



# **Die Entwicklung einer gestengesteuerten Drohne**

**Maturaarbeit**

**Gymnasium Muttenz**

**Luis Zusman 4A**

**Betreuungsperson: Alex van den Brandhof**

**14. Oktober 2024**

## Inhaltsverzeichnis

1	Vorwort .....	1
2	Einleitung .....	3
3	Theoretische Grundlagen .....	4
3.1	Was ist Künstliche Intelligenz? .....	4
3.2	Wie funktioniert KI? .....	5
3.3	Keywords .....	7
3.4	Arten von Datensätzen .....	9
3.5	Voraussetzungen für einen guten Datensatz .....	9
3.6	Das Modell .....	10
3.7	CNN .....	11
4	Das Produkt .....	12
4.1	Das Erstellen des Datensatzes .....	12
4.2	Das 1. Modell .....	13
4.3	Exkurs: Filtergrößen .....	13
4.4	Das 2. Modell .....	15
4.5	Exkurs: Transferlearning .....	15
4.6	Finales Modell und Mediapipe .....	16
4.7	Die Drohne .....	17
5	Resultate .....	20
5.1	1. Modell .....	20
5.2	1. Modell mit grossen Filtern .....	21
5.3	2. Modell (angepasste Filter) .....	22
5.4	Finales Modell (Mediapipe-Modell) .....	23
5.5	Vergleich .....	24
6	Diskussion .....	25
7	Schlusswort .....	26
8	Verzeichnisse .....	27
8.1	Literaturverzeichnis und Internetquellen .....	27
8.2	Abbildungsverzeichnis .....	28
9	Anhang .....	30
10	Redlichkeitserklärung .....	31

# 1 Vorwort

«Hallo! Es ist 8:15 und in Muttenz ist es 22 Grad warm». Derselbe Roboter, welcher mich eben geweckt hat, bringt mir nun mein Frühstück ans Bett. Während ich meinen Orangensaft trinke, befehle ich dem Roboter mit einer leichten Handbewegung die Storen zu öffnen... Während dieses Szenario noch vor wenigen Jahren komplett im Gebiet der Science-Fiction verortet worden wäre, scheint es heute mit den Fortschritten in Künstlicher Intelligenz gar nicht mehr so unrealistisch.

Schon immer haben mich technische Errungenschaften begeistert und die KI hat bereits beim ersten Kontakt mein Interesse geweckt. Durch die Lektüre von "Machine Learning für Softwareentwickler"<sup>1</sup> habe ich mich vor einigen Jahren das erste Mal etwas vertiefter mit dem Thema beschäftigt. Paolo Perrotta führt darin den Leser Schritt für Schritt in das Thema KI ein. Mithilfe dieses Buches schrieb ich Anfang des Gymnasiums – mit ausschliesslich einfachen mathematischen Funktionen wie Matrizenmultiplikationen oder Durchschnittsfunktionen – ein Programm, welches handgeschriebene Zahlen aus dem *MNIST-Datensatz* (Abb. 1) erkennen konnte.



Abbildung 1: *MNIST Datensatz*

Noch immer fasziniert von der damaligen Erfahrung, wo ich einen Einblick in die Funktionsweise von KI-Systemen erhalten hatte, stand für mich bei der Suche der Maturaarbeit ganz schnell fest, dass dies eine einzigartige Gelegenheit ist, mich noch während meiner Schulzeit intensiv auf das Thema einzulassen und endlich einmal ein richtiges Programmierprojekt mit einer praktischen Anwendung zu realisieren.

Auf der Suche nach möglichen Ideen für eine Maturaarbeit im Bereich KI bin ich auf «*kaggle*»<sup>2</sup>, einer Internetseite, welche viele Datensätze enthält, auf interessante Themen gestossen und habe diese ausprobiert. Unter anderem faszinierte mich das maschinelle Erkennen von Gebärdenzeichen und der Anzahl gezeigter Finger. Letzteres

---

<sup>1</sup> Perrotta, Paolo: Machine Learning für Softwareentwickler. Von der Python-Codezeile zur Deep-Learning-Anwendung, Heidelberg 2020.

<sup>2</sup> Kaggle, [www.kaggle.com](http://www.kaggle.com), (12.9.2024)

funktionierte schon ganz gut und brachte mich auf eine neue Idee: Einen Roboter mit Hilfe von Finger Zeichen zu steuern.

Ich überlegte, aus einem Raspberry Pi einen Roboter zusammenzubauen, doch dann fiel mir plötzlich die Drohne ein, welche ich einige Jahre zuvor gekauft hatte und die bis jetzt nur Staub gefangen hatte. Diese Drohne ist mit einer App des Anbieters von einem Mobiltelefon aus steuerbar und zu meinem Vorteil auch programmierbar. Somit stand meine Idee fest:

### **Die Steuerung einer Drohne durch Fingerzeichen.**

Mit diesem Projekt erhoffe ich mir, mehr praktische Erfahrung im Bereich KI zu sammeln und zu lernen, welche Schritte notwendig sind, um so ein Problem anzugehen und wissenschaftlich zu dokumentieren.

Ich danke Herrn Van den Brandhof für seine positive und unterstützende Begleitung meines Projekts. Ebenfalls möchte ich mich bei Herrn Geissmann bedanken, der mir ganz zu Beginn der Arbeit wichtige Impulse gegeben hat. Auch danke ich meiner Mutter, Barbara von Wartburg, für das Gegenlesen der Arbeit.

## 2 Einleitung

Das Erkennen und Verstehen von Gesten spielen nicht nur im utopischen Beispiel meines Vorworts eine Rolle. Schon heute finden sie in mehreren Technologien Anwendung. So wollen beispielsweise Mobiltelefonanbieter ihre Produkte immer intuitiver machen: von Tasten über Trackpad und vor einigen Jahren schliesslich zum Multitouchdisplay. Anfang dieses Jahres ist Apple Vision Pro auf den Markt gekommen, eine VR-Brille, deren eingebundene Anzeigen ausschliesslich durch Gesten und Handbewegungen gesteuert werden können.

Der Gegenstand der Maturaarbeit soll ein kleines Projekt in diesem Entwicklungsfeld sein. Die Arbeit wird sich aus der Darstellung der theoretischen Grundlagen einerseits und der Entwicklung eines Programms andererseits zusammensetzen.

Die Leitfragen, die dieser Arbeit zugrunde liegen, lauten demnach:

- **Wie kann eine Computer Objekte auf Bildern erkennen am Beispiel von Gestenerkennung?**
- **Wie schreibt man ein Programm, welches Gesten erkennt und damit eine Drohne steuert?**

Ich werde im Rahmen dieser Maturaarbeit eine KI programmieren, welche meine Handzeichen erkennt. Die KI werde ich mit Python, eine Programmiersprache, welche ich mir vor einigen Jahren selbst beigebracht habe und welche die am häufigsten verwendete Sprache für KI ist, entwickeln. Ich werde das Programm in Jupyternotebooks in Visualstudio-Code schreiben. Somit kann ich einzelne Abschnitte mit einem Titel beschriften und Erklärungen zum Code hinzufügen. Für die Programmierung der KI werde ich in erster Linie TensorFlow verwenden, welches viele für Maschinelles Lernen spezialisierte Funktionen und Tools bereitstellt. Für die finale Version kommt noch ein Programm von Mediapipe zum Tragen.

Die ganze Arbeit, inklusive der Programme, Videos der gesteuerten Drohne und das trainierte KI-Modell, werde ich auf GitHub öffentlich zur Verfügung stellen (QR-Code).

Die hier vorliegende schriftliche Dokumentation des Projekts gibt in einem ersten Schritt eine kleine Einführung in die Grundlagen von Maschinellem Lernen und klärt einige wichtige Keywords für die folgenden Kapitel.

Der zweite Teil (Kap. 4) beschäftigt sich mit der Entwicklung des Produktes. Nachdem aufgezeigt wird, wie der Datensatz für das Training des Algorithmus erstellt wurde, wird die erste Version des Programms vorgestellt und auf eine spezielle Architektur für Bilderkennung eingegangen. Nach der Verbesserung des ersten Prototyps wird in einem kurzen Exkurs das Thema Transferlearning angespielt, eine Technik im Programmieren von KI, mithilfe welcher die finale Version des Modells geschrieben wurde.

Im letzten Unterkapitel des praktischen Teils (Kap. 4.7) wird schliesslich aufgezeigt, wie das finale Modell auf die Drohne übertragen wurde, um diese zu steuern.

Die in Kapitel 5 dargestellten Resultate werden in der Diskussion (Kapitel 6) interpretiert. Die Stärken und Schwächen des entwickelten Programms werden analysiert und mögliche Weiterentwicklungen vorgestellt.

### 3      Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Hintergründe der vorliegenden Maturaarbeit beleuchtet. Nachdem Begriffe wie "Künstliche Intelligenz", "Maschinelles Lernen" erklärt werden, wird die Funktionsweise dieser Algorithmen auf unterster Stufe erklärt wie auch mathematisch und bildlich dargestellt. Nach dem Definieren weiterer Keywords wird das Konzept und die Rolle von Datensätzen erläutert. Zum Schluss wird das Herz der KI, das KI-Modell präsentiert.

#### 3.1 Was ist Künstliche Intelligenz?

Was ist ein KI eigentlich? Wie ist es möglich, dass ein Programm basierend auf mathematischen Operationen in der Lage ist, so komplexe Probleme wie das Erkennen von Gesten in einem Bild, zu lösen?

Das Europäische Parlament definiert Künstliche Intelligenz wie folgt:

"Künstliche Intelligenz ist die Fähigkeit einer Maschine, menschliche Fähigkeiten wie logisches Denken, Lernen, Planen und Kreativität zu imitieren.

KI ermöglicht es technischen Systemen, ihre Umwelt wahrzunehmen, mit dem Wahrgenommenen umzugehen und Probleme zu lösen, um ein bestimmtes Ziel zu erreichen. Der Computer empfängt Daten (die bereits über eigene Sensoren, zum Beispiel eine Kamera, vorbereitet oder gesammelt wurden), verarbeitet sie und reagiert.

KI-Systeme sind in der Lage, ihr Handeln anzupassen, indem sie die Folgen früherer Aktionen analysieren und autonom arbeiten.<sup>3</sup>

Künstliche Intelligenz (KI) ist der Überbegriff für alle Programme, welche eine menschliche Fertigkeit erlernen können. Dabei kann es sich um eine Klassifizierung handeln, wie beim erwähnten Problem der Erkennung von Handzeichen, um eine Regression (z.B. die Vorhersagen von Börsendaten) oder auch um eine generative Leistung, wie bei ChatGPT oder Dall-E, welche einen Inhalt kreieren können. Der Begriff "Maschinelles Lernens" (ML) beschreibt den Vorgang, wie ein Computer solche Fähigkeiten erlernen kann. Der verbreitetste Ansatz dafür sind künstliche Neuronale Netze, welche später noch vertieft angeschaut werden. Sie basieren, wie der Name schon vermuten lässt, auf der Grundlage von biologischen Neuronen. Sie empfangen, verarbeiten und leiten Informationen weiter, um so einen Output zu berechnen.<sup>4</sup>

---

<sup>3</sup> Was ist künstliche Intelligenz und wie wird sie genutzt?, Europäisches Parlament, <https://www.europarl.europa.eu/topics/de/article/20200827STO85804/was-ist-kunstliche-intelligenz-und-wie-wird-sie-genutzt>, (4.9.2024)

<sup>4</sup> Künstliches neuronales Netz, Wikipedia, [https://de.wikipedia.org/wiki/K%C3%BCnstliches\\_neuronales\\_Netz](https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz), (6.9.2024)

Deep Neural Network (DNN) und das dazugehörige Deep Learning (DL) erzielen heutzutage unglaubliche Ergebnisse. Sei es ein präziser Übersetzer wie DeepL, welche die darunterliegende Technologie im Namen trägt, oder ein Large Language Model (LLM) wie ChatGPT. Dabei beruht dieses Deep Learning im Prinzip lediglich auf einem neuronalen Netz mit vielen Schichten. Diese Schichten sind höchst spezialisiert. Somit können sie komplexe Korrelationen und Zusammenhänge in Daten erkennen, weshalb sie für schwierige und umfangreiche Probleme genutzt werden. Da diese vielschichtigen neuronalen Netze erst durch leistungsfähige Computer funktionieren, kamen diese erst in den letzten Jahren auf.

