

**Dokumentation   
Computer and Robot Vision**

Handgestenerkennung mittels Bildverarbeitung und Neuronalen Netzen

im Masterstudiengang Mechatronik und Robotik an der Hochschule Heilbronn

**Patrick Bertsch**

Matrikelnummer 194894

**Peter Blinzinger**

Matrikelnummer 207603

**Felix Fabian Fees**

Matrikelnummer 207605

**Eingereicht am:** 19.01.2021

**Prüfende Personen:**

Erstprüfer:

Prof. Dr. Dieter Maier

Hochschule Heilbronn

Max-Planck-Str. 39

74081 Heilbronn

Telefon: 07131/ 504 399

**Kurzfassung**

**Sache:**   
Projektarbeit im Wintersemester 2020/2021 an der Hochschule Heilbronn im Masterstudiengang Mechatronik und Robotik im Computer and Robot Vision

**Aufgabe:**   
Erkennung von zehn verschiedenen Handgesten, welche durch Bildverarbeitung segmentiert und anschließend durch ein Neuronales Netz detektiert werden.

**Zusammenfassung:**

**Ergebnis:**

**Schlussfolgerung**

Inhalt

[I Abbildungsverzeichnis 4](#_Toc62381424)

[1 Einführung 5](#_Toc62381425)

[1.1 Aufgabenstellung 5](#_Toc62381426)

[1.2 Softwareumgebung 6](#_Toc62381427)

[1.3 Hardwareumgebung 7](#_Toc62381428)

[2 Projektablauf 8](#_Toc62381429)

[3 Architektur der Handerkennung 10](#_Toc62381430)

[4 Segmentierung 12](#_Toc62381431)

[4.1 Algorithmus 12](#_Toc62381432)

[4.1.1 Ermittlung von Superpixeln 13](#_Toc62381433)

[4.1.2 Klassifizierung der Pixelfarbe 14](#_Toc62381434)

[4.1.3 Auffälligkeitsmetrik 14](#_Toc62381435)

[4.1.4 Bayessche Inferenz 18](#_Toc62381436)

[4.1.5 Nachbearbeitung 18](#_Toc62381437)

[4.1.6 Ergebnis 19](#_Toc62381438)

[5 Neuronales Netz 21](#_Toc62381439)

[5.1 Conconutial Neutal Network (CNN) 21](#_Toc62381440)

[5.2 Architektur 22](#_Toc62381441)

[5.3 Datensatz 23](#_Toc62381442)

[5.4 Vergleich des Neuronalen Netzes auf „China Datensatz“ 24](#_Toc62381443)

[5.5 Vergleich des Neuronalen Netzes auf „Eigenen Datensatz“ 25](#_Toc62381444)

[5.6 Confusion Matrix 26](#_Toc62381445)

[6 Ausblick 27](#_Toc62381446)

[II Literaturverzeichnis 28](#_Toc62381447)

1. Abbildungsverzeichnis

[Abbildung 1: Detektion von zehn verschiedenen Handgesten 5](#_Toc62381448)

[Abbildung 2: Python PyCharm IDE 6](#_Toc62381449)

[Abbildung 3: Webcam HD 720p 7](#_Toc62381450)

[Abbildung 4: Webcam mit Stativaufbau 7](#_Toc62381451)

[Abbildung 5: Projektplan mit zugehörigen Aufgaben 10](#_Toc62381452)

[Abbildung 6: Ablaufschema der Segmentierung 10](#_Toc62381453)

[Abbildung 7: Ablauf der Verarbeitung mittels des neuronalen Netzes 11](#_Toc62381454)

[Abbildung 8: Angepasster Ablauf der Segmentierung 13](#_Toc62381455)

[Abbildung 9: Beispielhafte Unterteilung in Superpixel mittels SLIC-Algorithmus im Lαβ-Farbraum 13](#_Toc62381456)

[Abbildung 10: Binäres Klassifizierungsergebnis für die Hautfarbe 14](#_Toc62381457)

[Abbildung 11: Quantisierung des Eingangsbilds im Lαβ-Farbraum 15](#_Toc62381458)

[Abbildung 12: Einfluss von auf die Ausprägung der pixelbasierten Auffälligkeitsmetrik 15](#_Toc62381459)

[Abbildung 13: Beispielhafte Darstellung der pixelbasierten Auffälligkeitsmetrik 16](#_Toc62381460)

[Abbildung 14: Regionsbasierte Auffälligkeit 17](#_Toc62381461)

[Abbildung 15: Pixelbasierte Auffälligkeit, regionsbasierte Auffälligkeit, kombinierte Auffälligkeit 17](#_Toc62381462)

[Abbildung 16: Deutliche Störeinflüsse in der Hautfarbenwahrscheinlichkeit 17](#_Toc62381463)

[Abbildung 17: Verfeinerung der groben Maske mittels Bayesscher Inferenz 18](#_Toc62381464)

[Abbildung 18: Feine Maske (links) und Maske nach der Nachbearbeitung 19](#_Toc62381465)

[Abbildung 19: Erfolgreiche Segmentierung der Hand vor einem einfarbigen Hintergrund 19](#_Toc62381466)

[Abbildung 20: Segmentierungsergebnisse der Hand vor komplexen Hintergründen 20](#_Toc62381467)

[Abbildung 21: Grundsätzlicher Schematischer Aufbau des Neuronalen Netzes 21](#_Toc62381468)

[Abbildung 22: Darstellung der zum Pretrained Network hinzugefügten Netz-Elemente 22](#_Toc62381469)

[Abbildung 23 Auswahl der Handgesten 23](#_Toc62381470)

[Abbildung 24 Ergebnis des Neuronalen Netzes auf den China Datensatz 24](#_Toc62381471)

[Abbildung 25 Ergebnis des Neuronalen Netzes auf den Eigenen Datensatz 25](#_Toc62381472)

[Abbildung 26: Ergebnis der Confusion-Matrix 26](#_Toc62381473)

[Abbildung 27: Erkennung aller zehn Handgesten mit dem „Predicted Label" 26](#_Toc62381474)

# Einführung

Für die Vorlesung Computer und Robot Vision kurz CRV entschieden wir uns eine Handgestenerkennung mit Hilfe der klassischer Bildverarbeitung und der neuartigeren Bildverarbeitung mit Neuronalen Netzen umzusetzen. Die Idee hinter der Handgestenerkennung ist es, das für ausgewählte bzw. spezifische Handgesten eine Aktion oder Steuerung einer Computerapplikation erfolgt. Man kann sich das Prinzip ähnlich einer Touchscreen Steuerung vorstellen, wo durch verschiedene Tipp- oder Finger-Bewegungen, Aktionen auf einem Smartphone, Tablet oder Computer ausgeführt werden. Das ganze Prinzip soll für die Handgestenerkennung nur ohne die Verwendung eines Touchscreen sondern durch eine Kamera bzw. Webcam erfolgen. Anstatt der Detektion von Tipp- oder Finger-Bewegungen soll zunächst die Hand mit klassischer Bildverarbeitung und anschließend die Handgeste mit Hilfe des neuronalen Netz erkannt werden.

## Aufgabenstellung

Es sollen verschiedene Handgesten mittels klassischer Bildverarbeitung und der neuen Bildverarbeitung mittels neuronaler Netze bzw. Depp-Learning detektiert und erkannt werden. Die Erkennung der Hand bzw. Der Handgeste soll möglichst unabhängig des Bildhintergrundes und verschiedener Störfaktoren wie falsche Gegenstände etc. erfolgen. Im Anschluss daran soll exemplarisch eine Anwendung mit den ermittelten bzw. Detektion im digitalen Umfeld erfolgen.

Bei Betrachtung und Recherche verschiedener Handgesten im Internet wurde die Auswahl zunächst auf Zahlen Handgesten festgelegt bzw. begrenzt. Eine Auswahl von Zahlen Handgesten sind in Abbildung 1 auf relativ Kontrastreichen Hintergründen dargestellt. Für die Zahlen Handgesten wurden die ziefern null bis neun entsprechend dem Handzeichen zugeordnet. Es gibt dementsprechend zehn Klassen die durch das neuronale Netzwerk am Ende entsprechend interpretiert und erkannt werden müssen.



Abbildung : Detektion von zehn verschiedenen Handgesten

Die Risiken und Herausforderungen dieser Projektaufgabe betrifft einerseits die Detektion der Hand welche durch verschiedene Einflüssen wie Störobjekte, Helligkeit und Kontraast unterschiede noch erschwert werden bzw. nicht ganz so stabil Umsetzbar ist. Weitere Risiken betrifft die Genauigkeit bzw. die Performance des ausgewählten vortrainierten neuronalem Netzwerks, welches sehr stark von Vorverarbeiten Bilder abhängig sein könnte. Eine mögliche Folge könnte sein das die Trefferquote zu den eingegeben Bilder relativ schlecht Ausfällt.

## Softwareumgebung

Für das Projekt wird Python als Programmiersprache gewählt, da diese die meisten Schnittstellen zu verschiedenen Deeplearning Anwendung besitzt sowie recht gut Dokumentierte Standardbibliotheken besitzen. Ebenfalls können bislang erworbene Kenntnisse mit Python und der Vorlesung Deeplearning beim Professor Stache mit in dieses Projekt einfliegen. Als Programmierumgebung verwenden wir PyCharm, da hier die Python spezifische Syntax und Formatierung am besten unterstützt wird, Abbildung 2 zeigt die Oberfläche der Programmierumgebung Pycharm.



