Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

BACHELORTHESIS

Gestenerkennung für den RoboCup mit OpenPose und neuronalen Netzen

vorgelegt von

Lennart Uhrmacher

MIN-Fakultät

Fachbereich Informatik

Studiengang: Informatik

Matrikelnummer: 6948328

Erstgutachter: Dr. Matthias Kerzel

Zweitgutachter: Marc Bestmann

Abstract

Seit geraumer Zeit spielen Gesten eine große Rolle in der Kommunikation zwischen Mensch und Roboter. Es gibt viele Ansätze zur Erkennung von Gesten, aber viele von ihnen haben räumliche Beschränkungen oder sind nicht in Echtzeit umsetzbar. Im Rahmen des RoboCup präsentieren wir in dieser Arbeit einen Ansatz zur Erkennung von Gesten mit Hilfe eines Frameworks zur Posenerkennung, OpenPose, und neuronalen Netzen. Weiterhin präsentieren wir zwei Datasets zur Gestenerkennung, welche in dieser Arbeit genutzt werden.

Gestures and body language are not only an important part of non-verbal communication between humans, but also a popular method of communication between humans and robots. There are many approaches to recognize gestures, but many of them have spatial limitations or are not suitable for real time. As part of the RoboCup, we present an approach to the detection of gestures with the help of the multi-person pose estimation framework *OpenPose* and neural networks. We also present two datasets for gesture recognition, which are used in this work.

Inhaltsverzeichnis

1 Einleitung

Die Körpersprache ist ein großer Bestandteil der nonverbalen Kommunikation zwischen Menschen, da sie sehr eindeutig Informationen vermittelt. Sie bietet zwar nicht das selbe Potential, Informationen zu übermitteln, wie die Kommunikation über das Sprechen, aber eignet sich hervorragend dazu, einfache Befehle zu übertragen. Gerade deshalb sind Gesten ein beliebtes Mittel zur Kommunikation zwischen Mensch und Roboter geworden. Auch bei den Hamburg Bit-Bots der Universität Hamburg, welche mit ihren fußballspielenden Robotern an dem internationalen Turnier RoboCup teilnehmen, ist Gestenerkennung ein Thema von großem Interesse. Sie könnte eine erste Kommunikation zwischen den Robotern und dem menschlichen Trainer ermöglichen.

Die Erkennung von Gesten wurde in den vergangenen Jahren mit den verschiedensten Methoden angegangen. Beispiele sind die Verwendung von Sensoren wie der Wii-Fernbedienung [13], per WiFi [9] oder anhand von Bildern mit *Convolutional Neural Networks* [14]. Viele Ansätze leiden jedoch unter räumlichen Einschränkungen oder sind an Sensoren gebunden, welche im RoboCup nicht erlaubt sind. Der Ansatz, den wir für diese Arbeit gewählt haben, ist unabhängig von räumlichen Gegebenheiten und funktioniert nur über die visuelle Wahrnehmung.

Es ist ein anspruchsvolles Unterfangen, Gesten nur anhand von Bildern zu erkennen, da es die präzise Wahrnehmung von Personen und deren Körpern innerhalb eines Bildes erfordert. Um mit dieser Schwierigkeit umzugehen, werden wir das Framework OpenPose nutzen, welches die Erkennung von Personen und deren Posen in Bildern ermöglicht. Wir kombinieren OpenPose mit neuronalen Netzen, um so ein System zur Erkennung verschiedener Gesten zu erhalten.

Im Rahmen dieser Arbeit sind zwei Datasets zur Gestenerkennung entstanden, welche veröffentlicht werden. Diese beiden Datasets und der Programmcode sind auf GitHub verfügbar (https://github.com/lenniuhr/gesture-recognition-for-robocup).

2 Grundlagen

In diesem Kapitel behandeln wir die Grundlagen dieser Arbeit. Hierbei beginnen wir in 2.1 mit einer Einführung in den RoboCup, einem internationalen Robotik-Wettbewerb, gefolgt von der Definition von Gesten und Körpersprache für diese Arbeit in 2.2. Abschließend geht es in 2.3 um das von uns verwendete Framework zur Posenerkennung, OpenPose.

2.1 RoboCup

Der RoboCup ist ein internationaler Robotik-Wettbewerb, bei welchem Teams aus aller Welt im Fußball gegeneinander antreten - es spielen allerdings keine Menschen, sondern Roboter, welche speziell dafür entwickelt werden, Fußball zu spielen. Durch die weltbekannte Sportart Fußball schafft es der RoboCup, die Forschung in den Bereichen Robotik und künstlicher Intelligenz für ein breites Publikum interessant und zugänglich zu machen. RoboCup wurde im Jahr 1993 von einem Team japanischer Wissenschaftler gegründet [3]. Das erste RoboCup-Turnier wurde daraufhin im Jahr 1997 mit bereits 40 verschiedenen Teams abgehalten. Seitdem findet die RoboCup-Veranstaltung jährlich statt [10]. Das offizielle Ziel der RoboCup-Initiative ist es, dass es in der Mitte dieses Jahrhunderts ein Team aus vollständig autonomen Robotern schafft, unter Einhaltung der offiziellen FIFA-Regeln gegen den Gewinner der Fußball-Weltmeisterschaft zu gewinnen [11].

Abbildung 2.1: RoboCup Standard Platform League 2019[1]

[1] http://s3.amazonaws.com/data.robocup.org/media_contents/photos/000/000/190/show/rcfri-292lr.jpg?1568809811 (Letzter Zugriff am 29.05.2020)

Um einem Roboter das Fußballspielen beizubringen, muss eine Unmenge an Herausforderungen gemeistert werden. Es muss ein breites Feld an Technologien in den verschiedensten Bereichen erforscht und verbessert werden [4]. Einerseits im Bereich der Robotik: Die Roboter müssen sich mit der Geschwindigkeit und Präzision von Profisportlern bewegen, Pässe mit dem Ball durchführen und Zweikämpfe bestreiten können. Es werden noch einige technologische Durchbrüche nötig sein, um einen Roboter zu entwickeln, welcher anatomisch mit einem menschlichen Fußballspieler mithalten kann. Ein weiterer Forschungsbereich im RoboCup ist die künstliche Intelligenz: Ein Roboter muss das Spielfeld, andere Mitspieler und den Ball präzise in Echtzeit erkennen, dazu das Spielgeschehen verstehen, interpretieren und entsprechend reagieren. Zudem muss das Team autonomer Roboter zusammen als Team spielen, so wie es bei Menschen der Fall ist. Doch nicht nur die Kommunikation zwischen Spielern ist eine Herausforderung. Da sich die Regeln des RoboCup nach den offiziellen Regeln der FIFA richten, ist es auch möglich, einen Trainer einzusetzen, welcher mit den fußballspielenden Robotern kommuniziert. Eine einfache Art der Kommunikation, welche auch im echten Fußball eine große Rolle spielt, ist die Körpersprache. In diesem Bereich der Forschung setzt diese Arbeit an.

2.1.1 Hamburg BitBots

Die Hamburg Bit-Bots sind das RoboCup-Team der RoboCup-AG der Universität Hamburg. Sie nehmen seit 2012 am RoboCup teil und spielen in der *Humanoid Kid Size League* [1]. In den *Humanoid*-Ligen müssen alle Roboter eine menschenähnliche Anatomie haben und es ist ihnen nur erlaubt, menschenähnliche Sinne zu benutzen. Abstandssensoren sind beispielsweise nicht erlaubt [12]. Diese Arbeit findet in Kollaboration mit den Hamburg BitBots statt, wo die Kommunikation zwischen Trainer und Roboter über Körpersprache genutzt werden soll.

Abbildung 2.2: Hamburg Bit-Bots mit zwei fußballspielenden Robotern[2]

[2] https://robocup.informatik.uni-hamburg.de/mitglieder/ (Letzter Zugriff am 29.05.2020)

2.1.2 Gestenerkennung für den RoboCup

Körpersprache und Gesten spielen eine große Rolle in nahezu jeder Sportart, vor allem im Fußball. Seien es die Gesten eines Schiedsrichters, welcher das Foul eines Spielers anzeigt, oder die Körpersprache eines Trainers, welcher auf einen guten oder schlechten Spielzug reagiert. Alle diese Gesten können eine direkte Reaktion des Fußballspielers zur Folge haben, an den diese Geste gerichtet ist. Beispielsweise ist das Ziehen einer gelben Karte (siehe Abbildung 2.3c) eine eindeutige Verwarnung an denjenigen Spieler, an den diese Karte gerichtet ist, und wird sein Verhalten im Rest des Spiels in den meisten Fällen beeinflussen. Auch ein verärgerter Trainer (siehe Abbildung 2.3b) kann ein eindeutiges Signal an einen Spieler sein, um ihn auf Fehler im Spiel aufmerksam zu machen und möglicherweise eine Verbesserung zu bewirken.

