KI-Taschenrechner: Entwicklung eines Prototypen zur Kombination von Spracheingabe und Handgestik zur Steuerung eines Taschenrechners

Rene Zeiml

BACHELORARBEIT

eingereicht am
Fachhochschul-Bachelorstudiengang

Hardware-Software-Design

in Hagenberg

im September 2021

Betreuung:

FH-Prof. DI Michael Bogner

© Copyright 2021 Rene Zeiml

Diese Arbeit wird unter den Bedingungen der Creative Commons Lizenz *Attribution-NonCommercial-NoDerivatives 4.0 International* (CC BY-NC-ND 4.0) veröffentlicht – siehe https://creativecommons.org/licenses/by-nc-nd/4.0/.

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt. Die vorliegende, gedruckte Arbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Hagenberg, am 1. September 2021

Rene Zeiml

Inhaltsverzeichnis

Erklärung **iv**

Danksagung **viii**

Kurzfassung **ix**

Abstract **x**

1 Einleitung **1**
1.1 Motivation 1
1.2 Aufgabenstellung und Ziel der Arbeit 2
1.3 Gliederung 3

2 Grundlagen der Künstlichen Intelligenz **4**
2.1 Definition 4
2.2 Biologisches Neuron 5
2.3 Künstliches Neuron 6
2.4 Künstliche Neuronale Netze 7
2.5 Maschinelles Lernen 7
2.6 Deep Learning 9
2.7 Convolutional Neural Network 10
2.7.1 Convolutional Layer 10
2.7.2 Pooling Layer 11
2.7.3 Rectified Linear Unit 12
2.8 Trainieren und Testen eines Modells 12
2.9 Bildklassifizierung 13
2.10 Spracherkennung 14

3 Stand der Technik **17**
3.1 Gestensteuerung 17
3.1.1 MediaPipe 17
3.1.2 ML5 Handpose 18
3.1.3 OpenCV 18
3.1.4 Fazit 19
3.2 Sprachsteuerung 19
3.2.1 Google Speech-to-Text 19

3.2.2 Vosk . . . 20
3.2.3 Picovoice . . . 20
3.2.4 Eigenes Modell erstellen . . . 20
3.2.5 Fazit . . . 21
3.3 Zusammenfassung . . . 21

4 Gesamtbild des Systems 22

5 Implementierung 24
5.1 Gestensteuerung . . . 24
5.1.1 Vorverarbeitung der Eingangsdaten . . . 24
5.1.2 Ergebnis von MediaPipe Hands . . . 24
5.1.3 Tests und Fazit . . . 25
5.2 Sprachsteuerung . . . 25
5.2.1 Datenbeschaffung . . . 25
5.2.2 Modellerstellung . . . 27
5.2.3 Fazit . . . 29
5.3 Ergebnis . . . 29

6 Implementierung mit Teachable Machine 30
6.1 Sprachsteuerung . . . 30
6.1.1 Teachable Machine . . . 30
6.1.2 Modellerstellung . . . 31
6.1.3 Anwendung des Modells . . . 33
6.1.4 Test und Fazit . . . 34
6.2 Gestensteuerung . . . 35
6.2.1 Teachable Machine für Gestensteuerung . . . 35
6.2.2 ML5 Handpose . . . 36
6.2.3 MediaPipe Hands mit JavaScript . . . 36
6.2.4 Auswertung der Erkennung . . . 36
6.2.5 Test und Fazit . . . 37
6.3 Kombination von Sprach- und Gestensteuerung . . . 37
6.3.1 Bedienoberfläche . . . 37
6.3.2 Funktion . . . 38
6.4 Zusammenfassung . . . 39

7 Tests 40
7.1 Test der Sprachsteuerung . . . 40
7.1.1 Headset . . . 40
7.1.2 Internes Mikrofon . . . 40
7.1.3 Fazit . . . 41
7.2 Test der Gestensteuerung . . . 42
7.2.1 Geeigneter Hintergrund . . . 42
7.2.2 Schlechte Lichtverhältnisse . . . 42
7.2.3 Direktes Licht . . . 42
7.2.4 Keine externen Lichtquellen . . . 43
7.2.5 Objekte im Hintergrund . . . 45

7.2.6 Fazit 46
7.3 Finale Demonstration 46

8 Schlussbemerkungen 48
8.1 Zusammenfassung und Ergebnis 48
8.2 Probleme und Erkenntnisse 48
8.3 Ausblick 49

A Ergänzende Inhalte 50
A.1 PDF-Dateien 50
A.2 Mediendaten 50
A.3 Code 50
A.4 Quellen 50

Quellenverzeichnis 51
Literatur 51
Medien 51
Software 52
Online-Quellen 52

Danksagung

In diesem Abschnitt möchte ich mich bei all denjenigen bedanken, die mich maßgeblich beim Erstellen dieser Arbeit unterstützt haben.

Mein Dank gilt meinem Betreuer FH-Prof. DI Michael Bogner für seine kritischen wie hilfreichen Anregungen während des gesamten Betreuungszeitraumes.

Ich möchte meiner Mama, meinem Papa und meiner Schwester danken, die mich immer wieder motiviert haben und mich daran erinnerten, warum ich das Ganze eigentlich mache. Sie stehen immer auf meiner Seite und unterstützen mich vollkommen.

Ein sehr großes Dankeschön möchte ich an meine guten Freunde und Kommilitonen Maximilian Burger, Daniel Stummer und Adam Kensy aussprechen. Ohne sie hätte ich das Studium und die Bachelorarbeit nur schwer bewältigen können. Die Gegenseitige Motivation und Hilfe im Studium, sowie die Gespräche und Aktivitäten außerhalb des Studiums, machen die Studienzeit erst zu dem was es ist.

Für das Korrekturlesen möchte ich mich besonders bei Dipl.-Ing. Dr.techn. Wilfried Lepuschitz, David Laßlberger, Michael Traxler und Andrea Haider-Pachtrog bedanken.

Kurzfassung

Traditionell werden Computer mit Tastatur und Maus gesteuert. Die neuen Fortschritte im Bereich der künstlichen Intelligenz erlauben Alternativen, welche die Kommunikation mit Computern erleichtern. Besonders Menschen mit Einschränkungen, Kinder oder alte Menschen können durch diese Eingabemethoden unterstützt werden. Zwei dieser alternativen Methoden werden in dieser Arbeit behandelt.

Im Rahmen dieser Bachelorarbeit wird deshalb ein audiovisuell gesteuerter Taschenrechner realisiert. Dieser soll sowohl per Sprache als auch per Handgestik gesteuert werden. Beide dieser Eingabemethoden sollen die zehn Ziffern, die Grundoperatoren der Addition, Subtraktion, Multiplikation und Division sowie das Gleichheitszeichen erkennen können. Diese bilden die fünfzehn Klassen zwischen denen unterschieden werden muss. Um dies zu erreichen wurde der Stand der Technik in den bereichen der Sprach- und Gestenerkennung betrachtet und es wurden verschiedene Ansätze und Bibliotheken für die Umsetzung verglichen. Anhand der zuvor definierten Parametern wurde für jede Steuerungseinheit ein Ansatz gewählt. Mit diesen wurde ein System für den Prototypen definiert. Zur Umsetzung wurde zuerst ein Sprachklassifizierungsmodell erstellt, welches die gesprochenen Wörter den obigen genannten Klassen zuordnet. Dies funktionierte mit einer Genauigkeit von ungefähr 77%. Die Gestensteuerung wurde mit einem Modell erstellt, welches zuerst die Hand in einem Bild erkennt und darauf dann ein Gestenerkennungsmodell angewendet wird. Dabei werden wichtige Punkte auf der Hand erkannt, wie zum Beispiel die Fingerspitzen und Fingeransätze. Mit diesen Punkten und dem Vergleichen deren Koordinaten konnten individuelle Gesten erkannt werden. Bei der Zusammenführung dieser zwei Eingabemethoden wurde darauf geachtet, dass diese dynamisch und ohne dem Einwirken des Benutzers funktioniert. Dies wurde erreicht indem eine periodische Abtastung beide Eingänge auswertet und diese in die eigentliche Rechnung übernimmt. Sobald das Gleichheitszeichen eingegeben wurde, signalisiert dies das Ende der Rechnung. Die Rechnung wird ausgewertet und das Ergebnis wird angezeigt.

Es wurde ein voll funktionsfähiger Taschenrechner erstellt, der mit Sprache und Handgestik gesteuert werden kann. Dabei wurde gezeigt wie die Kombination von zwei solcher alternativen Eingabemethoden funktionieren kann.

Abstract

Traditionally, computers are controlled with keyboard and mouse. The new advances in the field of artificial intelligence allow alternatives that facilitate communication with computers. Especially people with disabilities, children or old people can be supported by these input methods. Two of these alternative input methods are discussed in this paper.

In the context of this bachelor thesis an audiovisual pocket calculator will be realized. It will be controlled by speech as well as by hand gestures. Both of these input methods will be able to recognize the ten digits, the basic operators of addition, subtraction, multiplication and division as well as an equal sign. These form the fifteen classes between which distinctions must be made. To achieve this, the state of the art in the areas of speech and gesture recognition was considered and various approaches and libraries for implementation were compared. Based on the previously defined parameters, an approach was chosen for each control unit. With these, a system for the prototype was defined. For implementation, a speech classification model was first created which assigned the spoken words to the above classes. This worked with an accuracy of about 77%. The gesture control was created with a model that first recognizes the hand in an image and then a gesture recognition model is applied to it. This recognizes important landmarks on the hand, such as the fingertips and the base of the fingers. With these landmarks and by comparing their coordinates, individual gestures can be recognized. When merging these two input methods, care was taken to ensure that they functioned dynamically and without the user's intervention. This was achieved by a periodic sampling that evaluates both inputs and takes them over into the actual calculation. As soon as the equal sign is entered the end of the calculation is signaled. The calculation is evaluated and the result is displayed.

A fully functional calculator was created that can be controlled by speech and hand gestures. It was shown how the combination of two such alternative input methods can work.

Kapitel 1

Einleitung

1.1 Motivation

Ein derzeit sehr aktueller Bereich, wo künstliche Intelligenz angewendet wird, ist der der autonomen Fahrzeuge. Zum Beispiel verwenden Tesla und Waymo hochkomplexe KI-Algorithmen, sowie die Eingangssignale von Kameras und Sensoren, um schon jetzt vollautonomes Fahren zu ermöglichen [9][48]. Ein weniger offensichtlicher Anwendungsfall von KI sind Online-Dienstleistung, wie zum Beispiel Google Maps. Hierbei werden Algorithmen verwendet um verschiedene Eigenschaften des Verkehrs vorherzusagen und somit den bestmöglichen Weg zu finden [31].

KI wird also bereits in vielen Bereichen verwendet. Die Motivation hinter dieser Arbeit bezieht sich jedoch speziell auf die Verwendung von KI für alternative Eingabemethoden. Zu dieser Kategorie gehören zum Beispiel Steuerungen per Sprache und Gestik. Traditionell wird mit Computern nur durch Tastatur und Maus interagiert. Alternative Eingabemethoden hingegen ermöglichen nicht nur Menschen mit Behinderung, Kindern oder alten Menschen eine Alternative, sondern können auch jedem anderen im Alltag unterstützen. Ein interessanter Punkt dabei ist die Sprachverarbeitung bei Sprachsteuerungen und Sprachassistenten. Jedes Smartphone hat heutzutage einen persönlichen KI-Assistenten wie Siri, Alexa oder Google Assistent, dem man durch einfache Sprachbefehle Anweisungen gibt. Durch Anwendungen wie Voice Access von Google, ist die komplette Steuerung von Mobilgeräten möglich [43]. Dies eröffnet neue Möglichkeiten für Menschen mit Behinderung, aber auch für jeden anderen. Somit ist es möglich sein Smartphone zu steuern, auch wenn keine Hand frei ist und visueller Fokus benötigt wird, wie beim Kochen oder beim Autofahren. Steuerung durch Gestik ist eine weitere Alternative. Bereits 2003 war es möglich die Playstation 2 mit der EyeToy-Kamera zu steuern [6]. Später dann durch die Kinect, welche nicht nur für Spiele auf der Xbox verwendet wurde, sondern für viele weitere Projekte wie zum Übersetzen der Gebärdensprache [28]. Auch bei Virtual- und Augmented-Reality-Produkten wie der Oculus Quest 2 ist die Steuerung mit Gestik möglich [39].

Aus diesen Beispielen ist deutlich zu erkennen, dass auch die Kombination der verschiedenen Eingabemethoden neue interessante Aufgaben ermöglicht. Smartphones kön-

nen beispielsweise per Touchscreen, Sprache und/oder Gestik bedient werden. Durch solche Kombinationen eröffnen sich wieder neue Möglichkeiten, die den Alltag der Menschen etwas leichter machen können.

1.2 Aufgabenstellung und Ziel der Arbeit

Die Aufgabenstellung dieser Arbeit ist es einen audiovisuellen Taschenrechner zu realisieren. Dabei soll dieser sowohl durch Sprachbefehle als auch durch Gesten gesteuert werden können. Der Taschenrechner soll somit einfache Rechenoperationen wie Addition, Subtraktion, Multiplikation und Division ausführen können. Der Prototyp wird in Form eines Taschenrechners realisiert, da dessen Funktion nachvollziehbar und bekannt ist. Eine Person könnte zum Beispiel 'Eins' und dann 'Drei' sagen. Danach ein Pluszeichen mit der Hand bilden. Anschließend noch 'Vier' aussprechen und dann eine Zwei gestikulieren und schließlich noch 'ist gleich' sagen. Der Prototyp hat dann die Rechnung erkannt und zeigt die Eingabe und das Ergebnis auf einem Monitor an: $13+42 = 55$

Die Vorgehensweise dieser Arbeit ist in drei Teilprobleme aufgeteilt. Das Erste wird die Steuerung mittels Gestik behandeln. Hier ist die Herausforderung die Eingangsdaten einer Kamera richtig zu klassifizieren, also anhand der Bilder zu erkennen, ob eine Hand und deren Gestik richtig erkannt wird. Es soll möglich sein zwischen den Ziffern 0-9, sowie den Grundoperatoren und einem Gleichheitszeichen zu unterscheiden. Dazu wird es notwendig sein vorhandene Methoden zu finden, zu testen und zu analysieren. Falls es noch keine passende Lösung gibt, wird es nötig sein ein eigenes KI-Modell zu trainieren. Falls dies der Fall ist kommt als Zusatzproblem die Beschaffung eines großen Datensatzes an Bildern hinzu. Hierbei muss wiederum nachgeforscht werden, ob es schon einen passenden Datensatz gibt und wenn nicht, muss dieser erstellt werden.

Im nächsten Schritt soll eine Sprachsteuerung realisiert werden. Hierbei ist es nicht nötig ein komplettes Sprachmodell zu verwenden welches jedes Wort erkennt. Es müssen nur die zehn Ziffern, die Grundoperatoren sowie ein Gleichheitszeichen richtig kategorisiert werden. Um dies zu erreichen müssen vorhandene Methoden und Ansätze betrachtet und verglichen werden. Falls es keinen passenden Ansatz gibt wird ein eigenes Modell erstellt. Hierbei müsste ebenfalls ein eigener Datensatz gesammelt werden, falls es keine öffentlich verfügbaren Alternativen gibt. Danach muss ebenfalls ein KI-Modell entworfen und trainiert werden. Dies würde diese Aufgabe ebenfalls um die Beschaffung eines Datensatzes erweitern.

Das Dritte und letzte Problem ist die Kombination dieser Eingänge und die Visualisierung der Rechnung. Die Bedienung des Prototypen sollte dabei nicht einschränkend wirken, sondern möglichst intuitiv und dynamisch funktionieren. Wenn möglich sollte der Prototyp ohne Internetverbindung einsetzbar sein und auch auf eingebetteten Systemen laufen können.

1.3 Gliederung

Im Kapitel 2 werden alle Grundlagen erklärt die für das Verständnis dieser Arbeit wichtig sind. Anschließend wird im Kapitel 3 der Stand der Technik betrachtet. Es werden verschiedene Ansätze, Bibliotheken und Frameworks recherchiert und verglichen. Anhand dieser Ergebnisse wird in dem Kapitel 4 das Gesamtsystem und dessen Eigenschaften definiert. In den folgenden zwei Kapiteln 5 und 6 wird ein Prototyp des Taschenrechners realisiert und dessen Umsetzung behandelt. Anschließend wird dieser Prototyp im Kapitel 7 ausgiebig getestet und dessen Ergebnisse disskutiert. Zum Schluss wird im Kapitel 8 eine Zusammenfassung der geleisteten Arbeit, Probleme und Ergebnisse beschrieben.

Kapitel 2

Grundlagen der Künstlichen Intelligenz

In diesem Kapitel wird auf alle nötigen Grundlagen eingegangen, die für das Verständnis dieser Arbeit wichtig sind. Zuerst wird auf die Definition von KI eingegangen. Anschließend wird der Unterschied zwischen biologischen und künstlichen Neuronen erklärt und wie diese zu neuronalen Netzen verknüpft werden. Dann wird auf die Begriffe des maschinellen Lernens und dem des Deep Learnings eingegangen. Anschließend wird genauer auf CNN-Modelle und deren Training eingegangen, da diese für die Klassifizierung von Bild und Ton wichtig sind. Zum Schluss werden noch wichtige Informationen zur Gesten- und Spracherkennung mithilfe von KI erklärt.

2.1 Definition

> „AI began with an ancient wish to forge the gods“ [8].

Wie mit dem obigen Zitat von Pamela McCorduck sehr schön ausgedrückt wird, gibt es die Idee selbstbewusster Maschinen schon sehr lange. Jedoch wurde erst im letzten halben Jahrhundert die nötige Rechenleistung erreicht, um erste KI-Prototypen zu realisieren [1]. Obwohl dieses Thema schon lange erforscht wird, gibt es zum Zeitpunkt, an dem diese Arbeit verfasst wurde, noch keine allgemeingültige Definition von künstlicher Intelligenz. Der Hauptgrund dafür ist, dass es noch keine offiziell anerkannte Definition für Intelligenz an sich gibt [4]. Dennoch gibt es schon etliche Versuche, diese zu definieren. Eine umfangreiche Sammlung von Definitionen bietet das Paper *A Collection of Definitions of Intelligence* [4]. Unabhängig von der Definition für Intelligenz an sich wird in dieser Arbeit die Definition für künstliche Intelligenz von Lucy Suchman verwendet. Diese beschreibt den Fachbereich der künstlichen Intelligenz sehr gut und lautet wie folgt [35]:

> „AI is the field of study devoted to developing computational technologies that automate aspects of human activity conventionally understood to require intelligence.“

Künstliche Intelligenz ist also ein Überbegriff für verschiedene Computertechnologien. Auf die wichtigsten Technologien und Anwendungsfälle wird im folgenden Kapiteln

eingegangen. Eine weitere wichtige Unterscheidung ist die zwischen starker und schwacher KI.