Abbildung : Python PyCharm IDE

Zur Versions- und Codeverwaltung des Projekts wird Git eingesetzt. Hierzu wurde kostenfrei ein Projekt bzw. Branch auf „github“ angelegt, welches ermöglicht das jeder Projektteilnehmer unabhängig voneinander und ohne große Probleme am Quellcode gemeinsam arbeiten können. Dadurch können verschiedene Programmteil zunächst einzeln von jedem entwickelt und anschließend gemeinsam zusammen gefügt werden. Der Vorteil daran ist das jeder die Änderungen im Programmcode nach verfolgen kann und bei Problemen auch Teile von Codeabschnitten auf einen vorherigen Stand zurückgeführt werden können.

Als Standardbibliotheken für das Projekt werden folgende Bibliotheken verwendet:

* Tensorflow
* Numpy
* Scipy
* Matplotlib
* OpenCV
* Keras
* Pillow
* Sonstige Zusatz Bibliotheken

## Hardwareumgebung

Für das Projekt wird die Logitech HD720p Webcamera, siehe Abbildung 3 verwendet. Über eine Stativkonstruktion, welche zentral über einer planaren Ebene angeordnet wird, siehe Abbildung 4, können verschiedene Testfälle und Umgebungen der Handgeste mit und ohne Störeinflüsse simuliert werden. Durch den Aufbau ist es außerdem möglich verschiedene Datensätze mit und ohne zusätzliche Objekte für die Vorverarbeitung der klassischen Bildverarbeitung zu erstellen, auf dieser Basis soll anschließend das vortrainierte neuronale Netzwerk Trainiert bzw. angewandt werden. Zusätzlich dient dieser Aufbau im Anschluss des Training Vorgangs zur Validation des Gesamtsystems, in dem hier eine zufällige Umgebung zunächst generiert wird und anschließend eine beliebige Handgeste aus der erstellten Auswahl siehe Abbildung 1 im Erfassungsbereich der Webcam dargestellt wird.



Abbildung : Webcam HD 720p

Für das Training des Neuronalen Netzwerks wird außerdem ein Leistungsfähiger Computer (Intel i7 7700) mit Nvidia Grafikarte (GTX 1060) und Cuda Fähigkeit verwendet. Die Cuda Fähigkeit der Grafikkarte hat den Vorteil, dass die Rechenzeit bzw. das Training des Neuronalen Netzwerk massiv reduziert wird. Somit stehen die Ergebnisse relativ schnell zum Testen und Validieren des Gesamtsystems zur Verfügung oder können gar zügig angepasst bzw. in Abhängigkeit des Gesamtsystems optimiert werden.

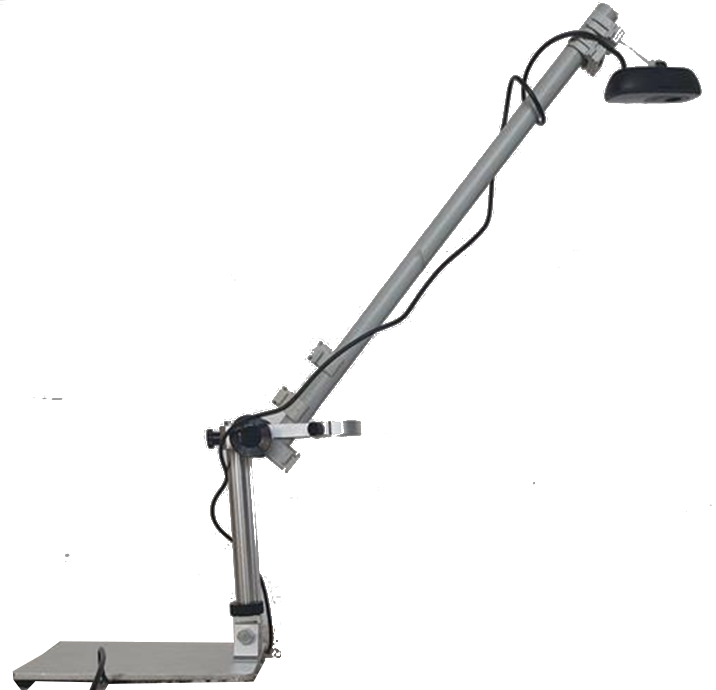


Abbildung : Webcam mit Stativaufbau

# Projektablauf

Für das Projekt Handgestenerkennung wurde zu Beginn des Projekts eine Projektplanung erstellt, siehe Abbildung 5, damit die entsprechenden anfallenden Aufgaben in Arbeitspakete gegliedert werden können und dementsprechend auch zeitlich eingeteilt bzw. strukturiert werden können. Für das Projekt wurden folgende zwölf Arbeitspakete / Aufgaben definiert:

* Datensatz erstellen und generieren
* Bilder Labeln
* Bilder und Videos aus Dateien laden
* Modelstruktur des Neuronalen Netzwerks auswählen
* Erstellung des Neuronalen Netzwerks
* Zusätzliche Test-, Trainings- und Validationsdaten erstellen
* Segmentierung
* Model auf den Testdatensatz trainieren und optimieren
* Das Gesamtsystem auf den erstellten Datensatz Trainieren und Testen
* Gesamtsystem auf unbekannte Bilder validieren
* Dokumentation erstellen

Zu jedem Arbeitspaket soll an dieser Stelle kurz beschrieben werden, welche Vorgänge der genannte Punkt beinhaltet bzw. wie die Vorgehensweise ablaufen sollte.

Datensatz erstellen und generieren

Die Erstellung eines Datensatzes dient als Grundlage des Gesamtsystems, mit diesem Datensatz soll später das neuronale Netzwerk zuerst trainiert werden. Für das erst Training des Neuronalen Netzwerks soll der Datensatz zunächst aus fremd Daten aus dem Internet nach Möglichkeit erstellt werden, damit möglichst schnell mit der Entwicklung des Gesamtsystems begonnen werden kann, im Anschluss soll der Datensatz kontinuierlich mit eigenem Bildmaterial ergänzt werden.

Bilder Labeln

Im Anschluss des Erstellens eines Datensatz aus verschiedenen Handgesten müssen die Bilder entsprechend ihrer Geste einer Beschreibung oder auch Label zugeordnet werden. Das Labeln der Bilder dient zur Einordnung der Bilder in Kategorien damit diese später vom Neuronalen Netzwerks und seines Algorithmus entsprechend der richtigen Klasse zugeordnet, trainiert und ausgewertet werden können. Dieser Vorgang ist einer der wichtigsten und muss sorgfältig erfolgen damit das Neuronale Netzwerk auch am Ende korrekt auf die Verwendeten Bilder trainiert werden kann.

Bilder und Videos aus Dateien laden

Ein weiterer Teilpunkt des Projekts ist das Einlesen von Bilddaten aus Bildern oder von einzelnen Frames aus Videos. Diese Funktionalität wird im Gesamten Projekt benötigt damit einerseits eine Vorverarbeitung von Bildern durch die klassische Bildverarbeitung erfolgen kann und andererseits zur Validation des Gesamtsystems bzw. insbesondere zur Validation des Neuronalen Netzwerks angewendet werden kann. Des Weiteren soll sich in diesem Bereich auch mit dem Einlesen von Webcam erzeugten Bilder beschäftigt werden, damit eine Live-Demonstration des Projektes am Präsentationstermin gezeigt werden kann.

Modelstruktur des Neuronalen Netzwerks auswählen

Für den Aufbau eines Neuronalen Netzwerks gibt es viele verschiedene Möglichkeiten. Für das Projekt soll möglich ein vortrainiertes Netzwerk verwendet werden, um den Trainingsprozess zu beschleunigen. In diesem Projektabschnitt sollen zunächst Informationen zu Vortrainierten neuronalen Netzen eingeholt und anschließend eines oder mehrere geeigneten Neuronalen Netzwerke ausgewählt werden, welches zur Realisierung einer Handgestenerkennung am besten geeignet ist.

Erstellung des Neuronalen Netzwerks

Im Anschluss der Auswahl eines geeigneten neuronalen Netzwerkes muss der Aufbau rund um das Neuronale Netzwerk erfolgen, damit auf den ersten erstellten Datensatz mit zugeordnetem Label eine Probetraining mit Analyse des Trainingsprozess stattfinden kann. Dies dient einerseits zur Planung von weiteren Optimierungsschritten am Neuronalen Netzwerk und andererseits zur Einstufung in wie weit die Bilddaten noch Vorverarbeitet werden müssen.

Zusätzliche Test, Trainings und Validationsdaten erstellen

Wie schon bei den Punkt „Datensatz erstellen und generieren“ angedeutet soll im Verlauf des Projekts zusätzlich eigene Daten in das Gesamtsystem einfließen bzw. gegen Ende hin ein Vergleich zwischen dem erstellten Datensatz aus dem Internet und dem eigenen Datensatz stattfinden.

Segmentierung

Je nach Performance des neuronalen Netzwerks soll im Verlauf des Projekts darüber entschieden werden, durch welche Möglichkeit die Bilder mithilfe der klassischen Bildverarbeitung Vorverarbeitet werden können damit die Aussage des neuronalen Netzwerks möglichst ohne große Problem das richtige Ergebnis zuverlässig liefert.

Model auf den Testdatensatz trainieren und optimieren

Im Anschluss soll nach Hinzunahme der Bildvorverarbeitung das System solange optimiert bzw. überarbeitet werden bis es entsprechend auf den oder die erstellten Datensätze optimal Performt.

