAR* (unten) und *S-SPHAR*-3 (oben) [125; 126].

Obwohl der *SPHAR*-Datensatz mit dem Ziel erstellt wurde, den Anwendungsfall zu repräsentieren, so konnte jedoch trotzdem nur einer begrenzte Anzahl bestehender Videos gefunden werden. Im visuellen Vergleich fällt auf, dass die synthetischen Videos des *S-SPHAR* Datensatzes den Anwendungsfall besser repräsentieren. Ein Nachteil des *S-SPHAR* Datensatzes ist jedoch, dass die gestellten Szenen und wiederverwendeten Animationen schnell eintönig wirken.

Die Experimente mit dem *HMDB* Datensatz sind vermutlich deswegen gescheitert, weil dessen Videos noch weniger Gemeinsamkeiten mit dem *S-SPHAR* Datensatz haben, weshalb *S-SPHAR* keine wertvollen Informationen über Handlungen des *HMDB* Datensatzes beinhaltet. Eine Sichtprüfung der Videos des *HMDB* Datensatzes hat zudem ergeben, dass sich in den verschiedenen Splits verschiedene ähnliche Ausschnitte der gleichen Spielfilmszenen befinden, weshalb die Genauigkeiten weniger gut für generelle Aussagen geeignet sind.

Abschließend kann keine konkrete mögliche Ursache als definitiv ausschlaggebend genannt werden. In Abschnitt 7.3 werden verschiedene Möglichkeiten zur weiteren Forschung an diesem Thema vorgeschlagen.

6. Anleitungen zur Installation und Bedienung

Die zwei in dieser Arbeit entwickelten Komponenten Simulation und Handlungserkennung können voneinander unabhängig installiert und genutzt werden.

Im Folgenden werden jeweils die Installation der Komponenten sowie Anleitungen für verschiedene Nutzungsszenarien beschrieben.

Im Unterschied zu den vorangehenden Kapiteln wird hier eine speziell auf reine Anwender fokussierte Anleitung gegeben, und es wird nicht auf Details der Implementierung, Begründungen zu Entscheidungen und Möglichkeiten zur Modifikation eingegangen.

Dieses Kapitel ist daher besonders hilfreich für zukünftige Anwender im Unternehmen.

6.1. Anleitungen für die Simulation

In diesem Abschnitt wird die Installation und Benutzung der Simulation beschrieben. Die Simulation kann in zwei Varianten genutzt werden. Wird sie mitsamt der Entwicklungsumgebung (Unity) installiert, können Szenen verändert oder hinzugefügt und Videos aufgenommen werden. Wird die Standalone-Version installiert, so ist nur die Aufnahme von Videos (in den bisher entwickelten Szenen) möglich.

6.1.1. Installation der Simulation

Zur Installation der vollständigen Simulation und Entwicklungsumgebung muss zunächst das Installationsprogramm für Unity von folgender Adresse heruntergeladen werden:

`https://store.unity.com/de/download`

Der von dort heruntergeladene „Unity Hub“ bietet nun die Möglichkeit, eine Lizenz einzurichten und eine Version des Unity-Editors herunterzuladen. In dieser Arbeit wurde die Version `2020.1.1f1` verwendet.

Nach der Installation sollte der komprimierte Projektordner der Simulation von der im Anhang beschriebenen DVD in ein bearbeitbares Verzeichnis extrahiert werden.

Anschließend kann über die *Add* Schaltfläche im *Projects* Reiter des Unity Hub dieser Ordner ausgewählt und importiert werden. Ein Klick auf den neu erstellten Eintrag in der Projektliste öffnet anschließend den Unity Editor, womit die Installation abgeschlossen ist.

Die relevante Unity-Szene befindet sich im Ordner `Assets\_Scenes\ActionSim`. Im folgenden Abschnitt „Bedienung“ wird die Nutzung der Unity-Szene beschrieben.

Soll stattdessen nur die Standalone-Version der Simulation installiert werden, sind weniger Schritte notwendig. Diese Version benötigt keinen Installationsvorgang. Es genügt, das Verzeichnis von der DVD zu entpacken und die für das jeweilige Betriebssystem passende enthaltene Datei zu öffnen, beispielsweise `ActionSim.exe` unter Windows.

Das Programm startet die Simulation automatisch. Wenn zwei Monitore verfügbar sind, wird die Vorschau der semantischen Segmentierung automatisch auf dem Zweitbildschirm angezeigt.

6.1.2. Bedienung der Simulation

Wie bereits erwähnt, sind die Möglichkeiten zur Bedienung der Simulation abhängig von der installierten Version.

Während das Projekt im Unity Editor auch die vollständige Bearbeitung der Szene ermöglicht, ist in der die Standalone-Version nur die Erzeugung neuer Trainingsdaten und die Anpassung der Szenen über die erstellte grafische Oberfläche möglich.

Im Folgenden werden einzelne mögliche Anpassungen näher beschrieben.

Erheben neuer Daten In beiden Versionen der Simulation ist das Erheben neuer synthetischer Daten möglich. Hierfür genügt der Start der Simulation, entweder per Klick auf die pfeilförmige „Play“ Schaltfläche im Editor oder durch Start der Standalone-Version.

Anschließend kann in der grafischen Oberfläche (siehe Abschnitt 4.5) ein Name für die Aufzeichnung eingegeben werden. Darüber hinaus können Einstellungen wie ein automatischer Szenenwechsel oder Wetterbedingungen eingerichtet werden. Zum Start der Aufnahme genügt ein Klick auf die Schaltfläche „Start Recording“.

Hinzufügen einer neuen Szene und Verändern von 3D-Objekten Das Hinzufügen einer neuen Szene, das Verändern von Objekten oder das Ändern des Aufnahmeverhaltens über die entwickelte grafische Oberfläche hinaus ist nur mit der Unity Editor Entwicklungsumgebung möglich. Wird der Unity Editor verwendet, so können neue, ähnliche wie die in Kapitel 4 beschriebenen Szenen oder Animationen hinzugefügt werden. In den meisten Fällen wurde auf die Verwendung gängiger Entwurfsmuster und Konventionen bei der Spieleentwicklung geachtet, wie beispielsweise das Trennen von Aussehen und Funktion.

Um eine neue Simulations-Szene zu erstellen, muss keine neue Unity-Szene erstellt werden. Stattdessen sollte eines der bestehenden Terrains ausgewählt werden und mithilfe der Terrain Tools entweder ein neues Terrain erstellt oder ein bestehendes Terrain ergänzt werden. Mit den zur Verfügung stehenden Werkzeugen können leicht Berge und Täler erzeugt werden. In Abb. 6.1 wird ein Ausschnitt des Editors gezeigt, mit dem gerade ein Terrain bearbeitet wird.

Im Ordner `_prefabs` befinden sich zahlreiche nach Kategorien sortierte 3D-Objekte, die per *drag & drop* in die Szene geschoben werden können. Zur genaueren Positionierung können das *Transform*-Werkzeug genutzt oder die Werte der Transform-Komponente geändert werden.

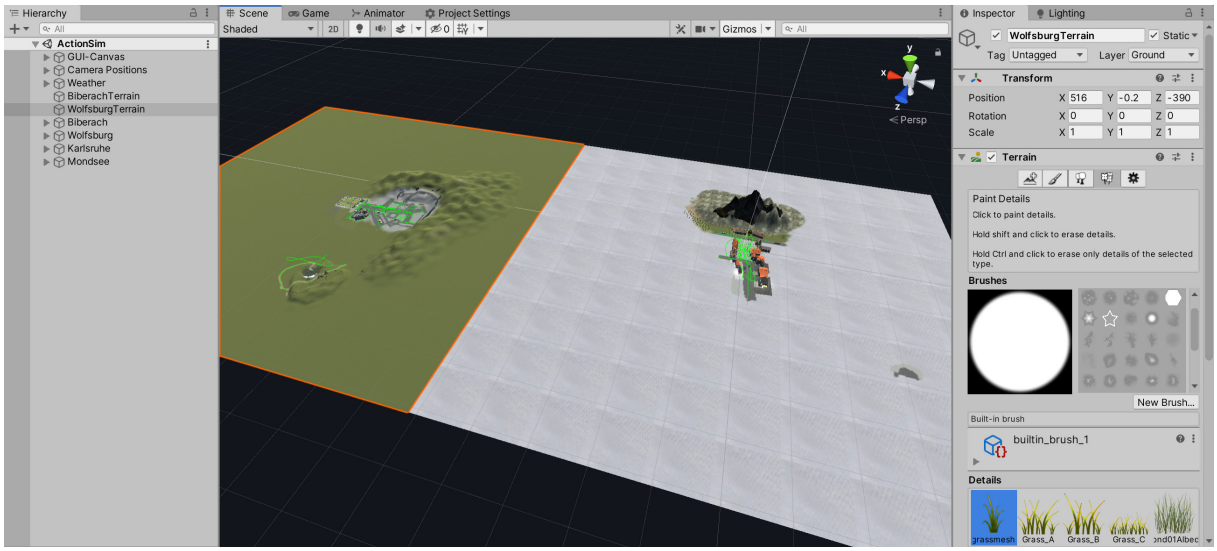


Abb. 6.1.: Bearbeitung von Simulations-Szenen in Unity [125; 126].

Zum Hinzufügen einer neuen Kamera kann ein bestehendes Kamera-Objekt in der Hierarchie (im Punkt „Camera Positions“) dupliziert, umbenannt und an die gewünschte Position verschoben werden. Anschließend kann das neue GameObject zur Liste des im Rahmen dieser Arbeit geschriebenen „Camera Position Changer“ Skripts hinzugefügt werden.

Anpassen der Segmentierung Auch die Klassen der semantischen Segmentierung und die Zuordnung dieser zu den 3D-Objekten kann angepasst werden. Hierfür kann das *Layer* Dropdown im Inspektor des 3D-Objekts sowie die dort angebotene *Add Layer* Funktion genutzt werden.

Das zentrale Skript zur Segmentierung ist eine Komponente des „Unity Main Camera“ GameObjects. Im Inspektor können dort weitere Optionen und andere Methoden zur Segmentierung ausgewählt werden.

Die Anpassung von Animationen, Charakteren und Bewegungspfaden wurde bereits in Abschnitt 4.4 detailliert beschrieben.

6.2. Anleitungen zur Handlungserkennung

Die Handlungserkennung basiert auf dem SlowFast Framework, das die Programmiersprache Python verwendet [59].

Um die Portabilität und Nachvollziehbarkeit der Installation des Frameworks zu erhöhen, wurde im Rahmen dieser Arbeit ein Dockerfile entwickelt.

In diesem Abschnitt wird die Installation und Bedienung der Handlungserkennung mit und ohne Dockerfile beschrieben.

6.2.1. Installation der Handlungserkennung

Die Installation der Handlungserkennung kann entweder über Docker oder in eine lokale virtuelle Python-Umgebung erfolgen.

Im Falle der Installation via Docker genügt es, die Projektstruktur von der im Anhang beschriebenen DVD zu kopieren und im Projektordner das Skript `build.sh` auszuführen. Voraussetzung hierfür ist eine installierte Version von Docker mit GPU-Unterstützung, beispielsweise durch Installation des NVIDIA Container Toolkits (*nvidia-docker*). Hier kann die jeweils aktuelle Anleitung des Herstellers von der folgenden Seite verwendet werden:

`https://github.com/NVIDIA/nvidia-docker`

Ohne Docker ist die Installation etwas umständlicher, da alle Komponenten manuell auf dem lokalen Computer installiert werden müssen. Neben Python 3.6+ und zahlreichen Python-Bibliotheken muss auch das CUDA-Framework installiert sein. Die Installation dieser Komponenten ist stark vom verwendeten Betriebssystem und den aktuellen Versionen der jeweiligen Software abhängig, weshalb in dieser Arbeit nicht auf alle Varianten eingegangen werden kann.

