

## [Team 15]

---

Frederik Rieß Pit-Aurel Ehlers Jascha Schmidt Felix Willrich

---

# **[Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]**

## **Systembeschreibung**

1.	Ziel und Zweck des Dokumentes .....	3
1.1.	Projektbeschreibung .....	3
1.1.1.	Kurzbeschreibung des Projekts .....	3
1.1.2.	Zweck des Projekts .....	4
1.1.3.	Hintergrund, Problemstellung, Motivation für das Projekt .....	4
1.1.4.	Ziele des Projekts .....	4
1.1.5.	Erfolgskriterien .....	4
2.	Systemübersicht .....	5
2.1.	CNN .....	5
2.2.	Skripte .....	7
2.2.1.	cutImages.py .....	7
2.2.2.	Copy_pictures.py .....	8
2.2.3.	Picture_random_sort.py .....	9
2.2.4.	picturesize.py .....	9
3.	Architektur und Designentscheide .....	10
3.1.	Programmiersprache und Umgebung .....	10
3.2.	Aufbau des Jupyter Notebooks .....	10
3.3.	Daten (Mengengerüst & Strukturen) .....	13
4.	Umgebungs-Anforderungen .....	16
4.1.	Technologie-Voraussetzungen .....	16
4.2.	Kooperierende Anwendungen und COTS-Komponenten .....	16
5.	Testplan .....	17
5.1.	Testverfahren .....	17
5.2.	Standardmodel .....	17
5.3.	Image Augmentation .....	18
5.3.1.	Rotation_range .....	19
5.3.2.	width_shift_range .....	22
5.3.3.	shear_range .....	25
5.3.4.	Flip Image .....	27
5.4.	Dropout .....	30
5.4.1.	Dropout von 0 .....	30
5.4.2.	Dropout von 0.2 .....	31
5.4.3.	Dropout von 0.5 .....	32
5.5.	Input Size .....	33
5.5.1.	Auflösung 124x60 .....	34
5.5.2.	Auflösung 60x54 .....	34
5.5.3.	Auflösung 51x42 .....	35
5.6.	BatchNormalization und Regularizer .....	36
5.6.1.	Ohne BatchNormalization .....	36
5.6.2.	Ohne BatchNormalization/mit Bias .....	36

5.6.3. Mit BatchNormalization .....	37
5.6.4. Kernel und Activity Regularizer .....	38
5.6.5. Kernel Regularizer .....	39
6. Ausblick .....	40
7. Abbildungsverzeichnis.....	41
8. Tabellenverzeichnis.....	42
9. Projektabschluss .....	43

## Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	14.03.2019	Felix Willrich	1. Entwurf + Eintragen aller Informationen	Abgeschlossen
0.2	08.06.2019	Felix Willrich	Kapitel 1,3.2, 5 angefangen	Abgeschlossen
0.3	09.06.2019	Frederik Rieß	Erste Verbesserungen durchgeführt Kapitel 2.1 fertiggestellt Kapitel 2.2 begonnen (cutlImages.py)	Abgeschlossen
0.4	10.06.2019	Frederik Rieß	Kapitel 3 angefangen	Abgeschlossen
0.5	10.06.2019	Felix Willrich	Kapitel 2 Skripte beschrieben	Abgeschlossen
0.6	14.06.2019	Felix Willrich	Kapitel 5 geschrieben + Tabellenverzeichnis + Abbildungsverzeichnis + Allgemeine Formatierung	Abgeschlossen
0.7	14.06.2019	Frederik Rieß	Kapitel 3 beendet Kapitel 5 beendet	Abgeschlossen
1.0	15.06.2019	Frederik Rieß	Generelle Formatierung Ausblick geschrieben Anlagen hinzugefügt	Abgeschlossen

## 1. Ziel und Zweck des Dokumentes

Dieses Dokument beschreibt die Anforderungen der T-Systems on site services GmbH. Es handelt sich hierbei um die Systemdefinition, die der Auftragnehmer für den Auftraggeber (Kunde) erstellt, sodass der Kunde versteht und validieren kann, was das System leisten wird.

### 1.1. Projektbeschreibung

Dieses Projekt wird im Rahmen des Modules „Teamprojekt“ durchgeführt, welches von Herrn Kircher und Frau Schiering doziert wird. Kunde für dieses Projekt ist Herr Philip May, welcher Angestellter bei der T-Systems on site GmbH ist und gleichzeitig die Rolle des Projektansprechpartners einnimmt.

#### 1.1.1. Kurzbeschreibung des Projekts

Das Projekt folgt einem gewissen Ablauf. Die Daten werden eingelesen, verarbeitet und ausgegeben. Diese drei Schritte werden anhand von folgendem Bild verständlich.

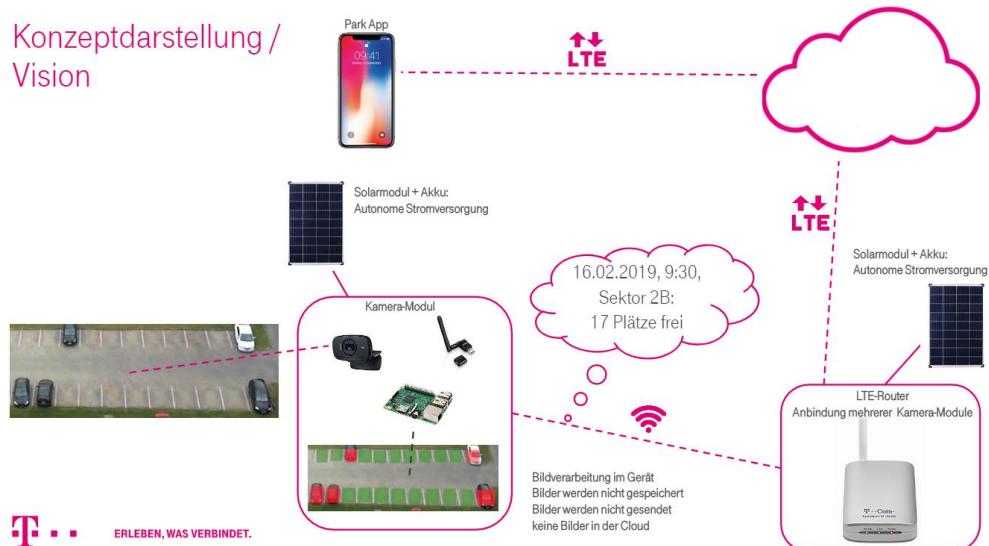


Abbildung 1: Produkt-Zyklus

Eine Kamera überträgt ein regelmäßig aufgezeichnetes Bild von einem Parkplatz an einen Mikrocontroller. Dieser verarbeitet die Daten, indem er das Bild in das konzipierte neuronale Netzwerk schickt. Anhand der Aufnahme soll bestimmt werden, wie viele Parkplätze frei sind. Diese Information wird an eine zentrale Stelle geschickt und weiterverteilt an eine Smartphone-App, damit die Nutzer zu jeder Zeit abrufen können, wohin sie fahren sollten. Die Bilder werden nicht vom Mikrocontroller weitergeschickt.

Da aufgrund der Zeit Abstriche gemacht werden müssen, werden wir uns in diesem Projekt auf die Erkennung von freien Parkplätzen konzentrieren. Das bedeutet, dass wir keine Live-Daten aus der Kamera bekommen werden, bzw. auch keine App erstellen werden, da dieses den Rahmen des Teamprojektes sprengen würde.

### 1.1.2. Zweck des Projekts

Das Projekt soll in erster Instanz zur Erkennung von freien Park-Flächen eingesetzt werden und im Rahmen von Parkplätzen und Parkhäusern genutzt werden. Die Technik kann mit verschiedenen Inputdaten auf verschiedenste Felder ausgeweitet werden. Beispiele wären, Lagerbestände oder den Füllstand von verschiedenen Containern erkennen.

Während des Projekts ist deutlich geworden, dass die Anforderungen geändert werden müssen. Dies lag vornehmlich an dem Zeitaufwand bzw. an der Kommunikation mit dem Kunden. Es wurden verschiedene Annahmen getroffen und zum Sprint 2 hat sich herauskristallisiert, dass dieses Produkt vornehmlich zum Erkenntnisgewinn für den Kunden und uns gelten sollen. Damit keine Unklarheiten auftreten, wurden die Ziele aus dem ersten Gespräch mit dem Kunden formuliert. Im weiteren Verlauf der Dokumentation werden die Testergebnisse und Erkenntnisse niedergeschrieben.

### 1.1.3. Hintergrund, Problemstellung, Motivation für das Projekt

Die T-Systems on site services GmbH in Person von Philip May benutzt im produktiven Sektor verschiedene Machine Learning/Deep Learning Applikationen und möchten durch dieses Produkt in weitere Felder stoßen bzw. weitere Erkenntnisse darüber gewinnen.

Für die Gruppe ergibt sich aufgrund von wenig Vorkenntnissen folgende Probleme:

- Neues Umfeld kennen lernen
- Geeignete Tools und Umgebung finden
- Datenbeschaffung zum Anlernen
- Prototypen erschaffen
- Genaue Erkennung implementieren
- Testumgebung

Die meisten Probleme werden oder wurden mit unserem Ansprechpartner besprochen und teilweise aufgearbeitet.

Die Motivation zu diesem Projekt ergibt sich aus dem ersten Stichpunkt der Probleme. Die Gruppe möchte in ein neues, aufstrebendes und sehr interessantes Thema einsteigen und dabei gleichzeitig Praxiserfahrung sammeln.

### 1.1.4. Ziele des Projekts

Das Projekt wird zuerst bis zu dem Schritt entwickelt, bis das Netz angelernt ist und verschiedene Parkplatzsituationen erkannt werden. Eine Genauigkeit von 99% wird angestrebt.

### 1.1.5. Erfolgskriterien

Sollte die gewünschte Genauigkeit erlangt worden sein, wird das Projekt als Erfolg bezeichnet. Weiterhin hinzukommen würden verschiedene Umgebungen, wie Schnee, Regen und andere Hindernisse wie Baulöcher oder Belegung von zwei Parkplätzen gleichzeitig. Sollten diese zusätzlichen Kriterien erfüllt werden, wird das Projekt in vollem Umfang als Erfolg gewertet. Es wird eine möglichst genaue Erkennung mit allen unterschiedlichen Faktoren angestrebt.

## 2. Systemübersicht

In diesem Kapitel wird zuerst das Grundkonzept eines CNNs beschrieben. Gleichzeitig werden hier die Skripte beschrieben, die für unser Projekt essential waren. Unser eigenes Netz wird unter dem Punkt 3 erklärt. Das ganze Projekt ist auf Github hochgeladen.  
<https://github.com/Scantraxx123/Parkplatzerkennung>

### 2.1. CNN

Convolutional Neural Networks haben sich bei vielen Aufgaben bewiesen, die der Bilderkennung dienen. Diese Netzwerke bestehen aus mehreren Convolutional und Max Pooling Layern und erkennen bestimmte Pattern (Ecken, Linien etc.) in Bildern. Je tiefer das CNN entwickelt ist, desto genauer kann das Netz bestimmte Objekte erkennen (in unserem Fall Parkplätze mit/ohne Autos).

Jedes Convolutional Layer hat eine gewisse Anzahl an Filtern, die diese Pattern erkennen. Zunächst werden die zu trainierenden Bilder mit einer festgelegten Größe und einer Anzahl an Channel eingelesen. Dabei kann durch die Channel angegeben werden, dass das Model zum Beispiel mit RGB (3 Channel) oder Graustufen (1 Channel) trainiert wird. Das erste Convolutional Layer liest die Bilder also ein und lässt anschließend die verschiedenen Filter über jedes Bild laufen. Ein Filter kann dabei als kleine Matrix gesehen werden, die eine bestimmte Anzahl an Zeilen und Spalten hat (hier haben sich häufig Filter einer Dimension 3x3 bewiesen). Die Werte in dieser Matrix werden mit zufälligen Werten initialisiert. Der Filter „gleitet“ dann über jeden Block von 3x3 Pixeln des zu trainierenden Bildes. Aus dem Filter und dem Ausschnitt des Bildes wird dann das Punktprodukt gebildet und an eine bestimmte Stelle der Output-Matrix geschrieben. Dabei ist die Output-Matrix so groß, wie es Möglichkeiten für den Filter gibt, über das Bild zu gleiten. Die Tiefe dieser Matrix wird durch die Anzahl der Channel bestimmt. Die Abbildung verdeutlicht diesen Vorgang.

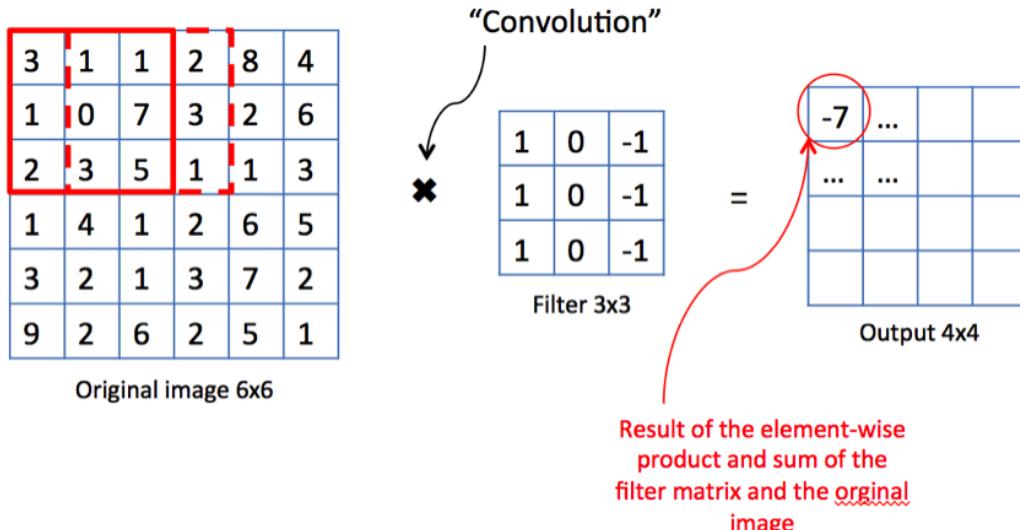


Abbildung 2: Convolution mit Filter

Anschließend wird das sogenannte Max Pooling angewandt. Dabei wird zunächst die Größe des Filters festgelegt, der über die vorherige Matrix laufen soll (häufig eine 2x2-Matrix). Zudem wird die Schrittweite festgelegt, die der Filter über das Bild laufen soll. Wenn nun der Filter über jeden Ausschnitt (2x2) des eingelesenen Bildes läuft, wird von diesen 4 Pixeln der höchste Wert

ermittelt und in eine neue Matrix geschrieben. Die folgende Abbildung zeigt diesen Schritt. f steht dabei für die Größe des Filters und s für die Schrittweite.

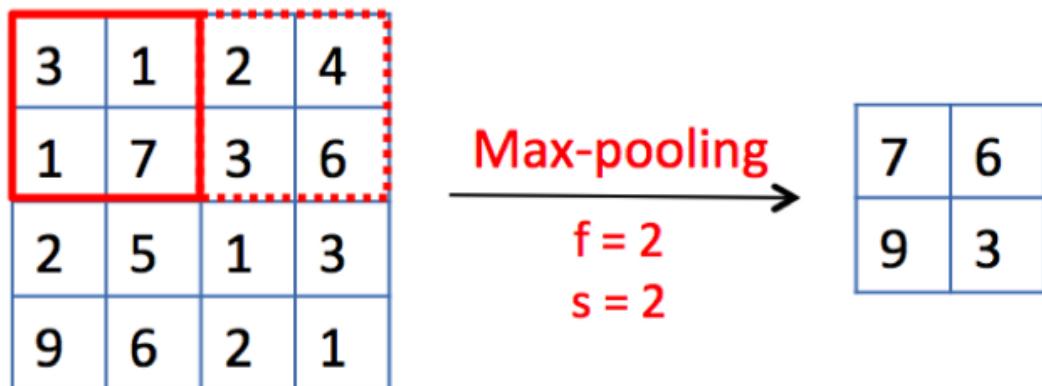


Abbildung 3: Max-Pooling

Wie zu sehen ist, wird durch diese Operation die Größe der ursprünglichen Matrix enorm reduziert. Im Gegensatz dazu wird die Anzahl der Filter immer höher.

Nach weiteren Convolutional und Max Pooling Layern wird ein sogenanntes Flatten genutzt, um die Matrix „abzuflachen“. Dies bedeutet einfach nur, dass die letzte Matrix in lediglich eine Spalte umgewandelt wird, um die Daten im Netz weiter zu verarbeiten. Anschließend folgen noch eine verschiedene Anzahl an Fully Connected Layern unterschiedlicher Größe und eine Funktion, wie die Daten am Ende auszugeben sind (in unserem Fall binär). Die Abbildung zeigt nochmals den kompletten Vorgang des CNNs mit einer abschließenden Softmax-Funktion, die für einen möglichen Output von mehr als zwei Kategorien sinnvoll zu verwenden ist.

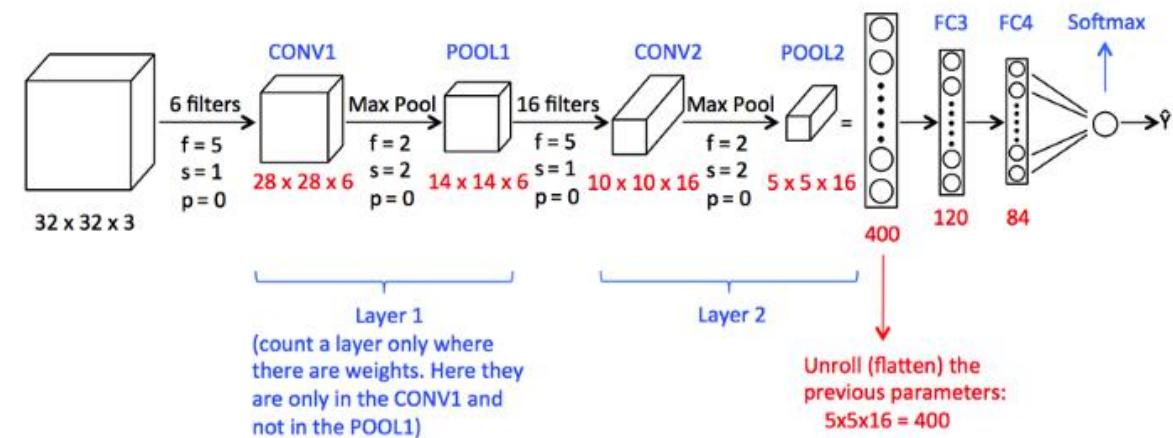


Abbildung 4: Übersicht über ein CNN

## 2.2. Skripte

In diesem Abschnitt werden alle Skripte beschrieben, die in unserem Workflow Verwendung gefunden haben.