Starke KI, *strong artificial intelligence* oder auch *artificial general intelligence* (AGI), ist eine theoretische Form der Künstlichen Intelligenz. Dabei handelt es sich um eine KI, welche mindestens so intelligent ist wie ein Mensch, ein Bewusstsein hat, lernen und Probleme lösen kann und auch die Fähigkeit besitzt, für die eigene Zukunft zu planen. Solch eine KI würde dem Menschen in nichts nachstehen und wäre nicht mehr von dem menschlichen Verstand zu unterscheiden [29]. Zu dem Zeitpunkt, an dem eine KI erschaffen wird, die den Beschreibungen der AGI entspricht, wird die technologische Singularität erreicht. Dabei handelt es sich um eine Theorie von Raymond Kurzweil. In seinem Buch *The Singularity Is Near* beschreibt er diese Singularität als einen Zeitpunkt, an dem sich die KI ohne das Einwirken des Menschen von alleine weiterentwickelt. Die Intelligenz der KI würde sich demnach exponentiell erhöhen [3].

Das Gegenstück zur Starken KI bildet die Schwache KI, *weak artificial intelligence*, welche sich nur auf spezielle Aufgaben konzentriert, wie zum Beispiel auf Schach spielen oder auf das Erkennen von Sprache. Schwache KI benötigt Menschen zum Definieren der Parameter der Lernalgorithmen [29].

Bis heute existiert noch keine Starke KI und es ist nur ein theoretisches Konzept. Das heißt, dass jedes derzeitig verwendete KI-System der Schwachen KI zugeordnet wird. In dieser Arbeit wird darum, wenn nicht anders explizit erwähnt, immer von Schwacher KI gesprochen, wenn KI erwähnt wird.

2.2 Biologisches Neuron

Nachdem nun die Definition geklärt ist, ist es sehr hilfreich, für die nächsten Schritte die Funktion des menschlichen Gehirns und dessen Neuronen zu betrachten. Hierfür wird grob die Grundfunktion eines Neurons erklärt. Nervenzellen, auch Neuronen genannt, bilden den Grundbaustein des Nervensystems. Alleine das menschliche Gehirn besitzt rund 10^{11} Neuronen [46].

Ein Neuron ist eine lebende Zelle, welche Informationen in Form von elektrischen Impulsen verarbeitet. Wie in Abbildung 2.1 zu sehen ist hat jedes Neuron einen Zellkörper. Dieser wirkt als Aktivierungsfunktion, wobei anhand der Eingangssignale ein Ausgangssignal berechnet wird. Auf Aktivierungsfunktionen wird später noch genauer eingegangen. Die Dendriten sind Verbindungen zu anderen Zellen und liefern das Eingangssignal zum Zellkörper. Das Axon ist wiederum der Weg des Ausgangssignals und ist über Synapsen wieder mit Dendriten anderer Zellen verbunden. Die Synapsen wirken als gewichtete Verbindungen. Dieser Weg von den Dendriten, über den Zellkörper und weiter zum Axon gibt dem Neuron eine Polarität [30].

Es handelt sich dabei also um ein System, welches ein oder mehrere gewichtete Eingangssignale hat, einen Kern mit einer Aktivierungsfunktion, und ein daraus entstehendes Ausgangssignal. Dies ist wichtig für das Verständnis eines künstlichen Neurons.

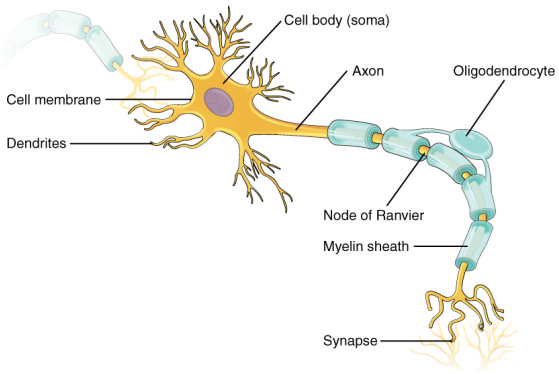


Abbildung 2.1: Aufbau eines Neurons [30]

2.3 Künstliches Neuron

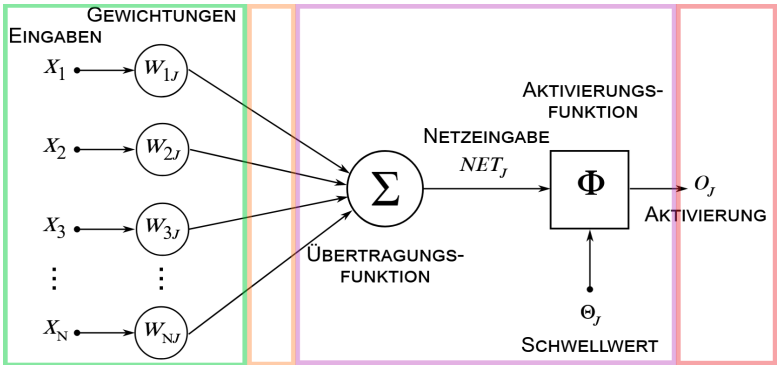


Abbildung 2.2: Modell eines künstlichen Neurons [7]

Das künstliche Neuron ist dem biologischen Neuron nachempfunden und dient als Informationsverarbeitungseinheit. In Abbildung 2.2 ist ein Modell eines künstlichen Neurons dargestellt. Ein oder mehrere Eingangssignale $x_{1..n}$ werden durch die jeweiligen Gewichtungen w_{1j} angepasst und anschließend summiert. Danach kommt noch eine Aktivierungsfunktion Φ, bei der es sich oft um eine Sigmoidfunktion handelt, welche zum Schwellwert Θ schaltet und somit einen binären Ausgang O erzeugt. Die mathematische Formel für dieses Modell lautet wie folgt [7]:

$$O_j = \Phi\left(\Theta_j, \Sigma \sum_{i=1}^{N} w_{ij} x_i\right)$$

Anhand dieser Beschreibung ist ersichtlich, wie das künstliche Neuron von seinem biologischen Gegenstück inspiriert worden ist. Um den Vergleichzu visualisieren wurde die Abbildung 2.2 mit Farben hinterlegt. Der grüne Teil entspricht den Synapsen, Orange den Dendriten, Lila dem Zellkern und Rot symbolisiert das Axon.

2.4 Künstliche Neuronale Netze

Ein einzelnes künstliches Neuron kann nicht viel ausrichten, deshalb werden mehrere zu einem Netzwerk verbunden. Dies nennt man ein künstliches neuronales Netz (KNN).

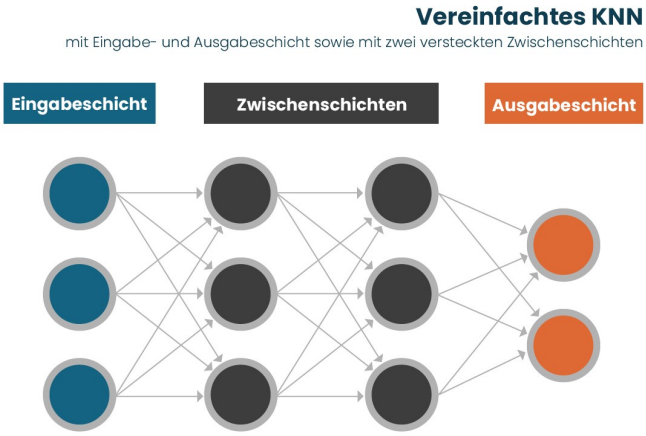


Abbildung 2.3: Graphdarstellung eines vorwärtsgerichteten mehrschichtigen KNN [49]

Typischerweise besteht ein solches Netzwerk aus mehreren Schichten, *Layer* im englischen. Die Abbildung 2.3 zeigt ein mehrschichtiges vorwärtsgerichtetes Netz, die Daten wandern also nur vorwärts in eine Richtung. Die Netzwerke werden meist als gerichtete Graphen visualisiert, wie auch in der Abblidung 2.3 gezeigt ist. Jeder Knoten stellt ein künstliches Neuron dar. Wie ebenfalls zu erkennen ist, werden diese Schichten in drei Kategorien unterteilt. Die Eingabeschicht, *Input Layer*, bildet den Eingang in das System. Danach gibt es ein oder mehrere Zwischenschichten, *Hidden Layers*. Diese sind untereinander verbunden. Die Neuronen in den Zwischenschichten sind der Hauptbestandteil und die einzelnen Gewichtungen und Funktionen der Neuronen bestimmen den Ausgang. Die Ausgabeschicht, *Output Layer*, ist, wie der Name es vermuten lässt, der Ausgang des Systems.

KNNs simulieren durch ihre Struktur die Funktion von biologischen Nervenverflechtungen und somit kann menschliches Denken nachgebildet werden. KNNs sind im Bereich des Machine Learning und des Deep Learning wichtig.

2.5 Maschinelles Lernen

Genau wie KI ist auch maschinelles Lernen, meist *Machine Learning* (ML), ein Sammelbegriff. In Abbildung 2.4 wird der Zusammenhang zwischen KI, dessen Teilbereich Machine Learning, so wie wiederum dessen Teilbereich, das Deep Learning, visuell dargestellt. Auf Deep Learning wird im nächsten Abschnitt noch genau eingegangen. Als Teilbereich von KI fokussiert maschinelles Lernen sich auf die Verwendung von Daten, Algorithmen und KNNs, um Muster und Zusammenhänge aus großen Datenmengen zu erkennen. Dies passiert ähnlich wie beim Menschen, wobei das System lernt und seine

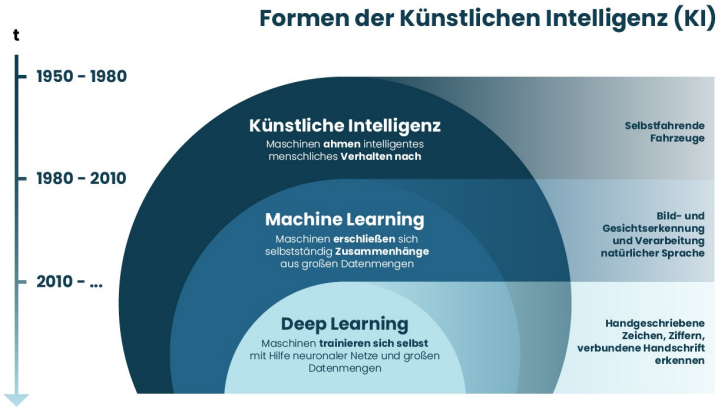


Abbildung 2.4: Zusammenhänge der Terminologien künstliche Intelligenz, maschinelles Lernen und tiefes Lernen [49]

Genauigkeit verbessert. Der Lernvorgang kann dabei in drei Grundkategorien unterteilt werden [17]:

- **Entscheidungsprozess**: Es wird eine Vermutung über die Art des Musters in den Daten aufgestellt, welches gefunden werden soll.
- **Fehlerfunktion**: Die Vermutung wird mit den bekannten Beispielen verglichen und daraus wird der Fehler bestimmt.
- **Aktualisierungs- oder Optimierungsprozess**: Anhand des Fehlers können nun die Parameter des Modells angepasst werden. Somit verkleinert sich der Fehler mit jedem Durchlauf.

Durch diesen Prozess wird nach und nach der Fehler verringert und das Modell optimiert. Es gibt insgesamt vier Ansätze des maschinellen Lernens [17]:

- **überwachtes Lernen (Supervised Learning)**: Der Datensatz ist im Vorhinein vom Benutzer klassifiziert und beschriftet worden, somit kann der Algorithmus bestimmen wie akkurat die Ergebnisse sind.
- **unüberwachtes Lernen (Unsupervised Learning)**: Der Datensatz ist unbeschriftet und der Algorithmus versucht ohne Hilfe des Benutzers, Muster in den Daten zu erkennen.
- **teilüberwachtes Lernen (Semi-Supervised Learning)**: Eine Kombination aus überwachtem und unüberwachtem Lernen. Der Datensatz besteht aus beschrifteten und unbeschrifteten Daten. Dies erlaubt es dem Algorithmus zu lernen und die Daten selbstständig zu beschriften.
- **verstärkendes Lernen (Reinforcement Learning)**: Hier gibt es ein Belohnungs- und Bestrafungssystem, welches dem Algorithmus Feedback gibt. Somit lernt der Algorithmus durch Versuch und Irrtum.

Im Folgenden wird auf die wichtigsten Anwendungsgebiete von maschinellem Lernen

eingegangen [5].

Bei der Klassifikation, oder *classification*, werden Daten einer Klasse beziehungsweise einer Kategorie zugeordnet. Ein gutes Beispiel hierfür ist das Klassifizieren von handgeschriebenen Ziffern [34]. Die Anzahl der Kategorien ist meist sehr gering, kann aber bei Problemen wie Spracherkennung sehr groß werden. Die Regression ist die Vorhersage von stetigen Werten. Diese kann zum Beispiel zur Vorhersage des Preises eines Hauses verwendet werden. Dies funktioniert anhand zuvor bekannter Daten von anderen Häusern. Bei der Gruppierung, oder *clustering*, werden Daten in homogene Teilgruppen unterteilt. Ein Anwendungsfall hierfür ist das Analysieren von sehr großen Datensätzen. Es wird zum Beispiel für das Identifizieren von Fake News oder Spam-Nachrichten verwendet. Die Dimensionsreduktion, oder auch *dimensionality reduction*, wird verwendet um mehrdimensionale Daten in einer niedrigeren Dimensionalität zu repräsentieren. Diese Vorgehensweise erlaubt das Verwenden von Clustering oder Regression. Dimensionsreduktion wird oft zur Vorbereitung der Daten bei Computer Vision Aufgaben verwendet.

2.6 Deep Learning

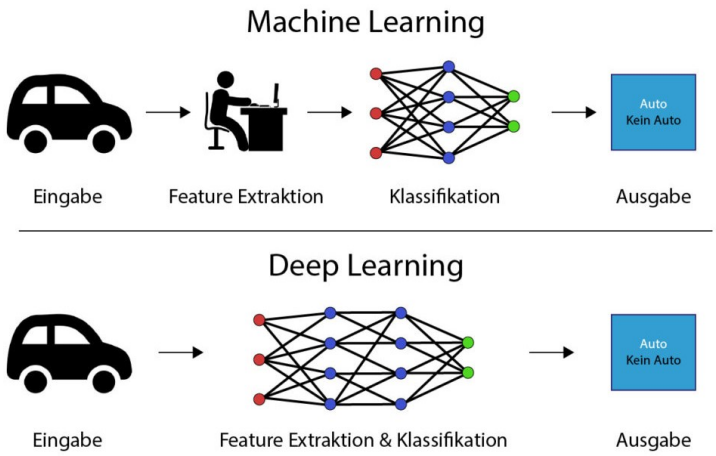


Abbildung 2.5: Vergleich zwischen Deep Learning und klassischem Maschine Learning [32]

Zusätzlich zu den oben genannten Lernmethoden gibt es noch das Konzept des tiefen oder mehrschichtigen Lernens, meist als *Deep Learning* (DL) bekannt. Worin sich tiefes Lernen von klassischem maschinellem Lernen hervorhebt, ist durch die spezielle Nutzung von künstlichen neuronalen Netzen. Mit diesen können riesige Mengen an unstrukturierten Daten verarbeitet werden. Wie in Abbildung 2.5 zu sehen ist, kommt *Deep Learning* ganz ohne vom Menschen bereitgestellte Regeln oder Vorkenntnisse über die Daten aus. Um ein gutes Modell zu erstellen benötigt werden jedoch sehr große Datensätze benötigt. Außerdem ist die Lerndauer ein entscheidender Punkt. Beim maschinellen Lernen

kann der Lernvorgang von wenigen Minuten bis ein paar Stunden dauern. Beim *Deep Learning* hingegen kann es oft Tage, Wochen oder gar Monate dauern um ein brauchbares Modell zu erhalten. Deshalb wird starke Hardware benötigt, meist GPUs, um die Modelle zu trainieren [33].

Eines der fortgeschrittensten Sprachmodelle, GPT-3, hat bis zu 175 Milliarden Parameter. Alleine diese brauchen schon 700 GB an Speicherplatz. Dazu kommt noch der Trainingsdatensatz von circa 20 Terabyte. Mit der derzeit schnellsten GPU auf dem Markt, der Tesla V100, würde es 355 Jahre dauern dieses Modell zu trainieren [40]. Dies ist natürlich ein extremes Beispiel, aber es zeigt, welche riesigen Datenmengen und Rechenkapazitäten gebraucht werden können.

Wie oben erwähnt, kann *Deep Learning* auch als mehrschichtiges Lernen übersetzet werden. Dies bezieht sich auf die Anzahl der Zwischenschichten. Mehrere Schichten erlauben es dem Netzwerk, Daten genauer und robuster zu verarbeiten und somit ihre „Intelligenz“ zu steigern und den Fehler zu verkleinern. Es gibt viele verschiedene *Deep-Learning*-Netzwerke mit unterschiedlichen Strukturen und Anwendungsgebiete.

2.7 Convolutional Neural Network

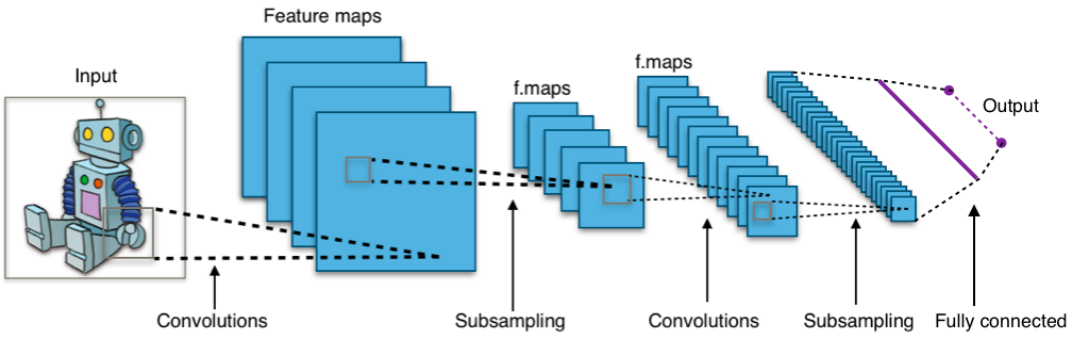


Abbildung 2.6: Abbildung eines typischen Beispiels für ein CNN [16]

Eine der wichtigsten Formen von mehrschichtigen KNNs ist das sogenannte faltende neuronale Netzwerk, allgemein bekannt als *Convolutional Neural Network* (CNN) [29]. CNNs werden am häufigsten für Bildklassifizierungen verwendet. Ein einfaches CNN besteht aus einem *Convolutional Layer* und einem *Pooling Layer*. In der Abbildung 2.6 ist so ein Beispiel eines CNNs mit zwei solcher Paare zu sehen. Hier entspricht das *Subsampling* dem *Pooling*. Komplexere Modelle enthalten mehrere Paare und noch weitere zusätzliche Schichten. Dies erlaubt es mehr Merkmale zu extrahieren. Im Folgenden wird auf die Funktion der wichtigsten Schichten eingegangen [15, 29].