Falls beispielsweise die neueste empfohlene Version des Betriebssystems Ubuntu, 20.04, genutzt wird genügt zur Installation von CUDA der Befehl `sudo apt install nvidia-cuda-toolkit`. Anschließend kann mit `nvcc -V` die installierte Version des Frameworks ausgegeben werden.

Abhängig von der jeweiligen Version kann nun die passende Variante von PyTorch heruntergeladen werden. Hierzu kann auf der folgenden Website der passende Befehl generiert werden:

`https://pytorch.org/get-started/locally/`

Die weiteren zur Installation benötigten Pakete können mit den gängigen Paketmanagern wie Pip oder Conda heruntergeladen werden. Eine Auflistung aller für das SlowFast Framework benötigten Bibliotheken sowie Tipps zu deren Installation befindet sich im Repository des Frameworks in der Datei `INSTALL.md` [59].

6.2.2. Bedienung der Handlungserkennung

Das Projekt zur Handlungserkennung kann sowohl zum Trainieren als auch zum Validieren von ML-Modellen genutzt werden.

Zur Auswahl und Einrichtung des Betriebsmodus werden im SlowFast Framework Konfigurationsdateien (*model configs*) verwendet. In diesem Abschnitt werden diese kurz beschrieben und der Start des Frameworks beschrieben.

Training und Test via Konfigurationsdateien Im Anhang dieser Arbeit wird beispielhaft eine Konfigurationsdatei gezeigt, die zum Trainieren eines Modells basierend auf dem *SPHAR* [126] Datensatz benutzt werden kann.

Zum Aktivieren der verschiedenen Modi sind die Einträge `TRAIN.ENABLE` und `TEST.ENABLE` ausschlaggebend. Abhängig von Installation müssen zudem gegebenenfalls die Dateipfade angepasst werden. Statt wie im Beispiel SPHAR kann auch HMDB als Datensatz verwendet werden. Hierfür ist darauf zu achten, neben den Namen und Pfaden auch die Anzahl der Klassen in der Konfigurationsdatei auszutauschen (`MODEL.NUM_CLASSES`). Die Anzahl der zu trainierenden Epochen kann mithilfe der Einstellung `SOLVER.MAX_EPOCH` festgelegt werden.

Zum Start des Frameworks via Docker genügt das Ausführen der Datei `run.sh` im Projektordner. Je nach gewünschter Konfiguration, kann zuvor die letzte Zeile des Dockerfiles verändert werden, um eine andere Konfigurationsdatei zu referenzieren. In diesem Fall muss vor dem Ausführen der Docker Abbild neu gebaut werden mit `build.sh`.

Ohne Docker kann der folgende Befehl zur Ausführung des Frameworks anhand der übergebenen Konfigurationsdatei verwendet werden:

```
python slowfast/tools/run_net.py --cfg models/I3D_Sphar_Train.yaml
```

Im geöffneten Konsolenfenster wird nun die Konfiguration des Frameworks ausgegeben und der Trainings- beziehungsweise Testfortschritt beschrieben.

Trainieren mit verschiedenen Datensätzen Um mit verschiedenen Datensätzen zu experimentieren und beispielsweise mit synthetischen Daten nachzutrainieren, werden wie in Abschnitt 5.2.1.2 beschrieben die Ordner und *CSV*-Dateien der jeweiligen Datensätze ergänzt. Hierfür ist kein erneutes Bauen des Docker Abbilds erforderlich.

Die Ergebnisse von Test und Training werden in Form von Zwischenständen der Modelle (*checkpoints*) sowie Log-Dateien gespeichert.

Sollen die Trainingsergebnisse später ausgewertet werden, so sollten nach dem Training alle Dateien des Ordners `output` und die Datei `stdout.log` in einen außerhalb des Projekts liegenden Ordner verschoben werden. Der letzte Modell-Checkpoint sollte zudem in den `models`-Ordner kopiert werden, wenn das Modell später in einer Konfigurationsdatei genutzt werden soll.

Auswertung der Trainingsverläufe Zur Evaluation des Trainings können die während des Vorgangs berechneten und im Log gespeicherten Genauigkeitswerte visualisiert werden. Hier kann das Skript `plot_errorrates.py` aus dem Anhang verwendet werden.

Dieses Skript wurde auch zur Generierung der in dieser Arbeit dargestellten Grafiken verwendet, beispielsweise für Abb. 5.12.

Während des Zeitraums dieser Arbeit wurde parallel auch das SlowFast Framework weiterentwickelt. In neueren Versionen ist das Visualisierungswerkzeug Tensorboard integriert, das zusätzliche Visualisierungen ermöglicht, beispielsweise der Gewichtsaktivierungen oder relevanten Merkmalen in Videos. Darüber hinaus können Graphen und auch Konfusionsmatrizen generiert werden, in dem in der Konfigurationsdatei verschiedene Einträge im Abschnitt TENSORBOARD gesetzt werden.

Die verschiedenen Möglichkeiten zur Visualisierung von SlowFast-Projekten mit Tensorboard werden auf der folgenden Webseite des Frameworks beschrieben:

`https://github.com/facebookresearch/SlowFast/blob/master/VISUALIZATION_TOOLS.md`

Da Tensorboard erst gegen Ende des Zeitraums dieser Arbeit in das Framework integriert wurde, konnte es im Rahmen dieser Arbeit nicht eingesetzt werden. Stattdessen wurde zur Auswertung der Trainingsverläufe unter anderem das oben genannte eigene Skript entwickelt.

7. Abschließende Überlegungen

In diesem Kapitel werden die Möglichkeiten zur kommerziellen Verwendung der Handlungserkennung und Simulation beschrieben sowie das Thema Datenschutz und Ethik in Zusammenhang mit der entwickelten Lösung diskutiert.

Anschließend werden Möglichkeiten zur Erweiterung der beiden Komponenten beschrieben und es wird als Ausblick eine Handlungsempfehlung formuliert, wie das kooperierende Unternehmen sowie andere interessierte Leser und Wissenschaftler die Forschung einsetzen und fortsetzen können.

7.1. Kommerzielle Verwendung

Grundsätzlich können die Komponenten dieser Arbeit auch für kommerzielle Zwecke genutzt werden, in einzelnen Fällen sind jedoch besondere Bedingungen zu beachten. Für diesen Abschnitt gilt es zu beachten, dass es sich hierbei nicht um eine verbindliche Rechtsberatung handelt, sondern lediglich nach bestem Wissen Lizenzbedingungen gelesen und recherchiert wurde.

Das zur Handlungserkennung eingesetzte SlowFast Framework [59] steht unter der Apache Lizenz, weshalb es unter Angabe der Original-Autoren auch kommerziell genutzt werden kann.

Die eigens entwickelte Simulation basiert auf Unity und kostenlosen Assets aus dem Unity Asset Store. Unity basiert auf einem mit dem Unternehmensumsatz steigenden Preismodell, weshalb ein Konzern wie die EnBW AG als Unternehmenskunde individuelle Lizenzierungspläne mit Unity verhandeln sollte. Die Assets selber können kostenfrei lizensiert werden.

Der im Rahmen dieser Arbeit erstellte synthetische *S-SPHAR* Datensatz ist ebenfalls kommerziell benutzbar, denn er wurde unter der *GNU GPL v3* Lizenz veröffentlicht [125].

Der *SPHAR* Datensatz [126] kann nicht vollständig kommerziell verwendet werden, da nicht alle der zugrundeliegenden Videos unter einer kommerziellen Lizenz veröffentlicht wurden. Anhand der Tabelle 5.3 auf Seite 85 können die jeweiligen Lizenzen geprüft und die Videos anhand der Datei- und Datensatz-Namen gefiltert werden.

Damit der *SPHAR* Datensatz zu Forschungszwecken verwendet werden darf, wurde im Rahmen der Veröffentlichung des *SPHAR* Datensatzes während dieser Arbeit Kontakt mit den jeweiligen Autoren aufgenommen, und deren Einwilligung zur Veröffentlichung auf GitHub eingeholt [126].

7.2. Datenschutz und Ethik

Bei der Entwicklung von KI-Software sollten stets auch ethische Aspekte berücksichtigt und Fragestellungen diesbezüglich reflektiert werden.

Eine von der EU-Kommission berufene unabhängige Expertengruppe für KI hat 2019 Richtlinien für vertrauenswürdige KI veröffentlicht [73], nach denen bei der Entwicklung und Nutzung von KI-Systemen die folgenden Schlüsselanforderungen erfüllt werden sollten:

1. Menschliches Mitwirken und Beaufsichtigen
2. Technische Robustheit und Sicherheit
3. Datenschutz und Datenverwaltung
4. Transparenz
5. Vielfalt, Nichtdiskriminierung und Fairness
6. Ökologisches und gesellschaftliches Wohlergehen
7. Verantwortung und Rechenschaftspflicht

Die im Rahmen dieser Arbeit entwickelte Lösung unterstützt wie einleitend beschrieben das Sicherheitspersonal im Leitstand bei der Planung von Einsätzen, weshalb menschliches Mitwirken sichergestellt wird. Zur Beaufsichtigung können regelmäßig Prüfungen durchgeführt werden, die die Qualität der KI-Modelle überprüfen.

Die technische Robustheit und Sicherheit wird durch die Verwendung etablierter und durch mehrere Parteien geprüfte Frameworks und aktueller Sicherheitsstandards sichergestellt. Bei der Übertragung von Daten zwischen einzelnen Komponenten können Verschlüsselungstechnologien eingesetzt werden. Darüber hinaus sollten die in der Softwareentwicklung üblichen Integrationstests sowie regelmäßige Funktionstests durchgeführt werden.

Wie einleitend erwähnt wird in dieser Lösung ein hoher Wert auf Anonymisierung, Wahrung der Privatsphäre und den Datenschutz gelegt. Die Abstraktion der Personen auf deren Handlungen in textueller Form stellt eine starke Form der Anonymisierung dar. Es findet keine Verfolgung von Personen über mehrere Kameras hinweg statt (*tracking*). Kombiniert mit den DIN-Richtlinien zur Anonymisierung [101, S. 36] (siehe Abschnitt 2.1) können auch alle Klarbilder datenschutzkonform verarbeitet werden. Auch bei einem Ausfall der KI-Komponenten kann so kein Bezug zu Personen hergestellt werden, weshalb auch keine Bewertung von Einzelpersonen (*social scoring*) möglich ist, wie es beispielsweise in China aufgebaut wird. Im aktuellen Konzept ist zudem lediglich eine Echtzeit-Verarbeitung und keine Speicherung der Daten vorgesehen.

Hinsichtlich der Transparenz und Erklärbarkeit des Algorithmus zur Handlungserkennung sollten noch weitere Schritte unternommen werden. Mit den im Rahmen dieser Arbeit verwendeten Evaluierungs- und Visualisierungs-Methoden ist nur schwer nachvollziehbar, anhand welcher Kriterien die KI ihre Entscheidungen trifft. Zum jetzigen Zeitpunkt können immerhin die öffentlich zugänglichen Trainingsdaten als großer Vorteil hinsichtlich der Transparenz gesehen werden.

Die Simulation erzeugt unvoreingenommen mithilfe zufällig ausgewählter Modelle Videos von Handlungen. Stereotype auf Basis des Geschlechts, der Herkunft, der Hautfarbe oder des Alters von Personen werden daher nicht berücksichtigt. Ungewollte Korrelationen wie Geschlechterdiskriminierung (*gender-bias*) oder rassistische Beurteilungen können daher vermieden werden.