### 2.2.1. cutImages.py

Dieses Skript wird dazu verwendet, um aus einem aufgenommenen großen Parkplatz und der dazugehörigen XML-Datei die einzelnen Parkplätze auszuschneiden. Dies geschieht mit den Python-Bibliotheken os, cv2 und xml.etree.ElementTree.

Das Einlesen der Bilder und der XML-Dateien geschieht mit der Bibliothek os. Nach dem Einlesen des Bildes wird unter demselben Namen die XML-Datei gesucht. Aus dieser werden wiederum die Koordinaten der Parkplätze sowie der Tag «Occupied» mit der Bibliothek xml.etree.ElementTree ausgelesen. Durch die Koordinaten wird anschließend mithilfe der Bibliothek cv2 die größtmögliche rechteckige Kontur innerhalb der vier Pixel gewählt, da letztere nicht ganz rechteckig sind und leichte Verschiebungen aufweisen. Diese einzelnen Rechtecke werden dann ausgeschnitten und unter einem passenden Namen in einen Ordner kopiert. Der exakte Ort des Bildes hängt dabei davon ab, ob der Tag «Occupied» des Parkplatzes dabei 0 oder 1 war. Dementsprechend werden die einzelnen Bilder auf die Ordner «Empty» und «Occupied» kopiert. Zur Veranschaulichung der Daten werden auch die einzelnen Parkplätze auf dem großen Bild markiert. Dieses wird anschließend in denselben Ordner neben «Empty» und «Occupied» gelegt. Ein belegter Parkplatz wird dabei rot umrandet, während ein grüner Parkplatz entsprechend freie Parkplätze zeigt.

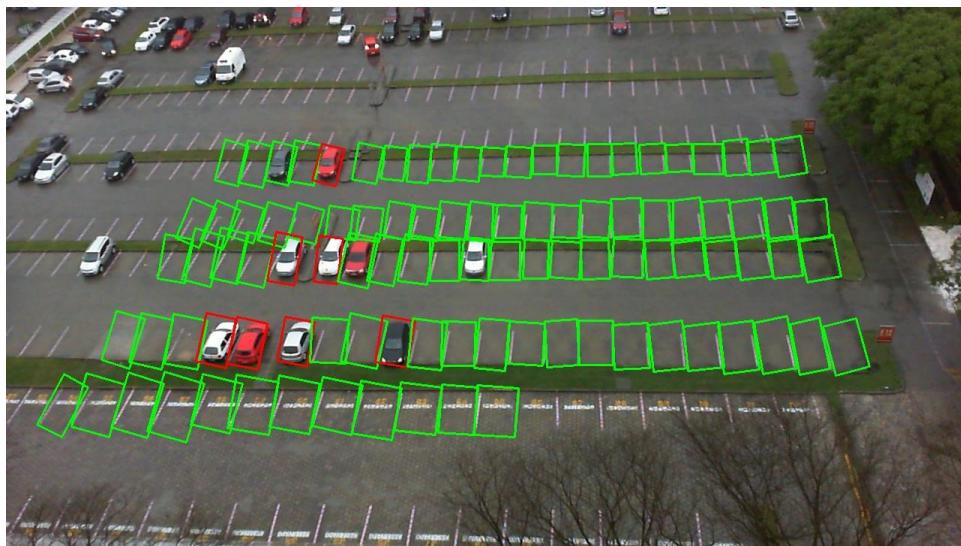


Abbildung 5:Datenfehler

Dabei ist aufgefallen, dass nicht alle Daten in der XML-Datei korrekt sind. Wie auf dem Bild zu sehen, haben einige Parkplätze falsche Einträge unter dem Tag «Occupied» und werden so falsch markiert. Da dies jedoch nicht bei allen Dateien auftritt ist der Einfluss zwar da, aber fällt bei der Masse an Daten kaum ins Gewicht.

Das Skript an sich ist für unser Projekt letzten Endes nicht von erheblicher Bedeutung gewesen, da es bei uns nur um die tatsächliche Erkenntnisgewinnung der zu trainierenden Daten ging und die Daten (Parkplätze) selbst schon einzeln ausgeschnitten waren. Allerdings kann es gut für zukünftige Arbeiten benutzt werden, wenn ein Bild mit einer XML-Datei vorliegt, von denen noch nicht die einzelnen Parkplätze ausgeschnitten wurden.

### 2.2.2. Copy\_pictures.py

Damit jedes Bild von den unterschiedlichen Datensätzen in unserer festgelegten Struktur vorliegt, wurden zwei Skripte geschrieben, die uns dabei unterstützen. `Copy_pictures.py` und `CNR_COPY.py`. Beide Skripte funktionieren mit dem gleichen Mechanismus.

Es gibt einen Quellordner und zwei Zielordner (`Empty`,`Occupied`). Danach wird der Quellordner mit all seinen Unterordnern nach Bildern durchsucht. Im Fall von `copy_pictures.py`, welches für das Parkinglot-Set genutzt worden ist, gibt es die Unterscheidung, ob das Bild aus dem Ordner „empty“ oder „occupied“ stammt. In dieser Fallunterscheidung wird berücksichtig in welchem Ordner die Bilder gespeichert werden müssen.

Bei dem Skript `CNR_COPY.py`, welches für das CNR-Set benutzt worden ist, wird die .txt Datei mit den Pfaden und dem jeweiligen Status (`Empty`,`Occupied`) noch eingelesen. Dies hilft, wie bei dem Skript zuvor bei der Fallunterscheidung und gleichzeitig dem Einsortieren.

Bei beiden Skripten wurde die Bibliothek „shutil“ benutzt. Diese besitzt Kopierfunktionen die Python direkt ab Werk mitliefert.

Da uns nach dem ersten Durchlaufen der Skripte aufgefallen ist, dass ein Kopiervorgang von fast 700.000 Bildern bzw. 150.000 Bildern mehrere Stunden dauern kann, haben wir uns nach Alternativen umgesehen. Wir sind in der Recherche auf die Libary „pyfastcopy“ (<https://pypi.org/project/pyfastcopy/>) gestoßen. Diese wird über den Packetmanager von Python(pip) installiert und ersetzt die alten `shutil`-Funktionen. Nach einem weiteren Durchlauf wurde der Kopiervorgang merklich beschleunigt.

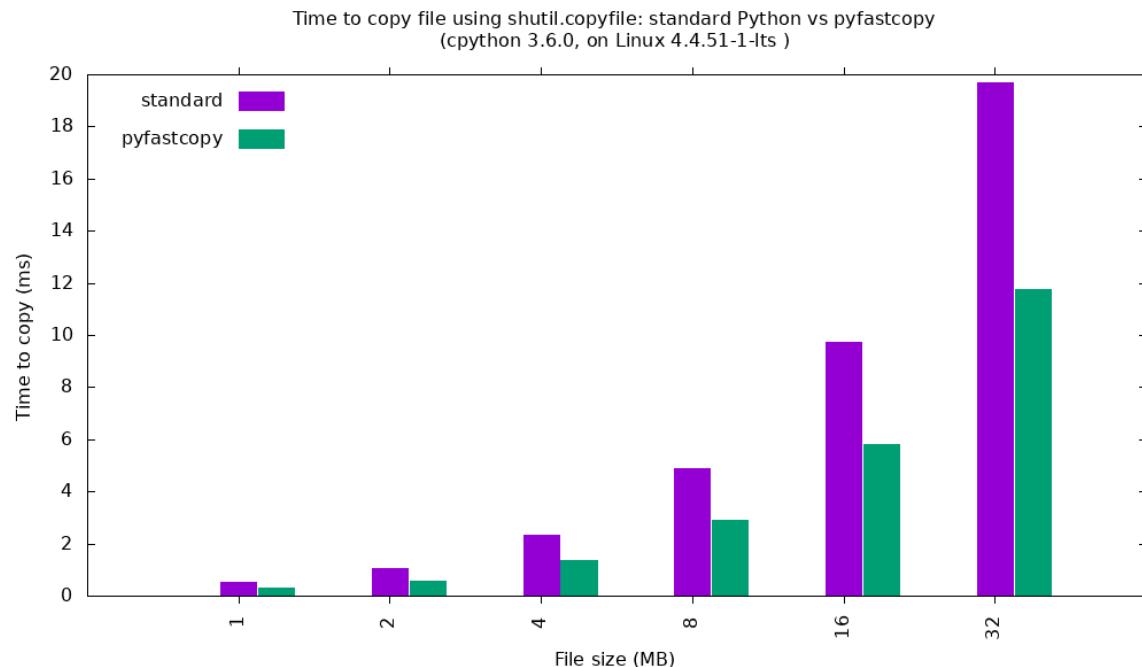


Abbildung 6: Vergleich pyfastcopy zu Standard

### 2.2.3. Picture\_random\_sort.py

Einige Bildpakete setzen voraus, dass alle Bilder zufällig einsortiert werden, damit das Netz keine falschen Muster lernt. Um dies zu garantieren, wurde ein Skript geschrieben, welches alle Bilder zufällig einsortiert in die jeweiligen Ordner.

Die Basis des Skripts funktioniert wie das copy\_pictures.py Skript. Es werden Verzeichnisse durchsucht und die jeweiligen Bilder in die Ordner einsortiert. Der einzige Unterschied liegt im Zufälligkeitsmechanismus. Beim Durchsuchen der Ordner werden alle Pfade in eine Liste geschrieben. Nachdem diese Liste befüllt ist, wird mit der Funktion „shuffle“ die Liste verdreht. Dies ermöglicht beim Kopieren der Bilder eine zufällige Einsortierung.

### 2.2.4. picturesize.py

Einer der Tests, die von uns in Sprint 2 und 3 durchgeführt worden sind, versucht herauszufinden, welche Größe von Bildern die Beste für das CNN ist. Damit das jeweilige Teammitglied einen Anhaltspunkt hat, welche Größen geeignet sein könnten, wurde ein Skript angefertigt, welches alle Größen ausliest und in eine Datei schreibt.

Die Libary „PIL“ besitzt diverse Funktionen, um Bilder auszulesen oder zu bearbeiten. Auch in diesem Skript werden alle Unterordner auf Bilder durchsucht. Sobald ein Bild gefunden worden ist, wird dies mit der Libary geöffnet und die Höhe und Weite der Bilder ein Dictionary geschrieben. Der Key ist die Höhe und Weite. Der Wert zu dem jeweiligen Key besteht aus einer Zahl, die aussagt wie häufig diese Bildgröße vorkommt. Alle Werte werden am Ende in .txt geschrieben. Außerdem wird der Durchschnitt berechnet aller Bilder.

Height	Width
Average:	70.29199785586282 44.725359308242716
54 36:	5493x
37 32:	4232x
43 36:	4236x
48 38:	4241x
57 34:	4233x
52 38:	4242x
41 39:	4231x
42 38:	4246x
47 35:	8473x
52 42:	4230x
73 52:	4235x
71 56:	4262x
69 54:	4275x
64 55:	4218x
56 33:	4242x
43 30:	4244x
47 31:	4244x
43 32:	4245x
57 33:	4238x
51 42:	8473x
59 46:	10738x
67 53:	4253x
70 54:	4220x

Abbildung 7: Größe Parklinglot Daten

### 3. Architektur und Designentscheide

#### 3.1. Programmiersprache und Umgebung

Python kann als Programmiersprache viele Aufgaben bewältigen. Im Bereich Data Science stehen dem Entwickler sehr viele nützliche Bibliotheken zur Verfügung. So gibt es auch im Umgang mit Jupyter Notebook eine einfache Möglichkeit, dies zu nutzen. Jupyter Notebook ist eine Open Source Web Application, die es ermöglicht, Dokumente mit Code oder auch Text und Visualisierungen online zu teilen.

In dem Notebook selbst kann durch das Aufteilen in verschiedene Zeilen einzelner Code geändert und ausgeführt werden, ohne dass dabei das gesamte Programm nochmals komplett durchlaufen wird. Dies ist vor allem nützlich, da zu Beginn eine Masse an Daten benötigt wird, die dann nicht nochmal geladen werden muss.

Da im Machine Learning und vor allem in der Bilderkennung die GPU eine große Leistung haben muss und wir selbst keine starke Grafikkarte zur Verfügung hatten, wurden die Notebooks über das Google Colab ausgeführt. Dort wird über eine Cloud gratis GPU für Aufgaben zum Maschinellen Lernen zur Verfügung gestellt. Zudem sind schon viele Bibliotheken wie Tensorflow, Keras oder OpenCV implementiert. In Zusammenarbeit mit den Jupyter Notebooks wird die Arbeit so enorm erleichtert (siehe auch Punkt 4).

#### 3.2. Aufbau des Jupyter Notebooks

##### Aufbereitung der Daten

Um die eingelesenen Daten optimal und effektiv zu bearbeiten, wurde die Klasse ImageDataGenerator genutzt. Diese wird von Keras zur Verfügung gestellt und liefert viele Möglichkeiten mit, die Bilder zu verarbeiten. Unter anderem können so die Daten eingelesen werden und gleich die Größe der Bilder angegeben werden. Außerdem kann über den ImageDataGenerator auch Augmentation betrieben werden, um Variationen in den einzelnen Bildern zu erzeugen. Ein weiter wichtiger Punkt war für diese Entscheidung, dass allein aufgrund der Ordnerstruktur beim Einlesen die Daten schon korrekt gelabelt werden.

Mit dieser Klasse können also viele zwingend notwendige Aufgaben in dem Projekt erledigt werden.

##### CNN

Bei dem Model für das neuronale Netz handelt es sich um ein Convolutional Neural Network, mit dem schon eine relativ hohe Validation-Accuracy von 99% erreicht wird. Das Model wurde so gewählt, dass es relativ viele Pattern in den Bildern erkennt, aber zeitgleich soll das Trainieren des Models pro Epoche nicht allzu lange dauern.

Die Aufgabe war es nun aus diesem Model durch die nötigen Tests Erfahrungen zu erzielen, die Änderungen an selbem erzeugen. Das Model besteht aus mehreren Convolutional und Max Pooling Layern. Als Aktivierungsfunktion wurde für die Convolutional Layer die Funktion «relu» (Rectified Linear Unit, Abbildung x) genutzt. Diese ist die meist genutzte Aktivierungsfunktion bei Deep Learning Models. Wenn die Funktion einen negativen Input erhält, wird 0 zurückgegeben. Andernfalls wird der Input auch wieder zurückgegeben ( $f(x) = \max(0, x)$ ).

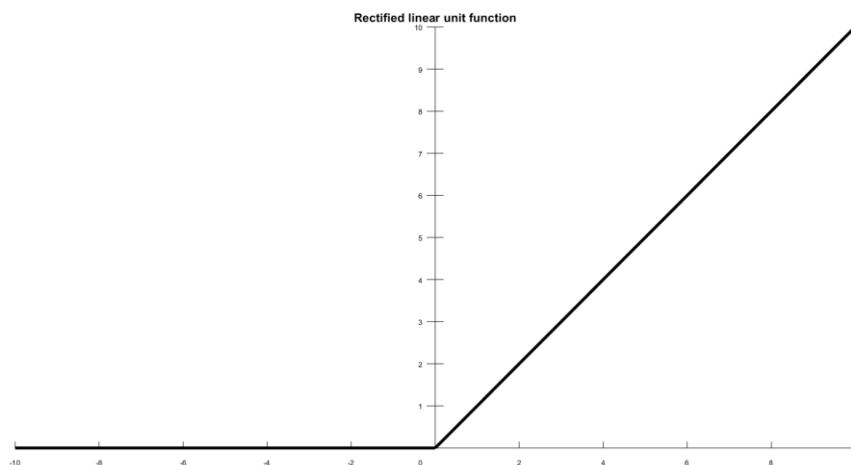


Abbildung 8: linear unit function

Da wir noch nicht viel Erfahrung mit neuronalen Netzen haben, orientieren wir uns also an dieser Funktion. Als letzte Aktivierungsfunktion musste jedoch die Sigmoid-Funktion gewählt werden, da wir aufgrund eines binären Problems (Empty oder Occupied) am Ende als Output einen Wert zwischen 0 und 1 haben.