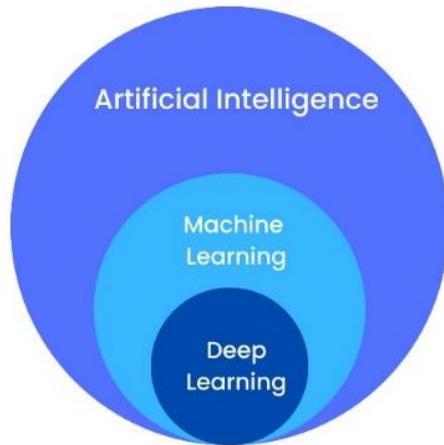


Abbildung 2: Begriffserklärung: KI, ML und DL

### 3.2 Wie funktioniert KI?

Nach der Klärung des Begriffs KI wird nun die Grundlage der Funktionsweise beleuchtet. Die allermeisten ML-Algorithmen basieren auf dem sogenannten *Gradientabstieg*.

Als Heranführung kann das folgende einfache Beispiel dienen:

Wir versuchen mit Hilfe von linearer Regression, Wohnungspreise anhand deren Wohnfläche vorauszusagen. Wir haben Daten von X Wohnungen in der Schweiz gesammelt und stellen diese in einem 2D Koordinatensystem dar (Abb.3).

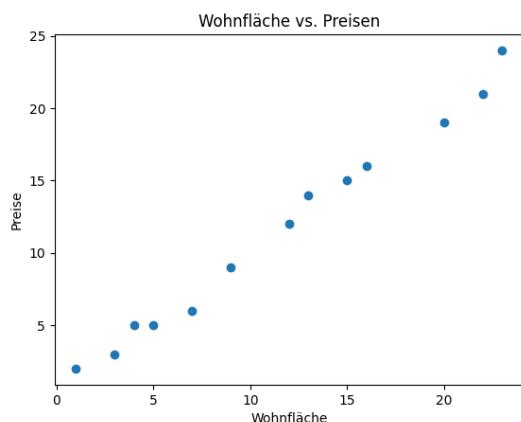


Abbildung 3: Punktediagramm: Wohnfläche vs. Preise

Nun suchen wir eine Gerade, mit der Form  $a*x$ , wobei  $x$  die entsprechende Wohnfläche ist. Diese Gerade soll möglichst genau an die Punkte angepasst sein, so dass wir danach anhand der Wohnfläche Voraussagen für die Preise machen können. Einfachheitshalber lassen wir die Konstante  $b$ , welche in der Geradengleichung den Y-Achsenabschnitt beschreibt, weg, so dass die Gerade aus dem Ursprung entspringt.

Wir setzen  $a$  vorerst auf 0 und schauen uns die Gerade an.

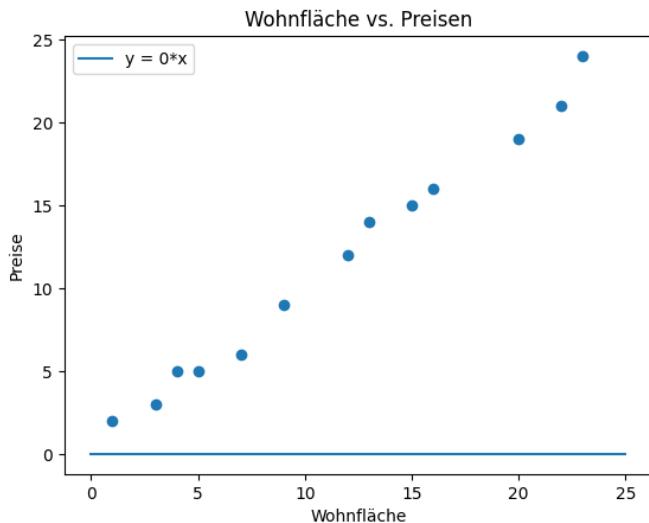


Abbildung 4: Punktendiagramm mit Gerade  $a = 0$

Die resultierende Gerade ist nicht hilfreich, da sie die Verteilung der Daten in keiner Weise darstellt. Der Fehler der entstandenen Geraden lässt sich mit der folgenden Verlustfunktion beschreiben:

*MSE (Mean squared error, deutsch: mittlere quadratische Abweichung)*

$$L(y, y_{pred}) = \frac{1}{N} \sum (y - y_{pred})^2$$

Wir addieren also jeweils das Quadrat der Differenz zwischen unserer Vorhersage und dem wirklichen Preis und dividieren schliesslich die Summe durch die Anzahl Beispiele ( $N$ ).  $y$ , welches die wirklichen Preise darstellt, ist konstant, während  $y_{pred}$ , unsere Vorhersage, von  $a$  abhängt.

Somit lässt sich unsere Verlustfunktion relativ zu  $a$  in einem Koordinatensystem darstellen (Abb. 5). In diesem Graphen lässt sich nun den besten Wert für  $a$  ablesen:

Der tiefste Punkt auf dem Graphen, oder in Fachsprache das globale Minimum, beschreibt den kleinsten Verlust und somit den bestmöglichen Wert für  $a$ , welcher sich ungefähr bei  $a = 1$  befindet. In Abbildung 6 können wir unsere Vorhersagekurve mit  $a = 1$  betrachten.

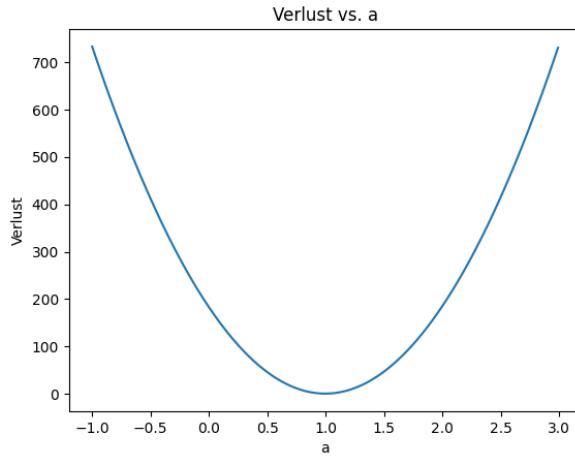


Abbildung 5: Verlustkurve

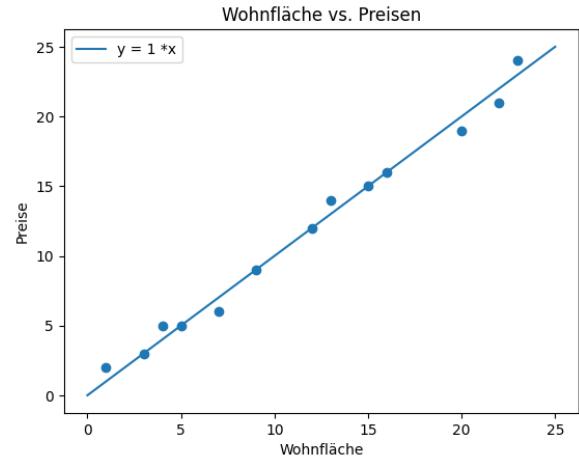


Abbildung 6: Punktediagramm mit  $a = 1$

Mit  $a = 0$  starten wir ganz oben auf der Kurve. Nun stellen wir uns eine Kugel vor, welche durch die Gravitation immer dem steilsten Weg nach unten folgt. Mathematisch kann diese steilste Route nach unten mit dem negativen Gradienten bestimmt werden. Somit suchen wir in jeder Iteration den steilsten Abhang und machen einen Schritt in diese Richtung. Dieser Schritt ist grösser, wenn der Abhang steiler ist, und kleiner, wenn dieser abflacht. Folglich bleiben wir, wenn wir unten sind, stehen, da dort keine Steigung vorliegt und die Schritte 0 werden. Diese Technik wird *Gradientabstieg* genannt. Der Vorgang dieser Anpassung von  $a$  wird als *Training* bezeichnet.

Dies sind die Grundlagen, auf welchen die meisten KI-Anwendungen basieren. Für die meisten realen Probleme und Aufgaben werden jedoch viel grössere Datenmengen in deutlich höheren Dimensionen gebraucht. Auch die Netze werden viel komplexer, ausfeilter und problemangepasster, um die Korrelationen zwischen den oft auch nicht linearen Daten besser erkennen zu können.

Für spezifische Anwendungen gibt es spezialisierte Architekturen, welche jedoch immer noch auf dem gleichen Prinzip basieren. Für Bilderkennung performen sogenannte *Faltungsnetze* (eng. Convolutional Neural Network CNN) besonders gut, weshalb in dieser Arbeit (Kap. 3.7) noch vertiefter auf diese Architektur eingegangen wird.

### 3.3 Keywords

Um im weiteren Verlauf der Arbeit auf detailliertere Aspekte von KI eingehen zu können, werden im Folgenden einige Schlüsselbegriffe geklärt.

#### 3.3.1 Überwachtes und unüberwachtes Lernen

Beim maschinellen Lernen unterscheidet man *überwachtes* und *unüberwachtes Lernen*. Beim überwachten Lernen hat man einen Datensatz (z.B. mehrere Bilder) und die dazu gehörige „Lösung“ (in Fachsprache *Labels*). Im Falle der Gestenerkennung wäre dies der Name der auf dem Bild gezeigten Geste. Beim unüberwachten Lernen soll das Programm selbst durch Zusammenhänge in den Daten die Lösung erkennen.

### 3.3.2 Layers und Modell

Komplexere Probleme fordern oft nicht-lineare Netze. Ein komplexes nicht-lineares Neuronales Netz besteht aus mehreren Schichten. Diese Schichten werden *Layers* genannt und enthalten die Gewichte, welche während des Trainings erlernt werden. Das Modell, welches im Kapitel 3.6 genauer beleuchtet wird, beschreibt den Aufbau der KI. Es enthält unter anderem die Gewichte und die Wahl der Hyperparameter.

### 3.3.3 Aktivierungsfunktionen

Aktivierungsfunktionen sind nicht-lineare Funktionen, welche auf den Output einer Layer angewendet werden. Sie verleihen mehrschichtigen neuronalen Netzen ihre Nicht-Linearität. In Kombination mit mehreren Layers sind sie in der Lage, komplexe, nicht lineare Zusammenhänge zu erkennen. Ein Beispiel für eine Aktivierungsfunktion ist die Sigmoidfunktion (Abb.7).

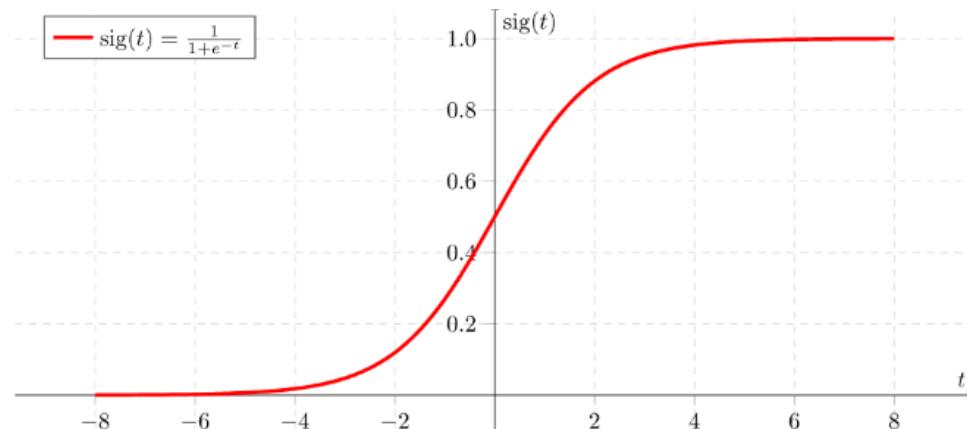


Abbildung 7: Sigmoidfunktion

### 3.3.4 Parameter und Hyperparameter

Parameter sind die während dem Training erlernten Gewichte. OpenAIs GPT4 hat schätzungsweise mehr als 1.8 Billionen Parameter.<sup>5</sup> Hyperparameter dagegen, sind Kontrollwerte, welche manuell festgelegt werden. Sie bestimmen beispielsweise die Anzahl Layers, die Anzahl Trainingsiterationen (Epochen), die Aktivierungsfunktion, die Verlustfunktion oder die Filtergrößen in einem CNN, auf welche hier später noch genauer eingegangen wird.<sup>6</sup>

<sup>5</sup> Howarth, Josh, Anzahl der Parameter in GPT-4, Exploding Topics [https://explodingtopics.com.translate.goog/blog/gpt-parameters?\\_x\\_tr\\_sl=en&\\_x\\_tr\\_tl=de&\\_x\\_tr\\_hl=de&\\_x\\_tr\\_pto=rq&\\_x\\_tr\\_hist=true#](https://explodingtopics.com.translate.goog/blog/gpt-parameters?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=rq&_x_tr_hist=true#), (19.9.2024)

<sup>6</sup> Nyuytiymby, Kizito, Parameters and Hyperparameters in Machine Learning and Deep Learning, Medium, <https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac>, (10.10.2024)

### 3.3.5 Over- und Underfitting

Die zwei häufigsten Probleme beim Trainieren eines neuronalen Netzes sind die Überanpassung oder Unteranpassung an die Daten, auf Englisch das *Overfitting* und das *Underfitting*.<sup>7</sup> Beim Overfitting ist das Modell „zu gut“ an die Trainingsdaten angepasst. Es lernt nicht relevante Merkmale. Das Modell performt gut an den Trainingsdaten, versagt jedoch an den Testdaten. Beim Underfitting ist das Gegenteil der Fall. Das Netz lernt nicht genügend Merkmale. Es performt dann meist bei den Trainingsdaten als auch bei den Testdaten schlecht (Vergleich Abb. 8).