Die KI-gestützte Lösung verfolgt das Ziel, öffentliche Plätze sicherer zu machen und so zu einem höheren Sicherheitsgefühl und damit auch dem Wohlbefinden der Gesellschaft beizutragen. Die Erkennung gefährlicher Situationen hilft bei der rechtzeitigen Detektion und Gefahrenabwehr, bekämpft aber nicht die Ursachen für diese Probleme. Auf langfristige Sicht sollte daher auch in die Bekämpfung der Ursachen der Kriminalität investiert werden, beispielsweise zur Verbesserung des Bildungssystems oder zur Verringerung der sozialen Ungleichheit.

Die Umwelt wird durch diese Lösung nur wenig beeinflusst. Die entstehenden Stromkosten können durch den geringeren Treibstoffverbrauch durch eingesparte Streifenfahrten relativiert werden.

Die Rechenschaftspflicht sollte bei dieser Lösung bei der Leitstelle liegen, die die Situation anhand der Empfehlung der KI beurteilt und somit das letzte Wort hat. Nur bei grob fahrlässigen Fehlern in der Entwicklung, wie beispielsweise dem bewussten Einbau von Sicherheitslücken oder voreingenommen Modellen in die Software sollte der Herausgeber der KI-Lösung verantwortlich gemacht werden.

7.3. Erweiterungsmöglichkeiten

Wie jede Forschung im Rahmen einer Abschlussarbeit ist auch dieses Projekt zeitlich begrenzt, weshalb zahlreiche Experimente nicht durchgeführt oder die These eindeutig bewiesen oder wiederlegt werden konnte.

In diesem Abschnitt werden daher einige Möglichkeiten zur Erweiterung der in dieser Arbeit entwickelten Simulation und Handlungserkennung vorgestellt sowie Ansätze für auf diese Arbeit aufbauende Forschung beschrieben.

7.3.1. Erweiterung der Simulation

Die 3D-Simulation kann in vielerlei Hinsicht erweitert werden. Die Liste möglicher Ergänzungen kann unterteilt werden in zusätzliche Funktionen und mögliche Modifikationen bestehender Funktionen sowie anhand der Ziele der verschiedenen Ergänzungen.

Um die Diversität der Szene zu erhöhen, könnte eine Funktion implementiert werden, mit der einzelne Texturen statt ausschließlich ganze Modelle ausgetauscht werden können, beispielsweise um verschiedene Bodenbeläge zu verwenden. Darüber hinaus könnten zusätzliche Objekte eingeführt werden wie Fahrräder, Hunde, Mopeds, Kinderwagen oder Rollstühle. Realistischer würde die Szene zudem, wenn Personen und Objekte sich an die Wetterbedingungen anpassen würden, beispielsweise durch Tragen verschiedener Kleidung, durch öffnen von Regenschirmen, oder durch einen von der Jahreszeit abhängigen Wechsel von Laubfarben. Auch verschiedene Tageszeiten könnten Parameter der Simulation verändern, so befinden sich Nachts etwa generell weniger Personen auf den Straßen. Hinsichtlich der Beleuchtung könnten auch verschiedene Farbfilter verwendet werden, um Sonnenstände und Wetterbedingungen besser zu repräsentieren, beispielsweise für die Abendröte. Auch ganze neue Szenen könnten implementiert werden, beispielsweise eine sehr moderne Stadt mit Glasbauten oder einen Pausenhof mit Spielplatz.

Auch die Simulation der gezeigten Handlungen und die dahinter liegenden Animationen können noch verbessert werden. Einige in der Realität häufig vorkommenden Handlungen wie Personen, die auf ein Smartphone schauen oder Personen, die ein Auto oder Fahrrad parken und aus-, ab-, auf- oder einsteigen, kommen in der Simulation noch nicht vor. Auch die Bewegungspfade könnten optimiert werden, beispielsweise durch nicht-konstante Geschwindigkeiten bei der Fortbewegung oder durch realistische Berücksichtigung der Umgebung, beispielsweise das Warten auf vorbeifahrende Fahrzeuge. Hierfür können auf klassischer Spieleprogrammierung basierende Lösungen oder KI-Methoden wie der *Unity ML Agent* eingesetzt werden [180]. Für ein besonders realistisches Verhalten sollten die Charaktere auch ihre aktuelle Animation der Gehgeschwindigkeit anpassen.

Von den von Mixamo heruntergeladenen Animationen wurden aus Zeitgründen bisher nur eine Untermenge in die Szene implementiert, weshalb einige noch ohne großen Aufwand eingebunden werden können. Die Animationen selber könnten sogar anhand von Echtvideos mithilfe einer KI „im Studio“ trainiert und dann in den jeweiligen simulierten Welten eingesetzt werden [169].

Bei der Berechnung der Bounding Boxes kann es zurzeit zu Fehlern kommen, da das Tracking von Handlungsinstanzen außerhalb der Simulation in einem Python-Skript anhand der Segmentierungsmaske erfolgt. Würde die Zuordnung von Handlungsinstanzen und die Berechnung von Koordinaten einzelner Handlungen bereits innerhalb Unity stattfinden, könnte das Tracking robuster und genauer realisiert werden, es wären allerdings aber auch zusätzliche Annotationsdateien neben den Segmentierungsmasken notwendig. Grundsätzlich wäre die Simulation auch zur Generierung

von Bildern mit Thermalinformationen in der Lage. Derartige Kameras werden vor allem nachts und an schlecht beleuchteten Orten eingesetzt. Im einfachsten Fall können die Texturen der Personen und Objekte ausgetauscht werden durch Thermalverteilungen, in denen warme Stellen besonders hell dargestellt sind [60].

Zur Erhöhung der Benutzerfreundlichkeit der Simulation könnte in der *standalone*-Version, die ohne die Oberfläche des Unity Editors ausgeführt werden kann, in die grafische Oberfläche eine Vorschau der semantischen Segmentierung integriert werden. Darüber hinaus könnte die Oberfläche zum Exportieren von Videos weiter verbessert werden, beispielsweise durch Einbindung eines interaktiven Ordner-Auswahl-Fensters statt dem jetzigen Textfeld. Auch während der Aufnahme wäre eine zusätzliche Oberfläche denkbar, die als *action launcher* dient und Personen im Bild „auf Knopfdruck“ spezielle Handlungen ausführen lässt. Die jetzige Version bestimmt alle auszuführenden Handlungen zu Beginn der Aufnahme, weshalb zur Aufnahme verschiedener Handlungen am gleichen Ort ein Bewegungspfad für die Charaktere oder eine Neubestimmung aller Animationen notwendig ist.

Bei der Erstellung von Segmentierungsmasken treten zurzeit bei transparenten und halbtransparenten Objekten wie Schnee, Laub oder Feuer Fehler auf, da diese oft größer als erwartet in der Segmentierungsmaske dargestellt werden. Falls diese Objekte häufig genutzt werden sollen, sollte dieser Fehler behoben werden.

Eine Möglichkeit zur besseren Automatisierbarkeit der Simulation ist die Verwendung von Skripten. In einem festgelegten Dateiformat könnte beschrieben werden, mit welchen Parametern die Simulation starten soll und welche Handlungen ausgeführt werden sollen. Diese Konfigurationsdatei könnte dann beim Start der Simulation eingelesen werden, um reproduzierbare Ergebnisse zu erhalten. In der aktuellen Version entscheidet bei jedem Start der Zufall über die Initialwerte. Anschließend können zahlreiche Parameter über die grafische Oberfläche durch den Benutzer angepasst werden. Weitere Möglichkeiten zur Automatisierung der Simulation und zur Implementierung der Handlungserkennung in produktive Geschäftsprozesse werden in Abschnitt 7.3.3 vorgestellt.

Unabhängig von den bereits vorgestellten Erweiterungsmöglichkeiten kann auch das Konzept der *Domain Randomization* (siehe Abschnitt 2.3) wieder aufgegriffen werden. Durch deutlich mehr Zufälligkeit generierte Trainingsdaten können zu einer großen Verbesserung hinsichtlich der Generalisierbarkeit der Modelle führen [176, S. 1088], benötigen aber mehr Rechenkapaziät, die im Rahmen dieser Arbeit nicht zur Verfügung stand.

7.3.2. Erweiterung der Handlungserkennung

Auch das Vorgehen zur Erkennung von Handlungen kann in verschiedenen Bereichen weiterentwickelt werden.

Beim Training können größere Datensätze wie AVA [69], AVA-Kinetics [107] oder HACS Clips [205] verwendet werden, wenn beispielsweise durch Verwendung von Cloud-Rechnern größere Rechenkapazitäten zur Verfügung stehen.

Es kann zudem noch mit der Auswahl der Handlungen in sowohl den realen als auch den synthetischen Videos experimentiert werden, beispielsweise könnten gefährliche oder seltene Handlungen absichtlich überrepräsentiert werden. Da sich Videos einiger Klassen, wie beispielsweise *hit* und *kick* sehr ähnlich sind, ist es unter Umständen sinnvoll, die Kriterien zur Einordnung von Videos in diese Klassen genauer zu definieren.

In der aktuellen Version werden die Pixel des RGB-Video-Zuschnitts ausgewertet. Eine Alternative hierzu ist die Auswertung von Körperhaltungen [124, S. 31] in Form von Skelett-Posen im Verlauf der Zeit. Das Skelett besteht aus deutlich weniger Werten als ein RGB-Bild. Dank dieser Dimensionsreduktion können effiziente spezialisierte Netze eingesetzt werden. Dadurch, dass kein Hintergrund betrachtet wird, generalisieren diese Modelle sehr schnell. Da jedoch auch keine Objekte, mit denen interagiert wird, in das Training einfließen, können bestimmte Handlungen auch schwerer unterschieden werden [124, S. 31]. Es gibt allerdings bereits erste Methoden, die auch Objekte im Kontext einfließen lassen, was diesen Ansatz in der Zukunft sehr vielversprechend macht [197]. Da zur Erkennung von Körperhaltungen ein weiterer ML-Algorithmus erforderlich ist, kann sich je nach Anwendungsfall und eingesetzten Algorithmen mit dieser Methode der Rechenaufwand erhöhen. Die Charaktere der Simulation sind zur Unterstützung der Animationen bereits mit 3D-Skeletten ausgestattet. Diese sollten sich mit entsprechendem Code auslesen und zur Generierung eines synthetischen Datensatzes einsetzen lassen können.

Das eingesetzte SlowFast Framework wurde parallel zur Erforschung eines Ansatzes zur Auswertung von audiovisuellen Videos entwickelt [199]. Die Funktion Audio auszuwerten ist zum Zeitpunkt der Arbeit noch nicht in das Framework integriert, aber für einen späteren Zeitpunkt eingeplant [59]. Die Simulation nutzt bereits Soundeffekte bei Regen sowie Vogelgeräusche, speichert diese bisher aber nicht beim Export.

Das eingesetzte Framework bietet darüber hinaus die Möglichkeit, andere Netzarchitekturen und Hyperparameter zu verwenden, also Einstellungen vorzunehmen, die das Verhalten des Trainings signifikant verändern können [66, S. 428–436]. Es ist zu erwarten, dass andere Parameter zu deutlich anderen Ergebnissen führen. Da die benutzte Netzarchitektur jedoch mit Bedacht ausgewählt wurde, wird hier keine deutliche Verbesserung der Genauigkeiten durch das Nachtrainieren erwartet.

Mögliche Ansatzpunkte zur weiteren Optimierung sind die Verwendung spezieller dynamischer Splits während des Trainings (*Cross-Validation*), das Herstellen gleichmäßiger verteilter Datensatz-Klassen durch Hinzunahme zusätzlicher Videos sowie die Hinzunahme zusätzlicher Schichten, die einen Teil der gelernten Gewichte bewusst verwerfen um Überanpassungen zu verhindern (*Dropout Layers*).