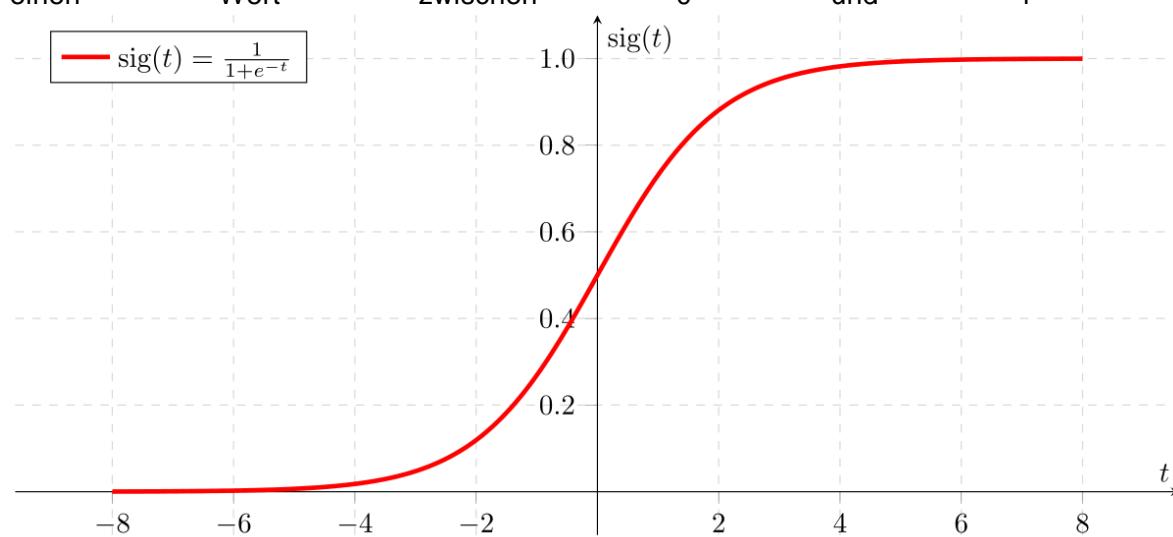


Abbildung 9: Sigmoid-Funktion

Dadurch, dass der Output nur zwischen 0 und 1 liegen kann, werden große Schwankungen beim Output verhindert und so Fehlerwerte vermieden. Zudem ist die Funktion an jeder Stelle differenzierbar, was insbesondere für die Backpropagation später von großem Nutzen ist.

Als Loss-Function wurde aufgrund des binären Problems die „binary\_crossentropy“ genommen. Der Optimizer für die Backpropagation ist „adam“, der für viele Problemstellungen im Deep Learning nützlich und immer eine gute Wahl ist. Für das Testen möglicher anderer Optimizer blieb leider keine Zeit. Des Weiteren sind auch mehrere Dropout-Layer und BatchNormalization in dem Model zu finden. Diese werden später bei den Tests nochmals genauer erklärt.

Wie genau das Model aufgebaut ist, ist der Abbildung x zu entnehmen.

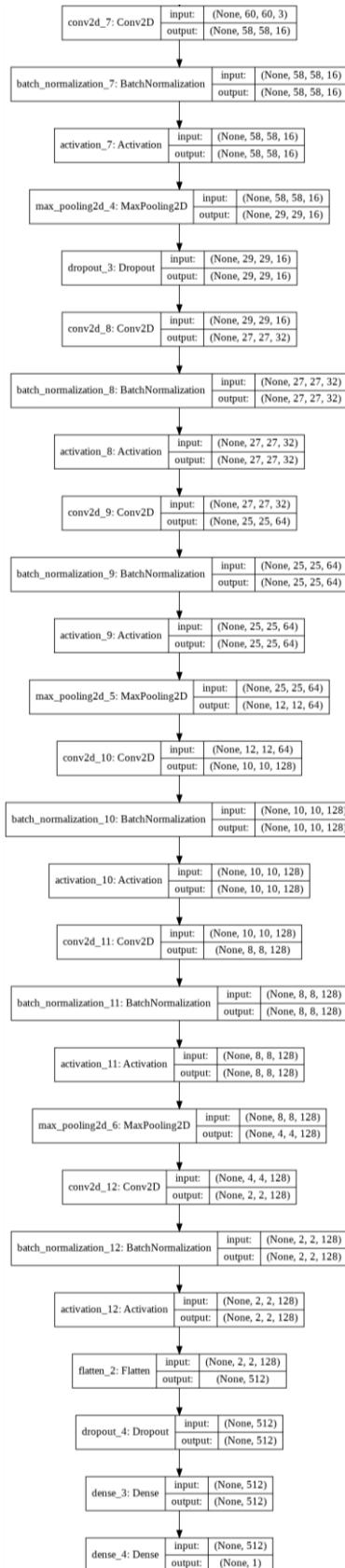


Abbildung 10: Verwendetes Netz mit Layern

### 3.3. Daten (Mengengerüst & Strukturen)

Alle Daten stammen aus zwei verschiedenen Quellen. Die erste Quelle (<https://web.inf.ufpr.br/vri/databases/parking-lot-database/>) wurde von unserem Kunden vorgegeben. Diese Daten stammen von der Informatik Fakultät der brasilianischen Universität «Universidade Federal do Paraná». In diesem Paket sind 2 Parkplätze über einen längeren Zeitraum aufgenommen worden. Ein Parkplatz davon aus zwei unterschiedlichen Kamerawinkeln. Dabei enthält das Paket die Aufteilung in Wettersituationen (sunny, cloudy, rainy), Tagen und in den jeweiligen Status des Parkplatzes (empty, occupied). Alle einzelnen Parkplätze wurden ausgeschnitten und in die einzelnen Ordner sortiert. Die ausgeschnittenen Bilder besitzen unterschiedliche Größen. Gleichzeitig gibt es die jeweiligen Gesamtbilder der Parkplätze mit einer dazugehörigen XML-Datei. Diese beinhaltet u.a. die Koordinaten, Größe sowie einen Booleanwert, ob der Parkplatz besetzt (1) oder frei ist (0). Insgesamt beinhaltet dieses Paket ca. 700.000 Bilder.

Der zweite Parkplatz (<http://cnrspark.it/>) ist bei Recherche zu Alternativen gefunden worden. Dieses Paket umfasst ca. 150.000 Bilder und nimmt Parkplätze mit neun verschiedenen Kameras auf. Auch hier wurden die Bilder in Wettersituation und Tage aufgeteilt. Der Unterschied in der Datenstruktur liegt dabei, dass die Bilder noch in Kameras aufgeteilt werden. Damit bestimmt werden kann, welche Parkplätze frei oder belegt sind, wurde eine .txt Datei angelegt mit Pfaden und einem Booleanwert.

Im Laufe des Projekts wurden diverse unterschiedliche Pakete angelegt, die verschiedenen Testsituationen zu Gute gekommen sind. Die Daten wurden jeweils aufgeteilt in train/validation Daten. Alle Daten wurden zufällig tageweise zugeordnet, damit keine doppelten Bilder vorkommen.

Name	Beschreibung
UFPR04_05_train_val_test.zip	Zum Ende hin bevorzugtes Paket. 80/20/20 train/validation/test split Insgesamt 66049 Bilder
PUC_50k.zip	50000 Bilder des PUC Parkplatzes vom Parkinglot Dataset in 80/20 Verteilung.
UFPR05_50k.zip	50000 Bilder des UFPR05 Parkplatzes vom Parkinglot Dataset in 80/20 Verteilung.
UFPR04_50k.zip	50000 Bilder des UFPR04 Parkplatzes vom Parkinglot Dataset in 80/20 Verteilung.
50k_All_Parking_Spaces.zip	50000 Bilder von allen Parkplätzen
ALL_ALL.zip	Alle Parkplätze in 80/20 Split. 837991 Bilder
CNR_TEST.zip	Paket, um den Testgenerator mit einem anderen Datensatz zu laden. Insgesamt 144965. Kein Split.
ALL_100K.zip	Zufällige Verteilung der Bilder (100000) der Parkplätze PUC/UFPR05/UFPR04/CNR in 80/20 Verteilung.
ALL_50K.zip	Zufällige Verteilung der Bilder (50000) der Parkplätze PUC/UFPR05/UFPR04/CNR in 80/20 Verteilung.
PUC_UFPR05_04_50_50.zip	Alle Bilder (695851) der Parkplätze

	PUC/UFPR05/UFPR04 vom Parkinglot Dataset in 50/50 Verteilung.
PUC_50_50.zip	Alle Bilder (424223) des Parkplatzes PUC vom Parkinglot Dataset in 50/50 Verteilung.
UFPR05_50_50.zip	Alle Bilder (165785) des Parkplatzes UFPR05 vom Parkinglot Dataset in 50/50 Verteilung.
UFPR04_50_50.zip	Alle Bilder (103522) des Parkplatzes UFPR04 vom Parkinglot Dataset in 50/50 Verteilung.
PUC.zip	Alle Bilder (424223) des Parkplatzes PUC vom Parkinglot Dataset in 80/20 Verteilung.
UFPR04_05.zip	Alle Bilder (261956) der Parkplätze UFPR04/UFPR05 vom Parkinglot Dataset in 80/20 Verteilung.
UFPR05.zip	Alle Bilder (165785) des Parkplatzes UFPR05 vom Parkinglot Dataset in 80/20 Verteilung.
UFPR04.zip	Alle Bilder (103522) des Parkplatzes UFPR04 vom Parkinglot Dataset in 80/20 Verteilung.

Tabelle 1: Daten-Aufteilung

Alle Pakete besitzen eine einheitliche Datenstruktur. Im Stammverzeichnis der Pakete befinden sich die Unterordner train und val. Der train-Ordner beinhaltet alle Bilder, die das Netz lernen soll und der val-Ordner alle, um das Netz zu überprüfen ob es richtig lernt. In den Ordner befinden sich nochmals die Ordner empty und occupied. Damit sind gleichzeitig alle Bilder mit Labels ausgestattet.

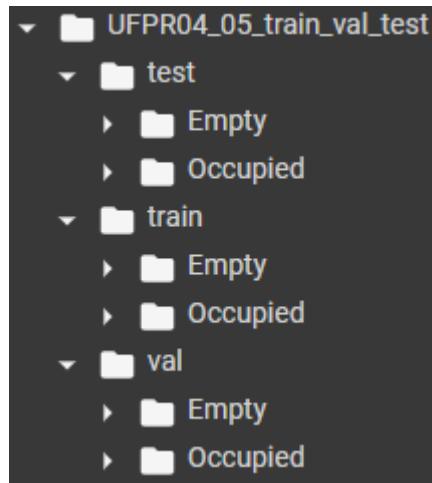


Abbildung 11: Ordnerstruktur Daten

Alle Pakete werden per Befehl aus OneDrive heruntergeladen, entpackt und in das Netz geladen.

```
!wget --no-check-certificate
```

```
"https://onedrive.live.com/download?cid=11F71A1654195330&resid=11F71A1654195330%2145939&authkey=AE2mvzAS3IHwVOs"
```

```
!unzip -q
```

```
"download?cid=11F71A1654195330&resid=11F71A1654195330!45939&authkey=AE2mvzAS3IHwVO
```

Abbildung 12: Befehle zum Herunterladen und Entpacken der Daten

Sobald diese Daten entpackt worden sind befindet sich die Struktur in dem jeweiligen Notebook. Damit die Bilder geladen werden können, wird programmatisch der Pfad angelegt.

```
base_train_dir = os.path.join(data_dir, 'train')
base_val_dir = os.path.join(data_dir, 'val')
base_test_dir = os.path.join(data_dir, 'test')
```

Abbildung 13: Ordnerstruktur im Code

## 4. Umgebungs-Anforderungen

### 4.1. Technologie-Voraussetzungen

Das System ist geschlossen aufgebaut und benötigt dadurch wenig Technologien. Im Folgenden werden die eingesetzten Technologien erklärt.

Name	Beschreibung
<b>Python 3.X</b>	Grundlage jeglicher Programmierung
<b>Keras</b>	Paket für maschinelles Lernen in Python geschrieben
<b>NumPy</b>	Paket, um das Rechnen mit Matrizen und Vektoren zu vereinfachen, Geschrieben in NumPy und wird passiv mitgenutzt
<b>Matplotlib</b>	Library für die Darstellung von Diagrammen
<b>Tensorflow</b>	Keras stützt sich auf Tensorflow
<b>Jupyter Notebook</b>	Sozusagen die IDE
<b>Foto Datenbank</b>	<a href="https://web.inf.ufpr.br/vri/databases/parking-lot-database/">https://web.inf.ufpr.br/vri/databases/parking-lot-database/</a> <a href="http://cnrpark.it/">http://cnrpark.it/</a> Grundlage zum Anlernen
<b>Google Colab</b>	Hardware Grundlage
<b>One Drive</b>	Speichermedium für die Bildpakete

Abbildung 14: Verwendete Technologie

Für unser Programm waren vor allem Keras bzw. Tensorflow und die Umgebung Google Colab wichtig. Keras bietet uns auf einer Hochsprachen-Ebene diverse Funktionen, um unser Netz zu konzipieren. Google Colab hingegen ist eine Plattform auf der potenteen Hardware kostenlos zur Verfügung gestellt wird. Es können dort Notebooks hochgeladen werden und durchgeführt werden. Dies haben wir benutzt, da kein Teammitglied zuhause ausreichende Hardware zur Verfügung stehen hatte.

### 4.2. Kooperierende Anwendungen und COTS-Komponenten

Das Notebook greift auf die hochgeladenen Pakete im OneDrive zu. OneDrive wird nur als Speichermedium genutzt, damit alle Teammitglieder jederzeit die Möglichkeiten haben, die Pakete herunterzuladen. Somit ist nur das Notebook von Nöten. Dazu wird ein Befehl ausgeführt, der sich per «wget» das Paket herunterlädt und danach entpackt. Die Pakete sind unter dem Punkt 3.2. genauer erklärt.

## 5. Testplan

Das Projekt sieht vor, wie Eingangs erklärt, ein neuronales Netzwerk zur Parkplatzerkennung aufzubauen. Da es in diesem Kontext keine sinnvollen Möglichkeiten gibt das Modell zu testen, wurden die Tests abgewandelt. Die Tests konzentrieren sich vor allem auf verschiedene Parameter und deren Wirkung auf das Netz und die Genauigkeit. Jedes Teammitglied hat sich dabei auf verschiedene Eigenschaften konzentriert. Diese Tests wurden ausdrücklich vom Kunden gewünscht.

### 5.1. Testverfahren

Jede getestete Eigenschaft wurde unter einem bestimmten Verfahren geprüft. Jede Eigenschaft wurde in 30 Durchläufen getestet, um ein möglichst genaues Ergebnis zu bekommen. In diesen 30 Durchläufen wurde das Prinzip des «early-stopings» benutzt. Dies sagt aus, dass, sobald nach 5 Epochen kein besseres Ergebnis erzielt worden ist, das Trainieren gestoppt wird. Die beste Validation Accuracy wurde dabei niedergeschrieben und zum Schluss in Konkurrenz mit den anderen Testergebnissen derselben Reihe gesetzt.

```
es = EarlyStopping(monitor='val_acc', patience=5, restore_best_weights=True, verbose = 2)
```

Abbildung 15: Early-Stopping

### 5.2. Standardmodel

Das Standardmodel bietet die Grundlage für jegliches weitere Testen. Das Model wurde unter dem Punkt 3 erklärt. Jede weitere Eigenschaft ist eine Erweiterung zum Standardmodel. Dies bedeutet gleichermaßen, dass sich alle Testreihen mit dem Standardmodel messen lassen müssen. Das Ziel der einzelnen Tests besteht darin das Standardmodel zu erweitern bzw. zu verbessern.

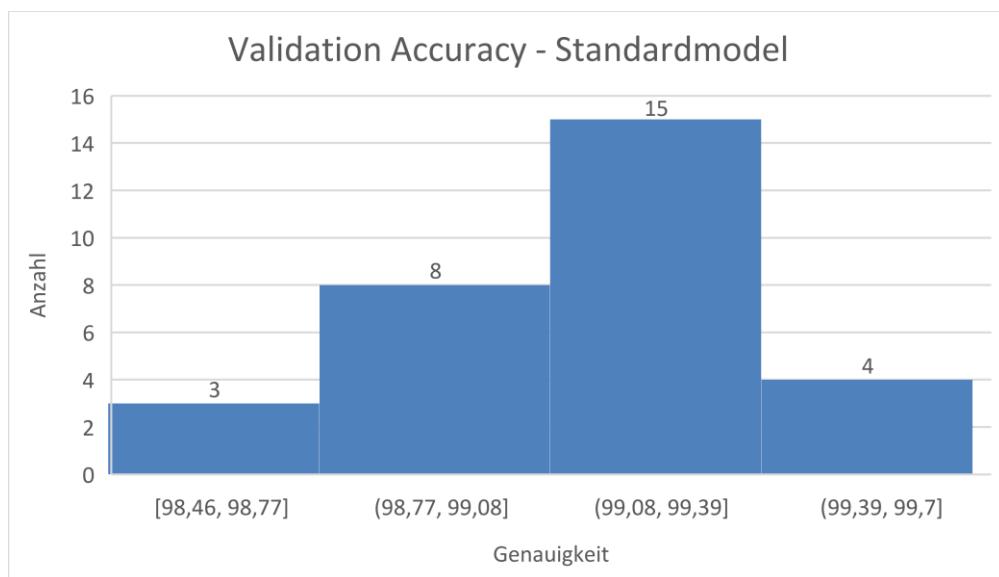


Abbildung 16: Validation Accuracy- Standardmodel - Histogram

Das Standardmodel ist in unserem Fall schon sehr genau. Ein Durchschnittswert von 99,08-99,39 wird zu 15 Mal erzielt in unseren Tests. Dies bedeutet für weitere Testreihen, dass wir versuchen werden die letzten Prozente aus der Simulation herauszukitzeln oder zumindest erkennen können, dass eine Veränderung des Netzes keine weiteren Vorteile für uns bringt.

Die hohe Genauigkeit kommt daher, dass unsere Testdaten relativ ähnlich sind und das Modell teilweise Parkplätze trainiert, die es gleichzeitig auch validiert. Diese sind zwar von anderen Tagen bzw. anderen Uhrzeiten, aber der Parkplatz hat sich in der Zwischenzeit nicht grundlegend verändert.

