TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Entwicklung eines Modells zur Handgestenklassifikation am Beispiel eines Interaktionssystems in der industriellen Baustellenmontage

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs

unter der Leitung von

Univ.-Prof. Dr.-Ing. Sebastian Schlund
(E330 Institut für Managementwissenschaften,
Bereich: Human Centered Cyber Physical Production and Assembly Systems)

Patrick Rupprecht, MSc MSc MA
(E330 Institut für Managementwissenschaften,
Bereich: Human Centered Cyber Physical Production and Assembly Systems)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

von

Martin Gratl

01128206

Vivenotgasse 17

1120 Wien

Wien, im April 2021

_____

Martin Gratl

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre hiermit Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im April 2021

_____

Martin Gratl

Danksagung

Die folgenden Worte gelten den Personen, die mich bei der Umsetzung der vorliegenden Arbeit unterstützt haben und bei denen ich mich herzlich bedanken möchte.

Als Erstes möchte ich Herrn Prof. Dr. Sebastian Schlund meine Dankbarkeit ausdrücken, für die Möglichkeit meine Diplomarbeit in seinem äußert interessanten Forschungsgebiet schreiben zu dürfen. Herrn Patrick Rupprecht MSc MSc MA möchte ich hiermit für die sehr gute Zusammenarbeit und kompetente Betreuung meiner Diplomarbeit danken.

Meiner Freundin Lilli gebührt ein besonderer Dank. Sie hat mich stets ermutigt und hatte immer ein offenes Ohr mich. Ich möchte mich auch bei ihren Eltern Resi und Robert bedanken, die mir immer mit Rat und Tat zur Seite standen.

An dieser Stelle möchte ich mich auch bei meinen Studienkollegen, Freunden und allen Personen, die mich auf dem Weg meines Studiums begleitet haben, bedanken. Ein spezieller Dank geht dabei auch an Lena, Lukas und Stefan für das Korrekturlesen meiner Arbeit.

Abschließend möchte ich mich bei meinen Eltern Maria und Gebhard bedanken. Sie haben mir dieses Studium ermöglicht und mich während dieser Zeit immer unterstützt und gefördert.

Kurzfassung

Bei der Montage großer Bauteile wird häufig die industrielle Baustellenmontage als Organisationsform gewählt. Bei dieser Organisationsform kann die Informationsbereitstellung Einsparpotential bieten. Der Stand der Technik ermöglicht es bereits Anweisungen und Informationen für den Monteur direkt am Bauteil darzustellen. Die Interaktion mit solchen Werkerinformationssystemen kann als Handgestensteuerung konzipiert werden, um Wegzeiten bei der Informationsbereitstellung einzusparen. Techniken des maschinellen Lernens (engl. *Machine Learning*) und des maschinellen Sehens (engl. *Computer Vision*) erlauben es dabei unterschiedliche Handgesten zu erkennen, zu klassifizieren und in weiterer Folge Interaktionen auszulösen. Zur Erstellung eines ML- Klassifikationsmodels ist ein adäquater Datensatz, der die Funktionen des Interaktionssystems abbildet, notwendig. Die vorliegende Arbeit untersucht Handgestendatensätze und analysiert selbige auf deren Brauchbarkeit in der industriellen Baustellenmontage. Nachdem Eckpunkte und Charakteristiken von vorhandenen Datensätzen aufbereitet worden sind, wird eine Nutzwertanalyse zur Bewertung der Eignung der Datensätze für die industrielle Baustellenmontage durchgeführt. Die wichtigsten Erkenntnisse aus der Analyse sind die zunehmende Menge an Daten über die letzten Jahre, die sich mit der vorherrschenden allgemeinen Entwicklung hin zu einer erhöhten Datenmenge deckt. Bei der durchgeführten Nutzwertanalyse stellte sich heraus, dass der Einsatz von vordefinierten Handgestendatensätzen für die industrielle Baustellenmontage oft nicht zielführend ist. Das Erstellen eines neuen Datensatzes stellt somit eine sinnvolle Alternative dar. Auf Basis der Erkenntnisse aus der Datensatz-Analyse und einschlägiger Normen wird ein Datensatz für statische Handgesten erstellt. Insgesamt besteht er aus 2400 Bildern eingeteilt in 6 Gestenkategorien durchgeführt von 2 Akteuren. Der Datensatz wird anschließend verwendet um mit Techniken des maschinellen Lernens sowie des maschinellen Sehens einen Algorithmus zu trainieren der effektiv und in Echtzeit verschiedene Gesten unterscheidet. Unter Verwendung der Programmiersprache *Python* und ihrer *Deep Learning* Bibliotheken *Tensorflow* und *Keras* wird der Algorithmus mit Hilfe der Technik „*Transfer learning*“- Technik trainiert. Als Basismodell des Algorithmus wird ein leichtgewichtiges *Convolutional Neural Network* (CNN), nämlich das MobileNetV2 Modell verwendet. Die Parameter des Modells werden in einem iterativen Prozess angepasst. Schließlich wird das Klassifizierungsmodell auf einem Testdatensatz, der mit drei Testpersonen aufgezeichnet wurde, getestet. Mit einer durchschnittlichen Genauigkeit von 99,6% konnten die unterschiedlichen Gesten korrekt klassifiziert werden. Ein Test in Echtzeit lieferte ein schwächeres Ergebnis mit 94,6% korrekter Gestenvorhersagen über die Testdauer.

Abstract

Industrial site assembly is often chosen as the organizational form for the assembly of large components. In this form of organization, the provision of information can offer potential savings. The state of the art already makes it possible to display instructions and information for the worker directly on the component. The interaction with such worker information systems could be designed as hand gesture control to save travel time for information provision. Machine Learning and Computer Vision techniques allow to recognize and classify different hand gestures and to trigger interactions. In order to build a ML classification model, an adequate dataset representing the features of the interaction system is necessary. This thesis investigates hand gesture datasets and analyzes them for their usefulness in industrial site assembly. After key points and characteristics of existing datasets have been prepared, a utility analysis is performed to evaluate the suitability of the datasets for industrial site assembly. The main findings from the analysis are the increasing amount of data over the last years, which is in line with the prevailing general trend towards an increased amount of data. During the utility analysis performed, it was found that the use of predefined hand gesture datasets for industrial site assembly is often not purposeful. Thus, the creation of a new data set represents a reasonable alternative. Based on the findings of the data set analysis and industry standards, a data set for static hand gestures is created. In total, it consists of 2400 images divided into 6 gesture categories performed by 2 actors. The dataset is then used to train an algorithm that effectively distinguishes different gestures in real time using machine learning techniques. Using the Python programming language and its deep learning libraries Tensorflow and Keras, the algorithm is trained using the transfer learning technique. A lightweight Convolutional Neural Network (CNN), namely the MobileNetV2 model, is used as the base model of the algorithm. The parameters of the model are adjusted in an iterative process. Finally, the classification model is tested on a test dataset, recorded with three subjects. With an average accuracy of 99.6%, the different gestures could be correctly classified. A test in real time provided a weaker result with 94.6% correct gesture predictions over the test duration.

Inhaltsverzeichnis

1 Einleitung

1.1 Einführung in das Themenfeld

Das maschinelle Sehen (engl. *Computer Vision*) ist ein Forschungsgebiet, welches sich mit Problemen beschäftigt, die einen Sehsinn erfordern. Hierzu zählen beispielsweise die Erkennung von Objekten in Bildern oder Videos. Für den Menschen ist es ein Leichtes, Objekte zu erkennen und zuzuordnen. Für Maschinen hingegen ist dieser Prozess deutlich anspruchsvoller. Es bedarf einiger Bearbeitungsschritte, um mittels *Computer Vision* (CV) Informationen aus Bildmaterialien zu erkennen und korrekt einordnen zu können.

Bis vor einigen Jahren konnten mittels *Computer Vision* Probleme nur in begrenztem Umfang bzw. geringer Komplexität gelöst werden. Ein Grund für diese Einschränkung war die limitierte Rechenleistung und die geringe Datenmenge, die zur Verfügung stand. Dank neuester Entwicklungen im Bereich der künstlichen Intelligenz und Innovationen in den Bereichen *Deep Learning* (DL) und künstlicher neuronaler Netze konnte das Gebiet jedoch in den letzten Jahren große Fortschritte erzielen. Auch die Datenmenge hat sich in der Zwischenzeit vervielfacht. Einsatz findet Computer Vision beispielweise in selbstfahrenden Autos/Kraftfahrzeugen, in der medizinischen Diagnostik, Gesichtserkennung (z. B. am Smartphone), in Drohnen und Robotern oder bei Steuerungskonzepten. In der Industrie kann CV etwa bei der Analyse von Beständen, Qualitätskontrollen, Prozessoptimierungen oder Mensch-Maschine-Interaktionskonzepten wie etwa der Steuerung mittels Gesten eingesetzt werden.

Beim sog. Maschinellen Lernen (engl. *Machine Learning*) lernen Programme aus vorhandenen Daten und wenden dieses Wissen auf neue Daten an (Géron, 2019). Hierbei analysiert ein Algorithmus (die Maschine) Daten, erkennt Muster und versucht somit zukünftige Ergebnisse vorherzusagen. Im Bereich des maschinellen Lernens gibt es eine Reihe verschiedener Lernvarianten. Eine davon ist das sog. „Tiefe Lernen“ (eng. *Deep Learning*). Diese Lernvariante nutzt künstliche neuronale Netze mit diversen Schichten, die Informationen auf mehreren Ebenen verarbeitet. Dabei ist kein manuelles Eingreifen nötig. Die Maschine lernt aus Rohdaten, ähnlich wie ein menschliches Gehirn, wobei sie verschiedene Arten sensorischer Eingaben nutzt.

Obwohl die historischen Wurzeln des *Deep Learning* mehrere Jahrzehnte in die Vergangenheit zurück gehen (Schmidhuber, 2015), wurde diese Art des maschinellen Lernens erst in den letzten zehn Jahren durch die gesteigerte Rechenleistung praktikabel. Ein Durchbruch gelang zu Beginn der 2010er Jahre mit der Publikation *ImageNet Classification with Deep Convolutional Neural Networks* (Krizhevsky et al., 2012), welches für die damalige Zeit herausragende (State of the Art) Resultate bei

der ImageNet Challenge lieferte. Die ImageNet Challenge, ein Software-Wettbewerb im Bereich der Objekterkennung, findet seit dem Jahr 2010 alljährlich statt. Das Ziel besteht darin, Objekte auf dem herausfordernden ImageNet-Datensatz korrekt zu erkennen und zu klassifizieren (Deng et al., 2009). Der große Erfolg von *Deep Learning* ist vor allem auch den großen, qualitativ hochwertigen, öffentlich zugänglichen und gekennzeichneten Datensätzen geschuldet. In Kombination mit einer erhöhten Rechenleistung ermöglichen es neuronale Netze, Muster effektiv zu erlernen (Voulodimos et al., 2018, S. 1). Eine Form der neuronalen Netze im Deep Learning sind die *Convolutional Neuronal Networks* (CNN), die vor allem in der Klassifikation von Bilddateien vielfach verwendet werden.

Moderne *Open Source Libraries* für *Deep Learning* wie *Tensorflow* und *Keras* erlauben es, sinnvoll und schnell mittels State-of-the-Art-Techniken Bildklassifizierungsprobleme zu erstellen und zu testen. Unabdingbar sind hierbei jedoch Datensätze, die der Algorithmus für seinen Lernprozess benötigt. Erst dadurch werden sinnvolle Vorhersagen möglich.

In der Mensch Maschine Interaktion (MMI) gibt es Herausforderungen, welche mit Ansätzen der Computer Vision in Kombination mit maschinellen Lernmethoden bearbeitet werden können. Dazu zählen etwa Arbeitsplatzgestaltung, Körperhaltungsanalysen oder Steuerungskonzepte. Ein solches Steuerungskonzept könnte z. B in geräuschintensiven Bereichen und Umgebungen sinnvoll mit Hilfe gestischer Interaktion gelöst werden.

1.2 Problemstellung und Forschungsfragen

Die industrielle Montage von Großgeräten wird oft als Baustellenmontage realisiert. Herkömmliche Informationsbereitstellung mittels Papiers oder PC-Terminal sind dabei oft mit längeren Wegzeiten verbunden. Um dieses Einsparungspotential zu nutzen, präsentierten Rupprecht et al. (2020) ein dynamisches Projektionssystem, welches Arbeitsanweisungen direkt am Bauteil anzeigt. Die Informationsbereitstellung durch dieses Projektionssystem bietet im Vergleich zu jener mittels PC-Terminal erhebliche Vorteile zur Prozessverbesserung und in der Benutzerfreundlichkeit. Zum Zeitpunkt der Veröffentlichung wurden Interaktionen mit der projizierten Benutzeroberfläche noch manuell durchgeführt (Rupprecht et al., 2020). Zu diesem Zweck bietet sich die Entwicklung eines Steuerkonzeptes mittels Gesteninteraktion an. Dabei ist ein zuverlässiges und robustes Modell zur Klassifizierung von verschiedenen Gesten unerlässlich.

An dieser Stelle soll die vorliegende Diplomarbeit anknüpfen und zunächst den aktuellen Stand der Technik im Bereich der Gestensteuerung aufbereiten. Zudem sollen State-of-the-Art-Machine Learing-Techniken vorgestellt und anschließend im Praxisteil der Arbeit angewendet werden. Ein Problem bei der Implementierung sind

verfügbare „gelabelte“ Datensätze. Gelabelte Daten bezeichnen Gruppen von Proben, die mit einem Label versehen sind und bei einem Klassifikationsproblem einer Klasse oder einer Kategorie entsprechen. Der Vorgang des Einlernens des Algorithmus wird als „Training“ bezeichnet. Eine Vielzahl von Datensätzen, die zur Erkennung von Handgesten verwendet werden, sind öffentlich verfügbar. Sie wurden im Rahmen der Forschung in den Themenbereichen *Computer Vision* und *Deep Learning* entwickelt. Viele dieser Datensätze sind jedoch insofern problematisch, als dass sie sehr unterschiedlich in ihren Ausprägungen sind und oft nur in sehr speziellen Szenarien anwendbar sind. Die Art der enthaltenen Gesten sind oft breit gefächert, was der Komplexität des Themengebietes Gestik geschuldet ist. Es deckt nämlich ein sehr breites Spektrum von komplexer Gebärdensprache bis hin zu einfachen Zeigegesten ab.

Folgende Forschungsfragen ergeben sich aus der Ausganglage:

„Welche Datensätze eignen sich zum Trainieren von Computer Vision-Gestenerkennung für Hand- und Armgesten und wie können diese kategorisiert werden?“

„Welche Datensätze sind für das Training eines Handgesten-Erkennungs-Modells in der industriellen Baustellenmontage geeignet und welche Eigenschaften sind dafür entscheidend?“

Um die Problemstellung lösen zu können, sind folgende Teilfragen zu beantworten:

- Welche Datensätze sind für das Trainieren von Gestenerkennungsmodellen vorhanden?
- Welche Eigenschaften charakterisieren die Gesten-Datensätze?
- Welche Datensätze sind in Bezug auf Bildqualität, Verfügbarkeit und Datenschutzkonformität vorhanden?
- Wie lassen sich die Gestendatensätze kategorisieren?

1.3 Lösungsansatz

Um die Forschungsfragen und die daraus abgeleiteten Teilfragen beantworten zu können, muss zunächst eine Einführung in die theoretischen Grundlagen der Themengebiete erfolgen. Hierbei wird insbesondere auf Interaktionsformen mittels Gesten eingegangen. Es werden unterschiedliche Einteilungsformen und Taxometrien betrachtet, um bei der anschließenden Analyse der Gestendatenbanken Einteilungen vornehmen zu können. Der zweite zentrale Punkt des theoretischen Teils besteht in

der Aufbereitung theoretischer Grundlagen in den Bereichen *Computer Vision* und *Deep Learning*.

Um die wesentlichen Charakteristiken, Unterschiede und Restriktionen der am Markt vorhandenen Gestendatenbanken bewerten zu können, soll eine State-of-the-Art-Recherche durchgeführt werden. Einen besonderen Schwerpunkt sollen dabei der Aufbau, die Datenmenge und die Zusammensetzung der vorhandenen Datensätze bilden. Anschließend werden Eckdaten und Aufbau aller betrachteten Datensätze tabellarisch zusammengefasst. Die Tabelle bildet den Ausgangspunkt für eine statistische Analyse. Mit Hilfe dieser Analyse können Best Practice und Trends bei der Auswahl einer Gestendatenbank herausgearbeitet werden und die Ergebnisse visualisiert werden. Mit Hilfe einer Nutzwertanalyse können anschließend Kriterien für den Einsatz in der industriellen Baustellenmontage festgelegt werden und geeignete Datensätze erkannt werden.

In weiterer Folge wird im praktischen Teil der vorliegenden Arbeit ein Modell zur Klassifizierung von Handgesten trainiert und mit Hilfe eines Versuchs validiert und getestet. Dieses Modell kann als Grundlage für ein Steuerungskonzept für die Informationsbereitstellung in der industriellen Baustellenmontage dienen.

Mittels *Convolutional Neuronal Networks* (CNNs) und dem *Tensorflow & Keras Framework* soll der Datensatz eingelernt werden und anschließend mit Gestenbildern, die der Algorithmus nie zuvor gesehen hat, getestet werden. Der Test der Gestenklassifizierung wird mit mehreren Personen durchgeführt. Zur Aufnahme der Bilder sowie der Datenvorverarbeitung wird eine konventionelle Webcam sowie die *Python Library OpenCV* verwendet. Das Einlernen erfolgt einmalig mittels *Cloud Computing*. Anschließend kann das Klassifizierungsmodell auf seine Genauigkeit auf Testbildern und im Echtzeitversuch getestet werden. Schließlich werden die Ergebnisse des Versuchs ausgewertet.

1.4 Aufbau und Struktur der Arbeit

Aufbauend auf der allgemeinen Einführung in das Themenfeld wurde in den vorherigen Kapitel 1.2 und 1.3 die Problemstellung, das Einsatzgebiet sowie die daraus abgeleiteten Forschungsfragen definiert. Zum Abschluss dieser Einführungskapitel wird ein Überblick über die folgende Arbeit gegeben.

Um ein grundlegendes Verständnis des Einsatzgebietes sowie der Interaktion zwischen Mensch und Maschine zu erlangen wird im folgenden Kapitel 2 zunächst auf diese Themen eingegangen. Im Anschluss wird das Themengebiet der natürlichen

Interaktion erläutert. Zum Abschluss des theoretischen Grundlagenkapitels werden die Grundlagen zu den Themen *Machine Learning* und *Deep Learning* erläutert.

Kapitel 3 stellt die Methoden und Konzepte, die in der vorliegenden Arbeit verwendet werden vor. Zunächst wird die Vorgehensweise bei einer Nutzwertanalyse erklärt. Im Anschluss daran wird die Vorgehensweise bei *Supervised Machine Learning* Klassifikationsaufgaben erläutert und Leistungsmetriken vorgestellt. Schließlich wird auf die *Transfer Learning* Technik eingegangen.

In Kapitel 4.1 wird eine Analyse zu aktuellen Handgesten-Datenbanken durchgeführt. Ziel dabei ist es die Datensätze zu klassifizieren um anschließend werden in Kapitel 4.2 Trends und Best Practice bei Handgesten-Datensätzen herauszuarbeiten. In Kapitel 4.3 wird eine Nutzwertanalyse zum Einsatz von Handgestendatensätzen in der industriellen Baustellenmontage durchgeführt.

Das Kapitel 5 stellt den Use Case für den praktischen Teil der Arbeit vor. Zudem wird der Datensatz für den Use Case und dessen Zusammensetzung näher erläutert.

Das Kapitel 6 erläutert die Modellierung in den *Python* Bibliotheken *Tensorflow* und *Keras* und geht im Speziellen auf den iterativen Prozess bei der Modellierung ein.

Das Kapitel 7 stellt die Vorgehensweise beim Testversuch dar. Es wird auf den Versuchsaufbau und die Durchführung eingegangen. In Kapitel 8 werden die Versuchsergebnisse nach den Leistungsmetriken aus Kapitel 3 ausgewertet.

Abschließend folgt in Kapitel 9 eine Diskussion der Ergebnisse sowie ein Ausblick auf zukünftige Schritte.

2 Theoretische Grundlagen

In Kapitel 2 wird zunächst auf die Organisationsform der industriellen Baustellenmontage eingegangen. Anschließend werden wichtige Begriffe und das Einsatzgebiet von Gestensteuerungen erläutert. In weiterer Folge wird die Thematik Gestensteuerung und das Forschungsfeld, in welchem jene Anwendung findet, näher ausgeführt. Schließlich folgen in Kapitel 2.4 und 2.5 die Grundlagen zu den Themen *Machine Learning* und *Deep Learning*. Im Speziellen wird auf *Convolutional Neural Networks* eingegangen, welche insbesondere in der Bilderkennung Anwendung finden.

2.1 Industrielle Baustellenmontage

Das Montieren umfasst laut der VDI-Richtlinie 2860 die Gesamtheit aller Vorgänge, die dem Zusammenbau von geometrisch bestimmten Körpern dienen. Eine Organisationsform für die Montage von Werkstücken mit großen Abmessungen und Gewichten stellt die Baustellenmontage dar. Anwendung findet dieses Prinzip unter anderem im Anlagenbau oder im Großmaschinenbau (Wiendahl et al., 2009). Diese Montageform hängt stark vom Produktgewicht, Produktabmessung und der Produktionsrate ab. Bei der reinen Baustellenmontage bleibt sowohl Montageobjekt als auch Montagepersonal stationär (Lotter & Wiendahl, 2013). In Abbildung 1 ist die industrielle Baustellenmontage schematisch dargestellt.

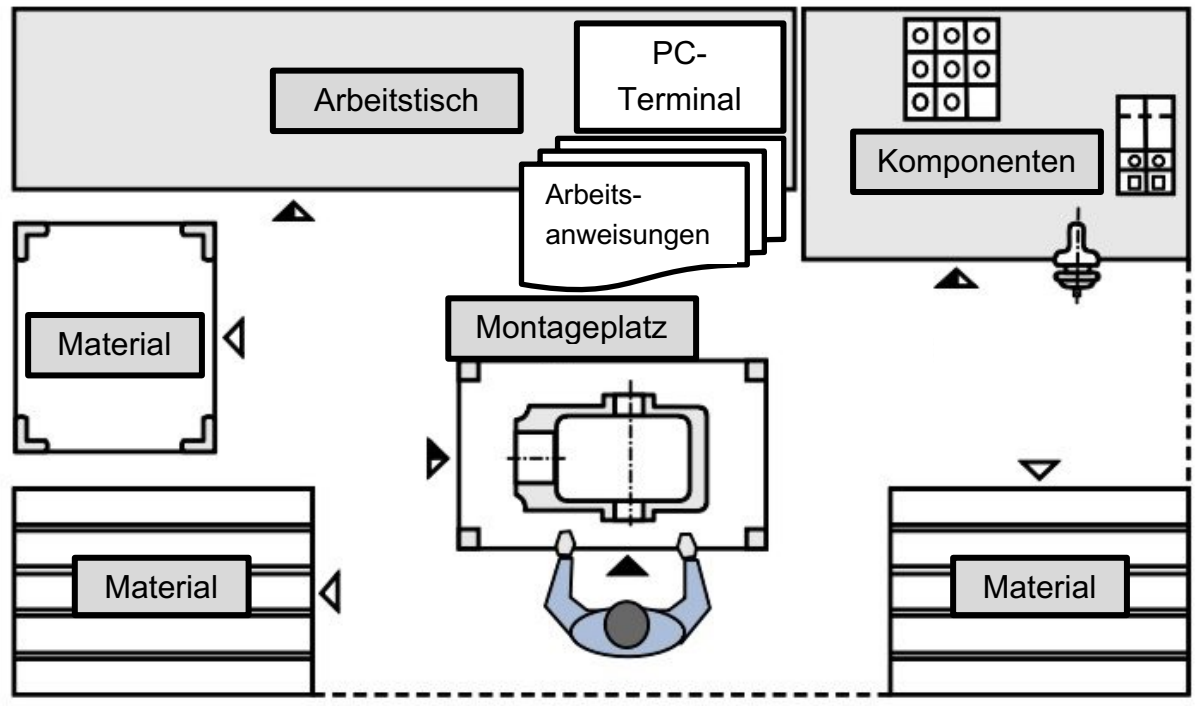


Abbildung 1: Schematische Darstellung der industriellen Baustellenmontage (Lotter & Wiendahl, 2013) zitiert nach Rupprecht et al., 2020.

Bei dieser Montageform bleibt das Werkstück fest an seiner Position (Baustelle) und Arbeitskräfte und Produktionsmittel bewegen sich zur Baustelle. In den meisten Fällen kommt bei der Baustellenmontage eine Einzelstückmontage zum Einsatz. Der Platzbedarf der Baustellenmontage ist in der Regel durch die große Einzelbauanzahl sowie der notwendigen Zugänglichkeit sehr hoch (Lotter & Wiendahl, 2013). Ein weiterer Aspekt, der eine wichtige Rolle spielt, ist die Informationsbereitstellung. Der aktuelle Stand der Technik bei der Informationsbereitstellung bei der industriellen Baustellenmontage sind Informationen in Papierformat sowie stationäre PC-Terminals. Ein entscheidender Nachteil sind dabei die langen Wegzeiten die zur Informationsbeschaffung zurückgelegt werden müssen (Rupprecht et al., 2020).

2.2 Mensch Maschine Interaktion (MMI)

Das Entwerfen und Gestalten von Gestensteuerungskonzepten fällt in das multidisziplinäre Forschungsgebiet der Mensch Maschine Interaktion. Die Mensch Maschine-Interaktion wird von Bendel (2019) folgendermaßen definiert:

> Die Mensch Maschine Interaktion (MMI), im Englischen *human-machine interaction* (HMI) genannt, behandelt die Interaktion zwischen Mensch und Maschine. Synonym oder mehr auf die Kommunikation bezogen spricht man auch von Mensch-Maschine-Kommunikation (eng. *human-machine communication*).

Dabei wird oft, wenn von „Maschine“ die Rede ist, ein Computer oder ein Gerät, welches Informations- und Kommunikationstechnologie (IKT) und Anwendungs-Informationssysteme (z. B Smartphones) enthält, gesprochen. Diese fallen in den Forschungsbereich Mensch Computer Interaktion (MCI), im Englischen *"human-computer interaction"* (HCI), welcher enge Beziehungen und erhebliche Überschneidungen mit der MMI aufweist (Bendel, 2019).

Dix et al. (2003) beschreiben die MCI als die Untersuchung der Art und Weise, wie die Computertechnologie die menschliche Arbeit und Aktivitäten beeinflusst. Der Begriff „Computertechnologie“ umfasst dabei die meisten Technologien, von offensichtlichen Computern mit Bildschirmen und Tastaturen bis hin zu Smartphones, Haushaltsgeräten, Navigationssystemen im Auto und eingebetteten Sensoren und Aktoren wie z. B. automatische Beleuchtung.

Die aktuelle Arbeitsdefinition des Begriffs Mensch Computer Interaktion (MCI) des ACM SIGCHI Kurrikulum lautet wie folgt (Hewett et al., 1992):

> *“Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. “*

Um die Mensch Computer Interaktion als Bereich (grob) charakterisieren zu können, geben Hewett et al. (1992) einige Aufgabengebiete der MCI vor:

- Die gemeinsame Ausführung von Aufgaben durch Mensch und Computer
- Die Struktur der Kommunikation zwischen Mensch und Computer
- Die menschliche Fähigkeit zur Nutzung von Maschinen, einschließlich der Lernfähigkeit von verschiedenen Schnittstellen (engl. *interfaces*)
- Algorithmen und Programmierung der Schnittstelle
- Der Prozess der Spezifikation, des Entwurfs und der Implementierung von Schnittstellen sowie Kompromisse beim Entwurf selbiger

Zusammenfassend lässt sich sagen, dass die Mensch Computer Interaktion wissenschaftliche, technische und Design-Aspekte enthält. Da MCI die Kommunikation zwischen Mensch und Computer untersucht, kann sie auf eine Vielzahl von Techniken zurückgreifen. Dazu zählen im Hinblick auf Computer Techniken der Computergrafik (engl. *Computer Vision*), Betriebssysteme, Programmiersprachen und Entwicklungsumgebungen. Auf den Menschen bezogen sind Kommunikationstheorie, Linguistik, Sozialwissenschaften, kognitive Psychologie und menschliche Leistung relevant. Auch ingenieurwissenschaftliche und Design-Aspekte sind von großer Bedeutung (Hewett et al., 1992). Die MCI ist multidisziplinär in ihrer Entstehung und interdisziplinär in ihrem Verhalten. Sie stützt sich auf menschliche Faktoren, Ergonomie, kognitive Psychologie, Verhaltenspsychologie, Systemtechnik und Informatik (Hartson, 1998). Das Ziel der MCI besteht darin, einen Dialog zwischen Mensch und Maschine zu erzeugen.

In der MMI sind verschiedenste Interaktionsformen möglich. Die vorliegende Diplomarbeit befasst sich mit der Interaktion mittels Gesten und um diese Form der Interaktion zu verstehen, muss zunächst die Frage: „Was ist eine Geste?“ geklärt werden. Der Begriff „Geste“ ist eine Entlehnung aus dem lateinischen Wort *gestus*, der „*Gebärde, Mienenspiel*“ bedeutet. Gesten werden oft als Zeichen der nonverbalen Kommunikation betrachtet. Die „Geste“ ist eine (Dudenredaktion, o. J.):

> „spontane oder bewusst eingesetzte Bewegung des Körpers, besonders der Hände und des Kopfes, die jemandes Worte begleitet oder ersetzt (und eine bestimmte innere Haltung ausdrückt)“

Eine weitere Bedeutung des Begriffs „Geste“ ist „Handlung oder Mitteilung, die etwas indirekt ausdrücken soll“ (Dudenredaktion, o. J.).Diese Bedeutung macht im Sinne einer Interaktionsform weniger Sinn und deshalb wird in der Folge, wenn von Gesten gesprochen wird, von einer Bewegung des Körpers ausgegangen.

Das Thema „Gesten“ ist nicht nur als mögliche Interaktionsform spannend. Eine Reihe von verschiedenen Fachgebieten beschäftigt sich mit Gesten im Alltag. Dazu zählen unter anderem die Kommunikationsforschung, die Psychologie, die Linguistik, die Semiotik und die Verhaltensforschung. Das Thema wird dabei von verschiedenen Standpunkten beleuchtet. Zu erwähnen ist, dass mit Gesten nicht nur Bewegungen mit den Händen gemeint sind, sondern auch Kopf- und Armbewegungen bezeichnet werden (Preim & Dachselt, 2015). Gesten und die Gestik sind generell eng mit Sprache verbunden. Sprache und Geste drücken denselben Ideeninhalt aus, allerdings auf ihre eigene Art und Weise (McNeill, 2008). Er beschreibt dieses Verhalten als co-expressiv. Dabei finden sie synchron, aber nicht redundant statt. Diese Synchronität impliziert, dass der Verstand im Moment des Sprechens (und des Gestikulierens) auf zwei Arten dasselbe tut.

Wenn nun die MCI betrachtet wird, kann man feststellen, dass nicht alle Arten von Gesten sinnvoll für eine Interaktion genutzt werden können. In der einschlägigen Literatur wird der Begriff „Geste“ relativ frei gebraucht, allerdings liegt der Fokus auf Gesten, welche ein Kommando auslösen oder mittels manipulierender Bewegungen bestimmte Parameter steuern (Preim & Dachselt, 2015).