7.3.3. Integration in Geschäftsprozesse

Als wichtige Komponente moderner Softwareentwicklung gilt die Automatisierung von Build-, Deployment- und Test-Prozessen, also das automatische Zusammenstellen, Verteilen und Testen von Software.

In diesem Abschnitt werden daher Maßnahmen zur Automatisierung der Simulation und der Handlungserkennung vorgeschlagen. Die Umsetzung dieser Maßnahmen erlaubt die Integration dieser Lösung in Geschäftsprozesse eines Unternehmens.

Eine Möglichkeit zum Betrieb und zur Weiterentwicklung der Simulation und der Handlungserkennung ist die Verwendung moderner Cloud-Plattformen wie Microsoft Azure, Amazon Web Services oder Google Cloud. Diese Plattformen bieten die Möglichkeit, Cloud-Infrastruktur nach Bedarf und sogar code-basiert zu mieten. Auch die automatisierte Ausführung von sogenannten serverlosen Funktionen ist möglich. Für diese Skripte wird bei Bedarf ad-hoc dynamisch Hardware allokiert. Serverlose Funktionen werden vor allem zur Umsetzung von Microservice-Architekturen genutzt, die den Programmcode in kleine Funktionen aufteilen, die voneinander unabhängig arbeiten.

In Abb. 7.1 wird ein mögliches Zusammenspiel der verschiedenen Komponenten dieser Arbeit gezeigt.

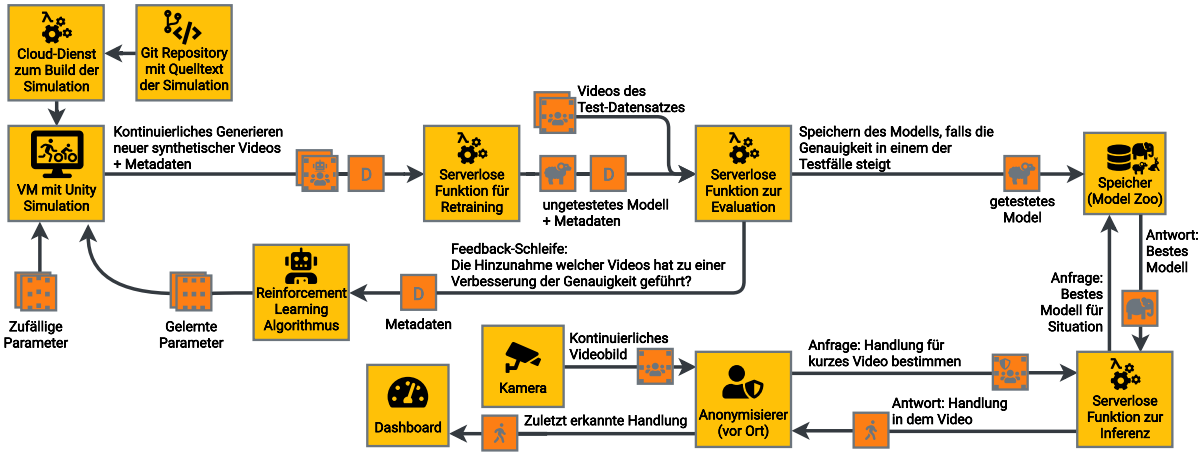


Abb. 7.1.: Architekturentwurf einer Cloud-basierten Pipeline zur kontinuierliche Verbesserung des Handlungserkenners durch Hinzunahme weiterer synthetischer Daten.

Die Code-Verwaltung erfolgt in dieser Architektur mithilfe des Versionsverwaltungssystems Git. Bei Änderungen auf dem Hauptzweig des Projekts kann automatisiert eine Funktion ausgeführt werden, die bestehende Instanzen der Simulation aktualisiert. Diese läuft in einer virtuellen Maschine, da sie im Gegensatz zu den anderen Funktionen kontinuierlich und nicht nur sporadisch verwendet wird. Die serverlosen Funktionen verwenden, da sie sich bei jedem Start neu initialisieren, automatisch die neusten zur Verfügung stehenden Versionen der Skripte. Für die Definition der Funktionen können die Dockerfiles dieser Arbeit leicht angepasst verwendet werden.

In der virtuellen Maschine, in der die Unity Simulation läuft, wird neben der reinen Aufnahme von Klarbild und Segmentierungsmasken auch das Skript zum Zuschnitt ausgeführt. Die zugeschnittenen Videos können dann in regelmäßigen Abständen an die Retraining-Funktion geschickt werden, um ein Modell zu erhalten.

Die gelernten Modelle werden, wenn diese die Genauigkeit der Handlungserkennung verbessern, in einem zentralen Speicher, dem sogenannten *Model Zoo*, abgelegt.

Vor Ort an den öffentlichen Plätzen nimmt eine Kamera kontinuierlich ein Videobild auf. Ein lokaler Computer unterteilt das Video in kleinere und kürzere auf Personen zugeschnittene Videos und anonymisiert die Videos. Diese anonymisierten Videos werden anschließend an eine serverlose Funktion zur Erkennung von Handlungen gesendet, die jeweils das beste Modell des *Model Zoo* verwendet. Die Rückantwort wird zurück an den lokalen Computer gesendet, der das weitere Vorgehen basierend auf der erkannten Handlung entscheidet. Je nach Anwendungsfall können beispielsweise als gefährlich eingestufte Handlungen eine Sicherheitsleitstelle alarmieren oder Einträge in einem Dashboard für Sicherheitsmitarbeiter hinzugefügt werden.

Darüber hinaus sollte auch ein mit allen Komponenten verbundenes zentrales Status-Dashboard mit Logging genutzt werden, das anzeigt, welche Funktionen und Server zurzeit laufen und welche in der Vergangenheit aktiv waren, und welchen Status diese melden oder ob diese erfolgreich beendet wurden. Die meisten Cloud-Anbieter bieten derartige Funktionen ohne weiteres Zutun an, weshalb diese Komponente nicht in der Skizze eingezeichnet ist.

Zur kontinuierlichen Verbesserung der synthetisch generierten Daten können die Parameter der Simulation nach jedem Test eines neuen Modells angepasst werden. Abhängig von den Testresultaten kann ein Reinforcement-Learning-Algorithmus (siehe Abschnitt 2.2.1) lernen, welche Simulations-Parameter zu einer Verbesserung des Modells führen, und welche nicht.

Um den Trainingserfolg auch für weniger technisch interessierte Stakeholder sichtbar zu machen, kann am Ende der Testprozesse ein menschenlesbares Report-Dokuments im `.pdf`-Format generiert werden. Dieses kann alle relevanten Plots und Kennzahlen enthalten, sodass jederzeit ein Überblick über den Stand des Projekts gegeben werden kann. Als erste Komponente kann hier das im Rahmen dieser Arbeit entwickelte `plot_errorrates.py` Skript zum Einsatz kommen.

7.3.4. Aufbauende Forschung

Auf diese Arbeit aufbauend kann ein weiterer Versuch unternommen werden, die These dieser Arbeit zu beweisen oder zu widerlegen. In diesem Abschnitt werden die hier verfolgbaren Ziele zusammengefasst.

Die voraussichtliche größte Chance auf einen Erfolg hat das Erstellen einer neuen Szene in der Simulation. Diese sollte nicht, wie in dieser Arbeit, das Ziel verfolgen, den Anwendungsfall aus der Einleitung möglichst genau nachzubilden. Stattdessen sollte sie bestehende Datensätze im Bereich der Handlungserkennung nachbilden. Hierfür könnte die Simulation statt überwiegend Videos aus Obersicht auch solche aus seitlicher Perspektive aufnehmen. Darüber hinaus könnten Handlungen verwendet werden, die in den Datensätzen häufiger vorkommen. Durch diese Änderungen entfernt sich allerdings die Aussagekraft hinsichtlich des in dieser Arbeit betrachteten Anwendungsfall, weshalb diese Änderungen auch nicht früher und noch im Rahmen dieser Arbeit umgesetzt wurden.

Eine weitere mögliche Frage, die auf diese Forschung aufbauend geprüft werden kann, ist inwieweit KI-Modelle zur Handlungserkennung von der Distanz von Personen zur Kamera abhängig sind. Mithilfe der synthetischen Daten können problemlos Handlungen in verschiedenen Entfernungen aufgenommen werden. Diese Einschätzung ist besonders interessant für Kundenbetreuer und Projektmanager, die für neue Installationen einschätzen sollen, wie gut die KI-Modelle vor Ort funktionieren. Auch die Frage, bis zu welcher minimalen Auflösung das KI-Modell „gut sieht“ kann mithilfe der Simulation und einer anschließenden Evaluation getestet werden.

7.4. Handlungsempfehlung

In diesem Abschnitt werden mögliche nächste Schritte für das kooperierende Unternehmen sowie für interessierte Leser und Wissenschaftler vorgeschlagen.

Das Unternehmen sollte aufgrund des Interesses an besseren Modellen und der bestehenden schlechten Datenlage anhand der in dieser Arbeit gewonnenen Erkenntnissen an dem Thema weiterforschen, zum Beispiel in Form einer Werkstudententätigkeit. Hierfür können die Vorschläge des Abschnitts 7.3 „Erweiterungsmöglichkeiten“ integriert und getestet werden, idealerweise mit einer höheren zur Verfügung stehenden Rechenkapazität. Die Anleitungen aus Kapitel 6 ermöglichen einen schnellen Einstieg in den bestehenden Code.

Da im Rahmen dieser Arbeit keine konkreteren Zahlen hinsichtlich der möglichen Verbesserung im gewünschten Anwendungsfall ermittelt werden konnten, sollten hierfür nicht zu viele Ressourcen aufgewendet werden, das Thema sollte aber „im Hinterkopf“ beibehalten werden.

Eine schnellere und mit höherer Erfolgswahrscheinlichkeit mögliche Option ist das Aufnehmen neuer Realer Daten für den Anwendungsfall. Diese Aufnahmen können auch ideal in der weiteren Forschung hinsichtlich der These dieser Arbeit verwendet werden, beispielsweise als Validierungsdaten. Ein Nachteil dieser Methode ist, dass professionelle Schauspieler und Aufnahmeorte organisiert werden müssten, was Zeit und Geld in Anspruch nimmt.

Um die Aufnahme echter Szenen an regulär besuchen öffentlichen Plätzen zu ermöglichen, ist auch eine Kooperation mit der Stadt oder der Polizei denkbar, beispielsweise in Form eines Kooperationsprojekts oder einer Sondergenehmigung. Auch ein Verein für Kampfsport könnte zur Verfügungstellung von Laiendarstellern angefragt werden.

Bei jeglicher Aufnahme von Realdaten sollte darauf geachtet werden, die Handlungsklassen sowie die gezeigten Perspektiven und Kamerawinkel exakt auf den Anwendungsfall abzustimmen und mit ausreichender Varietät in Szenen und Handlungen aufzuzeichnen.

Eine direkte Implementierung des aktuellen im Rahmen dieser Arbeit entwickelten Stands der Handlungserkennung in Produktivsysteme wird nicht empfohlen, da der Anwendungsfall aufgrund der fehlenden Realdaten nicht ausgiebig getestet werden konnte.

Auch interessierte Leser und Wissenschaftler können gerne an der Beantwortung der These weiterforschen. Hierfür können die im Abschnitt 7.3.4 „Aufbauende Forschung“ beschriebenen Ansätze weiter verfolgt werden.

8. Fazit

In diesem letzten Kapitel wird eine Zusammenfassung über die behandelten Themen gegeben. Es wird das Konzept, die Implementierungen und die Experimente kritisch betrachtet und auf Schwierigkeiten, Ergebnisse und Gelerntes eingegangen.