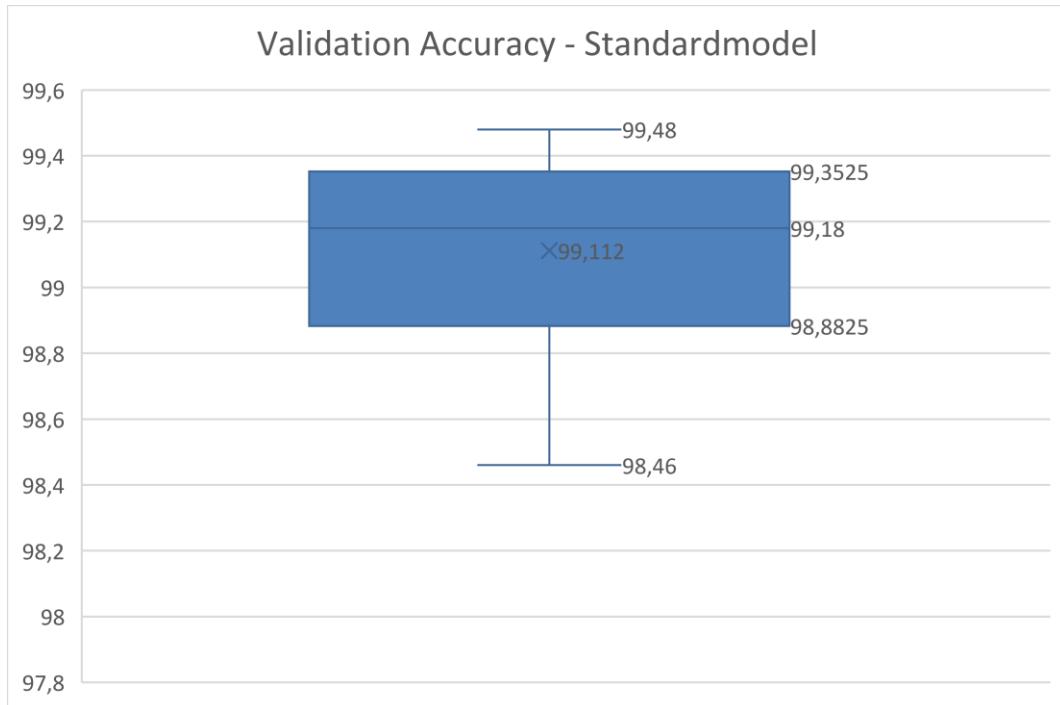


Abbildung 17: Validation Accuracy - Standardmodel - Boxplot

Als Referenzwert wird der Wert 99,112% angesehen.

### 5.3. Image Augmentation

Das Framework Keras bietet die Möglichkeit über den sogenannten «ImageDataGenerator» (<https://keras.io/preprocessing/image/>) eine Augmentation (Veränderung/Bearbeitung) der Bilder durchzuführen. Dies soll dazu dienen mehr Testdaten zu erzeugen oder wie unserem Fall dem Modell unterschiedliche Arten von Bildern zuzuführen. Die Bilder können teilweise in schlechten Verhältnissen aufgenommen werden, wie z.B. eine Verzerrung, weil die Linse nicht fokussiert ist. Damit wir testen können, ob eine Augmentation das Netz positiv beeinflusst, wurden fünf verschiedene Eigenschaften getestet, wovon zwei Eigenschaften in unterschiedlichen Eigenschaften getestet worden sind. Zu beachten ist, im Standardmodell sind diese Eigenschaften nicht aktiv.

Eigenschaft	Wert
rotation_range	45
rotation_range	135
width_shift_range	9
width_shift_range	15
horizontal_flip	True
vertical_flip	True
shear_range	9

Tabelle 2: Verwendete Eigenschaften Augmentation

### 5.3.1. Rotation\_range

Die Eigenschaft «rotation\_range» bewirkt, dass das Bild sich um einen gewissen Grad dreht. Diese Eigenschaft wurde ausgewählt, da alle Parkplätze in verschiedenen Winkeln aufgenommen werden. Die Kamera kann nicht für jeden Parkplatz gerade ausgerichtet werden. Damit das Netz sich auf verschieden gedrehte Bilder einstellen kann, wurden zwei verschiedene Bereiche ausgewählt. Zuerst 0-45 Grad und danach 0-135 Grad.



Abbildung 19: Beispiel rotation\_range = 45



Abbildung 18: Beispiel rotation\_range = 135

Während der Tests ist sehr schnell deutlich geworden, dass die Drehung der Bilder das Ergebnis verschlechtert um einen geringen Prozentsatz. Eine geringe Drehung kann mit dem Standardmodel mithalten, wobei eine starke Drehung das Ergebnis verschlechtert. Schaut man sich die Beispielbilder an, wird deutlich, dass die Bilder sich stärker verzerrten durch die größere Rotation. Dies könnte dazu führen, dass das Netz die Bilder nicht mehr vernünftig erkennen kann.

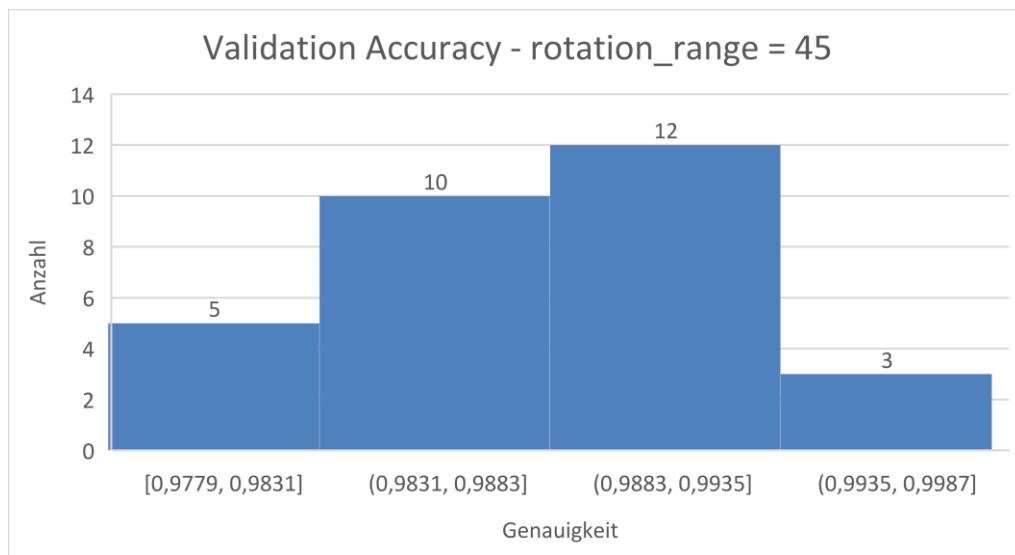


Abbildung 20: Validation Accuracy - rotation\_range = 45 - Histogram

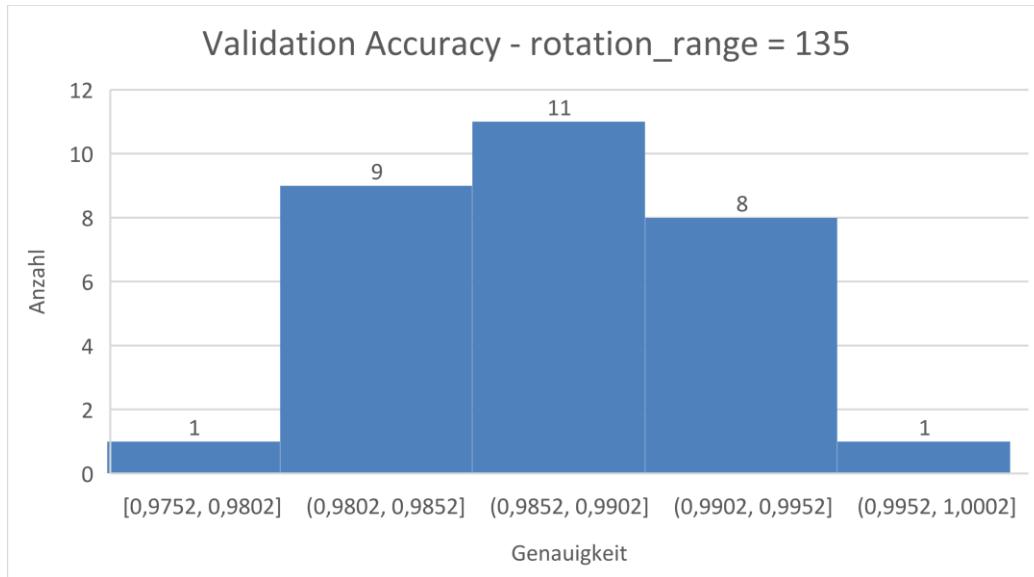


Abbildung 21: Validation Accuracy - rotation\_range = 135 - Histogram

Die Boxplots zeigen deutlich, in welchem Bereich die beiden Werte arbeiten. Der Durchschnitt liegt ca. 0,3% unter dem Standardmodell. Hierbei wird ersichtlich, dass eine Drehung der benutzten Daten für uns nicht sinnvoll ist. Sobald die rotation\_range hochgesetzt wird, verschlechterten sich die Ergebnisse auf allen Ebenen. Selbst die Ausreisser nach oben und unten werden schlechter.

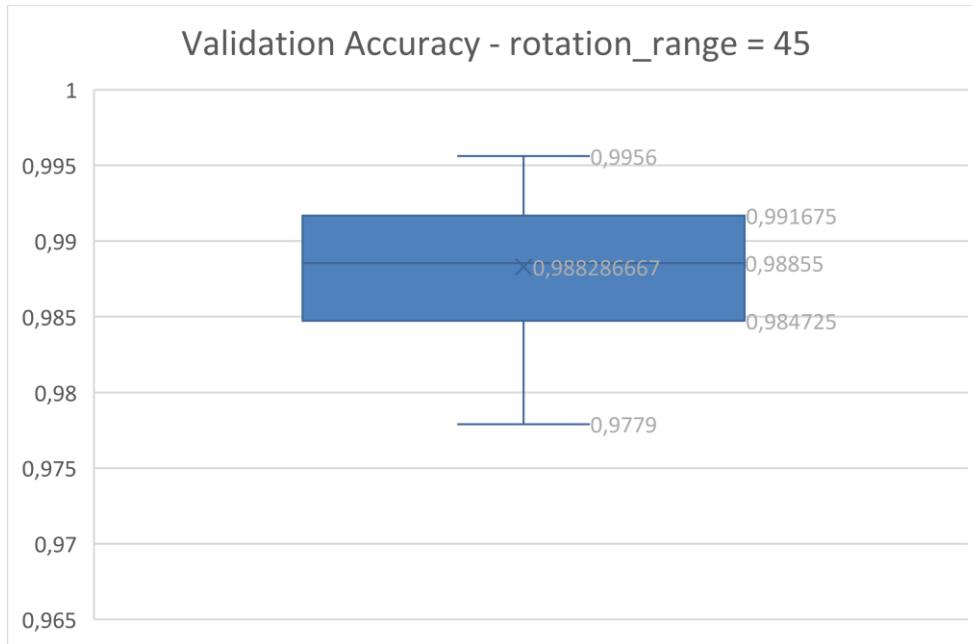


Abbildung 22: Validation Accuracy - rotation\_range = 45 - Boxplot

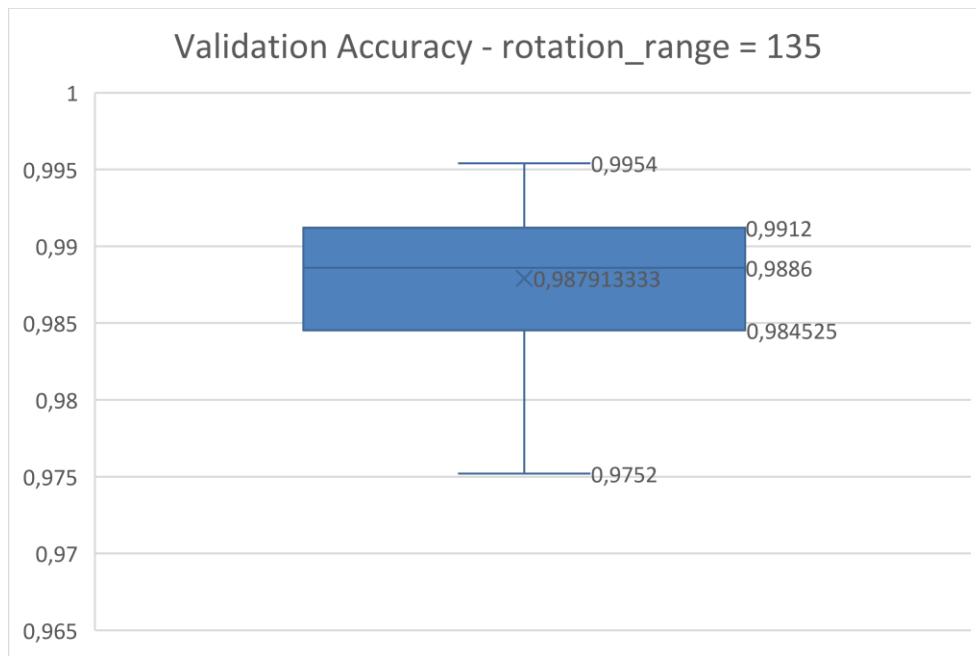


Abbildung 23: Validation Accuracy - rotation\_range = 135 - Boxplot

Somit ist die Eigenschaft «rotation-range» nicht sinnvoll für unser Model mit diesen Daten.

### 5.3.2. width\_shift\_range

Die Eigenschaft `width_shift_range` verschiebt das Bild um eine gewisse Anzahl von Pixeln nach links oder rechts. Da auch hiermit eine Verzerrung des Bildes simuliert wird, wurde diese Eigenschaft ausgewählt. Bei der Analyse der Ausgangsbilder wurde festgestellt, dass diverse Bilder eine leichte Verzerrung aufweisen.

Auch hier wurden zwei verschiedene Bereiche ausgewählt, um den Einfluss zu simulieren. Zuerst 0-9 und danach 0-15. Die Bereiche geben um wie viele Pixel die Bilder verschoben werden dürfen.



Abbildung 25: Beispiel `width_shift_range` = 9



Abbildung 24: Beispiel `width_shift_range` = 15

Bei der `width_shift_range` wird es schwieriger eine eindeutige Aussage zu treffen. Zuerst betrachten wir die Verschiebung um neun Pixel. Zuerst einmal wird deutlich, dass die Verschiebung sehr unterschiedlich ausfällt. In 16 von den 30 Fällen liegt sie unterhalb des Durchschnittswerts des Standardmodells. Gleichzeitig liegen aber auch 14 von 30 Fällen oberhalb des Ergebnisses. Daraus könnte man schließen, dass die Reihenfolge der Bilder wie diese trainiert werden eine Veränderung herbeiführen können. Die Reihenfolge wie die Bilder trainiert werden erfolgt zufällig.

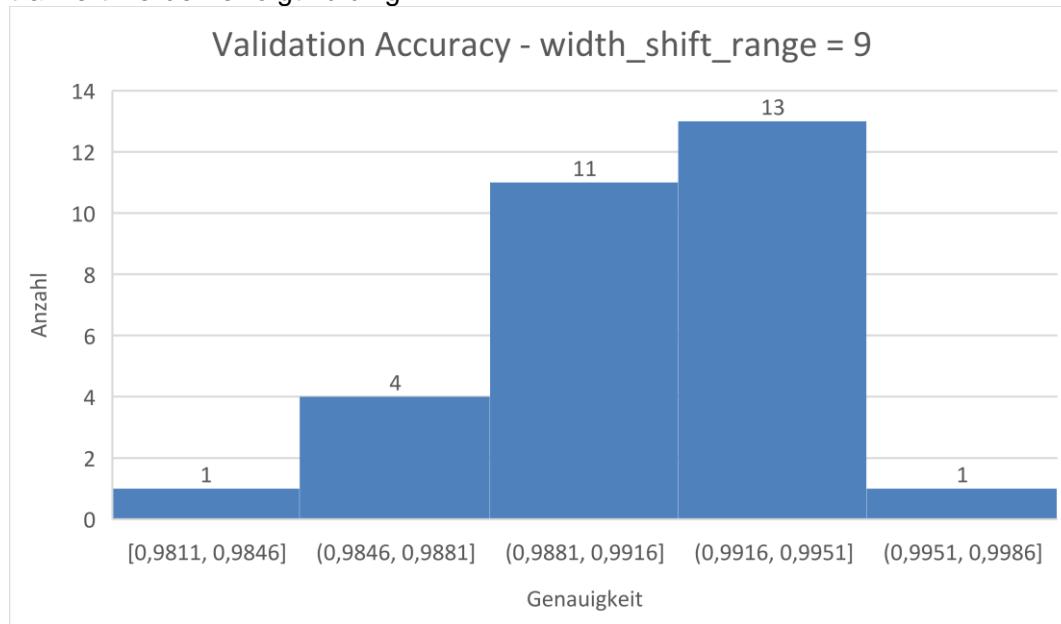


Abbildung 26: Validation Accuracy - `width_shift_range` = 9 - Histogram

Schaut man sich hingegen eine größere Verschiebung der Bilder an, wird auch hier deutlich, dass die Ergebnisse sehr kontrovers sind. Es scheint, dass die Bilder fast schon zufällig erkannt werden. Betrachtet man das Beispielbild entsteht eine starke Verzerrung des Bildes, sobald das Bild um 15 Pixel verschoben wird. Das Auto ist zu erkennen, aber es scheint, als könnte das Bild nicht komplett zugeordnet werden. Hierzu müsste eine tiefere Analyse der Ergebnisse durchgeführt werden.