2.7.1 Convolutional Layer

Das Ziel eines *Convolutional Layers* ist es, eine *Feature Map* zu extrahieren und dadurch Merkmale zu erkennen. In der Bildklassifizierung ist der Eingang eines solchen

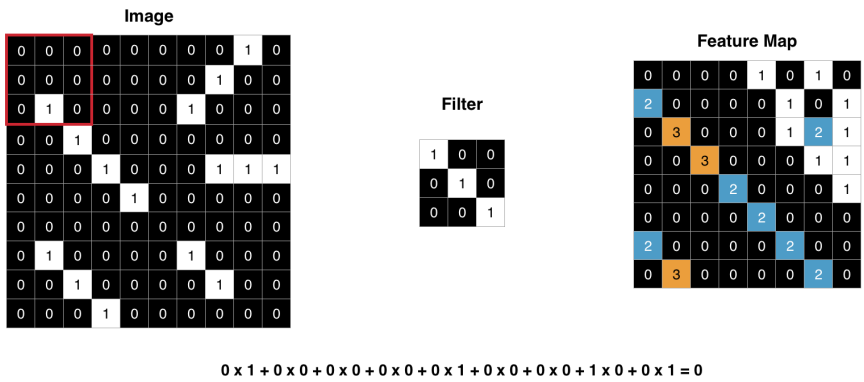


Abbildung 2.7: Extraktion einer Feature Map mittels diskreter Faltung von Bild und Filter [15]

Layers meist eine zwei- oder dreidimensionale Bildmatrix oder eine weitere *Feature Map*. Hierbei wird eine meist sehr kleine Filtermatrix, auch als *Kernel* bekannt, angewendet. Durch eine diskrete Faltung zwischen Eingabematrix und der Filtermatrix entsteht die neue *Feature Map*. In Abbildung 2.7 ist diese Funktion abgebildet. Die Filtermatrix wird dabei über die Bildmatrix gelegt und in dem überlappenden Bereich wird das jeweilige Skalarprodukt gebildet. Die Summe dieser Produkte stellt den Wert für die *Feature Map* dar. Mit der Schrittgröße wird angegeben um wie viele Pixel die Filtermatrix weiterwandert. Somit ist die Größe der resultierenden *Feature Map* w_m von der Größe des Eingänge w_i, der Filtergröße w_f und der Schrittgröße s abhängig [15].

$$w_m = \frac{(w_i - w_f)}{s} + 1$$

Es ist meistens der Fall, dass nicht nur ein Filter pro Faltung angewendet wird. Es werden mehrere unterschiedliche Filtermatrizen verwendet um jeweils verschiedene Merkmale zu extrahieren.

2.7.2 Pooling Layer

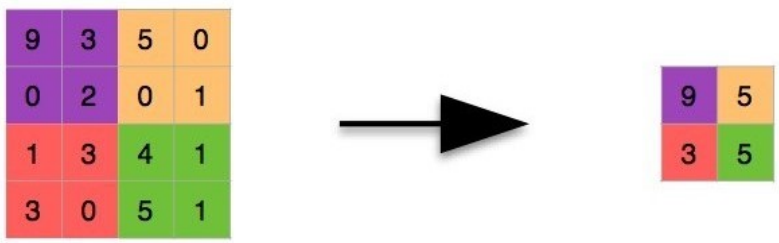


Abbildung 2.8: Max Pooling einer Feature Map [15]

Der *Pooling Layer* verwirft Informationen und behält nur die wichtigen Werte. Dadurch kann das CNN-Merkmale besser erkennen, unabhängig davon, wo sie gefunden worden sind. Zusätzlich verringert dieser Vorgang den Speicherbedarf eines Netzwerkes und erhöht dessen Verarbeitungsgeschwindigkeit. Es gibt mehrere Arten von *Pooling*, wobei die am weitesten verwendete Art das *Max Pooling* ist. Hierbei wird die Eingabematrix in Untermatrizen unterteilt und von diesen nur der höchste, also maximale Wert, übernommen. Dieser Prozess ist vergleichbar mit dem des *Conolutional Layer*, nur dass statt der diskreten Faltung der Maximalwert übernommen wird. In Abbildung 2.8 ist dieser Vorgang farblich dargestellt. Somit ist die Ausgabematrix dieser Schicht wieder von der Eingabegröße, der Bereichsgröße in welcher der Wert gesucht wird und der Schrittgröße abhängig [15].

2.7.3 Rectified Linear Unit

Die ReLU-Schicht, kurz für *Rectified Linear Unit*, verwendet einen Gleichrichter und folgt meistens einem *Convolutional Layer* oder einem *Pooling Layer*. Diese Schicht erlaubt es einem CNN, nichtlineare Beziehungen zwischen Ein- und Ausgang zu berücksichtigen. Hierbei werden alle negativen Werte zu null und die positiven Werte bleiben unverändert. Beschreiben lässt sich dieses Verhalten durch folgende mathematische Formel, wobei x der Eingangswert eines künstlichen Neurons ist [15]:

$$f(x) = max(0, x)$$

In Abbildung 2.9 ist diese Funktion gut visualisiert. Es ist zu erkennen wie alle negativen Werte auf null gesetzt werden.

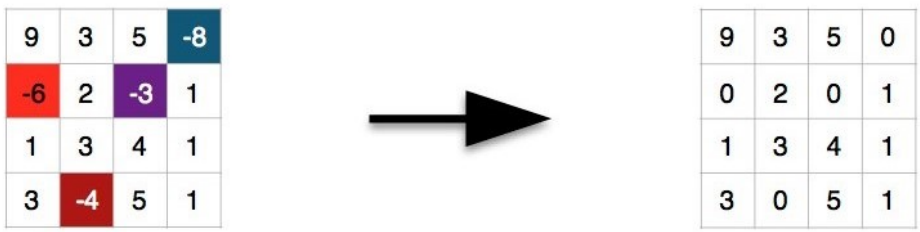


Abbildung 2.9: Gleichrichten einer Feature Map mittels ReLU [15]

2.8 Trainieren und Testen eines Modells

Sobald ein Modell erstellt wurde und ein genügend großer Datensatz zur Verfügung steht, kann das Modell trainiert werden. Wie das Training abläuft, wird durch Hyperparameter festgelegt. Die wichtigsten sind *Epoch* und *Batch*. Ein *Epoch*, also eine Epoche, ist ein Hyperparameter welcher angibt, wie oft das gesamte Modell den gesamten Datensatz durcharbeitet. Ein *Batch*, also eine Ladung, gibt an, wie viele Daten eines Datensatzes abgearbeitet werden, bevor die Modellparameter angepasst werden.

Diese Anpassung geschieht durch eine sogenannte *loss function*, zu deutsch eine Verlust-Funktion. Diese wird je nach Anwendungsfall unterschiedlich gewählt und bewirkt die Optimierung des Modells [18].

Das Testen eines Modells wird mit einem ihm unbekannten Datensatz durchgeführt. Die Genauigkeit des Modells wird dann dadurch bestimmt, wie viele Daten richtig bestimmt worden sind und wie viele falsch. Der Grund für ein schlechtes Ergebnis ist meist die Über- beziehungsweise Unteranpassung, bekannt als *overfitting* und *underfitting*. Überanpassung geschieht, wenn das Modell zulange mit denselben Daten lernt. Somit erlernt es jedes Detail dieses Datensatzes und verliert die Fähigkeit zur Verallgemeinerung. Es kann dadurch nur schlecht auf neue Daten reagieren. Der Grund für Überanpassung ist meistens das zu lange trainieren und kann durch einen Vorgang namens *early stopping*, also verfrühtes Stoppen, verhindert werden. Das Gegenteil dazu ist die Unteranpassung. Hierbei gelingt es dem Modell nicht, die Beziehung zwischen Ein- und Ausgang herzustellen und dadurch entsteht eine sehr hohe Fehlerrate. Die Hauptgründe dafür sind meistens eine zu geringe Trainingsdauer und zu wenige Trainingsdaten [18, 29].

Ein Modell zu trainieren ist ein sehr komplexer und aufwendiger Vorgang, bei dem die Modellstruktur, die Daten, und der Lernprozess immer wieder angepasst werden müssen, um ein optimiertes Modell zu erhalten. Wie Modelle trainiert und verwendet werden und welche Bereiche für diese Arbeit wichtig sind, wird in den folgenden Abschnitten erklärt.

2.9 Bildklassifizierung

Um eine Gestensteuerung zu ermöglichen, müssen die Bilder der Kamera verarbeitet und erkannt werden. Deshalb wird in diesem Abschnitt genauer auf die Bilderkennung und Bildklassifizierung mittels KI eingegangen.

Wenn Menschen ein Bild sehen oder ein Video schauen, können sie ohne Mühe Menschen, Objekte, Situationen und viele andere Details erkennen. Das Ziel der Bilderkennung ist es, dies dem Menschen nachzuahmen und einer KI, also einem Computer, beizubringen, Bilder und deren Details zu erkennen [36]. In der Bilderkennung gibt es viele Teilbereiche, zum Beispiel die Objektlokalisierung oder die Objektsegmentierung. Der für diese Arbeit wichtige Teil ist jedoch die Objektklassifizierung.

Die Bildklassifizierung, oder auch Objektklassifizierung, wird verwendet, um Bilder und Bildausschnitte in Kategorien einzuordnen. Dabei können ganze Bilder kategorisiert werden oder nur einzelne Objekte. In Abbildung 2.10 ist ein Beispiel dafür zu sehen. Es wird eine Klassifizierung mit den Kategorien Katze und Hund durchgeführt. Das Ergebnis ist das es mit einer Wahrscheinlichkeit von 97% eine Katze ist und zu 3% ein Hund. Damit so eine Klassifizierung erfolgen kann, muss das Bilderklassifizierungsmodell lernen . Dazu benötigt es einen großen Datensatz. Wenn zu wenige Daten vorhanden sind, kann das Modell allgemeine Muster zwischen den Daten nicht erkennen. Dies führt zu einem schlechten Ergebnis.

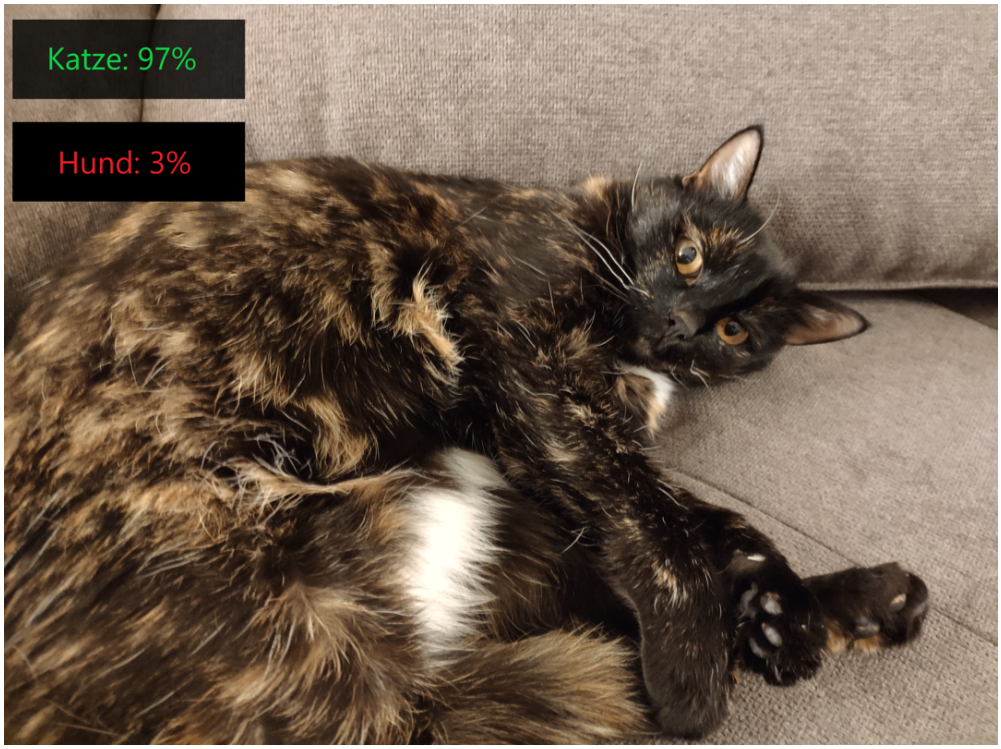


Abbildung 2.10: Bildklassifizierung zwischen Katze und Hund

Der Grundablauf eines Systems zur Bildklassifizierung beginnt mit einem einzelnen Bild. Dieses wird meistens vorverarbeitet. Dabei wird zum Beispiel das Format des Bildes angepasst, etwaige Filter werden angewendet und das Bild wird zu Graustufen umgewandelt. Somit werden die Daten normiert und sind dadurch leichter zu verarbeiten. Dadurch steigen die Geschwindigkeit und die Genauigkeit des Modells. Sobald die Vorverarbeitung beendet ist, versucht das Modell verschiedenste Merkmale und Besonderheiten aus den Bildern zu extrahieren und anhand dieser Parameter eine Klassifizierung durchzuführen. Der Ausgang des Systems ist eine Liste aus Wahrscheinlichkeiten. Diese geben an wie sicher sich das Modell ist, dass es zu einer Kategorie gehört. Das Ergebnis ist die Kategorie mit der höchsten Wahrscheinlichkeit. Im Beispiel der Abbildung 2.10 wäre das Ergebnis eine Katze.

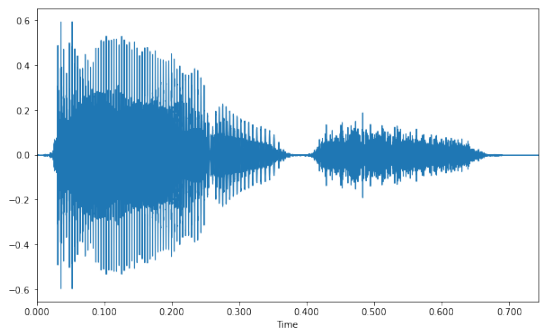
2.10 Spracherkennung

Spracherkennung ist das automatische beziehungsweise maschinelle Erkennen gesprochener Sprache [21]. Es gibt verschiedene Ansätze, ein Spracherkennungssystem zu erstellen. Ein System wäre zum Beispiel einen großen Wortschatz zu erstellen. Darin befinden sich alle oder zumindest die wichtigsten Wörter einer Sprache. Da es jedoch viele Wörter gibt, die je nach grammatikalischer Funktion eine unterschiedliche Bedeutung haben,

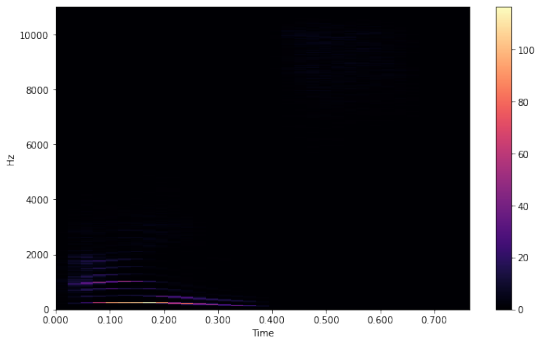
müssen diese ebenso in den Wortschatz aufgenommen werden. Homofone, also Wörter die gleich ausgesprochen werden, aber unterschiedliche Bedeutungen haben, erschweren dies noch zusätzlich. Ein weiterer Ansatz ist es, ein akustisches Modell zu erstellen, welches die Vokale und Konsonanten erkennt. Anhand eines Wortschatzes und eines Sprachmodells können sinnvolle zusammenhängende Sätze erkannt werden. Diese Ansätze fallen in den Bereich des *natural language processing* (NLP), also dem Verarbeiten natürlicher Sprache.

Für diese Arbeit genügt es jedoch zwischen wenigen Sprachbefehlen unterscheiden zu können. Deshalb wird hier erklärt, wie das Erkennen einzelner Wörter funktioniert. In der Abbildung 2.11 ist der Vorgang zum Erstellen eines sogenannten Mel-Spektrogramms anhand einer gesprochenen 'Eins' zu sehen. Diese Bilder wurden mithilfe eines Programms von Ketan Doshi erzeugt [20].

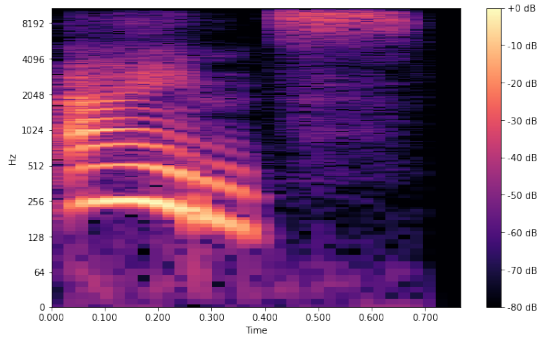
In der Abbildung 2.11a ist zu sehen, wie ein eingelesenes Wort als Schwingungsverlauf der Amplitude in Abhängigkeit der Zeit visualisiert werden kann. Diese Ansicht eig-



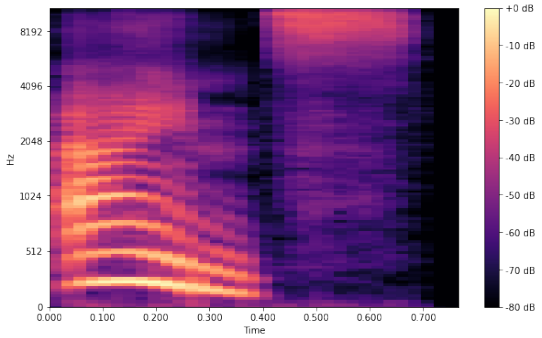
(a) Amplituden der Eingangsschwingungen in Abhängigkeit der Zeit



(b) Spektrogramm nach der Fourier-Transformation. Die Färbung beschreibt die Amplitude



(c) Spektrogramm bei dem Frequenz in logarithmischer Skalierung und Amplitude in Dezibel angezeigt wird



(d) Mel-Spektrogramm. Frequenz in Mel-Skalierung

Abbildung 2.11: Schritte zum Erstellen eines Spektrogramms für die Bildklassifizierung anhand des Beispiels einer gesprochenen 'Eins'

net sich jedoch nicht für eine Klassifikation, da die einzelnen Frequenzen nicht erkennbar sind. Darum wird das Signal mithilfe der Fourier-Transformation in den Frequenzbereich gebracht. In Abbildung 2.11b ist das Spektrogramm im Frequenzbereich zu sehen. Die Färbung beschreibt die Amplitude, ist jedoch auf diesem Bild nur schwer im unteren Bereich zu erkennen. Um diese besser zu visualisieren ist es hilfreich die Skalierung der Frequenz logarithmisch darzustellen und die Amplitude in Dezibel umzurechnen. Diese neuskalierte Betrachtung wird in Abbildung 2.11c gezeigt. Somit wird eine sehr gute Visualisierung des Signals erreicht. Mit dieser Darstellung kann ein Bildklassifizierungsmodell trainiert werden und somit Töne klassifizieren. Allerdings hat sich gezeigt, dass es noch eine bessere Darstellung gibt.