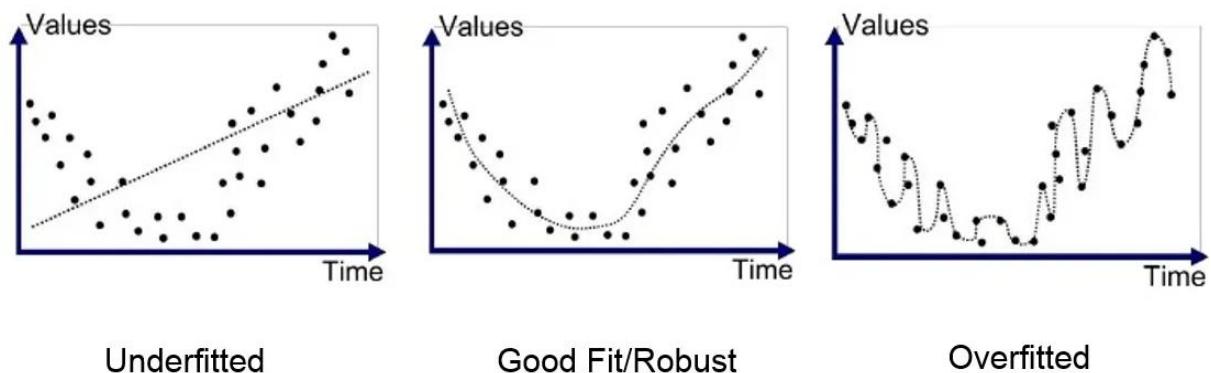


Abbildung 8: Vergleich Under- & Overfitting

## 3.4 Arten von Datensätzen

Um ein Modell zu trainieren, braucht es erstmal einen Datensatz. Datensätze gibt es in aller Art. Dies können Tonaufnahmen sein, um eine KI zu trainieren menschliche Stimmen zu imitieren, Texte, um ein Large Language Model (LLM) wie ChatGPT zu trainieren, Daten aus vergangenen Börsenaktivitäten, um zukünftige vorherzusagen oder eben Bilder von Handzeichen, um diese danach live zu erkennen und damit eine Drohne zu steuern.

## 3.5 Voraussetzungen für einen guten Datensatz

Die Qualität des Datensatzes ist unerlässlich für das Training eines verlässlichen Modells. Ein Modell kann noch so komplex oder lange trainiert werden, wenn die Daten fehlerhaft, nicht einheitlich, zu wenige oder nicht vielfältig genug sind, hat der Computer keine Chance Zusammenhänge zwischen den Zahlen zu erkennen. Für einen brauchbaren Datensatz gibt es mehrere Voraussetzungen<sup>8</sup>, die im Folgenden aufgeführt werden.

<sup>7</sup> Biswal, Avijeet, The Complete Guide on Overfitting and Underfitting in Machine Learning, simplilearn, <https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting>, (5.9.2024)

<sup>8</sup> Grösse und Qualität eines Datensatzes, Machine Learning, <https://developers.google.com/machine-learning/crash-course/overfitting?hl=de>, (16.8.2024)

### 3.5.1 Anzahl

Um verlässliche Erkenntnisse zu erhalten, muss der Datensatz genügend umfangreich sein. Die Anzahl Daten hängt von verschiedenen Faktoren ab:

Einerseits spielt die Komplexität der Daten eine Rolle: Wie schwierig ist das zu lösende Problem? Die Gestenerkennung beispielsweise ist weniger kompliziert vor einer weißen Wand als in nicht störungsfreier Umgebung. Weiter kommt es auf die Anzahl der zu klassifizierenden Labels an. Man braucht (pro Label) mehr Daten, um ein Modell zu trainieren, zwei Klassen voneinander zu unterscheiden, als wenn 500 Klassen differenziert werden müssen. Zudem ist die Komplexität des KI-Modells wie auch die gewünschte Genauigkeit und Robustheit massgeblich. So erfordert es mehr Daten, um ein Netz mit hunderttausend Parametern zu trainieren als eines mit nur einigen hundert.

### 3.5.2 Aufteilung

Die Daten müssen gleichmäßig aufgeteilt werden. Für das Beispiel einer Gestenerkennung sollte der Datensatz also nicht zwei Fotos der ersten Geste und 100 Fotos der zweiten Geste enthalten. In gewissen Kontexten ist dieses Kriterium eher schwierig zu erfüllen, wenn z.B. in medizindiagnostischen Tools für die Bestimmungen von Krankheitsbildern die Mehrzahl der Getesteten gesund sind.

### 3.5.3 Vielfältigkeit

Der Datensatz sollte möglichst vielfältige Einträge beinhalten. Wiederum am Beispiel der Gestenerkennung verdeutlicht, sollte anstelle der gleichen Hand, mit dem gleichen Abstand zur Kamera und immer dem gleichen Hintergrund, mit den genannten Variablen gespielt werden, damit das Netz möglichst viele verschiedene Vorkommnisse und Fälle sieht.

## 3.6 Das Modell

Das Modell beschreibt, wie das neuronale Netz aufgebaut ist, und enthält die Parameter, welche während dem Training erlernt werden. Der Aufbau des Modells beschreibt, welche Elemente dieses enthält, wie viele Neuronen die einzelnen Schichten haben, welche Aktivierungsfunktionen genutzt werden und ob noch spezielle Elemente wie z.B. CNN-Layers enthalten sind. Der Aufbau des Modells ist essenziell und muss auf das Problem und die Daten angepasst sein. So macht es keinen Sinn, ein hochkomplexes Netz für ein linear abhängiges Problem, wie etwa der Zusammenhang zwischen Wohnfläche und Preis in unserem Anfangsbeispiel (Kap.3.2) zu verwenden. Ein lineares Modell auf einen nicht-linearen Trend zu trainieren ist jedoch auch zwecklos. Durch das Trainieren des Modells können dessen Parameter auf die Daten angepasst werden, um danach einen Output liefern zu können. In der folgenden Abbildung ist ein dreischichtiges Modell zu sehen. Die Layers haben alle die Sigmoid Aktivierungsfunktion (Abb.9).

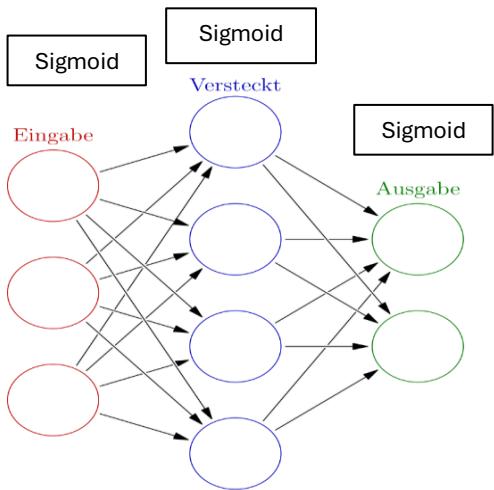


Abbildung 9: KI-Modell mit 3 Layers

### 3.7 CNN

Für Bilderkennung wird meistens eine spezielle Art von neuronalen Netzen verwendet: Convolutional Neural Networks (CNN) oder zu Deutsch *Faltungsnetze*. Das Prinzip dieser Netze basiert darauf, kleine rechteckige Filter über das Bild laufen zu lassen und eine Operation (Frobenius-Skalarprodukt<sup>9</sup>) zwischen dem Filter und den darunterliegenden Pixel anzuwenden (s. Abb. 10 und die dazugehörige Animation auf Github Repository). Der Output der CNN-Layer ist ein weiteres Bild. CNN sind so effektiv, da sie lokale Merkmale erkennen und hervorheben können. Dies könnten wie in Abbildung 11 die Kanten und Ecken sein, jedoch auch komplexere Formen wie Gesichter und Gegenstände.

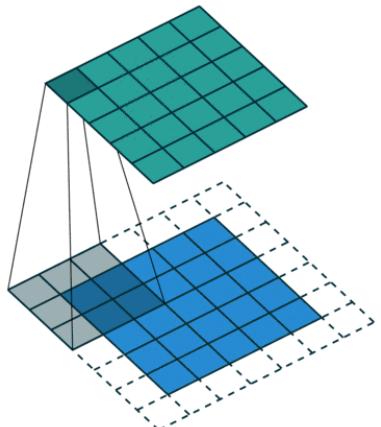


Abbildung 10: Funktionsweise von CNN

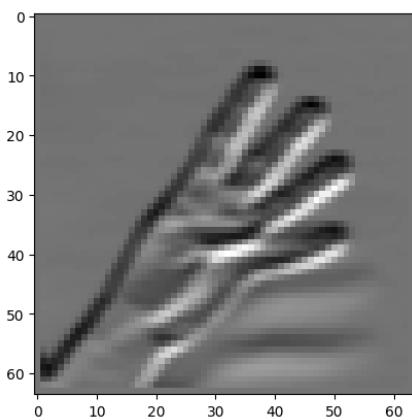


Abbildung 10: Kantenerkennung durch CNN Filter

Nach einer oder mehreren dieser CNN Layers wird das entstandene Bild in ein «normales» neuronales Netz eingespielen, welches dann mit den hervorgehobenen Merkmalen einfacher einen Output liefern kann.<sup>10</sup>

<sup>9</sup> Die Multiplikation der übereinanderliegenden Pixel und die anschliessende Summe dieser Produkte. Vgl. Frobenius-Skalarprodukt, Wikipedia, <https://de.wikipedia.org/wiki/Frobenius-Skalarprodukt>, (11.10.2024)

<sup>10</sup> Convolutional Neural Network, Wikipedia, [https://de.wikipedia.org/wiki/Convolutional\\_Neural\\_Network#:~:text=Ein%20Convolutional%20Neural%20Network%20\(CNN%20oder%20ConvNet\),%20zu,\\_\(27.9.2024\)](https://de.wikipedia.org/wiki/Convolutional_Neural_Network#:~:text=Ein%20Convolutional%20Neural%20Network%20(CNN%20oder%20ConvNet),%20zu,_(27.9.2024))

## 4 Das Produkt

Im folgenden Teil dokumentiere und erkläre ich, wie ich mein Produkt entwickelt habe: Ich begann damit, den Datensatz zu erstellen. Daraufhin designte ich ein Modell, welches anhand dieser Daten die Zeichen erlernen und klassifizieren soll. Nach einem Exkurs in das Thema Filtergrößen begann ich mithilfe der neuen Erkenntnisse die Überarbeitung des ersten Prototyps. Für die finale Version meines Programmes wandte ich mich dem Thema Transferlearning zu und implementierte ein Tool namens *Mediapipe*, um die Zeichenerkennung robuster und sicherer für die Drohnensteuerung zu machen. Zum Schluss galt es noch, die Drohne zu steuern und die KI auf die Drohne zu übertragen.