Aufgabe dieser Arbeit war es zu prüfen, ob die Hinzunahme synthetischer Daten zu einem auf realen Daten trainierten Modell zur Handlungserkennung dessen Genauigkeit steigern kann. Hierbei sollten dem Anwendungsfall entsprechend Videos von Handlungen auf öffentlichen Plätzen verwendet werden.

Die Erklärung der Grundlagen verlief ohne Probleme und kann als eine gute Möglichkeit zur Einarbeitung in das Thema auch für neue Mitarbeiter oder Fachfremde gesehen werden.

Um die anspruchsvolle These zu prüfen, wurde ein Konzept erarbeitet, mit welchen Programmen, Datensätzen, Anbietern und Opensource-Frameworks gearbeitet werden kann.

Hierfür wurden zahlreiche Möglichkeiten zur Erzeugung synthetischer Daten recherchiert und zum Teil getestet und entschieden, eine eigene Simulation in der Spiele-Engine Unity zu erzeugen.

Für die KI-basierte Erkennung von Handlungen wurden ebenfalls verschiedene Anbieter und Frameworks untersucht, wobei die Wahl schlussendlich auf das SlowFast Framework fiel [59].

Zum Test der Verbesserung durch synthetische Daten musste das Modell zunächst auf realen Daten trainiert werden. Hierfür wurde eine sehr umfangreiche Recherche durchgeführt, die über 60 verschiedene Datensätze auf ihren Inhalt überprüft und miteinander vergleicht. Da bei dieser Recherche kein Datensatz gefunden wurde, der alle Kriterien für den in dieser Arbeit erforderlichen Test erfüllt, wurde aus mehreren der recherchierten Datensätze ein eigener Datensatz aggregiert. Hierbei wurden nicht nur Klassen vereinigt, sondern auch anhand der Annotationen Videos zeitlich und räumlich zugeschnitten, die jeweils genau eine Handlung enthalten. Um die Forschung im Bereich der Handlungserkennung zu unterstützen, wurde dieser Datensatz veröffentlicht und frei zugänglich gemacht [126].

Die Simulation wurde wie geplant entwickelt. In vier Szenen konnten aus 13 Kameraperspektiven über 6.000 synthetische Videos mit hoher Diversität und beschrifteten Handlungen generiert werden.

Bei den Experimenten wurde sowohl der neu erstellte Datensatz als auch ein weiterer Datensatz zur Kontrolle zum Trainieren von Modellen zur Handlungserkennung verwendet. Dieses Training verlief aufgrund der zur Verfügung stehenden Rechenkapazität langsam, aber ohne weitere Probleme. Die Hinzunahme synthetischer Daten wurde mit vier verschiedenen Methoden getestet. Leider konnte bei keinem der durchgeführten Experimente eine deutliche Hinzunahme der Genauigkeit gemessen werden, weshalb die These im Rahmen dieser Arbeit nicht eindeutig bestätigt oder widerlegt werden konnte.

Als mögliche Ursache für die ermittelten Testergebnisse wird eine zu hohe Differenz zwischen realen und synthetischen Daten vermutet. Es wird angenommen, dass die zur Verfügung stehenden

realen Daten den Anwendungsfall nicht nah genug abdecken, weshalb die passenden synthetischen Daten bei diesen Realdaten zur Validierung keine Verbesserung hervorrufen können. Eine Möglichkeit zur Erzeugung von anwendungsfallübergreifenden und damit von der Art der Realvideos unabhängigen synthetischen Daten wäre die Erzeugung mittels dem *domain randomization* Verfahren. Dieses konnte im Rahmen dieser Arbeit nicht eingesetzt werden, da es hohe Rechenkapaziäten erfordert.

Um zukünftige Experimente zur Untersuchung der These zu unterstützen, wurden anschließend Anleitungen zur Installation und Bedienung der verwendeten Software geschrieben sowie Gründe für das Scheitern der Überprüfung der These sowie zahlreiche Erweiterungsmöglichkeiten und Ansatzpunkte für aufbauende Forschung aufgezeigt.

Neben den bereits erwähnten umfangreichen Komponenten wie Datensätzen, Simulation und Handlungserkennung wurden auch zahlreiche Skripte geschrieben, die die Entwicklung in diesem Bereich vereinfachen und als zentrale Komponenten für weitere Arbeiten dienen klonen.

Das in Abschnitt 1.3 definierte Ziel der Arbeit wurde daher erreicht.

Auch die gewonnenen Erkenntnisse können als ein Erfolg gewertet werden, da sie eine wichtige Orientierung für das weitere Vorgehen des kooperierenden Unternehmens geben. Im vorangehenden Kapitel wurde diesbezüglich eine Handlungsempfehlung formuliert, auf ethische Fragen hinsichtlich des Anwendungsfalls eingegangen und die Möglichkeiten zur kommerziellen Verwendung der entwickelten Software beschrieben.

Literaturverzeichnis

[1] 214 Technologies Inc. *Trueface | Our Technology*. Juli 2020. URL: https://trueface.ai/our-technology (besucht am 30. 07. 2020) (ref. auf S. 50).

[2] Martın Abadi u. a. „TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems“. In: *arXiv preprint arXiv:1603.04467* (2015) (ref. auf S. 51, 52).

[3] Sami Abu-El-Haija u. a. „YouTube-8M: A Large-Scale Video Classification Benchmark“. In: *arXiv preprint arXiv:1609.08675* (2016) (ref. auf S. 46, 47).

[4] A. Adcock u. a. „Classy Vision“. Version d95c16c. In: *GitHub repository* (2019). URL: https://github.com/facebookresearch/ClassyVision (besucht am 29. 04. 2020) (ref. auf S. 52).

[5] Adobe Systems Inc. *Mixamo*. Aug. 2020. URL: http://mixamo.com (besucht am 26. 08. 2020) (ref. auf S. 42).

[6] Adobe Systems Inc. *Mixamo | Häufige Fragen*. Juni 2020. URL: https://helpx.adobe.com/de/creative-cloud/faq/mixamo-faq.html (besucht am 26. 08. 2020) (ref. auf S. 42).

[7] AI.Reverie, Inc. *Home*. Juli 2020. URL: https://aireverie.com (besucht am 04. 07. 2020) (ref. auf S. 37).

[8] AI.Reverie, Inc. *Our Products*. Juli 2020. URL: https://aireverie.com/products (besucht am 04. 07. 2020) (ref. auf S. 37).

[9] Michael Alba. *Unreal Engineering: How a Game Engine is Playing in New Industries*. März 2020. URL: https://new.engineering.com/article/19974 (besucht am 24. 08. 2020) (ref. auf S. 38).

[10] 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. 48, 49).

[11] James F Allen. „AI growing up: The changes and opportunities“. In: *AI magazine* 19.4 (1998), S. 13–23 (ref. auf S. 10).

[12] Martin Anderson. *A comprehensive guide to the state-of-art in how AI is transforming the visual effects (VFX) industry*. Mai 2019. URL: https://rossdawson.com/futurist/implications-of-ai/comprehensive-guide-ai-artificial-intelligence-visual-effects-vfx/ (besucht am 29. 05. 2020) (ref. auf S. 26, 27).

[13] A. Andrade. „Game engines: a survey“. In: *EAI Endorsed Transactions on Serious Games* 2.6 (Nov. 2015) (ref. auf S. 38, 39).

[14] Mykhaylo Andriluka u. a. „2D Human Pose Estimation: New Benchmark and State of the Art Analysis“. In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 2014, S. 3686–3693 (ref. auf S. 46).

[15] Rowel Atienza. *Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more*. Packt

Publishing Ltd, 2018 (ref. auf S. 11, 19, 24).

[16] Bad Wildungen Live. „Kamera am Kurschattenbrunnen“. In: Agentur Floren, 2014. URL: https://kurschattenbrunnen.de (besucht am 08. 06. 2020) (ref. auf S. 48, 49).

[17] Baidu. *Apollo Synthetic Dataset*. Aug. 2020. URL: https://apollo.auto/synthetic.html (besucht am 22. 08. 2020) (ref. auf S. 29).

[18] 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. 48, 49).

[19] Mohammadamin Barekatain u. a. „Okutama-Action: An Aerial View Video Dataset for Concurrent Human Action Detection“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, S. 28–35 (ref. auf S. 46, 81–85).

[20] Alexander Bauer u. a. „N.E.S.T. – Network Enabled Surveillance and Tracking – Towards next generation surveillance systems“. In: *Future Security 3rd Security Research Conference Karlsruhe*. 2008 (ref. auf S. 3).

[21] Steven M. Bellovin, Preetam K. Dutta und Nathan Reitinger. „Privacy and synthetic datasets“. In: *Stanford Technology Law Review* 22 (2019), S. 1 (ref. auf S. 29).

[22] Christopher Berner u. a. „Dota 2 with Large Scale Deep Reinforcement Learning“. In: *arXiv preprint arXiv:1912.06680* (2019) (ref. auf S. 28).

[23] Dharti M Bhoraniya und Tushar V Ratanpara. „A Survey on Video Genre Classification Techniques“. In: *2017 International Conference on Intelligent Computing and Control (I2C2)*. 2017, S. 1–5 (ref. auf S. 26).

[24] Christoph Birkel u. a. *Der deutsche Viktimisierungssurvey 2017: Opfererfahrungen, kriminalitätsbezogene Einstellungen sowie die Wahrnehmung von Unsicherheit und Kriminalität in Deutschland*. Bundeskriminalamt, 2019 (ref. auf S. 1).

[25] Scott Blunsden und RB Fisher. „The BEHAVE video dataset: ground truthed video for multi-person behavior classification“. In: *Annals of the BMVA* 4.1-12 (2010), S. 4 (ref. auf S. 46).

[26] Erik Bochinski, Volker Eiselein und Tomas Sikora. „Training a convolutional neural network for multi-class object detection using solely virtual world data“. In: *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2016, S. 278–285 (ref. auf S. 40).

[27] Tom B Brown u. a. „Language Models are Few-Shot Learners“. In: *arXiv preprint arXiv:2005.14165* (2020) (ref. auf S. 27).

[28] Miles Brundage u. a. „The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation“. In: *arXiv preprint arXiv:1802.07228* (2018) (ref. auf S. 26).

[29] Bundesverband Digitale Wirtschaft (BVDW) e.V. „Digitale Trends: Umfrage zum Thema Künstliche Intelligenz“. In: Sep. 2018 (ref. auf S. 10).

[30] Daniel J Butler u. a. „A Naturalistic Open Source Movie for Optical Flow Evaluation“. In: *European conference on computer vision*. Springer. 2012, S. 611–625 (ref. auf S. 29).

[31] Camect, Inc. *Camect Home: Your Cameras Made Easy, Smart, Private, and Affordable*. Mai 2020. URL: https://camect.com (besucht am 30. 07. 2020) (ref. auf S. 50).

[32] Juan Cao u. a. „MCG-WEBV: A benchmark dataset for web video analysis“. In: *Beijing: Institute of Computing Technology* 10 (2009), S. 324–334 (ref. auf S. 46).

[33] Joao Carreira und Andrew Zisserman. „Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset“. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, S. 6299–6308 (ref. auf S. 26, 86, 87).

[34] Joao Carreira u. a. „A Short Note about Kinetics-600“. In: *arXiv preprint arXiv:1808.01340* (2018) (ref. auf S. 46).

[35] Joao Carreira u. a. „A Short Note on the Kinetics-700 Human Action Dataset“. In: *arXiv preprint arXiv:1907.06987* (2019) (ref. auf S. 46, 86).

[36] Andrea Cavallaro, Olivier Steiger und Touradj Ebrahimi. „Tracking Video Objects in Cluttered Background“. In: *IEEE Transactions on Circuits and Systems for Video Technology* 15.4 (2005), S. 575–584 (ref. auf S. 26).