(a) Beide Hände hochhalten[3]

(b) Verärgerung über etwas[4]

(c) Gelbe Karte wird gezückt[5]

(d) Zur Seite zeigen[6]

Abbildung 2.3: Verschiedene Gesten von Fußballtrainer Jürgen Klopp und zwei Schiedsrichtern

[3] http://www.flickr.com/photos/foto_db/15595920590 (Letzter Zugriff am 29.05.2020)
[4] http://commons.wikimedia.org/wiki/File:J%C3%BCrgen_Klopp_Saarbr%C3%BCcken_by_Fotosaar.jpg (Letzter Zugriff am 29.05.2020)
[5] https://img.fifa.com/image/upload/t_l1/hm5epuahu6zmvdguwinq.jpg (Letzter Zugriff am 29.05.2020)
[6] https://c.pxhere.com/photos/de/fb/soccer_referee_female_football_sport_game_match_whistle-802671.jpg!d (Letzter Zugriff am 29.05.2020)

Die Roboter der Hamburg Bit-Bots verfügen aktuell nicht über Fähigkeiten zur Erkennung von Gesten oder Körpersprache von Personen. Es gibt allerdings bereits einige Anwendungsfälle, in welchen dies sehr nützlich sein könnte. Ein konkretes Beispiel für die Roboter der Hamburg Bit-Bots ist Folgendes: Fällt ein Roboter zu Boden, richtet sich dieser wieder auf. Hierbei kann es passieren, dass der Roboter von da an in die falsche Richtung, also auf das eigene Tor spielt. Offensichtlich kann dies fatale Konsequenzen für den Ausgang des Spiels haben, und es gibt aktuell keine Möglichkeit, in das Spiel einzugreifen und den Roboter neu auszurichten. Durch Kommunikation zwischen dem Trainer und dem Roboter per Körpersprache wäre dies jedoch möglich: Der Trainer könnte im Sichtbereich des Roboters eine vordefinierte Geste durchführen, welche den Roboter darauf hinweist, sich umzudrehen und auf das andere Tor zu spielen. Andere denkbare Kommandos vom Trainer wären beispielsweise, dass ein Roboter aggressiver oder defensiver spielen soll, oder dass ein Roboter zur Verteidigung in den Rückraum laufen soll.

2.2 Gesten und Körpersprache

Eine Geste wird häufig als eine bewusst eingesetzte Bewegung des menschlichen Körpers definiert, an welcher die Arme, Hände oder weitere Gliedmaßen beteiligt sein können [5]. Gesten können verschiedene Bedeutungen haben, welche von anderen Menschen wahrgenommen und interpretiert werden können. Sie machen einen großen Teil der nonverbalen Kommunikation zwischen Menschen aus. Ein klassisches Beispiel wäre das Ausstrecken eines Armes, mit dem in eine Richtung gedeutet wird. Innerhalb dieser Arbeit wird zwischen zwei Arten von Gesten unterschieden: dynamische Gesten und statische Gesten.

2.2.1 Dynamische Gesten

Dynamische Gesten sind Gesten, die eine Bewegung von Teilen des Körpers voraussetzen. Abbildung 2.4 zeigt eine Abfolge von Bildern, auf denen eine Person klatscht. Auf der Grundlage eines einzelnen Bildes ist jedoch nicht erkennbar, dass die Person klatscht, da sie auch die Hände gefaltet vor dem Körper halten könnte. Um eindeutig festzustellen, ob die Person tatsächlich klatscht, muss ein längerer Zeitraum betrachtet werden, oder technisch gesagt, eine Abfolge von Bildern. Für die dynamischen Gesten, die in den verschiedenen Datasets dieser Arbeit enthalten sind, gilt eine weitere Einschränkung: Sie sind alle eine wiederkehrende Bewegung, für die weder Anfang noch Ende festgelegt ist. Beispiele hierfür sind Klatschen, Jubeln oder Tanzen. Für diese Einschränkung gibt es zwei Gründe: Erstens sind viele Gesten, die im Kontext eines Fußballspiels stattfinden, genau solche Gesten. Zweitens vereinfacht es die Erstellung eines Datasets, wenn nicht für jede Geste der Anfang und das Ende festgelegt werden müssen.

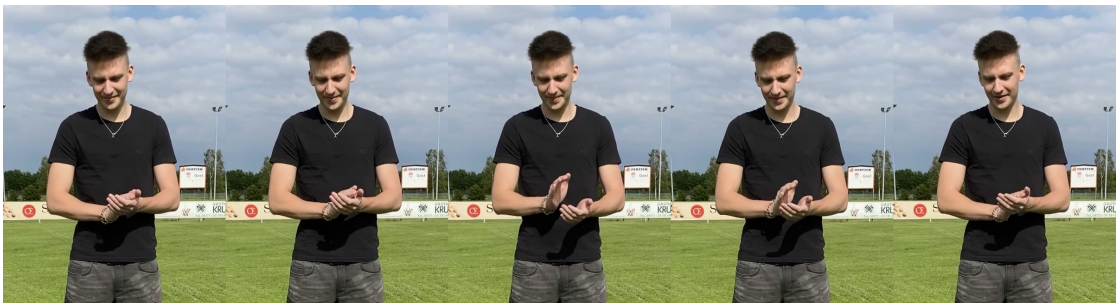


Abbildung 2.4: Eine klatschende Person

2.2.2 Statische Gesten

Statische Gesten sind Gesten, welche keine Bewegung enthalten und bei denen der Körper in einer Position ausharrt, solange die Geste durchgeführt wird. Ein Beispiel für solch eine Geste wäre das zuvor gezeigte Bild einer Schiedsrichterin (2.3d), welche mit dem Arm in eine Richtung zeigt. Natürlich wird initial eine Bewegung benötigt, um den Arm in die richtige Position zu bringen, allerdings ist die Geste daraufhin nahezu bewegungslos. Diese initiale Bewegung wird im Rahmen dieser Arbeit außen vor gelassen und nicht berücksichtigt. Abbildung 2.5 zeigt drei Beispiele für statische Gesten. Im ersten Bild steht die Person mit verschränkten Armen, im Zweiten steht die Person mit beiden Armen in den Hosentaschen und im Dritten sind beide Arme über den Kopf gestreckt. Solche Gesten sind innerhalb eines Augenblicks erkennbar, da sie keine Bewegung enthalten. Technisch gesprochen reicht also ein einzelnes Bild aus, um solche Gesten zu erkennen. Bei Schiedsrichtern im Sport sind solche Gesten sehr gebräuchlich, da für Spieler und andere Beteiligte ein einzelner Augenblick reicht, um die Geste zu erkennen. Im Falle eines Freistoßes kann ein Schiedsrichter also für einige Sekunden den Arm in die Luft halten, um anzugeben, dass ein Freistoß ausgeführt werden soll. Jeder Beobachter, der nun die ausgeführte Geste sieht, weiß sofort über die Situation Bescheid. Im Kontext des RoboCup können statische Gesten sehr nützlich sein, da ein Roboter anhand eines einzelnen Bildes eine solche Geste erkennen und direkt reagieren kann.

Abbildung 2.5: Drei statische Gesten

2.3 OpenPose Framework

Wie bereits erwähnt, ist das Thema dieser Arbeit die Erkennung von Gesten über die visuelle Wahrnehmung, also in Bildern. Dies ist eine sehr komplexe Aufgabe, da sie die Erkennung von Personen und deren Körpern in Bildern erfordert, um daraus verschiedene Gesten abzuleiten. Um mit dieser Komplexität umzugehen, benutzen wir das Framework OpenPose. OpenPose ist ein Framework zur Erkennung von Personen und deren Posen in Bilden. Die Nutzung von OpenPose ist recht einfach: Zuerst gibt man OpenPose ein Bild als Input, auf welchem sich Personen befinden. OpenPose verarbeitet dieses Bild und gibt die sogenannten *body keypoints* für jeden erkannten Körper heraus (siehe Abbildung 2.6). Es gibt bis zu 25 verschiedene *body keypoints*, dazu gehören beispielsweise beide Schultern, die Hände und die Knie. Des Weiteren ist es möglich, auch *keypoints* für die Hände und das Gesicht zu erhalten (auf Kosten der Laufzeit). In dieser Arbeit benutzen wir jedoch nur die *body keypoints*, da wir uns hauptsächlich mit Gesten befassen werden, die mit den Armen und dem Oberkörper ausgeführt werden.