### 4.1 Das Erstellen des Datensatzes

Einen Datensatz zu erstellen ist oft sehr zeitintensiv, weshalb man, wenn möglich, auf Vorgefertigte zurückgreift. Im Internet gibt es eine riesige Menge davon. Auf kaggle.com, einer Webseite für KI-Interessierte bis ML-Ingenieure gibt es Kurse, Diskussionen, Foren, Wettbewerbe und vor allem eine grosse Anzahl an Datensätzen.

Nach langem Suchen fand ich mehrere Datensätze, welche gewisse Fingerzeichen oder Handgesten enthielten, jedoch fand ich keines, welches mir ganz entsprach. Entweder hatten sie zu wenige Zeichen, die Zeichen wurden nur vor einer weissen Wand gezeigt oder die Zeichen waren nicht sinnvoll für die Steuerung der Drohne. Somit entschied ich mich dazu einen eigenen Datensatz zu erstellen.

Von einem Freund lieh ich mir eine Webcam und machte mich daran, ein Programm zu schreiben, welches mehrere Fotos pro Sekunde schoss und in einem Ordner abspeicherte. Ich habe vier Zeichen (fist 🤚, telephone 📞, thumb 🤝, hand 🤖) definiert und musste diese nun lediglich in verschiedenen Posen mit verschiedenen Abständen und Kleidungsstücken in die Kamera zeigen. Um nicht unnötig Rechenleistung und Speicherplatz zu brauchen, verkleinerte ich die Fotos, so dass die wesentlichen Merkmale immer noch gut erkennbar waren (100x100x3).

Die Bilder habe ich dann in ein Trainingsset und ein Testset aufgeteilt. Das Testset wird nicht beim Training gebraucht und dient zum Vergleich, ob die KI die Merkmale wirklich erkennt oder einfach die Trainingsbilder „auswendig gelernt hat“.

## 4.2 Das 1. Modell

Nun galt die Frage, wie ich das Netz aufbaue: Wie viele Filter und welche Grösse, wie viele Schichten, usw.

Ich recherchierte im Internet und lehnte mich an den Aufbau eines Modelles für ein ähnliches Problem<sup>11</sup> und passte einige Hyperparameter an (Input Grösse, Output Grösse). Zu meinem Verwundern funktionierte das Modell reibungslos und ich erzielte über 99% Genauigkeit sowohl bei Trainings und Testdaten. Umso mehr war ich enttäuscht, als ich das Programm live testen wollte und lauter falscher Resultate bekam.

Das Programm war an die Trainingsdaten überangepasst (overfittet, Kap. 3.3.5), obwohl auch die Testdaten eine so hohe Genauigkeit hatten. Das Problem lag jedoch daran, dass die Bilder zu schnell hintereinander aufgenommen wurden, und daher oft sehr ähnlich wie die Bilder zuvor und danach waren. Somit waren Test- und Trainingsdaten oft so ähnlich, so dass das „Auswendigelernte“ auch auf die Testdaten übertragbar war.

Um diesen Fehler zu beheben, verwendete ich für die Testdaten einen komplett neuen Datensatz, welcher somit nicht von den Trainingsdaten abgeschaut werden konnte. Für das Erstellen der Trainingsdaten verlängerte ich die Zeit zwischen den Fotos, fügte weitere Bilder hinzu und machte den Datensatz so grösser und vielfältiger. Auch versuchte ich, es durch verschiedene Personen und Hintergründe noch abwechslungsreicher zu gestalten. Ich fügte so viele Fotos hinzu, dass der Datensatz zum Schluss fast 9'000 Bilder enthielt.

Ich liess das Modell ein weiteres Mal trainieren und das Programm scheiterte erneut. Jedoch widerspiegelte die Genauigkeit nun wirklich, wie schlecht das Modell performte. Durch die vielfältigeren Trainingsdaten konnte das Modell die Daten nicht mehr „auswendig“ lernen, jedoch schien es die Handzeichen auch nicht zu erkennen. Das Problem musste nun also beim Aufbau des Modells liegen.

## 4.3 Exkurs: Filtergrößen

In meiner ersten Version waren die Filter sehr klein. Deshalb hat mir der um Rat gefragte Lehrer, Herr Geissmann, empfohlen, grössere Filter zu verwenden, so dass diese auch einen grösseren Teil der zu erkennenden Merkmale überdecken und diese somit besser erkennen können.

Ich spielte mit verschiedenen grösseren Filtern (7x7, 10x0,15x15), erzielte mit dem neuen Programm jedoch keine besseren Ergebnisse.

Im Internet scheint es keinen eindeutigen Weg zur genauen Bestimmung der Filtergrößen zu geben. Nach intensiver Recherchearbeit habe ich jedoch einige Hinweise gefunden, anhand derer man die Filtergrößen einfacher bestimmen kann. Im Folgenden werden diese Erkenntnisse kurz erläutert, um dann damit die Weiterentwicklung zum zweiten Modell aufzuzeigen.

---

<sup>11</sup> Convolutional Neural Network, TensorFlow, <https://www.tensorflow.org/tutorials/images/cnn>, (10.8.2024)

### 4.3.1 Ungerade und quadratische Filtergrößen

In CNNs werden meist ungerade quadratische Filtergrößen benutzt, da die Operation eine Interpolation von den umgebenden Pixeln zum mittleren Pixel ist. Der neue Pixel wird also durch die Pixel beschrieben, welche ihn vorher umgeben haben. Mit einer geraden Filtergröße gibt es keinen mittleren Pixel. Somit kann es zu Verzerrung im entstehenden Bild kommen (Abb.12).<sup>12</sup>

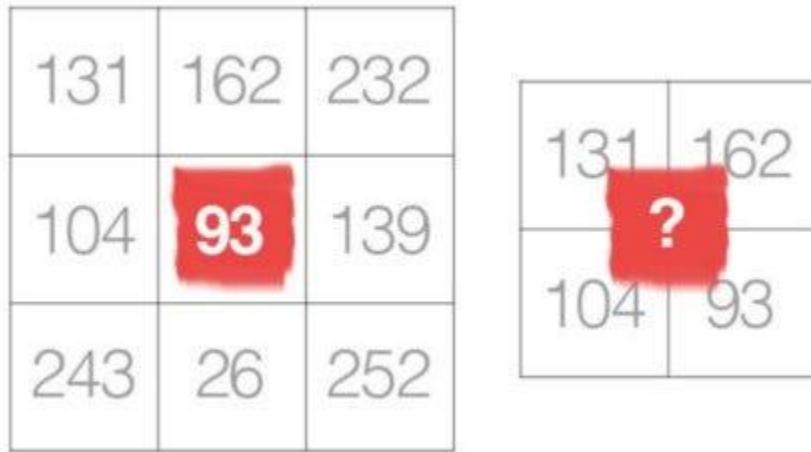


Abbildung 12: Gerade Filter haben keinen mittleren Pixel

### 4.3.2 Kleine Filter vs. grosse Filter

Kleine Filter haben den Vorteil, dass sie viel einfacher und recheneffizienter zu trainieren sind. Somit kann man sie gut schichten und für lokale Merkmalerkennung einsetzen. Grosse Filter hingegen sind schwieriger zu trainieren und rechnen ineffizienter. Dafür decken sie eine grössere Fläche ab und können so breitere Kontexte verstehen und mehr Information aufnehmen und verarbeiten.<sup>13</sup>

### 4.3.3 Filtergröße mit Gradientabstieg erlernen

Als letztes habe ich mich noch mit einem Artikel der Universität von South Carolina aus dem Jahr 2017 auseinandergesetzt<sup>14</sup>. Die Autoren der Studie stellen eine neuartige Idee vor, um die Filtergrößen zu bestimmen, welche ich in meinem Fall zwar nicht anwenden kann, aber trotzdem erwähnen möchte.

In diesem Ansatz wurden die Filtergrößen als Parameter während der Trainingsphase mitgelernt. Indem man schaute, ob der Verlust mit einem grösseren oder kleineren Filter stieg oder sank, konnte so mithilfe des Gradientabstiegs erkannt werden, ob ein Filter vergrössert oder verkleinert werden sollte.

---

<sup>12</sup> Dixit, Prashant, Why odd sized kernel preferred over even sized kernel, Medium, [https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77#:~:text=If%20we%20put%20any%20kernel\\_size,we%20use%20Odd%20kernel%20size](https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77#:~:text=If%20we%20put%20any%20kernel_size,we%20use%20Odd%20kernel%20size), (10.8.2024)

<sup>13</sup> Chhetry, Shivam Kunwar, Unveiling the Impact of Kernel Size on Convolutional Neural Network Architectures, Kaggle, <https://www.kaggle.com/discussions/general/461216>, (10.8.2024)

<sup>14</sup> Han, Shizhong et al., Optimizing Filter Size in Convolutional Neural Networks for Facial Action Unit Recognition, arxiv, <https://arxiv.org/pdf/1707.08630.pdf>, (12.8.2024)

Das Modell mit den variablen Filtergrößen wurde mit normalen Modellen mit fixen Filtergrößen verglichen. Dabei wurden die Modelle mit dem BP4D dem DISFA Datensatz gegenübergestellt. Beim ersten handelte es sich um 3D-Videos von jungen Erwachsenen mit verschiedenen Gesichtsausdrücken. Dasselbe gilt beim zweiten, nur dass es sich dort um 2D-Videos handelte. Das Ziel war es, die Emotionen der Personen zu klassifizieren, was für KI eine schwere Aufgabe ist. Im Schnitt überragte das Modell mit den trainierbaren Filtergrößen die «starren» CNN in allen Bereichen.

## 4.4 Das 2. Modell

Mithilfe der Recherche zu den Filtergrößen und weiterem Ausprobieren kam ich auf eine Zusammenstellung, welche mir die bisher mit Abstand besten Ergebnisse lieferte. Diese Zusammenstellung implementierte ich in meiner 2. Version des Programmes: In der ersten Layer benutzte ich kleinere Filter (3x3). In der zweiten dann etwas grössere (5x5) und zum Schluss in der 3. Layer noch grössere Filter (7x7).

Grosse Filter sind wesentlich schwieriger zu trainieren, während die kleinen Filter nur kleine, lokale Merkmale erkennen können (Kap.4.3.2). Ich erklärte mir den Erfolg des neuen Aufbaus so, dass die kleinen Bilder das Bild vereinfachen und die Merkmale/Zeichen sichtbarer machen (z.B. Kanten und Ecken), so dass die etwas grösseren Filter nicht mehr so grosse Schwierigkeiten haben, die wesentlichen Merkmale wie z.B. Finger und Handflächen zu erkennen.

Obwohl das Modell bisher am besten performte, war es mir für die Steuerung der Drohne immer noch zu unsicher, weshalb ich mich auf den Rat von Herrn Geissmann hin mit dem Thema «Transferlearning» und dem Toolkit MediaPipe beschäftigte.