Das Gesamtsystem auf den erstellten Datensatz Trainieren und Testen

Das Gesamtsystem soll zum Abschluss auf einem Gemeinsamen Datensatz aus eigenen Bildern trainiert werden, so dass dieses am Ende unabhängig der verwendeten Hand eine möglichst gute Vorhersage der Handgeste ausgibt.

Gesamtsystem auf unbekannte Bilder validieren

Im Anschluss zum Trainingsverlauf des Gesamtsystems sollen auf unbekannte Bilder oder gar Webcam Bilder zur Livedemonstration einer Validierung auf zufällige Handgesten stattfinden. Durch dies soll eine schlussendliche Aussage darüber getroffen werden wie die Systemgenauigkeit das Gesamtsystems ist und welche schwächen dennoch nicht Komplet reduziert werden konnten. Im Anschluss daran soll abhängig von der Handgeste eine Computerapplikation gesteuert werden.

Dokumentation erstellen

Die Dokumentation des Projekts soll die Vorgehensweise und erhaltene Ergebnisse aufzeigen, damit diese auch für zukünftige Projekte aufgegriffen bzw. nachvollzogen werden können.

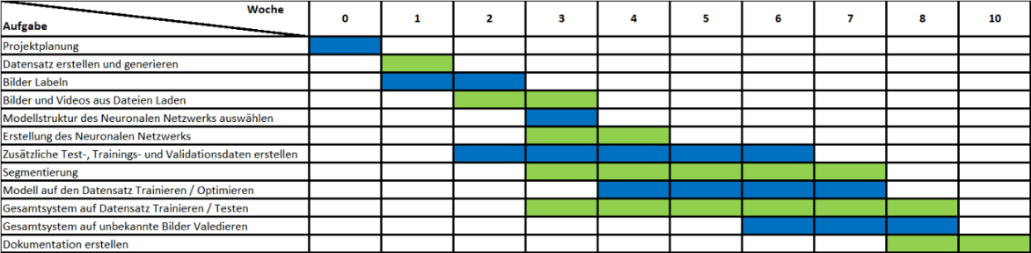


Abbildung : Projektplan mit zugehörigen Aufgaben

# Architektur der Handerkennung

Grundsätzlich lässt sich die Handgestenerkennung mittels Bildverarbeitung und Neuronalen Netzen in zwei Architekturbereiche unterteilen, der erste Teilbereich ist die Segmentierung, der zweite Teilbereich betrifft die Auswertung der Bildinformationen mittels Deep Learning. In diesem Kapitel soll zunächst ein Überblick über diese gewählte Architektur gegeben werden. In den Nachfolgenden Kapiteln 4 und 5 wird Fortlaufend tiefer auf die genaue Umsetzung und das Prinzip der Teilbereiche Segmentierung mittels klassischer Bildverarbeitung und Deep Learning eingegangen.

Bei der Segmentierung, siehe Abbildung 6, werden Bilddaten mittels klassischer Bildverarbeitung soweit Aufbereitet, dass diese möglichst einfach mittels Neuronalen Netzwerk ausgewertet bzw. interpretiert werden können. Die Interpretation von Bilddaten ist am besten wenn im zugeordneten Bild möglichst geringe Stör- bzw. Randinformation im Bildhintergrund vorhanden sind. Für die Handgestenerkennung bedeutet dies, dass möglichst nur die Hand dem Neuronalen Netzwerk übergeben werden soll, um die Geste im Anschluss richtig auswerten und Interpretieren zu können. Dazu wurde die Segmentierung in folgende drei Grundbereiche Unterteilt:

* Bild, Videos oder Webcam Daten einlesen
* Bilddaten Vorverarbeiten
* Anwendung des SLIC, Hautfarben und Saliency-Algorithmus

Im ersten Teilbereich der Segmentierung geht es hauptsächlich zunächst, um das Einlesen von Bilddaten aus vorhanden Bilddateien, Videodateien oder einer Liveaufnahme mittels Webcam. Im Anschluss daran sollen diese Bilddaten über eine Vorverarbeitung aufgewertet werden, so dass ein möglichst großer Kontrastbereich, Helligkeitsbereich oder eine etwaige andere Korrektur des Bildes stattfindet. Im Anschluss daran soll die Hand aus den Bildinformationen detektiert werden, möglich wird dies durch die Erkennung der Hautfarbe aus den Bildinformationen. Zum Einsatz soll hier ein SLIC, Hautfarben und Saliency-Algorithmus, siehe hierzu Kapitel 4.

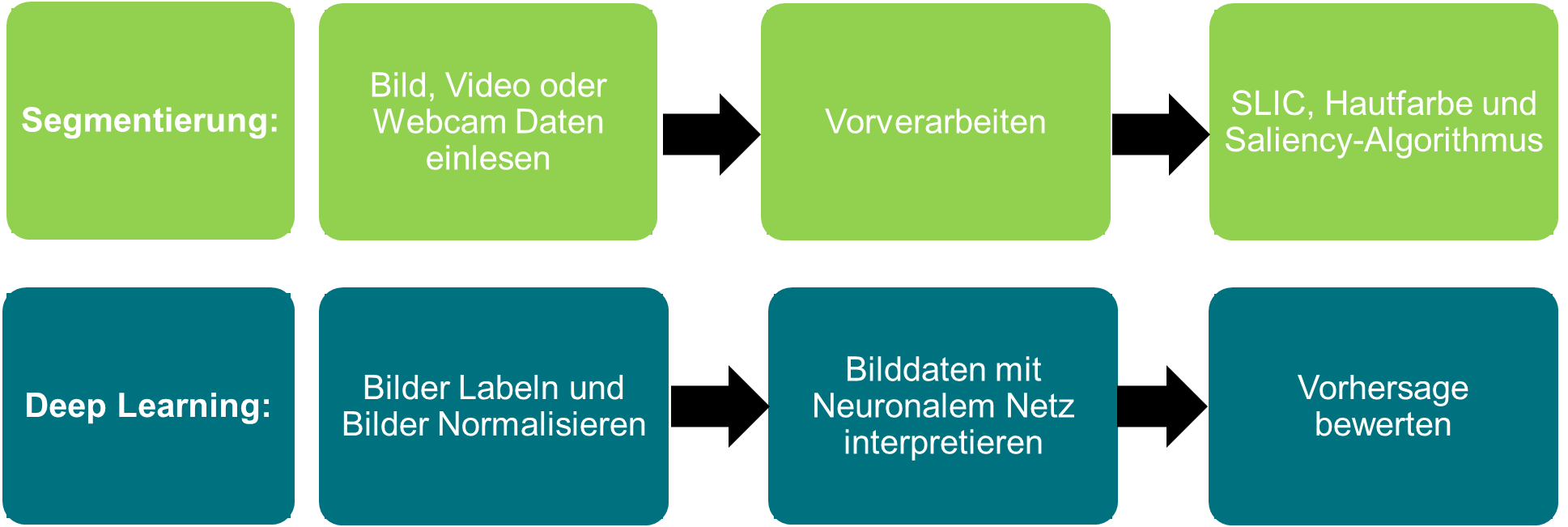


Abbildung : Ablaufschema der Segmentierung

Der zweite Teilbereich der Handgestenerkennung ist der Einsatz eines neuronalen Netzes basierend auf Deep Learning, siehe Abbildung 7. Der Einsatz von Deep Learning hat den großen Nutzen das Bildinformationen direkt mit einer Beschreibung bzw. externen Information (Label) verknüpft werden können, dies vereinfach Schlussendlich die Interpretation von unbekannten Bildern. Für den Teilbereich Deep Learning wurden folgende drei Zwischenschritte definiert:

* Bilder Labeln und Normalisieren
* Bilddaten mit Neuronalem Netz interpretieren
* Vorhersage bewerten

Der wichtigste Vorgang beim Einsatz von Deep Learning ist es die Bilder zu Labeln, hier werden einige Beschreibende Informationen bzw. Bildmerkmale dem entsprechenden Bild zugeordnet. Diese dienen im Anschluss einerseits zum Training des neuronalen Netzwerks und zur Vorhersage Bewertung des Trainingsverlaufs. Die Normalisierung des Bildes oder der Bilder ist besonders entscheidend damit auf gleicher Bezugsbasis Informationen zum Bild interpretiert werden können und somit eine Verallgemeinerung stattfindet. Im Anschluss muss das neuronale Netz bzw. alle Neuronen auf die erstellten Bildinformationen, in unterschiedliche Sichtweisen des Bildes trainiert werden. Im Verlauf des Trainingsprozess sollte sich dann allmählich eine Verallgemeinerung dieser Informationen abbilden, welche es dann schlussendlich ermöglichen sollte eine Vorhersage zu einem unbekannten Bild zu ergeben. Die Bewertung dieser Vorhersage ist entscheiden, da diese Aussagen über die Treffer Wahrscheinlichkeit bzw. die Richtigkeit der Systemantwort wieder gibt. Eine detailliertere Beschreibung zum Aufbau des Datensatzes und des neuronalen Netzwerks wird in Kapitel 5 behandelt.