Menschen nehmen Töne logarithmisch war. Also können sie zum Beispiel den Unterschied zwischen den tieferen Noten C2 und C4 (65 - 262 Hz) nicht hören. Zwischen den höheren Noten G6 und A6 (1568 - 1760 Hz) kann man jedoch deutlich einen Unterschied erkennen, obwohl beide Beispiele circa 200 Hz auseinanderliegen. Mit der Mel-Skalierung, Mel von dem englischen Wort *melody*, kann der wahrgenommene Ton beschrieben werden. Somit kann man die ursprüngliche Frequenz umrechnen und in Mel-Frequenz darstellen. Der Unterschied zwischen 100 und 110 Mel ist gleich wie der zwischen 1000 und 1010 Mel. Die Frequenzen sind verschieden aber die Wahrnehmung ist gleich [10]. Die Umrechnungen erfolgt wie folgt:

$$m = 2595 \cdot log(1 + \frac{f}{500})$$

$$f = 700 \cdot (10^{\frac{m}{2595}} - 1)$$

Das Mel-Spektrogramm ist nicht nur geeignet Töne zu visualisieren um sie für Menschen verständlich zu machen, sondern ist auch im Bereich der Tonklassifizierung zu bevorzugen. Es wurde gezeigt, dass Mel-Spektrogramme generell bessere Ergebnisse erzielen als normale Spektrogramme [2].

Die Klassifizierung von Wörtern funktioniert also indem man Töne mittels Spektrogrammen visualisiert und anhand dieser Bilder einfache Bildklassifizierung durchführt. Im nächsten Kapitel wird der Stand der Technik betrachtet und es werden mögliche Lösungsansätze für die Aufgabenstellung dieser Arbeit gegenübergestellt.

Kapitel 3

Stand der Technik

In diesem Kapitel wird die Vorarbeit und der Stand der Technik betrachtet. Es werden verschiedene Ansätze, Frameworks und Bibliotheken gegenübergestellt und verglichen. Dabei wird zuerst auf Ansätze für die Lösung der Gestensteuerung eingegangen und im Anschluss werden Lösungen für die Sprachsteuerung betrachtet. Am Ende jedes Teils werden die Lösungsansätze verglichen und es wird einer der Ansätze gewählt. Der gewählte Ansatz wird anschließend im nächsten Kapitel behandelt.

3.1 Gestensteuerung

Im Folgenden werden vorhandene Ansätze gegenübergestellt, welche eine Steuerung per Gestik ermöglichen. Dafür wird ein System benötigt, welches es erlaubt mindestens eine Hand zu lokalisieren und deren Gestik zu bestimmen. Bevorzugt wäre eine Lösung, die ohne Internetanbindung und auf eingebetteten Systemen verwendet werden kann.

3.1.1 MediaPipe

MediaPipe ist ein Cross-Plattform-Framework erstellt von Google. Dabei werden mehrere der modernsten ML-Modelle zur Verfügung gestellt. Unter anderem sind Modelle zur Gesichtserkennung, Haarsegmentierung, Objekterkennung und -verfolgung und Posenerkennung verfügbar. Viele dieser Modelle werden in Google-Produkten des täglichen Gebrauchs verwendet, wie zum Beispiel Gmail, Lens, Maps und Youtube [23].

Der für diese Arbeit interessante Teil ist jedoch die Handerkennung und -verfolgung. Dabei ist es möglich die Form und die Position von mehreren Händen gleichzeitig zu detektieren. Die sogenannte MediaPipe Hands *Pipeline* besteht aus mehreren Modellen. Ein Handflächendetektor erkennt die Position und Orientierung der Hand. Auf diesem Bildausschnitt erkennt dann ein weiteres Modell die sogenannten *Landmarks* einer Hand, also Punkte wie Fingerspitzen und Fingeransätze [24]. Diese *Landmarks* sind in der Abblidung 3.1 als Rote Punkte dargestellt und für die bessere Visualisierung mit Strichen verbunden. Es werden insgesammt 21 *Landmarks* detektiert und deren dreidimensionalen Koordianten erkannt. Mit diesen *Landmarks* und deren Koordinaten ist es möglich sehr komplexe Gesten zu bestimmen.

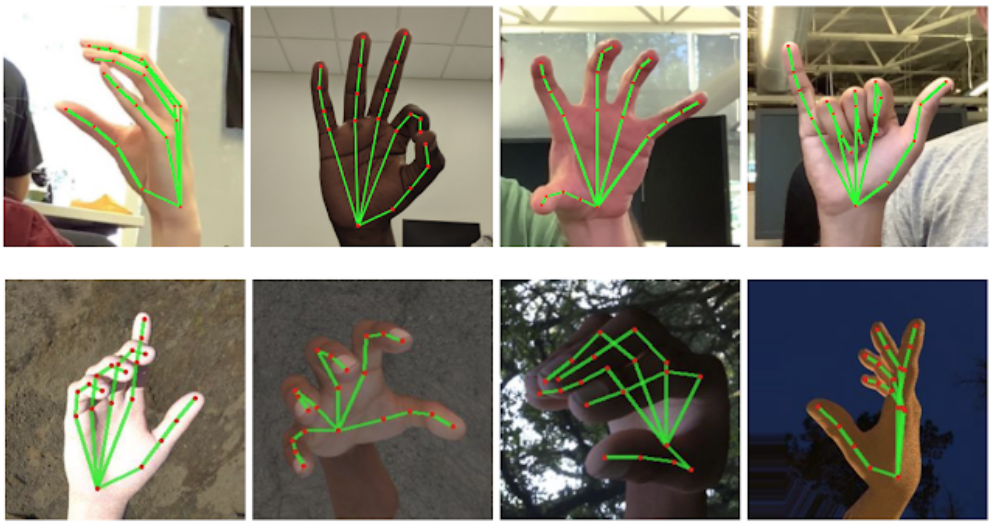


Abbildung 3.1: Ergebnis von MediaPipe Hands[24]

MediaPipe Hands basiert auf Tensorflow und funktioniert ohne Internetverbindung. Es ist verfügbar für Android, iOS, C++, Python, und JavaScript [23].

3.1.2 ML5 Handpose

Die Open-Source-Bibliothek ml5.js macht maschinelles Lernen leichter zugänglich für Künstler, Programmierer und Schüler. Sie basiert auf Tensorflow.js und erlaubt es, ML-Algorithmen und -Modelle im Browser zu verwenden [37]. ml5.js ist für p5.js geeignet. p5.js ist eine JavaScript-Bibliothek welche den Fokus darauf legt, Programmieren für jeden zugänglich zu machen. Dabei stehen die einfache Verwendung und die mächtigen Umsetzungsmöglichkeiten im Vordergrund [42]. ml5.js bietet viele verschiedene vortrainierte Modelle, sowie die Möglichkeit eigene Modelle zu trainieren. Unter den bereits vortrainierten Modellen befinden sich Modelle zur Posen- und Gesichtserkennung, zur Körpersegmentierung oder auch ein StyleTransfer, welches es erlaubt den Stil eines Bildes auf ein anderes zu übertragen.

Der für uns wichtige Teil wird *ML5 Handpose* genannt und bestimmt wie der Name sagt die Position der Hand. Dabei wird aber nicht nur die Position der gesamten Hand angegeben sondern es werden wieder 21 dreidimensionale *Landmarks* detektiert und verfolgt. ml5.js ist auch ohne Internetverbindung verwendbar, ist jedoch nur für JavaScript verfügbar. Um JavaScript ohne Internetverbindung und auf eingebetteten Systemen verwenden zu können, kann Node.js verwendent werden [38].

3.1.3 OpenCV

Ein weiterer Ansatz wäre ganz auf KI zu verzichten und mit der Bibliothek OpenCV ein Gestenerkennungsmodell zu erstellen. Dabei müssen Segmentierung und verschiedene Methoden angewendet werden um diese Aufgabe zu meistern. Das Ergebnis eines solchen

Modells ist jedoch nicht perfekt und den anderen vorgestellten Ansätzen unterlegen, sowohl in Genauigkeit als auch in Flexibilität. zusätzlich wird die Hand nicht lokalisiert, sondern muss in einen vorgegebenen Bereich gehalten werden [11, 41].

3.1.4 Fazit

Die Methode mit OpenCV scheitert an der mangelnden Genauigkeit und dem großen Aufwand. Die anderen vorgestellten Methoden sind dieser in fast allen Punkten überlegen. Mit der einfachen Anwendung, den 21 *Landmarks* und dem Lokalisieren der Hand sind Mediapipe und ml5.js die beste Wahl. Mediapipe ist sowohl für Python als auch für JavaScript verfügbar. Falls die Umsetzung mit Python nicht oder nur schwer möglich ist, kann zu Javascript gewechselt werden. In diesem Fall werden sowohl Mediapipe als auch ml5.js getestet und gegenübergestellt. Das bessere Modell wird dann verwendet. Da nun die Grundlage für die Bildklassifizierung gegeben ist, werden im nächsten Schritt die Möglichkeiten zur Sprachsteuerung betrachtet.

3.2 Sprachsteuerung

Im Folgenden werden vorhandene Ansätze, Frameworks und Bibliotheken gegenübergestellt welche eine Steuerung mittels Sprache ermöglichen. Es wird eine Lösung gesucht, welche es erlaubt ein eigenes System zu erstellen. Dieses System soll zwischen den zehn Ziffern, den Grundoperatoren und Hintergrundgeräuschen unterscheiden können. Die Sprache die unterstützt werden soll, ist Deutsch oder Englisch. Ebenso wie bei der Gestensteuerung ist ein System zu bevorzugen, welches ohne Internet und auf eingebetteten Systemen verwendet werden kann. Um die spätere Kombination von Gesten- und Sprachsteuerung zu gewährleisten ist es wichtig, dass die beiden Ansätze kompatibel sind, also zum Beispiel beide in Python oder beide in JavaScript implementiert werden.

3.2.1 Google Speech-to-Text

Wie der Name es vermuten lässt ist *Speech-to-Text* (STT) eine Möglichkeit, Sprache zu Text zu transkribieren. Google bietet so einen Ansatz indem eine API für ein STT-Modell angeboten wird [25]. Viele dieser Sprache-zu-Text-Lösungen sind etwas zu mächtig für diese Aufgabenstellung und der Großteil der bereitgestellten Möglichkeiten würde nicht genutzt werden. So würden alle gesprochenen Sätze transkribiert und man müsste die benötigten Wörter herausfiltern. Dies stellt einen zusätzlichen Verarbeitungsschritt und somit eine Verzögerung dar.

Google STT und weitere ähnliche Spracherkennungsbibliotheken sind in der Python-Bibliothek SpeechRecognition zusammengefasst. Darin sind unter anderem auch Modelle von Microsoft Bing, IBM und Wit.at vorhanden [14]. Da die meisten dieser Systeme jedoch nur mit Internetverbindung zu verwenden sind, kann es zu einer Geschwindigkeitsreduktion führen.

3.2.2 Vosk

Vosk ist eine offline open-source Spracherkennungs-API für Android, iOS, Raspberry Pi und ist für Programmiersprachen wie Python, Java, C#, C++ und Node.js implementiert. Sie unterstützt über neunzehn Sprachen, darunter auch Deutsch und Englisch. Es handelt sich dabei um sehr kleine Modelle, rund 50 Mb, und ist somit sehr gut für eingebettete Systeme geeignet [47]. Die Nachteile sind ähnlich wie bei den oben erwähnten STT Modellen. Es wird nur ein minimaler Teil der erkannten Wörter gebraucht und somit ist der Großteil der Berechnungen ungenutzt. Der Vorteil ist ganz klar die kompakte Größe der Modelle und deren Möglichkeit diese ohne Internet und auf eingebetteten Systemen zu verwenden.

3.2.3 Picovoice

Picovoice ist eine Webplattform welche es erlaubt vollständige Sprachprodukte zu entwickeln. Dabei funktioniert es im Gegenteil zu Alexa und Google ohne Internetverbindung direkt auf dem Gerät. Es kann auch auf eingebetteten Systemen wie dem RaspberryPi oder einem STM32-Mikrocontroller laufen [44].

Picovoice wird in zwei Teile unterteilt. Der eine ist Porcupine. Porcupine ist eine Weckworterkennung. Dies erlaubt es ein eigenes Weckwort zu trainieren. Ein Weckwort wird als Befehl verwendet, er aktiviert das aktive zuhören des Modells. Als Beispiel ist 'Hey Google' bei Google Home oder 'Alexa' bei Amazon Alexa das Weckwort. Eine Idee wäre es also für diese Arbeit 15 Weckworterkennungmodelle zu verwenden. Somit könnte man zum Beispiel erkennen wenn eine Ziffer oder ein Operator gesagt wird. Das Problem, dass dabei auftritt ist, dass kurze einsilbige Wörter wie 'eins','zwei', oder auch 'mal', nur sehr schlecht zu unterscheiden und zu erkennen sind. Daher eignen sich diese nicht als Weckwörter. Der zweite Teil ist Rhino. Dies ist eine Anwendung die es erlaubt die Intention hinter gesprochenen Sätzen zu verstehen. Als Ergebnis erhält man also keinen Text wie bei den STT-Modellen, sondern ein Intentionsmodell im JSON-Format. Die Kombination aus Weckworterkennung und Intentionserkennung bildet ein vollständiges Sprachsteuerungssystem. Unterstützt werden Systeme von Python, Java, Flutter, Android, iOS, Node.js, C und noch viele weitere. Ein Nachteil sind jedoch die Kosten sein eigenes Weckworterkennungsmodell zu erstellen [44].

3.2.4 Eigenes Modell erstellen

Falls keines der gefundenen Frameworks in Frage kommt bleibt noch die Möglichkeit ein eigenes Modell zu erstellen. Das Hauptproblem beim Erstellen eines eigenen Modells ist der benötigte Datensatz. Um ein robustes System zu erstellen wird eine große Anzahl and Daten benötigt. Es könnte durchaus bereits einen Datensatz geben mit dem man eigene Modelle trainieren kann und bei dem es sich um die Daten handelt die für diese Arbeit benötigt werden. Wenn diese Daten vorhanden sind kommt der nächste Teil, das Vorbereiten der Daten. Wie bereits in den Grundlagen erklärt wurde, funktioniert die Klassifizierung von Tönen genauso wie von Bildern. Die Töne werden also in Form von Spectrogrammen visualisiert und können so zum Trainieren verwendet werden. Der nächste Schritt ist die Erstellung eines CNN-Modells. Die Schwierigkeit ein CNN-Modell

zu erstellen, ist durch die vielen Variablen jedoch sehr hoch. Der Vorteil beim Erstellen eines eigenen Modells ist, dass es genau an das Problem angepasst werden kann. Somit können die Vorgaben perfekt erfüllt werden und unnötige Klassifizierungen und Zwischenberechnungen vermieden werden.

3.2.5 Fazit

Google STT und Vosk wären eine Option, jedoch sind diese Frameworks fast zu mächtig. Das Ergebnis der STT-Lösungen müsste erst gefiltert werden, um die Sprachbefehle zu extrahieren. Somit ist der Großteil der Eingangsdaten nutzlos. Dieser Mehraufwand wirkt sich auch negativ auf die Geschwindigkeit des Modells aus. Picovoice ist nicht für diese Arbeit geeignet, da einsilbige Sprachbefehle nicht oder nur schlecht funktionieren. Auf dieses Problem wurde aber schon genauer eingegangen. Somit bleibt der Lösungsansatz ein eigenes Modell zu erstellen.

Das Hauptproblem beim Erstellen eines eigenen Modells ist der benötigte Datensatz. Um ein robustes System zu erstellen wird eine große Anzahl an Daten benötigt. Optimal wäre es wenn es schon einen Datensatz geben würde, bei dem es sich um genau die Daten handelt, die für diese Arbeit benötigt werden. Falls dem nicht so ist muss ein eigener Datensatz erstellt werden. Sobald die Daten vorhanden sind kommt der nächste Teil, das Vorbereiten der Daten. Wie bereits in den Grundlagen erklärt wurde, funktioniert die Klassifizierung von Tönen genauso wie von Bildern. Die Töne werden also in Form von Spektrogrammen visualisiert und können so zum Trainieren verwendet werden. Der nächste Schritt ist die Erstellung eines CNN-Modells. Die Schwierigkeit ein CNN-Modell zu erstellen, ist durch die vielen Variablen und Einstellmöglichkeiten sehr hoch. Der Vorteil beim Erstellen eines eigenen Modells ist, dass es genau an das Problem angepasst werden kann. Somit können die Vorgaben perfekt erfüllt und unnötige Klassifizierungen und Zwischenberechnungen vermieden werden.

3.3 Zusammenfassung

Mit dem Vergleich der verschiedenen Ansätze zur Gestensteuerung wurde entschieden, dass diese mit MediaPipe Hands implementiert wird. Für die Sprachsteuerung wurde kein optimales Framework gefunden und deshalb wird ein eigenes Sprachklassifizierungsmodell erstellt. Sobald diese zwei Teile fertig implementiert und getestet sind, können diese zu einem fertigen Prototypen kombiniert werden.

Kapitel 4

Gesamtbild des Systems

Bevor die Implementierung des Prototypen behandelt wird, ist es hilfreich sich einen Überblick über das Gesamte System zu verschaffen. In diesem Kapitel wird deshalb der Aufbau des gesamten Systems beschrieben.