Preim & Dachselt (2015) definieren daher den Begriff „Geste“ wie folgt:

> „Eine Geste ist die Bewegung von Fingern, Händen und Armen – oder auch weiterer Körperteile, wie Kopf, Augen und Lippen – aufgrund einer kommunikativen Absicht. Damit enthält die Bewegung als solche signifikante Informationen, die an den Computer übermittelt werden sollen.“

Im späteren Teil der Arbeit wird genauer auf die Taxometrie und Einteilung von Gesten eingegangen (siehe Kapitel 2.3). Der folgende Abschnitt beschäftigt sich mit der Rolle des Menschen im Kontext der Industrie 4.0. Im Speziellen wird auf neue Herausforderungen für die Arbeitswelt eingegangen, die sich durch neue Technologien ergeben. Daraus lassen sich Rückschlüsse und Erkenntnisse für die Gestaltung der MMI ziehen.

Die Gestaltung der Mensch-Maschine-Interaktion

In der industriellen Fertigung und in der Montage hat sich mit dem Einzug neuer Technologien im Zuge der vierten industriellen Revolution einiges verändert. So wurden beispielsweise früher und teilweise heute noch, manuelle Montagetätigkeiten über gedruckte Arbeitsanweisungen bereitgestellt. Hochauflösende Displays, Smart Devices oder Lösungen aus dem Bereich *Augmented Reality* könnten diese veraltete Informationsbereitstellung ersetzen. Trotz der zahlreichen technischen Entwicklungen im Kontext der Industrie 4.0 sind sich Produktionsexperten einig, dass auch in Zukunft weiterhin der Mensch eine zentrale Rolle in der Produktion von morgen spielen wird. Bis zur dritten industriellen Revolution war es das vorrangige Ziel den arbeitenden

Menschen im beruflichen Umfeld zu entlasten. Monotone und teils gefährliche Arbeiten sollten minimiert werden, wodurch entsprechende Sicherheitskonzepte sowie Gestaltungsrichtlinien entstanden, die es ermöglichten, die Aufgaben der Arbeiter in der Produktion zu erleichtern und gleichzeitig die Produktivität zu steigern. Vermehrt wurde auch auf Ausbildung und Schulung gesetzt. Industrie 4.0 bringt viele technische sowie organisatorische Entwicklungen mit sich. Eine Vielzahl von Sensoren und Aktoren halten Einzug in die Produktion. Auch die Anzahl an Daten (in der Produktion) ist deutlich gestiegen. Für den Faktor „Mensch“ in der Produktion bedeutet dies eine erhöhte kognitive Belastung. Die großen Datenmengen und Informationsfülle fordert Arbeits- sowie Langzeitgedächtnis des Arbeiters zunehmend (Reinhart, 2017).

Es kann davon ausgegangen werden, dass Veränderungen sowohl bei den Aufgaben als auch bei den Anforderungen an den Menschen im Rahmen der Industrie 4.0 eintreten (Vogel-Heuser et al., 2017). Dabei soll das Ziel allerdings nicht eine menschenleere Produktion sein. Einen Ansatz für ein solches Szenario gab es bereits in den 1980er-Jahren. Unter dem Namen *Computer Integrated Manufacturing* (CIM) versuchte man die Aufgabe des Menschen auf eine Überwachungs- und Beobachtungsfunktion zu reduzieren. Dies ist jedoch nicht das Ziel der Initiative Industrie 4.0. Laut Vogel-Heuser et al. (2017, S. 217) „soll der Mensch unter optimalem Einsatz seiner ureigenen Fähigkeiten in das cyber-physische Gefüge eingebunden werden.“ Das cyber-physische Gefüge stellt dabei die Beziehung zwischen einem cyber-physischem System (CPS) und dem Menschen auf abstrakte Weise dar. Eine intelligente Einbindung des Menschen bedeutet, seine Kompetenzen zu kennen und diese möglichst gut in das Gefüge aus Mensch und Technik einzubinden.

Das Hauptziel für den arbeitenden Menschen in der Industrie 4.0 besteht darin, seine Tätigkeit weiterhin so effizient und effektiv wie möglich zu verrichten. Um dies gewährleisten zu können, spielen ergonomisch gestaltete Anzeigen sowie Bedienkonzepte für Informationssysteme eine zunehmend wichtigere Rolle. Es ist anzunehmen, dass sich Aufgaben hinsichtlich ihrer Komplexität und Diversität ändern werden und die Interaktion mit der Maschine zu schwer nachvollziehbaren Ergebnissen führen kann. Speziell in diesen Fällen der MMI sollte der Arbeiter zu keinem Zeitpunkt überfordert werden. Wichtige Aspekte, die es bei der Gestaltung der MMI zu beachten gilt, sind die Systemtransparenz, Erlernbarkeit und Mode-Awareness. Letztere ermöglicht dem Arbeiter ein Bewusstsein für den aktuellen Zustand der Maschine. Es gilt eine Balance zu finden zwischen Überforderung des Arbeiters und einer Bevormundung durch die Maschine. Beide Extreme sind nicht erwünscht und im Optimalfall soll die Freude an der Anwendung der Maschine steigen (Reinhart, 2017).

Ein wichtiger Begriff in der Gestaltung der MMI ist jener der „Gebrauchstauglichkeit“ (eng. *usability*). Die DIN EN ISO 9241-11 (2018, S.9) definiert die

Gebrauchstauglichkeit als: „Ausmaß, in dem ein System, ein Produkt oder eine Dienstleistung durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.“ Wobei mit „effektiv“ die Genauigkeit und Vollständigkeit der Zielerreichung gemeint ist. Die Effizienz wird als eingesetzte Ressourcen im Verhältnis zu den Ergebnissen festgelegt. Die Zufriedenstellung ist das Ausmaß der Übereinstimmung der Reaktionen des Benutzers auf Benutzererfordernisse und der Benutzererwartungen.

Die Gestaltung der MMI ist ein komplexes und sehr vielfältiges Unterfangen. Dies liegt vor allem an der Informationsverarbeitung des Menschen, welche sich auf sehr vielfältige Art und Weise vollzieht. Daher spielen die Mechanismen der Wahrnehmung und der Kognition eine zentrale Rolle bei der Gestaltung einer Benutzerschnittstelle. Die Bedürfnisse des Nutzers sollen im Vordergrund stehen, um ein effizientes und einfach anwendbares System zu entwickeln. Die drei Elemente Benutzer, Aufgabe und Werkzeug (Maschine) müssen laut Butz & Krüger (2017) passgenau aufeinander abgestimmt sein. Die Schwierigkeit liegt dabei in der Anpassung der Stärken und Schwächen des Nutzers auf eine vorgegebene Aufgabe. Wie bereits zuvor erwähnt ist die zentrale Aufgabe bei der Gestaltung einer MMI eine effektivere und effizientere Aufgabenerledigung, welche dem Nutzer eine möglichst große subjektive Zufriedenheit gibt. Dabei ist es wichtig die Stärken und Schwächen sowohl des Menschen, als auch der Maschine zu kennen. Zu den Stärken des Menschen zählen das Deuten komplexer Signale (Sprache) und das Erkennen von komplexen räumlichen Konfigurationen. Die Stärken der Maschine oder des Computers sind unter anderem das Verarbeiten großer Datenmengen, das Lösen von algorithmisch formulierbaren Problemen und Ausführen wiederholbarer Operationen (Butz & Krüger, 2017).

Ein wichtiger Grundsatz bei der Gestaltung der MMI ist, die kognitive Belastung des Menschen nicht zu überfordern. Eine solche Überforderung tritt beispielsweise beim Lösen schwieriger Aufgaben unter Zeitdruck auf. Weitere Gründe können emotionale Belastung oder das Bewältigen mehrerer Aufgaben zur selben Zeit sein. Eine Faustregel, um die kognitive Belastung nicht zu überfordern, liefert Miller in seinem 1956 (manuell) erschienen Journalbeitrag „*The magical number seven, plus or minus two: Some limits on our capacity for processing information.*“ Sie besagt, dass das Arbeitsgedächtnis im Regelfall 7 (+/- 2) Gedächtnisinhalte zur selben Zeit behandeln kann. Diese pauschale Aussage aus den 1950er Jahren wird inzwischen als strittig angesehen. Unumstritten gilt allerdings, dass das Arbeitsgedächtnis, welches auch als Kurzzeitgedächtnis bezeichnet wird, limitiert ist (Butz & Krüger, 2017). Die kognitive Belastung des Mitarbeiters muss stets im Auge behalten werden. Hierzu ist ein gezieltes Management derselben durch intelligent geplante Informationsgestaltung notwendig. Einen Beitrag dazu können Werkerinformationssysteme (WIS) liefern. Ihre

Aufgabe besteht darin, den Menschen in der Produktion Informationen zu den verlangten Arbeitsaufgaben zu liefern (Reinhart, 2017). Eine Definition für den Begriff „Werkerinformationssystem“ liefert (Lang, 2007, S. 19): „Werkerinformationssysteme vereinen Teilbereiche des Informationsmanagements und des Wissensmanagements. Mit ihnen soll gewährleistet werden, dass Werkerinformation zum richtigen Zeitpunkt in der richtigen Form dem Mitarbeiter, assoziiert zu seiner aktuellen Tätigkeit, zur Verfügung gestellt wird“. Ein Synonym zum Begriff „Werkerinformationssysteme“ ist der Begriff „Mitarbeiterinformationssystem“, welcher allgemeiner gefasst ist. Um ein Werkerinformationssystem gestalten zu können, sind drei Gestaltungsfelder ausschlaggebend (Reinhart, 2017):

- Degree - Informationsgrad, welcher Art und Menge der bereitgestellten Informationen darstellt. Damit sind beispielsweise produktbezogene, prozessbezogene oder auftragsbezogene Informationen gemeint.
- Design - Informationspräsentation, befasst sich mit der Präsentation der Information, wobei hierbei meist der visuelle Sinn angesprochen wird. Die drei Präsentationsformen Text, Bild oder Video können beim visuellen Sinn unterschieden werden.
- Device - Informationsübertragung, beschäftigt sich mit der Frage „Womit wird die Informationspräsentation umgesetzt?“ Unterschieden wird hierbei zwischen mündlicher, papiergebundener oder digitaler Übermittlungsform (Neuschwinger, 2003). Bei der Informationsübertragung gilt es auch die Interaktionsmöglichkeiten bei der Gestaltung zu beachten.

Mit Blick auf die Zukunft bieten intelligent konzipierte Werkerinformationssysteme großes Potential für die Produktion und Montage von morgen. Abbildung 2 zeigt das Werkerinformationssystem „ActiveAssist“ von Bosch Rexroth.

Abbildung 2: WIS ActiveAssist; Bildquelle: (Bosch Rexroth AG, 2020)

Bei der Betrachtung der Informationsübertragung bringen Sprach- oder Gestensteuerung einige Vorteile mit sich. So kann eine Sprachsteuerung ein unterbrechungsfreies Arbeiten mit beiden Händen ermöglichen, wobei es bei einem solchen Steuerungskonzept den Umgebungslärm zu beachten gilt. Eine Interaktion mittels Gesten bietet Einsparungspotential hinsichtlich der Wegzeiten, die bei einer Informationsübermittlung mittels eines fixen Eingabegeräts anfallen würden (Reinhart, 2017).

2.3 Interaktion mit Gesten

In diesem Kapitel liegt der Fokus auf der Interaktion mittels Gesten. Konzepte der Gesteninteraktion aus dem Alltag können für die Gestaltung einer MMI übernommen werden. Zunächst wird in diesem Kapitel das Thema „Natürliche Interaktionen“ behandelt, welche unter anderem die Interaktion mittels Gesten beinhaltet. Schließlich werden systematische Einteilungen und Klassifikationssysteme erklärt, welche sich bei der Entwicklung von Gestensteuerungskonzepten etabliert haben.

Die MCI eignet sich für das Verständnis einer Gestensteuerung besser als das Forschungsfeld der MMI. Die beiden Bereiche überschneiden sich zwar teilweise, allerdings geht die MCI insbesondere in der Auslegung der Benutzerschnittstelle mehr in die Tiefe. Deshalb wird im folgenden Kapitel gezielter auf die MCI eingegangen.

2.3.1 Natural Gesture Interfaces

Viele Menschen sind mit Desktop-Computer und *Graphical User Interface* (GUI) im Alltag ständig in Kontakt, somit erscheint die Anwendung bereits logisch und intuitiv. Es gibt bei *Graphical User Interfaces* allerdings auch einige Limitationen und Schwächen. Die Trennung von Eingabegerät (Maus und Tastatur) und Ausgabegeräte (Monitor) und die indirekte Manipulation, die dadurch entsteht, lassen Spielraum für sensorisches und motorisches Potential des Menschen. Der Faktor 3-Dimensionaliät aus der realen Umgebung des Menschen lässt sich nicht optimal auf WIMP-Benutzerschnittstellen, welche zumeist auf 2D-Monitoren stattfinden übertragen. Das Akronym WIMP steht dabei für Windows, Icons, Menus und Pointers. *Virtuell Reality* und *3D-User Interfaces* versuchen diesen Mangel zu beheben. Auch wenn der Umgang mit Tastatur und Maus für viele zum alltäglichen Leben gehört, so bedarf es für die Nutzung meist doch einen gewissen Gewöhnungs- und Trainingsprozess. Für Computerneulinge ist diese Interaktionsform nicht die natürlichste und sie ist auch nicht für alle Anwendungsfälle optimal geeignet (Preim & Dachselt, 2015).

Bei näherer Betrachtung der Nachteile einer WIMP-Interaktionsform ist es nachvollziehbar, dass die Suche nach „natürlicheren“ Interaktionsformen seine Berechtigung hat. Ein Ansatzpunkt kann sicher die Einbindung der zwischenmenschlichen Interaktionsformen Sprache und Geste sein. Mit „natürlich“ ist

der Kontrast zu den bekannten Destop-Benutzerschnittstellen gemeint. Einige Aspekte des Begriffs „natürlich“ im Kontext der Mensch Computer Interaktion sind die Vertrautheit, Intuität und Einfachheit. Die natürliche Interaktion soll idealerweise kinderleicht zu erlernen und auf Erfahrungen aus der Realwelt basieren. Der Verzicht auf Zusatztechnik, wie etwa Maus und Tastatur, wird angestrebt. Darüber hinaus ist es wichtig die Bedienung so einfach wie möglich zu gestalten, der Benutzer sollte wenn möglich Spaß an der Interaktionsform haben (Preim & Dachselt, 2015).

Preim & Dachselt (2015, S. 472) definieren den Begriff *Natural User Interface (NUI)*:

> „Eine natürliche Benutzungsschnittstelle ist ein System zur Mensch Computer Interaktion, mit dem Benutzer mittels intuitiver und zumeist direkter Bedienhandlungen interagieren, die einen klaren Bezug zu natürlichem, realweltlichem menschlichen Alltagsverhalten aufweisen. Natürlich heißt dabei nicht angeboren, sondern bezieht sich auf dem Benutzer durch den Alltag vertraute und erlernte Handlungen bzw. auf solche Handlungen, die Benutzern im Moment der Interaktion als angemessen erscheinen.“

Eine Form dieser natürlichen Interaktionsweise ermöglichen gestische Benutzeroberflächen (engl. *Gestural User Interface*). Diese Interaktionsform ermöglicht die Steuerung eines Computers durch Handgesten aber auch andere Körperteile (Preim & Dachselt, 2015).

Vor- und Nachteile von Gestensteuerungen mittels Freihand werden in der folgenden Tabelle gegenübergestellt (Preim & Dachselt, 2015).

Tabelle 1: Vor- und Nachteile einer Gestensteuerung nach (Preim & Dachselt, 2015)

Vorteile:	**Nachteile:**
Intuitiver Gebrauch, vorausgesetzt die Gesten wurden sorgfältig designt.	**Sichtbarkeit und Entdeckbarkeit**: Anders wie bei GUIs sind visuelle Interaktionselemente bei gestischen UIs meist nicht gut sichtbar.
Technikfreiheit und Kontaktlosigkeit: Kein spezielles Equipment ist für die Nutzung notwendig.	**Erlernen und Behalten**: Nutzer müssen vor der Bedienung des Systems wissen, welche Gesten das System versteht.
Schnelligkeit: Bei gutem Design und einfacher, intuitiver und ergonomischer Gestaltung könne Gesten sehr schnell Befehle auslösen.	**Indirektes oder fehlendes Feedback:** Kein unmittelbares kinästhetisches Feedback.

Ermüdung: Bei längerem Einsatz und vielen Wiederholungen kann sehr schnell Ermüdung einsetzen, daher sollen Gesten schnell und präzise sein.

Limitierte Anzahl und Komplexität mittels Gestenalphabet abbildbar.

Durch intelligentes Design der gestischen Benutzeroberfläche lassen sich viele der Nachteile in den Griff bekommen. Das nächste Kapitel geht auf die verschiedenen Einteilungsmöglichkeiten unterschiedlicher Gesten ein.

2.3.2 Taxonomie und Klassifikation von Gesten

Dieser Abschnitt beschäftigt sich genauer mit der Klassifikation von Gesten und wird Anhaltspunkte liefern, welche Gesten für die Gestaltung einer Steuerung bzw. einer Schnittstelle sinnvoll sind. Das Hauptaugenmerk wird hierbei auf Handgesten gelegt. Gesteninteraktion beinhaltet allerdings auch Interaktionen mittels Kopfes, Fuß, Augen oder dem gesamten Körper. Handgesten sind die häufigste Anwendungsform in der Gesteninteraktion. Zunächst werden die verschiedenen Funktionen der Hand betrachtet (Cadoz, 1994):

- **Die epistemische Funktion** beschreibt die Hand als Wahrnehmungsorgan. Durch Berührung ist es möglich Informationen aus unserer physischen Umgebung wahrzunehmen.
- **Die Handlungs- und Betätigungsfunktion** beschreibt den unmittelbaren Energieaustausch mit den Händen, beispielsweise durch Verschiebung, Deformation oder Manipulation. Die Hand kann als Werkzeug zur Manipulation unserer Umwelt angesehen werden.
- **Die semiotische Funktion** beschreibt die Informationsabgabe an die Umwelt. Bei dieser Funktion geht es um Sprache und Informationsaustausch. Mit Ihrer Hilfe werden Botschaften mittels Gesten übermittelt.

In der Mensch Computer Interaktion wird vorrangig die semiotische Funktion der Hand genutzt. Diese kann weiter unterteilt werden in Kommandogesten, Gesten der Gebärdensprache und sprachbegleitende Gesten. Unter Kommandogesten werden einfache, explizite Gesten, die in dialogorientierten Kontexten angewendet werden, verstanden. Sie sind unabhängig vom Sprachkanal. Bei Gesten der Gebärdensprache ist eine fest vorgelegte Syntax vorgegeben. Die Gebärdensprache kann als eigenständige Sprache betrachtet werden (Marcel, 2002).

In der MCI wird oft auf die Einteilung der sprachbegleitenden Gesten von (McNeill, 1992) zurückgegriffen. Er unterscheidet folgende Kategorien:

- **Emblematische Gesten**, in der Literatur auch oft als symbolische Gesten bezeichnet, sind Gesten, die eine klare sprachliche Definition haben. Ein Beispiel dafür ist die „Daumen hoch"- Geste, welche ein Zustimmen signalisiert.
- **Ikonische Gesten** bilden die Realität in übertragener Form ab. Das Nachahmen von Handlungen oder die Darstellung von räumlichen Beziehungen sind Beispiele für ikonische Gesten.
- **Metaphorische Gesten** stellen eine Metapher dar und sollen ein Konzept zum Ausdruck bringen.
- **Deiktische Gesten** können auch als Zeigegesten bezeichnet werden. Beispielsweise eine Bewegung der Hand oder des Kopfes auf eine referenzierte Person, Ort oder Gegenstand.
- **Rhythmisierende Gesten** werden im Englischen als *beat gestures* bezeichnet und ergänzen den Rhythmus einer Aussage oder unterstreichen Teile der Sprachinformation. Sie enthalten keine semantische Information.

Bei der Gesteninteraktion zwischen Menschen und Computer sind vor allem Gesten relevant, welche eine bewusste Kommunikationsabsicht signalisieren. Dazu zählen das Zeigen, Ablehnen, Entfernen, Nehmen oder Zeichnen (Marcel, 2002). Die DIN EN ISO 9241-960:2017 spricht im Interaktionszusammenhang auch von „Gestenbefehl", welcher ein Kommando an das System darstellt. Gruppierungen von Gesten und deren Abbildung auf Gestenbefehle werden als Gestensatz bezeichnet (DIN EN ISO 9241-960, 2017). Als Synonym findet man in der wissenschaftlichen Literatur auch oft die Begriffe Gestenalphabet und Gestenvokabular. Preim & Dachselt (2015)) definieren den Begriff Gestenalphabet folgendermaßen:

> „Ein Gestenalphabet (auch Gestenvokabular, eng. *Gesture Set*) ist eine Menge von konfliktfreien, unterscheidbaren und erlernbaren Gesten, mit denen eine Menge an Funktionen für eine bestimmte Applikation, Anwendungsdomäne oder anwendungsübergreifend auf Betriebssystemebene aktiviert werden kann. Während ein Gestenvokabular oft nur eine kleine, aber zusammenhängende Menge von speziellen Gesten beschreibt, umfasst ein Gestenalphabet eine vollständige Menge an basalen Gesten, mit denen sich durch Kombination eine Vielzahl von komplexen Gesten durchführen und Funktionen aktivieren lassen."

Beim Entwurf eines Gestenalphabets oder auch Gesten-Set genannt, stehen eine Vielzahl möglicher Gesten zur Verfügung. Bei der Konzeption einer Benutzerschnittstelle muss stets im Auge behalten werden, welche Anwendungsdomäne und welcher Kontext vorliegen. Ein weiterer wichtiger Gesichtspunkt, welcher beachtet werden muss, ist die Komplexität der Benutzeroberfläche.

Beim Entwurf eines Gesten-Sets muss auch auf kulturelle Unterschiede eingegangen werden. Die Bedeutung einzelner Gesten kann länder- oder kulturspezifisch stark abweichen. So kann eine „harmlos“ erscheinende Geste in bestimmten Regionen der Erde eine Beleidigung bedeuten. Die Ausführung einer Geste kann von Person zu Person unterschiedlich ausfallen und von Belastung, Stimmung, Müdigkeit oder Arbeitskontext abhängen (Preim & Dachselt, 2015). Die DIN EN ISO 9241-960:2018-01 gibt bei Gesten in einem Gestensatz eine gewisse Konsistenz vor. Um sich Gesten besser in Erinnerung rufen zu können, sollten sie innerhalb des Gestensatzes die Gesten ähnlich, jedoch so gestaltet sein, dass eine Unterscheidung möglich ist. Wie bereits zuvor erwähnt, spielt die kognitive Belastung des Menschen hier eine Rolle. Um die Erinnerung an die verschiedenen Gesten zu verbessern empfiehlt sich eine ausführliche Dokumentation selbiger durchzuführen.

Zum Abschluss dieses Kapitels werden in Unterkapitel 2.3.3 einige Anwendungsbereiche für die Interaktion mit Gesten beschrieben.

2.3.3 Gesteninteraktion in der Praxis

Ein Meilenstein für gestische Interaktionsformen waren Sensoren, die bei der *Microsoft Kinect* zum Einsatz kommen, eine Hardware zur Steuerung der Spielekonsolen *Xbox 360* und *Xbox One* mittels Gesten. Markstart war im Jahr 2010. Sie ermöglichten dieser Interaktionsform den Zugang zur Massentauglichkeit zumindest im Gaming-Sektor. Mit den *Kinect for Windows Human Interface Guidelines* stellt Microsoft Gestaltungrichtlinien zur Gesteninteraktion für den von ihnen entwickelten MS Kinect Sensor zur Verfügung (Microsoft, 2020b). Der Sensor war bei Entwicklern und Forschern im Bereich der gestischen Interaktion beliebt. Microsoft entschied sich dennoch die Produktion der *Microsoft Kinect* mit 2017 zu stoppen (Der Standard, 2017). Als Nachfolger entwickelte Microsoft 2019 die Azure Kinect, welches ein Developer-Kit für räumliche Datenverarbeitung ist. Dabei handelt es sich um ein integriertes Kamerasystem, das als IoT-Sensor an die Azure-Cloud gebunden ist. Microsoft sieht die Einsatzgebiete in der Produktion, dem Einzelhandel, dem Gesundheitssektor sowie in Medienunternehmen(Microsoft, 2020a). In Abbildung 3 sind rechts die MS Kinect und die Azure Kinect Dk dargestellt und links mögliche gestische Interaktionsfelder.

Abbildung 3: MS Kinect und Azure Kinect Dk inklusive Anwendungsfeld (Bildquelle: Microsoft)

Ein weiteres Einsatzgebiet für Gestensteuerungen ist das Automobil. Viele Automobilhersteller bieten ein berührungsfreies Öffnen der Gepäckraumklappe mittels Fußbewegung im Bereich des Fahrzeughecks an. Somit kann der Nutzer den Kofferraum öffnen auch wenn er keine Hand frei hat, weil er zum Beispiel schweres Gepäck transportiert. Auf Abbildung 4 wird links diese „Easy Open“-Funktion bei einem Volkswagen Tiguan verdeutlicht (Volkswagen AG, 2020). BMW stellte auf dem Mobile World Congress 2019 in Barcelona sein neues Interaktionskonzept für den Fahrzeuginnenraum vor. Unter dem Namen BMW Natural Interaction verbirgt sich ein kombiniertes System aus Sprachsteuerung, Gestensteuerung und Blickerkennung, welches ab 2021 serienmäßig zur Verfügung gestellt wird (BMW AG, 2019). Das erste Fahrzeug in dem serienmäßig Gestensteuerung kommerziell eingesetzt wurde ist der BMW 7er (vgl. Abbildung 4 rechts). BMW lieferte diese Interaktionsform unter dem Namen „BMW Gestensteuerung“ (BMW AG, 2015).

Abbildung 4:Links Easy Open Funktion beim VW Tiguan (Bildquelle: Volkswagen AG); rechts BMW Gestensteuerung im BMW 7er (Bildquelle: BMW AG)

Ein weiteres Anwendungsgebiet ist die Medizintechnik. Sauberkeit und Hygiene sind vor allem in Operationssälen oder sterilen Bereichen von größter Wichtigkeit. Eine berührungsfreie Interaktion kann hier von Vorteil sein. Die Firma TedCas bietet eine gestengesteuerte Benutzeroberfläche für den OP-Saal namens TedCube an (vgl. Abbildung 5). Chirurgen wird somit der Zugriff auf digitale Informationen ermöglicht, ohne sich desinfizieren oder um Hilfe bitten zu müssen (TedCas, 2020).

Abbildung 5: Gestengesteuerte Benutzeroberfläche im OP (Bildquelle: TedCas)

Nachdem einige Einsatzgebiete der Interaktion mittels Gesten genannt wurden, wird im nächsten Kapitel auf das Themengebiet *Machine Learning* eingegangen.

2.4 Machine Learning

Zur Erkennung von Handgesten mit Hilfe von maschinellen Lernmethoden müssen zunächst einige Grundbegriffe eingeführt werden. Im folgenden Kapitel wird zunächst

erläutert, um was es sich bei *Machine Learning* (ML) handelt und wie dieser Bereich eingeteilt werden kann.

Das sog. „Maschinelle Lernen“ oder im Englischen *Machine Learing* ist ein Teilgebiet der künstlichen Intelligenz (KI). Das Feld der KI oder im Englischen *Artificial Intelligence* (AI) versucht laut Russel & Norvig (2002) nicht nur intelligente Gebilde zu verstehen, sondern selbige aufzubauen. Es existiert eine Vielzahl verschiedener Definition des Begriffes AI. Für einen ausführlichen Überblick der verschiedenen AI-Definitionen wird auf (Russell & Norvig, 2002) verwiesen. *Deep Learning* (DL) in der deutschsprachigen Literatur auch als „Tiefes Lernen“ bezeichnet ist wiederum eine Unterkategorie von ML. Eine Einteilung der Gebiete wird in Abbildung 6 veranschaulicht.

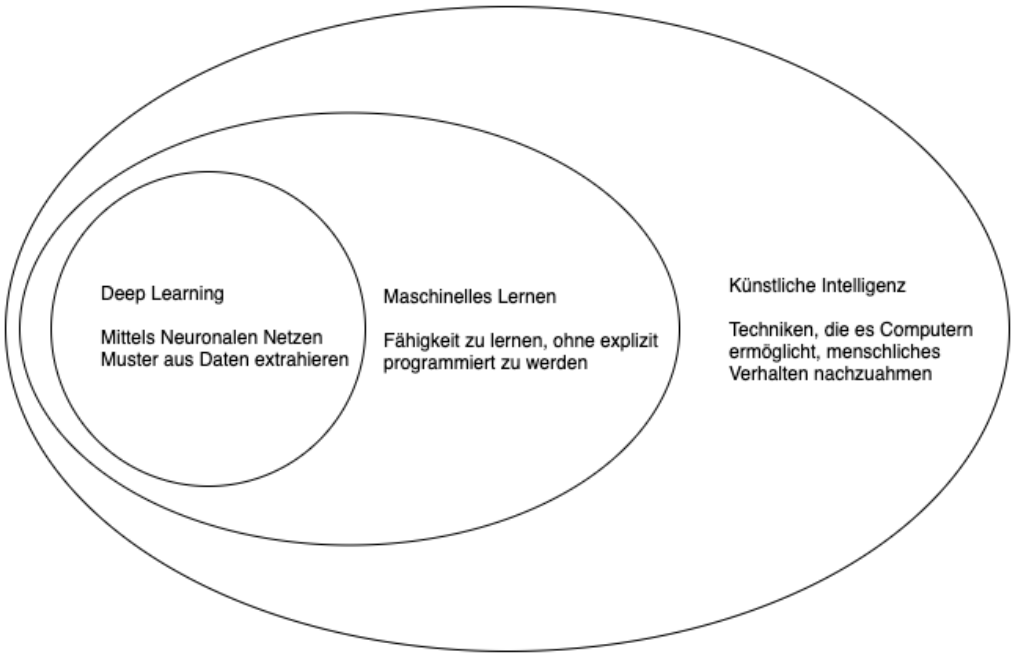


Abbildung 6:Venn Diagramm AI, ML, DL in Anlehnung an Goodfellow et al. (2016, S9)

Maschinelles Lernen ist die Wissenschaft, Computer so zu programmieren, dass sie aus Daten lernen können (Géron, 2019, S. 2). Eine etwas generellere Definition liefert Arthur Samuel (1959) der das maschinelle Lernen als das Studiengebiet, das Computern die Fähigkeit zum Lernen verleiht, ohne explizit programmiert zu werden, bezeichnet. Was dabei mit „Lernen“ gemeint ist definiert Mitchell (1997) wie folgt:

> *“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”*

Unter der Aufgabe (engl. *task*) versteht man die Art und Weise wie das ML-System ein Beispiel verarbeiten soll. Dabei stellt ein Beispiel eine Kollektion von Merkmalen (engl.

features) dar, die quantitativ von einem Objekt oder einem Ereignis gemessen wurden, welche das maschinelle Lernsystem verarbeiten soll. Mathematisch wird ein Beispiel typischerweise mit einem Vektor $x \in \mathbb{R}^n$ dargestellt, wobei jedes x_i ein Merkmal darstellt. Eine häufig vorkommende Aufgabe im ML ist die Klassifikation, dabei soll das ML-System angeben, zu welcher Kategorie *k* ein Beispiel gehört. Bei Merkmalen eines Bildes beispielsweise, handelt es sich normalerweise um die Werte der Pixel (Goodfellow et al., 2016, S. 99). Die Beispiele, die das System zum Lernen verwendet, werden Trainingsdaten genannt. Die Trainingsdaten enthalten das Wissen, das der Lernalgorithmus extrahieren und lernen soll. Bei der Auswahl der Trainingsdaten, aber auch der Testdaten (im folgenden Absatz erklärt) ist darauf zu achten, dass es sich um eine für die zu lernende Aufgabe repräsentative Stichprobe handelt (Ertel, 2016, S. 195). Die Gesamtheit der Trainingsdaten wird auch Trainingssatz (engl. *training set*) genannt. Zusammenfassend lässt sich ein Datensatz als eine Sammlung von Beispielen beschreiben, die wiederum Sammlungen von Merkmalen sind (Goodfellow et al., 2016, S. 106).