## 4.5 Exkurs: Transferlearning

Beim Transferlearning wird ein Modell, welches auf ein verwandtes Problem trainiert worden ist, als Basis für ein neues Problem verwendet. So kann ein Modell, welches zur Erkennung von Hunden trainiert wurde, nützlich sein, um Wölfe zu erkennen. Man nimmt also einen Teil des vorherigen Modells und setzt ein paar neue Layers darauf und trainiert das Modell auf den neuen Datensatz. Somit müssen beispielsweise die Filter für das CNN nicht mehr trainiert werden, um Hundenasen oder Ohren zu erkennen, da diese fast gleich wie diejenigen der Wölfe sind. Die daraufgesetzten Layers sind für den Feinschliff zuständig.<sup>15</sup>

---

<sup>15</sup> Murel, Jacob et al. Was ist Transferlernen, IBM, <https://www.ibm.com/de-de/topics/transfer-learning#:~:text=Transferlernen%20ist%20eine%20Technik%20des%20maschinellen%20Lernens,%20bei>, (23.9.2024)

## 4.6 Finales Modell und Mediapipe

Mediapipe bietet eine Vielzahl an Bibliotheken und Tools für Machine Learning Anwendungen an. Zum Beispiel finden sich dort Programme, welche Gesichtsmerkmale erkennen und ein Koordinatennetz darüberlegen (Abb.13), was beispielsweise für die Analyse von Emotionen hilfreich ist. Ebenso stellt Mediapipe Tools zur Verfügung, die in der Lage sind, Körperposen zu erkennen, indem sie die Koordinaten von mehreren Körperpunkten angeben (Abb. 14).<sup>16</sup>

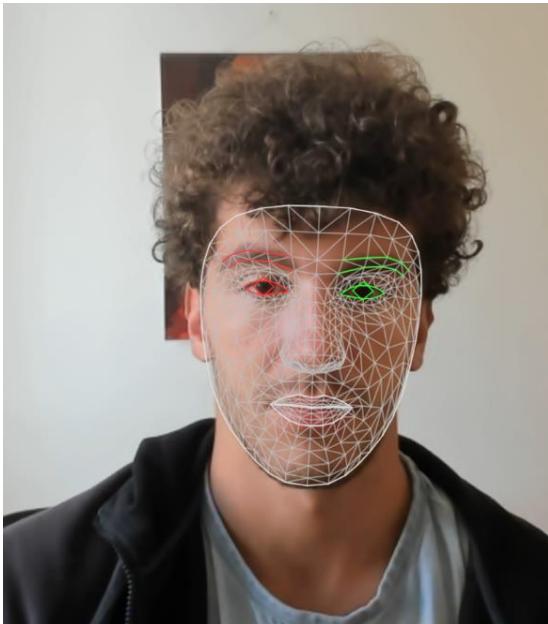


Abbildung 13: Gesichtserkennung



Abbildung 14: Körperpunkteerkennung

Solche Modelle können als Basis für ein Programm mit echter Anwendung benutzt werden. So kann aus den Körperkoordinaten einfacher klassifiziert werden, welche Bewegung ein Mensch ausführt anstelle der Analyse eines gesamten Videos. Mit der Gesichts-Punkte-Erkennung könnte beispielsweise ein Modell trainiert werden, welches Emotionen erkennt. Für mein Projekt habe ich ein Modell von Mediapipe genommen, welches in einem Foto mehrere Punkte auf der Hand erkennt und als Output dessen Koordinaten ausspuckt (Abb. 15).

---

<sup>16</sup> MediaPipe-Lösungsleitfaden, google AI for developers, <https://ai.google.dev/edge/mediapipe/solutions/guide?hl=de>, (23.9.2024)

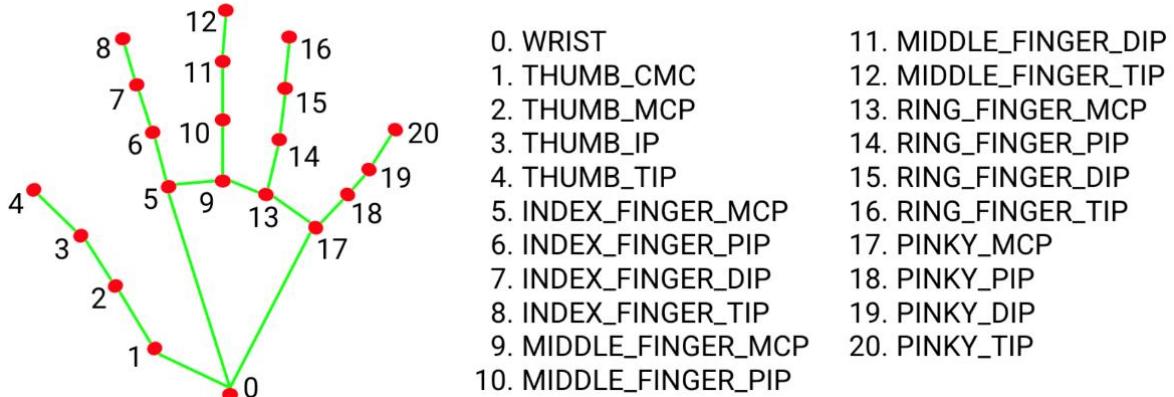


Abbildung 15: Handpunkt Markierungen

Für die 3. (finale) Version meines Modells habe ich dieses Programm als Basis genommen und einige Layers daraufgesetzt, welche mithilfe der Koordinaten das Handzeichen erkennen.

Diese Layers mussten noch trainiert werden. Für dieses Training habe ich einen Datensatz kreiert, welcher nur die Koordinaten und die entsprechenden Labels enthält. Um mehr Möglichkeiten zur Steuerung der Drohne zu haben, habe ich den Datensatz um einige Handzeichen erweitert: Nebst *fist*, *telephone*, *thumb* und *hand*, die ich für die vorherigen Modelle verwendet habe, führte ich noch die Gesten *up* (👉), *down* (👇), *left* (👈) und *right* (👉) ein. Das *Telephon*-Zeichen, wie auch das *thumbs*-Zeichen habe ich letztendlich weggelassen, da (wie später in den Resultaten zu sehen) das Programm bei dessen Erkennung die grössten Schwierigkeiten hatte.

Daraufhin erstellte ich ein einfaches Modell, welches lernt, aus diesen Koordinaten die Gesten zu erkennen. Da es sich nun nicht mehr um Bilder handelte, verwendete ich auch kein CNN mehr. Das neue Modell performte auf Anhieb sehr gut und schlug die vorherigen um Welten.

## 4.7 Die Drohne

### 4.7.1 Tello

Die Tello-Drohne ist eine Drohne der Firma Ryze Robotics. Wie die meisten Drohnen verfügt sie über vier Propeller. Unten am Rumpf hat sie eine TOF-Kamera, um die Distanz zum Boden zu messen.<sup>17</sup> Mit einer 5-Megapixel Kamera kann sie Fotos und Videos aus der Luft aufnehmen.<sup>18</sup> Sie ist mit einer App steuerbar und kann auch programmiert werden.

---

<sup>17</sup> Tello (Drohne), Wikipedia, [https://de.wikipedia.org/wiki/Tello\\_\(Drohne\)](https://de.wikipedia.org/wiki/Tello_(Drohne)), (5.10.2024)

<sup>18</sup> Gross, Richard J., DJI Ryze Tello: Ein vollständiger detaillierter Testbericht (2024), Propel, <https://www.propelrc.com/de/dji-ryze-tello/#:~:text=DJI%20Ryze%20Tello%3A%20Kamera%20Wenn%20es%20um%20die,die%20Fotos%20zeigen%20satte%20Farben%20und%20feine%20Details.>, (5.10.2024)

## 4.7.2 Das Programmieren der Tello Drohne

Die Drohne wird über WLAN mit dem steuernden Gerät verbunden und kann nach dem Implementieren der *djitelopy* Bibliothek mit intuitiven Befehlen (s. Abb. 16) gesteuert werden.

Um dies auszuprobieren habe ich ein Programm für die manuelle Steuerung der Drohne mittels Computertastatur geschrieben. Dies funktionierte einwandfrei, weshalb ich mich gleich daran machte meine KI zu implementieren.

Command	Description	Response
command	Enter command mode	OK False
takeoff	Auto takeoff	OK False
land	Auto landing	OK False
up <span style="border: 1px solid red; padding: 2px;">xx</span>	Fly upward [20, 500] cm	OK False
down <span style="border: 1px solid red; padding: 2px;">xx</span>	Fly downward [20, 500] cm	OK False
left <span style="border: 1px solid red; padding: 2px;">xx</span>	Fly left [20, 500] cm	OK False
right <span style="border: 1px solid red; padding: 2px;">xx</span>	Fly right [20, 500] cm	OK False
forward <span style="border: 1px solid red; padding: 2px;">xx</span>	Fly forward [20, 500] cm	OK False
cw <span style="border: 1px solid red; padding: 2px;">xx</span>	Rotate clockwise [1, 360] degrees	OK False
ccw <span style="border: 1px solid red; padding: 2px;">xx</span>	Rotate counter-clockwise [1, 360] degrees	OK False
flip <span style="border: 1px solid red; padding: 2px;">x</span>	Flip [l, r, f, b]	OK False
speed <span style="border: 1px solid red; padding: 2px;">xx</span>	Set speed [1, 100] cm/s	OK False
Speed?	Get current speed	<span style="border: 1px solid red; padding: 2px;">xx</span>
Battery?	Get current battery percentage	<span style="border: 1px solid red; padding: 2px;">xx</span>
Time?	Get current flight time	<span style="border: 1px solid red; padding: 2px;">xx</span>

Abbildung 16: Steuerungsbefehle für die Tello-Drohne

## 4.7.3 Die Übertragung der KI auf die Drohne

Die Übertragung der KI auf die Drohne war unkompliziert: Ich liess das Modell in einer Dauerschleife Vorhersagen über die Zeichen treffen, welche ich in die Kamera zeigte, und wies dann der Drohne mithilfe mehrerer *if-else*-Statements die richtigen Befehle zu (Abb.17). Auf dem Bildschirm meines Computers liess ich in einem Fenster das Bild der Webcam neben dem Bild der Drohnenkamera zeigen.

```
while True:
    img = camera.get_image()
    prediction = model.predict(img)
    if prediction == "Links":
        tello.move_left(10)
    if prediction == "Rechts":
        tello.move_right(10)
    if prediction == "Hoch":
        tello.move_up(10)
    if prediction == "Runter":
        tello.move_down(10)
```

Abbildung 17: Vereinfachter Codeausschnitt

#### 4.7.4 Threading

Jedoch trat mit diesem Programm ein kleines Problem auf: während die Drohne, die vom Computer erkannte Bewegung ausführte, fror das Bild der Web-Kamera ein, da der Computer den Code sequentiell (Zeile für Zeile) ausführt und nur weitergehen kann, wenn die vorherige Linie vollständig durchgeführt ist. Um dieses Problem zu beheben, recherchierte ich etwas, und kam dann auf die Idee *threading* zu benutzen. Normalerweise laufen die programmierten Programme in einem einzelnen Ausführungsstrang (*Thread*), was bedeutet, dass der Code sequentiell also Zeile für Zeile ausgeführt werden muss. Um die Bildübertragung wie auch die Bewegungsausführung gleichzeitig auszuführen, lasse ich die beiden Prozesse in separaten Threads laufen.

## 5 Resultate

In diesem Kapitel werden die Ergebnisse der verschiedenen Versionen dargestellt und verglichen. Dabei werde ich vier Modelle vergleichen: das erste Modell, eine Variante davon mit grossen Filtern, das Modell mit den angepassten Filtergrößen (2. Version) und die finale Version mit Mediapipe. Die ersten drei Modelle werde ich am überarbeiteten Datensatz mit 9000 Bildern trainieren und in Vergleich stellen. Das Mediapipe-Modell werde ich an dem Datensatz mit den Koordinaten trainieren und evaluieren. Die ersten drei Modelle haben 4 Zeichen, nur das Mediapipe-Modell wurde an 7 trainiert (vgl. Kap.4.6). Das Training wurde für alle Modelle mit 30 Iterationen durchgeführt. Der Code zu den Programmen und Diagrammen ist im Github Respository abgelegt.

### 5.1 1. Modell

#### Finale Resultate

Trainingsverlust	Trainingsgenauigkeit	Testverlust	Testgenauigkeit
0. 0130	0. 9967	0. 5845	0. 7529

Bei der ersten Version gab es das Problem mit dem Overfitting. Das Modell lernte nicht die wesentlichen Merkmale, um die Zeichen zu entziffern, stattdessen werden die Bilder «auswendig» gelernt. Die Filter schaffen es nicht, die Gesten so hervorzuheben, dass sie vom «normalen» Neuronale Netz danach erkannt werden können (Abb.18).