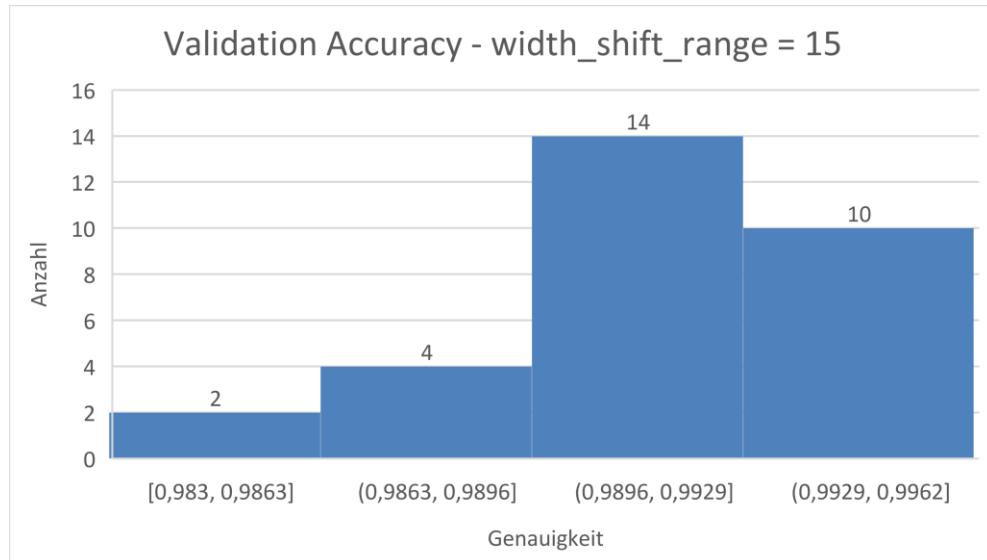


Abbildung 27: Validation Accuracy - width\_shift\_range = 15 - Histogram

Werden die Durchschnittswerte betrachtet, scheint eine größere Verzerrung das Netz sogar voranzubringen in der Erkennung. Daraus könnte man ableiten, dass eine Verzerrung für das Netz geeignet ist.

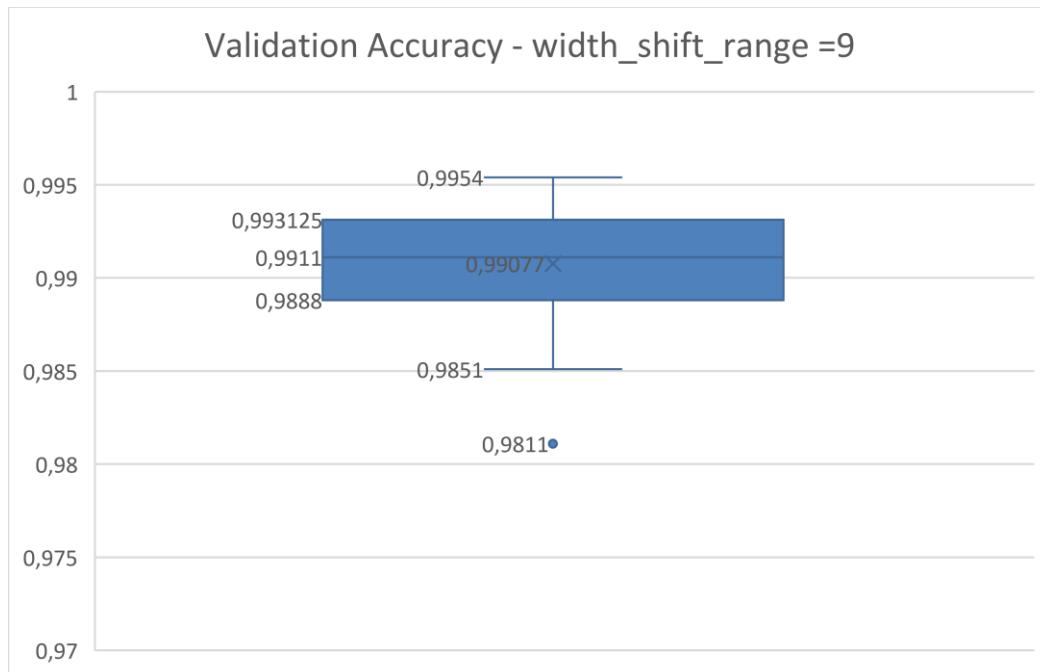


Abbildung 28: Validation Accuracy - width\_shift\_range = 9 - Boxplot

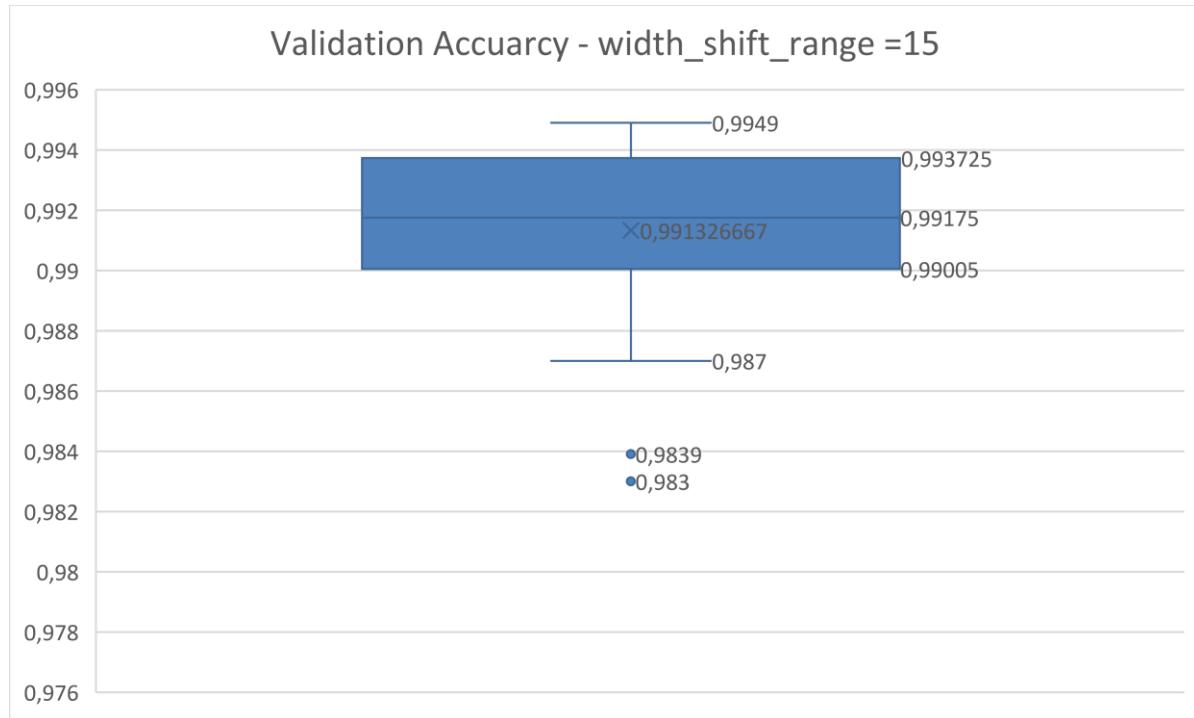


Abbildung 29: Validation Accuracy - width\_shift\_range = 15 - Boxplot

### 5.3.3. shear\_range

Die Eigenschaft Shear-Range bearbeitet das Bild mit einer sogenannte Scheren Optik. Hierzu wird die X-Achse festgesetzt und alles oberhalb der Achse nach rechts oder links in einer trapezähnlichen Form verschoben.

Aufgrund von fehlender Zeit wurde nur eine Größe getestet. Diese belief sich auf neun Pixel.

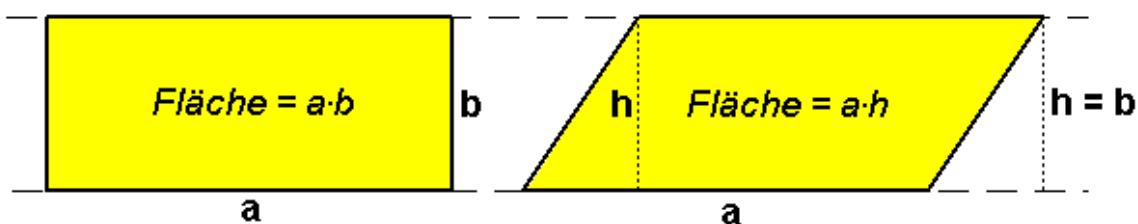


Abbildung 30: Geometriebeispiel shear-range

Wie in den vorrangingen Eigenschaften wurde diese Eigenschaft ausgewählt, da das Bild verzerrt wird und davon ausgegangen wird, dass nicht alle Bilder in einem perfekten Format bzw. Aufnahmestatus vorliegen.

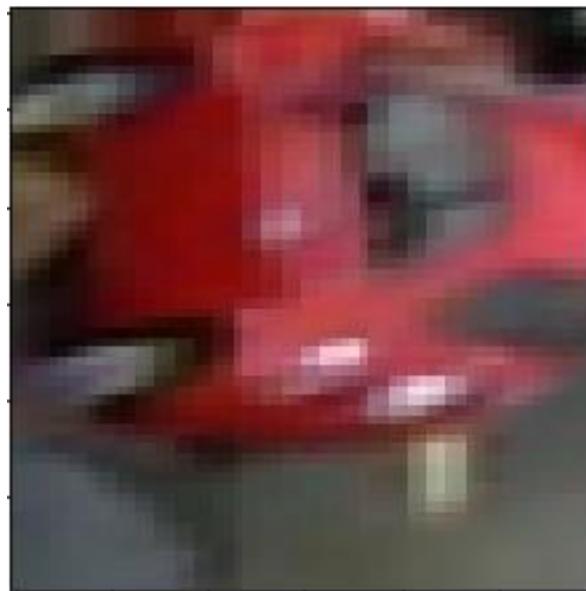


Abbildung 31: Beispiel shear-range

Bei der Betrachtung der Ergebnisse fällt direkt auf, dass die Genauigkeit allgemein sehr hoch ausfällt. Allein 21 von 30 Ergebnissen liegen über 99%. Wohingegen das niedrigste Ergebnis bei 98,71% liegt.

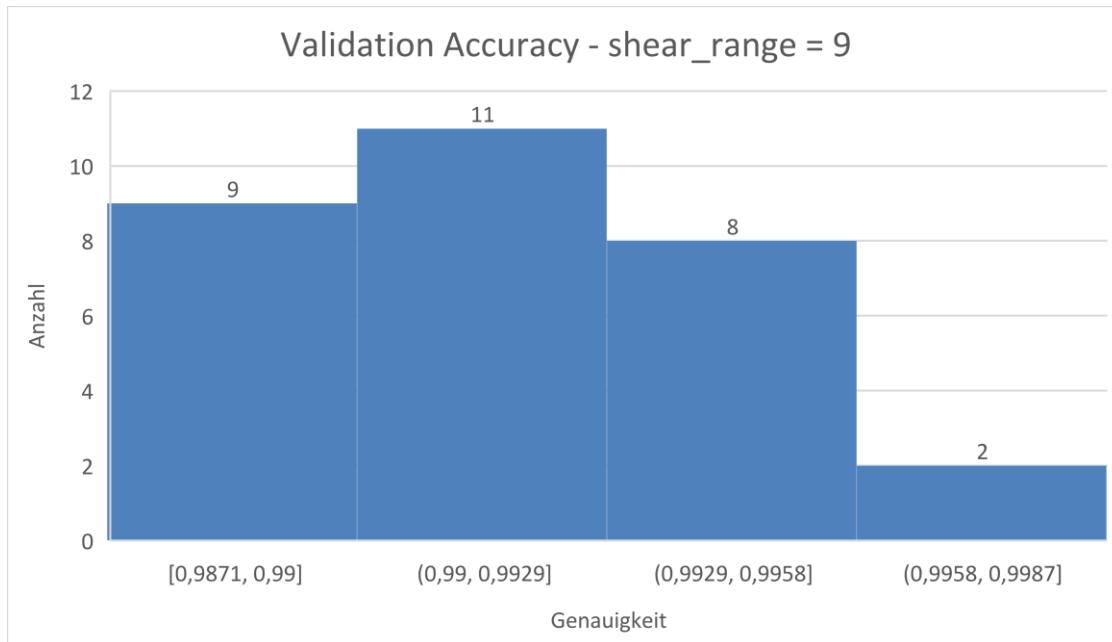


Abbildung 32: Validation Accuracy - shear\_range 9 - Histogram

Nehmen wir den Box-Plot der 30 Testergebnisse hinzu, kann direkt erkannt werden, dass die durchschnittliche Genauigkeit von 99,15% über dem Ergebnis des Standardmodels liegt. Darauf lässt sich ableiten, dass auch die Shear-Range ein Gewinn erzielen kann. Dieser fällt marginal aus, aber um die letzten Prozent herauszuholen wäre dies ein geeigneter Parameter.

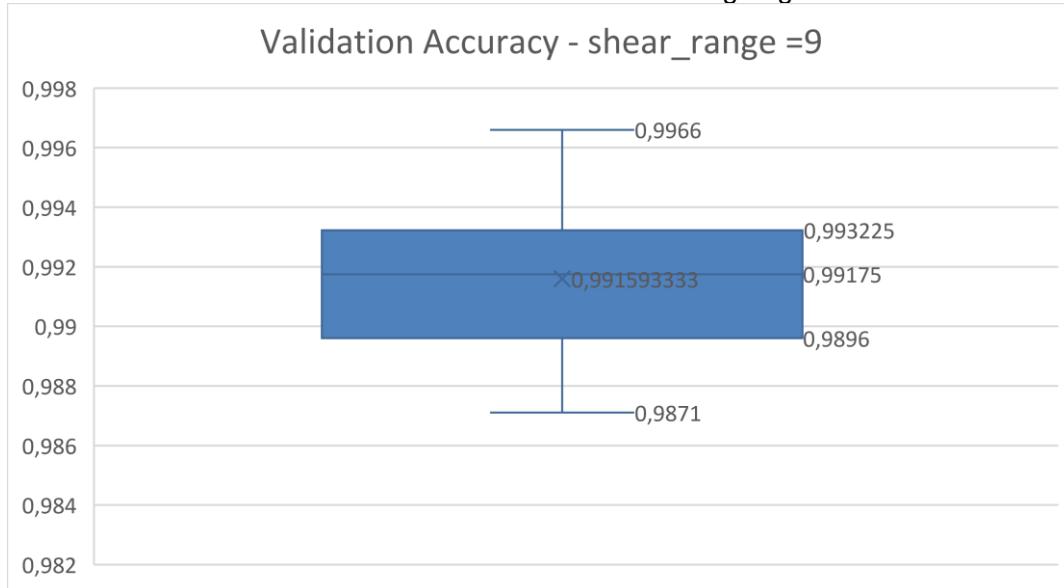


Abbildung 33: Validation Accuracy - shear\_range = 9 – Boxplot

Um die Grenzen der Shear-Range zu testen, sollten noch weitere Tests mit höheren Werten durchgeführt werden.

### 5.3.4. Flip Image

Als letzte Eigenschaft wurde das drehen von Bildern getestet. Dazu gibt es zwei Eigenschaften. Zuerst das komplette Drehen horizontal und danach vertikal. Dazu wird das ganze Bild an der horizontalen oder vertikalen Linie gespiegelt. Diese Eigenschaften wurden ausgewählt, da jegliche Parkplätze in unterschiedlichen Lagen aufgenommen werden können.



Abbildung 35: Beispiel horizontal\_flip = true



Abbildung 34: Beispiel vertical\_flip = true

Sobald die Werte des Histogramms betrachtet werden, fällt auf das sowohl beim horizontal\_flip als auch der vertical\_flip die Genauigkeit gegenüber dem Standardmodel nachlässt. Die Konzentration ist vor allem im Bereich knapp unter 99%. Dies ist dahingehend erstaunlich, da das Drehen das Bild nicht verzerrt und somit das Bild vom Netz «normal» erkannt werden sollte. Das Netz sollte jede glich die Möglichkeit haben unterschiedliche Situationen genauer erkennen zu können.