Um die Fähigkeit eines Algorithmus in Bezug auf die Aufgabe messen zu können, bedarf es eines Leistungsmaßes (engl. *performance measure*). Das Leistungsmaß ist normalerweise spezifisch auf die ausgeführte Aufgabe angepasst und stellt einen quantitativen Maßstab der Aufgabenerfüllung dar. Bei einer Klassifikation wird oft als Leistungsmaß die Genauigkeit (engl. *accuracy*) verwendet. Die Genauigkeit gibt an welcher Prozentanteil der Beispiele korrekt zugeordnet werden konnten. Üblicherweise ist das Ziel eine hohe Genauigkeit auf Beispielen zu erzielen, die das ML-System zuvor nicht gesehen hat. Solche Beispiele werden Testdaten genannt und getrennt vom Trainingsset für die Evaluation der Leistungskennzahl verwendet (Goodfellow et al., 2016, S. 103 f.).

In Bezug auf die Definition von (Mitchell, 1997) lassen sich ML-Systeme in zwei grobe Kategorien einteilen, was ihre *experience* betrifft. Zum einen in überwachtes Lernen (engl. *supervised learning*) und zum anderen in unüberwachtes Lernen (engl. *unsupervised learning*). Der Begriff „überwachtes Lernen“ ist der Tatsache geschuldet, dass dem ML-System das Label oder das Ziel (eng. *target*) von einem Lehrer vorgegeben wird. Dem ML-System wird somit gezeigt, was zu tun ist (Goodfellow et al., 2016, S. 104 f.). Beim überwachten Lernen enthält der Trainingsdatensatz, der dem Algorithmus zugeführt wird, bereits die gewünschten Lösungen, sogenannte Labels. Ein Beispiel dafür ist etwa ein Spam Filter, bei dem Nachrichten in mit zwei möglichen Labels, „Spam“ oder „Kein Spam“, eingeteilt werden können. Beim unbeaufsichtigten Lernen (engl. *unsupervised learning*) sind die Trainingsdaten, mit keinem Label versehen (unbeschriftet). Das ML-System versucht hier selbständig Muster zu erkennen. Es existieren weitere Kategorien zur Einteilung in Bezug auf die Überwachung des Trainings. In der vorliegenden Diplomarbeit werden diese zwei

Kategorien, nämlich *Semi-Supervised Learning* und *Reinforcement Learning,* nicht weiter berücksichtigt.

Bei ML-Systemen lässt sich eine weitere Typenunterscheidung durchführen. Es gibt ML-Algorithmen die durch den Vergleich neuer Datenpunkte mit bekannten Datenpunkten arbeiten (*Instance-base Learning*) oder stattdessen durch die Erkennung von Mustern in den Trainingsdaten und die Erstellung eines Vorhersagemodells (*Model-based Learning*). Beim *Instance-based Learning* lernt das System die Beispiele auswendig und verallgemeinert sie dann auf neue Fälle, indem es ein Ähnlichkeitsmaß verwendet, um sie mit den gelernten Beispielen (oder einer Teilmenge davon) zu vergleichen. Eine andere Möglichkeit, aus einer Reihe von Beispielen zu verallgemeinern, besteht darin, ein Modell dieser Beispiele zu erstellen und dieses Modell dann zu verwenden, um Vorhersagen zu treffen. Diese Art des Lernens wird *Model-based Learning* genannt (Géron, 2019, S. 17 f.).

Die zentrale Herausforderung in ML ist die Leistung bei der Vorhersage von neuen, zuvor ungesehenen Beispielen. Die Fähigkeit, bei zuvor unbeobachteten Eingaben gute Leistungen zu erbringen, wird als Generalisierung bezeichnet. Die Hauptfaktoren sind dabei die Auswahl eines geeigneten Algorithmus und das Training auf einem geeigneten Datensatz. Voraussetzung dabei ist eine ausreichende Menge an Daten. Zudem müssen diese Trainingsdaten repräsentativ für neue Fälle sein. Ein weiterer wichtiger Aspekt ist die Qualität der Daten (Géron, 2019, S. 23 ff.). Typischerweise kann beim Trainieren eines ML-Systems ein Leistungsmaß für die Performance auf den Trainingsbeispielen erhoben werden. Dieser wird Trainingsfehler (eng. *training error*) genannt. Das Ziel muss sein, diesen möglichst gering zu halten. Was ML von einem Optimierungsproblem unterscheidet, ist der sogenannte Generalisierungsfehler (engl. *generalization error*) auch Testfehler (engl. *test error*) genannt. Der Testfehler ist definiert als der erwartete Wert des Fehlers auf ein neues ungesehenes Beispiel. Dabei sind Beispiele gemeint, die in dieser Form im Anwendungsgebiet des ML Systems erwartet werden. Ausgewertet wird der Testfehler typischerweise mit Hilfe eines Testdatensatzes, mit Beispielen die getrennt vom Trainingsdatensatz gesammelt wurden (Goodfellow et al., 2016, S. 110). Es ist wichtig, die Generalisierungsfähigkeit des ML Systems auf dem System unbekannten Testdaten zu testen. Der Test- Vorgang ist unerlässlich, andernfalls würde jedes ML-System, das gerade Trainingsdaten gespeichert hat, allein durch den Abruf der gespeicherten Daten optimal zu funktionieren scheinen (Ertel, 2016, S. 196).

Bei der Auswahl sowie der Einstellungen eines ML Systems können verschiedene Probleme auftreten. So kann beispielsweise der gewählte Algorithmus eine gute Leistung auf Trainingsdaten erreichen, aber er lässt sich nicht optimal auf Testdaten generalisieren. Dieses Problem wird im ML *Overfitting* genannt. Dabei passt sich der Algorithmus zu sehr den Trainingsdaten an. Das Gegenteil von *Overfitting* wird als

Underfitting bezeichnet. In diesem Fall ist das Modell zu simpel ist und die Trainingsdaten werden nicht optimal abgebildet(Goodfellow et al., 2016).

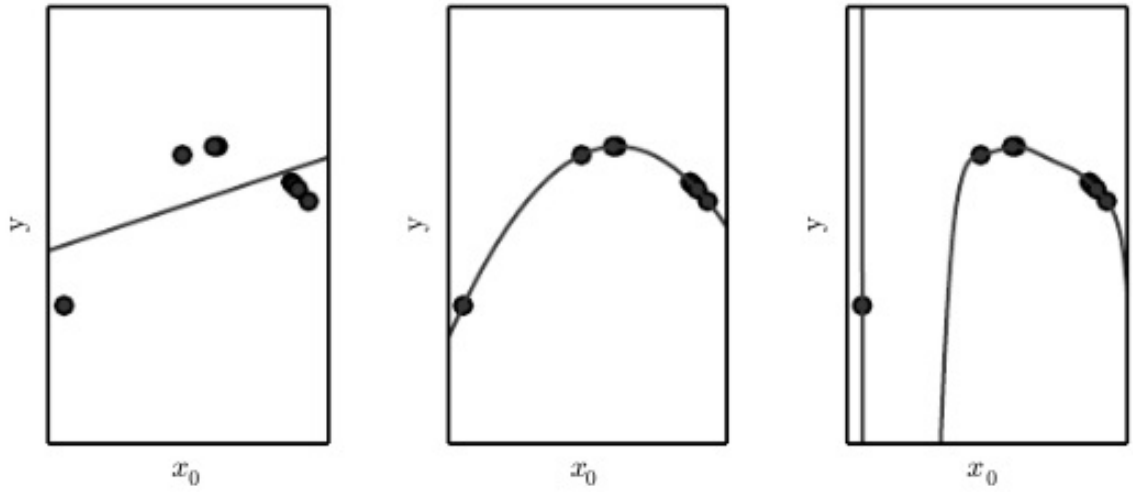


Abbildung 7: Underfitting (links), Overfitting (rechts) und ausreichende Generalisierung (Mitte) (Bildquelle: Goodfellow et al., 2016, S. 113)

Abbildung 7 zeigt drei synthetisch erzeugte Modelle. Auf der linken Seite wird versucht die Daten mit einer linearen Funktion abzubilden. Die Krümmung die in den Daten vorhanden ist, wird nicht erfasst. Dies ist ein Beispiel für *Underfitting*. Auf der rechten Seite wird ein Fall von *Overfitting* dargestellt. Die Funktion passt sich zu sehr den Datenpunkten an und generalisiert nicht gut. Die mittlere Abbildung stellt die Verteilung der Datenpunkte in ausreichender Form dar. Es gilt sowohl *Overfitting* als auch *Underfitting* zu vermeiden. Hier muss die richtige Balance zwischen Anpassung an die Trainingsdaten und Einfachheit des Modells sichergestellt werden.

Ein wichtiger Begriff im ML ist jener der Regularisierung (engl. *regularization*). Hiermit wird jede Änderung des Lernalgorithmus bezeichnet, der den Generalisierungsfehler reduzieren soll, ohne dabei den Trainingsfehler zu verändern. Um das Verhalten des Lernalgorithmus einstellen zu können, existieren sog. „Hyperparameter“. Diese Hyperparameter werden nicht selbständig vom ML-System angepasst, sondern im Vorfeld des Trainings eingestellt. Zu keinem Zeitpunkt dürfen allerdings die Testdaten zur Einstellung der Hyperparameter verwendet werden. Zu diesem Zweck wird eine weitere Sammlung von Datenpunkten eingeführt, das sog. Validierungsset (engl. *validation set*). Das Validierungsset ist meist eine Teilmenge der Trainingsdaten. Damit kann der Generalisierungsfehler während oder nach dem Training abgeschätzt werden. Anschließend können die Hyperparameter entsprechend angepasst werden (Goodfellow et al., 2016, S. 120 f.).

Die Grundbegriffe, Einteilungsmöglichkeiten und Herausforderungen beim ML wurden in diesem Kapitel erläutert. Bei der Handgestenerkennung handelt es sich um ein

Klassifizierungsaufgabe, sowie eine Aufgabe der maschinellen Bildverarbeitung. Dafür wird oft eine Art des DL, *Convolutional Neural Networks* genannt, verwendet. Im nächsten Kapitel wird darauf genauer eingegangen.

2.5 Deep Learning

Goodfellow et. al (2016, S. 167) bezeichnen DL als einen sehr leistungsfähigen Rahmen für überwachte Lernmethoden. Es handelt sich um eine Art des maschinellen Lernens, welches die Welt als eine verschachtelte Hierarchie von Konzepten darstellt (Goodfellow et al., 2016, S. 7). Viele Ideen und Prinzipien auf dem Gebiet der neuronalen Netze stammen aus der Hirnforschung und dem verwandten Fachgebiet der Neurowissenschaften (Ertel, 2016, S. 3).

Neuronale Netze sind Netzwerke von Nervenzellen in den Gehirnen von Menschen und Tieren. Das menschliche Gehirn hat ungefähr 100 Milliarden Nervenzellen. Künstliche neuronale Netze (KNN) sind dem biologischen Vorbild des Menschen oder eines Tieres nachempfunden. Die ersten Durchbrüche im Bereich der künstlichen neuronalen Netze sind auf McCulloch und Pitts (1943) zurückzuführen. In ihrem Artikel "*A logical calculus of the ideas immanent in nervous activity*" stellten sie Neuronen als Schaltelement des Gehirns mit Hilfe eines mathematischen Modells vor (McCulloch & Pitts, 1943). Die McCulloch-Pitts-Zelle liefert den Grundstein der mathematischen Darstellung eines Neurons in einem KNN (Ertel, 2016, S. 265).

Das Perceptron ist eine der simpelsten Formen von KNNs und wurde 1957 von Frank Rosenblatt erfunden. Die Basis bildet ein künstliches Neuron, welches auch *threshold logic* unit (TLU) genannt wird. Ein anschauliches Modell hinter dieser einfachsten Form eines Neuronalen Netzes ist in Abbildung 9 dargestellt. Die Signale x_1, x_2,..., x_m entsprechen neuronalen Signalen, die als Inputsignale bezeichnet werden. Die Inputsignale werden mit den zugehörigen Gewichten w_1, w_2,..., w_m multipliziert. Darüber hinaus wird im Allgemeinen ein sogenannter Bias-Term w_0 hinzugefügt. Die Eingabe x_0 wird dabei auf 1 gesetzt (Géron, 2019, S. 285). Das Netzwerk, inklusive Bias ist in Abbildung 8 abgebildet. In der Literatur wird der *Bias* oftmals auch mit *b* angegeben. Die Ausgabe eines *Perceptron* wird wie folgt berechnet. Zunächst wird das Punktprodukt zwischen den Eingaben und dem Gewichtsvektor gebildet und der Bias addiert. Anschließend wird auf diesen Term die nichtlineare Aktivierungsfunktion angewendet.

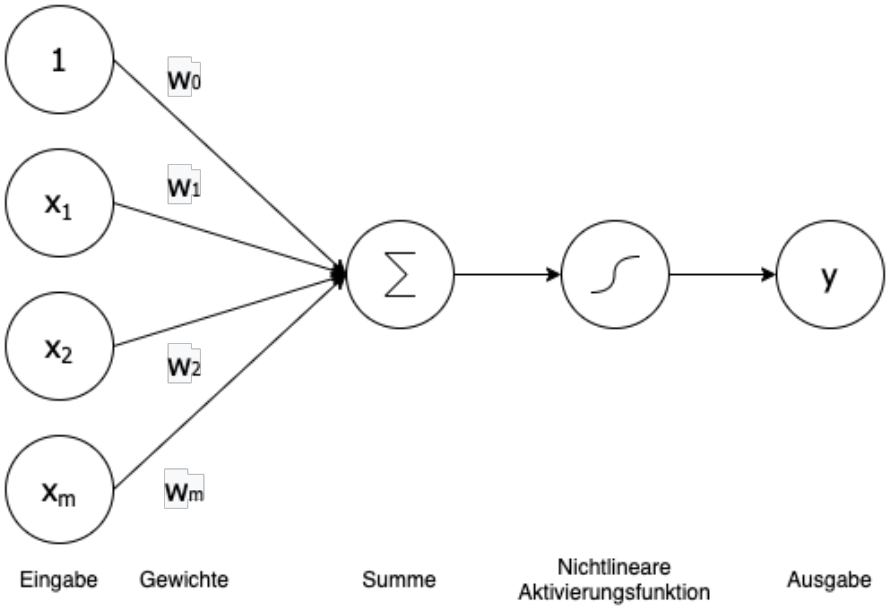


Abbildung 8: Das Perceptron in Anlehnung an Amini & Soleimany, 2020

Formel 1 beschreibt den mathematischen Zusammenhang des Perceptrons.

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i\, w_i\right)$$

Formel 1: Berechnung des Ausgangs eines Perceptrons (Amini & Soleimany, 2020)

In Formel 2 wird Formel 1 in Vektorschreibweise gebracht.

$$\hat{y} = g(w_0 + X^T \mathrm{W})$$

Formel 2: Berechnung des Ausgangs eines Perceptrons in Vektorschreibweise (Amini & Soleimany, 2020)

Wobei gilt:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ und } W = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Die Aktivierungsfunktion *g* bringt die Nicht-Linearität in die Funktion. Diese Nicht-Linearität bildet die Voraussetzung dafür, dass der Trainingsalgorithmus, der auf der Kettenregel basiert funktioniert. Die *Rectified Linear Unit* (ReLU)-Funktion ist gegenwärtig einer der am häufigsten verwendeten Aktivierungsfunktionen. Weitere Möglichkeiten sind die Sigmoidfunktion (1/ (1+exp(-x)) oder der Tangens hyperbolicus (Lecun et al., 2015, S. 458).

Das Ziel des tiefen Lernens besteht in einem Lernprozess des Neurons. Hierbei sollen Muster, z. B. Merkmalsvektoren (engl. *feature vectors*) von Beispielen, die ihm gezeigt werden, in geeigneter Weise erkannt werden. Der Prozess beginnt mit einer zufälligen Initialisierung der Gewichte. Anschließend beginnt der eigentliche Lernprozess. Die Inputsignale x_i werden aus den Trainingsset zugeführt und anschließend mit den Gewichten multipliziert. Mit Hilfe der Aktivierungsfunktion *g* wird das Output-Signal erzeugt. Dieses Output-Signal ist die Reaktion des Neurons auf das Muster, das ihm gezeigt wurde. Die Hauptidee des Lernprozesses besteht darin die Gewichte des Neurons in Abhängigkeit von seiner Reaktion auf das gezeigte Muster zu verändern. Dies geschieht entsprechend der gewählten Lernmethode (z.B. überwachtes Lernen). So wird im letzten Schritt des Hauptzyklus der Gewichtsvektor w des Neurons modifiziert. Anschließend wird das nächste Muster des Trainingssatzes eingegeben. Nachdem dem Neuron alle Merkmalsvektoren des Trainingssets gezeigt wurden, kann eine Entscheidung darüber getroffen werden, ob das Neuron gelernt hat, Muster zu erkennen. Wenn es auf Muster angemessen reagiert und somit die Gewichte richtig eingestellt sind, hat das Neuron gelernt. Falls dies nicht der Fall ist beginnt der Trainingsprozess von Neuem (vgl. Flasiński, 2016, S. 161).

Ein Perceptron besteht aus einer einzelnen Schicht von TLUs, wobei jedes TLU mit allen Eingängen verbunden ist**.** Wenn alle Neuronen in einer Schicht mit jedem Neuron in der vorherigen Schicht verbunden sind, wird die Schicht als dichte Schicht (*engl. Dense Layer*) oder als *fully connected layer* bezeichnet (Géron, 2019, S. 285). Wenn mehrere Perceptrons in einem Netzwerk zusammenfügt werden, nennt man dies ein Multilayer Perceptron (MLP). Dies ist eine simple Form eines KNNs. Die erste Schicht eines KNNs wird als Eingabeschicht (engl. *Input layer*) bezeichnet, die letzte Schicht als Ausgabeschicht (engl. *output layer*). Die Zwischenschichten werden als verborgene Schichten (engl. *hidden layers*) bezeichnet. Wenn eine KNN mehrere verborgenen Schichten enthält, wird das Netz als tiefes neuronales Netz bezeichnet (vgl. Géron, 2019, S. 289). Je höher die Anzahl der verborgenen Schichten desto „tiefer“ ist das Netzwerk. Daher stammt der Begriff „Tiefe Neuronale Netze“ (engl. *Deep Neural Networks*). Abbildung 9 zeigt schematisch ein tiefes neuronales Netz. Der Begriff „neuronal“ stammt aus den Neurowissenschaften. Dabei ist es keineswegs das Ziel von KNNs das Gehirn in perfekter Art und Weise nachzuahmen (Goodfellow et al., 2016, S. 169).

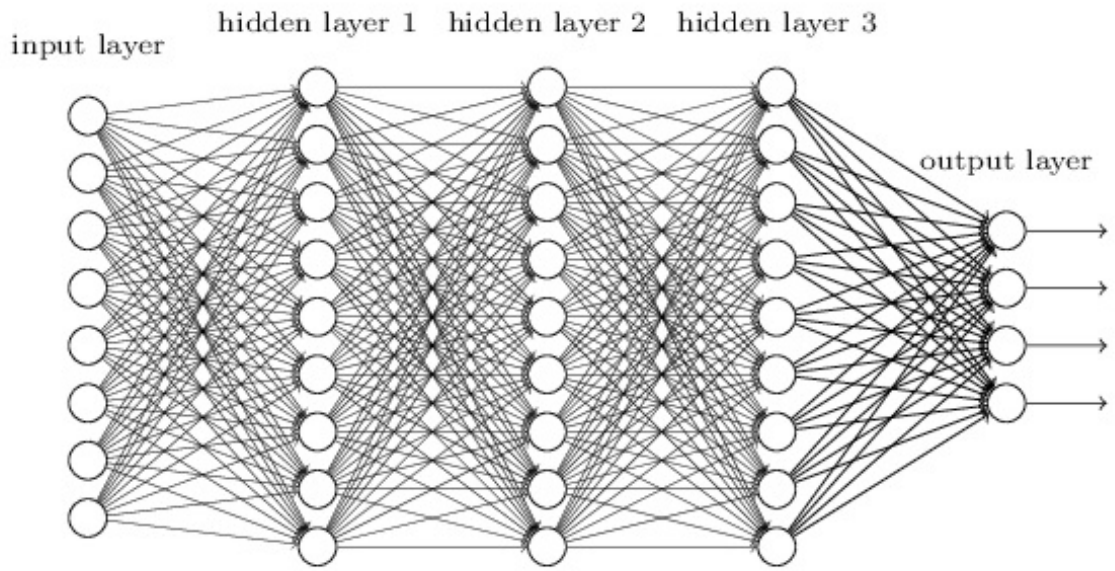


Abbildung 9: Tiefes Neuronales Netz (Nielsen, 2015, S. 169)

Um komplexere Neuronale Netze trainieren zu können, wird der Backpropagation-Trainings-Algorithmus verwendet. Die Idee geht auf mehrere Forschungsteams aus den 1970ern und 1980ern zurück. Dabei ist der Backpropagation-Algorithmus nichts anders als eine Anwendung der Kettenregel. 1986 wurde dieser Algorithmus erstmals unter der Bezeichnung „Backpropagation“ in der Publikation von Rumelhart, Hinton und Williams beschrieben (Rumelhart et al., 1985). Der Backpropagation- Algorithmus macht zunächst eine Vorhersage (engl. *forward pass*) für jede Trainingsinstanz und misst den resultierenden Fehler. Anschließend geht er jede Schicht in umgekehrter Reihenfolge durch, um den Fehlerbeitrag jeder Verbindung zu messen (engl. *reverse-pass*). Das Ziel besteht darin, zu ermitteln wie viel jede Verbindung zum Fehler beiträgt. Dies erfolgt analytisch durch Anwendung der Kettenregel. Schließlich optimiert der Algorithmus die Gewichte, um den Fehler zu reduzieren (*Gradient Descent step*). Damit dieses Verfahren funktioniert, müssen die Gewichte der verborgenen Schichten zufällig initialisiert werden. Ein weiterer wichtiger Punkt ist die nicht-lineare Aktivierungsfunktion (Géron, 2019, S. 291). Mittels DL ist es möglich komplizierte Strukturen in großen Datensätzen zu entdecken, indem es den Backpropagation-Algorithmus verwendet. Dieser gibt an, wie ein ML-System interne Parameter ändern soll, die zur Berechnung der Darstellung in jeder Schicht aus der Darstellung in der vorherigen Schicht verwendet werden (Lecun et al., 2015, S. 431).

Ein Bild kommt beispielsweise in Form eines Arrays von Pixelwerten, und die gelernten Merkmale in der ersten Darstellungsebene repräsentieren typischerweise das Vorhandensein oder Fehlen von Kanten an bestimmten Ausrichtungen und Stellen im Bild. Die zweite verborgene Schicht erkennt Motive typischerweise aufgrund der Anordnungen von Kanten, unabhängig von kleinen Abweichungen in den

Kantenpositionen. Die dritte Ebene kann Motive zu größeren Kombinationen zusammensetzen, die Teilen bekannter Objekte entsprechen. Nachfolgende Ebenen können Objekte schließlich als Kombinationen dieser Teile erkennen (Lecun et al., 2015).

Wenn beispielsweise ein ML-System aufgebaut werden soll, das auf Bildern Autos, Personen und Tiere klassifizieren soll, kann man dazu das sog. überwachte Lernen nutzen. Zunächst wird ein großer Datensatz mit Bildern von Autos, Menschen und Tieren gesammelt und mit der zutreffenden Kategorie (engl. *label*) versehen. Während des Trainings wird dem ML-System ein Bild gezeigt. Daraufhin erzeugt es eine Ausgabe in Form eines Vektors mit einer Bewertung, je eine pro Kategorie. Anschließend wird eine Funktion berechnet, die den Fehler (oder die Diskrepanz) zwischen der Ausgabebewertung und der gewünschten Bewertung misst. Diese Funktion wird als *loss function* oder *error function* bezeichnet. Das ML-System modifiziert anschließend die Gewichte, um diesen Fehler zu minimieren. Um den Gewichtsvektor richtig einzustellen, berechnet der Lernalgorithmus einen Gradientenvektor, der für jedes Gewicht angibt, um welchen Betrag sich der Fehler vergrößert oder verkleinert, wenn das Gewicht um einen geringen Betrag erhöhen würde. Der Gewichtsvektor wird dann in entgegengesetzter Richtung zum Gradientenvektor eingestellt. Oft wird dazu ein Algorithmus angewendet, der als *stochsatic gradient descent* (SGD) bezeichnet wird. Dabei werden dem ML-System einige Beispiele des Trainingssets gezeigt, anschließend der Ausgang bzw. der Fehler berechnet und schließlich der Durchschnitt für die ermittelten Gradienten berechnet. Im Anschluss können die Gewichte entsprechend angepasst werden (Lecun et al., 2015).

Anstelle des SGD Algorithmus sind in den letzten Jahren weiter Optimierungs-Algorithmen entstanden. Der bekannteste darunter ist der von Kingma und Ba (2014) entwickelte *Adam*- Algorithmus. Der Name Adam stammt von *adaptive moment estimation*. Zwei Vorteile gegenüber SGD sind die einfache Implementierung und die rechnerische Effizienz. Häufig wird der *Adam* - Algorithmus verwendet, um gute Ergebnisse möglichst schnell zu erzielen. Empirische Ergebnisse zeigen, dass Adam in der Praxis gut funktioniert und im Vergleich zu anderen stochastischen Optimierungsmethoden günstig ist (Kingma & Ba, 2014).

Eine Deep-Learning-Architektur ist ein mehrschichtiger Stapel von einfachen Modulen oder Einheiten, von denen die meisten einem Lernprozess unterliegen. Das Wort „Architektur“ bezieht sich auf die Gesamtstruktur des Netzwerks, z. B darauf, wie viele Einheiten es haben soll und wie diese Einheiten miteinander verbunden werden sollen. Zwei grundlegende Entscheidungen sind dabei die Tiefe des Netzwerkes sowie die Breite der Layer. Die ideale Architektur des Netzwerkes wird dabei oft experimentell mit Hilfe des Validierungssets ermittelt (Goodfellow et al., 2016, S. 197 f.). Bei einer

Tiefe von 5 bis 20 Schichten kann ein System extrem komplizierte Funktionen seiner Eingaben implementieren, die gleichzeitig auf kleinste Details reagieren (Lecun et al., 2015, S. 438).

Convolutional Neural Networks (CNNs)

Convolutional Networks (Le Cun et al., 1989), auch bekannt als *Convolutional Neural Networks*, *ConvNets* oder CNNs, sind eine spezielle Art von neuronalen Netzwerken zur Datenverarbeitung, die eine bekannte, gitterartige Geometrie haben. Der Name „*Convolutional neural network*" stammt von der mathematischen Operation „*Convolution*". Im Deutschen wird dieser Operator „Faltungsoperator" genannt. Dabei nutzen CNNs diese spezielle lineare Operation anstelle der normalen Matrix Multiplikation in zumindest einem ihrer Layer (Goodfellow et al., 2016, S. 330). Ein Beispiel für eine gitterartige Geometrie oder auch Topologie genannt, ist etwa ein Farbbild, das aus drei zweidimensionalen Arrays besteht. Diese drei 2D-Arrays enthalten die Pixel-Intensitäten der Farbkanäle (RGB).

Es gibt drei zentrale Ideen, die hinter einem CNN stecken: spärliche Interaktionen, gemeinsame Nutzung von Parametern und äquivariante Darstellungen. Bei einem typischen KNN interagiert jede Input-Einheit mit jeder Output-Einheit. Mit spärlicher Interaktion (eng. *sparse interactions* oder oft auch als *sparse weights* bezeichnet) wird beschrieben, dass ein Kernel, welcher auf den Input angewendet wird, viel kleiner ist als der Input. Dies führt zu einer geringeren benötigten Rechenleistung und einer gleichzeitigen Effektivitätssteigerung. Veranschaulicht wird dies in Abbildung 10, wobei unten der Input x_3 mit allen Ausgängen verbunden ist und im oberen Teil der Abbildung eine spärliche Interaktion des Inputs x_3 dargestellt wird. Unter der gemeinsamen Nutzung von Parametern (engl. *parameter sharing*) versteht man die Nutzung desselben Parameters für mehrere Funktionen in einem Model. Ein Synonym für *parameter sharing* sind sog. gebundene Gewichte (engl. *tied weights*) weil der Wert des auf einen Eingang angewandten Gewichts an den Wert eines, an einer anderen Stelle angewandten Gewichts gebunden ist (Goodfellow et al., 2016, S. 335 ff.). Dies ermöglicht es mit geringere Rechenleistung Features an verschiedenen Lokalitäten auf einem Bild zu erkennen. So können beispielsweise im ersten *Hidden Layer* Ecken und Kanten unabhängig von ihrer Position auf dem Bild erkannt werden.

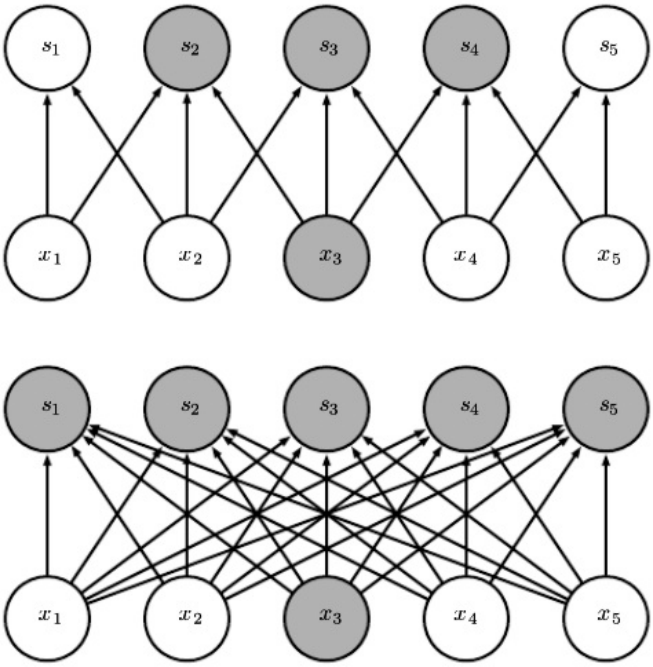


Abbildung 10: Sparse interactions (oben), Fully connected layer (unten) (Goodfellow et al., 2016, S. 337)

Äquivariante Darstellungen bezeichnen eine Eigenschaft, die es CNNs ermöglicht, Objekte zu erkennen, bei denen es nicht erforderlich ist, dass dieses Objekt eine feste Position im Bild einnimmt. Es kann beispielsweise rotiert sein. Einfacher gesagt bedeutet äquivariant Folgendes: Wenn der Input sich verändert, verändert sich gleichzeitig der Output (vgl. Goodfellow et al., 2016, S. 338).

Die Architektur eines CNNs ist in eine Reihe von verschiedenen Phasen gegliedert. Die erste Phase besteht meist aus einer Kombination aus *Convolutional Layer* und *Pooling Layers,* welche im Folgenden näher erläutert werden.

Der *Convolution Operator* wird in Formel 3 mathematisch dargestellt, wobei die mathematische Faltung mit einem Sternchen dargestellt wird. In der CNN-Terminologie wird das erste Argument (die Funktion *x*) oft als Input bezeichnet und das zweite Argument (hier die Funktion *w*) als Kernel. Der Output wird als *Feature Map* bezeichnet (Goodfellow et al., 2016).

$$s(t) = (x * w)(t)$$

Formel 3: *Convolutional* Operator

Die diskrete Faltung kann als Multiplikation mit einer Matrix betrachtet werden. Ein anschauliches Beispiel dafür, wie die Faltungsoperation in CNNs angewendet wird, ist in Abbildung 11 dargestellt. Die Pfeile symbolisieren, wie der Output mittels Kernels berechnet wird.