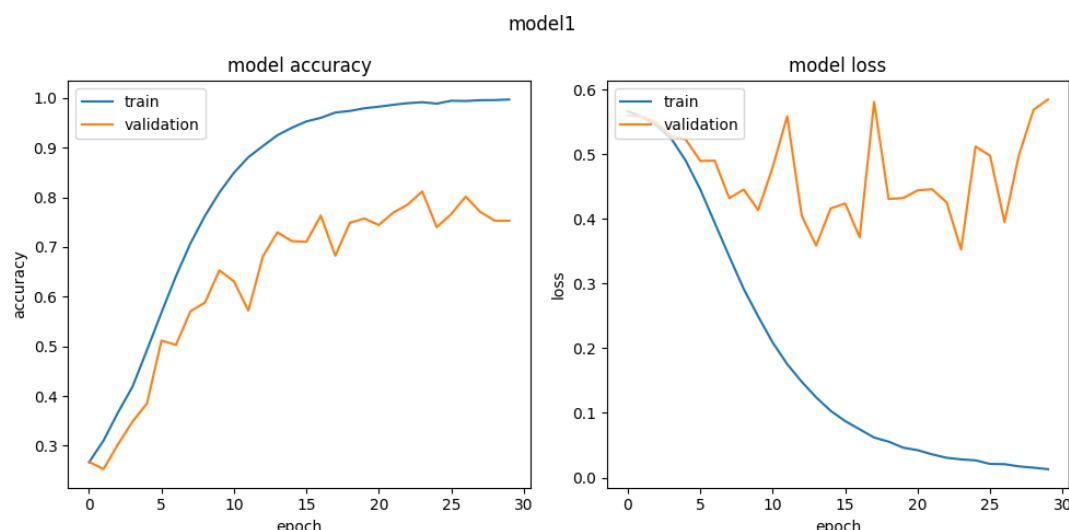


Abbildung 18: Grafik 1. Version

Die Konfusionsmatrix zeigt die Häufigkeit, mit der jede Geste korrekt oder fälschlicherweise als ein anderes Handzeichen klassifiziert wurde. Dadurch sieht man, bei welchen Zeichen das Netz Mühe hat mit der Zuordnung. Wie in der Konfusionsmatrix (Abb. 19) zu sehen ist, Lediglich mit der Klassifikation der offenen Hand (Label 3) und der geschlossenen Faust (Label 0) ist das Programm erfolgreich).

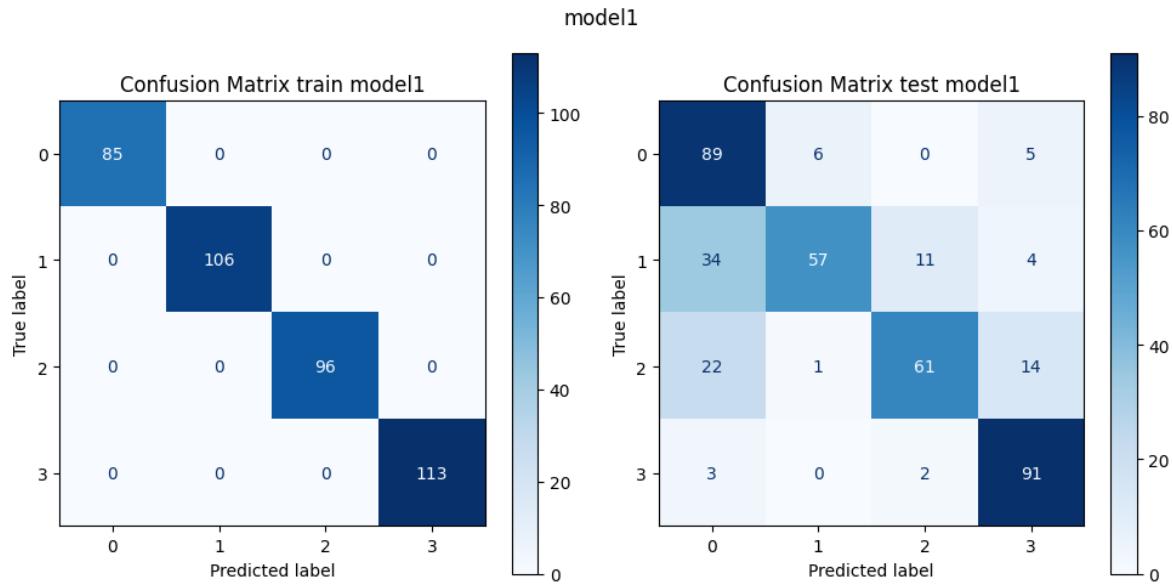


Abbildung 19: Konfusionsmatrix 1. Version

## 5.2 1. Modell mit grossen Filtern

### Finale Resultate

Trainingsverlust	Trainingsgenauigkeit	Testverlust	Testgenauigkeit
0. 5303	0. 3966	0. 5434	0. 3353

Das CNN mit den grossen Filtern brauchte deutlich länger für das Training, da es deutlich mehr Parameter enthält. Auch tut es sich schwerer die Merkmale zu erkennen (Konfusionsmatrix, Abb. 21), weshalb es nach 30 Iterationen erst bei einer Genauigkeit von knapp 40% beim Trainingsset und 34% beim Testset stand (Abb. 20).

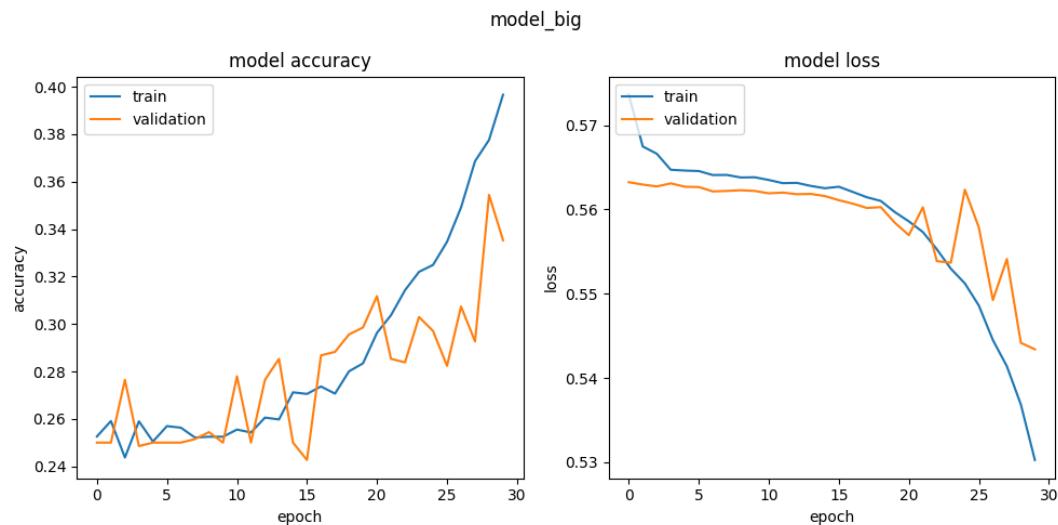


Abbildung 20: Grafik 1. Version mit grossen Filtern

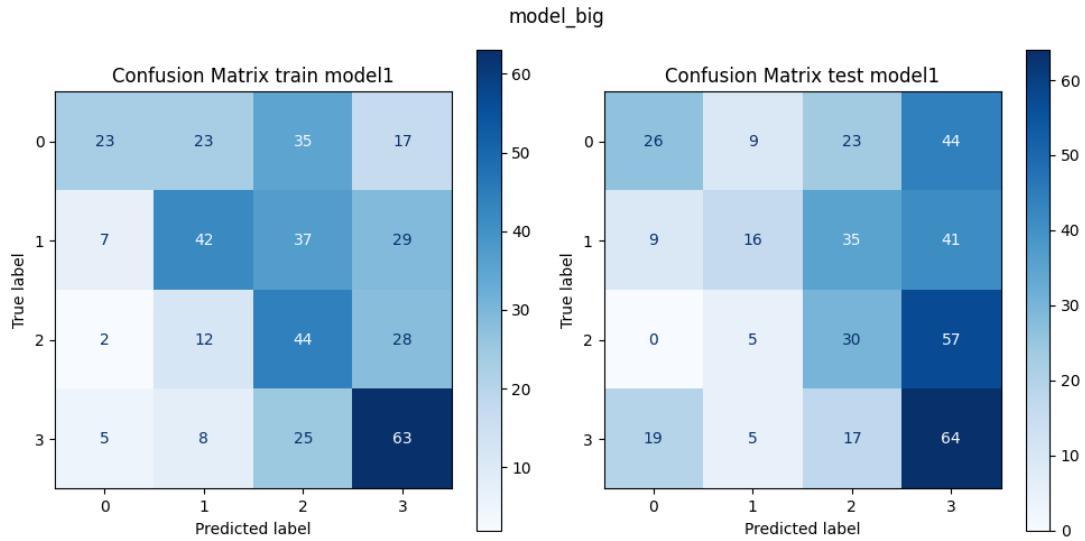


Abbildung 21: Konfusionsmatrix 1. Version mit grossen Filtern

### 5.3 2. Modell (angepasste Filter)

#### Finale Resultate

Trainingsverlust	Trainingsgenauigkeit	Testverlust	Testgenauigkeit
0. 0069	0. 9970	0. 1695	0. 9235

Das Modell mit den grösser werdenden Filtern performt von den komplett selbst gemachten am besten. Die Genauigkeits- und Verlustkurve des Trainings sehen vielversprechend aus (Abb. 22). Auch dieses Modell hat aber mit dem Overfitting zu kämpfen, da es nicht relevante Details erlernt, welche ihm beim Testset zum Verhängnis werden. Laut der Konfusionsmatrix hat das Modell die grössten Schwierigkeiten damit, die geschlossene Faust (✊) von dem Telephon-Zeichen (📞) zu unterscheiden (Abb. 23).

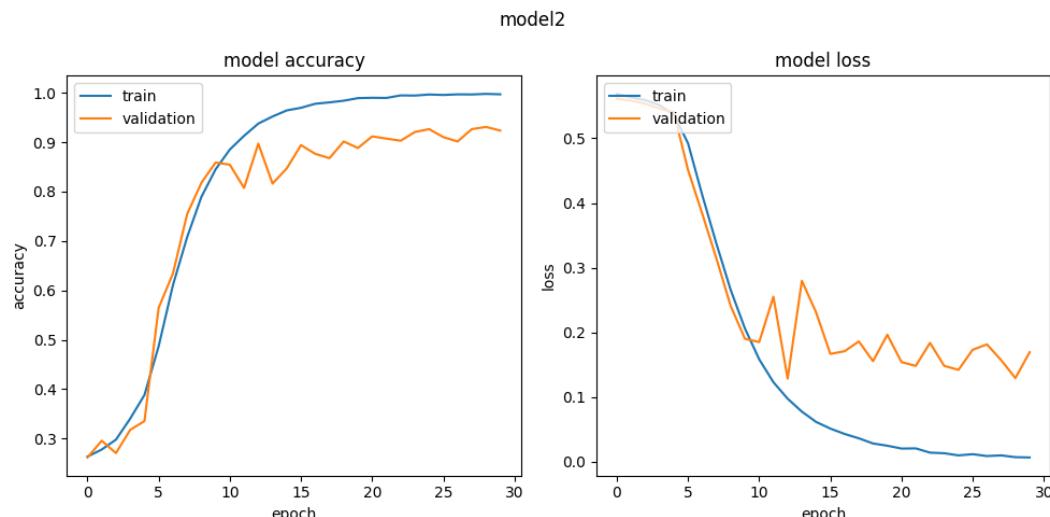


Abbildung 22: Grafik 2. Version

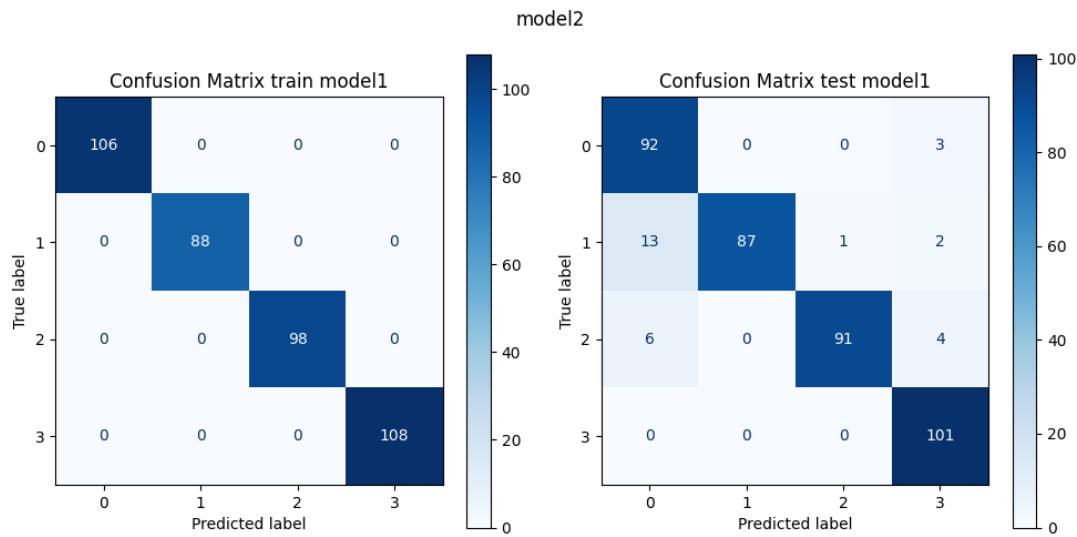


Abbildung 23: Konfusionsmatrix 2. Version

## 5.4 Finales Modell (Mediapipe-Modell)

### Finale Resultate

Trainingsverlust	Trainingsgenauigkeit	Testverlust	Testgenauigkeit
0. 0123	0. 9931	0. 0155	0. 9815

Die Genauigkeit- und Verlustkurve des Mediapipe-Modells sieht aus wie aus dem Bilderbuch. Schon nach 6 von 30 Iterationen waren Genauigkeit beim Training wie auch beim Testen schon über 90% (Abb. 24). Die Konfusionsmatrix bestätigt, dass das Modell praktisch fehlerlos ist (Abb. 25).