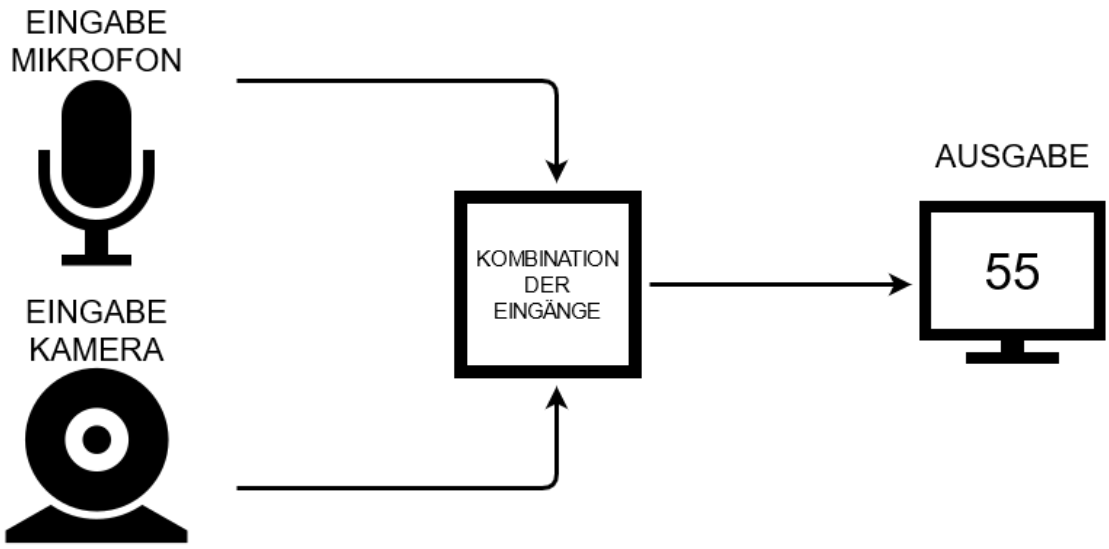


Abbildung 4.1: Aufbau des Prototypen

In der Abbildung 4.1 wird das Gesamtbild des zu erstellenden Prototypen dargestellt. Auf der linken Seite befinden sich die Eingänge in das System. Beide Eingänge beschränken sich auf die Klassifizierung der zehn Ziffern, der Grundrechenoperatoren für die Addition, Subtraktion, Multiplikation und Division sowie einem Gleichheitszeichen. Die Sprachsteuerung wird über ein Mikrofon funktionieren. Als Mikrofon wird dafür ein externes Headset verwendet welches per USB an den Laptop angeschlossen ist. Bei der Aufnahme der Sprache ist darauf zu achten das möglichst wenige Nebengeräusche vorhanden sind. Das Headset-Mikrofon bietet eine sehr gute Aufnahmequalität und ermöglicht es so, schöne Spektogramme für die Klassifizierung zu erzeugen. Für die Gestensteuerung wird eine Kamera benötigt. Hierfür wird die interne Webcam des Lap-

tops verwendet. Die Qualität der Aufnahme ist mit der Webcam mehr als ausreichend und bietet sich somit gut an. Hierbei ist bei der Aufnahme speziell auf den Hintergrund und auf die Beleuchtung zu achten. So wird ein einfarbiger Hintergrund und eine gute indirekte Beleuchtung bevorzugt. Diese Kriterien erlauben es die Hand und deren Gestik leichter und genauer zu erkennen. Die erkannten Sprachbefehle und Gesten werden anschließend zum nächsten Teil weitergereicht.

In der Mitte der Abbildung 4.1 ist die Kombination dieser erkannten Daten dargestellt. Diese Box symbolisisert ein System mit einem Algorithmus welcher die Eingänge auswertet und dynamisch auswählt, welcher verwendet werden soll. Für diese Auswertung der Eingänge wird es nötig sein eine Priorität festzulegen für den Fall, dass beide Eingänge gleichzeitig aktiv sind. Welcher der beiden Eingänge höhere Priorität erhält ist nicht wichtig. Wichtig ist nur das entschieden wird welcher in so einem Fall verwendet wird. In diesem Algorithmus passiert auch die Auswertung der Rechnung, sobald diese fertig eingegeben wurde.

Rechts in der Abbildung 4.1 ist der Ausgang des Prototypen dargestellt. Hierbei wird eine Möglichkeit zur Visualisierung der Ergebnisse benötigt. Da schon die interne Webcam des Laptops verwendet wird, bietet sich der Laptop-Bildschirm für diese Ausgabe an. Auf diesem werden die erkannten Steuerungsbefehle sowie die Rechnung angezeigt. Dieser letzte Schritt bildet das Ende des Systems. Somit ist der Prototyp vollständig definiert und die Implementierung kann beginnen.

Kapitel 5

Implementierung

Nachdem nun das Gesamtbild des Systems betrachtet wurde, kann die Implementation des Taschenrechnerprototypen beginnen. Da die Erstellung eines eigenen CNN-Modells meistens in Python geschieht, wird diese Implementation ebenfalls in Python umgesetzt [19]. Als Erstes wird die Gestensteuerung behandelt. Dabei wird gezeigt, wie MediaPipe verwendet und anschließend getestet wird. Anschließend wird die Sprachsteuerung behandelt. Dabei wird der komplette Vorgang zum Erstellen eines eigenen Modells beschrieben und dieses Modell auch anschließend getestet. Zum Schluss kommt die Kombination der beiden Steuerungen und der Test des Gesamtsystems.

5.1 Gestensteuerung

5.1.1 Vorverarbeitung der Eingangsdaten

Bevor die einzelnen Bilder der Kamera klassifiziert werden können ist es wichtig die Bilder vorzubereiten. Mit OpenCV2 ist es möglich Videos aus verschiedenen Quellen auszulesen. Welche Kamera als Eingang gewählt wird, kann als Index mitgegeben werden. Der Index null ist für die Standardkamera. Im Falle eines Laptops ist diese die Webcam. OpenCV2 liest Bilder standardmäßig im BGR-Format ein, also Blue-Green-Red. OpenCV2 erlaubt es mit einer Funktion das Bild vom BGR- ins RGB-Format zu konvertieren [41]. Diese Vorverarbeitung muss gemacht werden, da MediaPipe Hands ein RGB-Bild erwartet [24].

5.1.2 Ergebnis von MediaPipe Hands

MediaPipe Hands liefert als Ausgabe eine Liste der erkannten Hände. Jede erkannte Hand ist wiederum eine Liste bestehend aus den 21 *Landmarks*. In Abbildung 5.1 ist eine Übersicht der einzelnen *Landmarks* dargestellt. Jede *Landmark* hat drei Variablen. Diese drei Variablen sind x, y und z und speichern die dreidimensionalen Koordinaten der *Landmarks*. Die Koordinaten sind die Pixelkoordinaten des Eingangsvideos. Dabei ist der Ursprung links oben im Bild.

Mit dem *Landmark* und dem Index 8 können zum Beispiel die Koordinaten der Fingerspitze des Zeigefingers gelesen werden. Wenn diese nun mit den Koordinaten des

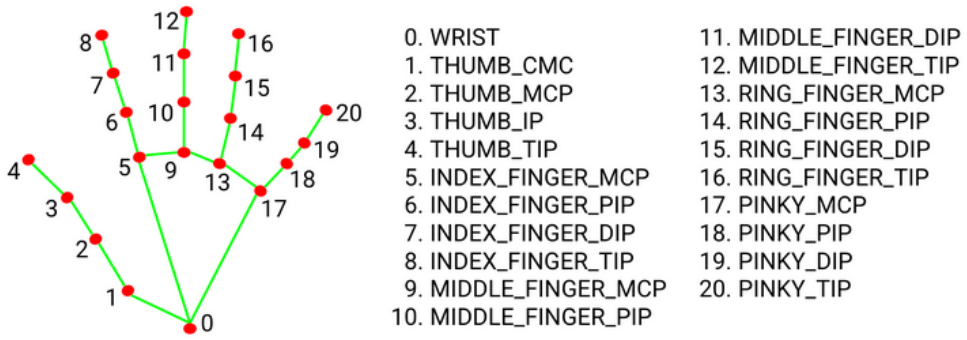


Abbildung 5.1: Übersicht der einzelnen *Landmarks* und deren Indizes über die darauf zugegriffen werden kann [26]

Fingeransatzes, also Index 5, verglichen werden, kann man bestimmen ob der Zeigefinger gestreckt ist oder nicht. Mit diesem Vorgang ist es möglich alle Gesten für dieses Projekt zu bestimmen. Die genauen dreidimensionalen Koordinaten erlauben es jedoch noch viel komplexere Gesten zu erkennen, besonders beim verwenden von zwei Händen oder mehr Händen gleichzeitig. Für diese Arbeit reicht es aber eine Hand zu verwenden.

5.1.3 Tests und Fazit

Erste Tests zeigen, wie einfach die Anwendung ist und wie genau die Erkennung der Hände ist. Dieser Ansatz ist für die Verwendung der Gestensteuerung des Taschenrechners gut geeignet. Wie die Verwendung der Gestensteuerung funktioniert, wird bei der Kombination der beiden Teilbereiche beschrieben. Auf die weiteren Tests wird im Kapitel 7 eingegangen.

5.2 Sprachsteuerung

5.2.1 Datenbeschaffung

Das Hauptproblem von KNNs ist die Datenbeschaffung. Die einfachste Methode ist es, bereits existierende Datensätze zu verwenden. Es wurden Datensätze gefunden, welche die Ziffern 0-9 beinhalten, aber zu den benötigten Operatoren wurde nichts passendes gefunden. Die persönliche Sammlung der Audiobeispiele würde sehr viele unterstützende Personen benötigen. Jeder von ihnen müsste ein oder mehrere Aufnahmen pro benötigter Klasse durchführen. Im Umfang dieser Arbeit und durch die Einschränkungen der Pandemie wurde dieser Ansatz nicht gewählt.

Ein weiterer möglicher Weg ist ein Text-zu-Sprache-Modell, also *Text-to-Speech* (TTS), zu verwenden. Somit ist es möglich, ein komplettes Datenset zu erstellen, welches genau die Wörter beinhaltet, die gebraucht werden. Dieser Ansatz wurde gewählt, da dieser am besten umzusetzen ist.

Vorbereitung

Für die Umsetzung wurde das TTS-Modell von Google gewählt [27]. Auf der Google -Cloud-Plattform kann man unter APIs und Services die sogenannten *credentials* beantragen. Dabei handelt es sich um eine JSON-Datei in dem Authentifizierungstoken und anderen wichtigen Informationen enthalten sind. Mit diesen *credentials* und der Google-Cloud-Pythonbibliothek ist es möglich, das Sprachmodell zu verwenden. Zum Einlesen und Schreiben von Audiodateien wird die Python-Bibliothek SoundFile verwendet. Für die Datenaugmentierung wird nlpaug verwendet.

Erstellung der Daten

Mit der Text-to-Speech-Bibliothek kann ein *Client* erstellt werden. Mit diesem können alle verfügbaren Stimmen erhalten und die deutschen Stimmen extrahiert werden. Mit diesen erhält man auch die Stimmenparameter. Als Nächstes muss noch das erwünschte Audioformat gewählt werden. Mit Klassennamen, Stimmenparameter und Audioformat sind die Vorbereitungen vollständig und die eigentliche Text-zu-Sprach-Synthese kann gestartet werden. Das Ergebnis der Text-zu-Sprach-Synthese kann als Datei im WAV-Fromat gespeichert werden. Dieser Vorgang wird für jede Stimme in der gewünschten Sprache und für jeden Eingabewert durchgeführt. Für die weitere Benutzung des Datensets ist es wichtig die Klassennamen, den Dateinamen und eine Klassen-ID abzuspeichern. Diese Daten wurden gleich beim Erstellen der Audiodateien in einer CSV-Datei hinterlegt. Auf die Verwendung dieser CSV-Datei wird beim laden genauer eingegangen.

Datenaugmentierung

Um aus den erstellten Sprachbeispielen mehr Daten zu gewinnen, wird eine Datenaugmentierung durchgeführt. Dies ist ein Vorgang, bei dem die originalen Daten zum Beispiel durch Filter verändert werden oder Informationen hinzugefügt und entfernt werden. Dies ermöglicht dem Modell robuster zu werden und dadurch seine Genauigkeit zu verbessern. Bei der Augmentation von Audiodaten gibt es mehrere Möglichkeiten. Die Lautstärkenaugmentierung erlaubt es die Lautstärke zu verändern. Dabei wird die Amplitude erhöht oder verringert um die Lautstärke zu erhöhen oder zu senken. Mit der Augmentierung für Rauschen kann dem Signal zufälliges Rauschen hinzugefügt werden. Weitere Möglichkeiten bieten die Augmentierungen zum Verändern der Tonhöhe, der Tongeschwindigkeit sowie der zeitlichen Verschiebung.

Für jede Datei im Datensatz wird eine beliebige Anzahl an augmentierten Daten erstellt. Für jede Audiodatei werden zufällige Augmentierungen aus den obigen Erwähnten durchgeführt. Die Werte für die einzelnen Augmentierungen werden ebenfalls zufällig gewählt. Das augmentierte Signal kann anschließend wieder als WAV-Datei gespeichert werden. Beim erstellen dieser neuen Daten werden deren Dateinamen, Klassennamen und Klassen-ID ebenfalls in die CSV-Datei eingetragen. Somit wurden alle Daten erstellt und der nächste Schritt, die Erstellung eines CNN-Modells, kann beginnen.

5.2.2 Modellerstellung

Zur Erstellung eines Modells ist erst einmal das Laden des Datensatzes wichtig. Anschließend wird die Struktur des Modells, dessen Training und die Tests beschrieben.

Vorbereitung

Das Modell wird mit PyTorch erstellt. Bevorzugt wird das Modell über die GPU mittels CUDA trainiert [22]. Für das Laden und Vorbereiten des Datensets wird die Python-Bibliothek pandas verwendet [12]. Weiters wird auch Torchaudio für das Laden, Transformieren und das Erstellen des Spektrogramms benötigt [13].

Laden der Daten als Datensatz

Der nächste Schritt ist es, den Datensatz zu laden und mit PyTorch kompatibel zu machen. Dazu wird eine neue Klasse erstellt, dessen Überklasse Dataset von PyTroch ist. Zwei Methoden müssen erstellt werden um den eigenen Datensatz mit PyTorch verwenden zu können. Die erste Funktion ist `__len__()`, diese gibt die Anzahl der Daten zurück. Die zweite Funktion ist `__getitem__()`. Dabei wird das jeweilige Audiosignal geladen und vorverarbeitet. Die Daten werden dabei wenn nötig, zurechtgeschnitten beziehungsweise verlängert und die Abtastfrequenz angepasst. Um auf die richtigen Daten zuzugreifen zu können, werden die Namen und deren Klassen benötigt. Diese befinden sich in der CSV-Datei welche beim Erstellen des Datensatzes erstellt wurde. Diese Vorbereitung normiert die Dimensionen der Daten und erlaubt so den Modellaufbau leichter zu gestalten.

Struktur des Modells

Bevor die Daten für das Training verwendet werden können, muss das Modell an sich definiert werden. Die verwendete Modellstruktur wurde von RoyT erstellt [45]. Der Aufbau des Modells, die einzelnen Schichten, deren Ausgangsdimensionen sowie die Anzahl der trainierbaren Parameter sind im Programmausschnitt 5.1 zu sehen.

Training des Modells

Nun, da der Datensatz vollständig und bereit für die Verwendung ist und das Modell definiert wurde, kann das Training beginnen. Der Datensatz wird im Verhältnis 8:2 in Trainings- und Testdaten aufgeteilt. Beim Instanziieren des Netzwerkes ist es wichtig, dass ausgewählt wird, dass das Modell durch CUDA auf der GPU läuft. Dies reduziert die Trainingsdauer enorm. Für die Hyperparameter wurden Batch-Größe 128 und Epoch-Anzahl 10 gewählt. Diese Zahlen wurden zufällig gewählt und können beim erneuten Trainieren angepasst werden um ein besseres Modell zu erhalten.

Test des Modells

Das Training dauerte circa eine halbe Stunde. Mit den Testdaten kann nun die Genauigkeit des Modells getestet werden. Hierbei werden die Testdaten eingespeist, das Modell klassifiziert diese und das Ergebnis wird dann mit der tatsächlichen Klasse abgeglichen.

Programm 5.1: Aufbau des Sprachklassifizierungsmodells [45]

```
1  ----------------------------------------------------------------
2          Layer (type)               Output Shape         Param #
3  ================================================================
4              Conv2d-1           [-1, 32, 62, 45]             832
5         BatchNorm2d-2           [-1, 32, 62, 45]              64
6                ReLU-3           [-1, 32, 62, 45]               0
7              Conv2d-4           [-1, 32, 60, 43]          25,632
8         BatchNorm2d-5           [-1, 32, 60, 43]              64
9                ReLU-6           [-1, 32, 60, 43]               0
10           MaxPool2d-7           [-1, 32, 30, 21]               0
11             Conv2d-8           [-1, 64, 28, 19]          51,264
12        BatchNorm2d-9           [-1, 64, 28, 19]             128
13              ReLU-10           [-1, 64, 28, 19]               0
14            Conv2d-11           [-1, 64, 28, 19]          36,928
15       BatchNorm2d-12           [-1, 64, 28, 19]             128
16              ReLU-13           [-1, 64, 28, 19]               0
17          MaxPool2d-14            [-1, 64, 14, 9]               0
18            Conv2d-15            [-1, 64, 14, 9]          36,928
19       BatchNorm2d-16            [-1, 64, 14, 9]             128
20              ReLU-17            [-1, 64, 14, 9]               0
21            Conv2d-18            [-1, 64, 14, 9]          36,928
22       BatchNorm2d-19            [-1, 64, 14, 9]             128
23              ReLU-20            [-1, 64, 14, 9]               0
24          MaxPool2d-21             [-1, 64, 7, 4]               0
25            Conv2d-22             [-1, 64, 7, 4]          36,928
26       BatchNorm2d-23             [-1, 64, 7, 4]             128
27              ReLU-24             [-1, 64, 7, 4]               0
28            Conv2d-25             [-1, 64, 7, 4]          36,928
29       BatchNorm2d-26             [-1, 64, 7, 4]             128
30              ReLU-27             [-1, 64, 7, 4]               0
31          MaxPool2d-28             [-1, 64, 3, 2]               0
32            Flatten-29                  [-1, 384]               0
33            Dropout-30                  [-1, 384]               0
34             Linear-31                   [-1, 15]           5,775
35            Softmax-32                   [-1, 15]               0
36 ================================================================
37 Total params: 269,039
38 Trainable params: 269,039
39 Non-trainable params: 0
40 ----------------------------------------------------------------
41 Input size (MB): 0.01
42 Forward/backward pass size (MB): 6.18
43 Params size (MB): 1.03
44 Estimated Total Size (MB): 7.22
45 ----------------------------------------------------------------
```

Das Ergebnis des Tests war, dass 15% der Klassifizierungen richtig waren. Diese Zahl ist sehr gering und nicht akzeptabel. Weitere Tests mit Anpassungen der Modellstruktur und der Trainingsparametern wurden durchgeführt. Die beste erzielte Genauigkeit lag bei rund 19%.

5.2.3 Fazit

Beim Erstellen eines eigenen Spracherkennungsmodells sind sehr viele Schwierigkeiten aufgetreten. Das entstandene Modell ist nicht ausreichend zuverlässig, um verwendet werden zu können. Es ist unklar, welche Probleme genau aufgetreten sind, aber es wird versucht, mögliche Probleme zu erörtern.