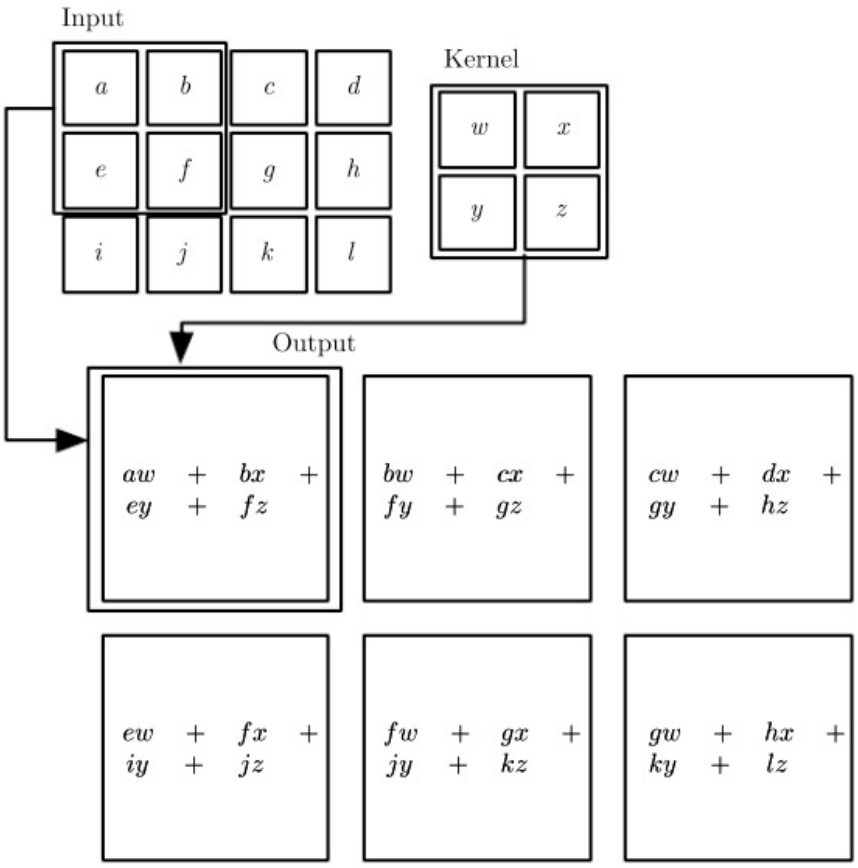


Abbildung 11: Anwendung des Convolutional Operators (Goodfellow et al. 2016, S. 334)

In Abbildung 12 wird der *Convolution Operator* in dreidimensionaler Form dargestellt. In der Abbildung wird auch dargestellt, wie der Kernel zum nächsten Pixel wandert. Diese Schrittweite wird auch *stride* genannt. Dabei wird zwischen horizontalen *stride* und vertikalem *stride* unterschieden.

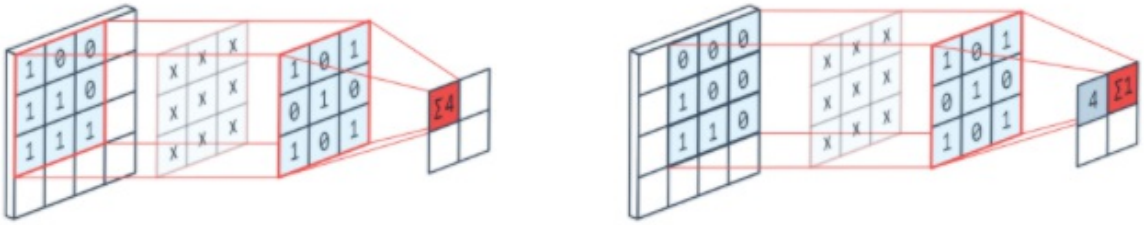


Abbildung 12: 2D Convolutional Operator; Bildquelle:(Peltarion, 2020a)

Ein weiterer wichtiger Baustein ist der sog. *Pooling Operator*. Der *Pooling-Operator* ersetzt die Ausgabe des Netzes an einem bestimmten Ort durch eine zusammenfassende Statistik der nahegelegenen Ausgänge. Dabei existieren verschiedene Arten von Pooling Operatoren. Eine viel verwendete Pooling Operation

ist das sog. *Max pooling*, bei dem der maximale Output innerhalb eines rechteckigen Fensters bestimmt wird. Eine grafische Darstellung dieses Operators ist in Abbildung 13 abgebildet.

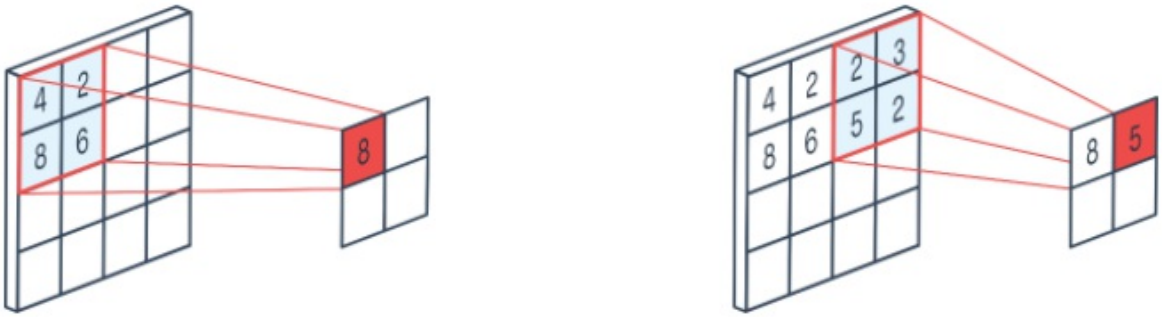


Abbildung 13: 2D Max Pooling Operator ; Bildquelle: (Peltarion, 2020b)

Typischerweise wird der Pooling Operator in Kombination mit einem Convolution Layer und einer Nicht-Linearität (z.B ReLU) angewendet. Die Reihenfolge, dargestellt in einer einfache Layer-Terminologie ist in Abbildung 14 zu sehen.

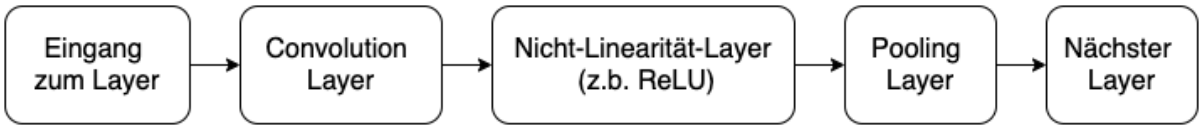


Abbildung 14:Typische Layer-Anordnung in CNNs in Anlehnung an (Goodfellow et al., 2016, S. 341)

Convolutional Layers haben in der Regel mehrere Kernels (ein Synonym, welches in der Literatur auch häufig verwendet wird, ist „Filter“). Ein *Convolutional Layer* wendet gleichzeitig mehrere trainierbare Kernels, die verschiedene *Features* (z.B. horizontale Kanten oder vertikale Kanten) erkennen können, an (Géron, 2019, S. 451).

Eine typische CNN-Architektur reiht die Kombination aus *Convolution Layers* gefolgt von Nicht-Linearitätsschichten und *Pooling Layer* aneinander. Dabei wird normalerweise das Bild kleiner je tiefer es sich im Netz befindet. Gegen Ende des Netzes werden *Fully connected Layers* (gefolgt von Nicht-Linearitätsschichten) hinzugefügt um am Ausgang des Netzes mittels einer Softmax-Funktion eine Voraussage über die Klassenzugehörigkeit treffen zu können (Géron, 2019, S. 460 f.). Eine solche Softmax-Funktion oder Softmax-Einheit wird verwendet, um für jede mögliche Klasse eine Wahrscheinlichkeit zu erhalten. Sie wird oft als Output eines Klassifizierungsproblems verwendet und kann im übertragenen Sinn als eine Art Schalter gesehen werden (Goodfellow et al., 2016, S. 184). Abbildung 15 zeigt eine typische Architektur eines CNNs, wobei hier zwei Phasen unterschieden werden. Zum

einen in eine sog. *Feature Learning*-Phase und zum anderen in eine Klassifikationsphase am Ende des Netzwerkes.

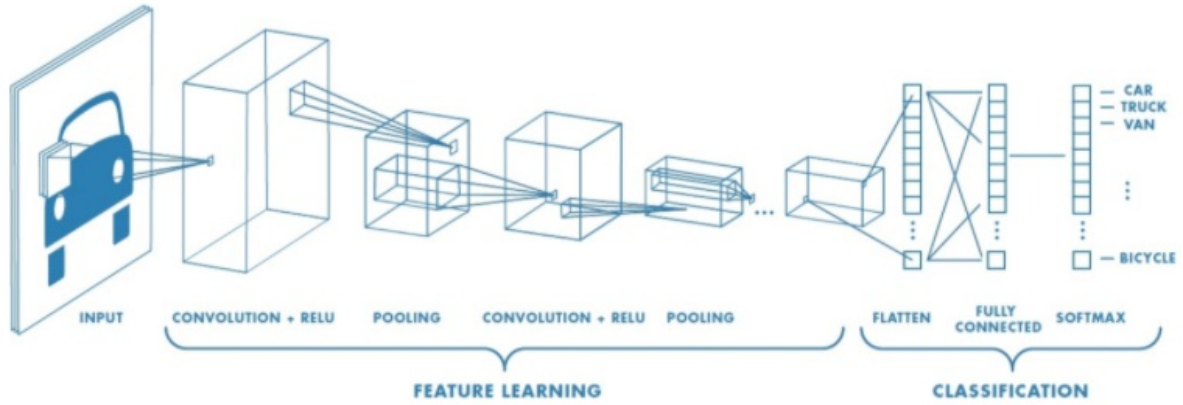


Abbildung 15: CNN-Architektur; Bildquelle: (The MathWorks, 2020)

Die erste Anwendung eines CNNs geht auf LeCun et al. (1989) zurück. Ziel dabei war es, mittels eines CNNs (trainiert mittels Backpropagation) handgeschriebene Ziffern zu erkennen. Den großen Durchbruch schafften CNNs durch die Arbeit von Krizhevsky et al. (2012). Es gelang ein für die damalige Zeit erstaunliches Ergebnis bei der Klassifizierung auf dem anspruchsvollen *ImageNet*-Datensatz (Deng et al., 2009). *ImageNet* gilt als das Benchmark-Datenset in der Bildklassifizierung und wird zum Training und zum Test neuer CNN Architekturen verwendet. In den darauffolgenden Jahren wurden weitere Fortschritte in Bezug auf CNN-Architekturen erreicht. Einige bekannte CNN-Architekturen sind *AlexNet* (Krizhevsky et al., 2012), *VGGNet* (Simonyan & Zisserman, 2014), *GoogLeNet* (Szegedy et al., 2015) und *ResNet* (He et al., 2016).

Im folgenden Kapitel werden Methoden zur Eingrenzung und zur Bestimmung geeigneten Datensatzes vorgestellt.

3 Verwendete Methoden und Konzepte

In Kapitel 3 werden die verwendeten Methoden und Konzepte beschrieben, welche im Anschluss am praktischen Use-Case angewendet werden. Dabei wird zunächst in Kapitel 3.1 die Nutzwertanalyse zur Auswahl eines geeigneten Datensatzes erläutert. Anschließend wird das Konzept bei einer Supervised ML-Klassifikationsaufgabe erläutert. Abschließend wird in Kapitel 3.3 auf das Konzept des Transfer Learning eingegangen.

3.1 Nutzwertanalyse

Eine Nutzwertanalyse kommt bei der Analyse von Entscheidungssituation zum Einsatz. Diese systematische Planungsmethodik kann als Hilfsmittel bei der Auswahl zwischen verschiedenen Lösungsalternativen eingesetzt werden. Besonders bei Entscheidung-Situationen mit einer Vielzahl von entscheidungsrelevanten Zielkriterien ist eine Nutzwertanalyse sinnvoll (Zangemeister, 2014).

Bei der Nutzwertanalyse wird das Gesamtproblem in Teilprobleme zerlegt. Dieses Prinzip wird auch Fragmentierung genannt und läuft im Menschen oft unbewusst ab. Das Ziel ist die Vereinfachung von komplexen, umfangreichen Problemen. Durch diese Zerlegung des Problems wird der emotionalen Bindung bzw. Präferenz für eine gewünschte Lösung vorgebeugt. Eine Nutzwertanalyse ist immer sinnvoll, wenn mindestens einer der folgenden Faktoren zutreffen (Kühnapfel, 2014):

- Es gibt eine Vielzahl von Bewertungskriterien
- Die Bewertungskriterien sind unterschiedlich (quantitativ, qualitativ)
- Es gibt keine eindeutige Reihenfolge der Bewertungskriterien
- Es sind mehrere Personen mit individuellen Meinungen und Erfahrungen im Entscheidungsprozess involviert
- Eine Entscheidung auf Basis von Erfahrungen oder unternehmerischen Instinkten ist nicht möglich
- Die Entscheidungsfindung soll dokumentiert werden

Der Ablauf einer Nutzwertanalyse kann in folgenden drei Phasen unterteilt werden (Herbig, 2016):

- **Konzeptionsphase:** In dieser Phase werden die Ziele formuliert. Daraufhin werden für die Ziele Bewertungskriterien ermittelt und Alternativen gesammelt.

Die Zielkriterien sollten in dieser Phase kritisch überdenkt und evtl. angepasst werden.

- **Bewertungsphase:** In dieser Phase werden die Bewertungskriterien gewichtet und untereinander verglichen. Ziel dabei ist es den unterschiedlichen Stellenwert der Bewertungskriterien zu ermitteln.

- **Ergebnisphase:** In dieser Phase wird eine Rangordnung auf Basis des Teilnutzwerts und des Gesamtnutzwerts der verfügbaren Alternativen erstellt. Der Gesamtnutzwert quantifiziert den Zielerreichungsgrad einer Alternative in allen berücksichtigten Bewertungskriterien. Wohingegen der Teilnutzwert den Zielerreichungsgrad einer Alternative hinsichtlich eines Bewertungskriteriums quantifiziert.

In Kapitel 5.5 wird eine Nutzwertanalyse zur Auswahl eines geeigneten Datensatzes durchgeführt.

3.2 Supervised Machine Learning

Um die vorliegende Problemstellung zu definieren muss zunächst geklärt werden um welche Art von Aufgabe es sich handelt. Da die richtigen Antworten zu den Beispielen bei den vorliegenden Datensätzen bereits als Label mitgeliefert werden kann als Lernstil das überwachte Lernen (engl. supervised learning) verwendet werden. Zudem kann die vorliegende Lernaufgabe als Multiklassen-Klassifikation eingestuft werden. Das bedeutet dem ML-Algorithmus werden Bilder von Gesten gezeigt, die er daraufhin richtig zuordnen muss. Dabei stehen mehr als zwei Klassen zur Auswahl.

Im Folgenden wird zunächst der generelle Ablauf bei einem überwachten ML-Projekt vorgestellt. Anschließend werden Leistungsmaße zur Beurteilung von Klassifikationsmodellen erläutert. Der generelle Ablauf eines bildbasierten Klassifikationsproblems mit Hilfe von CNNs folgt dem Standard-Workflow zur Lösung von realen Problemen mit Hilfe von *Machine Learning*. Abbildung 16 stellt den strukturierten Prozess einer *Machine Learning Pipeline* dar.

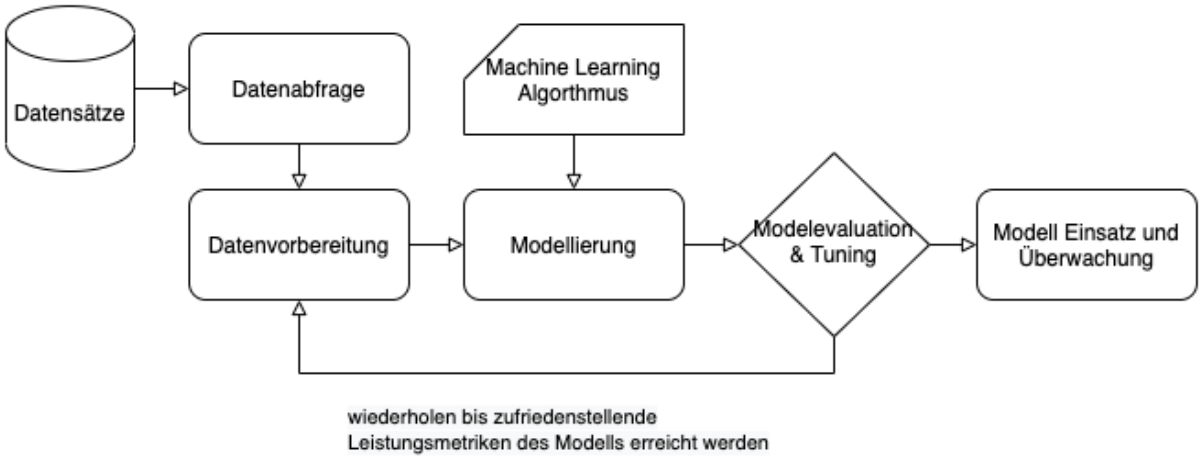


Abbildung 16: ML-Workflow in Anlehnung an Sarkar et al., 2018, S. 53

Eine Pipeline für maschinelles Lernen besteht hauptsächlich aus folgenden Elementen (Sarkar et al., 2018):

- Abfrage und Extrahieren von Daten
- Datenvorbereitung
- Modellierung
- Modelevaluation und Auswertung
- Einsatz und Überwachung

Dieser Prozess wird vom CRISP-DM-Modell (Wirth & Hipp, 2000) abgeleitet und ist diesem ähnlich. Die beschriebene ML-Pipeline setzt einige Schritte, die im CRISP-DM enthalten sind, voraus. Dazu zählen etwa Business-Verständnis, ML-Technikauswahl sowie Bewertung von Risiken, Annahmen und Einschränkungen. Für Details zum CRISP-DM Modell wird auf Wirth und Hipp (2000) verwiesen. Im nächsten Abschnitt wird auf Leistungsmaße und Möglichkeiten der Modell-Evaluierung von Multiklassen-Klassifikationsmodellen eingegangen.

Performance-Metriken zur Gestenklassifikation

Im vorliegenden Abschnitt wird auf Leistungsmaße und Möglichkeiten des Evaluierens von Klassifizierungsmodellen eingegangen. Die Wahrheitsmatrix, in der englischsprachigen Literatur als *Confusion Matrix* bezeichnet, ist ein häufig verwendetes Visualisierungswerkzeug. Im Folgenden wird die Matrix und die von ihr abgeleiteten Metriken beschrieben. Dazu wird der englische Begriff *Confusion Matrix* verwendet. Zunächst wird die *Confusion Matrix* für ein binäres Klassifikationsproblem gezeigt. Die enthaltenen Prinzipien können auf Multiklassen-Klassifikationsaufgaben übertragen werden.

Ein Klassifikationsmodell (engl. *Classifier*) ist eine Zuordnung von Instanzen zu vorhergesagten Klassen. In Abbildung 17 ist eine *Confusion Matrix* für eine binäre Klassifizierungsaufgabe dargestellt.

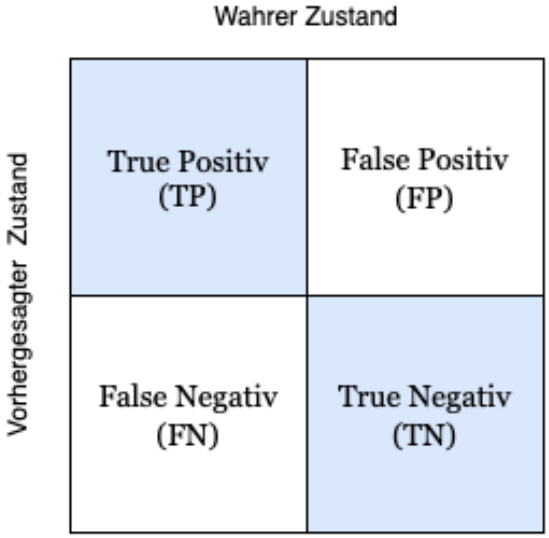


Abbildung 17:Confusion Matrix in Anlehnung an Fawcett, 2006, S. 862

Bei der Zuordnung einer Instanz in einem binären Klassifikationsmodell gibt es vier mögliche Ergebnisse. Wenn die Instanz positiv ist und positiv vorhergesagt wird, wird sie als richtig positiv (engl. *True positive*) klassifiziert. Wird die positive Instanz als negativ klassifiziert ist sie als falsch negativ (engl. *False negative*) einzuordnen. Wenn die Instanz negativ ist und als negativ klassifiziert wird, wird sie als richtig negativ (engl. *True negative*) erfasst. Eine negative Instanz die positiv vorhergesagt wird, ist als falsch positiv (engl. *False positive)* einzustufen. Die *Confusion Matrix* ist die Grundlage für die Leistungskennzahlen für Klassifikationsmodelle. Zunächst kann die Genauigkeit (engl. *accuracy*) eines Klassifikationsmodells wie folgt berechnet werden.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Formel 4: Formel für Accuracy (Fawcett, 2006, S. 862)

Zwei der wichtigsten Metriken sind *Precision* und *Recall*. Die Precision gibt den Anteil an richtig vorhergesagten positiven Ergebnissen (TP) bezogen auf die Gesamtheit aller als positiv vorhergesagten Ergebnisse an. Recall wird auch Hit-Rate genannt und gibt den Anteil der korrekt als positiv klassifizierten Ergebnisse (TP) bezogen auf die Gesamtheit der tatsächlich positiven Ergebnisse an (Fawcett, 2006). Die Formeln 5 und Formel 6 beschreiben die Zusammenhänge.

$$precision = \frac{TP}{TP + FP}$$

Formel 5: Formel für *Precision* (Fawcett, 2006, S. 862)

$$recall = \frac{TP}{TP + FN}$$

Formel 6:Formel für *Recall* (Fawcett, 2006, S. 862)

Die Kombination aus *Precision* und *Recall* ergibt den F_1 Score. Konkret wird das harmonische Mittel zwischen *Precison* und *Recall* gebildet (vgl. Formel 7).

$$F_1 = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

Formel 7: Formel für F_1 (Géron, 2019, S. 92)

Beim vorliegenden Klassifizierungsproblem handelt es sich um eine Multiklassen-Klassifikationsaufgabe (engl. *Multi-class classification task*) bei dem ein Input genau einer Klasse zugeordnet werden soll. Der Input wird in eine der sich nicht überlappenden Klassen eingeordnet. Um die Qualität eines Multiklassen-Klassifikationsmodell zu bestimmen, gibt es zwei Möglichkeiten. Zum einen kann für jede Klasse K_i *Accuracy*, *Precision*, *Recall* und F_1-Score berechnet werden, um anschließend den Mittelwert zu bilden. Dieses Vorgehen wird als *Macro-Averaging* bezeichnet. Bei der zweiten Methode, dem sog. *Micro-Averaging* werden die kumulierten Werte für TP_i, FP_i, FN_i, TN_i berechnet und anschließend die Performance-Metriken berechnet (Sokolova & Lapalme, 2009).

3.3 Transfer Learning

Um sich für spezielle Bildklassifikationsaufgaben, die Erkenntnisse vortrainierte CNN-Architekturen zunutze zu machen, gibt es den Ansatz des sog. „Transferlernens“ (engl. *transfer learning*). Die Kernidee von *Transfer Learning* liegt darin, dass bereits erlangtes Wissen beim Training eines Modells auf einem großen Datensatz auf eine neue Klassifikationsaufgabe zu übertragen (Sarkar et al., 2018, S. 505). *Transfer learning* bezieht sich somit auf die Situation, in der das, was in einer Umgebung gelernt wurde, zur Verbesserung der Generalisierung in einer anderen Umgebung genutzt wird. Man beginnt mit einem bereits trainierten Netz und verwendet es für eine neue

Aufgabe. Vor allem die sogenannten *Low Level* Features in den ersten *Layern* eines CNNs wie etwa Ecken, Kanten und geometrische Formen sind für viele Bildklassifikationsaufgaben sinnvoll. *CNNs* welche für Transfer Learning genutzt werden, sind häufig auf dem *ImageNet* Datensatzes vortrainiert (Goodfellow et al., 2016, S. 536). Die *Transfer Learning*-Technik benötigt signifikant weniger Bildmaterial für das effektive Training des Modells, was ein großen Vorteil darstellt (Géron, 2019). Für *Transfer Learning* gibt es eine Vielzahl unterschiedlicher Strategien und Einsatzszenarien. Interessierte Leser werden auf die Publikation „A Survey of Transfer Learning“ verwiesen (Weiss et al., 2016).

Wie bereits im Kapitel 2.5 erwähnt, sind in den letzten Jahren eine Vielzahl verschiedener CNN-Modell-Architekturen entstanden. Die Auswahl einer Modell-Architektur für praktische Anwendungen gestaltet sich oft schwierig und hängt von mehreren Faktoren ab. Zum einen ist die Genauigkeit ein wichtiger Indikator für die Auswahl einer CNN-Architektur. Mit Hilfe von Benchmark-Datensätzen wie *ImageNet* können CNN-Modell-Architekturen miteinander verglichen werden. CNN-Architekturen, deren Ziele es ist, die *ImageNet-Challange* zu gewinnen, nehmen eine höher Rechenkapazität im Gegenzug für höhere Genauigkeit in Kauf. Sie werden meist auf Supercomputern mit sehr hoher Rechenleistung trainiert. Für praktische Anwendungen ist allerdings nicht nur die Genauigkeit entscheidend, auch Speicherauslastung und Geschwindigkeit spielen eine entscheidende Rolle (Huang et al., 2017). Der Trend geht in Richtung Leichtbau-Netzwerkstrukturen, ohne dabei größere Einbußen im Bereich der Genauigkeit hinnehmen zu müssen. So können CNNs auch für Hardware mit Ressourcenbeschränkungen zugänglich gemacht werden (Khan et al., 2020). Es gilt hier einen Kompromiss zu finden zwischen großen genaueren Modellen, die allerdings viel Rechenleistung benötigen und allgemein langsamer sind, oder kleineren Modellen, bei denen ein Abstrich in puncto Genauigkeit gemacht werden muss.

Für Anwendungen in Echtzeit ist die Testzeit-Performance, also die Zeit, die für eine Vorhersage benötigt wird, wichtig. Huang et al. (2017) geben in ihrer Arbeit einen Leitfaden für die Auswahl einer Modellarchitektur für Objekterkennungsprobleme. Das Ziel ist es die richtige Balance zwischen Geschwindigkeit, Speicherauslastung und Genauigkeit zu erreichen. Dabei ist der Vergleich von State-of-the-Art-Modell-Architekturen schwierig, da beispielsweise unterschiedliche Bildauflösungen, die verwendete Hardware bzw. Software eine große Rolle spielen. Meist sind auch die Ziele für die Entwicklung unterschiedlich. In vielen Anwendungen wie etwa in der Robotik, bei selbstfahrenden Autos und *Augmented Reality* müssen die Erkennungsaufgaben zeitnah und oft auf rechnerisch begrenzten Plattformen stattfinden (Huang et al., 2017). *MobileNet* (Howard et al., 2017) ist eine Leichtbau-Modell-Architektur, die für effiziente Vorhersagen in der mobilen Bildverarbeitung konzipiert wurde. Die Modell-Architektur *ModelNetV2* (Sandler et al., 2018) baut auf

der von *MobileNet* auf, verbessert allerdings die Genauigkeit auf verschiedenen Benchmark-Datensätzen.

4 State-of-the-Art Analyse von Datensätzen zur Handgestenklassifizierung

Zur Einteilung verschiedener Handgestendatensätze gibt es bereits einige Review- und Survey-Papers. Zu nennen sind hierbei (Ruffieux et al., 2014), (Vuletic et al., 2019) sowie (Firman, 2016) und (Zhang et al., 2016), welche sich allgemein mit RGB-D-Datensätzen beschäftigen. Diese Publikationen können ein Anlaufpunkt für Forscher sein, die Datensätze zur Gestenerkennung verwenden oder neue Datensätze generieren wollen. In den letzten Jahren wurden im Zuge der Forschung einige Datensätze zur Handgesten-Klassifizierung vorgestellt. Zu diesem Zweck wird im Folgenden ein aktueller Überblick zu den vorhandenen Datensätzen gegeben.

4.1 Einteilung der Datensätze

Aktuelle Datensätze wurden mit Hilfe der Survey Papers und einer Internetrecherche gesammelt und aufbereitet. Damit entstehen teilweise Mischformen einzelner Kategorien. Zur übersichtlichen Darstellung und Gegenüberstellung werden alle Datensätze in tabellarischer Form vorgestellt. Insgesamt wurden 32 Datensätze betrachtet.

Berücksichtigt wurden jene Datensätze, die aus einer wissenschaftlichen Publikation heraus entstanden sind. Der älteste Datensatz geht auf das Jahr 1996 zurück. Bei der Recherche wurde darauf geachtet, dass die Datensätze zum Zweck der Hand- und/oder Armgestenerkennung erhoben wurden. Handposen-Datensätze, bei denen es in erster Linie um die Position der Finger sowie deren Ausrichtung geht und Datensätze, welche Gesten mittels mobiler Geräte erfassen, wurden bei der Recherche nicht berücksichtigt.

Tabelle 2 zeigt die betrachteten Kriterien sowie deren Bedeutung und Unterscheidungsformen. Die Kriterien sind ähnlich wie in Ruffieux et al. (2014) gewählt. Oft lassen sich die Datensätze einer Kategorie nicht eindeutig zuordnen. Damit entstehen teilweise Mischformen einzelner Kategorien.

Tabelle 2: Kriterien zur Einteilung der Gestendatensätze

Kriterien:	Bedeutung und Unterscheidungsformen:
Ausführung	Unterschieden wird zwischen statischer und dynamischer Ausführung. Eine Kombination aus beiden Kategorien ist möglich.
Körperteile	Hierbei wird zwischen Händen, Armen, Oberkörper und Ganzkörper unterschieden, wobei die Übergänge zwischen den Kategorien fließend sein können.
Körperhaltung	Bei der Körperhaltung werden sitzende und stehende Körperhaltungen unterschieden. Einige wenige Datensätze beinhalten die Bewegung des gesamten Körpers.
Sensor/Kamera	Aufnahmesensor für den Gestendatensatz.
Sensorausrichtung	Grundsätzlich wird zwischen Frontalansicht, Seitenansicht oder Ansicht von Oben unterschieden. Auch mehrere Ausrichtungen für einen Datensatz sind möglich.
Hintergrund	Unterschieden wird zwischen einheitlich und komplex. Mischformen sind möglich.
Bildgröße und Frequenz	Die Bildgröße wird in Pixel (Breite x Höhe) angegeben. Sofern es sich um Videodateien handelt, wird zusätzlich die Bildfrequenz in Hertz angegeben.
Gestentyp	Die Gestentypen werden in der Datensatz-Tabelle nach (McNeill, 1992) (vgl. Kapitel 2.1 S16f.) eingeteilt, wobei deiktische- sowie rhythmisierende Gesten in den Datensätzen keine Rolle spielen. Mit der Kategorie Gebärdensprache wird ein weiterer Typ zur Gestenunterscheidung ergänzt.
Akteure	Anzahl der Teilnehmer zur Aufnahme des Gestendatensatzes.
Anzahl Gestenkategorien	Kategorien oder Klassen unterschiedlicher Gesten. Diese sind gleichzusetzen mit dem Begriff *Label* aus der ML-Theorie.
Beispielgesten	Gesamtanzahl an Bildern oder Videosequenzen von Gesten, die in einem Datensatz enthalten sind.
Verfügbarkeit	In dieser Kategorie wird zwischen „öffentlich“ und „auf Anfrage“ unterschieden.

Nachdem die Kriterien zur Einteilung der Datensätze festgelegt wurden, wird im Folgenden die tabellarische Einordnung der Datensätze vorgenommen. Tabelle 3 enthält die allgemeinen Kriterien und Tabelle 4 zeigt die Einteilung bezüglich technischer Kriterien.