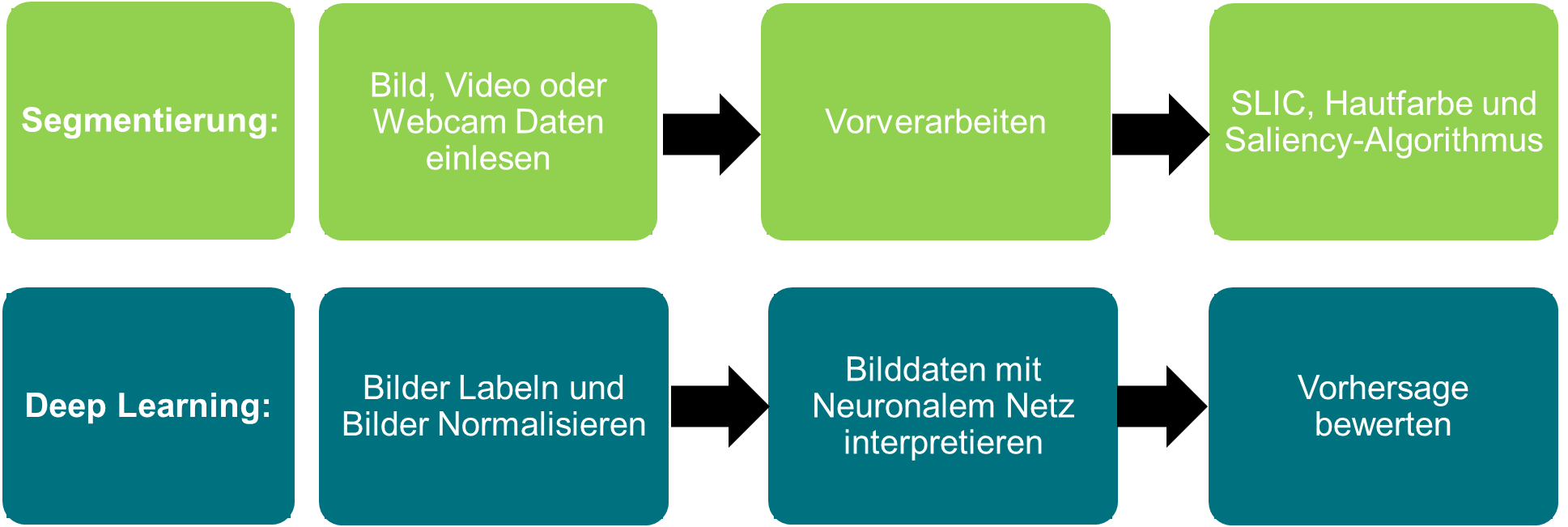


Abbildung : Ablauf der Verarbeitung mittels des neuronalen Netzes

# Segmentierung

Vor der Interpretierung der Handgesten wird zunächst eine Segmentierung des Eingabebilds mittels Methoden der klassischen Bildverarbeitung durchgeführt. Das Ziel der Segmentierung ist die Reduzierung der Hintergrundkomplexität und damit die Vereinfachung der Gesteninterpretation.

## Algorithmus

Die Detektion von Händen mittels klassischer Bildverarbeitung wird durch die Vielzahl an möglichen Gesten und Hauttönen erschwert. Allgemeine Probleme bei der Segmentierung sind unterschiedliche Lichtverhältnisse und teilweise Verdeckung des zu segmentierenden Objekts. Bei der Segmentierung wird daher der Fokus auf einen generellen Farbbereich für Hauttöne und eine Auffälligkeitsmetrik gelegt. Auf klassische kanten- und konturbasierte Methoden wird daher verzichtet, da eine allgemeine Beschreibung einer Handkontur schlecht möglich ist. Im Rahmen der Arbeit werden verschiedene Gesten untersucht, sodass z.B. die Anzahl der sichtbaren Finger kein sicheres Kriterium darstellt.

Der gewählte Segmentierungsalgorithmus orientiert sich an [1] und verwendet ein Farbmodell aus [2]. Die in den Artikeln beschriebenen Abläufe wurden implementiert und um morphologische Filter erweitert. Die Einstellparameter wurden durch eigene Tests und unabhängig von den Artikeln bestimmt. Eine Übersicht des implementierten Algorithmus ist in Abbildung 8 dargestellt. Man kann die folgenden Schritte einteilen:

1. Quantisierung des Bilds in Stufen je Farbkanal im -Farbraum
2. Vorsegmentierung des Bilds in Superpixel mittels Simple Linear Iterative Clustering (SLIC)
3. Berechnung von Kenngrößen für jede Region (geometrisch, Farbverteilungen)
4. Bestimmung einer Farbmaske für die Hautfarbe bestehend aus Termen im RGB-Raum
5. Berechnung einer Auffälligkeitsmetrik mithilfe der Kenngrößen und der Farbmaske
6. Binarisierung des Auffälligkeitsbilds mit der Otsu-Methode
7. Verfeinerung der Auffälligkeit durch Bayessche Inferenz mithilfe der Farbhistogramme und der vorhergehenden Maske
8. Binarisierung der verfeinerten Auffälligkeit mit der Otsu-Methode
9. Korrektur der erhaltenen Maske mit morphologischen Filtern
10. Segmentierung des Eingabebilds mithilfe der finalen Maske

Im Folgenden wird auf die einzelnen Schritte im Detail eingegangen. Der SLIC-Algorithmus, die Hautfarbenmaske und die Auffälligkeitsmetrik stellen die zentralen Punkte dar.

**

Abbildung : Angepasster Ablauf der Segmentierung

, Verwendung von Superpixeln,  
einer Auffälligkeitsmatrix und eines Farbfilters für die Hautfarbe

### Ermittlung von Superpixeln

Das Eingabebild wird zunächst mithilfe des SLIC-Algorithmus in Superpixel unterteilt. Das Verfahren führt ein k-means-Clustering unter Berücksichtigung aller Kanäle Pixelfarbe und der Pixelposition durch. Die verwendete Implementierung [3] verwendet dazu den -Farbraum und die Vorgabe der Anzahl der gewünschten Superpixel. Eine höhere Anzahl an Superpixeln kann die Segmentierung einzelner Bildobjekte fördern, allerdings erhöht sie auch die Laufzeit des SLIC-Algorithmus und der nachfolgenden Schritte. Der SLIC-Algorithmus liefert eine Matrix aus Labels zurück, welche für jeden Pixel den zugehörigen Superpixel benennen. Die Labels können verwendet werden, um die Superpixel graphisch darzustellen (Abbildung 9). Der Algorithmus erzielt das gewünschte Ergebnis.

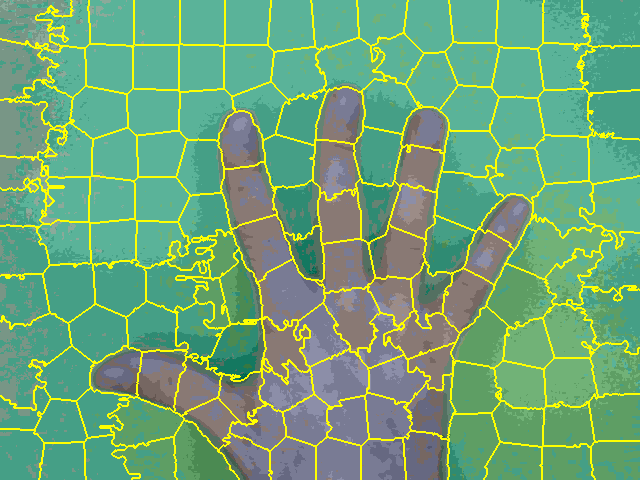


Abbildung : Beispielhafte Unterteilung in Superpixel mittels SLIC-Algorithmus im Lαβ-Farbraum

.  
links: Eigener Datensatz, rechts: Fremddatensatz

### Klassifizierung der Pixelfarbe

Die Klassifizierung der Pixel in die beiden Klassen „Hautfarbe“ und „keine Hautfarbe“ erfolgt entgegen [1] basierend auf einfachen Termen im RGB-Raum aus [2]. Die Autoren ermittelten die beste Kombination der möglichen Terme durch Einsatz von Machine Learning. Sowohl die Anzahl der Terme als auch deren Komplexität wurden im Artikel begrenzt. Die Grenzwerte wurden geringfügig für den eigenen Datensatz angepasst. Die aus dem Artikel ausgewählten Terme lauten:

|  |  |
| --- | --- |
|  | (.) |
|  | (.) |
|  | (.) |

Aus den Termen wird durch Vergleich mit den Grenzwerten und anschließende logische Verkettung die folgende Entscheidungsvorschrift gebildet:

|  |  |
| --- | --- |
|  | (.) |

Man erhält als Ergebnis eine erste Maske für das Bild, welches allerdings Objekte mit ähnlicher Farbverteilung (

Abbildung 10) enthält. Da die Grenzwerte jedoch nicht direkt auf die einzelnen Farbkanäle angewandt werden, erhält man eine deutlich genauere Abgrenzung des relevanten Farbraums.

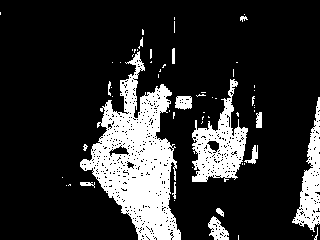


Abbildung 10: Binäres Klassifizierungsergebnis für die Hautfarbe

,  
links: Eigener Datensatz, rechts: Fremddatensatz

### Auffälligkeitsmetrik

Für die Segmentierung des Bildvordergrunds werden verschiedene Auffälligkeitsmetriken eingesetzt. Gemäß [1] wurden eine pixelbasierte und eine regionsbasierte Metrik eingeführt und kombiniert. Im Rahmen der Arbeit wurden Ergebnisse ausschließlich mit der pixelbasierten Metrik untersucht, da damit stabilere Ergebnisse erzielt werden konnten. Die regionsbasierte Metrik enthält aufgrund der verwendeten binären Zuweisung der Hautfarbe für einzelne Pixel überwiegend homogene Zonen, sodass der Informationsgewinn eher gering ist.