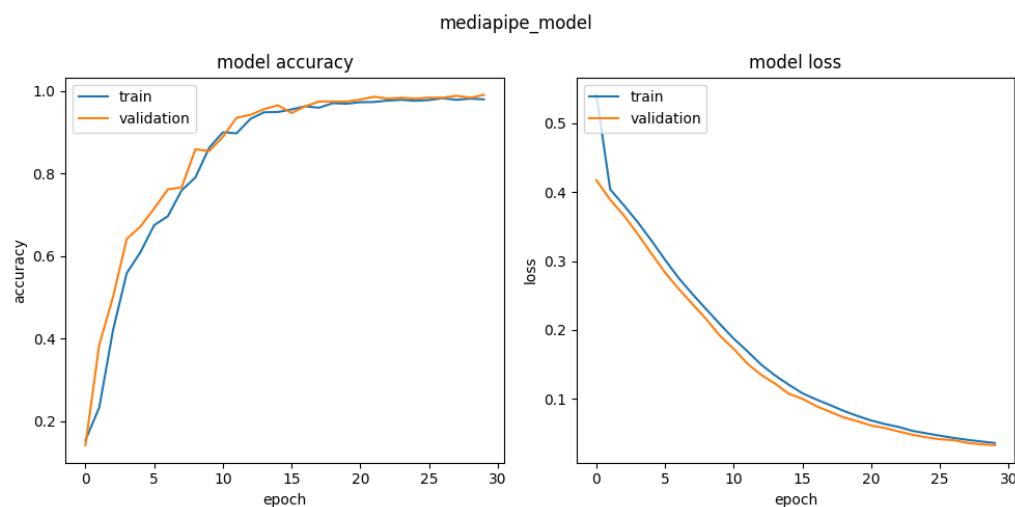


Abbildung 24: Grafik Mediapipe-Modell

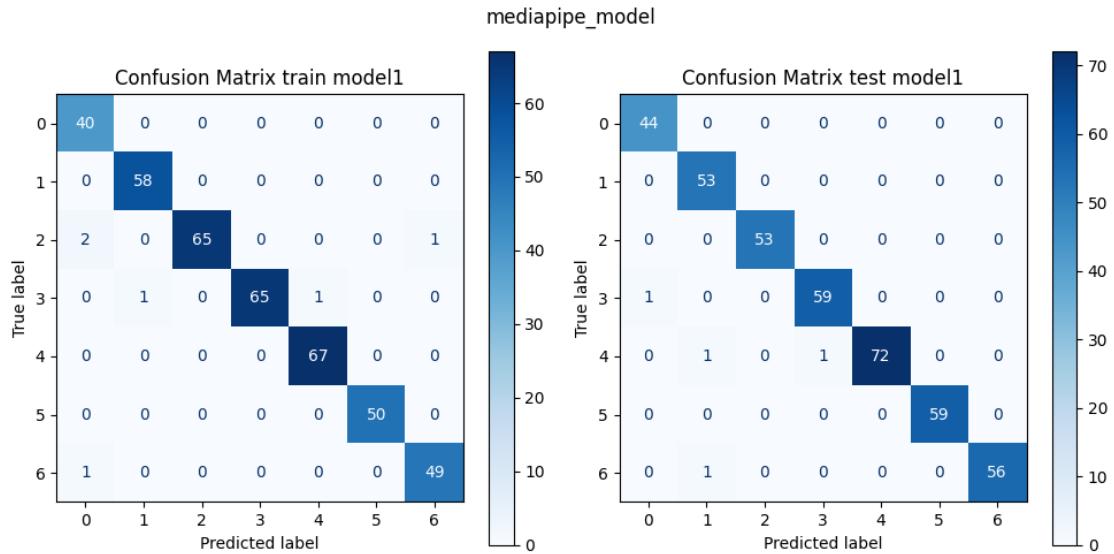


Abbildung 25: Konfusionsmatrix Mediapipe-Modell

## 5.5 Vergleich

Das erste Programm mit den kleinen Filtergrößen hatte Probleme damit, die Zeichen zu erkennen, da es zu wenige Pixel abdeckte, um nennenswerte Informationen über die Geste zu entziffern. Die grossen Filter der zweiten Version hatten Mühe damit die Zeichen zu erkennen, da sie sich schwerteten, die Konturen in den Bildern zu erfassen. Darüber hinaus waren die grossen Filter durch ihre Anzahl Parameter sehr ineffizient zu trainieren.

Das finale Modell verbindet das Beste beider Welten. So heben die kleinen Filter die Umrisse und lokale Merkmale hervor, so dass die grossen Filter diese erkennen und dank ihrer Grösse mit umgebenden Informationen in Verbindungen setzen können, um so das gesamte Zeichen zu erkennen.

Durch die Handpunkterkennung des Mediapipe-Tools fällt die ganze Erkennung der Merkmale weg. Das Netz muss nun nicht mehr die relevanten Merkmale finden, sondern bekommt diese bereitgestellt. Es wäre vergleichbar damit, wenn das Zeichen immer am gleichen Ort vor einer weissen Wand gezeigt würde. Solche Aufgaben sind für KI viel leichter.

Ein weiterer Vorteil des Mediapipe-Modells ist deshalb auch die Robustheit gegen neue Umgebungen, Personen, Lichtverhältnisse etc., da diese keinen Einfluss auf die Koordinaten, welche als Input für das Netz dienen, haben. Dies ist in meinem Fall für die Steuerung der Drohne besonders wichtig

Da der Datensatz für das Mediapipe-Modell pro Bild nun nur noch aus 63 Zahlen (3\*21 für x,y,z Koordinate) besteht, ist das Training auch viel effizienter.

## 6 Diskussion

Mir ist es gelungen selbst ein Programm zu schreiben, welches meine Handzeichen erkennt. Für das finale Modell, welches meine Drohne steuert, habe ich jedoch ein Tool von Mediapipe implementiert, um das Netz robuster und aussagekräftiger zu machen.

Es ist nun möglich die Drohne zu steuern, entweder durch das Zeigen der Gesten in die Computerkamera oder auch in die Kamera der Drohne.

Mit der Dokumentation der theoretischen Grundlagen (Kap.3) führe ich an die erste Leitfrage: «**Wie kann eine Computer Objekte auf Bildern erkennen am Beispiel von Gestenerkennung?**» heran. Die Erklärung des Gradientabstiegs legt die Basis für die meisten ML-Algorithmen. Zudem habe ich mit den CNN die gängigste Architektur für Objekterkennung und Computervision vorgestellt.

In meinem Projekt habe ich mein Ziel, welches ich in der zweiten Leitfrage formuliert habe, erreicht: «**Wie schreibt man ein Programm, welches Gesten erkennt und damit eine Drohne steuert?**». Ich habe das gesamte ML-Programm selbst geschrieben und dabei gelegentlich das Internet nach spezifischen Funktionen durchsucht sowie solche aus verschiedenen Bibliotheken implementiert. Für das Training des Modells habe ich einen eigenen Datensatz erstellt. Letztendlich habe ich erfolgreich die finale Version meines Programms auf die Drohne übertragen, um diese damit zu steuern.

### Bedeutung von Bilderkennung / Gestenerkennung

Mein Projekt gehört in die Kategorie des maschinellen Sehens. Dabei soll ein Algorithmus Bilder analysieren und Inhalte erkennen. Dies hat Anwendung in zahlreichen Bereichen, von Gesichtserkennung, über selbstfahrende Autos bis in die Medizintechnik.

Die Erkennung von Fingerzeichen könnte als eine Vorstufe zur Erkennung vom Phonetenalphabet oder gar der Gebärdensprache angesehen werden. Auch zu diesem Thema gibt es zahlreiche Datensätze im Internet. Dies könnte die Kommunikation zwischen Taubstummen und Hörenden erheblich vereinfachen.

Ausserdem kann es Robotern helfen, menschliche Gesten zu erkennen und deren Bedeutung zu verstehen. Dies könnte die Interaktion zwischen humanoiden Robotern und Menschen vereinfachen und natürlicher machen.

### Weiterentwicklung

Als naheliegende Weiterentwicklung meiner Programmierung wäre es natürlich möglich, weitere Zeichen für Kunststücke der Drohne hinzuzufügen. Zudem könnte man gewisse Routen vorprogrammieren und dann lediglich mit einer Geste ausführen.

Weiter könnte das Programm auf einen Raspberry mit einer Kamera und einem Display laufen. Das Bild auf dem Display könnte verdoppelt werden, um dann so in eine VR-Brille gepackt zu werden und die Sicht der Drohne aus einer Firstperson-view zu betrachten. Der kleine Raspberry Computer mit Batterie und Kamera könnte also vor mir aufgestellt, die VR-Brille aufgesetzt und die Drohne so von überall aus gesteuert werden. Fraglich dabei wäre allerdings, ob der Raspberry Computer in der heutigen Version genug leistungsfähig für dieses Programm ist.

## 7 Schlusswort

Mit dieser Arbeit hatte ich die Möglichkeit, mich vertieft mit dem Thema KI zu beschäftigen. Zuvor hatte ich mich nur oberflächlich und theoretisch informiert. Durch dieses Projekt musste ich mich intensiv mit dem Aufbau und der Funktionsweise von KI auseinandersetzen und das Gelernte in die Praxis umsetzen. Zudem habe ich mich mit einem mir vorher unbekannten Tool, Mediapipe, auseinander gesetzt, wie auch meine Programmierfähigkeiten in Python auf das nächste Level gebracht. Vor allem habe ich gelernt, wie man ein solches Programmierprojekt durchführt und habe durch die Arbeit mit den CNNs die Wichtigkeit von Hyperparametertuning verstanden. Ich habe das Konzept von Filtern und deren Größen verstanden und eine bessere Intuition für dessen Wahl bekommen.

Das Programmieren erfordert Problemlösungsfähigkeiten, Kreativität und Durchhaltevermögen. Trotz einiger kleiner Krisen bereitete mir die Erarbeitung des Produkts wie auch das Verfassen der Dokumentation viel Freude. Beim Programmieren bin ich oft auf Probleme gestossen, welche mich eine Ewigkeit gekostet haben. Ich musste oft stundenlang auf Internetseiten und Foren wie Stack Overflow und Reddit nach Lösungen suchen. Ich weiss, dass das zum Leben jedes Softwareengineer gehört, aber ich konnte durch dieses Projekt auch gewisse Techniken finden, wie ich diese «Bugs» schneller beheben kann.