OpenPose wurde in den vergangenen Jahren von einem Team aus Wissenschaftlern in Amerika entwickelt und ist als C++ Framework auf GitHub verfügbar. Außerdem gibt es ebenfalls einen Python Wrapper, welcher für diese Arbeit genutzt wurde. [2]

Abbildung 2.6: OpenPose: Input Bild und Output Bild mit *body keypoints*

2.3.1 Funktionsweise

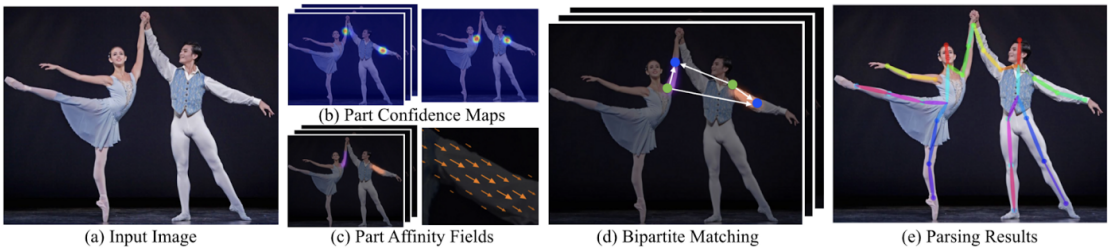


Abbildung 2.7: OpenPose Funktionsweise: Das Bild (a) wird als Input in ein *Convolutional Neural Network* gegeben. Dieses erkennt *confidence maps* (b), welche für jeden Pixel die Wahrscheinlichkeiten angeben, dass sich dort ein bestimmtes Körperteil befindet, sowie *part affinity fields* (c). Im darauf folgenden Schritt (d) werden diese beiden kombiniert, um alle möglichen Kandidaten zu vollständigen Körpern zusammenzusetzen. Das Ergebnis (e) sind die *body keypoints* für alle erkannten Personen. [2]

Ein verbreiteter Ansatz zur Erkennung von Posen in Bildern ist es, zuerst die Körper der Personen zu erkennen, und daraufhin für jeden einzelnen Körper die Pose einzuschätzen. Dies nennt man einen *top-down*-Ansatz. Es gibt allerdings zwei Probleme bei *top-down*-Ansätzen: Erstens kann die initiale Entdeckung des Körpers scheitern, wenn sich große Mengen von Personen im Bild befinden oder Teile des Körpers von anderen Elementen verdeckt werden. Wird bereits die Person nicht erkannt, ist natürlich auch die Erkennung der Pose gescheitert. Zweitens ist die Laufzeit bei *top-down*-Ansätzen proportional zu der Anzahl der Personen im Bild, für welche die Posenerkennung durchgeführt wird. OpenPose verfolgt jedoch einen sogenannten *bottom-up*-Ansatz, was bedeutet, dass zuerst alle möglichen *keypoints* im Bild erkannt und diese dann zu den entsprechenden Körpern zusammengesetzt werden. Dadurch lassen sich auch stark verdeckte Körper erkennen und die Laufzeit ist nahezu unabhängig von der Anzahl der verschiedenen Personen im Bild. [2]

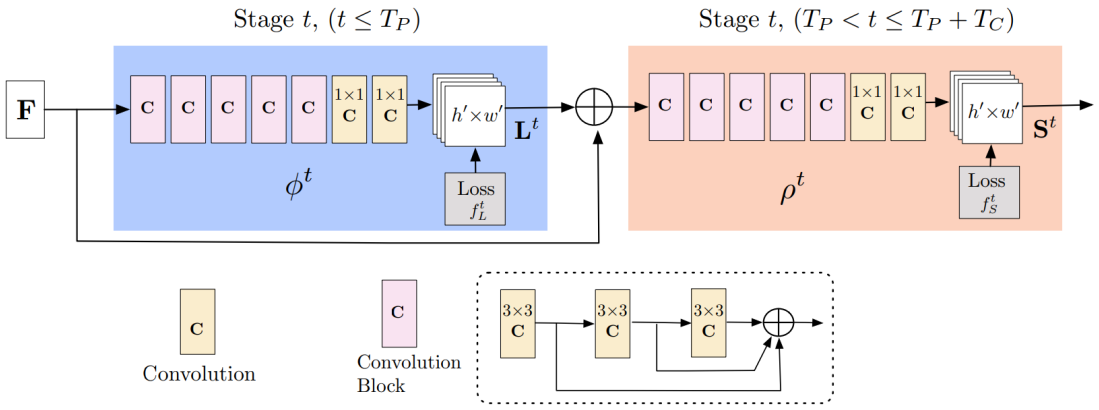


Abbildung 2.8: Struktur von OpenPose: OpenPose ist ein vielschichtiges *Convolutional Neural Network*, welches sich in zwei Komponenten aufteilen lässt. Die Erste erkennt die *confidence maps* (in blau dargestellt), die Zweite die *part affinity fields* (in orange dargestellt). [2]

OpenPose ist grundsätzlich ein vielschichtiges *Convolutional Neural Network* (siehe Abbildung 2.8) und sich in zwei Komponenten aufteilen:

Die Erste berechnet sogenannte *confidence maps* für die Positionen der *body keypoints*. Dies sind zweidimensionale *Maps*, welche für jeden Pixel die Wahrscheinlichkeit angeben, dass sich ein gegebenes Körperteil dort befindet. Dadurch lassen sich verschiedene Körperteile getrennt voneinander in dem Bild erkennen. Hierbei ist es auch kein Problem, wenn dasselbe Körperteil (beispielsweise die rechte Schulter) mehrfach im Bild vorkommt, da sich mehrere Personen im Bild befinden. Jedes vorkommende Körperteil ist als ein Ausschlag in der *confidence map* bemerkbar.

Die zweite Komponente berechnet sogenannte *part affinity fields*. Diese werden genutzt, um alle Körperteile, welche durch die *confidence maps* entdeckt wurden, zu vollständigen Körpern zusammenzusetzen. In einem einzelnen Bild kann es mehrere Kandidaten für zwei zusammengehörige Körperteile geben, die zusammen ein Gliedmaß bilden können. Durch die *part affinity fields* wird herausgefunden, welche Kandidaten am ehesten zusammengehören. Abbildung 2.7 veranschaulicht dieses Verfahren. [2]

2.3.2 Performance

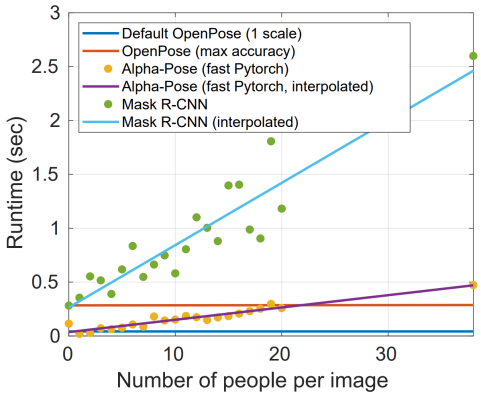


Abbildung 2.9: OpenPose Laufzeit im Vergleich [2]

Als OpenPose entwickelt wurde, hatte es im Vergleich zu anderen Frameworks zur Posenerkennung mehrerer Personen (z.B. Mask R-CNN oder Alpha-Pose) eine sehr gute Performance (siehe Abbildung 2.9). Im Vergleich zu den beiden Alternativen ist die Laufzeit von OpenPose nahezu unabhängig von der Anzahl der Personen im Bild. Dies ist eine wichtige Eigenschaft für die Anwendung im RoboCup. Bei einem Fußballspiel können sich nicht nur die anderen Spieler, sondern auch große Mengen an Zuschauern im Sichtfeld eines Spielers befinden, und dies darf die Performance eines Roboters natürlich nicht beeinflussen.

GPU	Fps (body-only)
GTX 1080 Ti (11GB)	21.7
Titan X Pascal (12GB)	17.6
GTX 1060 Ti (3GB)	8.9
GTX 1080 Ti + Titan X Pascal	38.4
CPU	**Fps (body-only)**
i9-7900X	0.4
Intel Core i7-6850K	0.4
i7-6700K	0.3
i7-4710HQ (MacOs)	0.1

Tabelle 2.1: OpenPose 1.5 performance benchmark [8]

Wie man in Tabelle 2.1 sehen kann, ist die Laufzeit von OpenPose stark abhängig von der gegebenen Hardware. Um die Posenerkennung mit OpenPose in Echtzeit durchzuführen, wird zwingend eine gute Grafikkarte benötigt. Für die Nutzung von OpenPose im Kontext des RoboCup müssen also eine gute Hardware und genügend freie Ressourcen gewährleistet sein. Da die Grafikkarte der fußballspielenden Roboter auch von anderen Prozessen genutzt wird, wäre ein möglicher erster Ansatz, OpenPose nicht in Echtzeit zu nutzen, sondern nur eine sehr niedrige Framerate zu halten.

3 Konzept

In diesem Kapitel behandeln wir das grundlegende Konzept dieser Arbeit. In 3.1 wird der Aufbau unseres Ansatzes detailliert beschrieben, gefolgt von alternativen Ansätzen in 3.2.

3.1 Pipeline zur Gestenerkennung

Die grundsätzliche Idee zur Erkennung von verschiedenen Gesten ist, OpenPose mit einem neuronalen Netz zu kombinieren. Abbildung 3.1 zeigt die Pipeline für diesen Ansatz. Im ersten Schritt wird das aufgenommene Bild in OpenPose gegeben. OpenPose erkennt daraufhin die Personen und deren *body keypoints* in dem Bild, und gibt diese in Pixel-Koordinaten heraus. Diese können nun genutzt werden, um Werte zu berechnen, die für Gesten ausschlaggebend sind. Es werden einige festgelegte Winkel und Positionen berechnet, normalisiert und in ein neuronales Netzwerk gegeben. Dieses Netzwerk ist darauf trainiert, verschiedene vorgegebene Gesten zu erkennen.