[37] Cawamo Ltd. *Our Services | Cawamo*. Juli 2020. URL: https://cawamo.com/serivces (besucht am 30. 07. 2020) (ref. auf S. 50).

[38] Center for Biometrics and Security Research (CBSR). „CASIA action database for recognition“. In: Institute of Automation, Chinese Academy of Sciences (CASIA), 2007. URL: http://www.cbsr.ia.ac.cn/english/Action%20Databases%20EN.asp (besucht am 05. 06. 2020) (ref. auf S. 46, 82, 84, 85).

[39] Center for Research in Computer Vision. „UCF-ARG Data Set“. In: University of Central Florida, 2008. URL: https://crcv.ucf.edu/data/UCF-ARG.php (besucht am 06. 06. 2020) (ref. auf S. 46, 82, 84, 85).

[40] Center for Research in Computer Vision. „UCF Aerial Action Data Set“. In: University of Central Florida, Jan. 2009. URL: https://crcv.ucf.edu/data/UCF_Aerial_Action.php (besucht am 06. 06. 2020) (ref. auf S. 46, 81–85).

[41] Eleftheria Christopoulou und Stelios Xinogalos. „Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices“. In: *International Journal of Serious Games* 4 (Dez. 2017), S. 21–36 (ref. auf S. 38, 39).

[42] Djork-Arné Clevert, Thomas Unterthiner und Sepp Hochreiter. „Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)“. In: *arXiv preprint arXiv:1511.07289* (2015) (ref. auf S. 20).

[43] CVEDIA PVE Ltd. *Simulation Services - CVEDIA's SynCity, A Computer Vision Simulator*. Juni 2020. URL: https://cvedia.com/syncity (besucht am 02. 07. 2020) (ref. auf S. 36).

[44] CVEDIA PVE Ltd. *THEMIS Gun Detection*. Juni 2020. URL: https://cvedia.com/algorithm/gun-detection (besucht am 30. 07. 2020) (ref. auf S. 50).

[45] Leonid Datta. „A Survey on Activation Functions and their relation with Xavier and He Normal Initialization“. In: *arXiv preprint arXiv:2004.06632* (2020) (ref. auf S. 21).

[46] César De Souza u. a. „Procedural Generation of Videos to Train Deep Action Recognition Networks“. In: *arXiv preprint arXiv:1612.00881* (2016) (ref. auf S. 46).

[47] Defendry, LLC. *Defendry | Product Overview*. Juli 2020. URL: https://defendry.com/product (besucht am 30. 07. 2020) (ref. auf S. 50).

[48] 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. 48, 49).

[49] Deutsches Forschungszentrum für Künstliche Intelligenz. *Forschungsthemen: Autonome Systeme*. Mai 2020. URL: https://dfki.de/web/forschung/forschungsthemen/autonome-systeme (besucht am 19. 05. 2020) (ref. auf S. 28).

[50] Theo van Doesburg. *Composition XIII (Woman in studio)*. URL: https://wikiart.org/en/theo-van-doesburg/composition-xiii-woman-in-studio-1918 (besucht am 18. 05. 2020) (ref. auf S. 26).

[51] 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, S. 2625–2634 (ref. auf S. 23).

[52] EnBW Energie Baden-Württemberg AG. „EnBW-Integrierter Geschäftsbericht 2019". In: Bundesanzeiger, März 2020. URL: https://enbw.com/bericht2019 (besucht am 04. 05. 2020) (ref. auf S. 3).

[53] Epic Games. *Linux Quick Start | Unreal Engine Documentation*. Juli 2020. URL: https://docs.unrealengine.com/en-US/Platforms/Linux/BeginnerLinuxDeveloper/SettingUpAnUnrealWorkflow/index.html (besucht am 12. 07. 2020) (ref. auf S. 39).

[54] Epic Games, Inc. *Unreal Engine EULA For Publishing*. Aug. 2020. URL: https://unrealengine.com/en-US/eula/publishing (besucht am 24. 08. 2020) (ref. auf S. 39).

[55] Dave Epstein, Boyuan Chen und Carl Vondrick. „Oops! Predicting Unintentional Action in Video". In: *arXiv preprint arXiv:1911.11206* (2019) (ref. auf S. 46).

[56] Sergio Escalera u. a. „Chalearn looking at people challenge 2014: Dataset and results". In: *European Conference on Computer Vision*. Springer. 2014, S. 459–473 (ref. auf S. 46).

[57] Andre Esteva u. a. „Dermatologist-level classification of skin cancer with deep neural networks". In: *Nature* 542.7639 (2017), S. 115–118 (ref. auf S. 25).

[58] Bernard Ghanem Fabian Caba Heilbron Victor Escorcia und Juan Carlos Niebles. „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. 46).

[59] Christoph Feichtenhofer u. a. „SlowFast Networks for Video Recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. GitHub, 2019, S. 6202–6211. URL: https://github.com/facebookresearch/SlowFast (besucht am 29. 04. 2020) (ref. auf S. 52, 53, 77, 86, 102, 105, 110, 115).

[60] Miguel Ferreira und Jose De Oliveira. *Unite Berlin 2018 - Building Environments for Autonomous Vehicle Training*. Juli 2018. URL: https://youtu.be/4zOLq4h9ikQ (besucht am 04. 07. 2020) (ref. auf S. 36, 109).

[61] Robert Fisher, Jose Santos-Victor und James Crowley. „CAVIAR: Context Aware Vision using Image-based Active Recognition". In: University of Edinburgh EC Funded project/IST 200 137540, Sep. 2005. URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/ (besucht am 05. 06. 2020) (ref. auf S. 46, 82, 84, 85).

[62] Madan Ravi Ganesh u. a. „ViP: Video Platform for PyTorch". In: *arXiv preprint arXiv:1910.02793* (2019) (ref. auf S. 52).

[63] Abhishek Gangwar u. a. „Pornography and Child Sexual Abuse Detection in Image and Video: A Comparative Evaluation". In: *8th International Conference on Imaging for Crime Detection and Prevention (ICDP 2017)*. 2017, S. 37–42 (ref. auf S. 26).

[64] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2. Aufl. O'Reilly Media, 2019 (ref. auf S. 6, 7, 11–13, 15, 17–20, 22, 23, 28, 80).

[65] 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. 32, 78).

[66] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. MIT press, 2016 (ref. auf S. 11, 12, 19, 22, 24, 25, 27, 28, 110).

[67] Lena Gorelick u. a. „Actions as Space-Time Shapes". In: *Transactions on Pattern Analysis and Machine Intelligence* 29.12 (2007), S. 2247–2253 (ref. auf S. 46).

[68] Raghav Goyal u. a. „The Something Something Video Database for Learning and Evaluating Visual Common Sense." In: *ICCV*. Bd. 1. 4. 2017, S. 5 (ref. auf S. 46).

[69] Chunhui Gu u. a. „AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, S. 6047–6056 (ref. auf S. 31, 32, 46, 78, 110).

[70] Kevin Gurney. *An Introduction to Neural Networks*. CRC press, 1997 (ref. auf S. 6, 12, 15, 18, 20).

[71] Ankur Handa u. a. „Understanding Real World Indoor Scenes With Synthetic Data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, S. 4077–4085 (ref. auf S. 29).

[72] Heinrich Rüttgerodt GmbH & Co KG. „Webcam Marktplatz Einbeck". In: Einbecker Morgenpost, 2018. URL: https://einbecker-morgenpost.de/webcam-einbeck.html (besucht am 08. 06. 2020) (ref. auf S. 2, 48, 49).

[73] High-Level Expert Group on Artificial Intelligence. *Ethics Guidelines for Trustworthy AI*. European Commission, 2019 (ref. auf S. 106).

[74] Eric Hofesmann, Madan Ravi Ganesh und Jason J. Corso. „M-PACT: An Open Source Platform for Repeatable Activity Classification Research". In: *arXiv preprint arXiv:1804.05879* (2018) (ref. auf S. 52).

[75] Haroon Idrees u. a. „The THUMOS challenge on action recognition for videos "in the wild"". In: *Computer Vision and Image Understanding* 155 (2017), S. 1–23 (ref. auf S. 46).

[76] IfD Allensbach. „Haben Sie ein größeres Bedürfnis nach mehr Wohlstand oder nach mehr Sicherheit?" In: *FAZ Frankfurter Allgemeine Sonntagszeitung, Nr. 37*. Statista Studie 201442, Sep. 2011, S. 37 (ref. auf S. 1).

[77] Infratest dimap. „Ausweitung der Videoüberwachung an öffentlichen Plätzen". In: *ARD-DeutschlandTREND*. Tagesschau, Jan. 2016, S. 33 (ref. auf S. 1).

[78] Intelligence Advanced Research Projects Activity (IARPA). *The Multi-view Extended Video with Activities (MEVA) dataset*. Dez. 2019. URL: http://mevadata.org/ (besucht am 09. 06. 2020) (ref. auf S. 31, 46, 82–85).

[79] Intelligent Systems Research Group (ISRG). *People: Prof. Dr.-Ing. Astrid Laubenheimer*. Mai 2020. URL: https://www.iwi.hs-karlsruhe.de/ResearchGroups/ISRG/?page_id=325 (besucht am 08. 05. 2020) (ref. auf S. 3).

[80] A Iosifidis u. a. „The MOBISERV-AIIA Eating and Drinking multi-view database for vision-based assisted living". In: *Journal of Information Hiding and Multimedia Signal Processing* 6.2 (2015), S. 254–273 (ref. auf S. 46).

[81] IPICA Software LLC. *Videoüberwachung Projektierung Teil2 Kamerazonen, Identifizierung, Erkennung und DIN EN62676*. Okt. 2019. URL: https://youtu.be/CxJ-El7u3uc (besucht am 01. 09. 2020) (ref. auf S. 8).

[82] Imen Jegham u. a. „Vision-based human action recognition: An overview and real world challenges“. In: *Forensic Science International: Digital Investigation* 32 (2020), 1742–2876 (Article 200901) (ref. auf S. 26, 33).

[83] Dirk Jensen. „Livestream Webcam am Markt“. In: Bredstedt-Cam, 2013. URL: https://bredstedt.cam/webcam-markt.php (besucht am 08. 06. 2020) (ref. auf S. 2, 48, 49).

[84] Hueihan Jhuang u. a. „Towards understanding action recognition“. In: *Proceedings of the IEEE international conference on computer vision*. 2013, S. 3192–3199 (ref. auf S. 46).

[85] Yu-Gang Jiang u. a. „Consumer Video Understanding: A Benchmark Database and An Evaluation of Human and Machine Performance“. In: *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR), oral session*. 2011 (ref. auf S. 46).

[86] Yu-Gang Jiang u. a. *THUMOS challenge: Action recognition with a large number of classes*. 2014 (ref. auf S. 46).

[87] Yu-Gang Jiang u. a. „FCVID: Fudan-Columbia Video Dataset“. In: (2015) (ref. auf S. 46).

[88] 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. 46).

[89] Tero Karras, Samuli Laine und Timo Aila. „A Style-Based Generator Architecture for Generative Adversarial Networks, 2019 IEEE“. In: *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, S. 4396–4405 (ref. auf S. 26).

[90] Tero Karras u. a. „Analyzing and Improving the Image Quality of StyleGAN“. In: *arXiv preprint arXiv:1912.04958* (2019) (ref. auf S. 26).

[91] Will Kay u. a. „The Kinetics Human Action Video Dataset“. In: *arXiv preprint arXiv:1705.06950* (2017) (ref. auf S. 46, 86).

[92] Jiwon Kim, Jung Kwon Lee und Kyoung Mu Lee. „Accurate Image Super-Resolution Using Very Deep Convolutional Networks“. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juni 2016 (ref. auf S. 26).

[93] Alexander Kirillov u. a. „Panoptic Segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019, S. 9404–9413 (ref. auf S. 7, 8).