Da ich sehr interessiert an meinem Thema gearbeitet habe, fiel es mir oft leicht, stundenlang am PC zu sitzen und an meinem Programm zu feilen. Somit war ich auch schon früh mit meiner ersten Version fertig. Die Überarbeitung dieser hat mich etwas mehr Zeit gekostet, da ich mich intensiv mit den Problemen der Filtergrößen beschäftigen musste. Schneller ging mir die Erarbeitung der finalen Version und dessen Übertragung auf die Drohne von der Hand. Dank dem ich die Arbeitsschritte während der Programmierphase laufend gut dokumentiert hatte, hatte ich keine wesentlichen Probleme mit dem Zeitmanagement für die Fertigstellung der Arbeit.

Ich hatte eine konkrete Vorstellung, in welche Richtung sich mein Projekt entwickeln sollte, weswegen mir der nächste Schritt in meiner Arbeit immer klar war. Aufgrund meines grossen Interesses und meiner leidenschaftlichen Beschäftigung mit dem Thema kamen mir ständig neue Ideen für Verbesserungen und Weiterentwicklungen in den Sinn. Einzig ein Hardwareproblem meiner Drohne machten mir gegen Schluss meiner Arbeit einen Strich durch die Rechnung. So musste ich mich mit dem Tecsup- port in Verbindung setzen, um festzustellen, dass die Drohne mir wohl einmal zu viel an die Wand geflogen war. So war die Verbindung zwischen Computer und Drohne sehr wacklig und sensibel.

In diesen Tagen erreicht uns die Nachricht, dass John Hopfield und Geoffrey Hinton mit dem Nobelpreis für Physik geehrt werden. Sie gelten mit ihren Forschungen zu Neuronalen Netzen als Pioniere und können durchaus als die Gründerväter der künstlichen Intelligenz angesehen werden.

Der Einfluss, welcher KI auf unser heutiges Leben hat, ist überwältigend, ganz geschweige von der Zukunft. Beeindruckt habe ich in dieser Arbeit festgestellt, wie zugänglich Informationen und Hilfsmittel zu Machine Learning im Internet für Forschende als auch Private sind. Somit bin ich überzeugt, dass wir in diesem Gebiet noch unglaubliche Ergebnisse erzielen werden.

# 8 Verzeichnisse

## 8.1 Literaturverzeichnis und Internetquellen

Biswal, Avijeet, The Complete Guide on Overfitting and Underfitting in Machine Learning, simplilearn, <https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting>, (5.9.2024)

Chhetry, Shivam Kunwar, Unveiling the Impact of Kernel Size on Convolutional Neural Network Architectures, Kaggle, <https://www.kaggle.com/discussions/general/461216>, (10.8.2024)

Convolutional Neural Network, TensorFlow, <https://www.tensorflow.org/tutorials/images/cnn>, (10.8.2024)

Convolutional Neural Network, Wikipedia, [https://de.wikipedia.org/wiki/Convolutional\\_Neural\\_Network#:~:text=Ein%20Convolutional%20Neural%20Network%20\(CNN%20oder%20ConvNet\),%20zu,\\_](https://de.wikipedia.org/wiki/Convolutional_Neural_Network#:~:text=Ein%20Convolutional%20Neural%20Network%20(CNN%20oder%20ConvNet),%20zu,_) (27.9.2024)

Dixit, Prashant, Why odd sized kernel preferred over even sized kernel, Medium, [https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77#:~:text=If%20we%20put%20any%20kernel\\_size,we%20use%20Odd%20kernel%20size](https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77#:~:text=If%20we%20put%20any%20kernel_size,we%20use%20Odd%20kernel%20size), (10.8.2024)

Frobenius-Skalarprodukt, Wikipedia, <https://de.wikipedia.org/wiki/Frobenius-Skalarprodukt>, (11.10.2024)

Gross, Richard J., DJI Ryze Tello: Ein vollständiger detaillierter Testbericht (2024), Propel, <https://www.propelrc.com/de/dji-ryze-tello/#:~:text=DJI%20Ryze%20Tello%3A%20Kamera%20Wenn%20es%20um%20die,die%20Fotos%20zeigen%20satte%20Farben%20und%20feine%20Details.>, (5.10.2024)

Grösse und Qualität eines Datensatzes, Machine Learning, <https://developers.google.com/machine-learning/crash-course/overfitting?hl=de>, (16.8.2024)

Han, Shizhong et al., Optimizing Filter Size in Convolutional Neural Networks for Facial Action Unit Recognition, arxiv, <https://arxiv.org/pdf/1707.08630.pdf>, (12.8.2024)

Howarth, Josh, Anzahl der Parameter in GPT-4, Exploding Topics [https://explodingtopics-com.translate.goog/blog/gpt-parameters?\\_x\\_tr\\_sl=en&\\_x\\_tr\\_tl=de&\\_x\\_tr\\_hl=de&\\_x\\_tr\\_pto=rq&\\_x\\_tr\\_hist=true#](https://explodingtopics-com.translate.goog/blog/gpt-parameters?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=de&_x_tr_pto=rq&_x_tr_hist=true#), (19.9.2024)

Kaggle, [www.kaggle.com](http://www.kaggle.com), (12.9.2024)

Künstliches neuronales Netz, Wikipedia, [https://de.wikipedia.org/wiki/K%C3%BCnstliches\\_neuronales\\_Netz](https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz), (6.9.2024)

MediaPipe-Lösungsleitfaden, google AI for developers,  
<https://ai.google.dev/edge/mediapipe/solutions/guide?hl=de>,  
(23.9.2024)

Murel, Jacob et al. Was ist Transferlernen, IBM, <https://www.ibm.com/de-de/topics/transfer-learning#:~:text=Transferlernen%20ist%20eine%20Technik%20des%20maschinellen%20Lernens,%20bei>, (23.9.2024)

Nyuytiymbiy, Kizito, Parameters and Hyperparameters in Machine Learning and Deep Learning, Medium, <https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac>, (10.10.2024)

Perrotta, Paolo: Machine Learning für Softwareentwickler. Von der Python-Codezeile zur Deep-Learning-Anwendung, Heidelberg 2020.

Tello (Drohne), Wikipedia, [https://de.wikipedia.org/wiki/Tello\\_\(Drohne\)](https://de.wikipedia.org/wiki/Tello_(Drohne)), (5.10.2024)

Was ist künstliche Intelligenz und wie wird sie genutzt?, Europäisches Parlament, <https://www.europarl.europa.eu/topics/de/article/20200827STO85804/was-ist-kunstliche-intelligenz-und-wie-wird-sie-genutzt>, (4.9.2024)

## 8.2 Abbildungsverzeichnis

**Titelblatt:** Eigene Darstellung mit Bildelementen von Thingiverse: [https://www.thingiverse.com/challenges/pcbway?utm\\_campaign=pcbway&utm\\_medium=challenge&utm\\_source=frontpage](https://www.thingiverse.com/challenges/pcbway?utm_campaign=pcbway&utm_medium=challenge&utm_source=frontpage), (26.9.2024)

**Abbildung 1:** MNIST Datensatz, <https://de.wikipedia.org/wiki/MNIST-Datenbank#/media/Datei:MnistExamples.png>, (3.9.2024)

**Abbildung 2:** Begriffserklärung: KI, ML und DL, in: AI vs ML vs DL: Complete Understanding, <https://www.pycodemates.com/2022/12/ai-vs-ml-vs-dl-understanding-differences-and-connections.html>, (22.08.2024)

**Abbildung 3:** Punktediagramm: Wohnfläche vs. Preise, eigene Darstellung (Code im Anhang), (9.10.2024)

**Abbildung 4:** Punktediagramm mit Gerade  $a = 0$ , eigene Darstellung (Code im Anhang), (9.10.2024)

**Abbildung 5:** Verlustkurve, eigene Darstellung (Code im Anhang), (9.10.2024)

**Abbildung 6:** Punktediagramm mit Gerade  $a = 1$ , eigene Darstellung (Code im Anhang), (9.10.2024)

**Abbildung 7:** Sigmoidfunktion, Sigmoidfunktion, <https://de.wikipedia.org/wiki/Sigmoidfunktion>, (5.10.2024)

**Abbildung 8:** Vergleich Under- & Overfitting, Understanding Overfitting and Underfitting in Machine Learning, <https://medium.com/analytics-vidhya/understanding-overfitting-and-underfitting-in-machine-learning-2a2f3577fb27>Medium, (5.10.2024)

**Abbildung 9:** KI Modell mit 3 Layers, Künstliches Neuronales Netz, Künstliches neuronales Netz – Wikip [https://de.wikipedia.org/wiki/K%C3%BCnstliches\\_neuronales\\_Netz](https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz)edia, (2.10.2024)

**Abbildung 10:** Funktionsweise von CNN, Intuitively Understanding Convolutions for Deep Learning, <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>, (10.10.2024)

**Abbildung 11:** Kantenerkennung durch CNN Filter, eigene Darstellung (Code im Anhang), (9.10.2024)

**Abbildung 12:** Gerade Filter haben keinen mittleren Pixel, Why is Odd sized kernel preferred over Even sized kernel?, [https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77#:~:text=If%20we%20put%20any%20kernel\\_size,we%20use%20Odd%20kernel%20size](https://medium.com/geekculture/why-is-odd-sized-kernel-preferred-over-even-sized-kernel-a767e47b1d77#:~:text=If%20we%20put%20any%20kernel_size,we%20use%20Odd%20kernel%20size), (10.10.2024)

**Abbildung 13:** Gesichtserkennung, eigene Darstellung mit Mediapipe Tool

**Abbildung 14:** Körperpunkteerkennung, eigene Darstellung mit Mediapipe Tool

**Abbildung 15:** Handpunkt Markierungen, Anleitung zur Erkennung von Handmarken, [https://ai.google.dev/edge/mediapipe/solutions/vision/hand\\_landmarker?hl=de](https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker?hl=de), (17.9.2024)

**Abbildung 16:** Steuerungsbefehle für die Tello-Drohne, Python Programming, <https://tello.oneoffcoder.com/python.html>, (22.8.2024)

**Abbildung 17:** Vereinfachter Codeausschnitt, eigene Entwicklung

**Abbildung 18:** Grafik 1. Version, eigene Entwicklung

**Abbildung 19:** Konfusionsmatrix 1. Version, eigene Entwicklung

**Abbildung 20:** Grafik 1. Version mit grossen Filtern, eigene Entwicklung

**Abbildung 21:** Konfusionsmatrix 1. Version mit grossen Filtern, eigene Entwicklung

**Abbildung 22:** Grafik 2. Version, eigene Entwicklung

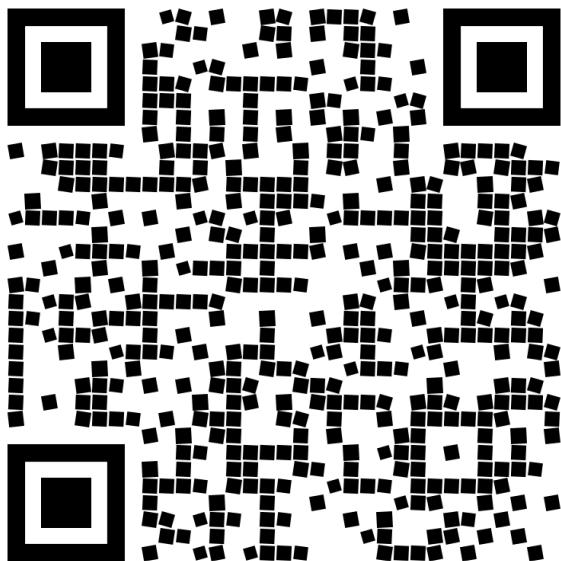
**Abbildung 23:** Konfusionsmatrix 2. Version, eigene Entwicklung

**Abbildung 24:** Grafik Mediapipe-Modell, eigene Entwicklung

**Abbildung 25:** Konfusionsmatrix Mediapipe-Modell, eigene Entwicklung

## 9 Anhang

GitHub Repository, <https://github.com/Luiszus05/MA-Luis-Zusman>, (11.10.2024)



Luis Maturaarbeit Github

## 10 Redlichkeitserklärung

Ich bestätige, dass ich die Arbeit selbständig durchgeführt, sämtliche Eigen- und Fremdleistungen deklariert und die verwendeten Quellen nach den Regeln wissenschaftlichen Arbeitens nachgewiesen habe.

Luis Zusman

Unterschrift: \_\_\_\_\_ Datum: \_\_\_\_\_