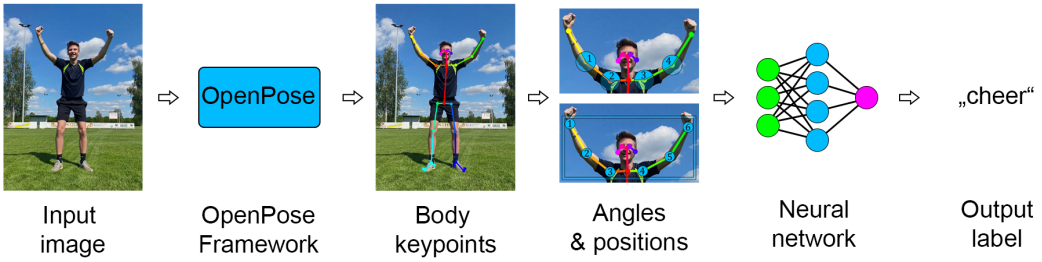
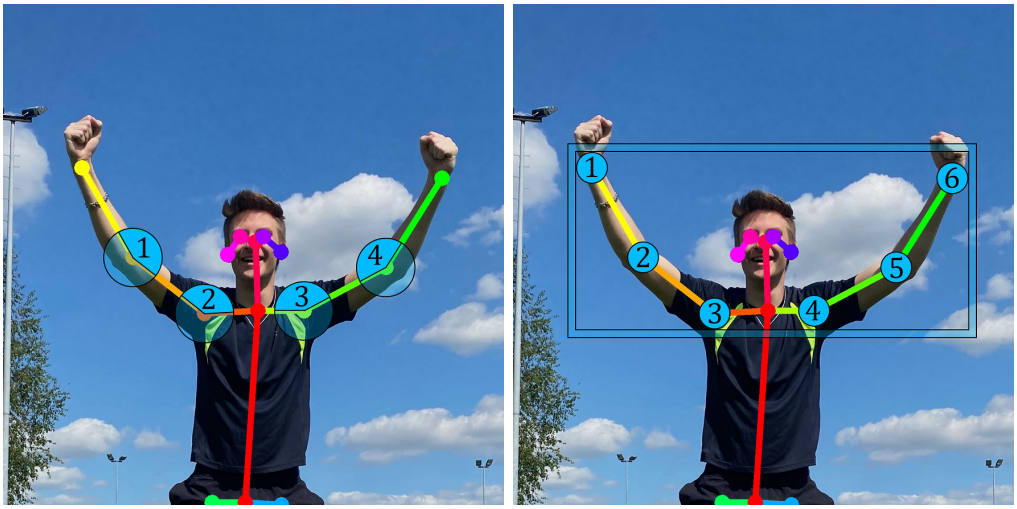


Abbildung 3.1: Pipeline zur Gestenerkennung

Da OpenPose die Körper mehrerer Personen in Bildern erkennt, muss theoretisch auch festgestellt werden, welche Personen im Bild relevant sind, sodass nur die Gesten dieser Personen erkannt werden. Im Kontext des RoboCup wäre dies beispielsweise ein Trainer oder Schiedsrichter, aber kein Zuschauer. Dieser Schritt wird in dieser Arbeit jedoch nicht berücksichtigt, stattdessen wird angenommen, dass sich nur eine Person in jedem Bild befindet. Auch die Trainingsdaten wurden absichtlich so gewählt.

Zu Beginn war eine zentrale Frage, was für Daten als Input in das neuronale Netz gegeben werden. Nachdem das Bild von OpenPose verarbeitet wurde, stehen 25 verschiedene *body keypoints* in Pixel-Koordinaten zur Verfügung. Weiterhin sind auch die *keypoints* von den Händen und dem Gesicht verfügbar, da sich diese Arbeit jedoch hauptsächlich mit Gesten beschäftigt, die mit den Armen ausgedrückt werden, werden diese *keypoints* nicht weiter beachtet. Es wäre natürlich möglich gewesen, die 25 *body keypoints* direkt in das neuronale Netz zu geben. Dadurch wären allerdings einige unnütze Informationen enthalten, die das Training des Modells nur erschweren würden. Stattdessen wurde entschieden, einige Berechnungen durchzuführen und dem neuronalen Netz ein wenig Arbeit abzunehmen. Um mehrere Gesten zu klassifizieren, ist es in erster Linie wichtig, wie die verschiedenen Körperteile angeordnet sind, oder anderes gesagt, welche Winkel an den Gelenken bestehen. Da sich diese Arbeit mit Gesten beschäftigt, die vom Oberkörper ausgehen, wurde entschieden, dass die Winkel beider Schultern und Ellenbogen ein wichtiger Input für das Modell sind (siehe Abbildung 3.2a).



(a) Winkel an Schulter und Ellenbogen (b) Positionen in der *bounding box*

Abbildung 3.2: Die Eingabe für das neuronale Netzwerk

Diese Winkel allein enthalten jedoch nicht alle Informationen, die für die Feststellung einer Geste wichtig sein können, wie beispielsweise die Position der Hände im Vergleich zum Rest des Körpers. Nachdem wir einige Zeit lediglich mit den Winkeln experimentiert hatten, haben wir ein Konzept entwickelt, um dem Modell die Positionen einiger wichtiger *body keypoints* zugänglich zu machen. Interessant sind hierbei wieder die Positionen von Schultern, Ellenbogen und den Händen. Um diese sechs Punkte wird eine *bounding box* gelegt, welche alle Punkte einschließt und so eng wie möglich gewählt ist (siehe Abbildung 3.2b). Daraufhin wird die relative Position der Punkte innerhalb dieser *bounding box* berechnet und ebenfalls als Input für das Netzwerk genutzt. Insgesamt gibt es also 4 Eingaben für die Winkel und $6 * 2 = 12$ Eingaben für die Positionen in der *bounding box*, also 16 Werte.

3.2 Alternative Ansätze

In früheren Forschungen zur visuellen Gestenerkennung wurden bereits Ansätze verfolgt, bei denen per Tracking-Methoden die Bewegungen von Kopf oder Armen aufgenommen und anhand dieser Daten mittels *Hidden Markov Models* oder *Conditional Random Fields* verschiedene Gesten klassifiziert wurden [6, 15, 16]. Dies hat Ähnlichkeiten mit unserem Ansatz, nur dass wir zur *Feature Extraction* OpenPose nutzen und statt *HMMs* und *CRFs* mit neuronalen Netzen arbeiten. Ein weiterer Ansatz aus dem Jahr 1991 war, mit einem System zur Erkennung von Handposen einzelne Buchstaben japanischer Zeichensprache zu erkennen. Daran wurde ein rekurrentes neuronales Netz geknüpft, um aus einer Sequenz an Zeichen ein gesamtes Wort zu erkennen [7]. Auch dies hat große Ähnlichkeiten mit dem von uns gewählten Ansatz.

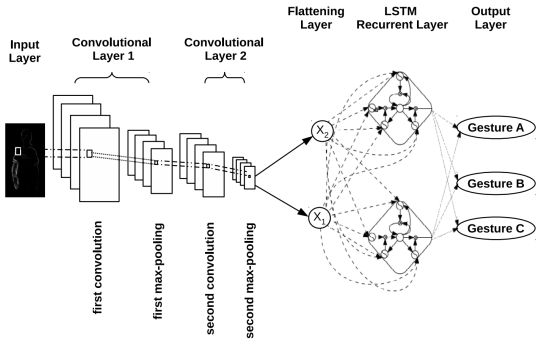


Abbildung 3.3: Pipeline eines *CNN-LSTM*-Ansatzes zur Gestenerkennung [14]

Ein alternativer Ansatz wäre gewesen, kein Framework zur Posenerkennung zu nutzen und die Bilder direkt mit einem eigenen *Convolutional Neural Network* zu verarbeiten. Ein Beispiel für diesen Ansatz ist [14], wo Gestenerkennung für ein Dataset mit neun verschiedenen dynamischen Gesten mittels einer *CNN-LSTM*-Architektur durchgeführt wurde. Ein Vorteil dieses Ansatzes ist vermutlich die Laufzeit, da OpenPose selbst ein vielschichtiges *CNN* nutzt und sehr resourcenabhängig ist. Ein Netzwerk wie das in [14] ist mit nur 2 *Convolutional Layern* wesentlich kompakter und wäre mit Sicherheit um ein Vielfaches schneller.

Es wurde sich jedoch für den Ansatz mit OpenPose entschieden, da dieser einige Vorteile bietet. OpenPose ist ein gut trainiertes Framework, welches auf Bildern in nahezu jeder Umgebung Personen erkennen kann. Außerdem spielt es auch keine Rolle, was für Kleidung die Personen im Bild tragen oder wie ihre körperliche Statur ist. Dadurch, dass wir OpenPose zu Beginn unserer Pipeline nutzen, profitieren wir aus diesen Eigenschaften und übernehmen sie für unser System. Das bedeutet, dass wir unsere Gestenerkennung unabhängig von der Umgebung und der Person durchführen können. Solange OpenPose funktioniert, tut unser System es auch. Weiterhin erhalten wir durch OpenPose die *body keypoints*, was eine vereinfachte Darstellung der Pose einer Person ist, die für uns jedoch nahezu alle relevanten Informationen enthält. Wir profitieren aus der Einfachheit dieser Daten und können bereits mit einfachen Machine-Learning-Techniken arbeiten. Mit OpenPose enthält unsere Pipeline außerdem eine Komponente, welche jederzeit durch andere oder bessere Frameworks zur Posenerkennung ausgetauscht werden kann. Entwicklungen in diesen Bereichen haben also positive Auswirkungen auf unser System, da wir direkt von ihnen profitieren können. Ein letzter Vorteil ist, dass die Gestenerkennung in unserem System durch OpenPose theoretisch für alle Personen im Bild angewendet werden kann. Dies ist in dieser Arbeit noch nicht relevant, da wir ohnehin nur mit Bildern arbeiten, auf denen sich nur eine Person befindet, aber es ist eine interessante Eigenschaft für zukünftige Entwicklungen.