[94] Orit Kliper-Gross, Tal Hassner und Lior Wolf. „The Action Similarity Labeling Challenge“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (2011), S. 615–621 (ref. auf S. 46).

[95] 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. 46, 81, 82, 84, 85).

[96] Hema Swetha Koppula, Rudhir Gupta und Ashutosh Saxena. „Learning Human Activities and Object Affordances from RGB-D Videos“. In: *The International Journal of Robotics Research* 32.8 (2013), S. 951–970 (ref. auf S. 46, 47).

[97] Petra Kowallik, Matthias Müller-Prove und Friedrich Strauß. „Requirements-Engineering im Spannungsfeld von Individual-und Produktsoftware“. In: *i-com* 4.3 (2005), S. 41–46 (ref. auf S. 35).

[98] Stanley Kubrick. *2001: Odyssee im Weltraum*. 1968 (ref. auf S. 10).

[99] Hildegard Kuehne u. a. „HMDB: a large video database for human motion recognition“. In: *2011 International Conference on Computer Vision*. IEEE. 2011, S. 2556–2563 (ref. auf S. 31, 46, 49, 53, 74, 77, 79, 80, 86, 88, 92).

[100] Sebastian Lague. *Bézier Path Creator*. Juni 2019. URL: https://assetstore.unity.com/packages/tools/utilities/136082 (besucht am 06. 09. 2020) (ref. auf S. 68).

[101] Landesbeauftragter für den Datenschutz und die Informationsfreiheit Baden-Württemberg (LfDI BW). *35. Datenschutz-Tätigkeitsbericht*. 2019 (ref. auf S. 8, 106).

[102] Landesregierung Baden-Württemberg. *Landesdatenschutzgesetz (LDSG) § 18 - Videoüberwachung öffentlich zugänglicher Räume*. Juni 2018 (ref. auf S. 8).

[103] Landesregierung Baden-Württemberg. *Verordnung der Landesregierung über infektionsschützende Maßnahmen gegen die Ausbreitung des Virus SARS-CoV-2 (Corona-Verordnung – CoronaVO) vom 23. Juni 2020*. Juni 2020 (ref. auf S. 48).

[104] Ivan Laptev u. a. „Learning realistic human actions from movies“. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, S. 1–8 (ref. auf S. 46).

[105] Yann LeCun u. a. „Gradient-Based Learning Applied to Document Recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324 (ref. auf S. 25).

[106] 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. 46, 82, 84, 85).

[107] Ang Li u. a. „The AVA-Kinetics Localized Human Actions Video Dataset“. In: *arXiv preprint arXiv:2005.00214* (2020) (ref. auf S. 46, 110).

[108] Yin Li, Miao Liu und James M Rehg. „In the Eye of Beholder: Joint Learning of Gaze and Actions in First Person Video“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, S. 619–635 (ref. auf S. 46, 47).

[109] Guilin Liu u. a. „Image Inpainting for Irregular Holes Using Partial Convolutions“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, S. 85–100 (ref. auf S. 26).

[110] Jun Liu u. a. „NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding“. In: *IEEE transactions on pattern analysis and machine intelligence* (2019) (ref. auf S. 46).

[111] LookCam.com. *Zakopane – ul. Krupówki*. 2017. URL: https://lookcam.com/zakopane-ul-krupowki (besucht am 08. 06. 2020) (ref. auf S. 48, 49).

[112] LookCam.com. *Warsaw – Castle Square*. 2018. URL: https://lookcam.com/warsaw-castle-square (besucht am 08. 06. 2020) (ref. auf S. 48, 49).

[113] Ilya Loshchilov und Frank Hutter. „SGDR: Stochastic Gradient Descent with Warm Restarts“. In: *arXiv preprint arXiv:1608.03983* (2016) (ref. auf S. 87).

[114] Alexander Loui u. a. „Kodak's Consumer Video Benchmark Data Set: Concept Definition and Annotation“. In: *Proceedings of the international workshop on Workshop on multimedia information retrieval*. 2007, S. 245–254 (ref. auf S. 46).

[115] 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. 48, 49).

[116] Elizabeth MacBride. *German Threat Detection Company Launches In The U.S., Aiming For $1B In Revenue In 5 Years*. Feb. 2019. URL: https://forbes.com/sites/elizabethmacbride/2019/03/01/could-this-german-threat-detection-company-become-a-1b-business (besucht am 21. 07. 2020) (ref. auf S. 50).

[117] Farzaneh Mahdisoltani u. a. „On the effectiveness of task granularity for transfer learning“. In: *arXiv preprint arXiv:1804.09235* (2018) (ref. auf S. 46).

[118] Javier Marin u. a. „Learning Appearance in Virtual Scenarios for Pedestrian Detection“. In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, S. 137–144 (ref. auf S. 29).

[119] Marcin Marszalek, Ivan Laptev und Cordelia Schmid. „Actions in Context“. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, S. 2929–2936 (ref. auf S. 46).

[120] Sharon Masige. „Australian researchers just released the world's first AI-developed vaccine and it could prevent another horror flu season“. In: *Business Insider Australia* (Juli 2019). URL: https://businessinsider.com.au/australian-researchers-just-released-the-worlds-first-ai-developed-vaccine-and-it-could-prevent-another-horror-flu-season-2019-7 (besucht am 29. 05. 2020) (ref. auf S. 28).

[121] Nikolaus Mayer u. a. „A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 4040–4048 (ref. auf S. 29, 30).

[122] Warren S McCulloch und Walter Pitts. „A Logical Calculus of the Ideas Immanent in Nervous Activity“. In: *The bulletin of mathematical biophysics* 5.4 (1943), S. 115–133 (ref. auf S. 15).

[123] Goeffrey J McLachlan. „Mahalanobis Distance“. In: *Resonance* 4.6 (1999), S. 20–26 (ref. auf S. 12).

[124] Alexander Melde. *Erkennung menschlicher Handlungen durch Auswertung der Körperhaltungen von Personen in einem Video mithilfe von Machine Learning und neuronalen Netzen*. 2018 (ref. auf S. 3, 46, 48, 110).

[125] Alexander Melde. „S-SPHAR: Synthetic Surveillance Perspective Human Action Recognition Dataset“. Version 83259da. In: *GitHub repository* (2020). URL: https://github.com/AlexanderMelde/S-SPHAR-Dataset (besucht am 03. 09. 2020) (ref. auf S. 73, 74, 78, 90, 98, 101, 105).

[126] Alexander Melde. „SPHAR: Surveillance Perspective Human Action Recognition Dataset“. Version 40c1b9e. In: *GitHub repository* (2020). URL: https://github.com/AlexanderMelde/SPHAR-Dataset (besucht am 18. 07. 2020) (ref. auf S. 53, 74, 77, 79, 81–83, 85, 86, 92, 98, 101, 103, 105, 115).

[127] Meta-Level Software AG. *Arbeiten mit Framework*. Sep. 2019. URL: https://meta-level.de/arbeiten-mit-framework (besucht am 28. 08. 2020) (ref. auf S. 51).

[128] Mathew Monfort u. a. „Moments in Time Dataset: one million videos for event understanding“. In: *IEEE transactions on pattern analysis and machine intelligence* 42.2 (2019), S. 502–508 (ref. auf S. 46).

[129] Mathew Monfort u. a. „Multi-Moments in Time: Learning and Interpreting Models for Multi-Action Video Understanding“. In: *arXiv preprint arXiv:1911.00232* (2019) (ref. auf S. 46).

[130] Monitoreal Ltd. and Smart View Systems LLC. *Monitoreal - Smart Home Monitoring*. Juli 2020. URL: https://monitoreal.com (besucht am 30. 07. 2020) (ref. auf S. 50).

[131] Kevin Murphy. *Machine Learning: A Probabilistic Perspective*. Cambridge: MIT Press, 2012 (ref. auf S. 12, 13, 20, 22, 23).

[132] Vinod Nair und Geoffrey E Hinton. „Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, S. 807–814 (ref. auf S. 21).

[133] Reiichiro Nakano. „Arbitrary style transfer in TensorFlow.js". Version 5c93b2. In: *GitHub repository* (2019). URL: https://github.com/reiinakano/arbitrary-image-stylization-tfjs (besucht am 18. 05. 2020) (ref. auf S. 26).

[134] Neuromation. *Enterprise AI Transformation*. Juli 2020. URL: https://neuromation.io (besucht am 30. 07. 2020) (ref. auf S. 51).

[135] NEXT LIMIT, S.L. *Next Limit introduces ANYVERSE™ - Next Limit*. Jan. 2018. URL: http://nextlimit.com/blog/2018/01/22/anyverse/ (besucht am 03. 07. 2020) (ref. auf S. 36).

[136] NEXT LIMIT, S.L. *All Synthetic Data is Not Made Equal*. Mai 2019. URL: https://anyverse.ai/2019/05/21/all-synthetic-data-is-not-made-equal/ (besucht am 03. 07. 2020) (ref. auf S. 36, 37).

[137] NEXT LIMIT, S.L. *ANYVERSE, the Right Synthetic Data for Advanced Perception*. Juli 2020. URL: https://anyverse.ai/ (besucht am 03. 07. 2020) (ref. auf S. 36).

[138] NEXT LIMIT, S.L. *How to Get Customizable Synthetic Data for Advanced Perception*. Mai 2020. URL: https://anyverse.ai/solutions/ (besucht am 03. 07. 2020) (ref. auf S. 36, 37).

[139] NEXT LIMIT, S.L. *Use Cases by ANYVERSE - From AV/ADAS to Smart Cameras*. Mai 2020. URL: https://anyverse.ai/use-cases/ (besucht am 03. 07. 2020) (ref. auf S. 36).

[140] 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. 46).

[141] 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. 46).

[142] Mohammad Norouzi, David J Fleet und Russ R Salakhutdinov. „Hamming Distance Metric Learning". In: *Advances in neural information processing systems*. 2012, S. 1061–1069 (ref. auf S. 12).

[143] Henry Friday Nweke u. a. „Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges". In: *Expert Systems with Applications* 105 (2018), S. 233–261 (ref. auf S. 32).

[144] 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. 46, 82–85).

[145] Taesung Park u. a. „Semantic Image Synthesis with Spatially-Adaptive Normalization". In: *Proceedings of the IEEE Conference on Computer Vision and Pat-*

tern Recognition. 2019, S. 2337–2346 (ref. auf S. 26).

[146] Adam Paszke u. a. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems*. 2019, S. 8026–8037 (ref. auf S. 51, 52).

[147] Adam Paszke u. a. *Automatic differentiation in PyTorch*. 2017 (ref. auf S. 19, 51).

[148] Beat Pfister und Tobias Kaufmann. *Sprachverarbeitung. Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer, 2017. ISBN: 9783662528372 (ref. auf S. 15, 19, 20, 27).

[149] Pluralsight. *Unity, Source 2, Unreal Engine 4, or CryENGINE - Which Game Engine Should I Choose?* März 2015. URL: https://pluralsight.com/blog/film-games/unity-udk-cryengine-game-engine-choose (besucht am 10. 07. 2020) (ref. auf S. 38, 39).

[150] Thomas Pollok u. a. „UnrealGT: Using Unreal Engine to Generate Ground Truth Datasets“. In: *Advances in Visual Computing, 14th International Symposium on Visual Computing*. Springer International Publishing, 2019, S. 670–682 (ref. auf S. 28, 39).

[151] PwC PricewaterhouseCooper GmbH. *Bevölkerungsbefragung: Künstliche Intelligenz*. Juli 2017. URL: https://www.pwc.de/de/managementberatung/pwc-umfrage-deutsche-halten-kuenstliche-intelligenz-fuer-nuetzlich-sehen-aber-auch-risiken.html (besucht am 08. 05. 2020) (ref. auf S. 10).