Tabelle 3: Gestendatensätze allgemeine Kriterien

Nr.	Name	Zitiert	Ausführung	Körperteile	Körperhaltung	Gestentyp	Verfügbarkeit
1.	Jochen Triesch Static Hand Posture Database	Triesch, J., & Von Der Malsburg, C., 1996	statisch	Hände	k.A	Metaphorisch	öffentlich
2.	Sebastien Marcel Static Hand Posture Database	Marcel, S.,1999	statisch	Hände	k.A	Metaphorisch	öffentlich
3.	Jochen Triesch Static Hand Posture Database II	Triesch, J., & Von Der Malsburg, C., 2001	statisch	Hände	k.A	Metaphorisch	öffentlich
4.	Cambridge Hand Gesture Dataset (CHGD)	Kim et al., 2007	dynamisch	Hände	sitzend	Metaphorisch	öffentlich
5.	Keck Gesture Dataset	Lin et al., 2009	dynamisch	Hände und Arme	stehend; Bewegung	Metaphorisch, symbolisch	öffentlich
6.	NATOPS Aircraft Handling Signals Database	Song et al., 2011	statisch & dynamisch	Oberkörper	stehend	Metaphorisch, symbolisch	öffentlich
7.	ChaLearn Gesture Dataset 2011 (CGD2011)	Guyon et al., 2012	dynamisch	Oberkörper, Hände	stehend	Ikonisch, metaphorisch	öffentlich
8.	MSRGesture3D	Kurakin et al., 2012)	dynamisch	Hände	k.A	Gebärdensprache	öffentlich
9.	G3D Dataset	Bloom et al., 2012	dynamisch	Ganzkörper	stehend	Ikonisch	öffentlich
10.	Microsoft Research Cambridge- 12 (MSCR-12)	Fothergill et al., 2012	statisch & dynamisch	Ganzkörper	stehend	Ikonisch, metaphorisch	öffentlich
11.	American Sign Language Lexicon Video Dataset (ASLLVD)	Neidle et al., 2012	dynamisch	Oberkörper	sitzend	Gebärdensprache	öffentlich
12.	3D Iconic Gesture Dataset (3DIG)	Sadeghipour et al., 2012	dynamisch	Hände und Arme	stehend	Ikonisch	öffentlich
13.	6D Motion Gesture Database (6DMG)	Chen et al., 2012	dynamisch	k.A	k.A	Ikonisch, metaphorisch	öffentlich
14.	Sheffield Kinect gesture dataset (SKIG)	Liu & Shao, 2013	dynamisch	Hände	sitzend	Ikonisch, metaphorisch	auf Anfrage
15.	ChAir Gest	Ruffieux et al., 2013	dynamisch	Hände und Arme	sitzend	Ikonisch, metaphorisch	auf Anfrage
16.	Montalbano Dataset	Escalera et al., 2013	dynamisch	Oberkörper	stehend	Metaphorisch	öffentlich
17.	LaRed Dataset	Hsiao et al., 2014	statisch & dynamisch	Hände	sitzend	Gebärdensprache; Metaphorisch	auf Anfrage

18.	Vision for Intelligent Vehicles and Applications (VIVA) Dataset	Ohn-Bar & Trivedi, 2014	dynamisch	Hände	sitzend	Metaphorisch	öffentlich
19.	LTTM MS Kinect & Leap Motion	Marin et al., 2014	dynamisch	Hände	sitzend	Gebärdensprache	öffentlich
20.	G3Di Dataset	Bloom et al., 2016	dynamisch	Ganzkörper	stehend	Metaphorisch	öffentlich
21.	HKU EEE DSP Kinect Gesture Dataset	Wang et al., 2014	statisch & dynamisch	Hände	sitzend	Metaphorisch	öffentlich
22.	LTTM Creative Senz3D	Memo et al., 2015	statisch	Hände	sitzend	Metaphorisch	öffentlich
23.	ChaLearn LAP Isolated Gesture Dataset (IsoGD)	Wan et al., 2016	dynamisch	Oberkörper, Hände	stehend	Ikonisch, metaphorisch	auf Anfrage
24.	ChaLearn LAP Continuous Gesture Dataset (ConGD)	Wan et al., 2016	dynamisch	Oberkörper, Hände	stehend	Ikonisch, metaphorisch	auf Anfrage
25.	OUHANDS Database	Matilainen et al., 2016	statisch	Oberkörper, Hände	sitzend	Metaphorisch	öffentlich
26.	NVIDIA Dynamic Hand Gesture Dataset	Molchanov et al., 2016	dynamisch	Hände und Arme	sitzend	Metaphorisch	auf Anfrage
27.	Tiny Hand Gesture Dataset	Bao et al., 2017	statisch	Oberkörper	stehend; sitzend	Metaphorisch, symbolisch	öffentlich
28.	RPPDI Gesture Dataset	Barros et al. 2017	dynamisch	Hände	k.A	Metaphorisch	öffentlich
29.	Static Hand Gesture ASL Dataset	Pinto et al., 2019	statisch	Hände	k.A	Gebärdensprache	auf Anfrage
30.	Multi-Modal Hand Gesture Dataset for Hand Gesture Recognition	Mantecón et al., 2019	statisch & dynamisch	Hände	k.A	Metaphorisch	öffentlich
31.	Jester Dataset	Materzynska et al., 2019	dynamisch	Hände	sitzend	Metaphorisch, symbolisch	öffentlich
32.	IPN Hand Dataset	Benitez-Garcia et al., 2020	statisch & dynamisch	Hände	sitzend	Metaphorisch	öffentlich

Tabelle 4: Gestendatensätze technische Kriterien

Nr.	Name	Jahr	Sensor/Kamera	Sensor-Ausrichtung	Hintergrund	Bildgröße	Frequenz	Akteure	Gestenkategorien	Beispielgesten	Größe (GB)
1.	Jochen Triesch Static Hand Posture Database	1996	Kamera schwarz/weiß	Frontalansicht	einheitlich; komplex	128x128	k.A	24	10	657	0,01
2.	Sebastien Marcel Static Hand Posture Database	1999	Farbkamera	Frontalansicht	einheitlich; komplex	77x66	k.A	10	6	5531	0,12
3.	Jochen Triesch Static Hand Posture Database II	2001	Farbkamera	Frontalansicht	einheitlich; komplex	128x129	k.A	19	12	1145	0,03
4.	Cambridge Hand Gesture Dataset (CHGD)	2007	Farbkamera	Ansicht von oben	einheitlich	320x240	k.A	2	9	900	1,00
5.	Keck Gesture Dataset	2009	Farbkamera	Frontalansicht Motion View	einheitlich; komplex	640x480	k.A	3	14	294	0,15
6.	NATOPS Aircraft Handling Signals Database	2011	Stereo Kamera	Frontalansicht	einheitlich	320x240	20	20	24	9600	20,00
7.	ChaLearn Gesture Dataset 2011 (CGD2011)	2011	MS Kinect	Frontalansicht	einheitlich; komplex	320x240	10	20	30	50000	30; 5 (verkl.)
8.	MSRGesture3D	2012	MS Kinect	Frontalansicht	einheitlich	130x130	20	10	12	336	0,03
9.	G3D Dataset	2012	MS Kinect	Frontalansicht	komplex	640x480	30	10	20	600	47,00
10.	Microsoft Research Cambridge-12 (MSCR-12)	2012	MS Kinect	Frontalansicht	komplex	k.A	30	30	12	6244	0,20
11.	American Sign Language Lexicon Video Dataset (ASLLVD)	2012	4 Farbkameras	3 Frontalansicht, Seitenansicht	einheitlich	640x480; 1600x1200	60; 30	6	2742	9794	1.6 (verkl.)
12.	3D Iconic Gesture Dataset (3DIG)	2012	MS Kinect	Frontalansicht	einheitlich	640x480	30	29	20	1739	k.A
13.	6D Motion Gesture Database (6DMG)	2012	Optischer Tracker, Wiimote	k.A	einheitlich	k.A	60	28	20	5600	0,02
14.	Sheffield Kinect gesture dataset (SKIG)	2013	MS Kinect	Ansicht von oben	einheitlich; komplex	320x240	10	6	10	1080	1,20
15.	ChAir Gest	2013	MS Kinect; Xsens IMU	Frontalansicht	komplex	640x480	30; 50	10	10	1200	3 (verkl.)
16.	Montalbano Dataset	2013	MS Kinect	Frontalansicht	einheitlich	640x480	20	27	20	13858	27
17.	LaRed Dataset	2014	Intel Tiefenkamera	Frontalansicht	komplex	28x28	15	10	27	243000	k.A
18.	Vision for Intelligent Vehicles and Applications (VIVA) Dataset	2014	MS Kinect	Ansicht von oben	komplex	640x480; 115x250	k.A	8	19	5550	2
19.	LTTM MS Kinect & Leap Motion	2014	MS Kinect; Leap Motion Sensor	Frontalansicht	komplex	k.A	k.A	10	10	1400	3,30
20.	G3Di Dataset	2014	MS Kinect	Frontalansicht	komplex	640x480	30	12	17	k.A	k.A

21.	HKU EEE DSP Kinect Gesture Dataset	2014	MS Kinect	Frontalansicht	komplex	k.A	k.A	5	10	1000	k.A
22.	LTTM Creative Senz3D	2015	Creative Senz3D Kamera	Frontalansicht	komplex	320x240	6 bis 60	4	11	1320	0,8
23.	ChaLearn LAP Isolated Gesture Dataset (IsoGD)	2016	MS Kinect	Frontalansicht	einheitlich; komplex	320x240	10	21	249	47933	9
24.	ChaLearn LAP Continuous Gesture Dataset (ConGD)	2016	MS Kinect	Frontalansicht	einheitlich; komplex	320x240	10	21	249	47933	4
25.	OUHANDS Database	2016	Intel RealSense F200	Frontalansicht	komplex	640x480	k.A	23	10	3150	3,9
26.	NVIDIA Dynamic Hand Gesture Dataset	2016	SoftKinetic Tiefenkamera (DS325) ; DUO 3D Kamera	Frontalansicht; Ansicht von oben	einheitlich	320x240	30	20	25	1532	k.A
27.	Tiny Hand Gesture Dataset	2017	Webcam	Frontalansicht	einheitlich; komplex	640x480	k.A	40	7	500000	54,6
28.	RPPDI Gesture Dataset	2017	Smartphone Kamera	Frontalansicht	einheitlich	640x480	k.A	1	7	188	0,2
29.	Static Hand Gesture ASL Dataset	2019	Webcam	Frontalansicht	einheitlich	400x400	k.A	8	24	11100	0,09
30.	Multi-Modal Hand Gesture Dataset for Hand Gesture Recognition	2019	Leap Motion Sensor	Frontalansicht	einheitlich	640x240	30 bis 200	25	15	75000	4,9
31.	Jester Dataset	2019	Webcam	Frontalansicht	komplex	Var x100	k.A	1376	27	148092	23
32.	IPN Hand Dataset	2020	Webcam	Frontalansicht	komplex	640x480	30	50	13	4218	14,6

4.2 Zusammenfassung und Analyse der Datensätze

Die Anzahl, der zur Verfügung stehenden Gestenkategorien, kann ein wichtiges Kriterium bei der Auswahl eines geeigneten Datensatzes sein. Sie gibt Auskunft wie viele verschiedene Labels in einem Datensatz zur Klassifizierung bereitstehen. Bei der Betrachtung der Anzahl der verschiedenen Gestenkategorien fällt auf, dass der Maximalwert an Gestenkategorien bei 2742 liegt. Die hohe Anzahl an Labels lässt sich mit der komplexen Struktur der Gebärdensprache, die im *American Sign Language Lexicon Video Dataset* dargestellt werden soll, erklären. Für eine Gestensteuerung macht eine derart hohe Anzahl an Gestenkategorien keinen Sinn. Die Datensätze *ChaLearn LAP Isolated Gesture Dataset (IsoGD/ConGD)* haben zwar deutlich weniger Gestenkategorien (249), allerdings würde auch diese Menge zu einer deutlichen kognitiven Überbelastung für den Mitarbeiter führen. Betrachtet man die Anzahl der Gestenkategorien pro Datensatz der restlichen 29 Datensätze kann man feststellen, dass ein Mindestmaß an 6 Gestenkategorien nicht unterschritten wird. Der arithmetische Mittelwert liegt bei 16 und der Median bei 14 Gestenkategorien pro Datensatz. Eine rechtsschiefe Verteilung ist zu beobachten. Die Klassifikation in 10 Gestenkategorien wird in 6 der 30 betrachteten Datensätze verwendet. Eine Häufigkeitsverteilung der Gestenkategorien-Anzahl ist in Abbildung 18 abgebildet.

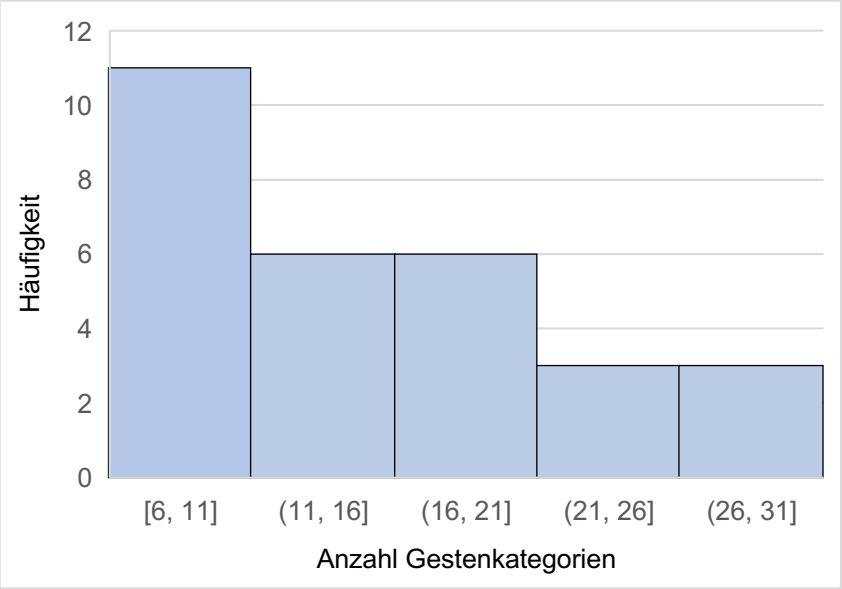


Abbildung 18: Häufigkeitsverteilung der Anzahl von Gestenkategorien (n=29)

Der Hintergrund in den Aufnahmen der Beispielgesten ist bei den meisten betrachteten Beispielgesten komplex oder zumindest zum Teil komplex. Unter komplex wird dabei ein nicht einheitlicher Hintergrund verstanden, bei dem andere Personen, Gegenstände oder dergleichen vorkommen. 12 der 32 betrachteten Datensätze bestehen ausschließlich aus Bildern oder Videoaufnahmen mit einheitlichem Hintergrund. Abbildung 19 gibt einen Überblick über die Verteilung der Hintergrundeigenschaften.

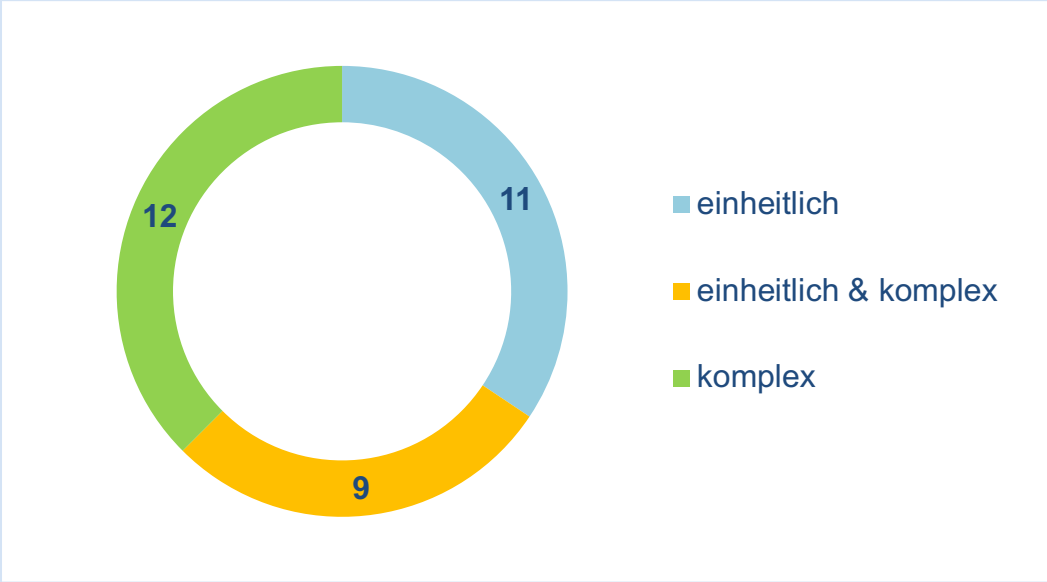


Abbildung 19: Verteilung der Hintergrund-Eigenschaften (n=32)

Beim Gestentyp wird zwischen metaphorisch, symbolisch (Synonym für emblematische Gesten vgl. Kapitel 2), ikonisch und der Kategorie Gebärdensprache unterschieden. Für die HCI besonders interessant sind metaphorische Gesten, da sie ein Konzept zum Ausdruck bringen. Jedoch sich auch symbolische Gesten allgemein gut verständlich und somit leicht erlernbar. Ein Teil der Handgesten-Datensätze spezialisiert sich auf Gebärdensprache. Die Gebärdensprache kann als Sonderform der gestischen Interaktion betrachtet werden. Häufig werden Gesten aus den Gebärdensprachenübernommen und für Interaktionskonzepte genutzt. Da der Übergang zwischen Gestentypen oft fließend ist oder in Datensätzen Gesten verwendet werden, die verschiedenen Kategorien zuteilbar sind, ist eine eindeutige Zuordnung oft nicht möglich. Einen Überblick über die Gestentypen in den betrachteten Datensätzen gibt Abbildung 20.

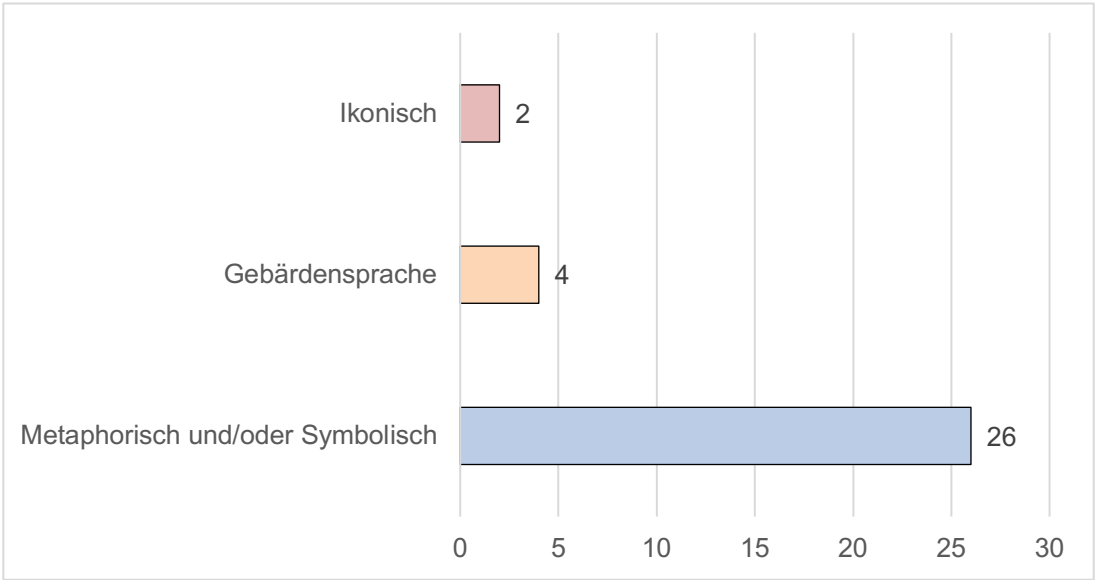


Abbildung 20: Häufigkeitsverteilung der Gestentypen (n=32)

Bei den Beispielgesten pro Gestenkategorie liegt der Durchschnitt bei 3425 (n=29). Der Median liegt bei 280 Beispielgesten pro Gestenkategorie und ist in diesem Fall aussagekräftiger. Zur Bestimmung wurde die Gesamtzahl der Beispielgesten durch die Gestenkategorien dividiert und anschließend gemittelt. Vor allem Ausreißer nach oben, wie etwa der *Tiny Hand Gesture*-Datensatz (Bao et al., 2017) mit 71429 Beispielgesten pro Kategorie oder der *LaRed*-Datensatz (Hsiao et al., 2014) mit 9000 Beispielgesten pro Kategorie verschieben hierbei den arithmetisches Mittelwert nach oben. In einigen Datensätzen herrscht eine ungleichmäßige Verteilung der Beispielgesten über die Kategorien. Diese Tatsache wird vernachlässigt, da es für eine grobe Schätzung keine Rolle spielt.

Wenn man sich die Gesamtzahl an Beispielgesten der letzten 5 Jahre (ab inklusive 2015) ansieht (Mittelwert = 72.924 Beispielgesten; Median = 22.864; n=12) so fällt auf, dass die Beispielgesten und somit die Datenmenge im Vergleich zu den Datensätzen bis inklusive 2014 (Mittelwert = ca. 17.976 Beispielgesten; Median =1570 22.864; n=20) deutlich zugenommen hat. Einige Ausreißer nach oben sind in den Daten festzustellen. Nennenswert ist dabei der *LaRed*-Datensatz (Hsiao et al., 2014) mit 243.000 Beispielgesten und der *Tiny Hand Gesture*-Datensatz (Bao et al., 2017) mit 500.000 Beispielgesten.

Der Jester-Datensatz (Materzynska et al., 2019) wurde mittels *Croudsourcing* entwickelt und enthält Videoclips von insgesamt 1376 Personen. Bei der Erstellung eines eigenen Datensatzes ist dies prinzipiell eine gute Möglichkeit, um möglichst viele Daten zu generieren und somit die Generalisierungs-Fähigkeit des Algorithmus zu erhöhen. Für die Entwicklung und den Test eines Handgesteninteraktionskonzepts stellt diese Art der Datenbeschaffung allerdings einen erheblichen Mehraufwand dar. Deshalb wird der Jester-Datensatz bei der folgenden Abschätzung zur durchschnittlich teilnehmenden Personenanzahl zur Gestenaufnahme nicht berücksichtigt. Der in Abbildung 21 dargestellte Boxplot stellt eine anschauliche Übersicht über die Teilnehmerzahl dar. Der Median liegt bei 12 Akteuren pro Datensatz (Mittelwert = ~ 17 Personen; n=31). Der Interquartilsabstand beträgt 8-24 =16. Dies zeigt, wie die mittleren 50% der Datensätze in Bezug auf die teilnehmenden Akteure verteilt sind.

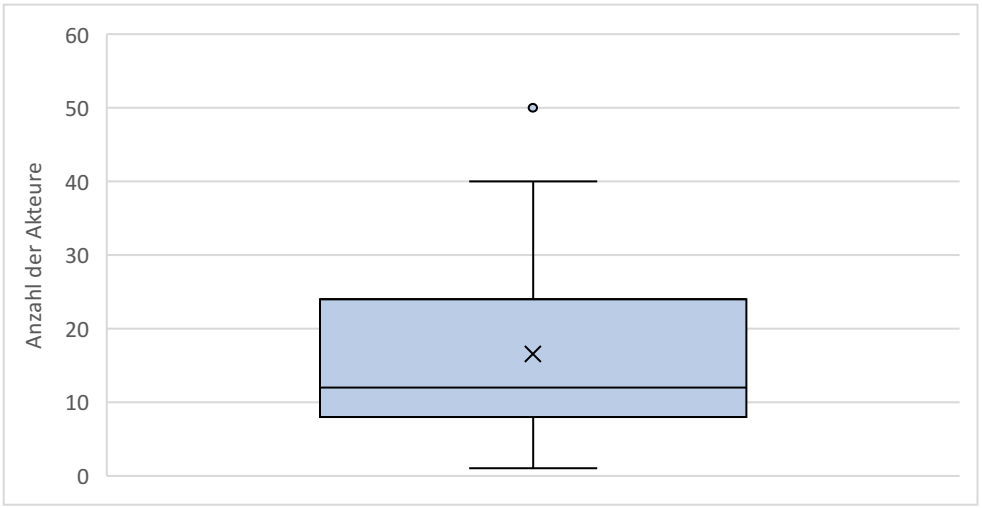


Abbildung 21: Boxplot Anzahl Akteure pro Datensatz (n=31)

Abbildung 22 zeigt wie die Anzahl der Akteure bis einschließlich 2014 (Median=10 Akteure; n=21) verteilt ist, im Vergleich zu der Verteilung ab 2015 (Median=21 Akteure; n=10). Dabei lässt sich ein eindeutiger Trend hin zu mehr Akteuren beobachten.

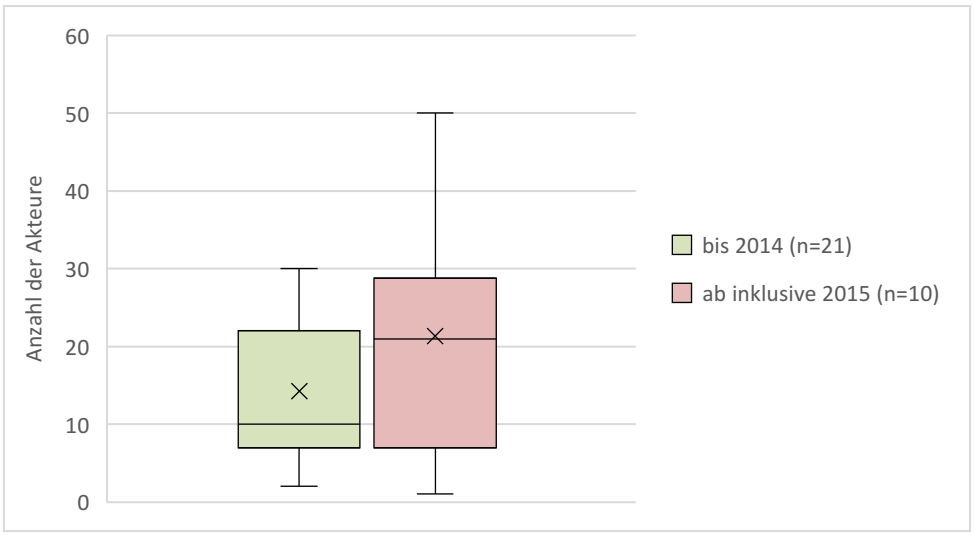


Abbildung 22: Boxplot Anzahl Akteure bis 2014 vs. ab inklusive 2015 (n=31)

Die Microsoft Kinect erfreute sich nach ihrer Einführung im Jahre 2010 in der Handgestenerkennung. großer Beliebtheit. Von 2010 bis 2016 wurden $^{2}/_{3}$ (14 von 21) der Datensätze mit Hilfe der MS Kinect erzeugt. Zurückzuführen ist dies auf die Möglichkeit der Tiefenmessung mittels integrierter Tiefensensoren. Der *Leap Motion Controller* ist ein weiterer Sensor der kontaktlose Hand- und Fingergesten-Erkennung ermöglicht. In Abbildung 23 wird der *Leap Motion Controller* sowie die *Microsoft Kinect* für die Spielekonsole *Xbox 360* dargestellt.

Abbildung 23: Rechts Leap Motion Controller (Bildquelle: Ultraleap); Links Microsoft Kinect (Bildquelle: Microsoft)

In den letzten Jahren ist jedoch ein Trend hin zu normalen Webcam-Aufnahmen zu erkennen. Eine mögliche Erklärung ist die große Verfügbarkeit sowie die Fortschritte im Bereich des DL, die eine zusätzliche Tiefenmessung nicht mehr zwingend erfordert.

Die meisten Datensätze stehen bei ordnungsgemäßer Zitierung öffentlich zur Verfügung. Sieben der 32 Datensätze sind nur auf Anfrage und nach Unterzeichnung einer Nutzungsberechtigung erhältlich. Eine allgemeine Aussage über gültige Lizenzvereinbarungen für Handgesten-Datensätze kann nicht getroffen werden.

Ziel der Analyse sind konkrete Gestaltungshinweise, die bei der Auswahl oder der Gestaltung eines eigenen Gestendatensatzes hilfreich sein können. Darüber hinaus sollen sie dazu beitragen, auch Trends und Anomalien in den Daten ausfindig zu machen.

Folgende Erkenntnisse gehen aus der Analyse der Handgestendatenbanken hervor:

- Der Durchschnitt der Gestenkategorien liegt bei 16 Gesten (Median = 14; n=30), wobei dies stark variiert und meist speziell an den Anwendungskontext angepasst wird.
- Im Großteil der Fälle wird bei der Gestaltung eines Datensatzes ein komplexer Hintergrund berücksichtigt (21 der 32 betrachteten Datensätze).
- Bei der Zusammenstellung eines Gestendatensatzes sind ausreichend Beispielgesten notwendig. Der Median der aufgenommenen Beispielgesten pro Kategorie liegt bei 280.
- Vergleicht man Datensätze bis 2014 und ab (inklusive) 2015 so lässt sich feststellen, dass sich ein Trend hin zu mehr Akteuren und Beispielgesten (größere Datenmengen) abzeichnet.

Wie bereits bei der Vorgehensweise beschrieben, wurden in der vorliegenden Datenbankrecherche nur Datensätze berücksichtigt, zu denen eine wissenschaftliche Publikation vorhanden ist. Eine weitere Anlaufstelle für Handgesten-Datensatze ist die

Plattform *kaggle.com*. Auf dieser Plattform können Anwender unterschiedlichen Datensätzen finden oder veröffentlichen, darunter auch Datensätze zur Handgestenerkennung bzw. Klassifikation.

4.3 Nutzwertanalyse zur Datensatzauswahl

Im folgenden Kapitel wird eine Nutzwertanalyse zur Auswahl eines geeigneten Datensatzes für die industrielle Baustellenmontage durchgeführt. Aufgeteilt ist die folgende Nutzwertanalyse in drei Teile. Die Nutzwertanalyse besteht aus der Konzeptionsphase, der Bewertungsphase und schließlich der Bewertungsphase.

4.3.1 Konzeptionsphase

In der **Konzeptionsphase** werden zunächst die Ziele für die Auswahl eines Datensatzes formuliert. Anschließend können Bewertungskriterien für das Modell zur Handgestenklassifizierung definiert werden. In weiterer Folge werden Ziele im Sinne der Usability eines Handgestenklassifizierungsmodells definiert. Mit Hilfe dieser Ziele und des durch den Use-Case vorgegebenen Handlungsspielraums sollen geeignete Bewertungskriterien und Lösungsalternativen gefunden werden.

- **Benutzerfreundlichkeit:** Die Benutzung des Systems soll möglichst einfach erlernbar sein. Eine hohe Anzahl an Gestenkategorien kann zu Verwirrung beim Nutzer führen und ihn evtl. kognitive überfordern. Datensätze, die für die Gebärdensprache konzipiert wurden, scheiden somit bei der Auswahl geeigneter Datensätze aus.

- **Erkennungsrate:** Die Erkennungsrate des Handgestenklassifizierungsmodells soll möglichst hoch sein. Das bedeutet bei der Auswahl der Datensätze müssen ausreichend Daten (Beispielgesten) zum Trainieren des Datensatzes vorhanden sein. Eine genaue Abschätzung kann in dieser Phase allerdings nicht getroffen werden.

- **Technische Umsetzung:** Das Bewertungskriterium „Ausführung“ in Kapitel 4.1 ist direkt mit der Komplexität der Umsetzung verbunden. Das bedeutet eine Handgestenerkennung mittels statischer Gesten ist technisch einfacher zu implementieren als mit dynamischen Gesten. Für den vorliegenden Use-Case kann ein Konzept mit statischen Handgesten das Interagieren mit einem Informationssystem bereits ausreichen abbilden. Zudem könnte es bei dynamischen Gesten im Arbeitsalltag vermehrt zu Fehlinteraktionen kommen.