Alle Farbkanäle des Eingabebilds im Lαβ-Farbraum werden zunächst in jeweils Stufen quantisiert. Ein beispielhaftes Ergebnis mit vierzehn Quantisierungsstufen ist in Abbildung 11 dargestellt. Es können also verschiedene Farben abgebildet werden. Die Quantisierung verringert den Berechnungsaufwand der pixelbasierten Metrik erheblich.



Abbildung : Quantisierung des Eingangsbilds im Lαβ-Farbraum

, in 14 Stufen je Farbkanal

Die pixelbasierte Auffälligkeitsmetrik basiert auf der Entropie der Farbe , welche um eine Gewichtung der Abstände einzelner Regionen zueinander erweitert wurde. Gemäß [1] wird die Kenngröße wie folgt berechnet:

|  |  |
| --- | --- |
|  | (.) |

Mit wird der Abstand der Schwerpunkte der Superpixel und bezeichnet. Außerdem beschreibt bzw. die relative Häufigkeit der Farbe in der Region bzw. . Für jede Farbe muss über alle Regionen iteriert werden. Es werden dann die relativen Häufigkeiten sowie der Abstand zu allen anderen Regionen und deren Häufigkeiten berücksichtigt. Die Entropie ist folglich umso größer je, gleichmäßiger die Farbverteilung in den Regionen ist und je weiter die Regionen voneinander entfernt sind. Die Entropie wird mit der Hautfarbenwahrscheinlichkeit wie folgt kombiniert:

|  |  |
| --- | --- |
|  | (.) |

Eine größere Entropie resultiert also in einer kleineren Auffälligkeit . In [1] wird dies mit der Annahme begründet, dass Hintergrundfarben breiter und ausgeglichener über das Bild verteilt sind. Durch Zuweisung des Ergebnisses für jeden Pixel der Farbe erhält man die pixelbasierte Auffälligkeit . Die Verteilung lässt sich durch Veränderung von anpassen (Abbildung 12). In Ergänzung zu [1] wird zunächst auf das Intervall [0;1] normalisiert, sodass der Einfluss von vergleichbar bleibt.



Abbildung : Einfluss von auf die Ausprägung der pixelbasierten Auffälligkeitsmetrik

Im Rahmen dieser Arbeit wurde die im folgenden Abschnitt erläuterte regionsbasierte Auffälligkeit nicht weiter berücksichtigt, sodass eine erste grobe Maske lediglich aus der pixelbasierten Auffälligkeit abgeleitet wird. Die berechnete Auffälligkeit wird auf das Intervall normalisiert und mittels eines quadratischen Gaußfilters mit einer Kantenlänge von fünf Pixeln verarbeitet. Das Ergebnis wird dann mithilfe der Otsu-Methode binarisiert (Abbildung 13). Der Algorithmus liefert schlechte Ergebnisse, wenn das gewählte ein zu dunkles Bild liefert. Aufgrund der binären Hautfarbenwahrscheinlichkeit verschwindet die Auffälligkeit für Pixel mit . Der homogene Hintergrundbereich sorgt dann dafür, dass Regionen mit geringem ebenfalls dem Hintergrund zugeordnet werden. Dadurch entstehen Löcher in der Maske, welche bei entsprechender Größe durch nachfolgende Schritte nicht mehr korrigiert werden können.



Abbildung : Beispielhafte Darstellung der pixelbasierten Auffälligkeitsmetrik

links: Graustufen, rechts: Binarisiert mithilfe der Otsu-Methode

In [1] wird nun eine zweite Metrik verwendet, welche möglichst zusammenhängende Regionen mit hoher Hautfarbenwahrscheinlichkeit hervorheben soll. Für die regionsbasierte Metrik werden ebenfalls die Hautfarbenwahrscheinlichkeit und Informationen der Superpixel verwendet. Allerdings erhalten nun Pixel desselben Superpixels und nicht desselben Farbwerts den gleichen Auffälligkeitswert. Die regionsbasierte Metrik wird gemäß [1] definiert als:

|  |  |
| --- | --- |
|  | (.) |

Mit wird erneut der Abstand der jeweiligen Superpixelschwerpunkte zueinander bezeichnet. Der Einfluss des Abstands kann durch beeinflusst werden und wird standardmäßig auf 0.7 gesetzt. Eine Normalisierung des Abstands bezüglich der Bilddimensionen ist denkbar, wurde allerdings nicht implementiert. In [1] wird als Distanz der Hautfarbenwahrscheinlichkeiten der Regionen und bezeichnet und wie folgt definiert:

|  |  |
| --- | --- |
|  | (.) |

Dabei entspricht der Anzahl der Pixel im Superpixel der Region bzw. dessen Fläche und der Hautfarbenwahrscheinlichkeit des Pixels in Region . Gemäß unserer Implementierung nimmt die Werte an. Für Superpixel mit gleicher Hautfarbenwahrscheinlichkeit verschwindet also und damit . Aufgrund der Normalisierung durch Division der Pixelanzahl im Superpixel ist unabhängig von der Größe der Superpixel. Die regionsbasierte Auffälligkeit wird auf [0;255] normalisiert (Abbildung 14). Man kann die einzelnen Superpixel noch gut im Bild von erkennen.



Abbildung : Regionsbasierte Auffälligkeit

,  
Eigener Datensatz, Hintergrund fast vollständig grün,

In [1] bestimmen die Autoren anschließend eine gemeinsame Auffälligkeitsmetrik aus und:

|  |  |
| --- | --- |
|  | (.) |

Bei der Implementierung muss berücksichtigt werden, dass und zwar 8-bit-Werte enthalten können, für das Multiplikationsergebnis allerdings 16-bit verwendet werden müssen. Der Exponentialterm implementiert eine Gewichtung von Pixeln in der Bildmitte [1]. Es wird also angenommen, dass sich die Hand in der Bildmitte befindet. Die Entfernung eines Pixels zur Bildmitte wird normalisiert durch die Hälfte der kleineren Seitenlänge des Bilds (Bildbreite , Bildhöhe ). Durch die Multiplikation werden fehlerhafte schwarze Pixel aus beiden Bildern übernommen (Abbildung 15, Abbildung 16).



Abbildung : Pixelbasierte Auffälligkeit, regionsbasierte Auffälligkeit, kombinierte Auffälligkeit



Abbildung : Deutliche Störeinflüsse in der Hautfarbenwahrscheinlichkeit

, diese werden nicht immer beseitigt

In unserer Implementierung setzen wir mit ähnlichen Ergebnissen:

|  |  |
| --- | --- |
|  | (.) |

### Bayessche Inferenz

Aus dem vorherigen Schritt ist eine erste grobe Maske für die Segmentierung in Hand/Vordergrund und Hintergrund (BG) bekannt. In [1] wird als A-priori-Wahrscheinlichkeit , dass ein Pixel zur Hand gehört, angenommen.

|  |  |
| --- | --- |
|  | (.) |

Die Auffälligkeitsmetrik wird entsprechend ihrer neuen Interpretation auf [0;1] normalisiert. Die beiden Größen und beschreiben die Plausibilität eines Pixels zum Vordergrund bzw. zum Hintergrund zu gehören. Die Größen entsprechen dabei der relativen Häufigkeit der Farbkombination ( des Pixels in den jeweiligen Regionen.

|  |  |
| --- | --- |
|  | (.) |

|  |  |
| --- | --- |
|  | (.) |

Durch Bayessche Inferenz bestimmen die Autoren in [1] dann die A-posteriori-Wahrscheinlichkeit eines Pixels zur Hand zu gehören:

|  |  |
| --- | --- |
|  | (.) |

In unseren Versuchen konnten wir eine Beseitigung von Störungen durch diese Methode beobachten (Abbildung 17). Die A-posteriori-Wahrscheinlichkeit wird durch Binarisierung mithilfe der Otsu-Methode in die finale, feinere Maske umgewandelt.