4 Single-Frame-Gestenerkennung

In diesem Kapitel behandeln wir die Durchführung unseres Ansatzes für die Erkennung von Gesten auf einzelnen Bildern. In 4.1 wird die gewählte Architektur des neuronalen Netzes beschrieben. Darauf folgt in 4.2 und 4.3 die Beschreibung zweier Datasets sowie die Auswertung unseres Ansatzes für beide Datasets.

4.1 Aufbau des neuronalen Netzes

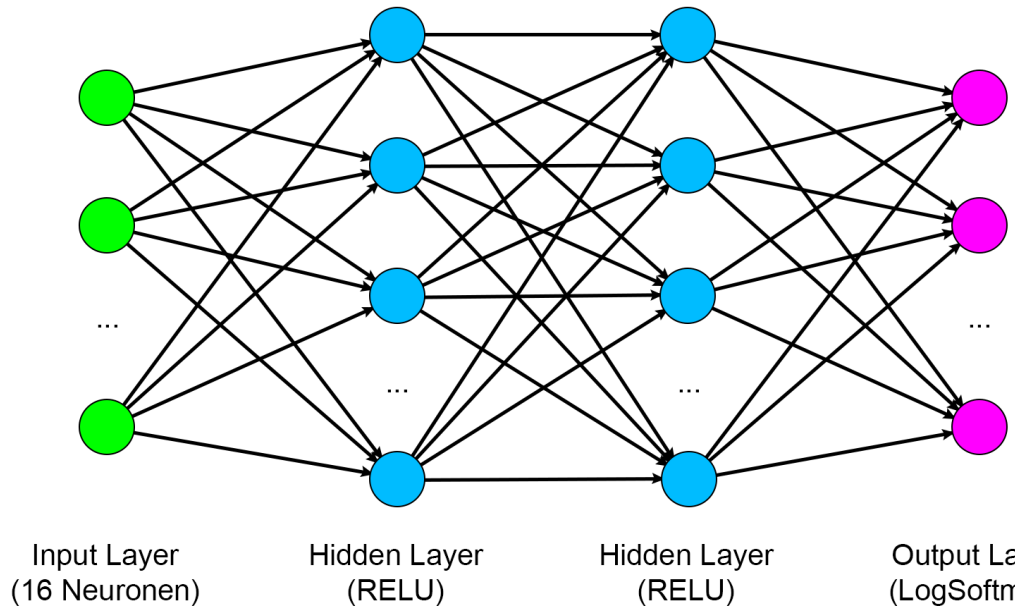


Abbildung 4.1: Neuronales Netz zur Single-Frame-Gestenerkennung

Um verschiedene Gesten anhand eines einzelnen Bildes zu klassifizieren, wurden zwei verschiedene Datasets aufgenommen, auf deren Grundlage jeweils ein neuronales Netz trainiert wurde. Figur 4.1 zeigt eine Abbildung eines der neuronalen Netzwerke, welche wir im Folgenden nutzen werden. Es besteht aus einem *Input Layer* mit 16 Neuronen. Diese Größe ergibt sich daraus, dass wir aus dem Output von OpenPose 16 verschiedene Werte errechnen (siehe Kapitel 3). Auf das *Input Layer* folgen ein bis mehrere *Hidden Layer*. In den folgenden Kapiteln werden unter anderen Hyperparametern auch die Anzahl der *Hidden Layer* variiert, um das bestmögliche Ergebnis zu erzielen. Als Aktivierungsfunktion für die *Hidden Layer* wurde die *rectifier linear unit* (RELU) gewählt, da sie beim Ausprobieren sehr gute Ergebnisse zur Folge hatte. Auf die *Hidden Layer* folgt ein logarithmisches *Softmax Output Layer*. Die Größe dieses *Layers* richtet sich nach der Anzahl der verschiedenen Gesten. Für das erste Dataset sind es 8 verschiedene Gesten, für das zweite Dataset sind es 20.

4.2 Erstes Modell für 8 Gesten

4.2.1 Struktur des Datasets

Das erste Dataset war das initiale Experiment, um mit der Funktion von OpenPose vertraut zu werden. Die Größe des Datasets ist mit 280 Einträgen relativ gering und auch die Komplexität der verschiedenen Gesten ist recht niedrig. Alle gewählten Gesten sind für einen Menschen eindeutig zu erkennen und klar voneinander zu unterscheiden. Weiterhin sind alle Gesten bis auf *clap* (4.2b) statisch, was bedeutet, dass sie keine Bewegung enthalten. Mit diesem ersten Dataset soll bestätigt werden, dass solche einfachen Gesten mit der Kombination aus OpenPose und einem simplen neuronalen Netz klar klassifiziert werden können. Insgesamt wurden für das Dataset acht verschiedene Gesten ausgewählt (4.2). Die meisten Gesten sind solche, welche tatsächlich von einem Trainer oder Schiedsrichter in einer Sportart wie Fußball ausgeführt werden können (*idle*, *show*, *arm-up*, *t-pose* und *clap*), *dab* hingegen wurde willkürlich gewählt, um die Vielfalt zu erhöhen. Gesten wie *show-right* oder *show-left* (4.2g und 4.2h) könnten genutzt werden, um einem fußballspielenden Roboter ein Kommando zu erteilen.

(a) *idle* (b) *clap* (c) *dab* (d) *t-pose*

(e) *arm-up-right* (f) *arm-up-left* (g) *show-right* (h) *show-left*

Abbildung 4.2: Die Gesten des ersten Datasets

4.2.2 Evaluation

Zur Evaluation eines Modells auf dem Dataset wurde eine Hyperparameteroptimierung durchgeführt. Es wurden drei Modelle mit einem, zwei und drei *Hidden Layern* getestet. Außerdem wurde die Größe der *Hidden Layer* sowie die Anzahl der Trainingsepochen variiert. Die *batch size* wurde auf 10 gesetzt, die *learning rate* auf 0.01. Diese beiden Werte ergaben sich im Verlauf der Arbeit als ein guter Kompromiss zwischen der Trainingsdauer und der Genauigkeit der Konvergenz. Es wurde für jede mögliche Modell-Konfiguration zehn Mal eine K-Fold-Cross-Validation mit k = 4 durchgeführt und dann aus der Präzision des Modells in allen Durchläufen der Mittelwert und die Standardabweichung errechnet.

# of epochs \ Layer size	5	10	20	30
25	94.1% ±4.0%	97.3% ±2.2%	98.1% ±2.2%	98.4% ±1.4%
50	94.9% ±3.9%	97.9% ±2.3%	98.6% ±1.3%	98.5% ±1.5%
100	96.3% ±2.7%	98.2% ±2.1%	**98.9% ±1.8%**	**98.8% ±1.8%**
200	95.3% ±4.8%	98.4% ±1.8%	**98.9% ±1.5%**	**98.9% ±1.4%**
400	96.5% ±2.2%	98.5% ±1.6%	**98.8% ±1.6%**	**98.8% ±1.5%**

(a) 1 *Hidden Layer*

# of epochs \ Layer size	5	10	20	30
25	84.4% ±10.0%	95.6% ±3.4%	97.8% ±2.5%	97.6% ±2.8%
50	89.2% ±9.2%	96.8% ±2.7%	98.1% ±1.7%	98.1% ±2.2%
100	93.6% ±5.7%	97.5% ±1.9%	98.2% ±1.6%	**98.6% ±1.2%**
200	93.6% ±9.9%	97.8% ±1.4%	98.4% ±1.9%	**98.8% ±1.7%**
400	94.3% ±5.4%	97.1% ±2.1%	98.4% ±1.5%	**98.9% ±1.3%**

(b) 2 *Hidden Layer*

# of epochs \ Layer size	5	10	20	30
25	51.9% ±11.9%	79.1% ±11.4%	92.4% ±3.7%	94.1% ±2.9%
50	70.5% ±16.3%	91.6% ±4.3%	97.1% ±2.1%	97.9% ±1.9%
100	85.0% ±16.6%	96.4% ±2.1%	**98.1% ±2.0%**	**98.0% ±2.0%**
200	89.3% ±12.2%	97.2% ±2.0%	**98.0% ±2.1%**	**98.1% ±2.4%**
400	84.8% ±19.6%	97.1% ±2.0%	**98.2% ±2.0%**	**98.2% ±1.9%**

(c) 3 *Hidden Layer*

Abbildung 4.3: Evaluation des ersten Modells für verschiedene Hyperparameter

Die Evaluation zeigt, dass bereits ein einfaches neuronales Netz mit nur einem *Hidden Layer* eine sehr hohe Genauigkeit von bis zu 98.9% erreichen kann. Ein Grund für dieses gute Ergebnis ist sicher die Tatsache, dass die Anzahl und Komplexität der Gesten recht gering war und sie so ausgewählt wurden, um es dem Modell recht einfach zu machen. Dennoch ist es bereits ein bemerkenswertes Ergebnis, was darauf hindeutet, dass sich mit Hilfe von OpenPose sehr einfach verschiedene Gesten klassifizieren lassen.