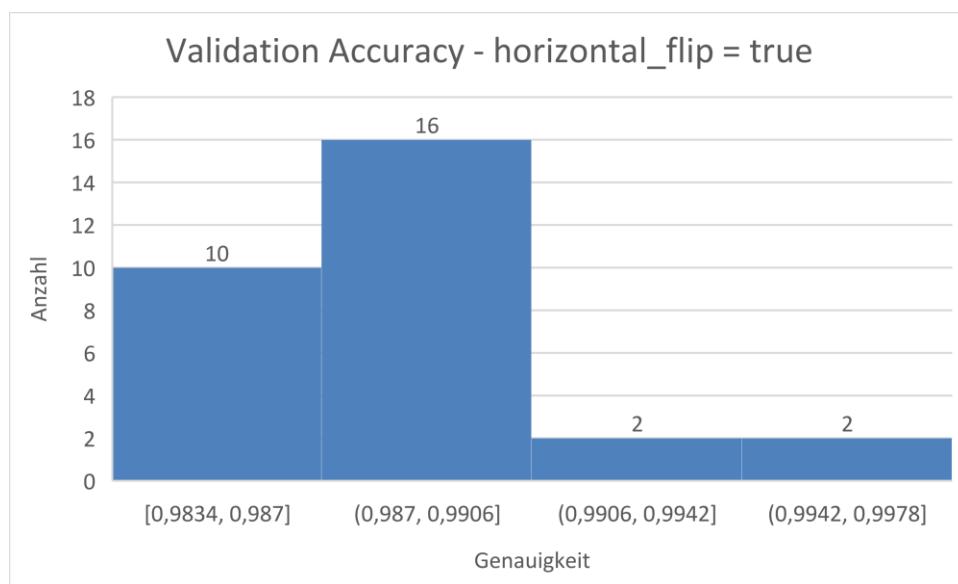
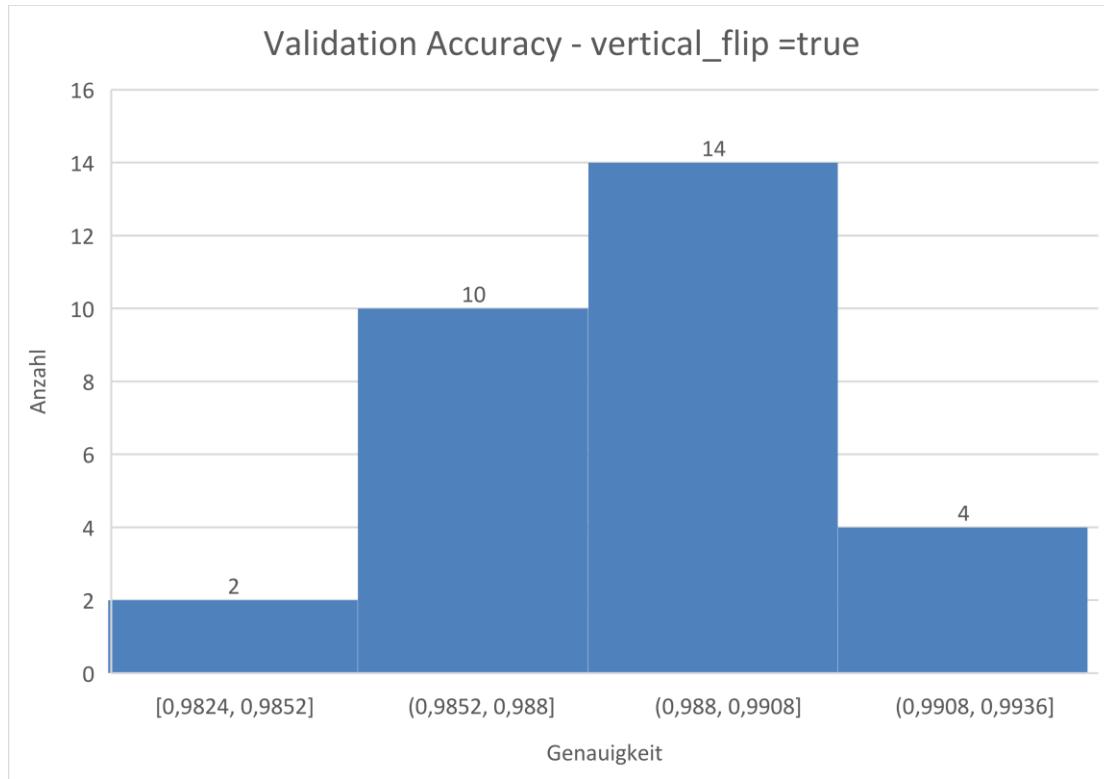
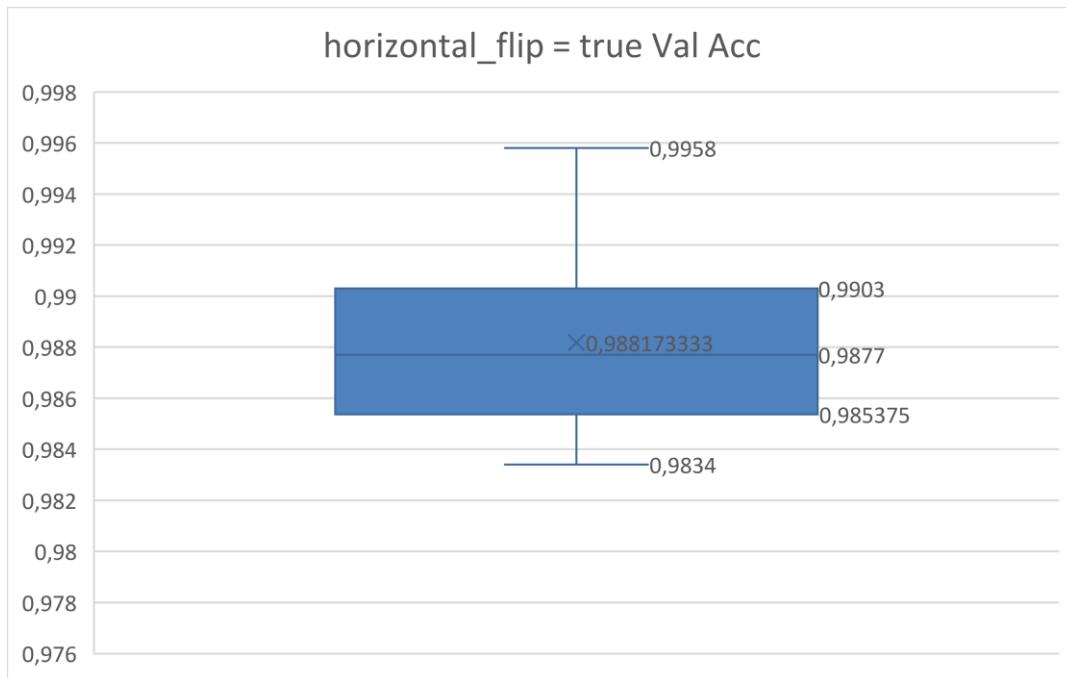


Abbildung 36: Validation Accuracy - horizontal\_flip = true - Histogram

Abbildung 37: Validation Accuracy - `vertical_flip = true` - Histogram

Werden die Box-Plots der Eigenschaften betrachtet, fällt auch hier auf, dass die Durchschnittswerte deutlich unter dem Standardmodell liegen. Dies führt zu der Annahme das diese Eigenschaften nicht für unser Model geeignet sind.

Abbildung 38: Validation Accuracy - `horizontal_flip = true` - Boxplot

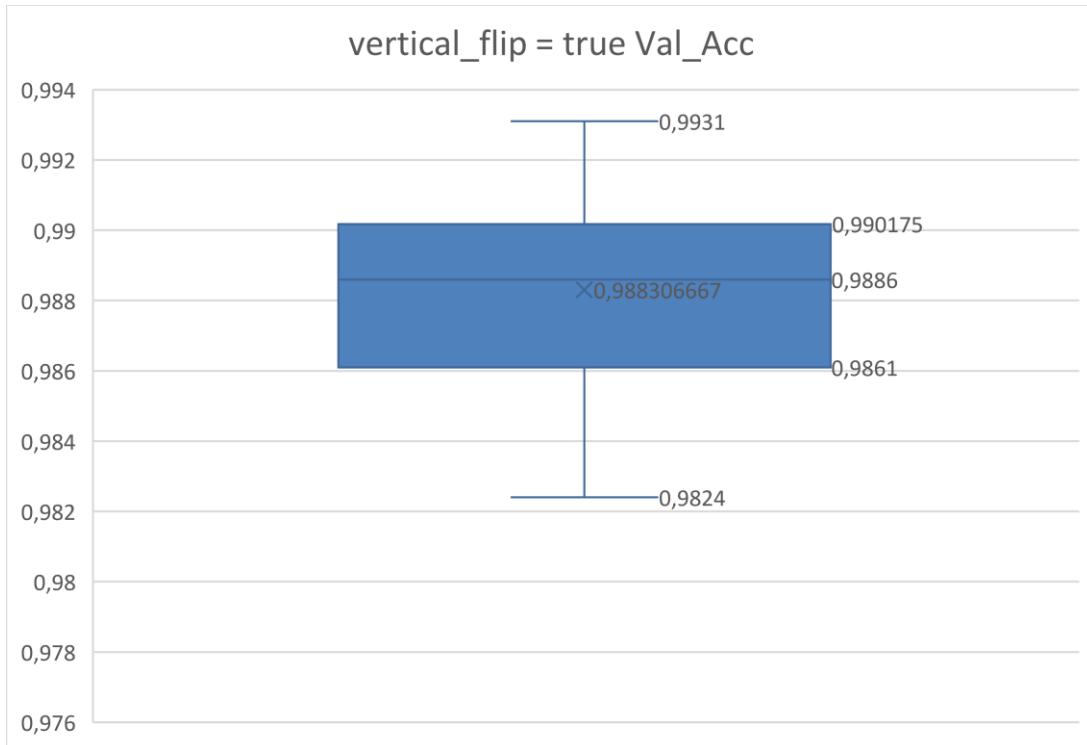


Abbildung 39: Validation Accuracy - vertical\_flip = true – Boxplot

Da diese Annahme für uns zurzeit noch nicht ganz verständlich ist, müssten auch hier weitere Tests durchgeführt werden. Es sollte geschaut werden, ob eine Veränderung am Netz notwendig ist, um eine höhere Genauigkeit zu erzielen bzw. es sollte erörtert werden woran es liegen kann, dass das CNN die Bilder um ca. 0,3-0,5% schlechter erkennt.

## 5.4. Dropout

Der Dropout ist eine effektive und oft genutzte Technik zur Vermeidung von Overfitting. In unserem Model kann der Dropout hinter jeden Max-Pooling Layer gesetzt werden. Dabei werden durch den Dropout zufällig einige Features des letzten Layers auf 0 gesetzt. Die Dropout-Rate gibt an, wie viele Features auf 0 gesetzt werden. Im Bereich des Machine Learnings haben sich hier Werte zwischen 20 und 50% (0.2-0.5) als praktikabel erwiesen.

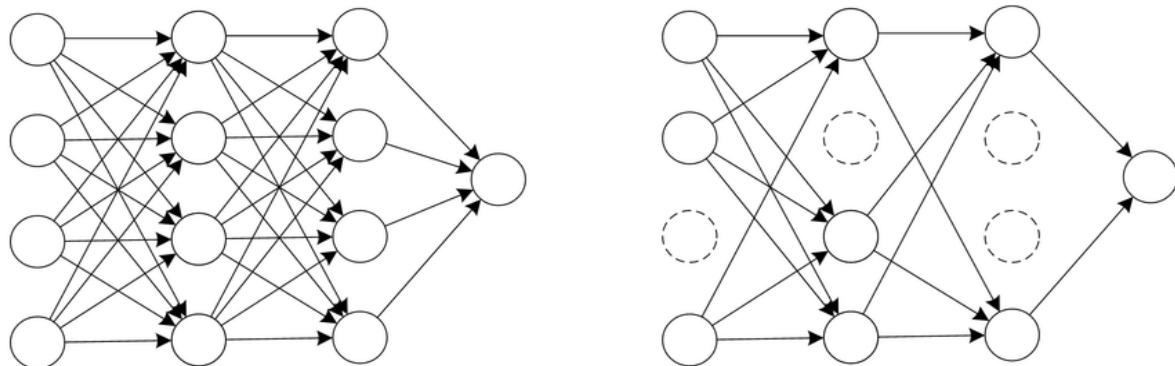


Abbildung 40: Dropout im neuronalen Netz

Wie der obigen Abbildung zu entnehmen ist, bedeutet dies, dass diese Features keinen Einfluss auf das aktuelle Trainieren haben. Dadurch ist eine Variabilität in den Daten gegeben und es werden ähnliche Muster vermieden. Da unsere Bilder der einzelnen Parkplätze recht ähnlich sind und das Model zu Overfitting neigt, müsste als Vermutung der Dropout also sinnvoll sein. Bei den folgenden Tests wurde nun der Dropout zwischen 0 und 0.5 analysiert und mit dem Standardmodel (Dropout=0.4) verglichen.

### 5.4.1. Dropout von 0

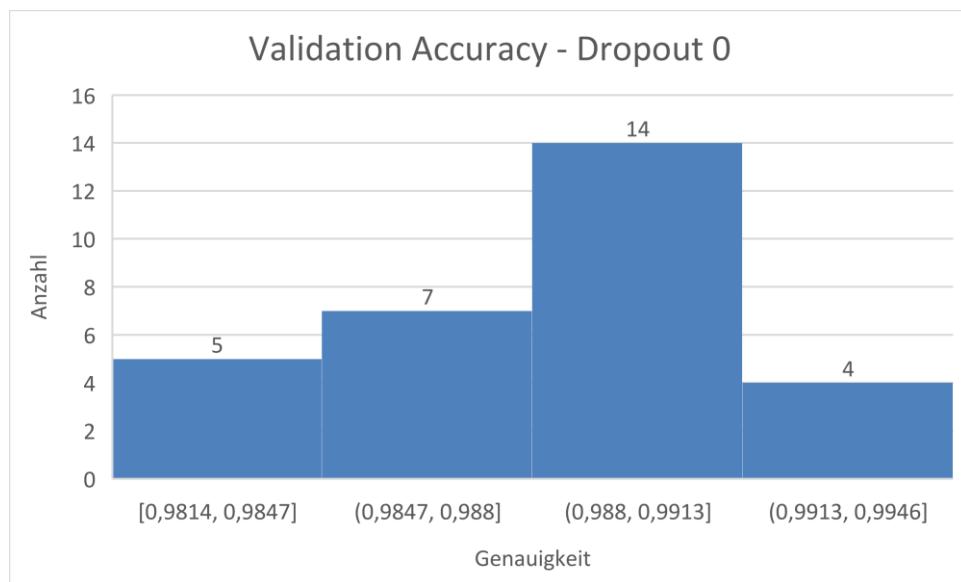


Abbildung 41: Validation Accuracy - Dropout 0 - Histogram

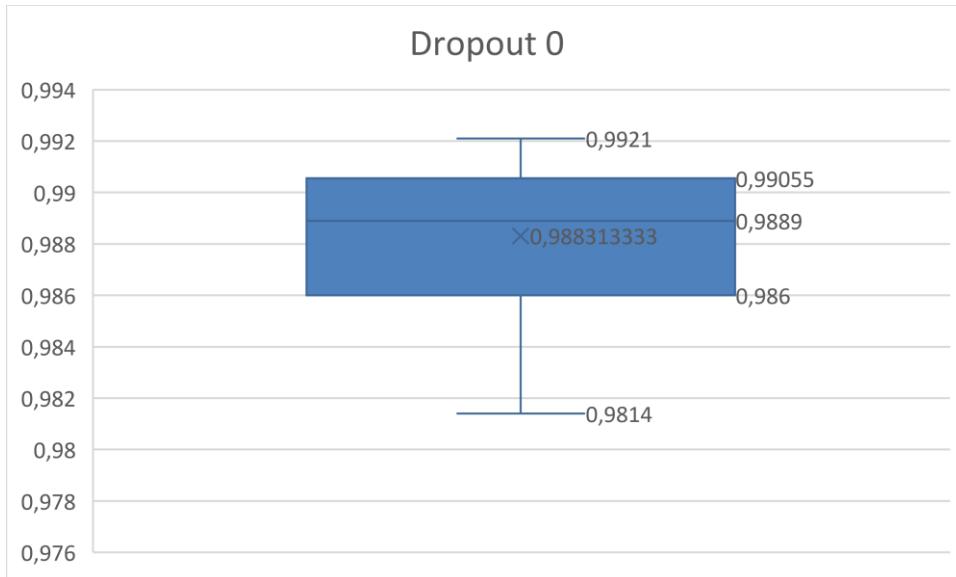


Abbildung 42: Validation Accuracy - Dropout 0 - Boxplot

Aus der Genauigkeit der Daten lässt sich bei einem Dropout von 0 erkennen, dass die Genauigkeit im Durchschnitt bei ca. 98,83 % und somit unter dem Standardmodell mit einer Genauigkeit von ca. 99,11 % liegt. Die Genauigkeiten fallen auch relativ unterschiedlich aus. Der größte Teil (nahezu die Hälfte) befindet sich in einer Range von 98,8 bis 99,13 %. Der restliche Teil ist jedoch unterschiedlich gewichtet und lässt so kein klares Bild erkennen. Die Vermutung der geringeren Genauigkeit scheint zunächst bestätigt zu werden, auch wenn der Unterschied marginal ausfällt. Es müssen also weitere Tests durchgeführt werden.

#### 5.4.2. Dropout von 0.2

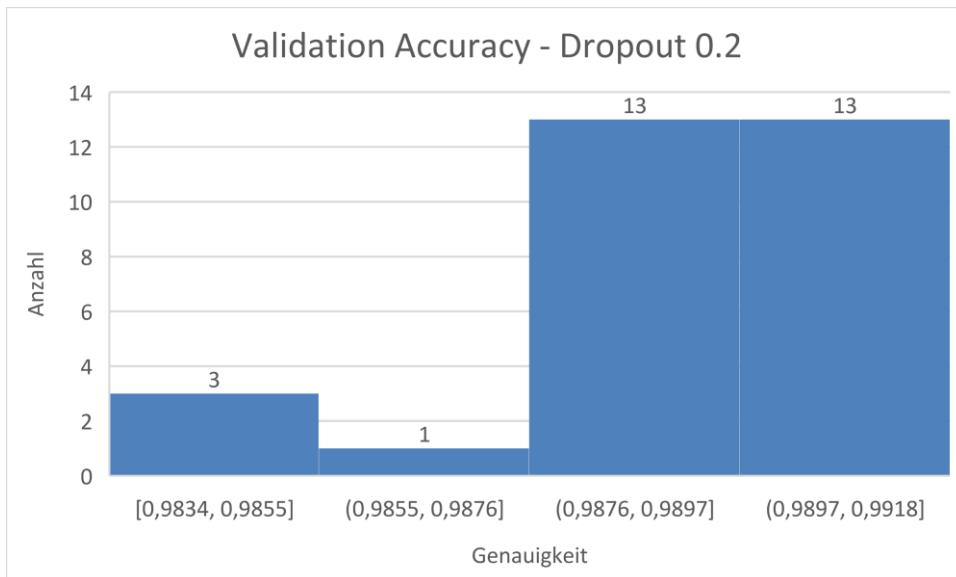


Abbildung 43: Validation Accuracy - Dropout 0.2 - Histogram

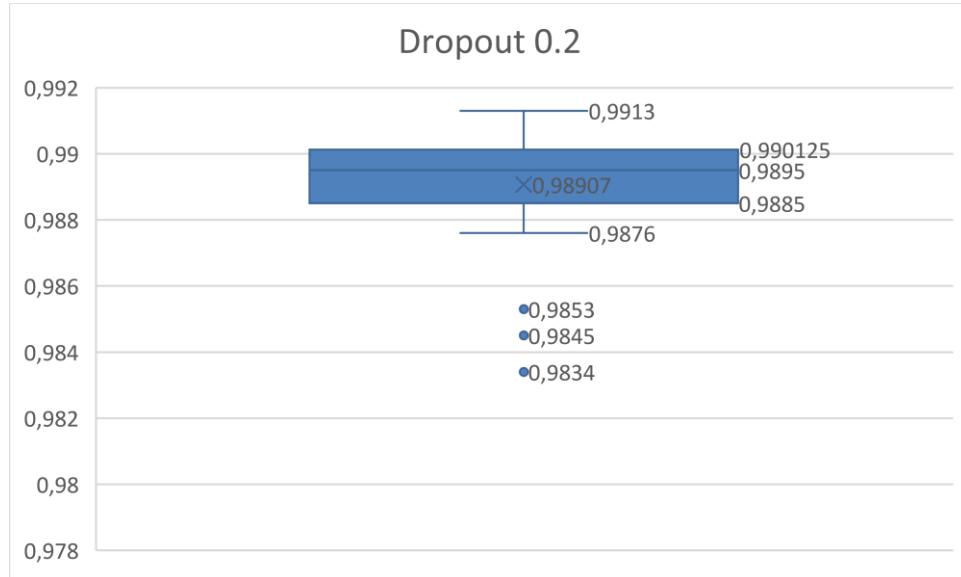


Abbildung 44: Validation Accuracy - Dropout 0.2 - Boxplot

Bei einem Dropout von 0,2 ist eine kleine Steigerung der durchschnittlichen Genauigkeit zu erkennen. Diese beträgt hier immerhin schon ca. 98,91 %. Auffällig ist auch, dass sich 26 der 30 gemessenen Genauigkeiten zwischen 98,76 und 99,18 % befinden. Dies ist ein relativ kleiner Bereich, auf den sich konzentriert wird. Es gibt lediglich ein paar Ausreißer nach unten. Um eine noch bessere Genauigkeit zu erzielen, wird jetzt noch ein Dropout von 0,5 getestet.