Einige Kriterien, die bei der Kategorisierung in Kapitel 4.1 verwendet wurden, sind bereits durch das Anwendungsgebiet in der industriellen Baustellenmontage vorgegeben:

- Die Ansicht von Oben ist als **Sensorausrichtung** für die industrielle Baustellenmontage zu bevorzugen. Allerdings können auch Datensätze, die in der Frontalansicht aufgezeichnete wurden, sinnvoll verwendet werden, wenn die Anordnung des Montageplatzes dies zulässt. Zudem kann durch geeignete Transformationen beispielweise das Spiegeln von Gesten in der Frontalansicht eine künstliche Ansicht von Oben erzeugt werden.
- Das Kriterium **Körperteile** kann bei der Suche nach geeigneten Datensätzen auf Hände und Arme begrenzt werden.
- Eine **stehende Körperhaltung** ist durch das Einsatzgebiet zwar vorgegeben, allerdings spricht nichts gegen die Anwendung von Datensätzen, die im Sitzen aufgenommen wurden.

Um die in Kapitel 4.1 recherchierten Datensätze auf eine angemessene Anzahl an Lösungsalternativen einzugrenzen werden folgende Einschränkungen getroffen:

- Zunächst wird die Auswahl der Lösungsalternativen auf die Ausführung „statisch“ begrenzt.
- Das Kriterium **Körperteile** werden Datensätze der Ganzkörperkategorie nicht berücksichtigt. Da der Übergang zwischen Oberkörper und Händegestendatensätze oft fließend ist, werden beide Kategorien berücksichtigt.
- Datensätze die als **Gestentyp** Gebärdensprache enthalten werden nicht berücksichtigt.
- Die Analyse der **Gestenkategorienanzahl** ergibt einen Median von 14 Kategorien (n=29). Datensätze mit mehr als 14 Kategorien werden als Lösungsalternative im Folgenden nicht berücksichtigt.

Die oben genannten Einschränkungen werden auf die recherchierten Datensätze aus Kapitel 4.1 angewendet und somit die Lösungsalternativen eingeschränkt. Aus den verbliebenen Datensätzen werden die vier Datensätzen mit den meisten Beispielgesten als Lösungsalternativen für die folgende Nutzwertanalyse herangezogen. Nachfolgende werden die Datensätze in chronologischer Reihenfolge im Detail beschrieben.

Sebastien Marcel Static Hand Posture Database

Dieser Datensatz enthält insgesamt 5531 Beispielgesten. Sie wurden in 4872 Trainingsbilder und 659 Testbilder gegliedert. Marcel unterscheidet sechs

Gestenkategorien. Das Aufnahmeformat beträgt 77x66 Pixel. Abbildung 24 zeigt die unterschiedlichen Gestenkategorien inklusive Begrenzungsrahmen (engl. *bounding boxes*) für die Handregion (Marcel, 1999).

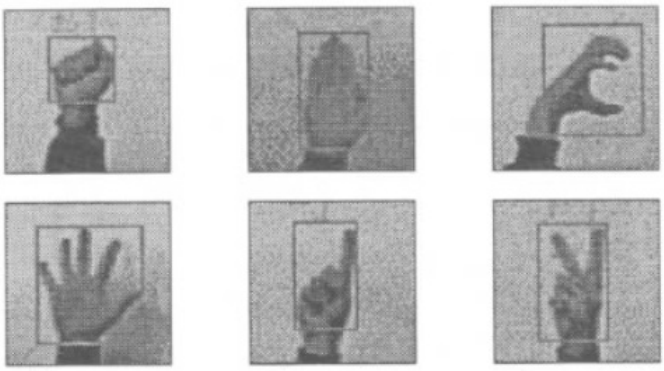


Abbildung 24: Gestensatz Sebastien Marcel Static Hand Posture Database

OUHANDS Database

Der OUHANDS-Datensatz enthält statische Handgesten. Der Datensatz kann sowohl für Handgesten-Klassifikation, als auch für Hand-Erkennungsaufgaben genutzt werden. Zur Aufnahme der Beispielgesten dient ein *Intel RealSense F200*-Sensor, der Bilder in einer Auflösung von 640x480 aufnimmt. Zusätzlich zu den RGB-Daten wurden auch die Tiefendaten erfasst. Der Datensatz enthält insgesamt 3150 Beispielgesten, die in 10 Gestenkategorien gegliedert wurden. 23 Akteure sind bei der Ausführung der Gesten für den OUHANDS-Datensatz beteiligt. Abbildung 25 zeigt die Gestenkategorien, ausgeführt von unterschiedlichen Akteuren. Hinzu kommen zudem 5288 Aufnahmen ohne Handgesten, die für Hand-Erkennungsaufgaben hilfreich sind (Matilainen et al., 2016).

Abbildung 25: Gesten aus dem OUHANDS Database

LTTM Creative Senz3D

Der LTTM Creative Senz3D Datensatz enthält 11 verschiedene Gestenkategorien, die von vier Akteuren ausgeführt wurden. Die Gesten wurden mit Hilfe einer Creative Senz3D-Kamera erfasst und im Zuge der Publikation von Memo, Minto & Zanuttigh, (2015) erstellt. Anschließend wurde der Datensatz öffentlich zugänglich gemacht. In Abbildung 26 ist eine Beispielgeste in RGB- sowie Tiefen-Format und die *Creative Senz Kamera* zu sehen. Jede Gestenkategorie wurde von jedem Akteur 30-mal wiederholt, was einer Gesamtzahl von 1320 Beispielgesten entspricht.

Abbildung 26: LTTM Creative Senz3D; links Creative Senz3D-Kamera, rechts Beispielgesten aus dem Datensatz

Tiny Hand Gesture Dataset

Der *Tiny Hand Gesture*-Datensatz besteht aus RGB-Videos mit sieben verschiedenen Gestenkategorien. Durchgeführt werden die Beispielgesten von insgesamt 40 Akteuren. Die Hälfte der Akteure führen Gesten vor komplexem Hintergrund durch, die andere Hälfte unter einheitlichen Hintergrundbedingungen. In Abbildung 27 sind die unterschiedlichen Gestenkategorien mit einheitlichem Hintergrund zu sehen. Die zu klassifizierende Geste macht in den Aufnahmen nur ca. 10% der Pixel des Gesamtbildes aus. Die Auflösung der Bilder beträgt 640x480. Insgesamt umfasst der Datensatz 500.000 Beispielgesten (Bao et al., 2017).

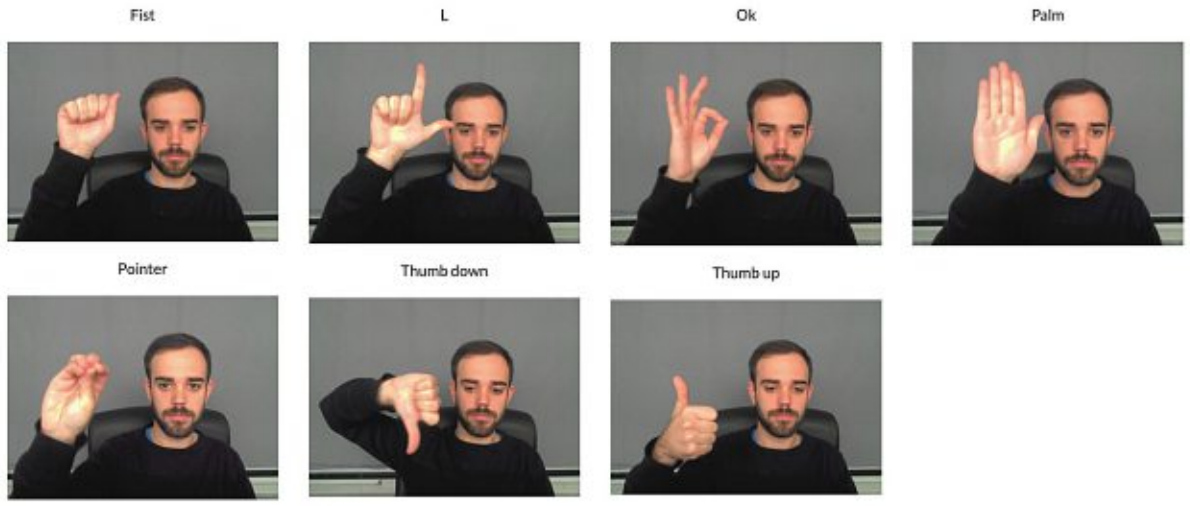


Abbildung 27: Gesten aus dem Tiny Hand Gesture Datensatz

4.3.2 Bewertungsphase

Im nächsten Schritt der Nutzwertanalyse werden Bewertungskriterien bestimmt. Bei der Auswahl der Bewertungskriterien gilt es zu beachten, dass diese möglichst quantifizierbare Merkmale enthalten. Außerdem sollten möglichst alle Kriterien für eine vorgegebenes Ziel angeführt werden. Schließlich sollten die Bewertungskriterien nicht den gleichen Inhalt abbilden und voneinander unabhängig sein (Herbig, 2016).

Für die Auswahl der Bewertungskriterien können viele der betrachteten Kriterien im State of the Art Teil übernommen werden. Das Kriterium „Anzahl Gestenkategorien" gibt nur zum Teil Aufschluss auf die Variation und Vielfalt des Datensatze. Um diese Eigenschaft besser bewerten zu können wird ein Bewertungskriterium für die Variation des Datensatzes eingeführt, welche qualitativ bewertet werden kann. Das Kriterium der Verfügbarkeit wird nicht als Bewertungskriterium berücksichtigt, weil nur verfügbare Lösungsalternativen in Betracht gezogen werden. Die ausgewählten Kriterien sind in Tabelle 5 mittels eines paarweisen Vergleichs dargestellt.

Tabelle 5: Paarweiser Vergleich der Bewertungskriterien

Bewertungskriterien	Hintergrund	Bildgröße	Bildqualität	Anzahl Akteure	Anzahl Gestenkategorien	Variation Gestenkategorien	Anzahl Beispielgesten	Größe (Datensatz)	**Summe**	**Gewichtungsfaktor**	**Gewichtungsfaktor (%)**
Hintergrund		2	0	1	1	0	0	2	6	0,107	10,7%
Bildgröße	0		0	0	0	0	0	1	1	0,018	1,8%
Bildqualität	2	2		2	1	1	1	2	11	0,196	19,6%
Anzahl Akteure	1	2	0		1	0	0	2	6	0,107	10,7%
Anzahl Gestenkategorien	1	2	1	1		0	1	2	8	0,143	14,3%
Variation Gestenkategorien	2	2	1	2	2		1	2	12	0,214	21,4%
Anzahl Beispielgesten	2	2	1	2	1	1		2	11	0,196	19,6%
Größe (Datenmenge)	0	1	0	0	0	0	0		1	0,018	1,8%
									56	1	100%

Tabelle 6 gibt einen Vergleich der vier Lösungsalternativen und gibt Auskunft über quantitative Kriterien sowie qualitative Eigenschaften hinsichtlich der Bewertungskriterien.

Tabelle 6: Vergleich der Lösungsvarianten

Bewertungskriterium	Sebastien Marcel Static Hand Posture Database	OUHANDS Database	LTTM Creative Senz3D	Tiny Hand Gesture Dataset
Hintergrund	einheitlich	komplex mit variierendem Hintergrund	komplex, teilweise Personen im Hintergrund	komplex mit variierendem Hintergrund
Bildgröße	77x66	640x480	320x240	640x480
Bildqualität	schlecht, sehr verrauschte Aufnahmen	ok	ok	gut
Anzahl Akteure	10	23	4	40
Anzahl Gestenkategorien	6	10	11	7
Variation Gestenkategorien	gut unterscheidbar	Großteils gut unterscheidbar	Großteils gut unterscheidbar	gut unterscheidbar
Anzahl Beispielgesten	5531	3150	1320	500.000
Größe (Datenmenge)	0,12	3,9	0,8	54,6

Als nächster Schritt werden die Zielerfüllungsfaktoren ermittelt. Bei diesem Schritt wird ermittelt, wie gut jede Alternative die Bewertungskriterien erfüllt. Mit Hilfe einer „Erfüllungsskala“ kann ermittelt werden welcher Datensatz die Bewertungskriterien am besten, zweitbesten usw. erfüllt (Herbig, 2016). Die Ergebnisse sind in der Spalte „Zielerfüllungsfaktor“ in Tabelle 8 zu sehen. Der Datensatz, der ein Bewertungskriterium am besten erfüllt, erhält in folgender Analyse den Erfüllungsfaktor 4. Der Datensatz, der das Bewertungskriterium am schlechtesten erfüllt erhält den Erfüllungsfaktor 1. Sind zwei oder mehrere Alternativen gleichwertig hinsichtlich des Erfüllungsfaktor wird das arithmetische Mittel der Rangplatzfaktoren angewendet. Die Summe einer Reihe ergibt hierbei immer den Wert 10 (4+3+2+1).

Tabelle 7: Zielerfüllungsfaktoren

Bewertungskriterium	Sebastien Marcel Static Hand Posture Database	OUHANDS Database	LTTM Creative Senz3D	Tiny Hand Gesture Dataset
Hintergrund	4	2	2	2
Bildgröße	1	3,5	2	3,5
Bildqualität	1	2	3	4
Anzahl Akteure	2	3	1	4
Anzahl Gestenkategorien	3,5	1,5	1,5	3,5
Variation Gestenkategorien	3,5	1,5	1,5	3,5
Anzahl Beispielgesten	3	2	1	4
Größe (Datenmenge)	1,5	3,5	3,5	1,5

4.3.3 Ergebnisphase

In Tabelle 8 ist die Nutzwertanalyse für die Auswahl eines geeigneten Datensatzes für den vorliegenden Use-Case dargestellt. Dabei wird der **Gewichtungsfaktor (GF)** mit dem **Zielerfüllungsfaktor (ZEF)** multipliziert. Dieser ergibt den **Teilnutzwert (TNW)**, welcher im Anschluss aufsummiert wird und den Nutzwert der jeweiligen Lösungsalternative ergibt.

Tabelle 8: Nutzwertanalyse Handgestendatensätze

Bewertungskriterium	GF	Alternativen							
		Sebastien Marcel SHPD		OUHANDS Database		LTTM Creative Senz3D		Tiny Hand Gesture Dataset	
		ZEF	TNW	ZEF	TNW	ZEF	TNW	ZEF	TNW
Hintergrund	0,11	4	0,43	2	0,21	2	0,21	2	0,21
Bildgröße	0,02	1	0,02	3,5	0,06	2	0,04	3,5	0,06
Bildqualität	0,20	1	0,20	2	0,39	3	0,59	4	0,79
Anzahl Akteure	0,11	2	0,21	3	0,32	1	0,11	4	0,43
Anzahl Gestenkategorien	0,14	3,5	0,50	1,5	0,21	1,5	0,21	3,5	0,50
Variation Gestenkategorien	0,21	3,5	0,75	1,5	0,32	1,5	0,32	3,5	0,75
Anzahl Beispielgesten	0,20	3	0,59	2	0,39	1	0,20	4	0,79
Größe (Datenmenge)	0,02	1,5	0,03	3,5	0,06	3,5	0,06	1,5	0,03
Gesamtnutzwert			**2,72**		**1,98**		**1,74**		**3,55**
Rangfolge			**2**		**3**		**4**		**1**

Den höchsten Nutzwert liefert das **Tiny Hand Gesture Dataset**. Bei den anschließenden Modellierungsversuchen und Tests mit dem *Tiny Hand Gesture Dataset* kam es aufgrund mehrerer Faktoren, darunter die Größe des Datensatzes und den sehr unterschiedlichen Gegebenheiten im Vergleich zum Use-Case zu keinen brauchbaren Ergebnissen. Auch die Verwendung der restlichen Datensätze führten zu keinem brauchbaren Ergebnis. Daraufhin wurde abgewogen ob es sinnvoller ist die Handregionen aller Bilder im **Tiny Hand Gesture Dataset** zu extrahieren oder einen komplett neuen Datensatz zu generieren. Der Aufwand wurde abgeschätzt und die Option der Schaffung eines neuen Datensatzes vorgezogen. In Kapitel 5 wird der Modellierungsteil der vorliegenden Arbeit durchgeführt.

5 Modell zur Handgestenklassifikation

Im praktischen Teil der vorliegenden Arbeit sollen mit Hilfe eines Handgestenklassifizierungsmodells möglichst effektiv diverse Handgesten in einer vordefinierten Bildregion erkannt und unterschieden werden. Ziel ist es ein System zu entwickeln, welches mit Hilfe von *State-of-the-Art*-CNN-Netzwerkarchitekturen sowohl genau als auch möglichst ressourcenschonend eingesetzt werden kann. So wird zusätzlich gewährleistet, dass für das Handgestenerkennungsmodell kein High-Tech-Equipment benötig wird.

5.1 Use-Case TU Wien Pilotfabrik

In der TU Wien Pilotfabrik wird aktuell ein praxisnaher Demonstrator eines Fan-Cowl-Werkzeuges gemeinsam mit Industrieunternehmen aufgebaut. Im dortigen Montageprozess werden Schicht für Schicht Carbonmatten aufgelegt und so sukzessive das Gehäuse eines Triebwerkes hergestellt. Der Demonstrator findet sich im Bereich industrieller Baustellenmontage, d. h. dass ein großes Produkt stationär montiert wird und das Montagepersonal, das Werkzeug sowie Baugruppen sich zum Produkt begeben. Zu diesem Zweck wurde ein dynamisches Projektionssystem entwickelt. Dieses dynamische Projektionssystem wurde mit der *State-of-the-Art*-Informationsbereitstellung via stationären Terminal in Bezug auf deren Gebrauchstauglichkeit verglichen. Dazu wurde ein Experiment am Demonstrator mit einem CFK-Auflege-Aufbau für Flugzeugkomponenten durchgeführt (Rupprecht et al., 2020). Der Aufbau für das dynamische Projektionssystem sowie des PC-Terminal-Experiment ist in Abbildung 15 dargestellt.

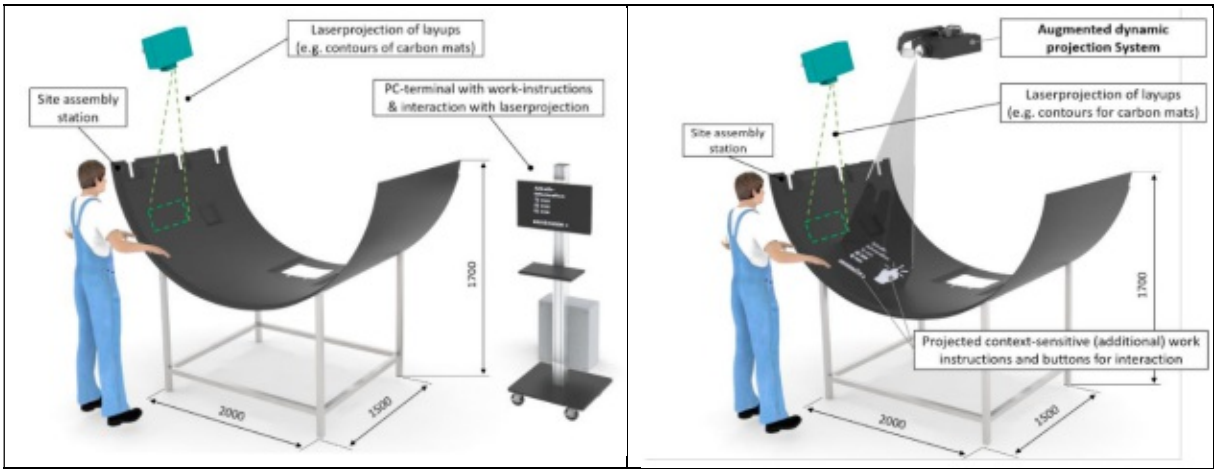


Abbildung 28: Informationsbereitstellung via stationären Terminal (links), dynamische Projektionssystem (rechts); (Rupprecht et al., 2020)

Das dynamische Projektionssystem wurde im Vergleich zur Informationsbereitstellung mittels PC-Terminal in Bezug auf deren Gebrauchstauglichkeit evaluiert und liefert

signifikante Verbesserungen. Verwendet wurden dabei zwei Fragebögen zur Gebrauchstauglichkeit. Zum einen die „Structured System Usability Scale (SUS)“ (Brooke et al., 1996) und zum anderen das „Technology Acceptance Model (TAM)“ (Davis et al., 1989). Insgesamt haben 16 Personen am Experiment teilgenommen. Zum Zeitpunkt des Experiments wurde die Informationsbereitstellung der Prozessschritte manuell am PC ausgelöst, da das automatische Auslösen der Schaltfläche noch nicht vollständig funktionsfähig war (Rupprecht et al., 2020).

Qualitative Erkenntnisse sowie der Gebrauchstauglichkeits-Wert des Experiments deuten darauf hin, mehr Forschung in Richtung eines freihändigen (ohne zusätzliche Devices oder Wearables) und kontextadaptiven Assistenzsystem zu betreiben (Rupprecht et al., 2020).

5.2 Konzeptioneller Aufbau des Interaktionssystems

An diesem Punkt knüpft der praktische Teil der vorliegenden Masterarbeit an und soll mittels eines Handgestenklassifizierungsmodells auf Basis eines CNNs möglichst effektiv verschiedene Handgesten unterscheiden. Dazu braucht es einen geeigneten Datensatz für das Training des ML-Systems. Die Auswahl geeigneter Datensätze wurde bereits mittels Nutzwertanalyse in Kapitel 4 durchgeführt. Diverse ergebnislose Tests führten zur Erkenntnis, dass es für den Use-Case sinnvoller ist einen neuen Datensatz zu erstellen.

Das vorgeschlagene Konzept sieht eine Klassifizierung von statischen Handgesten ausschließlich im dafür vorgesehen Bereich, der sogenannten *Region of Interest,* vor. Zusätzlich wird im linken unteren Bildschirmbereich das Menü für das User Interface dargestellt. Das Mockup (vgl. Abbildung 29) dient ausschließlich dazu, eine mögliche Menüführung für ein Informationssystem grafisch darzustellen. Das vorläufige Mockup ist zum Zeitpunkt dieser Arbeit technisch nicht mit der Gesteninteraktion verbunden und besitzt keine Funktionalität. Im Zentrum der vorliegenden Arbeit steht die Auswahl eines Datensatzes sowie das effektive Unterscheiden der Gestenkategorien mit Hilfe eines CNN-Modells. Das User-Interface-Design samt möglicher Arbeitsanweisungen ist nicht Teil der vorliegenden Arbeit.

Für das Gesteninteraktions-Konzept wird eine Always on-Erkennung konzipiert. Das bedeutet, dass eine Geste zu jedem Zeitpunkt in der ROI vorhergesagt wird. Dabei ist es notwendig, eine leere Klasse, welche das Fehlen einer Geste symbolisiert, darzustellen. Um bei diesem Konzept Fehlinteraktionen vermeiden zu können, werden zwei Maßnahmen getätigt:

- Eine ausgeführte Geste soll erst nach einer statischen Haltephase von ca. 2 Sekunden eine Interaktion mit dem Menü auslösen. Eine Anzeige oberhalb der ROI zeigt dem Nutzer welche Geste momentan ausgeführt wird.

- Eine Start- bzw. Stopp-Geste wird vordefiniert. Das Informationssystem wird erst nach Auslösen dieser Startgeste geöffnet. Die Auswahl einer geeigneten Startgeste kann das Risiko von fehlerhaften Interaktionen senken.

Die Interaktionsmöglichkeiten werden in der linken unteren Ecke dargestellt. In Abbildung wird das Interaktionsfeld aus der Ansicht von Oben exemplarisch dargestellt.

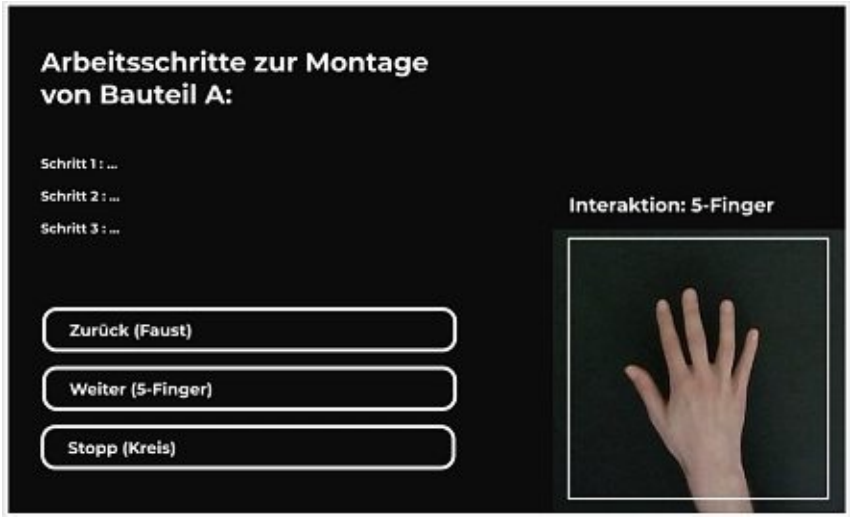


Abbildung 29: Konzept Mockup des Natural Interface

Die ROI soll zukünftig mit Hilfe des dynamischen Projektionssystems auf den Demonstrator projiziert werden. So wird der Interaktionsraum für den Arbeiter sichtbar gemacht. Das exemplarische Mockup in Abbildung 29 enthält folgende Bestandteile:

- *Region of Interest*, inklusive vom Benutzer ausgeführter Geste.
- Feedback zur erkannten Geste. Somit wird dem Benutzer stets ein Feedback zur ausgeführten Geste gegeben.
- Arbeitsanweisung in der linken oberen Projektionsfläche
- Möglichkeiten zur Interaktion auf dieser Menüebene (links unten)

Im exemplarischen Mockup sind drei Gesten mit einer Funktion auf dieser Menü-Ebene ausgestattet. Um eine höhere Flexibilität gewährleisten zu können, werden zunächst fünf verschiedene Gesten unterschieden. Der Datensatz und die zugehörigen Gestenkategorien werden in Kapitel 5.4 beschrieben.

5.3 Versuchsaufbau und verwendete Hardware-Komponenten

Im folgenden Kapitel werden der Versuchsaufbau sowie die verwendete Hardware beschrieben. Zum Zeitpunkt der Durchführung der vorliegenden Arbeit, ist ein Test am

Demonstrator nicht möglich. Alternativ wird ein ähnliches Szenario nachgebaut. Um ein reproduzierbares Ergebnis gewährleisten zu können, wird im Folgenden der Versuchsaufbau im Detail erläutert.

Der Versuchsaufbau ist in Abbildung 30 dargestellt. Der schwarze Hinter- bzw. Untergrund wird mittels Karton simuliert. Ein Projizieren der *Region of Interest* ist im nachgebauten Szenario nicht möglich. Die Interaktionsfläche wird für die Akteure auf dem einheitlichen Untergrund markiert.

Abbildung 30: Versuchsaufbau und verwendete Hardware

Um zu zeigen, dass die technologische Umsetzung kosteneffizient möglich ist, wird beim nachfolgenden Versuch auf Standardkomponenten bzw. -hardware zurückgegriffen. Zur Aufnahme der Gestenbilder sowie zum nachfolgenden Versuch wird eine handelsübliche Webcam der Marke *Jelly Comb* verwendet. Die Webcam wird in Kombination mit einem *Apple MacBook Air* (Version: *early 2015*) genutzt. Der verwendete Rechner hat eine 1,6 GHz Intel Core i5 Prozessor, 8 GB 1600 MHz DDR3 Arbeitsspeicher und eine Intel HD Graphics 6000 1536 MB Grafikkarte. Bei der Positionierung des Stativs bzw. der Ausrichtung des Kamerawinkels muss darauf geachtet werden die vorgegebene *Region of Interest* auf einem einheitlichen (in der vorliegenden Arbeit schwarzen) Hintergrund auszurichten, sowie die Zugänglichkeit für die Akteure bzw. Versuchspersonen zu gewährleisten. Da das vorliegende Kamerastativ keine Aufnahme für die verwendetet Webcam besitzt, wird diese mittels

Gummibandes stabil befestigt. Folgende Komponenten werden für den Versuch verwendet:

- Computerrechner *Apple Macbook Air (Version 2015)*
- Kamera *Jelly Comb 1080P HD Webcam*
- Kamerastativ
- Evtl. Befestigung für Kamera
- Evtl. Karton für einheitlich (schwarzen) Hintergrund

Im nächsten Kapitel wird die Zusammensetzung des Datensatzes vorgestellt.

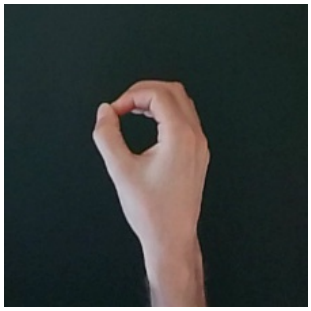
5.4 Zusammensetzung des Datensatzes

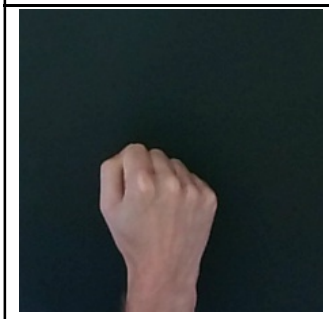
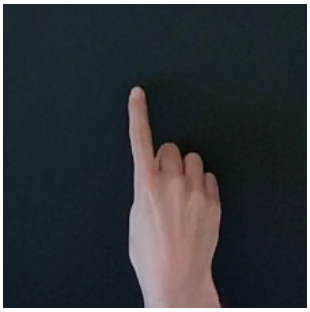
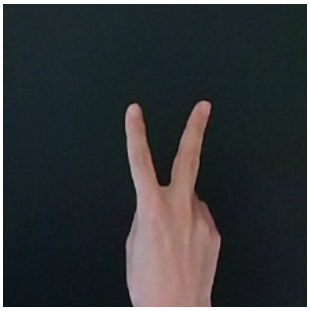
Bei der Auswahl des Gestensatzes gilt es sowohl die technischen Aspekte der Unterscheidbarkeit als auch Aspekte der ergonomischen Ausführbarkeit zu beachten. Um Fehlinteraktionen zu vermeiden, müssen die verwendeten Gesten eindeutig voneinander unterscheidbar sein. Bei der Gestaltung eines Gestensatzes ist eine ausführliche Dokumentation jeder einzelnen Geste wichtig. Laut DIN EN ISO 9241-960: 2018-01 sollte diese Dokumentation folgende Elemente enthalten:

- Die Bezeichnung der Geste
- Eine visuelle Beschreibung der Geste (z. B Bild, Animation oder Video)
- Eine Textbeschreibung zur Ausführung der Geste
- Eine Beschreibung des Zwecks der Geste

In nachfolgender Tabelle 9 werden die Gesten für die vorliegende Use Case-Interaktion beschrieben. Es handelt sich bei den Gesten um statische Gesten, die mit der rechten Hand ausgeführt werden. Aufgrund der Symmetrie der Gesten können im Verlauf der Modellierungsphase die Bilder auch spiegelverkehrt eingelesen werden, womit eine Interaktion mit der linken Hand auch möglich ist. Der Gestendatensatz enthält zudem eine leere Klasse, die das Fehlen einer Geste zu symbolisieren.

Tabelle 9: Gestendatensatz nach DIN EN ISO 9241-960: 2018-01

	**Bezeichnung:** Kreis-Geste **Beschreibung:** Die rechte Hand formt eine Kreis-Form mittels Daumen und einem beliebigen anderen Finger. **Zweck:** Die Geste kann als Start/Stopp-Geste interpretiert werden und dient dem Zweck das User-Interface zu starten und nach Gebrauch wieder zu schließen.