4.3 Zweites Modell für 20 Gesten

4.3.1 Struktur des Datasets

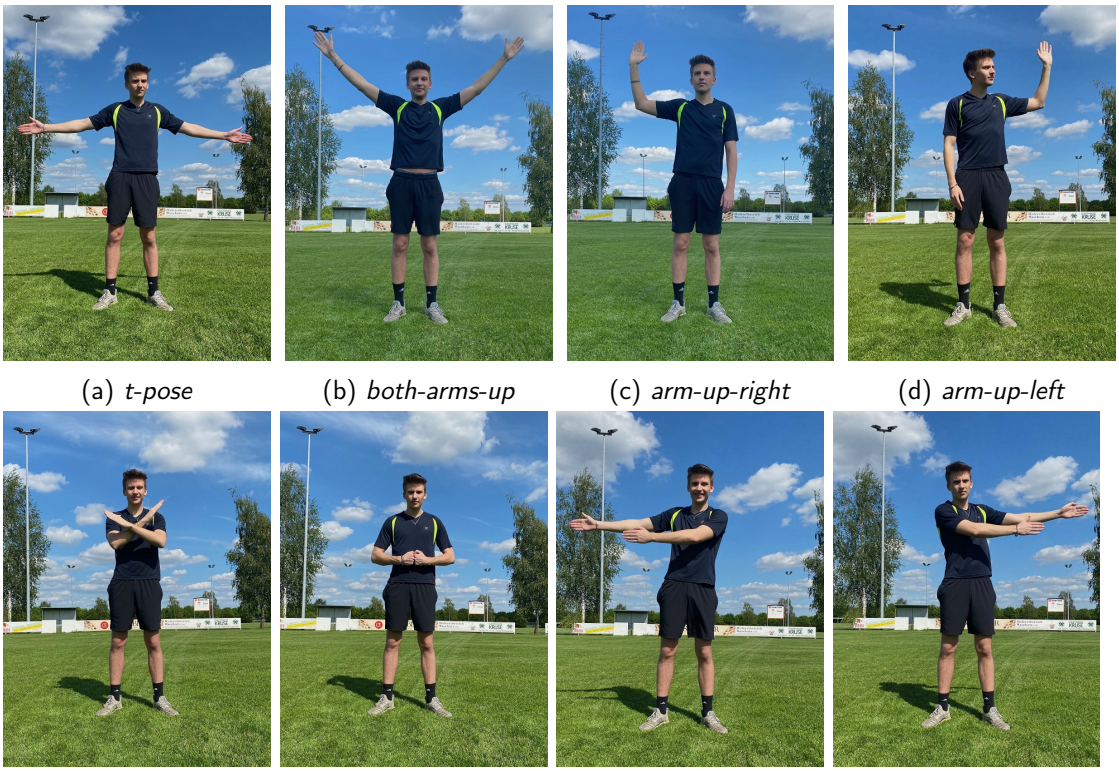
Im zweiten Dataset wurden sowohl die Anzahl als auch die Komplexität der Gesten erhöht, um mögliche Probleme bei der Gestenerkennung mit unserem System herauszufinden. Nachdem mit dem ersten Dataset gezeigt wurde, dass eindeutig unterschiedliche Gesten mit Leichtigkeit erkannt werden, wurden nun einige Gesten hinzugefügt, welche sich sehr stark ähneln und auch für Menschen schwer auseinander zu halten sind. Die ausgewählten Gesten sind sehr stark durch tatsächliche Gesten von Trainern und Schiedsrichtern aus verschiedenen Sportarten inspiriert. Ein Beispiel aus dem Fußball ist die Geste *show-up* (4.4c und 4.4d). Diese kann beispielsweise bedeuten, dass ein Foul passiert ist und nun ein Freistoß bevorsteht. Ein weiteres Beispiel ist die Geste *time-out* (4.4g und 4.4h), welche in diversen Sportarten ein Time-out anzeigt.

(a) *show-right* (b) *show-left* (c) *show-up-right* (d) *show-up-left*

(e) *arm-bow-right* (f) *arm-bow-left* (g) *time-out-low* (h) *time-out-high*

Abbildung 4.4: Gesten 1 - 8 des zweiten Datasets

Ein Beispiel für eine Gruppe von Gesten, welche eine Menge Ähnlichkeiten haben, sind die Gesten *time-out* (4.4g), *fists-together* (4.5f) und *clap* (4.6b). Bei jeder dieser Gesten werden die beiden Hände vor dem Oberkörper zusammen gehalten. Eine weitere Gruppe ähnlicher Gesten sind *t-pose* (4.5a), *both-arms-up* (4.5b), *cheer* (4.6c) und *complain* (4.6d). Sie alle haben gemeinsam, dass die Person die Arme ausgestreckt hat und auf der Höhe der Brust oder des Kopfes hält.



(a) *t-pose* (b) *both-arms-up* (c) *arm-up-right* (d) *arm-up-left*

(e) *cross-arms* (f) *fists-together* (g) *both-arms-right* (h) *both-arms-left*

Abbildung 4.5: Gesten 9 - 16 des zweiten Datasets

Außerdem wurden einige Gesten ausgewählt, die keine speziellen Kommandos darstellen, sondern zur natürlichen Körpersprache des Menschen im Kontext eines Fußballspiels gehören (4.6). Bei der Geste *idle* (4.6a) steht die Person mit verschränkten Armen, herabhängenden Armen oder den Händen in den Hosentaschen und verfolgt aufmerksam das Spiel. Bei der Geste *clap* (4.6b) applaudiert die Person, indem sie in beide Hände klatscht. Bei der Geste *cheer* (4.6c) streckt die Person beide Arme zum Jubeln in die Luft, bei *complain* (4.6d) ebenfalls, jedoch um sich stattdessen zu beschweren. Im Gegensatz zu den vorherigen Gesten sind dies dynamische Gesten mit einem Bewegungsablauf.

(a) *idle* (b) *clap* (c) *cheer* (d) *complain*

Abbildung 4.6: Gesten 17 - 20 des zweiten Datasets

Für dieses Dataset wurden je Geste circa zehn Bilder pro Person pro Betrachtungswinkel gemacht. Es gab zwei Models und drei verschiedene Betrachtungswinkel: Frontal, leicht von links und leicht von rechts. Insgesamt ergeben sich pro Geste also in etwa $10 * 2 * 3 = 60$ Bilder. Für einige ausgewählte Gesten wurden auch mehr Aufnahmen gemacht (*idle*, *clap*, *time-out-low* und *time-out-high*). Insgesamt enthält das Dataset 1813 Bilder.

Abbildung 4.7: Das erste Model aus verschiedenen Winkeln bei der Geste *both-arms-up*

Abbildung 4.8: Das zweite Model aus verschiedenen Winkeln bei der Geste *both-arms-up*

4.3.2 Evaluation

Für das zweite Modell wurde ebenfalls eine Hyperparameteroptimierung durchgeführt. Erneut wurden drei Modelle mit einem, zwei und drei *Hidden Layern* getestet, wobei die Größe der *Hidden Layer* sowie die Anzahl der Trainingsepochen variiert wurde. Die *batch size* ist nach wie vor 10 und die *learning rate* bei 0.01. Pro Konfiguration wurde erneut zehn mal eine K-Fold-Cross-Validation mit k = 4 durchgeführt, und dann Mittelwert und Standardabweichung berechnet.

# of epochs \ Layer size	10	20	35	50
25	91.7% ±1.6%	93.6% ±1.6%	94.4% ±1.1%	94.2% ±1.3%
50	93.0% ±1.5%	94.0% ±2.1%	94.8% ±1.7%	95.3% ±1.1%
100	93.7% ±1.3%	95.1% ±1.0%	95.2% ±1.3%	95.6% ±0.9%
200	94.0% ±1.4%	95.3% ±1.2%	95.5% ±1.5%	**95.7% ±1.2%**
400	93.7% ±1.4%	95.2% ±1.5%	**96.0% ±0.9%**	**95.9% ±1.0%**

(a) 1 *Hidden Layer*

# of epochs \ Layer size	10	20	35	50
25	89.4% ±3.1%	93.6% ±1.2%	94.3% ±1.4%	94.4% ±1.0%
50	91.4% ±1.7%	94.4% ±1.0%	95.1% ±1.5%	95.1% ±1.5%
100	92.3% ±1.5%	94.6% ±1.1%	95.2% ±1.4%	95.3% ±1.3%
200	93.3% ±1.4%	94.9% ±1.3%	**95.9% ±1.0%**	**96.0% ±1.1%**
400	93.2% ±1.7%	95.2% ±1.0%	95.7% ±1.1%	**95.9% ±1.2%**

(b) 2 *Hidden Layer*

# of epochs \ Layer size	10	20	35	50
25	85.1% ±3.8%	92.1% ±1.5%	94.0% ±1.4%	94.6% ±1.1%
50	88.6% ±2.4%	93.6% ±1.1%	95.1% ±1.0%	95.2% ±1.2%
100	90.6% ±2.5%	94.4% ±94.6%	95.5% ±1.4%	95.6% ±1.0%
200	91.2% ±1.9%	94.6% ±1.4%	95.5% ±1.0%	**95.9% ±0.9%**
400	91.6% ±3.2%	94.8% ±1.0%	**95.8% ±0.8%**	**95.9% ±1.1%**

(c) 3 *Hidden Layer*

Abbildung 4.9: Evaluation des zweiten Modells für verschiedene Hyperparameter

Auch auf dem komplexeren Dataset mit 20 Gesten schafft es unser System, eine Genauigkeit von bis zu 96% zu erreichen. Hierbei scheint die Anzahl der *Hidden Layer* keine große Rolle zu spielen, solange eine *layer size* von 35 oder mehr gewährleistet ist. Es lässt sich diskutieren, ob eine weitere Erhöhung der Anzahl der Trainingsepochen oder der *layer size* ein noch besseres Ergebnis bringen könnte, allerdings erkennt man am gesamten Verlauf, dass die Verbesserung der Präzision bei 400 Trainingsepochen und einer *layer size* von 50 schon stark abgeflacht ist.