#### 5.4.3. Dropout von 0.5

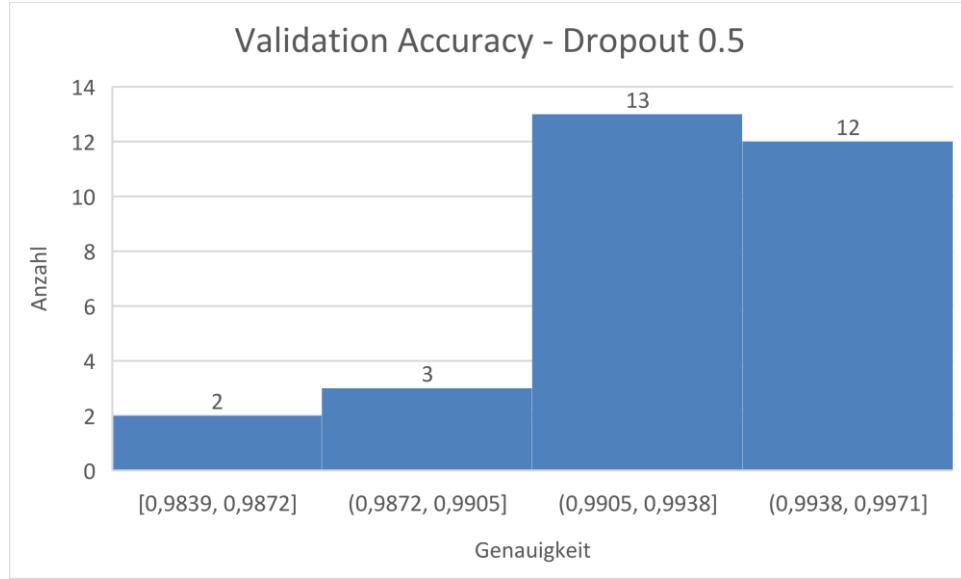


Abbildung 45: Validation Accuracy - Dropout 0.5 - Histogram

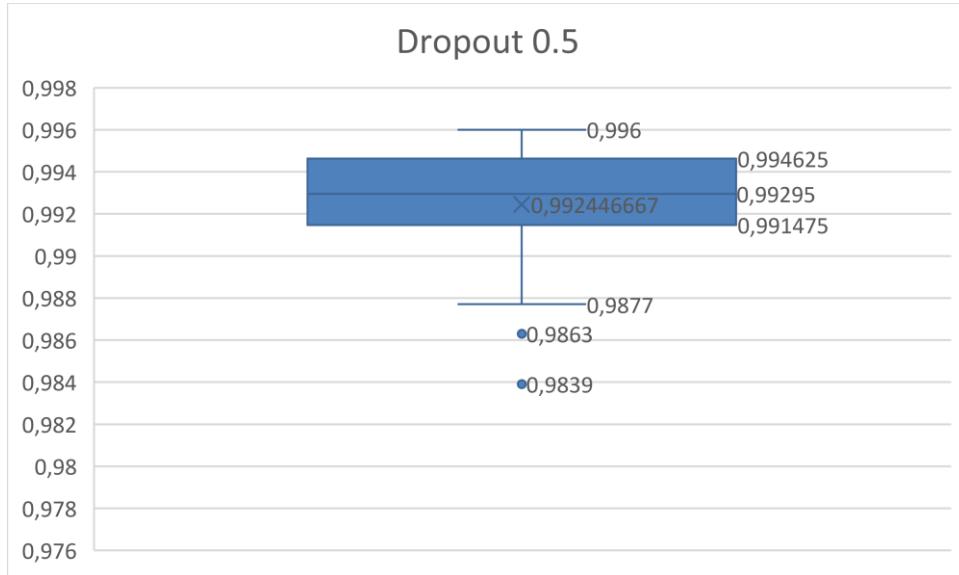


Abbildung 46: Validation Accuracy - Dropout 0.5 - Boxplot

Bei dem Dropout von 0,5 ist nochmals eine deutliche Steigerung zu erkennen. Auch hier sind die meisten Genauigkeiten (25) mit einer Range von 99,05 bis 99,71 % sehr dicht besiedelt. Auffällig ist vor allem, dass sich der höchste Wert bei einer Genauigkeit von 99,71 % befindet und somit sehr hoch ist.

Aus den Tests bezüglich des Dropouts ging schließlich hervor, dass dieser sinnvoll ist zu implementieren. Der in dem Standardmodel gewählte Wert von 0,4 kann bedenkenlos auf 0,5 erhöht werden.

## 5.5. Input Size

Ziel dieser Tests war es herauszufinden, ob die Input Size der Bilder, mit welcher die KI trainiert, eine Auswirkung auf die Accuracy der KI hat.

Die ersten Tests dauerten noch sehr lange, da wir sehr viele Bilder genutzt haben. Deshalb wurden nicht so viele verschiedene Input Sizes getestet. Die getesteten Werte wurden zufällig aus einer Liste von verschiedenen Bildgrößen entnommen. Diese Liste ist durch ein Python Skript entstanden, welches ausgewertet hat, welche verschiedenen Bildgrößen wir in unseren Ausgangsdaten haben. Nach diesen Tests kam es zu noch keinen Nennenswerten Ergebnis.

Die einzige Annahme, die man nach diesen Tests haben kann, ist dass es wahrscheinlich ist, dass die Inputgrößen die Genauigkeit beeinflusst. Um mehr Testergebnisse zu erzielen, haben wir nun deutlich weniger Bilder zum Trainieren genommen.

Mit mehr Ergebnissen hofften wir, dass wir herausfinden, wie die Height und Width geändert werden müssen, um ein möglichst genaues Ergebnis zu erzielen. Leider konnte man mit den Ergebnissen nicht herausfinden, wie man Height und Width verändern muss, da kein Muster zu erkennen ist, wie die Accuracy sich im Verhältnis zur Height und Width ändert. Bei den Letzten Testdurchläufen wurden verschiedene Tests mehrmals hintereinander durchgeführt, um mithilfe des Mittelwertes ein möglichst genaues Ergebnis zu erhalten.

Das Ergebnis hierbei wäre, dass eine Höhe von 51p und eine Breite von 42p zum besten Ergebnis führt.

Allerdings sind diese Werte kaum merklich besser als die anderen, weshalb es sehr wahrscheinlich ist, dass die Input Size nahezu gar keine Auswirkung auf die Accuracy hat.

Beim Betrachten der Boxplots fällt auch auf, dass auch dieser Test die geringste Streuung hat.

### 5.5.1. Auflösung 124x60

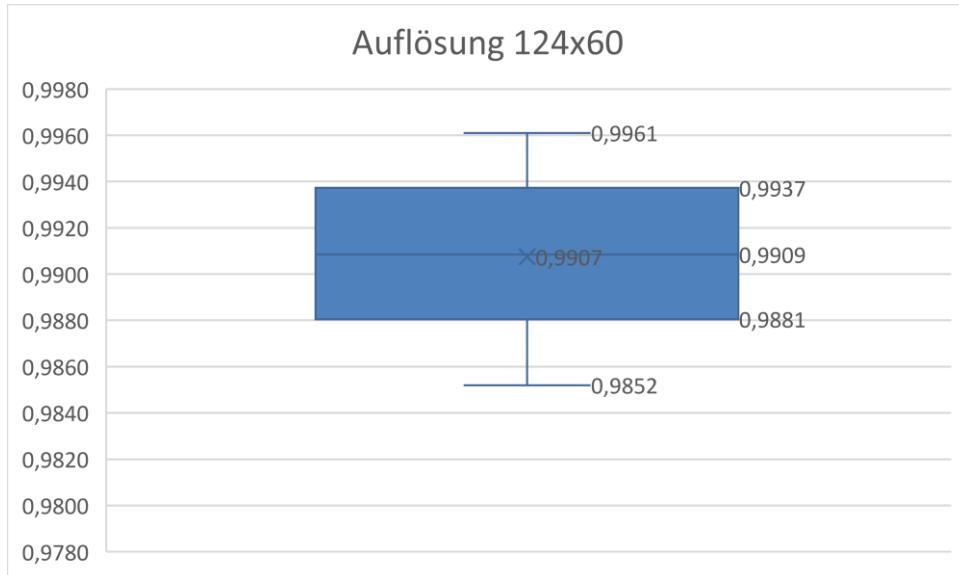


Abbildung 47: Validation Accuracy - Auflösung 124x60 - Boxplot

### 5.5.2. Auflösung 60x54

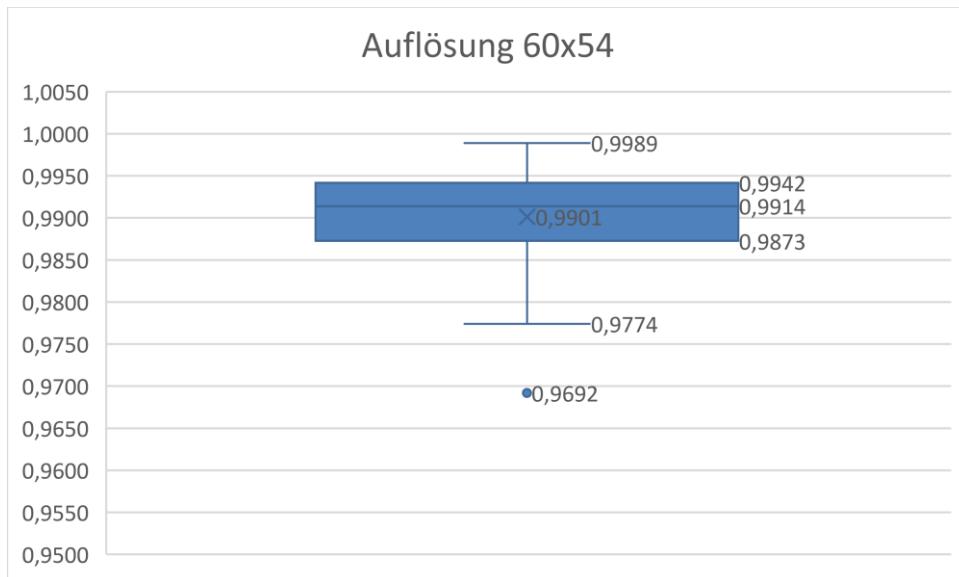


Abbildung 48: Validation Accuracy - Auflösung 60x54 - Boxplot

### 5.5.3. Auflösung 51x42

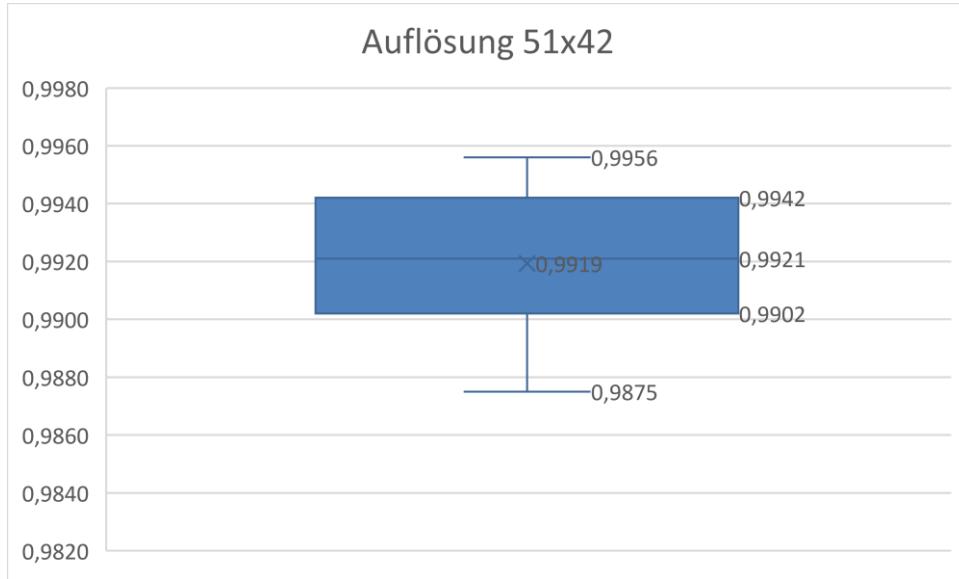


Abbildung 49: Validation Accuracy - Auflösung 51x42 - Boxplot

Es ist somit fast egal, welche Inputgröße genommen wird. Anscheinend hat sie einen Einfluss auf das Ergebnis, dieses ist jedoch so gering, dass es nicht beachtet werden muss.

## 5.6. BatchNormalization und Regularizer

Referenzwert: Als Referenzwert für die Testergebnisse habe ich das CNN ohne Batch Normalisation durchlaufen lassen. Dabei gab es zwei Fälle:

### 5.6.1. Ohne BatchNormalization

Das CNN mit „use\_bias=False“ hat eine durchschnittliche VAL\_ACC von 0,9871. Die Werte streuen zwischen 0,9791 und 0,9924 und sehen ansatzweise Normalverteilt aus, mit einem Maximum bei 0,9871-0,9881. Im Vergleich mit den anderen Testreihen ist diese Konfiguration eher schlechter.

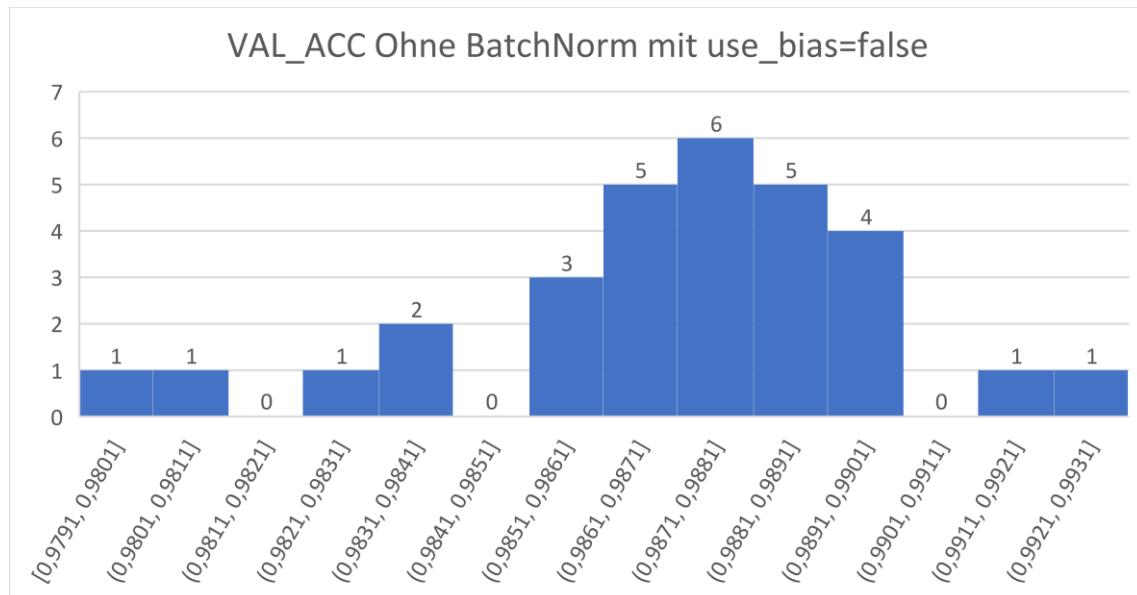


Abbildung 50: Validation Accuracy - Ohne BatchNorm - Histogram

### 5.6.2. Ohne BatchNormalization/mit Bias

CNN mit „use\_bias=True“ hat eine durchschnittliche VAL\_ACC von 0,9912. Die Werte streuen zwischen 0,9848 und 0,9951 und sehen ansatzweise Normalverteilt aus, mit einem Maximum bei 0,9908-0,9918. Im Vergleich mit den anderen Testreihen hat diese Konfiguration das beste Durchschnittsergebnis erzielt.