[152] Weichao Qiu und Alan Yuille. „UnrealCV: Connecting Computer Vision to Unreal Engine“. In: *European Conference on Computer Vision*. Springer. 2016, S. 909–916 (ref. auf S. 29, 30).

[153] German Ros u. a. „The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 3234–3243 (ref. auf S. 36).

[154] Frank Rosenblatt. „The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.“ In: *Psychological review* 65.6 (1958), S. 386 (ref. auf S. 18).

[155] David E Rumelhart, Geoffrey E Hinton und Ronald J Williams. *Learning Internal Representations by Error Propagation*. Techn. Ber. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (ref. auf S. 19, 21).

[156] Ruzzini Palace. *Multimedia-Galerie*. 2019. URL: https://ruzzinipalace.com/de/multimedia-galerie (besucht am 05. 09. 2020) (ref. auf S. 48, 49).

[157] 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. 46, 82–85).

[158] Daniel Sánchez, Miguel Ángel Bautista und Sergio Escalera. „HuPBA8k+: Dataset and ECOC-Graph-Cut based segmentation of human limbs“. In: *Neurocomputing* 150 (2014), S. 173–188 (ref. auf S. 46).

[159] 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. 46).

[160] Scylla. *Scylla Behavior Recognition*. Juli 2020. URL: https://scylla.ai/behavior-recognition (besucht am 21. 07. 2020) (ref. auf S. 50, 52).

[161] Amir Shahroudy u. a. „NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 1010–1019 (ref. auf S. 46).

[162] Rajalingappaa Shanmugamani. *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Packt Publishing Ltd, 2018 (ref. auf S. 6, 7, 20, 21, 26).

[163] David Silver u. a. „A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play“. In: *Science* 362.6419 (2018), S. 1140–1144 (ref. auf S. 28).

[164] Bharat Singh u. a. „A Multi-Stream Bi-Directional Recurrent Neural Network for Fine-Grained Action Detection“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 1961–1970 (ref. auf S. 46).

[165] Gurkirt Singh u. a. „Online Real-time Multiple Spatiotemporal Action Localisation and Prediction“. In: *ICCV*. 2017 (ref. auf S. 46).

[166] Tej Singh und Dinesh Kumar Vishwakarma. „Video benchmarks of human action datasets: a review“. In: *Artificial Intelligence Review* 52.2 (2019), S. 1107–1154 (ref. auf S. 46).

[167] Khurram Soomro, Amir Roshan Zamir und Mubarak Shah. „UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild“. In: *arXiv preprint arXiv:1212.0402* (2012) (ref. auf S. 46).

[168] Stadt Biberach. „Marktplatzcam“. In: MyBiberach, 2015. URL: http://marktplatzcam.mybiberach.de (besucht am 08. 06. 2020) (ref. auf S. 2, 48, 49, 58).

[169] Sebastian Starke u. a. „Neural State Machine for Character-Scene Interactions“. In: *ACM Trans. Graph.* 38.6 (2019), S. 209–1 (ref. auf S. 108).

[170] Waqas Sultani, Chen Chen und Mubarak Shah. „Real-world Anomaly Detection in Surveillance Videos“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, S. 6479–6488 (ref. auf S. 82, 84, 85).

[171] Jaeyong Sung u. a. „Unstructured Human Activity Detection from RGBD Images“. In: *2012 IEEE international conference on robotics and automation*. IEEE. 2012, S. 842–849 (ref. auf S. 46).

[172] Christian Szegedy u. a. „Going deeper with convolutions“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 1–9 (ref. auf S. 86).

[173] 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. 48, 49).

[174] 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. 48, 49).

[175] Josh Tobin u. a. „Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World“. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, S. 23–30 (ref. auf S. 30).

[176] Jonathan Tremblay u. a. „Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, S. 969–977 (ref. auf S. 29, 30, 109).

[177] Umbria Webcam. *Assisi Live Streaming*. 2014. URL: https://umbria.webcam/assisi (besucht am 05. 09. 2020) (ref. auf S. 48, 49).

[178] Unity Technologies ApS. „Image Synthesis for Machine Learning". Version 2b2bce9. In: *Bitbucket repository* (März 2017). URL: https://bitbucket.org/Unity-Technologies/ml-imagesynthesis (besucht am 24. 08. 2020) (ref. auf S. 38, 71).

[179] Unity Technologies ApS. *Asset Store EULA FAQ*. Sep. 2020. URL: https://assetstore.unity.com/browse/eula-faq (besucht am 06. 09. 2020) (ref. auf S. 43).

[180] Unity Technologies ApS. „Unity Machine Learning Agents Toolkit". Version 4845314. In: *GitHub repository* (2020). URL: https://github.com/Unity-Technologies/ml-agents (besucht am 11. 09. 2020) (ref. auf S. 108).

[181] Unity Technologies ApS. „Unity Perception". Version 385b68d. In: *GitHub repository* (Aug. 2020). URL: https://github.com/Unity-Technologies/com.unity.perception (besucht am 24. 08. 2020) (ref. auf S. 38, 72).

[182] Unlimited Visions BV. „Amsterdam – Dam Square 1". In: WebCam.NL, 2010. URL: https://webcam.nl/amsterdam (besucht am 08. 06. 2020) (ref. auf S. 48, 49).

[183] Unlimited Visions BV. „Elburg – Street cam". In: WebCam.NL, 2012. URL: https://webcam.nl/elburg (besucht am 08. 06. 2020) (ref. auf S. 48, 49).

[184] Ashish Vaswani u. a. „Attention Is All You Need". In: *Advances in neural information processing systems*. 2017, S. 5998–6008 (ref. auf S. 27).

[185] viisights Solutions Ltd. *video analytics | behavioral understanding systems*. Jan. 2019. URL: https://viisights.com/ (besucht am 22. 07. 2020) (ref. auf S. 50, 52).

[186] Vintra Inc. *FulcrumAI Real-Time - Powerful Video Analytics Solution*. Juli 2020. URL: https://vintra.io/fulcrumai-real-time (besucht am 30. 07. 2020) (ref. auf S. 50).

[187] VXG Inc. *Video Expert Group*. Apr. 2020. URL: https://videoexpertsgroup.com (besucht am 30. 07. 2020) (ref. auf S. 51).

[188] 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. 46).

[189] Philippe Weinzaepfel, Xavier Martin und Cordelia Schmid. „Human Action Localization with Sparse Spatial Supervision". In: *arXiv preprint arXiv:1605.05197* (2016) (ref. auf S. 46).

[190] Sanford Weisberg. *Applied Linear Regression*. 3. Aufl. John Wiley & Sons, 2005 (ref. auf S. 12).

[191] Johannes Wetzel, Astrid Laubenheimer und Michael Heizmann. „Joint Probabilistic People Detection in Overlapping Depth Images". In: *IEEE access* 8 (2020), S. 28349–28359 (ref. auf S. 3).

[192] Moritz Wichmann. „Mehrheit der Bürger spricht sich nach Anschlag von Berlin für mehr Polizei und Videoüberwachung aus". In: YouGov, Dez. 2016, S. 33. URL: https://yougov.de/news/2016/12/28/mehrheit-der-burger-spricht-sich-nach-dem-anschlag/ (besucht am 04. 05. 2020) (ref. auf S. 1).

[193] Wikipedia. *List of Unity games*. Aug. 2020. URL: https://en.wikipedia.org/wiki/List_of_Unity_games (besucht am 27. 08. 2020) (ref. auf S. 38).

[194] Wikipedia. *List of Unreal Engine games*. Aug. 2020. URL: https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games (besucht am 24. 08. 2020) (ref. auf S. 39).

[195] 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. 46).

[196] Oliver Wolff. „Pressemitteilung: Mehrheit der deutschen Jugendlichen weiß nicht, was Maschinelles Lernen ist“. In: Informationsdienst Wissenschaft (idw), Mai 2020. URL: https://idw-online.de/de/news725100 (besucht am 08. 05. 2020) (ref. auf S. 10).

[197] Xinxiao Wu u. a. „Action Recognition using Context and Appearance Distribution Features“. In: *CVPR 2011*. IEEE. 2011, S. 489–496 (ref. auf S. 110).

[198] Yuxin Wu u. a. „Detectron2“. Version bd2ea47. In: *GitHub repository* (2020). URL: https://github.com/facebookresearch/detectron2 (besucht am 18. 05. 2020) (ref. auf S. 25).

[199] Fanyi Xiao u. a. „Audiovisual SlowFast Networks for Video Recognition“. In: *arXiv preprint arXiv:2001.08740* (2020) (ref. auf S. 110).

[200] Bing Xu u. a. „Empirical evaluation of rectified activations in convolutional network“. In: *arXiv preprint arXiv:1505.00853* (2015) (ref. auf S. 21).

[201] Guangnan Ye u. a. „EventNet: A Large Scale Structured Concept Library for Complex Event Detection in Video“. In: *Proceedings of the 23rd ACM international conference on Multimedia*. 2015, S. 471–480 (ref. auf S. 46).

[202] 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, 9).

[203] ZeroEyes. *ZeroEyes AI Gun Detection*. Juli 2020. URL: https://zeroeyes.com (besucht am 30. 07. 2020) (ref. auf S. 50).

[204] Jianhua Zhang, Chen Ling und Sunan Li. „EMG Signals based Human Action Recognition via Deep Belief Networks“. In: *IFAC-PapersOnLine* 52.19 (2019), S. 271–276 (ref. auf S. 32).

[205] Hang Zhao u. a. „HACS: Human Action Clips and Segments Dataset for Recognition and Temporal Localization“. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, S. 8668–8678 (ref. auf S. 46, 110).

Danksagung

Ich möchte mich bedanken bei meinem Kollegen Sharif Elfouly, der mir bei fachlichen Fragen zur Seite stand sowie bei Dirk Doll, der mit seiner Kreativität zahlreiche Ideen für die Simulation beigesteuert hat. Darüber hinaus danke ich dem gesamten Team des ISRG für die hilfreichen Kommentare und Diskussionen, sowie meiner Familie, die mich unterstützt und motiviert hat, sowie hilfreiche Korrekturvorschläge gegeben hat. Abschließend möchte ich mich noch bei allen Autoren der Original-Datensätze von *SPHAR* bedanken für die freundliche Bereitstellung der Datensätze und der Kooperationsbereitschaft bei der Lizenzierung.

A. Anhang

Dieser Arbeit liegt eine DVD bei, die Kopien der Quelltexte aller entwickelten Komponenten sowie der veröffentlichten Datensätze enthält.

Aus rechtlichen Gründen werden die vollständigen Inhalte dieser DVD sowie die im Text referenzierten und daher im Anhang abgedruckten Quelltexte nur dem kooperierendem Unternehmen sowie dem Prüfungsausschuss zur Verfügung gestellt.

Die veröffentlichte Version dieser Arbeit enthält keine Anhänge. Stattdessen können die öffentlich zur Verfügung gestellten Inhalte unter den folgenden Adressen abgerufen werden:

`https://github.com/AlexanderMelde/SPHAR-Dataset`

`https://github.com/AlexanderMelde/S-SPHAR-Dataset`

Die DVD enthält alle Dateien der Simulation, alle Videos sowie Skripte zu den Datensätzen *SPHAR* und *S-SPHAR* sowie den Quellcode der Handlungserkennung inklusive Dockerfile und Konfigurationsdateien.

Die Dateien sind in zwei `.zip`-Dateien auf den beiden Seiten der DVD abgespeichert. Zum Extrahieren sollten die beiden komprimierten Dateien jeder Seite der DVD in einen gemeinsamen Ordner kopiert und dann zusammen ausgewählt und extrahiert werden, beispielsweise mit dem *7Zip* Dateimanager.