	**Bezeichnung:** keine Interaktion (Ruhemodus) **Beschreibung:** Es befindet sich keine der definierten Gesten im Interaktionsbereich. **Zweck:** Dient als Ruhemodus- Klasse, um einen interaktionsfreien gleichbleibenden Zustand des Systems zu gewährleisten.
	**Bezeichnung:** Faust **Beschreibung:** Die rechte Hand formt eine Faust. **Zweck:** Die Faust-Geste wird als „Schritt zurück"-Befehl in der Menüführung verwendet.
	**Bezeichnung:** 5 Finger-Geste (offene Hand) **Beschreibung:** Die Handfläche der rechten Hand ist offen. Alle Finger werden (leicht) voneinander gespreizt. **Zweck:** Dient als „Bestätigung" – Befehl, welcher die vorgeschlagene Aktion in der Menüführung bestätigt.
	**Bezeichnung:** 1-Finger-Geste **Beschreibung:** Bei der 1-Finger-Geste ist der Zeigefinger der rechten Hand gestreckt. **Zweck:** Kann als optionale Geste in der Menüführung eingesetzt werden und symbolisiert die erste Auswahlmöglichkeit.
	**Bezeichnung:** 2-Finger-Geste **Beschreibung:** Die rechte Hand formt eine ein V mit Zeige- und Mittelfinger. **Zweck:** Kann als optionale Geste in der Menüführung eingesetzt werden und symbolisiert die zweite Auswahlmöglichkeit.

Der Gestendatensatz stellt die Basis der gestischen Interaktion dar. Prinzipiell kann der Zweck der Gesten jederzeit geändert werden. Sollte sich im Laufe der Benutzung herausstellen, dass sich für eine gewünschte Funktion eine der anderen Gesten besser eignet, können jederzeit Änderungen im Gestendatensatz vorgenommen werden.

In diesem Kapitel wurden die Gestenklassen für den Use-Case Anwendung vorgestellt und deren Funktion erläutert. Im nächsten Kapitel wir die Modellierung der Multiklassen-Klassifikationsaufgabe beschrieben.

6 Modellierung mit Tensorflow und Keras

Bei dem vorliegenden Anwendungsfall handelt es sich um eine Multiklassen-Klassifikationsaufgabe bei der zwischen sechs Klassen (Kategorien) unterschieden wird. Der Ablauf der Modellierung folgt der Methodik und den Schritten Kapitel 3.2.

6.1 Datenbeschaffung und -Bereinigung

In diesem Kapitel wird zunächst auf die verwendete Software zur Datenbeschaffung eingegangen und im Anschluss der Vorgang der Bildaufnahme beschrieben.

Zur Aufnahme der Bilder für den Datensatz wird die freie Programmbibliothek *OpenCV* verwendet. *OpenCV* steht für *Open Source Computer Vision Library* und ist eine kostenlos zugängliche Computer Vision und *Machine Learning*-Software-Bibliothek Ursprünglich wurde *OpenCV* in der Programmiersprache C++ geschrieben, bietet jedoch zudem Python, Java und MATLAB Interfaces an.

Die Programmiersprache *Python* erfreut sich großer Beliebtheit in der *Machine Learning-Community* und wird für vorliegende Aufgabenstellung genutzt. Aufbauend auf Python existieren spezielle Libraries für DL-Anwendungen. Die bekanntesten sind *Pytorch* und *Tensorflow*. In der vorliegenden Arbeit werden *Tensorflow* und *Keras* genutzt. *Tensorflow* ist eine Open Source Plattform für Machine Learning-Anwendungen. Darauf aufbauend wird Keras als *high level*-Programmierschnittstelle (engl. *application programming interface* kurz API) verwendet. *Keras* ist eine Open Source Deep Learning Bibliothek, die seit 2015 auf dem Markt ist. Sie wurde mit dem Schwerpunkt des schnellen Experimentierens, entwickelt. Der Fokus liegt dabei auf modernen DL-Anwendungen.

Zur Aufnahme der Trainingsbilder wird ein Python-Script erstellt, welches im Anhang der vorliegenden Arbeit beigefügt wird. Als Programmierumgebung für die Bildaufnahme wird *Visual Studio Code 2* verwendet. Die Open Source Library Open CV bietet die Möglichkeit mittels Befehl **cv2.VideoCapture(1)** auf die verbundene Webcam zuzugreifen. Mit Hilfe einer Schleife kann so ein kontinuierliche Videoaufnahme erzeugt werden. Die Aufnahmen der verschiedenen Gesten aus dem Echtzeit-Video werden aus der *Region of Interest (ROI)* extrahiert. Die ROI gibt hierbei einen virtuellen Rahmen vor, in dem die statische Handgeste ausgeführt wird. Mit der Befehlseingabe (Leertaste) wird ein Bildausschnitt (ROI) aus dem Live-Video extrahiert. Die Befehlseingabe wird vom jeweiligen Akteur durchgeführt. Für die Trainingsbilder bzw. Validierungsbilder sollten die Beispielgesten in ihrer Ausführung variieren. Dabei ist jedoch stets auf die Eindeutigkeit sowie richtige Ausführung der Geste zu achten.

Um den Trainingsprozess zu vereinfachen, wird jedes Gestenbild automatisiert in ein Verzeichnis gespeichert, welches den Namen der Geste im Ordnernamen enthält. Dieser Ordnername dient später als *Label* für das Modell. Außerdem wird das Bild, bevor es gespeichert wird, in das nötige Format gebracht. Der verwendete Trainings-Algorithmus verlangt eine Größe von 224x224 Pixel sowie den Farbraum RGB. Als Format wird .jpg gewählt. Die Beschriftung der Gestenbilder enthält die Codierung nach Tabelle 10. Die vollständige Beschriftung der Gestenbilder sowie der Ordnerstruktur ist in Abbildung 31 dargestellt.

Tabelle 10: Gesten-Kodierung für die Modellierung

Zahlen Kodierung:	Englische Bezeichnung:	Gestenbezeichnung:
0_	circle	Kreis-Geste
1_	empty	Leer
2_	fist	Faust-Geste
3_	five	5-Finger-Geste
4_	one	1-Finger-Geste
5_	two	2-Finger-Geste

An dieser Stelle wird der gesamte Datensatz auf Ausreißer, wie etwa unscharfe Aufnahmen oder schlampig ausgeführte Gesten geprüft. Solche Ausreißer werden entfernt und mit aussagekräftigen Bildern ersetzt. Der Verzeichnis-Baum wird zum Zweck einer guten Übersicht und Benutzerfreundlichkeit, wie in Abbildung 60 dargestellt, aufgebaut. Der Hauptordner wird in zwei Unterordner für Trainings- bzw. Validierungsbilder unterteilt. In der nächsten Unterebene sind Ordner mit den Gestenkategorien enthalten. Schließlich werden in den Gestenkategorien -Ordnern die Bilder im .jpg Format gespeichert. Um die Bilder eindeutig zuordnen zu können, enthält der Dateiname die Akteur-Kennzeichnung (persA oder persB) und eine fortlaufende Nummerierung.

```
main_directory/
        ...images_train/
        ...images_val/
                ....0_circle/

                ....1_empty/

                ....2_fist/

                ....3_five/

                ....4_one/

                ....5_two/
                        ......5_persA_gest1.jpg

                        ......5_persA_gest2.jpg
```

Abbildung 31: Ordnerstruktur für die Modellierung

Der Datensatz enthält insgesamt 2400 Beispielgesten. Die Bilder sind hierbei in 1920 Trainingsbildern und 480 Validierungsbildern gegliedert. Die Aufteilung entspricht der üblichen Verteilung die etwa 20-30 % der Gesamtdaten als Validierungsdaten vorsieht (Goodfellow et al., 2016). Die Gesten werden von einer weiblichen Akteurin sowie einem männlichen Akteur zu gleichen Teilen ausgeführt. Pro Gestenkategorie ergeben sich somit 320 Bilder. Die durchschnittliche Anzahl von Bildern pro Gestenkategorien aus der Handgestendatenbank-Analyse (vgl. Kapitel 3.2) entspricht 280 Bildern. Der Hintergrund ist einheitlich schwarz. Eine Zusammenfassung der Eckpunkte des Datensatzes gibt Tabelle 11.

Tabelle 11: Überblick zum Trainingsdatensatz

Anzahl Akteure	Gestenkategorien	Trainingsbilder	Validierungsbilder	Gestenbilder Gesamt
2	6	1920	480	2400

Nachdem alle Beispielgesten in der Ordnerstruktur abgelegt wurden und von Ausreißern bereinigt wurden, kann im nächsten Kapitel mit der Modellierung zur Handgestenklassifikation begonnen werden.

6.2 Trainieren des ML-Systems mit Tensorflow und Keras

DL-Modelle werden auf Grafikprozessoren (engl. *graphics processing unit* kurz *GPU*) trainiert. Der Trainings-Prozess für DL-Modelle erfordert eine Vielzahl von Rechenschritten und ist ein sehr rechenintensiver Vorgang. Für die vorliegende Arbeit wird *Googles* Cloud-basierte Plattform *Google Colab* verwendet. Damit kann kostenlos auf GPU-Rechenleistung zugegriffen werden. Python Code kann in *Google Colab* direkt im Browser geschrieben und ausgeführt werden. Mit Hilfe des *Filehosting*-Dienstes *Google Drive* können die Bilder der Beispielgesten abgespeichert und mittels *Colab* abgerufen werden.

Zunächst werden die, nach der Struktur aus Kapitel 6.1 abgelegten, Daten auf *Google Drive* hochgeladen. Anschließend kann mit dem Modellierungsprozess begonnen werden. Die wesentlichen Schritte des Modellierungsprozesses sind im Flussdiagramm in Abbildung 32 dargestellt.

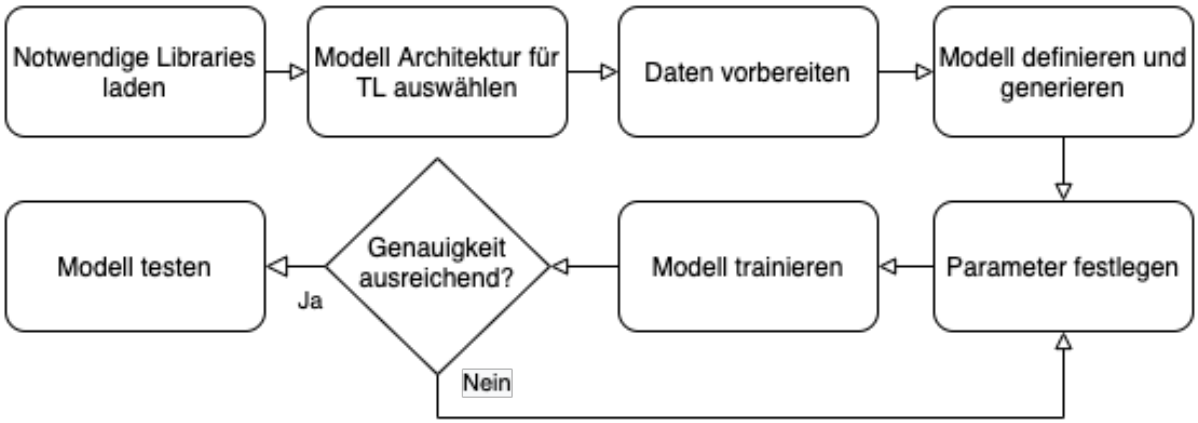


Abbildung 32: Flussdiagramm zur Modellierung mit Tensorflow und Keras

Laden der notwendigen Bibliotheken:

Zunächst wird *Google Drive* mit *Google Colab* verbunden, um Zugang zu den gespeicherten Beispielgesten zu erlangen. Als nächster Schritt wird *Keras* mit Hilfe des Package Managers installiert (**! pip install keras**). Anschließend werden die notwendigen *Libararies* sowie verwendeten *Models* und *Layer* geladen. Dazu zählen *OpenCv* (**cv2**) für *Computer Vision*- Manipulationen. Die *Numpy*- Bibliothek zur einfachen und schnellen Verarbeitung von Vektoren und Matrizen. Zudem wird die Abkürzung *tf* für *Tensorflow* eingeführt.

Tabelle 12: Code zum Laden der notwendigen Programmbibliotheken

```
import tensorflow as tf

from tensorflow.keras.layers import Input, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Die Schichten (engl. *Layer*) sind die Grundbausteine von neuronalen Netzen in *Keras*. Die Model API von Keras wird importiert und schließlich werden Dienstprogramme zur Datenvorbereitung geladen.

An dieser Stelle wird ein vortrainiertes Modell inklusive Gewichte geladen. Das Modell wurde auf dem *ImageNet*-Datensatz trainiert. *Keras* führt die verschiedenen Modelle unter der Bezeichnung *Keras Applications*. *Keras* bietet 26 verschiedene Modell-Architekturen an (Stand 25.11.20).

Auswahl der passenden CNN-Architektur:

Für die vorliegende Aufgabenstellung der Gestenklassifizierung ist sowohl Genauigkeit als auch Geschwindigkeit entscheidend. Eine Netzwerk-Architektur, die schnell und effizient Vorhersagen treffen kann, wird benötigt. Da die Gestenerkennung in Echtzeit erfolgt, liegt eine kleinere Leichtbau-Modell-Architektur nahe.

Für die vorliegende Problemstellung wird *ModelNetV2* verwendet. *ModelNetV2* ist eine neuronale Netzwerkarchitektur, die für mobile Applikationen und ressourcenschonende Anwendungen entwickelt wurde. Für das anschließende Testverfahren ist dies von entscheidender Bedeutung, da schnelle und genaue Vorhersagen notwendig sind. In Tabelle 13 wird der Code zum Laden der MobileNetV2-Modell Architektur dargestellt.

Tabelle 13: Code zum Laden der DL-Modell-Architektur

```
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
```

Datenvorbereitung:

Die Keras-Bibliothek bietet die Möglichkeit über die Klasse **ImageDataGenerator** die Datenvorbereitung vorzunehmen. Jede Netzwerk-Architektur verlangt einen Datenvorverarbeitungsschritt bevor Input-Bilder an das Modell weitergegeben werden.

Zudem lässt sich mit Hilfe der Klasse **ImageDataGenerator**- Klasse die Technik *Data Augmentation* durchführen.

Bei *Data Augmentation* handelt es sich um eine Technik, mit der die Größe eines Trainingsdatensatzes künstlich aufgebläht werden kann, indem modifizierte Versionen von Bildern im Datensatz erstellt werden. Ziel ist es Overfitting zu vermeiden (vgl. Kapitel 2.3). Data Augmentation setzt beim Problem des Overfittings direkt beim Trainingsdatensatz an. Es werden transformierte Bilder im Trainingsdatensatz erzeugt. Diese gehören zur gleichen Klasse wie das Originalbild (Shorten & Khoshgoftaar, 2019). Die simplen Transformationen wie etwa das Zuschneiden, Rotieren oder Spiegeln an einer der Hauptachsen können effektiv eingesetzt werden, um die Genauigkeit des Modells zu erhöhen (Perez & Wang, 2017). Der interessierte Leser wird auf das Survey-Paper von Shorten und Khoshgoftaar (2019) verwiesen.

Keras bietet für *Data Augmentation* eine simple Implementierung an. Bei jeder Transformation muss berücksichtigt werden, ob selbige zu einem Klassenwechsel führen kann. *Data Augmentation* wird typischerweise nur auf den Trainingsdatensatz angewendet, nicht aber auf den Validierungsdatensatz. Folgende Transformationen werden beim vorliegenden Modell angewendet:

- **horizontal_flip:** erlaubt das spiegeln an der y-Achse. Das Modell wird somit auch auf Gesten trainiert, die im Interaktionsfeld mit der linken Hand ausgeführt werden.
- **rotation_range:** gibt an um wie viel Grad das Originalbild maximal nach links bzw. rechts verdreht wird.
- **zoom_range:** dieser Parameter legt den Maximalwert (1+ zoom_range) und den Minimalwert (1-zoom_range) für eine Vergrößerungs- bzw. Verkleinerungsoperation fest.
- **brightness_range**: mit Hilfe dieses Parameters können Lichtverhältnisse künstlich verändern werden, um das Modell stabiler auf wechselnde Lichteinflüsse zu machen.

In Abbildung 33 sind augmentierte Bilder aus dem Trainingsdatensatz abgebildet. Tabelle 14 zeigt den Code zur Implementierung von Data Augmentation.

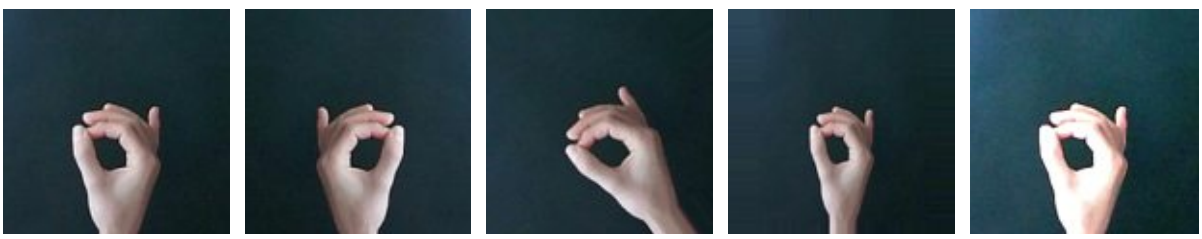


Abbildung 33: Data Augmentation von links nach rechts; Original, horizontal_flip, rotation_range, zoom_range, brightness_range

Tabelle 14: Code zur Data Augmentation Implementation

```
# Erstellen von Keras-ImageDataGenerator und Datenvorverarbeitung

# Trainingsset
gen_train = ImageDataGenerator(
      rotation_range=45,
      horizontal_flip=True,
      zoom_range= 0.5,
      brightness_range= [0.3,2.0],
      preprocessing_function=tf.keras.applications.mobilenet_v2.preproces
      s_input)

# Validierungsset
gen_val = ImageDataGenerator(
      preprocessing_function=tf.keras.applications.mobilenet_v2.preproces
      s_input)
```

Die **flow_from_directory**-Funktion erstellt einen für *Tensorflow* verwertbaren Datensatz auf Basis der Ordnerstruktur, die auf Google Drive abgelegt wurde. Die Funktion erzeugt Datenstapel sog. *Batches*, zusammengesetzt aus zufälligen Input-Bildern aus den verschiedenen Klassen, die dem Modell beim Trainingsprozess zugeführt werden. Die Größe der Datenstapel (**batch_size**) und die Bildgröße (**img_size**) werden in diesem Schritt festgelegt. Zudem wird der Pfad zu den Beispielbildern angeben. Es ist darauf zu achten, dass das verwendete Bildformat (beim vorliegenden Datensatz .jpg) von *Tensorflow* unterstützt wird.

Tabelle 15: Code für die Datentransfer-Einstellungen

```
#Grundeinstellungen des Datentransfers
img_size = [224, 224]
batch_size = 16

train_batches = gen_train.flow_from_directory(
    directory=train_path,
    target_size=img_size,
    batch_size=batch_size)

valid_batches = gen_val.flow_from_directory(
    directory=val_path,
    target_size=img_size,
    batch_size=batch_size)
```

Der Validierungs-Datensatz enthält 480 Beispielbilder und somit 20 % der gesamten Daten, die für das Training des Modells vorgesehen sind. Dieser 80/20-Split ist bei ML-Aufgaben in der Literatur üblich (Goodfellow et al., 2016, S. 121).

Modelldefinition:

In diesem Schritt wird das Modell definiert und die Klassenanzahl auf sechs unterschiedliche Klassen begrenzt. Dabei wird zunächst als Basis-Modell *MobileNetV2* inklusive der vortrainierten Gewichte (auf dem *Imagenet*-Datensatz) importiert. Darauf aufbauend wird ein *Global Average Pooling- Layer* (Lin et al., 2013) angewandt. Dieser ermittelt den Durchschnitt jeder einzelnen Feature Map. Anschließend wird die Dropout-Regularisierungstechnik auf das Modell angewandt. Das Ziel dieser Technik besteht darin, die Generalisierung des Neuronalen Netzes zu verbessern. Hierbei wird ein Teil der Einheiten und Verbindungen mit einer festgelegten Wahrscheinlichkeit zufällig übersprungen. Diese Technik sorgt dafür, dass sich der Algorithmus den Trainingsdaten nicht zu sehr anpasst und besser auf unbekannte Testdaten generalisiert (Hinton et al., 2012). Die Dropout-Rate liegt bei CNNs normalerweise bei 40-50% (Géron, 2019, S. 365). Auf dem Basismodell aufbauend werden neue trainierbare Schichten hinzugefügt. Dadurch sollen die bereits gelernten Features auf die neuen Trainingsdaten angewandt werden. Im nächsten Schritt wird auch der Output-Layer festgelegt. Hierzu wird ein *Dense*-Layer verwendet, auf den die Aktivierungsfunktion *Softmax* angewendet wird. Diese gibt Auskunft über kategorische Wahrscheinlichkeiten, sofern sie auf den letzten Layer angewendet wird. Die Elemente des Output-Vektors sind im Bereich zwischen 0 und 1 und summieren sich zu 1. Schließlich werden in der Modell- Klasse das Basismodell mit der Output-Schicht zusammengeführt und ein Objekt mit Trainings- und Inferenzmerkmalen erzeugt.

Tabelle 16: Code zur Modelldefinition

```
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import GlobalAveragePooling2D

#Klassenanzahl:
n_classes = 6

#Basis Model und Gewichte importieren
base_model = MobileNetV2(
    weights='imagenet',
    include_top=False
)
model = GlobalAveragePooling2D()(base_model.output)
model = Dropout(0.5)(model) #Dropout 50%

# Dense Layer als Outputschicht festlegen
output_layer = Dense(n_classes, activation='softmax')(model)

# Model erstellen
model = Model(base_model.input, output_layer)

# Fixieren ("einfrieren")der vortrainierten Schichten
for layer in base_model.layers:
  layer.trainable = False
```

Die bereits vortrainierten Schichten werden fixiert („eingefroren“), um die darin enthaltenen Informationen nicht zu zerstören. Sie werden beim folgenden Trainingsprozess nicht verändert werden. Das bedeutet, es werden zunächst nur die neuen Schichten des Modells trainiert.

Modellparameter festlegen

Die Datenvorbereitung mittels *Keras* bzw. *Tensorflow* ist abgeschlossen. Somit kann an diesem Punkt mit dem eigentlichen Training begonnen werden. Das erstellte Modell muss an dieser Stelle kompiliert werden (siehe Tabelle 14). Dazu werden der *Optimizer,* die *Loss-function* und die *Metric* definiert. Als benutzerfreundlicher und schneller *Optimizer* wird *Adam* gewählt. Der Hyperparameter *Learning Rate* wird beim Keras-Standardwert (lr = 0.001) belassen. Bei einer Multiklassen-Klassifikationsaufgabe, bei der jedes Bild genau einer Klasse zugeordnet werden soll, wird für die *Loss*-Funktion *categorical crossentropy* genutzt. Für Klassifizierungsprobleme kann die Genauigkeit (*accuracy*-Metrik) als Qualitätsmerkmal für ein Modell verwendet. Diese Genauigkeit wird nach jeder Epoche ausgewertet.

Modell trainieren

Als nächster Schritt kann mit dem eigentlichen Training begonnen werden. Dafür wird in *Keras* die **fit()** Methode verwendet. Hierfür werden die Trainingsdaten, die Validierungsdaten und die Anzahl der Epochen benötigt. Dabei bezeichnet eine Epoche einen vollständigen Durchlauf durch die Eingabedaten (Trainings- und Validierungsdaten).

Tabelle 17: Code zur Modellkomplierung und zum Training

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history = model.fit(
    train_batches,
    epochs=5,
    validation_data=valid_batches
)
```

Die Genauigkeit der Validierungsdaten liegt nach 5 Epochen bereits bei ca. 97%. Um die Genauigkeit weiter zu erhöhen, wird ein Finetuning des Modells vorgenommen.

Die zuvor fixierten Layer werden in den Zustand „trainierbar“ versetzt und das Finetuning durchgeführt. Wichtig ist hierbei eine deutliche Reduzierung der *Learning Rate* des Adam Optimizer (lr= 0.00001), da sonst die bereits gelernten Gewichte zerstört werden können. Zudem werden die Trainingsepochen auf 40 erhöht.

Tabelle 18: Code zum Fine-Tuning

```
# Fixierung lösen und alle Layer auf "trainierbar" setzen

for layer in base_model.layers:
  layer.trainable = True

# Fine-Tuning des Modells
model.compile(
    optimizer=tf.keras.optimizers.Adam(lr=0.00001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)


history = model.fit(
    train_batches_aug,
    epochs=40,
    validation_data=valid_batches
)
```

Nach 40 Epochen erreicht das Modell 99,61 Prozent Genauigkeit auf den Validierungsdaten. Die Ergebnisse werden in Kapitel 8 näher erläutert.

Das Modell erreicht mit den eingestellten Hyperparametern eine zufriedenstellende Genauigkeit. Zu diesem Zeitpunkt kann mit dem Test des Modells fortgefahren werden. Mit dem Befehl **model.save** kann das Modell gespeichert werden und später in einem Anwendungscode, wie etwa dem Echtzeit-Videotest wiederverwendet werden.

Im nächsten Kapitel wird der Versuchsablauf zur Aufnahme von Testbildern erläutert. Zudem wird das Versuchsprotokoll zum Test der Genauigkeit von Echtzeitgesten vorgestellt.

7 Versuchsablauf und -durchführung

Der erste Teil des Versuchs zielt darauf ab, einen Datensatz zum Test der Genauigkeit nach den Performance-Metriken gemäß Kapitel 3.2 zu erstellen. Im zweiten Teil des Versuchs soll die Genauigkeit der Modellvorhersagen in Echtzeit getestet werden. Diese ist von entscheidender Bedeutung für die weitere Entwicklung eines Informationssystems auf Basis der vorgeschlagenen Gesten.

Zur Durchführung des Versuchs werden drei Versuchspersonen (1 weiblich, 2 männlich) zur Aufnahme herangezogen. Um zuverlässige Ergebnisse zu erhalten, waren die Versuchspersonen nicht bei der Aufnahme der Trainings- bzw. Validierungsbilder involviert. Die Versuchspersonen sind einverstanden, Bilder ihrer Hände zum Zweck dieser Arbeit zur Verfügung zu stellen.

Der Versuchsaufbau zur Generierung der Testdaten wird wie in Kapitel 5.4 beschrieben vorgenommen. Die Datenbeschaffung zur Modellierung und der Versuch finden nicht am selben Tag bzw. derselben Tageszeit statt. Ein gleichbleibender Lichteinfluss auf Trainingsbilder bzw. Test des Modells kann nicht gewährleistet werden. Der Versuch umfasst die Aufnahme von Bildern mit statischen Handgesten sowie einen Test des Vorhersagemodells in Echtzeit. Für den ersten Teil des Experimentes werden jeweils 15 Gestenbilder von jeder Gestenkategorie von drei Versuchspersonen extrahiert.

Zu Beginn des Versuchs werden die Versuchspersonen über die fünf verschiedenen Interaktionsgesten aufgeklärt und es wird die Dokumentation des Gestendatensatz zur Durchsicht gereicht.

Beim ersten Teil des Versuchs werden die Probanden vom Versuchsleiter dazu angeleitet, die Geste im markierten Bereich (ROI) durchzuführen. Als zusätzliche Orientierung wird die Echtzeitvideoübertragung des Interaktionsfelds am Laptop vor dem Probanden dargestellt. Die Probanden führen den Versuch im Stehen durch, damit verkürzt sich der vertikale Abstand zur Kamera. Der vertikale Abstand zur Kamera beim Echtzeit-Versuch beträgt ca. 75 cm (+/- 5 cm) und variiert zwischen den einzelnen Gestenbildern. 15 Bilder werden pro Gestenkategorie ausgeführt. Nach jeder Geste wird der Proband angewiesen, eine Ruheposition außerhalb des Interaktionsfeld einzunehmen, um mehr Variation in den Test-Datensatz zu bringen. Der Auslöser (Leertaste) wird vom Versuchsleiter betätigt. Die Bilder werden im Anschluss an das Modell weitergegeben, welches Vorhersagen zur Klassenzugehörigkeit trifft. Die Ergebnisse sind im Ergebnis-Kapitel dargestellt.

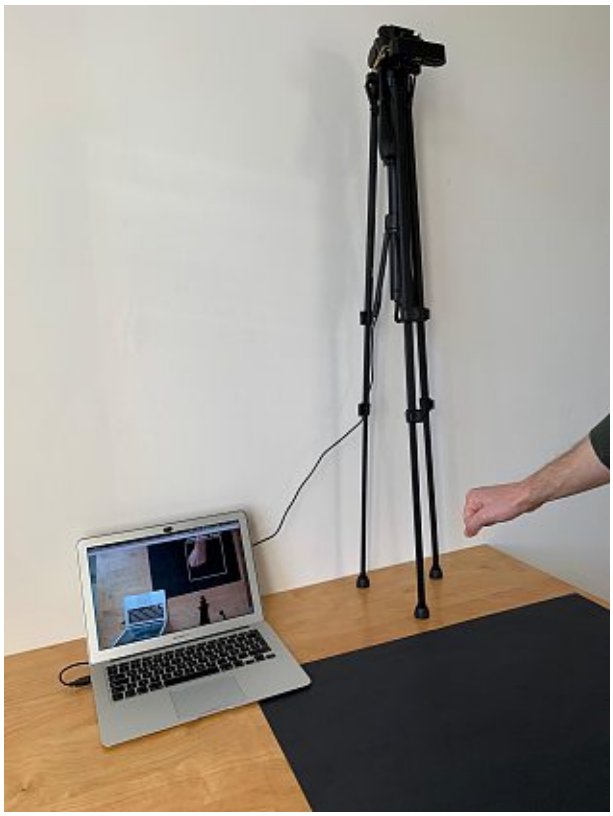


Abbildung 34: Versuchsaufbau zum Test des Modells

Die Probanden konnten sich im ersten Teil des Versuches mit den Gesten auseinandersetzen und die richtige Ausführung bereits üben. Im zweiten Teil wird die Vorhersage direkt am Bildschirm in Echtzeit ausgegeben. Das Szenario simuliert eine unmittelbare Interaktion, wobei wie bereits zuvor erläutert das User Interface keine Funktionalität aufweist. Der Versuchsleiter gibt die Geste laut Versuchsprotokoll vor. Daraufhin wird diese wird vom Probanden in der ROI ausgeführt. Hierbei wird eine Verweildauer von ca. 2 Sekunden eingehalten. Nach der Verweildauer wird die tatsächlich vorgegebene Geste mit der vorhergesagten Geste des Modells verglichen. Der Versuchsleiter trägt das Ergebnis in das Versuchsprotokoll ein (siehe Anhang). Für die Aufnahme der Test-Gestenbilder wird ein ähnlicher Python-Code wie für die Datenbeschaffung verwendet. Der Code für den Test der Echtzeit-Interaktion ist im Anhang angegeben. Die Ergebnisse des Versuchs wurden im nachstehenden Kapitel aufbereitet.

Aufbau des Versuchsprotokolls:

Das Versuchsprotokoll dient der einheitlichen Durchführung des Versuchs. Die Vorgehensweise ist für jede Testperson gleich. Der gleichbleibende Ablauf erleichtert die Reproduzierbarkeit des Versuchs. Die Reihenfolge der durchzuführenden Gesten wird per Zufallsgenerator festgelegt. Jede Geste wird jedoch gleich oft (10 Wiederholungen) ausgeführt. Außerdem wird keine Geste zweimal in Folge ausgeführt. Die Spalte Geste-Vorgabe gibt die zufällige Geste vor. Die Spalte Geste-

Vorhersage gibt die Modellvorhersage zur Klassenzugehörigkeit nach ca. zwei Sekunden an. Falls diese korrekt ist wird ein Haken gesetzt. Bei falscher Klassifikation wird ein x und die falsche Klasse vermerkt. Die Klasse „Leer“ wird nicht angeführt, da sie automatisch in jeder Interaktionspause eintritt. Während des Experiments wurde bei den Interaktionspausen stets die richtige Klasse („Leer“) vom Modell vorhergesagt. Ein Auszug aus dem Versuchsprotokoll ist in Tabelle 16 dargestellt.

Tabelle 19: Auszug Versuchsprotokoll

		Geste-Vorhersage		
Nr.	Geste-Vorgabe	Testperson 1	Testperson 2	Testperson 3
1	5-Finger	✓	✓	✓
2	Kreis	× (Faust)	✓	✓
3	1-Finger	✓	✓	✓

8 Auswertung der Versuchsergebnisse

In Kapitel 8 werden die Teil- Ergebnisse des Versuchs präsentiert. Zunächst wird dabei auf die Modellierungsergebnisse auf den Trainingsdaten eingegangen, um schließlich auf die Vorhersagen des Modells auf die Testdaten einzugehen.