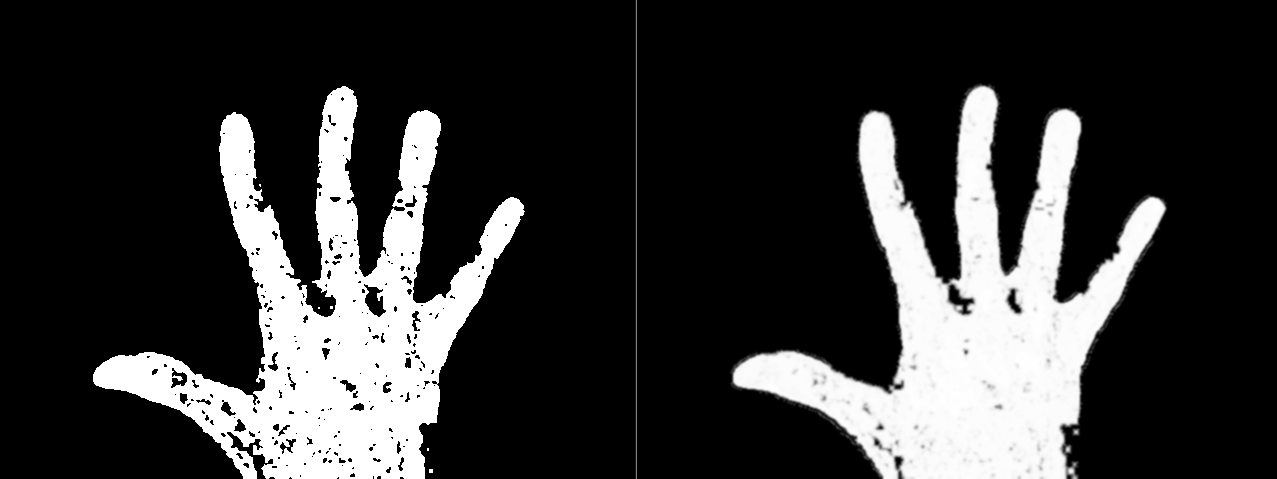


Abbildung : Verfeinerung der groben Maske mittels Bayesscher Inferenz

(eigener Datensatz)

### Nachbearbeitung

Als letzter Verarbeitungsschritt wird auf die Maske zunächst ein quadratisches, öffnendes Filter der Kantenlänge sieben angewandt, um kleine, freistehende Elemente der Maske zu entfernen. Anschließend wird ein quadratisches, schließendes Filter der Kantenlänge neun verwendet, um kleinere Löcher in den großen, zusammenhängenden Elementen zu schließen. Störungen in der Maske aus vorhergehenden Schritten werden dadurch weitgehend unterdrückt, sodass zusammenhängende Bildbereiche segmentiert werden können. Die Nachbearbeitung einer Maske ist beispielhaft in Abbildung 18 dargestellt.

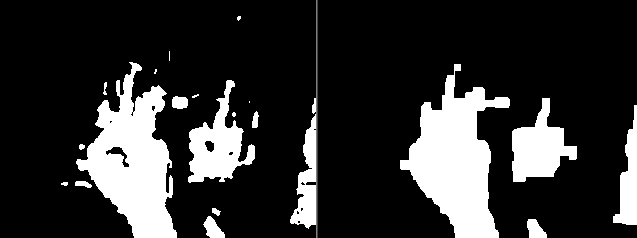


Abbildung : Feine Maske (links) und Maske nach der Nachbearbeitung

, mittels morphologischer Filter

### Ergebnis

Bei der Segmentierung von Bildern aus dem Fremddatensatz (Abbildung 20) konnte festgestellt werden, dass die Implementierung in den meisten Fällen die Hand erfolgreich dem Bildvordergrund zuordnet. Allerdings ist die Klassifizierung der Hautfarbe nicht ausreichend robust bei besonders schlechten Beleuchtungsverhältnissen, sodass Teile der Hand dem Hintergrund zugeordnet werden können. Eine Interpretation der Handgeste durch das Neuronale Netz liefert dann ein falsches Ergebnis. Eine Fehlererkennung für diesen Fall wurde nicht implementiert. Des Weiteren werden Objekte mit einer ähnlichen Farbverteilung (z.B. die rote Sitzfläche in der dritten Reihe in Abbildung 20) mit in den Vordergrund übernommen. Die Segmentierung erzeugt in den meisten Fällen keine klaren Umrisse der Hand. Dennoch wird ein Großteil von Störobjekten im Hintergrund durch die Segmentierung entfernt. Die Verwendung eines binären Farbmodells anstelle einer Wahrscheinlichkeitsverteilung verursacht mehrere Nachteile. Zunächst wird die regionale Auffälligkeitsmetrik nicht nutzbar, sodass sich die gesamte Auffälligkeitsmetrik auf die Pixelebene beschränkt, was sich nachteilig auf die Robustheit auswirkt. Außerdem ist die Verbesserung der Maske mithilfe Bayesscher Inferenz eingeschränkt. Die vielversprechende Segmentierungsqualität aus [1] kann daher nicht ganz erreicht werden.



Abbildung : Erfolgreiche Segmentierung der Hand vor einem einfarbigen Hintergrund

, (eigener Datensatz)

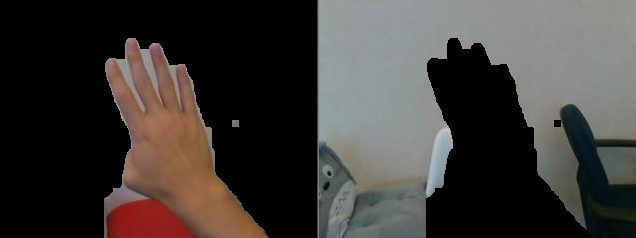
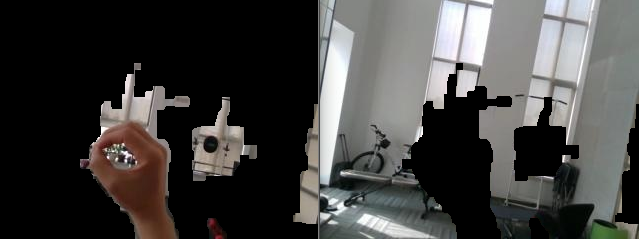


Abbildung : Segmentierungsergebnisse der Hand vor komplexen Hintergründen

(Fremddatensatz)

# Neuronales Netz

Für die Detektion der einzelnen Handgesten aus den Segmentierten Bildern wird ein Neutonales Netz verwendet. Durch die Anwendung von Transferlearning kann das Netz auf vortrainierten Strukturen aufgebaut werden. Dies hat den Vorteil, dass die Rechenzeit des Lernzyklus drastisch reduziert und Ergebnisse so schneller erzielt werden können.

## Conconutial Neutal Network (CNN)

Ein Convolutional Neural Network (CNN), ist eine Art von neuronalen Netzen. Die Idee besteht darin, inspiriert von den biologischen Prozessen in Gehirnen, Daten zu interpretieren und auszuwerten. Es ist ein Untergebiet des maschinellen Lernens. CNN- Architekturen werden vornehmlich im Zusammenhang mit maschinellen lernverfahren im Bereich von Bild- oder Audiodateien angewendet.

Die Struktur eines klassischen CNN besteht aus einer Kombination von einem oder mehreren sogenannten Convolutional Layern, immer gefolgt von einem Pooling Layer. Durch die Kombination dieser Paarung, in Verbindung mit weiteren Verfahren (z.B. Dropout, Flatten, …) entsteht so durch die Verschachtelung und Verbindung des einzelnen Layer ein sogenanntes Deep Neural Network. Dieser Prozess wird dann als Deep Learning bezeichnet. Die Architektur setzt sich im Wesentlichen aus den folgenden Punkten zusammen:

* 2D- oder 3D-Anordnung der Neuronen
* Geteilte Gewichte
* Lokale Konnektivität

Die Teilung der einzelnen Gewichte ist bei den Neuronalen Netzwerken ein Wesentlicher Punkt. So werden hier im Trainingsprozess durch die Iteration einer Vielzahl an Durchläufen die Gewichte auf die „gelabelten“ Daten immer genauer eingestellt. Anhand dieser Gewichte „entscheidet“ das Neuronale Netz, wie es die „gesehenen Daten“ interpretieren soll.

Um die Dichte an Informationen in einem Deep Neural Network verarbeiten zu können werden diese wie beschrieben komprimiert und immer weiter verdichtet. So werden wie in Abbildung 21 dargestellt, die Daten durch die Pooling- und Convolutuional Layer immer weiter „gestapelt“. Dadurch wird dann beim „fully connected Layer“ durch die vorab bestimmten Gewichte eine Entscheidung durch das neuronale Netz getroffen.

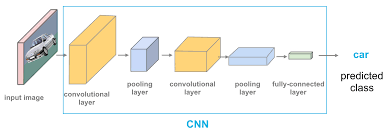


Abbildung : Grundsätzlicher Schematischer Aufbau des Neuronalen Netzes

## Architektur

Die Architektur des Neuronalen Netzes wird aus dem Vortrainierten Netzwerk sowie diverser Anpassungen im Rahmen erstellt. Als Basis des neuronalen Netzes wird hier auf das NasNETMobile Netzwerk gesetzt. Durch die Implementierung verschiedener Weiterer Layer, wird das Neuronale Netz auf die gewünschten Anforderungen angepasst.

Durch die Eingabe der Normalisierten Bilder in das Vortrainierte Netzwerk sowie die Anwendung von „Average Pooling“ wird die Bildinformation weiter komprimiert und zusammengefasst. Wie der Name schon sagt, wird hier der Durchschnitt der zusammengefassten Bildinformation verwendet. Im Anschluss werden die Informationen geebnet und durch einen „Dense Layer“ werden die Inputs und Outputs der Netzstruktur miteinander verbunden. Nachfolgend wird in diesem Prozess eine Gewisse Anzahl der Daten wieder Beseitigt (hier 0,5 um „Over Fitting“ des Trainings zu vermeiden. Anschließend werden nochmals „Dense Layer“ und Dropouts verwendet, bevor die „Gestapelten Daten Durch eine „Softmax“ Funktion ausgegeben werden. Abbildung 22 zeigt hier die im Anschluss an das Pretrained Network angefügten Elemente. Diese Elemente werden im Verlauf des Trainings trainiert, alle davor geschalten en Elemente (aus dem Pretrained Network) behalten Ihre Gewichtungen und werden nicht erneut gewichtet.