Tabelle 4.1 zeigt die Gesten, welche bei einer Konfiguration mit 2 *Hidden Layern*, einer *layer size* von 40 und 100 Trainingsepochen die niedrigste Genauigkeit haben. Dazu gehören die Gesten *clap*, *arm-bow-left*, *show-up-left*, *cheer* und *complain*. Es lässt sich vermuten, dass diese Ungenauigkeit daraus resultiert, dass sowohl *clap* und *arm-bow-left* als auch *cheer* und *complain* sehr ähnlich aussehen und anhand der *body keypoints* nicht ganz eindeutig voneinander zu unterscheiden sind. Abbildung 4.10 zeigt zwei mögliche Kandidaten hierfür: Zwei sehr ähnliche *body keypoints* für die Gesten *cheer* und *complain*.

Geste	Genauigkeit
show-up-right	94%
arm-bow-left	91%
clap	89%
cheer	89%
complain	76%

Tabelle 4.1: Genauigkeit bei 2 Hidden Layern, Layersize 40 und 100 Trainingsepochen

(a) *cheer* (b) *complain*

Abbildung 4.10: Zwei verschiedene Gesten mit ähnlichen *body keypoints*

5 Multi-Frame-Gestenerkennung

In diesem Kapitel behandeln wir die Durchführung unseres Ansatzes für die Erkennung von Gesten für Sequenzen aus mehreren Bildern. Es wird also nicht mehr nur ein einziges Bild als Geste betrachtet, sondern eine Abfolge von zehn Bildern. Diese zehn Bilder werden nacheinander durch unser System gegeben, um der gesamten Sequenz am Ende eine Geste zuzuordnen. Wie im vorherigen Kapitel wird in 5.1 die gewählte Architektur des neuronalen Netzes beschrieben. Daraufhin wird in 5.2 das erstellte Dataset vorgestellt und in 5.3 die Genauigkeit unseres Ansatzes ausgewertet. Des Weiteren werden in 5.4 die Multi-Frame-Gestenerkennung und die Single-Frame-Gestenerkennung miteinander verglichen.

5.1 Aufbau des neuronalen Netzes

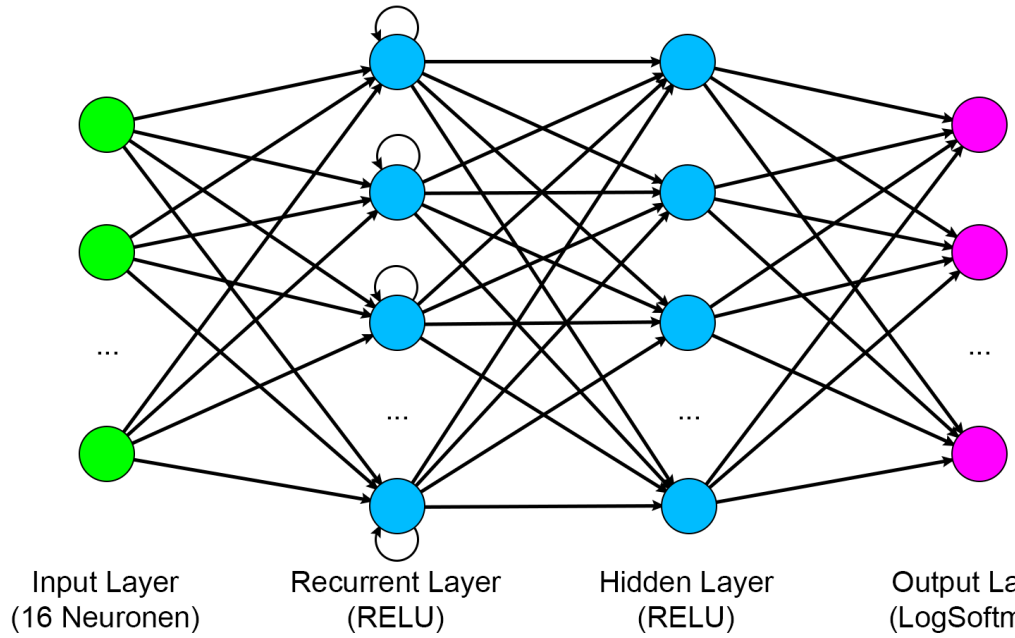


Abbildung 5.1: Rekurrentes neuronales Netz zur Multi-Frame-Gestenerkennung

Für die Multi-Frame-Gestenerkennung nutzen wir ein ähnliches neuronales Netz wie für Single-Frame-Gestenerkennung. Das *Input Layer* umfasst weiterhin 16 Neuronen und die genutzte Aktivierungsfunktion ist die *rectifier linear unit* (RELU). Um auch dynamische Gesten erfolgreich zu erkennen, haben wir jedoch eine Anpassung gemacht. Auf das *Input Layer* folgt ein *Recurrent Layer*, welches die Eigenschaft hat, dass die Neuronen nicht nur Verbindungen zu der nächsten Schicht, sondern auch zu der eigenen Schicht haben. Dadurch beeinflusst ein Input auch alle darauf folgenden Inputs. Geben wir nun also eine Sequenz aus zehn Bildern durch unser System, so hat bereits das erste Bild Auswirkungen auf das Gesamtergebnis. Die gesamte Sequenz durchläuft zuerst dieses *Recurrent Layer*, welches beim Abschluss einen Output liefert. Dieser Output wird daraufhin in ein weiteres *Hidden Layer* gegeben, welches dann zum *Output Layer* führt. In der Evaluation werden wie im vorherigen Kapitel verschiedene Konfigurationen mit unterschiedlichen Anzahlen der *Recurrent Layer* und *Hidden Layer* getestet.

5.2 Struktur des Datasets

Da unser Ansatz bei beiden Datasets der Single-Frame-Gestenerkennung sehr gut abgeschnitten hat, wurde bei der Auswahl der Gesten für dieses Dataset der Fokus darauf gerichtet, es dem System möglichst schwer zu machen. Es wurden sechs ähnliche Gesten gewählt, bei denen sich die beiden Arme vor dem Körper befinden. Hierbei sind die drei Gesten *clap* (5.2a), *spin* (5.3a) und *dance* (5.3c) dynamisch, die Gesten *fold* (5.2b), *idle* (5.3b) und *time-out* (5.3d) hingegen statisch.

(a) *clap*: Die Person klatscht in die Hände

(b) *fold*: Die Person steht still mit gefalteten Händen

Abbildung 5.2: Die Gesten *clap* und *fold*

Besonders bei den Gesten *clap* (5.2a) und *fold* (5.2b) kann es vorkommen, dass sich einzelne Frames sehr ähnlich sind, da sich beide Hände übereinander vor dem Körper der Person befinden. Selbiges gilt für die Gesten *spin* (5.3a), *idle* (5.3b) und *dance* (5.3c), bei welchen die Arme häufig eine verschränkte Stellung einnehmen.

(a) *spin*: Die Person rotiert beide Arme vor dem Körper

(b) *idle*: Die Person steht still mit verschränkten Armen

(c) *dance*: Die Person tanzt, beide Arme werden vor dem Körper bewegt

(d) *time-out*: Die Person bildet mit beiden Händen ein Time-out-Zeichen

Abbildung 5.3: Die Gesten *spin*, *idle*, *dance* und *time-out*

Für dieses Dataset wurde jede Geste von drei verschiedenen Personen ausgeführt. Des Weiteren wurde wieder der Kamerawinkel variiert, sodass jede Person frontal, leicht von links und leicht von rechts gefilmt wurde. Es wurden jeweils zehn Sekunden aufgenommen, also ergeben sich $3 * 3 * 10 = 90$ Sekunden Videomaterial pro Geste. Dieses wurde dann zu 90 eine Sekunde langen Sequenzen mit 10 Bildern pro Sekunde verarbeitet. Insgesamt enthält das Dataset also $6 * 90 = 540$ Sequenzen, die jeweils aus 10 Frames bestehen.