Die verwendete Modelstruktur ist ausgelegt für die Audioklassifizierung, aber es gibt immer ein besseres Modell. Es wurde auch ein Test durchgeführt, in dem das Modell mit den Trainingsdaten getestet worden ist. Auch dieser Test lieferte nur unzureichende Ergebnisse. Das Modell müsste mit den Trainingsdaten eigentlich sehr genaue Klassifizierungen treffen. Deshalb wird vermutet, es könnte sich um *underfitting* handeln und das bedeutet, dass das Modell länger trainiert werden müsste. Weiters werden Probleme im Datenset und dessen Erstellung vermutet. Mit TTS konnten nicht genug unterschiedliche Daten hergestellt werden. Ein zu kleiner Datensatz führt unweigerlich zu einem schlechten Ergebnis. Die Idee, den Datensatz mittels Augmentation künstlich zu vergrößern, hat nicht funktioniert. Das Problem dabei ist, dass die Augmentation nur die vorhandenen Daten verändert und keine neuen erzeugt. Dies kann zwar die Robustheit steigern, aber ersetzt keine echten Daten. Eine Möglichkeit zur Verbesserung wäre eventuell die weitere Einschränkung der Aufgabenstellung gewesen. Eine Beschränkung, nur die zehn Ziffern für das Sprachmodell zu verwenden, hätte es ermöglicht, vorhandene Datensets zu verwenden.

Sich genauer mit dieser Thematik auseinanderzusetzen würde einen enormen Zeitaufwand bedeuten, welcher den Rahmen für diese Arbeit übersteigt.

5.3 Ergebnis

Bei der Implementierung der Gestenerkennung sind keine Probleme aufgetreten. Die Spracherkennung war jedoch nicht erfolgreich. Deshalb wird auch nicht die Kombination der zwei Teilbereiche implementiert und der Prototyp wird nicht auf diese Weise fertiggestellt. Ein funktionierender Prototyp ist mit dieser Implementierung nicht möglich. Im Laufe dieser Implementierung wurde jedoch ein weiterer Ansatz zum Erstellen eines Sprachklassifizierungsmodells gefunden. Dieser Ansatz wird im nächsten Kapitel behandelt.

Kapitel 6

Implementierung mit Teachable Machine

Im Laufe der ersten Implementierung sind viele Probleme aufgetreten und das Ergebnis war nicht geeignet für einen Prototypen. Währenddessen wurde jedoch ein neuer Ansatz entdeckt, namens Teachable Machine (TM), welcher es erlaubt, einfach eigene Modelle zu erstellen. Dieser Ansatz ist jedoch derzeit noch nicht für Python verfügbar. Deshalb wurde entschieden, das Projekt in JavaScript zu implementieren. Es wurde auch entschieden, dass der Prototyp nicht ohne Internetverbindung funktionieren muss, es jedoch die Möglichkeit geben soll, ihn später auch ohne Internet und auf eingebetteten Systemen zu verwenden. Wie dies funktionieren kann, wird später beschrieben.

Als Erstes wird die Sprachsteuerung behandelt und anschließend werden noch Anpassungen an der Gestensteuerung gemacht. Zum Schluss wird das Gesamtsystem zusammengesetzt und dessen Erstellung beschrieben.

6.1 Sprachsteuerung

Der neue Ansatz zur Implementierung der Sprachsteuerung verwendet Teachable Machine. Bevor dieser Service verwendet wird, wird noch kurz erklärt was TM genau ist und wie es funktioniert. Anschließend folgt die Beschreibung zum Erstellen des eigenen Modells und zum Schluss noch ein Test.

6.1.1 Teachable Machine

Teachable Machine ist ein Service von Google direkt im Browser, welcher es erlaubt selber Modelle anzutrainieren. Der Vorgang ist einfach und intuitiv gestaltet und somit für jeden zugänglich. Wie in Abbildung 6.1 zu sehen ist werden derzeit Bildklassifizierungs-, Tonklassifizierungs- und Posenerkennungsmodelle zum Trainieren angeboten [26].

Um ein Modell mit TM zu trainieren, braucht man nur sehr kleine Datenmengen. Beim Erkennen von Bildern genügen schon Mengen von eins bis zehn Bildern, beim Erkennen von Tönen reichen acht Audiobeispiele. Um ein robusteres Modell zu erstellen, benötigt man jedoch mehr Daten, aber bei Weitem nicht so große Mengen, wie benötigt wird, um Modelle von Grund auf zu trainieren. Der Grund dafür ist eine Methode namens *Transfer Learning*, also Transfer-Lernen. Dabei wird das erlernte Wis-

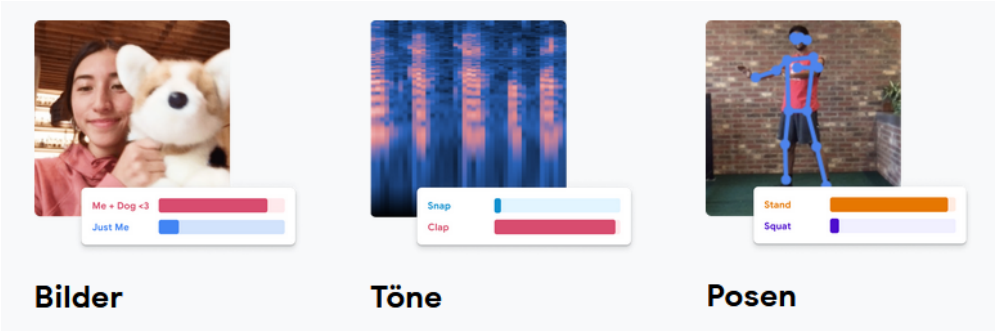


Abbildung 6.1: Übersicht der derzeit angebotenen Modelle von Teachable Machine [26]

sen beim Lösen eines Problems gespeichert und bei einem anderen ähnlichen Problem angewendet. Es gibt also schon vortrainierte Modelle, welche mit den vom Benutzer bereitgestellten Daten und deren Klassen noch fertig trainiert und angepasst werden. Dies erlaubt ein sehr schnelles Trainieren und braucht nur kleine Datensätze [26]. Teachable-Machine-Modelle sind TensorFlow.js-Modelle und sind daher kompatibel mit JavaScript und dessen Bibliotheken wie ml5.js, p5.js, und Node.js. Wie schon ML5 Handpose funktioniert deshalb auch TM ohne Internetverbindung, auf eingebetteten Systemen und generell überall wo JavaScript läuft.

6.1.2 Modellerstellung

Jetzt, da erklärt wurde, wie Teachable Machine funktioniert und was es ist, kann ein Modell erstellt werden.

Definition der Klassen

Der erste Schritt beim Erstellen eines neuen Tonklassifizierungsmodells mittels Teachable Machine ist die Definierung der Klassen. Zur Steuerung des Taschenrechners muss zwischen den zehn Ziffern, den vier Grundoperatoren der Addition, Subtraktion, Multiplikation und Division, sowie einem Ist-gleich-Zeiche unterschieden werden. Da die Erstellung mittels TM so einfach gestaltet ist, werden auch noch 'start' und 'stop' als Befehle hinzugefügt, welche für die generelle Steuerung des Taschenrechners verwendet werden können. Eine extra Klasse für Hintergrundgeräusche wird ebenfalls noch hinzugefügt. Somit ergeben sich achtzehn verschiedene Klassen. Nach dem Definieren der Klassen können die Audiobeispiele aufgenommen werden.

Sammeln der Daten

Teachable Machine erlaubt es, direkt ein Mikrofon auszuwählen und Tondaten zu sammeln. In Abbildung 6.2 kann man links oben den Klassennamen erkennen. Nach Auswahl des Mikrofons können die Tonproben aufgenommen werden. Sie können einzeln im Sekundentakt aufgenommen werden oder auch über längere Zeiten. Im Beispiel von Abbildung 6.2 wurden acht Sekunden aufgenommen und daraus anschließend die Audio-

Beispiele extrahiert. Für Hintergrundgeräusche ist ein Minimum an zwanzig Beispielen nötig, für jede Weitere sind mindestens acht Beispiele nötig. Als Daten für die Hintergrundgeräusche wurden stumme Aufnahmen, Aufnahmen mit Rauschen und Aufnahmen mit Stimmen verwendet. Bei den Stimmen wurde darauf geachtet, dass keine Wörter vorkommen die auch in den achtzehn Klassen enthalten sind. Sobald alle Beispiele aufgenommen wurden ist das Modell bereit für das Training.

Training

In Abbildung 6.3 wird die Trainingsübersicht dargestellt. Den einzigen Hyperparamter den man ändern kann ist die Anzahl der Epochen. Standardmäßig sind 50 Epochen eingestellt. Diese Zahl zu erhöhen verlängert die Trainingszeit, erhöht jedoch die Genauigkeit des Modells. Es ist auch möglich sich weitere Details anzeigen zu lassen. Zum einen wird in der Mitte die Genauigkeit pro Epoche angezeigt. Man erkennt wie die Genauigkeit im Laufe des Trainings steigt. Diese Genauigkeit ist der prozentuelle Anteil, der beim Training richtig klassifiziert worden ist. Der zweite Graph hanz rechts zeigt den Verlust pro Epoche. Dieser zeigt den Verlauf der *loss function*. Mithilfe dieser Details und der folgenden Modelldemo können noch Änderungen und Optimierungen gemacht werden.

Vorschau

Sobald das Modell fertig trainiert wurde, ist eine Vorschau verfügbar. Diese Vorschau dient zum Testen des Modells und erlaubt es, Anpassungen durchzuführen. Die Vorschau wird in Abbildung 6.4 gezeigt. Links ist das aktuelle Eingangssignal in Form eines Spektrogramms zu sehen. Es wird gerade die Zahl Eins gesprochen. Im rechten Teil kann man sehen, dass diese auch richtig klassifiziert worden ist. Hier ist auch ein guter

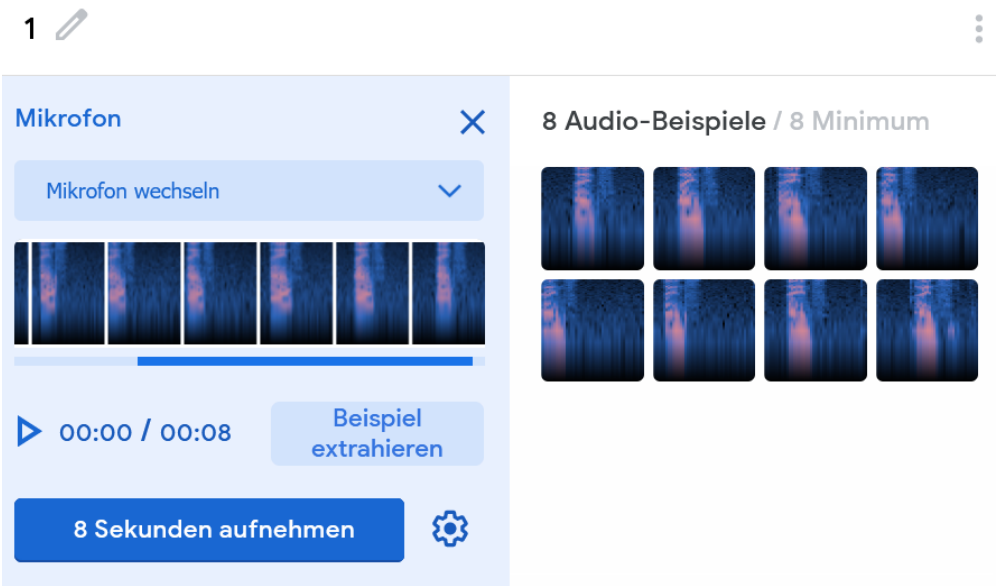


Abbildung 6.2: Aufnahme der Zahl Eins [26]

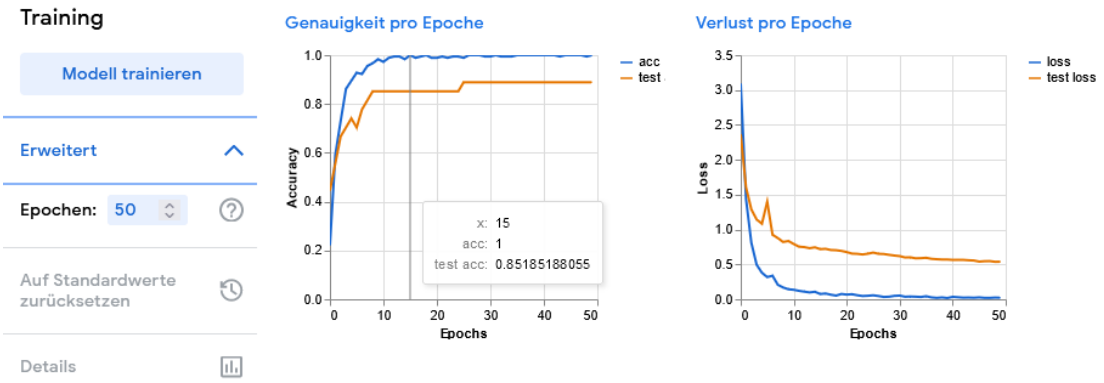


Abbildung 6.3: Einstellungsmöglichkeiten und Informationen über das Training des Modells [26]

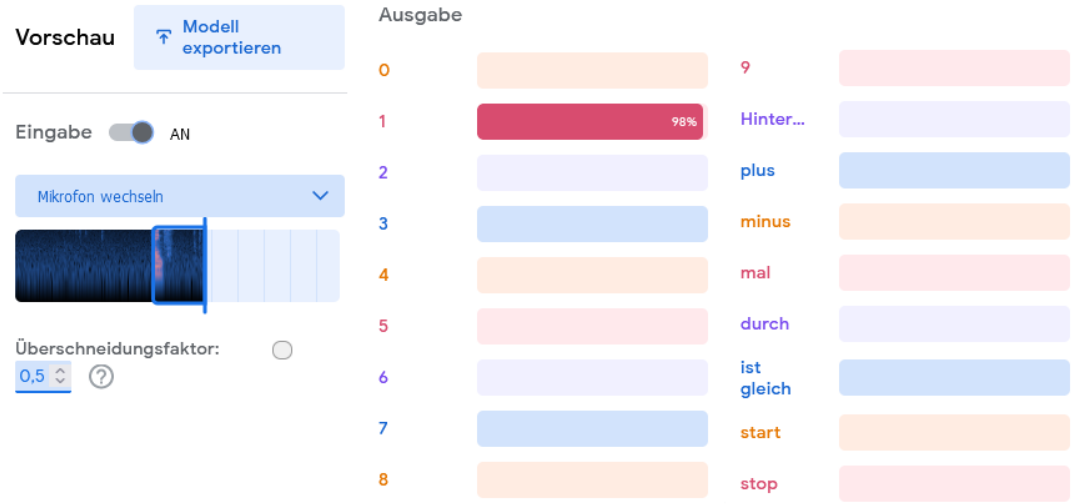


Abbildung 6.4: Vorschau des fertig trainierten Modells [26]

Überblick über die achtzehn Klassen zu sehen.

6.1.3 Anwendung des Modells

Um das Modell in der eigenen Anwendung verwenden zu können, muss es noch exportiert werden.

Exportieren

Das Modell kann exportiert werden, sobald es fertig trainiert und getestet wurde. Der Knopf dafür ist in Abbildung 6.4 zu sehen. In Abbildung 6.5 ist die Übersicht zu den verschiedenen Anwendungsmöglichkeiten zu sehen. Derzeit werden bei der Tonklassifizierung Tensorflow.js und Tensorflow Lite unterstützt. Das Tensorflow.js Modell kann

Abbildung 6.5: Exportmöglichkeiten des Modells [26]

entweder hoch- oder heruntergeladen werden. Wenn es hochgeladen wird, wird es gratis von Teachable Machine bereitgestellt und man erhält einen einzigartigen Link. Hinter diesem Link befindet sich eine eigene Webseite, die eine Live-Demo des eigenen Modells zeigt. Derselbe Link wird auch bei der Programmierung verwendet um das Modell zu laden. Weiter unten in Abbildung 6.5 stehen Code-Ausschnitte für JavaScript und p5.js zur Verfügung. Diese zeigen die einfache Verwendung des Modells.

Verwendung

Um die Übersichtlichkeit beizubehalten wurde das Modell zur Sprachsteuerung in eine eigene Datei ausgelagert. Es wurden zwei Funktionen erstellt. Die Erste ist die Initiierungsfunktion `initVoiceControl()`. Diese setzt die Optionen des Modells und ladet es mithilfe des oben erstellten Links. Die Option welche eingestellt wird ist der `probabilityThreshold`, welcher angibt wie sicher das Modell sein muss, um es einen neuen Ausgabewert zu setzen. Beim Testen wurde 0.5 verwendet und lieferte gute Ergebnisse und wird deshalb auch hier beibehalten. Die zweite Funktion `classifySound()` startet die Klassifizierung und aktualisiert bei neuen Ergebnissen die Ausgangsvariable `label`. Diese Variable kann schließlich als Ergebnis des Sprachklassifizierungsmodells gelesen werden. Wie dieser Teil im Endprodukt verwendet wird, wird weiter unten, nach der Anpassung der Gestensteuerung, gezeigt.

6.1.4 Test und Fazit

Die Tests haben ergeben, dass das System sehr gut funktioniert. Jedoch funktioniert es nur verlässlich mit der Stimme oder den Stimmen, mit denen es trainiert wurde. Durch die kleine Anzahl der Daten entsteht eine Voreingenommenheit, in Englisch *bias*, vom Modell. Auch das Mikrofon wirkt sich auf die Qualität aus. Die Daten für das

Training wurden mithilfe eines Headsets aufgenommen. Bei den Tests funktionierte die Klassifizierung über das Headset sehr gut. Die Tests, die mit einem im Laptop integrierten Mikrofon durchgeführt wurden, fielen sehr schlecht aus. Die Klassifizierung erzielte zu 90% die falschen Ergebnisse, die meisten Wörter wurden als Hintergrundgeräusche erkannt. Es wird vermutet, dass dies daran liegt, dass beim Aufnehmen der Hintergrundgeräusche auch gesprochen wurde. Die Idee dabei war es, das Modell robuster zu gestalten und auch damit es auch in Räumen mit mehreren Personen funktioniert. Es wurde bewusst darauf geachtet, dass keine Wörter gesprochen wurden, welche in den Klassennamen enthalten sind. Für die Verwendung mit anderen Mikrofonen müsste das Modell um weitere Audiodaten erweitert werden.

Dieser Ansatz funktioniert und kann für den finalen Prototypen verwendet werden. Zuvor werden aber noch Anpassungen bei der Gestensteuerung gemacht.