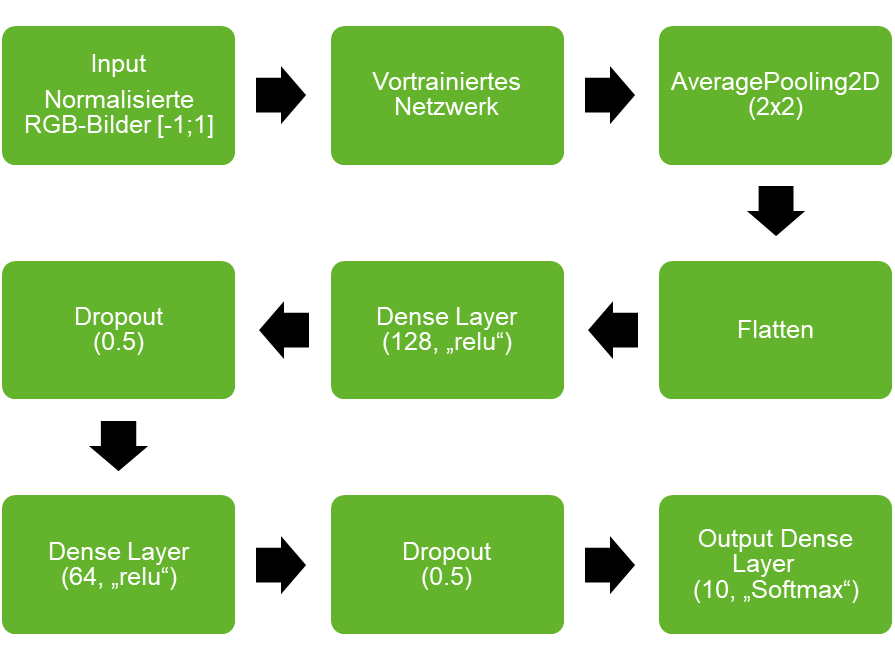


Abbildung : Darstellung der zum Pretrained Network hinzugefügten Netz-Elemente

## Datensatz

Für diese Aufgabe der Gestenerkennung, werden in diesem Projekt zwei Verschiedene Datensätze verwendet. Dies ist notwendig, um für das Neuronale Netz Schnell Ergebnisse zu erzielen und die Komplexität der Daten im Ersten Schritt zu reduzieren. Die verwendeten Handzeichen sind hierfür in Beiden Datensätzen identisch, jedoch wird bei dem „Eigenen Datensatz“ ein Homogenerer grüner Hintergrund verwendet. Hingegen wird bei dem verwendeten Datensatz aus China eine Variation von Hintergründen und Umgebungen verwendet, was die Komplexität entsprechend steigert.

**„Eigener Datensatz“**

Für die Erzeugung der Daten wurde wie in Kapitel 1.3 beschrieben die Kamera entsprechend aufgebaut, um diese auf den farblichen Hintergrund entsprechend auszurichten.

Anschließend wird von jeder der zehn Kategorien jeweils 800 Bilder durch ein Python Skript generiert. Während dessen wird die Hand Zyklisch unter der Kamera bewegt, um hier eine zusätzliche Variation der Position zu erzielen. Die Daten werden im Anschluss im Verhältnis 0.70 (Trainingsdaten), 0.15 (Testdaten) 0.15 (Validationsdaten) anhand eines Split-Algorithmus in den entsprechenden Ordnerstrukturen abgelegt. Dadurch kann im Späteren Verlauf des Trainingsprozesses auf für das Netzwerk unbekanntes Bildmaterial zurückgegriffen werden. In Abbildung 23 werden zehn Exemplare jeder Kategorie dargestellt, um die Zuordnung der einzelnen Symbole zu erkennen.



Abbildung Auswahl der Handgesten

**„China Datensatz“**

Bei dem Online verfügbaren Datensatz aus China, handelt es sich um einen durch Video Aufgezeichneten Datensatz mit weit über 40.000 Bilder. Dieser besitzt in Summe über 83 Klasen für die verschiedensten Hand-Gesten. Da dieser Datensatz eigentlich dafür gedacht ist, mittels Überlagerung der einzelnen Bilder auch Richtungen sowie eine Abfolge einer Geste zu erkennen, musste dieser vor Beginn stark selektiert werden. Neben der Verwendung nur einer Auswahl an Klassen, mussten auch die „bewegten“ Gesten aussortiert werden, welche für das menschliche Auge noch nicht als die gewünschte Geste zu erkennen war. Somit ist dieser Datensatz eine sehr gute Grundlage, jedoch nicht pauschal für unseren Fall anwendbar.

Der von uns als „China-Datensatz“ bezeichnete Datensatz entstammt der folgenden Quelle:

<http://www.nlpr.ia.ac.cn/iva/yfzhang/datasets/egogesture.html>

Auf der Seite kann der Downloadlink für die Daten per E-Mail beantragt werden.

## Vergleich des Neuronalen Netzes auf „China Datensatz“

Bei der Auswertung des Trainingsdatensatz des „Chinesischen Datensatzes“ fällt auf, dass das Neuronale Netz bei 50 Epochen hier trotz entsprechender Segmentierung der Daten Nur eine Genauigkeit von ca. 73% erreicht.

Gleichzeitig tendiert der „Loss“ des Trainingsprozesses gegen den Wert von 0,7, was darauf schließen lässt, dass selbst eine Erhöhung der Epochenzahl keinen weiteren Einfluss auf das Ergebnis hat. Hierbei ist bei der Analyse der Testbilder auffällig, dass Selbst bei der im vorigen Kapitel beschriebenen Überprüfung des Datensatzes, immer noch Bilder Erscheinen, welche nur ein Geübtes menschliches Auge der entsprechenden Kategorie zuordnen kann. Dies ist in Abbildung 24 eindeutig zu erkennen. Hier sind zum einen die Trainingsergebnisse über die einzelnen Epochen, sowie die vier Beispielbilder zu sehen. Schwierigkeiten bereitet hier dem Netzwerk im Besonderen die Unterscheidung der Zahlen 2 und 3 sowie je nach Position die Darstellung der Zahlen 0 und 7. Dies ist auf die Ähnlichkeit der jeweiligen Datenpaare zurückzuführen.

Aus diesem Grund werden alle weiteren Untersuchungen auf dem eigenen Datensatz durchgeführt.

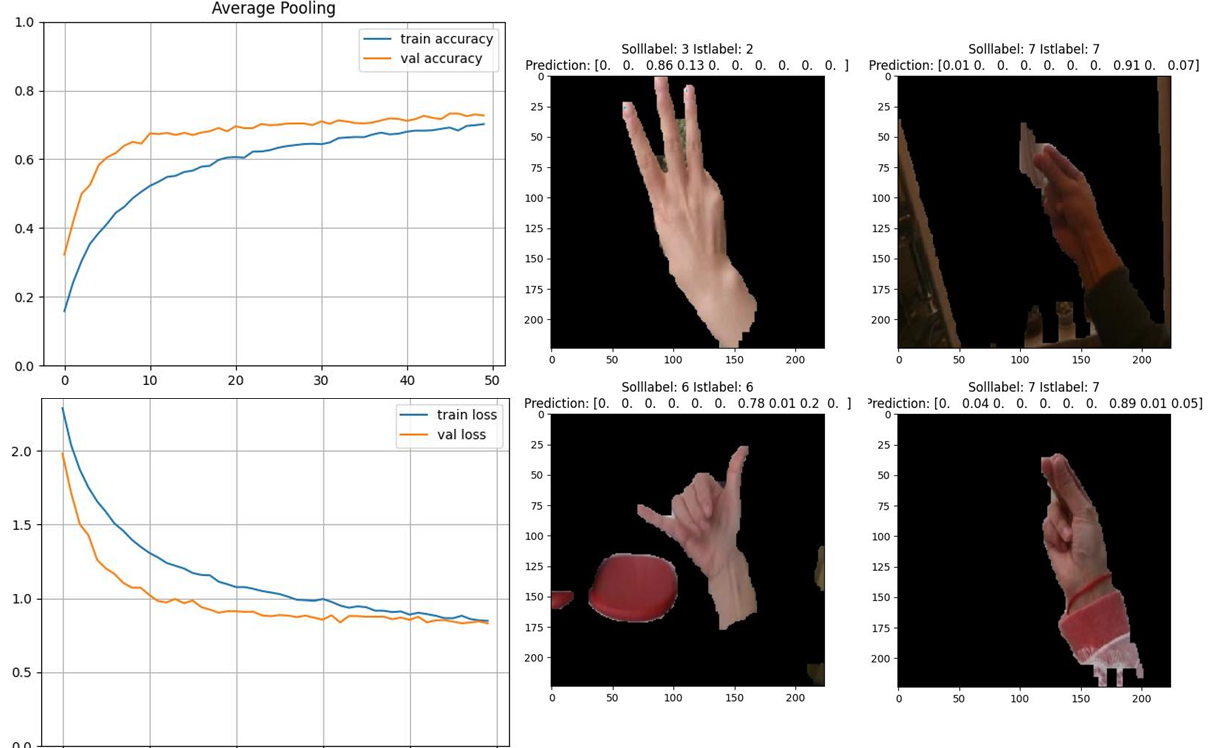


Abbildung Ergebnis des Neuronalen Netzes auf den China Datensatz

, mit vier Beispielbildern

## Vergleich des Neuronalen Netzes auf „Eigenen Datensatz“

Bei der Auswertung des Trainingsprozess in Bezug auf den eigenen Datensatz, wird im Vergleich zum Chinadatensatz eine Wesentlich höhere Genauigkeit bei gleicher Netzwerkarchitektur erzielt. Bei dem Eigenen Datensatz erzielt das Netzwerk eine Genauigkeit von 99% bei einem Loss von ca. 0.3. Dies ist bei unserer Ausgewählten Architektur ein sehr zufriedenstellender Wert. Der Verlauf über die einzelnen Epochen ist hierbei in der untenstehenden Abbildung 25 zu sehen. Die Vier Beispiel Bilder zeigen hier schön, dass alle Bilder auch bei zum Teil bei leicht fehlerhafter Segmentierung richtig erkannt werden.