Abbildung 5.4: Das erste Model aus verschiedenen Winkeln

Abbildung 5.5: Das zweite Model aus verschiedenen Winkeln

Abbildung 5.6: Das dritte Model aus verschiedenen Winkeln

5.3 Evaluation

Auch für dieses Modell wurde eine Hyperparameteroptimierung durchgeführt. Es wurden drei verschiedene Konfigurationen für die Anzahl der *Recurrent Layer* und *Hidden Layer* gewählt: 1 *Recurrent Layer* und 1 *Hidden Layer* (5.7a), 2 *Recurrent Layer* und 1 *Hidden Layer* (5.7b) sowie 1 *Recurrent Layer* und 2 *Hidden Layer* (5.7c). Es wurde weiterhin mit einer *batch size* von 10 trainiert, jedoch wurde die *learning rate* auf 0.005 halbiert, da da Standardabweichung ansonsten sehr hoch ausfiel.

Layer size / # of epochs	10	20	30	40
25	93.6% ±4.1%	95.2% ±3.4%	95.0% ±4.1%	94.5% ±6.0%
50	96.1% ±2.1%	95.7% ±4.0%	96.7% ±2.7%	96.5% ±3.0%
100	97.2% ±1.7%	97.1% ±3.3%	98.3% ±1.7%	98.4% ±2.4%
200	98.0% ±1.8%	98.6% ±1.1%	98.6% ±1.4%	**99.0% ±0.8%**
400	98.1% ±1.3%	98.7% ±1.1%	98.6% ±1.1%	98.7% ±1.2%

(a) 1 *Recurrent Layer*, 1 *Hidden Layer*

Layer size / # of epochs	10	20	30	40
25	90.6% ±5.8%	90.7% ±7.5%	91.9% ±7.8%	91.9% ±9.0%
50	95.3% ±3.6%	96.4% ±2.8%	95.6% ±5.5%	96.7% ±2.5%
100	95.1% ±13.4%	98.4% ±1.9%	98.5% ±2.0%	98.9% ±1.1%
200	98.4% ±1.3%	**99.0% ±0.8%**	**99.1% ±1.1%**	**99.1% ±0.9%**
400	98.5% ±1.1%	98.9% ±0.8%	96.8% ±13.0%	99.0% ±1.1%

(b) 2 *Recurrent Layer*, 1 *Hidden Layer*

Layer size / # of epochs	10	20	30	40
25	88.6% ±9.2%	93.6% ±4.6%	93.4% ±4.5%	94.0% ±5.1%
50	93.4% ±4.8%	95.2% ±7.2%	96.5% ±2.4%	96.4% ±3.2%
100	96.6% ±4.6%	97.9% ±2.4%	98.5% ±1.4%	98.3% ±1.8%
200	98.3% ±1.4%	98.1% ±2.0%	**98.7% ±1.0%**	**98.6% ±1.1%**
400	98.0% ±1.3%	98.3% ±1.0%	**98.6% ±1.0%**	**98.8% ±1.3%**

(c) 1 *Recurrent Layer*, 2 *Hidden Layer*

Abbildung 5.7: Evaluation für verschiedene Hyperparameter

Obwohl wir uns bei der Erstellung das Datasets bemüht haben, es unserem System schwierig zu machen, fallen die Ergebnisse extrem gut aus: Alle drei Konfigurationen erreichen bei einer ausreichenden *layer size* und genügend Trainingsepochen eine Präzision von annähernd 99%. Zum Vergleich: das Modell zur Single-Frame-Gestenerkennung erreichte auf 540 Bildern dieses Datasets in etwa 93%. Es ist uns also nicht vollständig gelungen, unser System herauszufordern, aber es ist dennoch ersichtlich, dass das Hinzufügen eines *Recurrent Layer* sowie das Betrachten von mehreren Frames eine beachtliche Verbesserung mit sich brachte. Man kann erneut diskutieren, ob eine Erhöhung der *layer size* auf über 40 einen weiteren Anstieg in der Präzision mit sich bringen könnte, dieser dürfte aber nur minimal ausfallen, zumal 99% ein bereits sehr gutes Ergebnis ist.

5.4 Vergleich mit Single-Frame-Gestenerkennung

Rein qualitativ betrachtet ist die Multi-Frame-Gestenerkennung der Single-Frame-Gestenerkennung voraus, da sie eine ganze Sequenz an Frames für die Gestenerkennung berücksichtigt und somit bessere Chancen hat, auch dynamische Gesten richtig festzustellen. Sie bietet langfristig mehr Möglichkeiten in ihrer Entwicklung. Beispielsweise könnte man das System dahingehend verbessern, komplexere Gesten zu erkennen, wie das einmalige Winken mit einem Arm. In ferner Zukunft wäre es sogar denkbar, mittels Multi-Frame-Gestenerkennung die Bewegungen eines gegnerisches Fußballspielers präzise einzuschätzen. So könnte beispielsweise festgestellt werden, dass ein gegnerischer Spieler mit dem Bein zum Schuss ausholt, und entsprechend reagiert werden.

Nichtsdestotrotz hat die Multi-Frame-Gestenerkennung auch Nachteile. Es muss stets eine hohe Framerate gewährleistet sein, um genügend Frames zu erhalten. Wie in 2.3.2 besprochen, sind wegen OpenPose hierzu eine gute Hardware sowie ausreichend Ressourcen notwendig. Außerdem ist es aufwendiger, ein rekurrentes neuronales Netz zu trainieren, da die Trainingsdaten um eine zeitliche Dimension erweitert werden. Die Single-Frame-Gestenerkennung hingegen kann bereits mit kleinen Datasets trainiert werden und bietet sich an, um statische Gesten zu erkennen. Sie könnte ein guter Startpunkt für die Nutzung von Gestenerkennung im RoboCup sein. Im Vergleich zur Multi-Frame-Gestenerkennung reicht ein einziger Blick auf eine Person, um eine Geste festzustellen. Es muss also keine hohe Framerate gewährleistet sein, wodurch die Single-Frame-Gestenerkennung auch recht ressourcenschonend eingesetzt werden kann.

6 Zusammenfassung

Der in dieser Arbeit präsentierte Ansatz zur Gestenerkennung zeigte erstaunlich gute Ergebnisse auf den Datasets, die im Rahmen dieser Arbeit angefertigt wurden. Durch die Nutzung von OpenPose konnten Bilder von Personen in eine vereinfachte, aber dennoch sehr informative Darstellung einer Pose überführt werden. Durch diese Einfachheit der Daten konnte bereits mit simplen neuronalen Netzen und rekurrenten neuronalen Netzen eine Präzision von 96% und 99% erreicht werden.

Das entwickelte System hat gute Voraussetzungen, um für den RoboCup eine erste Kommunikation zwischen Trainer und fußballspielenden Robotern zu ermöglichen. Ein erster Ansatz wäre, eine Gruppe von Kommandos zu definieren und diesen verschiedene statische Gesten zuzuordnen. Es könnte dann ressourcenschonend in regelmäßigen Abständen die Gestik des Trainers analysiert und im Falle eines Kommandos entsprechend reagiert werden.

Weiterhin bietet der präsentierte Ansatz Möglichkeiten zur weiteren Entwicklung und Verbesserung. Beispielsweise könnten durch OpenPose auch die *keypoints* der Hände ausgegeben werden, um Gesten zu erkennen, die durch die Hände ausgeführt werden.

Außerdem werden das Dataset aus Kapitel 4.3.1 mit 1800 Bildern und das Dataset aus Kapitel 5.2 mit 540 Sequenzen nach dieser Arbeit als wissenschaftlicher Beitrag zur Verfügung gestellt.

Literaturverzeichnis

[1] Hamburg Bit-Bots. Willkommen. https://robocup.informatik.uni-hamburg.de/willkommen. [Letzter Zugriff am 29.05.2020].

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[3] H. Kitano, M. Asada, Y. Kuniŷoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. *Proceedings of the First International Conference on Autonomous Agents*, pages 340–347, 1997.

[4] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup: A challenge problem for ai. *AI magazine*, 18(1):73–85, 1997.

[5] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[6] Louis-Philippe Morency, Ariadna Quattoni, and Trevor Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[7] Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 237–242, 1991.

[8] OpenPose. Performance. https://docs.google.com/spreadsheets/d/1-DynFGvoScvfWDA1P4jDInCkbD4lg0IKOYbXgEq0sK0/edit#gid=248758262. [Letzter Zugriff am 29.05.2020].

[9] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 27–38, 2013.

[10] RoboCup. A brief history of robocup. https://www.robocup.org/a_brief_history_of_robocup. [Letzter Zugriff am 29.05.2020].

[11] RoboCup. Objective. https://www.robocup.org/objective. [Letzter Zugriff am 29.05.2020].

[12] RoboCup. Robocupsoccer - humanoid league. https://www.robocup.org/leagues/3. [Letzter Zugriff am 29.05.2020].

[13] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14, 2008.

[14] Eleni Tsironi, Pablo VA Barros, and Stefan Wermter. Gesture recognition with a convolutional long short-term memory recurrent neural network. In *ESANN*, 2016.

[15] Sy Bor Wang, Ariadna Quattoni, L-P Morency, David Demirdjian, and Trevor Darrell. Hidden conditional random fields for gesture recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1521–1527. IEEE, 2006.

[16] Ying Wu and Thomas S Huang. Vision-based gesture recognition: A review. In *International Gesture Workshop*, pages 103–115. Springer, 1999.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den 05.06.2020

Lennart Uhrmacher

Lennart Uhrmacher

Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

Hamburg, den 05.06.2020

Lennart Uhrmacher

Lennart Uhrmacher