Nach dem ersten Training (5 Epochen) auf den Eingabebildern konnte das Modell eine Genauigkeit von ca. 97 % erreichen. Anschließend wurde ein Fine-Tuning über 40 Epochen vorgenommen. Nach 40 Epochen erreicht das Modell schließlich 99,61 % Genauigkeit auf den Validierungsdaten.

Die *Confusion Matrix* gibt bei der vorliegenden Multiklassen-Klassifikationsaufgabe Auskunft darüber, welche Klassen richtig bzw. falsch klassifiziert wurden. Der Testdatensatz enthält für jede Klasse die gleiche Anzahl an Testdaten (45 Bilder). In Summe ergibt dies 270 Testbilder. Die Hauptdiagonale gibt die Anzahl der richtig klassifizierten Einheiten pro Klasse an.

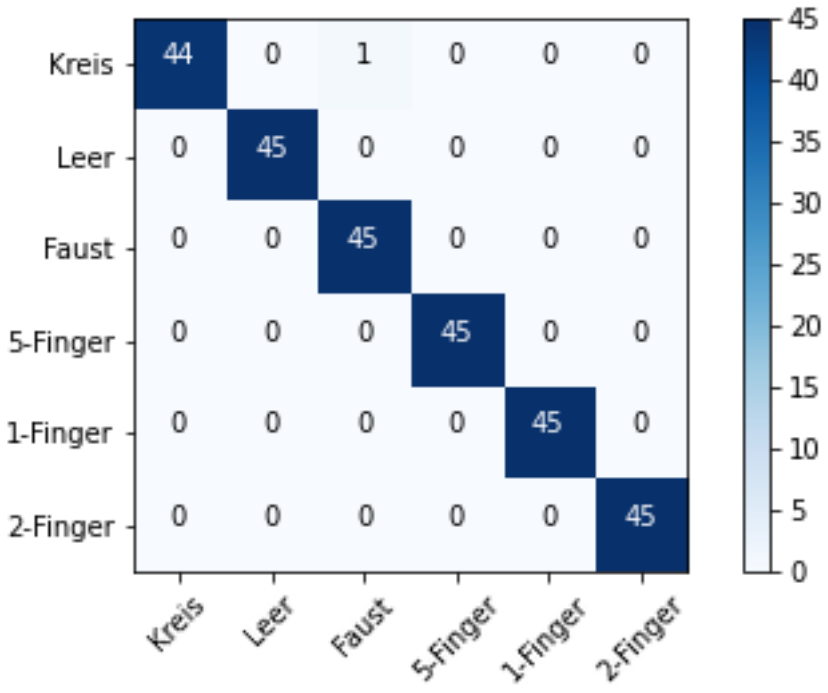


Abbildung 35: Confusion Matrix Testdaten

Die durchschnittliche Genauigkeit (engl. *average accuracy*) wird als Mittelwert der Hauptdiagonale berechnet. Von den vorliegenden Testbildern wurden mit Hilfe des in Kapitel 4 entwickelten Modells 269 von 270 richtig klassifiziert. Jede der einzelnen Klassen erreichte eine *Accuracy* von 100% bis auf die Kreis-Geste-Klasse (97,778 %) Unter Anwendung von *Macro-Averaging* ergibt dies eine Gesamtgenauigkeit von 99,63%. Die weiteren Leistungsmetriken berechnet nach den Formeln in Kapitel 3 sind in Tabelle 20 am Ende dieses Kapitels aufgelistet.

In der zweiten Phase des Versuchs wurde das Modell unter Echtzeitbedingungen getestet. Das Versuchsprotokoll (siehe Anhang) gibt dabei die Gestenreihenfolge vor. Insgesamt wurden 150 Interaktionen bewertet. Das Ergebnis ist in der Confusion Matrix in Abbildung 36 dargestellt.

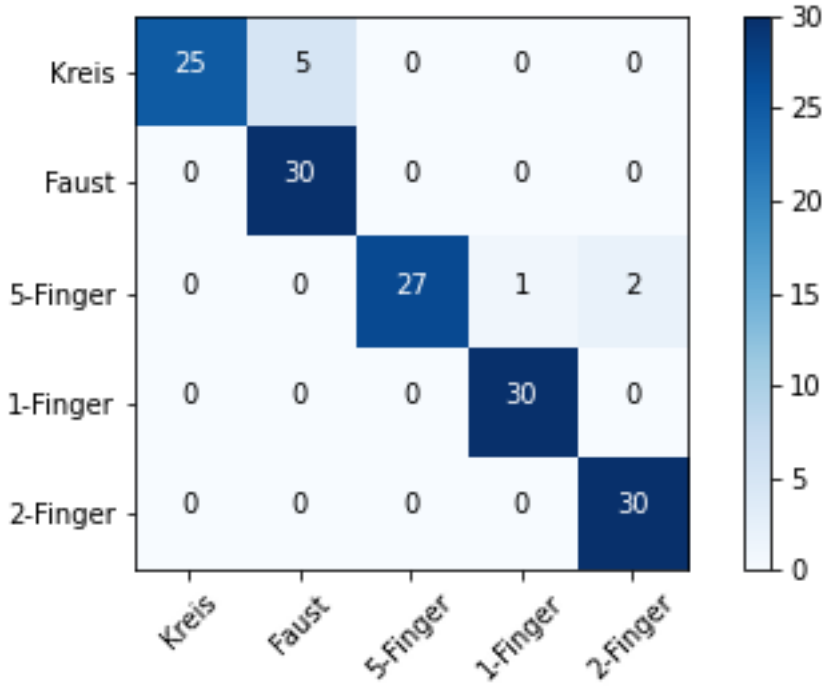


Abbildung 36: Confusion Matrix Echtzeitdaten

Beim Versuch in Echtzeit wurde eine durchschnittliche Genauigkeit von 94,67% erzielt wobei die Kategorie „leer“ vernachlässigt wurde. Es konnte beobachtet werden, dass diese bei der Ruheposition zwischen den Gesten stets richtig erkannt wurde. Die Confusion Matrix zeigt, dass das System die meisten Schwierigkeiten mit der Unterscheidung zwischen Kreis-Geste und Faust-Geste hatte. Fünf der acht falsch interpretierten Gesten (62,5%) sind auf diese Klassenunterscheidung zurückzuführen.

Die Unterscheidung zwischen Kreis und Faust bereitet dem System die größten Schwierigkeiten. 62,5% aller Fehler im Live Test sind auf falsch klassifizierte Kreis-Gesten zurückzuführen. Nur einer von acht Fehlern wurde in der zweiten Hälfte (nach 25 Gesteninteraktionen) des Echtzeitversuchs festgestellt. Diese könnte auf einen Lern- bzw. Anpassungsprozess der Testpersonen zurückzuführen sein.

Tabelle 20 gibt die aus der Confusion Matrix abgeleiteten Leistungsmetriken Average Accuracy, Average Precision, Average Recall, F1-Score an. Ermittelt wurden die Werte mit dem *Macro-Averaging*- Verfahren.

Tabelle 20: Performance Metriken

	Avg. Accuracy	**Avg. Precision**	**Avg. Recall**	**F1-Score**
Testdatensatz	99,63%	99,64%	99,63%	99,63%
Echtzeitdaten	94,67%	95,25%	94,67%	94,96%

9 Diskussion und Ausblick

In diesem Kapitel werden die Erkenntnisse aus der Handgestendatensatz-Analyse sowie die Ergebnisse aus dem Use-Case Versuch zusammengefasst und diskutiert. Einschränkungen der Ansätze und Ergebnisse werden aufgezeigt und abschließend mögliche nächste Schritte vorgeschlagen.

9.1 Resultate in Bezug auf die Forschungsfragen

In diesem Kapitel werden die Forschungsfragen aufgegriffen und beantworten.

„Welche Datensätze eignen sich zum Trainieren von Computer Vision-Gestenerkennung für Hand und Armgesten und wie können diese kategorisiert werden?“

Die recherchierten Datensätze sind sehr unterschiedlich in ihrer Ausprägung somit lässt sich die Forschungsfrage nicht eindeutig beantworten. Die Tabelle in Kapitel 4 gibt einen Überblick über vorhandene Datensätze, erhebt jedoch keinen Anspruch auf Vollständigkeit. Mit der zunehmenden Anzahl an Daten und der erleichterten Zugänglichkeit dieser Ressource nimmt die Anzahl zugänglicher Datensätze stetig zu. Ein Datensatz muss genau auf die Anwendbarkeit auf eine bestimmte Problemstellung untersucht werden. Was die Eigenschaften der Datensätze anbelangt, so haben diverse Survey Papers bereits gute Einteilungsmöglichkeiten präsentiert. Quantitative Merkmale können als Indikator für ausreichende Datenmengen dienen. Wobei ausreichende Datenmengen von Anwendungsfall zu Anwendungsfall sehr unterschiedlich ausfallen können. Die Qualität der Daten ist der entscheidende Faktor, um bestehende Datensätze für neue Anwendungen einzusetzen.

„Welche Datensätze sind für das Training eines Handgesten-Erkennungs-Modells in der industriellen Baustellenmontage geeignet und welche Eigenschaften sind dafür entscheidend?“

Aus den Erkenntnissen und Erfahrungen der vorliegenden Arbeit lässt sich schließen, dass das Erstellen eines neuen Datensatzes eine sinnvolle Alternative zur Verwendung bereits vorhandener Datensätze darstellen kann. Für Entwickler und Forscher im Bereich der Gesteninteraktion kann ein eigenes Gestenalphabet zudem mehr Freiraum bei der Gestaltung und Zuteilung der Gesten auf jeweilige Funktion bedeuten. Bei der Erstellung eines Datensatzes können Best Practice von bestehenden Datensätzen übernommen werden. So bietet es sich beispielsweise an Gestenkategorien von bereits vorhandenen Datensätzen als Anhaltspunkt für die Gestaltung eines neuen Datensatzes heranzuziehen. Dabei stellte sich heraus, dass

es oft schwierig sein kann mit Hilfe vorhandener Datensätze zufriedenstellende Resultate auf eine vorliegende Problemstellung zu erzielen.

Die Tabellen aus Kapitel 4.1 können als Ausgangspunkt für die Auswahl eines geeigneten Datensatzes herangezogen werden. Gegebenenfalls können mehrere Datensätze zusammengefügt oder verschiedenen Ansätze mittels unterschiedlicher Datensätze ausprobiert werden.

9.2 Diskussion der Versuchsergebnisse

Die Genauigkeit des Prototypen-Modells zur Handgestenerkennung auf statischen Bildern ist trotz der limitierten Anzahl an Testpersonen und Testbildern bereits sehr hoch. Der erste Teil des Versuchs lieferte mit 99,63% Genauigkeit eine zufriedenstellende Genauigkeit auf 270 Testbildern. Vernachlässigt man die Klasse, in der keine Geste ausgeführt wird, wird eine Genauigkeit von 99,55% erreicht.

Das Ergebnis des Echtzeit-Tests liefert bei 150 Interaktionen 8 Fehler. Das ergibt eine Genauigkeit von 94,67%. Das bedeutet, dass 5,33% der Interaktionen fehlerhaft klassifiziert werden. Dieser Wert würde bedeuten, dass jede zwanzigste Interaktion zu einem falschen Befehl führen würde, was erhebliche Einschränkung des Steuerungsprinzips bedeuten würde. Der Versuch wurde mit 3 Personen durchgeführt und stellt mit 30 Bilder pro Klasse eine relativ kleine Stichprobe dar. Die Aussagekraft ist zu diesem Zeitpunkt somit begrenzt. Allerdings lässt sich in dieser Phase bereits erkennen, dass die Performance in der aktuellen Anwendung nicht zufriedenstellend ist, wobei von einem gewissen Trainingseffekt des Mitarbeiters bei längerer Benützungsdauer auszugehen ist. Beim Live-Test fällt zudem auf, dass 7 der 8 Fehler bei der Interaktion in der ersten Hälfte (25 Gesten) des Live-Tests aufgetreten sind. Dies lässt auf einen gewissen Grad der Anpassung bei den Testpersonen schließen, zumal sie das Feedback direkt am Bildschirm angezeigt bekommen haben.

Das Verwenden von Standard-Hardware erleichtert die Reproduzierbarkeit des Versuchs und zeigt, dass für Klassifizierungsmodelle von Handgesten keine High Tech Equipment notwendig ist. Die *Data Augmentation*-Technik ermöglicht es vergleichsweise kleine Datensätze künstlich zu vergrößern. Die Technik ist gut geeignet variierende Lichtverhältnisse zu simulieren, um die Generealisierungsfähigkeit des Modells zu erhöhen. Die Implementierung mittels *Tensorflow* und *Keras* ist mit geringem Aufwand verbunden.

Das Auswerten in einem speziell dafür vorgesehen Fenster, der ROI, hat den Vorteil gegenüber einer Erkennung im gesamten Arbeitsbereich das spontan ausgeführte Gesten nicht zu einer Interaktion führen. Das vorgesehene Konzept sieht vor, dass mit der Kreis-Geste das Werkerinformationssystem gestartet bzw. gestoppt wird. Diese Maßnahme kann bereits Bedienfehler beim Start bzw. beim Beenden des User

Interfaces vorbeugen. Der Projektor am Demonstrator ermöglich das Einblenden des virtuellen Interaktionsfeld ROI. Da der Versuch in einer anderen Umgebung durchgeführt wurde, musste die ROI auf dem Untergrund markiert werden.

Bei dem vorgeschlagene Kamerasetting (Top-View) ist zu keinem Zeitpunkt das Gesicht des Nutzers abgebildet. Das Gesicht des Benutzers muss demnach nicht aus datenschutzrechtlichen Gründen unkenntlich gemacht werden. Beim durchgeführten Versuch wird die Geste, welche das Modell nach ca. 2 Sekunden ausgibt, im Versuchsprotokoll vermerkt. Dies ermöglicht es dem Nutzer seine Geste während dieser Ausführungszeit anzupassen. Eine Alternative wäre die durchzuführende Geste für eine vordefinierte Zeitdauer (etwa 2 Sekunden) zu halten, um eine Interaktion auszulösen. Zur Erkennung von statischen Gesten kann ein *Timer* im Hintergrund der Anwendung mitlaufen, welcher nach einer vordefinierten Zeitspanne interagiert. Hier gilt es abzuwägen wie lange ein solche Auslösung dauern soll. Der Benutzer soll nämlich einerseits das Gefühl einer Echtzeitzeit-Interaktion haben andererseits muss gewährleistet werden das Fehlinteraktionen bestmöglich vermieden werden. Momentan wird kontinuierlich eine Schätzung (engl. *prediction*) über die aktuell ausgeführte Geste bzw. Nicht-Geste in der *Region of Interest* gemacht. Ob eine Geste für eine gewünschte Funktion ausreichend gut passt, muss durch User-Befragungen evaluiert werden.

9.3 Nächste Schritte und Ausblick

Durch das Entfernen der Kreisgeste könnten möglicherweise ein Großteil der aktuell auftretenden Fehler eliminiert werden. Diese Hypothese müsste bei einem erneuten Versuch getestet werden. Falls die Kreisgeste weiterhin Bestandteil des Interaktionskonzeptes bleibt, sollte in der nächsten Phase die Anzahl der Trainingsbilder erhöht werden. Dadurch kann mehr Variation und eine bessere Anpassung des Datensatzes an das industrielle Arbeitsumfeld ermöglicht werden. Für die Akzeptanz der Anwendung ist eine hohe Genauigkeit unabdingbar.

Weitere Tests am Demonstrator könnten zudem Aufschluss über verschiedene Lichteinflüssen und Abstände zum Bauteil geben. Zudem sollte das Modell auf einer leistungsstärkeren Hardware getestet werden. Das Projizieren des Interaktionsfensters wurde in der vorliegenden Arbeit nicht berücksichtigt. Das reibungslose Zusammenspiel zwischen dynamischen Projektor und vorgeschlagenem Interaktionskonzept sollten in einem nächsten Schritt eingegangen werden. Außerdem gilt es eine Lösung für die Auslösung der Interaktionsbefehle zu finden. In dieser Arbeit wurde diese Interaktion mit ca. zwei Sekunden Verweildauer gelöst. Sobald die angeführten Herausforderungen beseitigt sind, kann im nächsten Schritt eine Gebrauchstauglichkeitsstudie durchgeführt werden. Bei der Gebrauchstauglichkeitsstudie sollten die Teilnehmer befragt werden, ob die jeweilige

Geste auch zur dazugehörigen Funktion passt. Außerdem wäre ein Vergleich mit *Wearables* naheliegend.

Ein interessanter Ansatz wäre das Interaktionsmenü mit Hilfe statischer Gesten zu aktiveren und den Gestendatensatz, um *Swipe*-Bewegungen zu erweitern. Diese kommen der natürlichen Interaktion wie beispielsweise am Smartphone näher. Ein solcher Ansatz ist technisch anspruchsvoller zu lösen und geht mit neuen Herausforderungen einher.

Bei der industriellen Baustellenmontage stellt sich zudem die Frage ob einer Erkennung von Gesten, die mit Handschuhen ausgeführt werden, sinnvoll ist. Die Farbe der Handschuhe sollte sich dabei vom Hintergrund abgrenzen.

10 Literaturverzeichnis

Amini, A., & Soleimany, A. (2020). *Introduction to Deep Learning*. http://introtodeeplearning.com

Bao, P., Maqueda, A. I., Del-Blanco, C. R., & García, N. (2017). Tiny hand gesture recognition without localization via a deep convolutional network. *IEEE Transactions on Consumer Electronics*, *63*(3), 251–257.

Bendel, O. (2019). *Mensch-Maschine-Interaktion*. https://wirtschaftslexikon.gabler.de/definition/mensch-maschine-interaktion-54079/version-368836

BMW AG. (2019). *Natürliche und vollständig multimodale Interaktion mit dem Fahrzeug und der Umgebung. Auf dem Mobile World Congress 2019 präsentiert die BMW Group erstmals BMW Natural Interaction.* https://www.press.bmwgroup.com/deutschland/article/detail/T0292196DE/natuerliche-und-vollstaendig-multimodale-interaktion-mit-dem-fahrzeug-und-der-umgebung-auf-dem-mobile-world-congress-2019-praesentiert-die-bmw-group-erstmals-bmw-natural-interaction?language=de

Brooke, J., Jordan, P. W., Thomas, B., Weerdmeester, B. A., & McClelland, I. L. (1996). *Usability evaluation in industry*.

Butz, A., & Krüger, A. (2017). *Mensch-Maschine-Interaktion*. Walter de Gruyter GmbH & Co KG.

Cadoz, C. (1994). *Le geste canal de communication homme/machine: la communication" instrumentale"*.

Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User acceptance of computer technology: a comparison of two theoretical models. *Management science*, *35*(8), 982–1003.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

Der Standard. (2017). *Du warst der Controller: Microsoft beerdigt Kinect*.

DIN EN ISO 9241-11. (2018). *Ergonomie der Mensch-System-Interaktion - Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte (ISO 9241-11:2018)*.

DIN EN ISO 9241-960. (2017). *Ergonomie der Mensch-System-Interaktion - Teil 960: Rahmen und Anleitung zur Gestensteuerung (ISO 9241-960:2017)*.

Dix, A., Dix, A. J., Finlay, J., Abowd, G. D., & Beale, R. (2003). *Human-computer interaction*. Pearson Education.

Dudenredaktion. (o. J.). *„Geste“ auf Duden online. URL*. https://www.duden.de/node/57136/revision/57172

Ertel, W. (2016). *Grundkurs künstliche Intelligenz: eine praxisorientierte Einführung*. Springer-Verlag.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861–874.

Firman, M. (2016). RGBD Datasets: Past, Present and Future. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *Section 3*, 661–673. https://doi.org/10.1109/CVPRW.2016.88

Flasiński, M. (2016). *Introduction to artificial intelligence*. Springer.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Bd. 1). MIT press Cambridge.

Hartson, H. R. (1998). Human–computer interaction: Interdisciplinary roots and trends. *Journal of systems and software*, *43*(2), 103–118.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Herbig, N. (2016). Nutzwertanalyse. *Eine Methode zur Bewertung von Lösungsalternativen und zur Entscheidungsfindung*, *2*.

Hewett, T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., & Verplank, W. (1992). ACM SIGCHI Curricula for Human-Computer Interaction. In *ACM SIGCHI Curricula for Human-Computer Interaction*. https://doi.org/10.1145/2594128

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hsiao, Y.-S., Sanchez-Riera, J., Lim, T., Hua, K.-L., & Cheng, W.-H. (2014). LaRED: a large RGB-D extensible hand gesture dataset. *Proceedings of the 5th ACM Multimedia Systems Conference*, 53–58.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., & Guadarrama, S. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7310–7311.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, *53*(8), 5455–5516.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097–1105.

Kühnapfel, J. B. (2014). *Nutzwertanalysen in Marketing und Vertrieb*. Springer.

Lang, S. (2007). *Durchgängige Mitarbeiterinformation zur Steigerung von Effizienz und Prozesssicherheit in der Produktion*.

Le Cun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard, R. E., & Hubbard, W. (1989). Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, *27*(11), 41–46.

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lotter, B., & Wiendahl, H.-P. (2013). *Montage in der industriellen Produktion: Ein Handbuch für die Praxis*. Springer-Verlag.

Marcel, S. (2002). *Gestures for Multi-Model Interfaces: A Review*.

Marcel, S. (1999). Hand posture recognition in a body-face centered space. *CHI'99 Extended Abstracts on Human Factors in Computing Systems*, 302–303.

Materzynska, J., Berger, G., Bax, I., & Memisevic, R. (2019). The Jester Dataset: A Large-Scale Video Dataset of Human Gestures. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2874–2882.

Matilainen, M., Sangi, P., Holappa, J., & Silvén, O. (2016). OUHANDS database for hand detection and pose recognition. *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 1–5.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McNeill, D. (1992). *Hand and mind: What gestures reveal about thought*. University of Chicago press.

McNeill, D. (2008). *Gesture and thought*. University of Chicago press.

Memo, A., Minto, L., & Zanuttigh, P. (2015). Exploiting silhouette descriptors and synthetic data for hand gesture recognition. *STAG: Smart Tools & Apps for Graphics*. https://doi.org/10.2312/stag.20151288

Microsoft. (2020a). *Azure Kinect DK*. https://developer.microsoft.com/de-de/windows/kinect/

Microsoft. (2020b). *Kinect für Windows*.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, *63*(2), 81.

Mitchell, T. M. (1997). *Machine Learning, volume 1 of 1*. McGraw-Hill Science/Engineering/-Math.

Neuschwinger, A. (2003). *Multimediales, informationsmodellbasiertes Arbeitsplatz-Kommunikationssystem: Ein modularer Ansatz mit normierten und standardisierten Datenstrukturen*. Shaker.

Nielsen, M. A. (2015). *Neural networks and deep learning* (Bd. 2018). Determination press San Francisco, CA.

Peltarion. (2020a). *2D Convolution block*. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-convolution-block

Peltarion. (2020b). *2D Max pooling block*. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-max-pooling-block

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Preim, B., & Dachselt, R. (2015). *Interaktive Systeme: Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces*. Springer-Verlag.

Reinhart, G. (2017). Handbuch Industrie 4.0 - Geschäftsmodelle, Prozesse, Technik. In *Handbuch Industrie 4.0*. https://doi.org/10.3139/9783446449893.fm

Richtlinie, V. D. I. (1990). VDI 2860-Montage-und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole. *Verein Deutscher Ingenieure*.

Ruffieux, S., Lalanne, D., Mugellini, E., & Abou Khaled, O. (2014). A survey of datasets for human gesture recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8511 LNCS*(PART 2), 337–348. https://doi.org/10.1007/978-3-319-07230-2_33

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation*. California Univ San Diego La Jolla Inst for Cognitive Science.

Rupprecht, P., Kueffner-Mccauley, H., & Schlund, S. (2020). Information provision utilizing a dynamic projection system in industrial site assembly. *Procedia CIRP*, *93*, 1182–1187.

Russell, S., & Norvig, P. (2002). *Artificial intelligence: a modern approach*.

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal of research and development*, *11*(6), 601–617.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

Sarkar, D., Bali, R., & Sharma, T. (2018). Practical machine learning with Python. *A*

problem-solvers guide to building real-world intelligent systems. Apress, Berkely.

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 60.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, *45*(4), 427–437.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

TedCas. (2020). *TedCube*. https://tedcas.es/tedcube/

The MathWorks, I. (2020). *Convolutional Neural Network*. https://de.mathworks.com/discovery/convolutional-neural-network-matlab.html

Vogel-Heuser, B., Bauernhansl, T., & Ten Hompel, M. (2017). Handbuch Industrie 4.0 Bd. 4. *Allgemeine Grundlagen*, *2*.

Volkswagen AG. (2020). *Der Tiguan Allspace*. https://www.volkswagen.at/tiguan-allspace

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, *2018*.

Vuletic, T., Duffy, A., Hay, L., McTeague, C., Campbell, G., & Grealy, M. (2019). Systematic literature review of hand gestures used in human computer interaction interfaces. *International Journal of Human Computer Studies*, *129*(September 2018), 74–94. https://doi.org/10.1016/j.ijhcs.2019.03.011

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, *3*(1), 9.

Wiendahl, H. P., Reichardt, J., & Nyhuis, P. (2009). Handbuch Fabrikplanung: Konzept. *Gestaltung und Umsetzung wandlungsfähiger Produktionsstätten, Hanser, München*.

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, 29–39.

Zangemeister, C. (2014). *Nutzwertanalyse in der Systemtechnik: eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen*. BoD–Books on Demand.

Zhang, J., Li, W., Ogunbona, P. O., Wang, P., & Tang, C. (2016). RGB-D-based action recognition datasets: A survey. *Pattern Recognition*, *60*, 86–105.

https://doi.org/10.1016/j.patcog.2016.05.019

11 Abbildungsverzeichnis

12 Formelverzeichnis

13 Tabellenverzeichnis

14 Abkürzungsverzeichnis

bzw.	beziehungsweise
DL	Deep Learning
CV	Computer Vision
ML	Machine Learning
MMI	Mensch Maschine Interaktion
IKT	Informations- und Kommunikationstechnologie
CPS	Cyber-Physischem System
usw.	und so weiter
WIS	Werkerinformationssystem
ISO	International Organization for Standardization
DIN	Deutsches Institut für Normung
GUI	Graphical User Interface
NUI	Natural User Interface
z.B.	zum Beispiel
MCI	Mensch-Computer-Interaktion
HMI	Human Machine Interaktion
KNN	Künstliche Neuronale Netze
ASL	American Sign Language
k. A.	Keine Angabe
ROI	Region of Interest
ReLU	Rectified Linear Unit
GF	Gewichtsfaktor
ZEF	Zielerfüllungsfaktor
TNW	Teilnutzwert
AVG	average
TL	Transfer Learning

15 Anhang

Im Anhang befindet sich der Code zur Aufnahme der Beispielgesten und der Code zum Test des ML-Modells.

Aufnahme der Beispielgesten

```
import cv2
import os
import sys

cam = cv2.VideoCapture(1) #Festlegen der Webcam

#Aufnahmefenster benennen
cv2.namedWindow("Bildaufnahme")

#Dateipfad festlegen
path = '/' #Speicherpfad festlegen
img_counter = 1 #Zähler starten

#Aufnahme starten
while True:
    ret, frame = cam.read()
    #frame = cv2.flip(frame, 1)

    #ROI festlegen und auf Bildschirm ausgeben
    cv2.rectangle(frame, (880, 50), (1234, 404), (255, 255, 255), 2)

    if not ret:
        print("no frame")
        break
    cv2.imshow("test", frame)

    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break
    elif k%256 == 32:
        # SPACE pressed
        roi = frame[56:402, 886:1232]
        roi = cv2.resize(roi, (224,224))
        save_path = os.path.join(path,
"1_persB_gest{}.jpg".format(img_counter))
        cv2.imwrite(save_path, roi)

        print("{} written!".format(save_path))
        img_counter += 1

cam.release()

cv2.destroyAllWindows()
```

Test der Echtzeit-Interaktion:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from keras.models import load_model

from keras.models import load_model
import cv2
import numpy as np
from random import choice

REV_CLASS_MAP = {
    0: "Kreis",
    1: "Leer",
    2: "Faust",
    3: "5-Finger",
    4: "1-Finger",
    5: "2-Finger",
}

def mapper(val):
    return REV_CLASS_MAP[val]

model=load_model("/")#Dateipfad zum Model festlegen

cam = cv2.VideoCapture(0)

while True:
    ret, frame = cam.read()
    frame = cv2.flip(frame, 1)

    #Markieren eines rechteckigen Bereichs (ROI) mit weißer Randmarkierung.

    cv2.rectangle(frame, (880, 50), (1234, 404), (255, 0, 50), 2)

    if not ret:
        print("no frame")
        break

    #Extrahieren der ROI und Format ändern
    roi = frame[56:402, 886:1232]
    #img = cv2.cvtColor(roi, cv2.COLOR_BGR2RGB)
    img = cv2.resize(roi, (224, 224),)
    img = tf.keras.applications.mobilenet_v2.preprocess_input(img)

    # Gestenvorhersage
    pred = model.predict(np.array([img]))
    pred_code = np.argmax(pred[0])
    gesture_name = mapper(pred_code)

    # vorhergesagte Geste auf Bildschirm ausgeben
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, "Interaktion: " + gesture_name,
                (50, 50), font, 1.2, (225, 0, 50), 2, cv2.LINE_AA)

    cv2.imshow("main_rps", frame)
```

```
    k = cv2.waitKey(1)
    if k%256 == 27:
        # ESC pressed
        print("Escape hit, closing...")
        break


cam.release()

cv2.destroyAllWindows()
```

Versuchsprotokoll:

		Geste-Vorhersage		
Nr.	Geste-Vorgabe	Testperson 1	Testperson 2	Testperson 3
1	5-Finger	✓	✓	✓
2	Kreis	× (Faust)	✓	✓
3	1-Finger	✓	✓	✓
4	Faust	✓	✓	✓
5	2-Finger	✓	✓	✓
6	Faust	✓	✓	✓
7	1-Finger	✓	✓	✓
8	Kreis	✓	✓	✓
9	5-Finger	✓	✓	✓
10	2-Finger	✓	✓	✓
11	Kreis	✓	× (Faust)	× (Faust)
12	Faust	✓	✓	✓
13	2-Finger	✓	✓	✓
14	1-Finger	✓	✓	✓
15	5-Finger	✓	× (2-Finger)	✓
16	2-Finger	✓	✓	✓
17	Kreis	✓	✓	✓
18	1-Finger	✓	✓	✓
19	5-Finger	× (1-Finger)	× (2-Finger)	✓
20	Faust	✓	✓	✓
21	5-Finger	✓	✓	✓
22	1-Finger	✓	✓	✓
23	Kreis	✓	✓	× (Faust)
24	Faust	✓	✓	✓
25	2-Finger	✓	✓	✓
26	1-Finger	✓	✓	✓
27	5-Finger	✓	✓	✓
28	Kreis	✓	✓	✓
29	2-Finger	✓	✓	✓
30	Faust	✓	✓	✓
31	2-Finger	✓	✓	✓
32	5-Finger	✓	✓	✓
33	Kreis	✓	✓	✓
34	1-Finger	✓	✓	✓
35	Faust	✓	✓	✓
36	Kreis	✓	✓	× (Faust)
37	1-Finger	✓	✓	✓
38	5-Finger	✓	✓	✓

39	2-Finger	✓	✓	✓
40	Faust	✓	✓	✓
41	1-Finger	✓	✓	✓
42	Faust	✓	✓	✓
43	2-Finger	✓	✓	✓
44	Kreis	✓	✓	✓
45	5-Finger	✓	✓	✓
46	Kreis	✓	✓	✓
47	Faust	✓	✓	✓
48	5-Finger	✓	✓	✓
49	2-Finger	✓	✓	✓
50	1-Finger	✓	✓	✓