6.2 Gestensteuerung

Im vorigen Kapitel wurde bereits ein funktionierender Ansatz zur Gestensteuerung vorgestellt. Mit den neuen Möglichkeiten, die Teachable Machine bietet, kommt die Frage auf, ob es nicht sinnvoller wäre, auch die Gestensteuerung mit TM zu gestalten. Auf diese Frage wird im nächsten Abschnitt eingegangen. Wie im Kapitel 3 schon gezeigt wurde, stellt ML5 Handpose eine alternative zu MediaPipe Hands dar. Da beide dieser Frameworks JavaScript unterstützen, wird getestet ob es einen Unterschied beziehungsweise Vor- und Nachteile gibt. Anhand dieser Informationen wird am Ende noch die Gestensteuerung angepasst und vervollständigt.

6.2.1 Teachable Machine für Gestensteuerung

Nachdem Teachable Machine so gut für das Trainieren der Sprachsteuerung funktioniert hat, tritt die Frage auf, ob es nicht auch eine bessere Lösung für die Gestensteuerung wäre.

Mit Hilfe von TM wäre es möglich, die Bilder der Handgesten aufzunehmen, ein Modell zu trainieren und somit den richtigen Klassen zuzuordnen. Der Vorteil ist ganz klar durch die einfache Bedienbarkeit und das schnelle Erstellen eines Modells gegeben. Der Nachteil ist jedoch, dass die Hand selbst nicht erkannt und lokalisiert wird. Dadurch würde es am besten in einem vorgegebenen Bereich mit einfarbigem Hintergrund funktionieren. Da dieses System auf die Bilderkennung ausgelegt und nicht auf Hände spezialisiert ist, wie MediaPipe und ML5, werden große Mengen an Daten und lange Trainingszeiten notwendig sein, um ein robustes System zu erstellen welches auch ohne vorbestimmten Bereich funktioniert. Die Spezialisierung von MediaPipe Hands und ML5 Handpose ist zu bevorzugen. Die 21 *Landmarks* und deren Koordinaten sind ebenfalls ein Vorteil der nicht zu vergessen ist.

6.2.2 ML5 Handpose

Nun, da begründet wurde, dass Teachable Machine nicht so gut für die Gestensteuerung geeignet ist, bleibt noch der Vergleich zwischen ML5 Handpose und MediaPipe Hands. Dazu wurde ein Beispiel implementiert, welches es erlaubt, per Webcam die Hand zu erkennen. In diesem Punkt funktionierte ML5 genau so gut wie MediaPipe. Das Problem stellte sich jedoch erst bei der Kombination mit dem Sprachmodell heraus. Sobald die Sprachsteuerung und die Gestensteuerung mit ML5 zusammen verwendet wurde, funktionierte es nicht mehr und das Programm frierte ein. JavaScript läuft auf einem einzelnen Kern. Diese Begrenzung in der Leistung ist hier ein Problem. Somit wurde gezeigt, dass ML5 Handpose nicht für den Prototypen in Frage kommt.

6.2.3 MediaPipe Hands mit JavaScript

In der vorherigen Implementierung im Abschnitt 5.1 wurden bereits die Funktion, die Konfiguration und die Ausgabe von MediaPipe Hands erklärt. Es müssen also nur Anpassungen für die Verwendung mit JavaScript gemacht werden.

Um auch hier die Übersichtlichkeit beizubehalten wurde die Gestensteuerung ebenfalls in eine eigene Datei ausgelagert. Es wurde eine Initiierungsfunktion mit dem Namen `initGestureControl()` implementiert. Darin wird zuerst die Videoaufnahme der Webcam gestartet. Danach werden die Optionen gesetzt. Die Optionen wurden ebenfalls im Abschnitt 5.1 beschrieben. Der letzte Teil der Initiierungsfunktion ist das Setzen einer Rückruffunktion. Diese aktualisiert beim Erkennen einer Hand die Koordinaten der *Landmarks*. Wie die Koordinaten der *Landmarks* dann im finalen Prototyp verwendet werden, ist weiter unten bei der Kombination der zwei Steuerungsarten beschrieben.

6.2.4 Auswertung der Erkennung

Im Abschnitt 5.1.2 wurde bereits die Ausgabe des MediaPipe-Modells erklärt. Um dieses Ergebnis nun richtig auszuwerten, wurden die Finger binär kodiert. Das heißt anhand der dreidimensionalen Koordinaten der *Landmarks* wird unterschieden ob ein Finger ausgestreckt oder eingerollt ist. Daraus bildet sich eine Binärkodierung mit fünf Stellen. Diese Art der Kodierung erlaubt es auf einfache Weise allte nötigen Gesten zu erkennen. Alternativ könnten komplexere Gesten verwendet werden. Zum Beispiel könnte das Gleichheitszeichen mit zwei horizontalen Fingern gezeigt werden. Für die Entwicklung des Prototypen wurde jedoch die Binärkodierung gewählt, da somit die Komplexität reduziert wird.

In der Tabelle 6.1 ist eine Übersicht über die Binärkodierung dargestellt. Sie zeigt wie jeder Finger stehen muss damit die jeweilige Klasse ausgewählt wird. So ist das Pluszeichen zum Beispiel durch die Kombintation 00100, also nur der Mittelfinger, auszuwählen. Diese Kodierung erlaubt es, sehr einfach die gwünschte Geste zu erkennen und eine Klassenbezeichnung zu setzten. Diese Klassenbezeichnung wird ebenso wie beim Sprachmodell in eine Ausgangsvariable `label` geschrieben. Aus dieser Variable kann später das Ergebnis für die Weiterverarbeitung gelesen werden.

Daumen	Zeigefinger	Mittelfinger	Ringfinger	Kleiner Finger	Klasse
0	0	0	0	0	0
1	0	0	0	0	1
1	1	0	0	0	2
1	1	1	0	0	3
1	1	1	1	0	4
1	1	1	1	1	5
0	1	0	0	0	6
0	1	1	0	0	7
0	1	1	1	0	8
0	1	1	1	1	9
0	0	1	0	0	+
0	0	1	1	0	-
0	0	1	0	1	/
0	0	1	1	1	*
1	1	0	1	1	=

Tabelle 6.1: Zeigt die Binärkodierung der Gesten für jede Klasse

6.2.5 Test und Fazit

Der Test hat gezeigt, das MediaPipe Hands effizient genug funktioniert, um Sprach- und Gestensteuerung gleichzeitig verwenden zu können. Ebenfalls ist die Genauigkeit von MediaPipe sehr gut, es werden alle Gesten ohne Probleme erkannt. Somit sind die Grundelemente des Taschenrechners fertiggestellt und die Kombination der beiden Teile kann beginnen.

6.3 Kombination von Sprach- und Gestensteuerung

6.3.1 Bedienoberfläche

Für die Kombination von Sprach- und Gestensteuerung, sowie die generelle Benutzung des Taschenrechners ist es sinnvoll eine Bedienoberfläche zu gestalten. Ein sehr zugänglicher Weg eine solche zu erstellen, ist p5.js. Dabei handelt es sich um eine Bibliothek für JavaScript, welche es erlaubt, einfache Darstellungen und Visualisierungen im Browser zu erstellen. Dabei stellt diese Bibliothek eine Vielzahl an Funktionen zur Verfügung, welche das Zeichnen von verschiedenen geometrischen Elementen erlaubt [42]. Dies erlaubt es, schnell eine Bedienoberfläche für den Taschenrechner zu schaffen.

In Abbildung 6.6 ist die Bedienoberflache des Taschenrechners zu sehen. Der obere Teil ist für die Wiedergabe des Kamerasignals und die darin visualisierte Gestenerkennung. Dies erlaubt eine einfachere Verwendung, da immer zu sehen ist ob die Hand im Bild ist oder nicht. Direkt darunter ist ein Balken, der das Abtasten der beiden Steuerungen visualisiert. Darunter befinden sich drei Variablen. Die Erste beschreibt das Ergebnis der Sprachsteuerung, darunter dass der Gestensteuerung und zum Schluss

Abbildung 6.6: Bedienoberfläche des Taschenrechners.

die aktuelle Rechnung. Die aktuelle Rechnung zeigt an welche Werte übernommen wurden und auch das Ergebnis dieser Rechnung. Ganz links unten ist eine Schaltfläche zum starten und stoppen der Abtastung. Dieser Überblick soll helfen die Funktion besser zu verstehen.

6.3.2 Funktion

Beim Starten der Anwendung wird bei Sprache und Gestik 'laden....' angezeigt. Dies signalisiert, dass die Modelle geladen und initialisiert werden. Sobald das Laden abgeschlossen ist werden sofort die erkannten Sprach- und Gestikbefehle erkannt und dort ausgegeben. Zu diesem Punkt wird noch nichts für die eigentliche Rechnung übernommen. Erst sobald der Timer zum Abtasten gestartet wird, ist der Rechner aktiv. Im Abtastbalken erscheint dann ein roter Punkt welcher sich von links nach rechts bewegt. Immer wenn er in der Mitte über dem hellroten Punkt ist, wird abgetastet. Bie der Abtastung wird überprüft welche Werte von den zwei Steuerungen anliegen. Für den Fall das bei beiden Kanälen valide Ergebnisse sind muss eine Priorität gewählt werden. Da die Sprachsteuerung nicht so verlässlich ist wie die Gestensteuerung, wurde der Sprache die höhrere Priorität gegeben. Bei der Abtastung wird also als erstes überprüft, ob ein valider Sprachbefehl vorliegt. Wenn dies der Fall ist, wird die jeweilige Ziffer oder der Operator übernommen und der Rechnung hinzugefügt. Falls kein Sprachbe-

fehl anliegt, wird überprüft ob eine Geste gezeigt wird. Falls dem so ist, wird auch diese übernommen. So kann eine vollständige Rechnung eingegeben werden. Falls das Gleichheitszeichen erkannt wird, stellt dies das Ende einer Rechnung dar. Somit wird die eingegebene Rechnung überprüft und ausgerechnet und das Ergebnis wird ebenfalls angezeigt. Sobald die Abtastung neu gestartet wird, verschwindet die alte Rechnung und eine neue kann gestartet werden.

6.4 Zusammenfassung

In diesem Kapitel wurde ein Sprachklassifizierungsmodell mittels Teachable Machine erstellt. Die Gestensteuerung wurde aus dem vorherigen Kapitel angepasst um mit JavaScript zu funktionieren. Anschließend wurden diese zwei Eingabemöglichkeiten kombiniert und mit einer einfachen Bedienoberfläche visualisiert. Der Prototyp des Taschenrechners ist nun fertig und wird im nächsten Kapitel ausgiebig getestet.

Kapitel 7

Tests

In diesem Kapitel wird der Taschenrechner ausführlich getestet. Dabei werden die Tests wieder in zwei Teile aufgeteilt. Zuerst wird die Sprachsteuerung und deren Genauigkeit getestet. Anschließend wird die Gestensteuerung genau betrachtet. Zum Schluss gibt es noch eine finale Demonstration des Gesamtsystems.

7.1 Test der Sprachsteuerung

7.1.1 Headset

Die Audiobeispiele zum Trainieren des Spracherkennungsmodells wurden mit einem externen Headset aufgenommen. Deshalb wird der erste Test mit dem gleichen Headset durchgeführt und soll zeigen, wie genau und robust das Modell ist. In Tabelle 7.1 sind die Testdaten dargestellt. Links sind alle Klassen zu sehen die getestet wurden. Es wurden insgesammt fünf Tests durchgeführt, also fünf Mal wurde der Sprachbefehl gesprochen und dessen Ergebnisse wurden dokumentiert. Leere Einträge bedeuten, dass kein Sprachbefehl erkannt wurde. An den Ergebnissen ist klar zu erkennen das einige Klassen schlechter erkannt werden als andere. 0 und 9 sind nur in einem Test richtig erkannt worden. Bei 8 wurde zwar immer etwas erkannt aber in drei Fällen war es eine 3. 5 und die Subtraktion wurden nur dreimal richtig erkannt. Insgesammt wurden von den 75 Tests 58 Spracheingaben richtig erkannt. Dies entspricht einer Genauigkeit von 77,3%. 7 Tests, also 9,3%, wurden falsch und 10 Tests, 13,3%, wurden gar nicht erkannt.

7.1.2 Internes Mikrofon

Zum vergleich wurde noch ein Test mit dem internen Mikrofon des Laptops durchgeführt. Der Aufbau der Ergebnisse ist gleich wie vorher und in der Tabelle 7.2 dargestellt. Auf dem ersten Blick lässt sich bereits erkennen, dass viel weniger Sprachbefehle erkannt wurden. 30 Tests wurden richtig erkannt, dass entspricht genau 40%. 15 Aufnahmen, 20%, wurden falsch und 30 Tests, 40%, wurden wieder gar nicht erkannt.

Klasse	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**
0			0		
1	1	1	1	1	1
2	2	=	2	2	2
3	3	3	3	3	3
4	4	4	4	4	7
5	5	5	7	5	
6	6	6	6	6	6
7	7	7	7	7	7
8	*8*	*3*	*3*	8	*3*
9		*9*			
+	+	+	+	+	+
-	-		-	+	-
*	*	*	*	*	*
/	/	/	/	/	/
=	=	=	=	=	=

Tabelle 7.1: Test mit externem Headset

Klasse	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**
0		0			0
1				1	
2	3				3
3	3			2	
4		4		7	4
5		0	5	7	0
6	7		6	6	
7		7	7	7	7
8	*8*	*8*	*8*	3	*8*
9	*9*	*9*			
+		+	+	+	*7*
-	-	-	4	+	
*	*		3	*	
/	/	/	/		/
=		7		6	

Tabelle 7.2: Test mit internem Mikrofon des Laptops

7.1.3 Fazit

Das Headset, mit dem auch die Audioaufnahmen für das Training aufgenommen wurden, erreichten nur eine Genauigkeit von rund 77,3%. Dieses Ergebnis ist einerseits auf den gewählten Schwellwert zurückzuführen. Dieser gibt die Genauigkeit an die erreicht werden muss, damit das Ergebnis ausgegeben wird. Durch verändern dieses Schwellwärts

könnte die Genauigkeit des Modells etwas gesteigert werden. Es wird jedoch vermutet, dass das Hauptproblem beim nicht oder falschem Erkennen daran liegt, dass zu wenige Audiobeispiele für das Training verwendet wurden. Mehr Beispiel würden die Robustheit und Genauigkeit des Modells steigern. Der Test mit dem internen Mikrofon zeigte, dass die Aufnahmequelle und vor allem die Aufnahmequalität auch sehr wichtig ist. So wirkt sich die schlechtere Qualität des internen Mikrofon darauf aus, dass es mit den höher qualitativen Audiobeispielen nur schlecht verglichen werden kann. Der Genauigkeitsverlust von 37,7% bestärkt diese Aussage. Es wird vermutet, dass es für die Steigerung der Genauigkeit des Modells hilfreich wäre, mehrere Audioaufnahmen mit unterschiedlichen Mikrofonen durchzuführen.

7.2 Test der Gestensteuerung

7.2.1 Geeigneter Hintergrund

Der erste Test in Abbildung 7.1 zeigt das Gestenerkennungsmodell bei einem einfarbigen weißen Hitnergrund und einer guten Beleuchtung. In der Abbildung 7.1a wird getestet, ob Gesten erkannt werden obwohl keine Hände im Bild sind. Wie zu sehen ist funktioniert dies bei den genannten Testbedienungen sehr gut. In der Abbildung 7.1b wurde getestet ob das Modell die Gesten richtig erkannt, wenn die Hand sehr schnell bewegt wird. Die visualisierung der Geste ist etwas verzögert aber sie wird richtig erkannt. In Abbildunden 7.1c und 7.1d werden undefinierte Gesten gezeigt. Wie zu sehen ist werden die einzelnen Landmarks immer noch gut erkannt. Die rchtige Klassifizierung der Gesten ist nur von dem Vergleich der Koordinaten der Landmarks abhängig. Deshalb wird in der Abbildung 7.1c keine der definierten Gesten erkannt. Hingegen in Abbildung 7.1d wird eine gültige Geste anhand der Koordinaten erkannt. Solche Fehler von falschen Eingaben sind aber zu vernachlässigen, da nicht einmal ein Mensch sagen könnte welche Gestik jetzt gemeint ist.

7.2.2 Schlechte Lichtverhältnisse

In Abbildung 7.2 wurde versucht mit andern Lichtverhältnissen bewusst eine falsche Geste herbeizuführen. Dieser Test soll zeigen, dass die erkannten Gesten nicht immer an einer Hand erkannt werden. In der Abbildung 7.2a wird richtig keine Geste erkannt. In Abbildung 7.2b wurde nun bewusst versucht durch verändern der Lichtverhältnisse im Gesicht eine falsche Geste hervorzurufen. Solch ein Fehler kann gelegentlich aufteten, wenn keine Hand im Bild ist.

7.2.3 Direktes Licht

In Abbildung 7.3 folgt der dritte Test mit einem schlechten Lichtverhältnis. Diese Aufnahmen wurden direkt bei einem Fenster mit direkter Lichteinstahlung aufgenommen. In Abbildung 7.3a ist zu sehen wie das direkte Licht verhindert, dass der kleine Finger richtig erkannt wird. Dies führt dazu, dass die Geste falsch klassifiziert wird. In der Abbildung 7.3b wird noch gezeigt, dass es durchaus möglich ist die Gestenerkennung in

(a) Keine Geste wird gezeigt

(b) Schnelle Handbewegungen

(c) Undefinierte Geste richtig nicht erkannt

(d) Undefinierte Geste falsch erkannt

Abbildung 7.1: Tests bei einfarbigen weißen Hintergrund und guter Beleuchtung

so einem Lichtverhältnis zu verwenden. Aufgrund der auftretenden Fehler wird dieses Lichtverhältnis jedoch nicht empfohlen.

7.2.4 Keine externen Lichtquellen

In Abbildung 7.4 wurde ein Test im Dunkeln druchgeführt. Dabei wurden keine externen Lichtquellen verwendet, sondern nur das Licht des Laptop-Monitors. Wenig Licht ist sofort an der schlechteren Aufnahmequalität zu erkennen. In Abbildung 7.4a ist jedoch zu sehen, dass auch in diesen Verhältnissen die Gestensteuerung verwendet werden

(a) Keine Geste wird erkannt, gute Lichtverhältnisse

(b) Erkennen einer falschen Geste aufgrund der Beleuchtung

Abbildung 7.2: Lichttest im Gesicht zur falschen Erkennung von Gesten

(a) Falsche Geste da ein Finger nicht erkannt wird

(b) Richtige Erkennung der Geste

Abbildung 7.3: Schlechtes Lichtverhältnis mit direktem Licht

kann. Wichtig ist dabei nur, dass die Hand nicht zu viel bewegt wird. Beim Bewegen der Hand sorgen die schelchten Lichtverhältnisse für ein verschwimmen der Hand. Dies wird in Abbildung 7.4b gezeigt. Hierbei kann keine Geste erkannt werden.