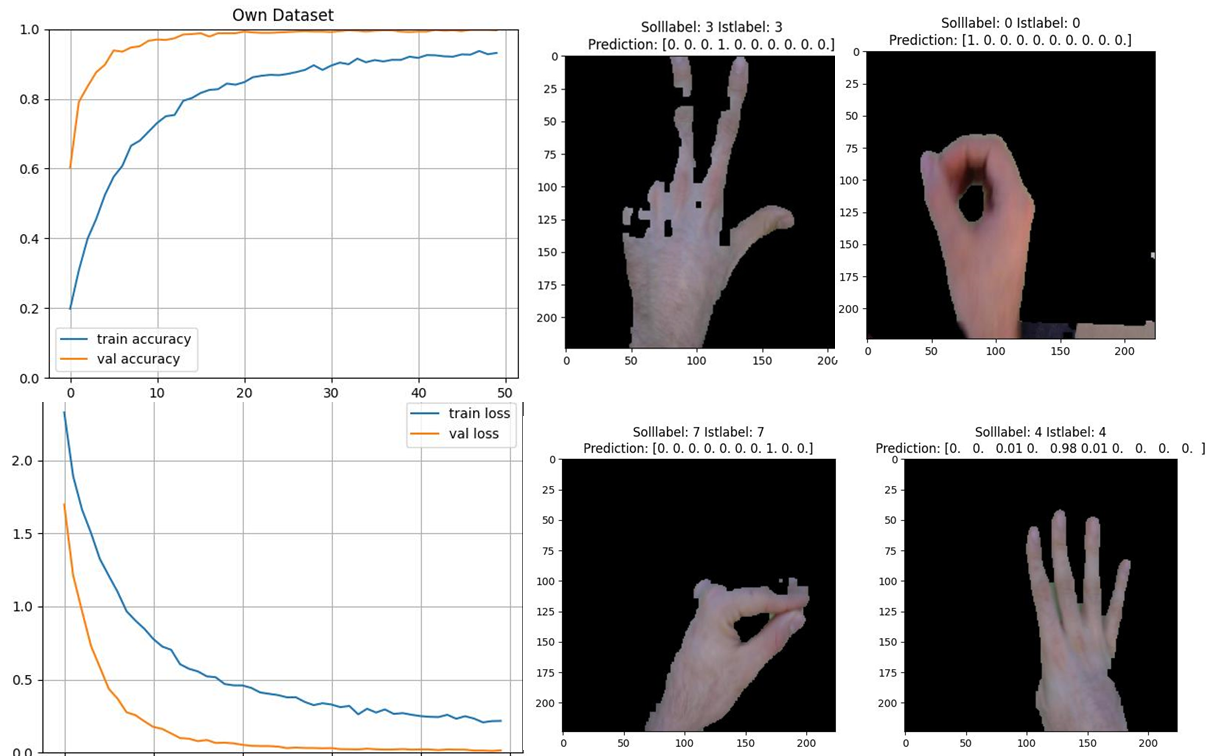


Abbildung Ergebnis des Neuronalen Netzes auf den Eigenen Datensatz

,  
mit vier Beispielbildern aus dem Test-Datensatz

## Confusion Matrix

Die Konfusionsmatrix stellt die Zusammenfassung der Vorhersageergebnisse zu einem Klassifizierungsproblem dar. Damit kann die Genauigkeit der einzelnen Klassen im Bezug zwischen ist- und Soll-label definiert werden. Die Anzahl der richtigen und falschen Vorhersagen wird mit Zählwerten zusammengefasst und nach jeder Klasse aufgeschlüsselt. Dies ist der Schlüssel zur Konfusionsmatrix. Durch die Konfusionsmatrix kann nicht nur ein Fehler aufgezeigt, sondern auch der Bezug zwischen ist und Solllabel dargestellt werden. Dadurch kann die Klassifizierungsgenauigkeit bestimmt werden.

Im Fall der Handgestenerkennung ist hierbei in der Untenstehenden Darstellung zu erkennen, dass es beim Eigenen Datensatz lediglich in drei verschiedenen Klassen zu Abweichungen kommt. So wird wie in der Abbildung zu sehen ist bei 90 Testbildern anstelle der „1“ eine „6“ erkannt. Dies ist auf die Ähnlichkeit zurück zu führen, da bei beiden jeweils ein Finger in der Vertikalen ausgestreckt wird. Ein Ähnliches Verhalten ist bei den Gesten „1“ und „2“ zu erkennen, diese unterscheiden sich lediglich durch einen mehr ausgestreckten Finger, was beim Neuronalen netz zu Problemen führt, wenn diese zu nahe beieinander sind. Ein Weiterer Fehler wie er i der Konfusionsmatrix ersichtlich wird, ist bei der“4“ zu erkennen, hier wird anstelle eine „2“ erkannt. Alle anderen Klassen werden vom System Fehlerfrei erkannt und sind entsprechend wie in Abbildung 26 zu erkennen.

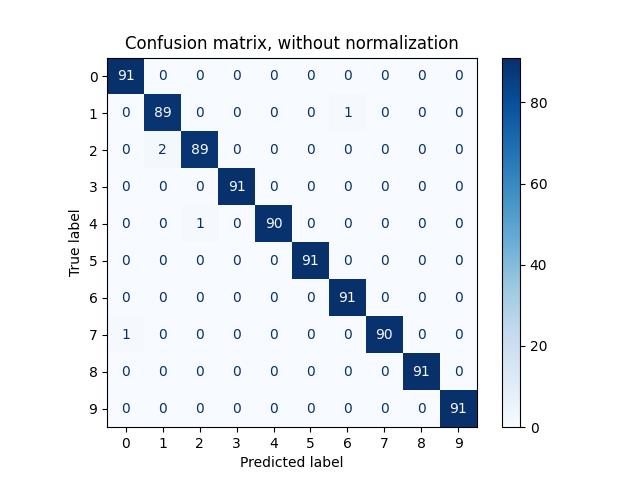


Abbildung : Ergebnis der Confusion-Matrix

Grundsätzlich kann man wie in Abbildung 27 zu erkennen ist, die Hohe Genauigkeit des Netzwerks in Verbindung mit der Segmentierung hervorheben. Die Einzelnen Gesten werden nahezu Fehlerfrei segmentiert und das entsprechende Label ausgegeben.

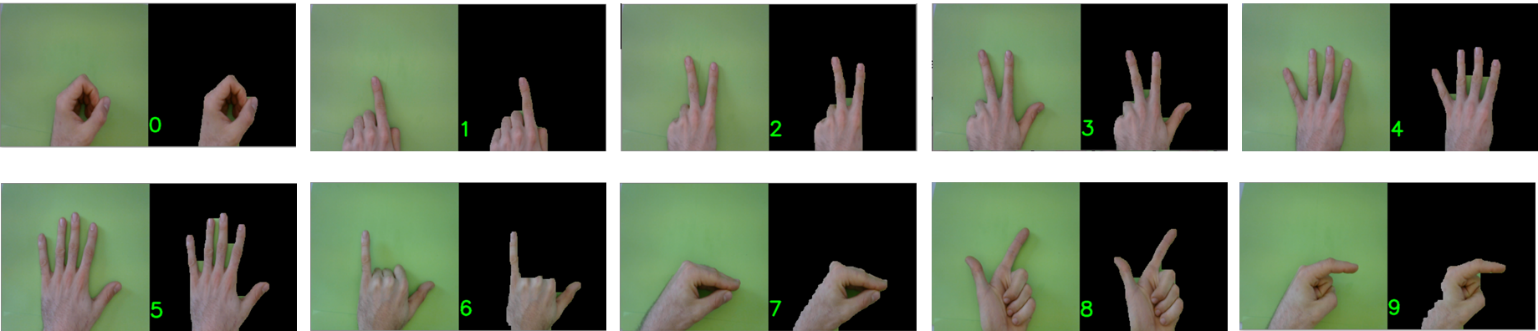


Abbildung : Erkennung aller zehn Handgesten mit dem „Predicted Label"

, des neuronalen Netzwerks

# Ausblick

1. Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | Q. Zhang, M. Yang, K. Kpalma, Q. Zheng und X. Zhang, „Segmentation of Hand Posture against Complex Backgrounds Based on Saliency and Skin Colour Detection,“ *IAENG International Journal of Computer Science,* Bd. 45, Nr. 3, pp. 435-444, August 2018. |
| [2] | G. Gomez und E. F. Morales, „Automatic feature construction and a simple rule induction algorithm for skin detection,“ 2002. |
| [3] | S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias und scikit-image contributors, „scikit-image 0.17.2: Image processing in Python,“ 2014. [Online]. Available: https://scikit-image.org/. [Zugriff am 14 Januar 2021]. |