(a) Richtige Erkennung bei still stehender Hand

(b) Keine Erkennung beim Bewegen der Hand

Abbildung 7.4: Keine externen Lichtquellen, nur das Licht des Monitors

7.2.5 Objekte im Hintergrund

Im letzten Test, dargestellt in Abbildung 7.5, wird gezeigt, dass Objekte und Pflanzen im Hintergrund sich nur wenig auf die Qualität auswirken. Es ist zu sehen, dass keine falsche Geste erkannt wird. Gelgentlich werdem jedoch Gestem im Hintergrund erkannt. Dies geschah jedoch sehr selten und konnte nicht aufgenommen werden.

Abbildung 7.5: Objekte und Pflanzen im Hintergrund

7.2.6 Fazit

Das Gestenerkennungsmodell MediaPipe Hands von Google ist sehr robut und genau. Es funktioniert sowohl in sehr dunklen, als auch in sehr hellen Umgebungen. Es wurde gezeigt, dass das Erkennen der Gesten jedoch nicht perfekt ist und manchmal falsche Ergebnisse entstehen.

7.3 Finale Demonstration

Zum Schluss wird noch eine finale Demonstration anhand einer Beispielrechnung in der Abbildung 7.6 gezeigt. Sobald das Laden der Modelle abgeschlossen ist wird die Abtastung gestartet. In Abbildung 7.6a wird die Zahl eins gesprochen, erkannt und in die aktuelle Rechnung übernommen. In Abbildung 7.6b wird mittels Gestikeingabe die Zahl 5 gezeigt und ebenfalls übernommen. Somit wurde die Zahl 15 mittels Sprach- und Gestensteuerung eingegeben. Anschließen wird in Abbildung 7.6c die Gestik für die Multiplikation gezeigt. In Abbildung 7.6d wird wieder die Sprachsteuerung verwendet und eine 7 eingegeben. Somit wurde die zweite Zahl 27 eingegeben. Für das Abschließen der Rechnung wird in Abbildung 7.6e das Zeichen für das Gleichheitszeichen gestikuliert. Dieses wird übernommen, beendet die Abtastung und das Ergebnis wird angezeigt. Somit wurde die Rechnung $15 * 27 = 405$ nur mithilfe von Sprach- und Gestensteuerung durchgeführt.

(a) Spracheingabe der Zahl 1

(b) Gestikeingabe der Zahl 2

(c) Gestikeingabe für die Multiplikation

(d) Spracheingabe der Zahl 7

(e) Gestikeingabe des Gleichheitszeichen und anzeige des Ergebnisses

Abbildung 7.6: Beispiel zur Benutzung des Taschenrechners

Kapitel 8

Schlussbemerkungen

8.1 Zusammenfassung und Ergebnis

Im Rahmen dieser Arbeit wurde ein Taschenrechner-Prototyp realisiert. Dieser wird mittels Sprach- und Gestiksteuerung gesteuert und beherrscht das Rechnen mit den Grundoperatoren der Addition, Subtraktion, Multiplikation und Division. Zur Lösung dieser Aufgabe wurden mehrere verschiedene Ansätze gegenübergestellt und deren Vor- und Nachteile abgewogen. Bevor die Implementierung behandlet wurde, wurde ein Überlick über das Gesamtsystem erstellt und dessen Eigenschaften definiert. Bei der Implementierung wurde gezeigt, welche Schwierigkeiten beim Erstellen eines eigenen Datensatzes und beim anlernen eines eigenen CNN-Modells auftreten können. Anschließend wurde mit Teachable Machine ein neuer Ansatz gewählt, welcher die Realisierung dieses Taschenrechners ermöglichte. Dieser Ansatz ist im Vergleich zur eigenen Implementierung in jedem Punkt vorzuziehen. Die Erstellung eines eigenen Modells und das Sammeln der Daten dauert hiermit nur wenige Minuten. Es wurde ebenfalls eine Bedienberfläche für die Benutzung des Taschenrechners erstellt. Da es sich jedoch nur um einen konzeptionellen Beweiß der Machbarkeit handelt wurde nur das Minimum an Funktionen implementiert. so beschränkt sich die Bedienoberfläche auf einfache Visualisierungen der Ein- und Ausgabemöglichkeiten. Zum Schluss wurde das System ausgiebig getestet und anhand einer Demonstration die Funktionalität des Taschenrechners gezeigt.

Die vorgegebenen Aufgaben wurden erfüllt. Der Taschenrechner kann sowohl mit Sprache als auch mit Gestik gesteuert werden. Es werden alle zehn Ziffern, die Grundrechenoperatoren und das Gelichheitszeichen richtig klassifiziert. Das Wechseln der Eingabeart funktioniert intuitiv und dynamisch. Der Prototyp funktioniert derzeit jedoch nicht ohne Internetverbindung und auch nicht auf eingebetteten Systemen.

8.2 Probleme und Erkenntnisse

Im Zuge der Arbeit sind einige Probleme aufgetreten. Hauptsächlich lagen diese beim Erstellen eines eigenen Spracherkennungsmodells. Die Komplexität eines solchen Modells ist aufgrund der vielen Einstellmöglichkeiten sehr hoch. Wenn es schon vortrainierte Varianten gibt, sollten diese wenn möglich verwendet werden. Bei der Erstellung des

Datensatzes hat sich gezeigt, dass die Technologie der Text-zu-Sprache-Synthese noch nicht ausreicht, um einen benutzerdefinierten Datensatz zu erstellen.

Beim Sprachmodell von Teachable Machine wurde in den Tests gezeigt, dass zu wenige Audiobeispiele von zu wenigen Audioquellen verwendet wurden. Daher erreichte das Modell nur eine Genauigkeit von cirka 77%. Diese Angabe bezieht sich jedoch auf das gesamte Modell. Einzelne Klassen wurden bereichts zu 100% richtig klassifiziert.

Bei der Gestensteuerung gab es keine nennenswerte Probleme. Das Modell von MediaPipe Hands funktioniert in fast jeder Umgebung und ist sehr genau.

Die Vorgehensweise dieser Arbeit könnte Rückblickend durch bessere Recherche verbessert werden. Wenn der Ansatz von Teachable Machine schon vorher entdeckt worden wäre, hätte dies viel Zeit gespart.

8.3 Ausblick

Diese Arbeit hat noch großes Potential und könnte als Projekt weitergeführt werden. Es könnte ein neur Versuch gestartet werden ein eigenes Modell zu entwerfen. Mit einer besseren Struktur, besserem Training und vor allem einem besseren Datensatz. Die einfachere möglichkeit das Sprachmodell des Taschenrechners robuster zu machen ist jedoch einfach mehr Audiobeispiele aufzunehmen und so die Genauigkeit des Teachable-Machine-Modell zu steigern. Es könnten neue Operatoren eingeführt werden um die Rechenmöglichkeiten zu erweitern. Dazu müsste die Gestiksteuerung auf zwei Hände erweitert werden. Weiters wäre es möglich eine schönere Bedienoberfläche mit zusätzlichen Funktionen zu erweitern. Mittels node.js ist es auch möglich das gesammte System offline und auf eingebetteten Systemen laufen zu lassen. Dies eröffnet wiederum viele neue Möglichkeiten.

Anhang A

Ergänzende Inhalte

Auflistung der ergänzenden Materialien zu dieser Arbeit, die zur digitalen Archivierung an der Hochschule eingereicht wurden (als ZIP-Datei).

A.1 PDF-Dateien

Pfad: /Bachelorarbeit

KI_Taschenrechner.pdf Finale Bachelorarbeit (Gesamtdokument)

A.2 Mediendaten

Pfad: /Bachelorarbeit/Bilder

*.png . . . . . Alle Bilder die in der Bachelorarbeit enthalten sind

A.3 Code

Pfad: /Bachelorarbeit/Code

*.js, *html, *.css, *.txt Finaler Code des Prototypen

A.4 Quellen

Pfad: /Bachelorarbeit/Quellen

*.html, *js, *.css . . . . Alle verwendeter Quellen dargestellt als Webseite

Quellenverzeichnis

Literatur

[1] Bruce G Buchanan. „A (Very) Brief History of Artificial Intelligence“ (), S. 8 (siehe S. 4).

[2] M. Huzaifah. „Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks“. *arXiv:1706.07156 [cs]* (21. Juni 2017). arXiv: 1706.07156. URL: http://arxiv.org/abs/1706.07156 (besucht am 22. 08. 2021) (siehe S. 16).

[3] Ray Kurzweil. „The Singularity Is near: When Humans Transcend Biology“. *Foreign Affairs* 85.3 (2006), S. 160. URL: https://www.jstor.org/stable/10.2307/20031996?origin=crossref (besucht am 26. 07. 2021) (siehe S. 5).

[4] Shane Legg und Marcus Hutter. „A Collection of Definitions of Intelligence“ (15. Juni 2007), S. 13 (siehe S. 4).

[5] Mehryar Mohri, Afshin Rostamizadeh und Ameet Talwalkar. *Foundations of machine learning*. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2018. 486 S. (siehe S. 9).

[6] Playstation. *EyeToy*. In: *Wikipedia*. Page Version ID: 1021937180. 7. Mai 2021. URL: https://en.wikipedia.org/w/index.php?title=EyeToy&oldid=1021937180 (besucht am 28. 07. 2021) (siehe S. 1).

[7] Simon Haykin. *Neural Networks A Comprehensive Foundation*. 2. Aufl. 2005 (siehe S. 6).

Medien

[8] Lex Fridman. *Pamela McCorduck: Machines Who Think and the Early Days of AI | Lex Fridman Podcast #34*. 23. Aug. 2019. URL: https://www.youtube.com/watch?v=i6rnzk8VU24 (besucht am 28. 07. 2021) (siehe S. 4).

[9] Tesla. *Tesla Autonomy Day*. URL: https://www.youtube.com/watch?v=Ucp0TTmvqOE (besucht am 14. 07. 2021) (siehe S. 1).

[10] Valerio Velardo - The Sound of AI. *Mel Spectrograms Explained Easily*. URL: https://www.youtube.com/watch?v=9GHCiiDLHQ4 (besucht am 22. 08. 2021) (siehe S. 16).

Software

[11] Gogul09. *Gesture Recognition*. original-date: 2017-04-04T09:20:34Z. 20. Aug. 2021. URL: https://github.com/Gogul09/gesture-recognition (besucht am 22. 08. 2021) (siehe S. 19).

[12] *pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/ (besucht am 25. 08. 2021) (siehe S. 27).

[13] *torchaudio — Torchaudio 0.9.0 documentation*. URL: https://pytorch.org/audio/stable/index.html (besucht am 25. 08. 2021) (siehe S. 27).

[14] Anthony Zhang (Uberi). *SpeechRecognition: Library for performing speech recognition, with support for several engines and APIs, online and offline*. Version 3.8.1. URL: https://github.com/Uberi/speech_recognition#readme (besucht am 23. 08. 2021) (siehe S. 19).

Online-Quellen

[15] Ifu Aniemeka. *A Friendly Introduction to Convolutional Neural Networks*. 22. Aug. 2017. URL: https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks (besucht am 29. 07. 2021) (siehe S. 10–12).

[16] Aphex34. *Typical cnn.png – Wikipedia*. 16. Dez. 2015. URL: https://commons.wikimedia.org/wiki/File:Typical_cnn.png (besucht am 17. 08. 2021) (siehe S. 10).

[17] UC Berkeley. *What Is Machine Learning? - I School Online*. UCB-UMT. 26. Juni 2020. URL: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (besucht am 27. 07. 2021) (siehe S. 8).

[18] Brownlee. *Machine Learning Mastery*. Machine Learning Mastery. URL: https://machinelearningmastery.com/start-here/ (besucht am 21. 08. 2021) (siehe S. 13).

[19] cuelogic. *Role of Python in Artificial Intelligence (AI) | Python Machine Learning*. Cuelogic Technologies Pvt. Ltd. 23. Juni 2016. URL: https://www.cuelogic.com/blog/role-of-python-in-artificial-intelligence (besucht am 01. 09. 2021) (siehe S. 24).

[20] Ketan Doshi. *Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better*. Medium. 21. Mai 2021. URL: https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505 (besucht am 22. 08. 2021) (siehe S. 15).

[21] *Duden | Spracherkennung | Rechtschreibung, Bedeutung, Definition, Herkunft*. URL: https://www.duden.de/rechtschreibung/Spracherkennung (besucht am 22. 08. 2021) (siehe S. 14).

[22] Facebook. *PyTorch*. URL: https://www.pytorch.org (besucht am 25. 08. 2021) (siehe S. 27).

[23] google. *MediaPipe*. opensource.google. URL: https://opensource.google/projects/ (besucht am 22. 08. 2021) (siehe S. 17, 18).

[24] google. *MediaPipe_Hands*. mediapipe. URL: https://google.github.io/mediapipe/solutions/hands.html (besucht am 22. 08. 2021) (siehe S. 17, 18, 24).

[25] google. *Speech-to-Text: Automatische Spracherkennung*. Google Cloud. URL: https://cloud.google.com/speech-to-text?hl=de (besucht am 23. 08. 2021) (siehe S. 19).

[26] google. *TeachableMachine*. URL: https://teachablemachine.withgoogle.com/ (besucht am 23. 08. 2021) (siehe S. 25, 30–34).

[27] google. *Text-to-Speech: Lebensechte Sprachsynthese*. Google Cloud. URL: https://cloud.google.com/text-to-speech?hl=de (besucht am 25. 08. 2021) (siehe S. 26).

[28] Guobin Wu. *Kinect Sign Language Translator - part 1*. Microsoft Research. 29. Okt. 2013. URL: https://www.microsoft.com/en-us/research/blog/kinect-sign-language-translator-part-1/ (besucht am 28. 07. 2021) (siehe S. 1).

[29] IBM. *IBM Cloud Learn Hub*. 12. Aug. 2020. URL: https://www.ibm.com/cloud/learn (besucht am 26. 07. 2021) (siehe S. 5, 10, 13).

[30] J. Gordon Betts, Tyler Junior College u. a. *Nervous Tissue · Anatomy and Physiology*. Apr. 2013. URL: https://philschatz.com/anatomy-book/contents/m46509.html (besucht am 14. 07. 2021) (siehe S. 5, 6).

[31] Johann Lau. *Google Maps 101: How AI helps predict traffic and determine routes*. blog.google. 3. Sep. 2020. URL: https://blog.google/products/maps/google-maps-101-how-ai-helps-predict-traffic-and-determine-routes/ (besucht am 14. 07. 2021) (siehe S. 1).

[32] Laurenz Wuttke. *Machine Learning vs. Deep Learning: Wo ist der Unterschied?* datasolut GmbH. 19. Feb. 2021. URL: https://datasolut.com/machine-learning-vs-deep-learning/ (besucht am 17. 08. 2021) (siehe S. 9).

[33] Laurenz Wuttke. *Machine Learning: Definition, Algorithmen, Methoden und Beispiele*. datasolut GmbH. 11. Aug. 2020. URL: https://datasolut.com/was-ist-machine-learning/ (besucht am 27. 07. 2021) (siehe S. 10).

[34] Yan LeCun. *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. URL: http://yann.lecun.com/exdb/mnist/ (besucht am 17. 08. 2021) (siehe S. 9).

[35] Lucy Suchman. *Unpriming the pump: Remystifications of AI at the UN's Convention on Certain Conventional Weapons | ICRAC*. Apr. 2018. URL: https://www.icrac.net/unpriming-the-pump-remystifications-of-ai-at-the-uns-convention-on-certain-conventional-weapons/ (besucht am 14. 07. 2021) (siehe S. 4).

[36] Matlab. *Object Recognition*. URL: https://www.mathworks.com/solutions/image-video-processing/object-recognition.html (besucht am 22. 08. 2021) (siehe S. 13).

[37] ml5. *Handpose*. URL: https://learn.ml5js.org/#/reference/handpose (besucht am 22. 08. 2021) (siehe S. 18).

[38] Node.js. *About*. Node.js. URL: https://nodejs.org/en/about/ (besucht am 22. 08. 2021) (siehe S. 18).

[39] Oculus. *Introducing Hand Tracking on Oculus Quest—Bringing Your Real Hands into VR*. 25. Sep. 2019. URL: https://www.oculus.com/blog/introducing-hand-tracking-on-oculus-quest-bringing-your-real-hands-into-vr/ (besucht am 28. 07. 2021) (siehe S. 1).

[40] OpenAI. *OpenAI's GPT-3 Language Model: A Technical Overview*. Lambda Blog. 3. Juni 2020. URL: https://lambdalabs.com/blog/demystifying-gpt-3/ (besucht am 27. 07. 2021) (siehe S. 10).

[41] opencv. *opencv*. URL: https://opencv.org/ (besucht am 25. 08. 2021) (siehe S. 19, 24).

[42] p5.js. *home*. URL: https://p5js.org/ (besucht am 22. 08. 2021) (siehe S. 18, 37).

[43] Patrick Clary. *Use your voice to access the world with a new Android app*. Google. 2. Okt. 2018. URL: https://blog.google/outreach-initiatives/accessibility/use-your-voice-access-world-new-android-app/ (besucht am 01. 09. 2021) (siehe S. 1).

[44] picovoice. *Picovoice: Edge Voice AI Platform*. URL: https://picovoice.ai/ (besucht am 23. 08. 2021) (siehe S. 20).

[45] RayT. *CNN with Pytorch using Mel features | Kaggle*. URL: https://www.kaggle.com/talmanr/cnn-with-pytorch-using-mel-features/notebook (besucht am 25. 08. 2021) (siehe S. 27, 28).

[46] *Scale of the Human Brain*. AI Impacts. Section: AI Timelines. 16. Apr. 2015. URL: https://aiimpacts.org/scale-of-the-human-brain/ (besucht am 14. 07. 2021) (siehe S. 5).

[47] *VOSK Offline Speech Recognition API*. VOSK Offline Speech Recognition API. URL: https://alphacephei.com/vosk/ (besucht am 21. 08. 2021) (siehe S. 20).

[48] Waymo. *Waymo*. Waymo. 2021. URL: https://waymo.com/ (besucht am 28. 07. 2021) (siehe S. 1).

[49] Redaktion Wirtschaftsforum.de. *Künstliche Intelligenz versus Machine Learning - Gefährliche Bedrohung oder perfekte Chance?* wirtschaftsforum.de. URL: https://www.wirtschaftsforum.de/expertenwissen/was-ist-denn-data-science/kuenstliche-intelligenz-versus-machine-learning/ (besucht am 17. 08. 2021) (siehe S. 7, 8).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —

width = 100mm
height = 50mm

— Diese Seite nach dem Druck entfernen! —