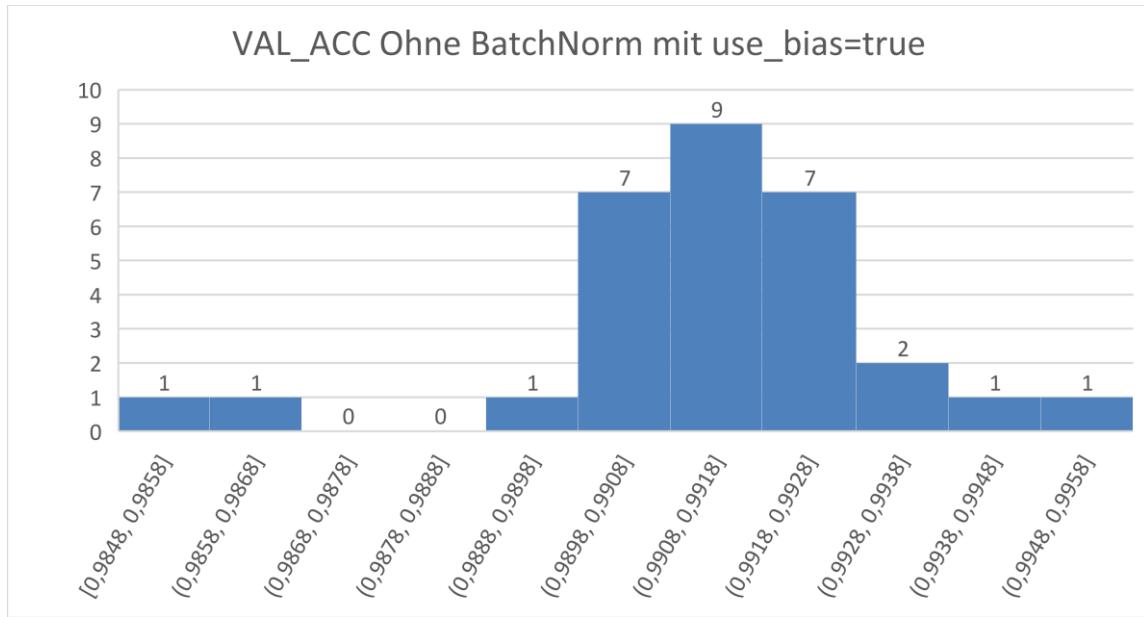


Abbildung 51: Validation Accuracy - Ohne BatchNorm/mit Bias - Histogram

### 5.6.3. Mit BatchNormalization

BatchNorm: Für die BatchNormalization wurde nach jedem Layer eine Batch Normalisierung eingefügt. Das Ergebnis sieht wie folgt aus:

CNN mit „BatchNorm für alle Layer“ hat eine durchschnittliche VAL\_ACC von 0,9901. Die Werte streuen zwischen 0,9250 und 0,9992 und sehen, bis auf einen Ausreißer, ansatzweise Normalverteilt aus, mit einem Maximum bei 0,9925-0,9950. Im Vergleich mit den anderen Testreihen hat diese Konfiguration das zweitbeste Durchschnittsergebnis erzielt.

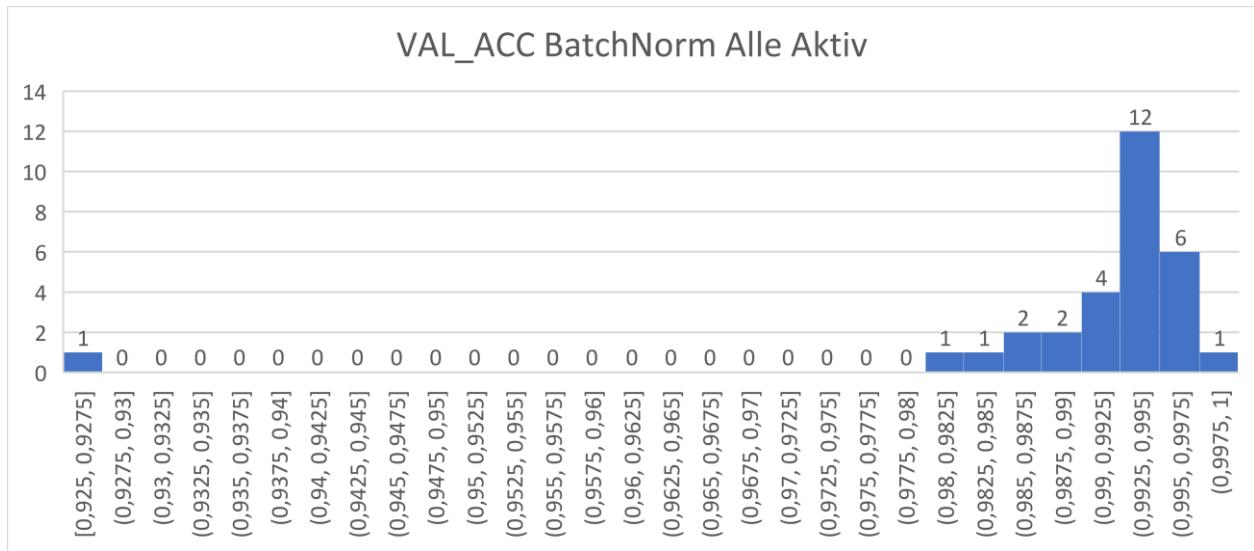


Abbildung 52: Validation Accuracy - Mit BatchNorm - Histogram

#### 5.6.4. Kernel und Activity Regularizer

Das CNN mit „kernel\_regularizer und activity\_regularizer“ hat eine durchschnittliche VAL\_ACC von 0,8961. Die Werte streuten zwischen 0,5962 und 0,9772 und sehen nicht wirklich Normalverteilt aus, mit einem Maximum bei 0,9462-0,9712. Im Vergleich mit den anderen Testreihen ist diese Konfiguration die Schlechteste.

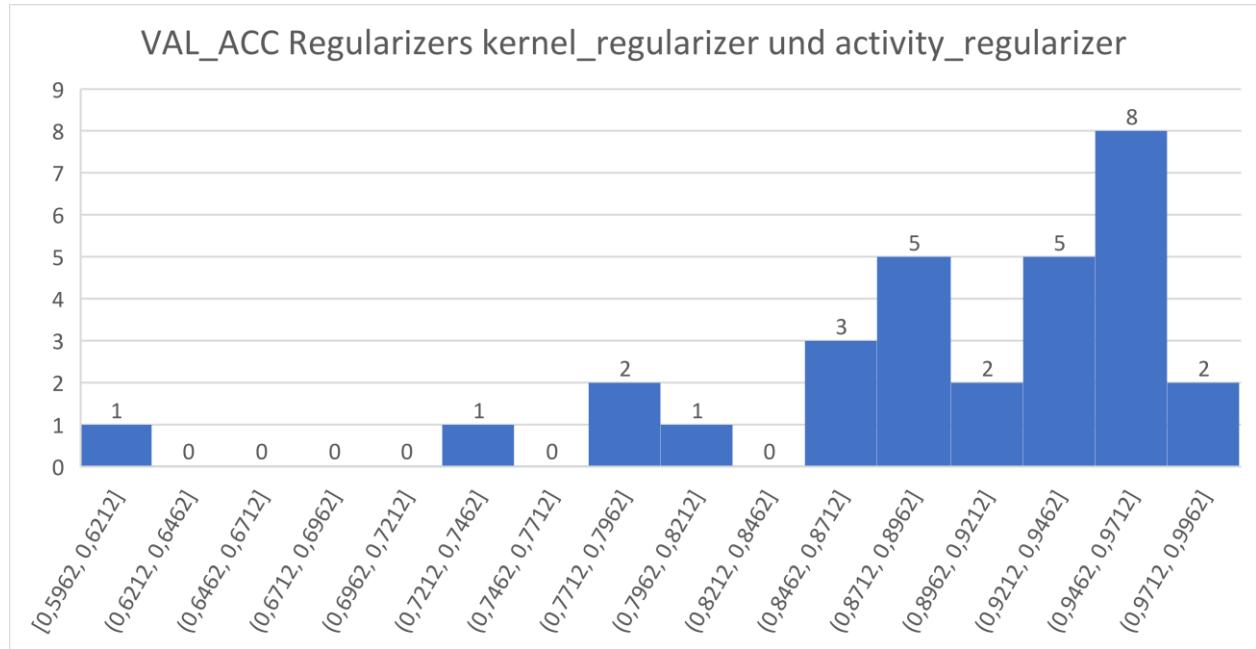


Abbildung 53: Validation Accuracy - Kernel/Activity Regularizer - Histogram

### 5.6.5. Kernel Regularizer

CNN mit „kernel\_regularizer“ hat eine durchschnittliche VAL\_ACC von 0,9883. Die Werte streuen zwischen 0,9767 und 0,9964 und sehen nicht wirklich Normalverteilt aus, ohne einem erkennbaren Maximum. Im Vergleich mit den anderen Testreihen ist diese Konfiguration eher schlechter.

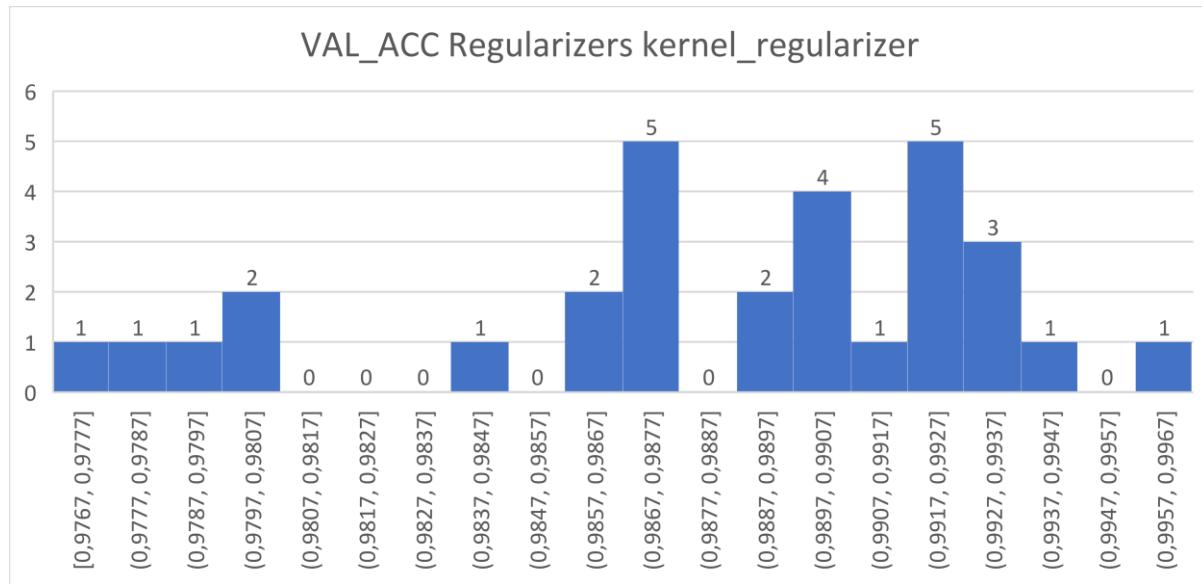


Abbildung 54: Validation Accuracy - Kernel Regularizer - Histogram

Ergebnis: Wir kommen zu dem Ergebnis, dass die Verwendung von Regularizern in unserem CNN nicht sinnvoll ist. Bei der BatchNormalization ist es deutlich knapper und wir würden die Verwendung als sinnvoll ansehen.

## 6. Ausblick

Dieses Projekt diente für uns persönlich dazu, erste Erkenntnisse im Bereich des Maschinellen Lernens zu erlangen. Jeder Gruppenteilnehmer verfügt nun über bestimmte Grundkenntnisse in dieser Thematik.

Das Projekt könnte nun auf vielen unterschiedlichen Arten fortgeführt werden. Zum einen lassen die Tests extrem großen Spielraum. Viele mögliche Parameter wurden aus zeitlich Gründen nicht getestet. Auch unterschiedliche Kombinationen der Tests konnten so auch nicht durchgeführt werden. Die Auswahl der Datenpakete innerhalb des Datensets erfolgte auch zufällig. Hier könnten auch nochmals andere Datenpakete zum Einsatz kommen. Außerdem können die Datensätze noch weiter verfeinert werden. Mit dem gefundenen zweiten Datensatz ist es möglich, das Model noch besser zu trainieren.

Die Daten sind schon recht gut, da sie unter anderem auch unterschiedliche Lichtverhältnisse mit einbeziehen. Hier könnten aber auch noch unterschiedliche Faktoren mit einbezogen werden. Zum Beispiel gibt es keine Bilder zu möglichem Schneefall, aber auch eine verschmierte Linse der Kamera wird nicht in Betracht gezogen.

Dies und weitere Einflüsse sind für ein robusteres Model noch zu implementieren. Für unsere Experimente sollte dies jedoch noch nicht zu große Bedeutung haben.

Zum anderen können die gewonnenen Erkenntnisse für zukünftige Problemstellungen, die eine ähnliche Aufgabe bewältigen müssen, verwendet werden. Da das eigentliche Projekt der Parkplatzerkennung noch nicht durchgeführt wurde, könnten diese gesammelten Informationen wichtig für ein zukünftiges Projekt sein. Dabei kann in Betracht gezogen werden, ein vortrainiertes Model zu nutzen, um ein besseres Ergebnis zu erzielen.

Da in unserem Projekt auch keine Fremddaten von einem zweiten Datensatz zum Testen des neuronalen Netzes genutzt wurden, müsste in Zukunft zum Beispiel mit dem zweiten vorhandenen Datensatz getestet werden. Natürlich fehlt hier nun auch noch die Anbindung an einen Raspberry Pi und die Entwicklung einer App, um das Projekt zu vervollständigen. Dennoch bieten die Erkenntnisse dieser Experimente eine erste Grundlage für den Einfluss verschiedener Parameter auf unsere Problemstellung der Parkplatzerkennung.

## 7. Abbildungsverzeichnis

Abbildung 1: Produkt-Zyklus .....	3
Abbildung 2: Convolution mit Filter .....	5
Abbildung 3: Max-Pooling .....	6
Abbildung 4: Übersicht über ein CNN .....	6
Abbildung 5: Datenfehler .....	7
Abbildung 6: Vergleich pyfastcopy zu Standart .....	8
Abbildung 7: GröÙe Parklinglot Daten.....	9
Abbildung 8: linear unit function.....	11
Abbildung 9: Sigmoid-Funktion .....	11
Abbildung 10: Verwendetes Netz mit Layern.....	12
Abbildung 11: Ordnerstruktur Daten.....	14
Abbildung 12: Befehle zum Herunterladen und Entpacken der Daten .....	15
Abbildung 13: Ordnerstruktur im Code .....	15
Abbildung 14: Verwendete Technologie .....	16
Abbildung 15: Early-Stopping .....	17
Abbildung 16: Validation Accuracy- Standardmodel - Histogram.....	17
Abbildung 17: Validation Accuracy - Standardmodel - Boxplot.....	18
Abbildung 18: Beispiel rotation_range = 135.....	19
Abbildung 19: Beispiel rotation_range = 45 .....	19
Abbildung 20: Validation Accuracy - rotation_range = 45 - Histogram.....	19
Abbildung 21: Validation Accuracy - rotation_range = 135 - Histogram.....	20
Abbildung 22: Validation Accuracy - rotation_range = 45 - Boxplot .....	20
Abbildung 23: Validation Accuracy - rotation_range = 135 - Boxplot .....	21
Abbildung 24: Beispiel width_shift_range = 15 .....	22
Abbildung 25: Beispiel width_shift_range = 9 .....	22
Abbildung 26: Validation Accuracy - width_shift_range = 9 - Histogram .....	22
Abbildung 27: Validation Accuracy - width_shift_range = 15 - Histogram .....	23
Abbildung 28: Validation Accuracy - width_shift_range = 9 - Boxplot .....	23
Abbildung 29: Validation Accuarcy - width_shift_range = 15 - Boxplot .....	24
Abbildung 30: Geometriebespiel shear-range .....	25
Abbildung 31: Beispiel shear-range .....	25
Abbildung 32: Validation Accuracy - shear_range 9 - Histogram .....	26
Abbildung 33: Validation Accuracy - shear_range = 9 – Boxplot .....	26
Abbildung 34: Beispiel vertical_flip =true .....	27
Abbildung 35: Beispiel horizontal_flip = true .....	27
Abbildung 36: Validation Accuracy - horizontal_flip = true - Histogram.....	27
Abbildung 37: Validation Accuracy - vertical_flip = true - Histogram.....	28
Abbildung 38: Validation Accuracy - horizontal_flip = true - Boxplot .....	28
Abbildung 39: Validation Accuracy - vertical_flip = true – Boxplot .....	29
Abbildung 40: Dropout im neuronalen Netz.....	30
Abbildung 41: Validation Accuracy - Dropout 0 - Histogram .....	30
Abbildung 42: Validation Accuracy - Dropout 0 - Boxplot .....	31
Abbildung 43: Validation Accuracy - Dropout 0.2 - Histogram .....	31
Abbildung 44: Validation Accuracy - Dropout 0.2 - Boxplot .....	32
Abbildung 45: Validation Accuracy - Dropout 0.5 - Histogram .....	32
Abbildung 46: Validation Accuracy - Dropout 0.5 - Boxplot .....	33
Abbildung 47: Validation Accuracy - Auflösung 124x60 - Boxplot.....	34
Abbildung 48: Validation Accuracy - Auflösung 60x54 - Boxplot.....	34
Abbildung 49: Validation Accuracy - Auflösung 51x42 - Boxplot.....	35

Abbildung 50: Validation Accuracy - Ohne BatchNorm - Histogram .....	36
Abbildung 51: Validation Accuracy - Ohne BatchNorm/mit Bias - Histogram .....	37
Abbildung 52: Validation Accuracy - Mit BatchNorm - Histogram .....	37
Abbildung 53: Validation Accuracy - Kernel/Activity Regularizer - Histogram .....	38
Abbildung 54: Validation Accuracy - Kernel Regularizer - Histogram .....	39

## 8. Tabellenverzeichnis

Tabelle 1: Daten-Aufteilung .....	14
Tabelle 2: Verwendete Eigenschaften Augmentation .....	18

## 9. Projektabschluss

Der Anhang untergliedert sich in folgende Punkte:

**Zeitplan**

**Anforderungen**

**Bearbeitungsplan**

**Sprint 1 Review**

**Product Backlog Sprint 2**

**Sprint 2 Review**

**Product Backlog Sprint 3**

**Sprint 3 Review**

# Zeitplan

Phase	Arbeitspakete	Pit	Jascha	Frederik	Felix
Vorbereitung	Interview mit Kunden führen	2	2	2	2
	Datenstruktur wählen			2	1
	Kommunikationsstruktur festlegen				0,5
Erster Sprint	Product Backlog			1	1
	Interview mit Kunden führen	2	2	2	2
	Erfassen der Lern- und Testdaten		1		
	Bearbeiten und Augmentation der Daten				2
	Auslesen der XML-Daten	4			
	Label für die Daten			2	
	Einlesen der Daten		2	4	
	Skalieren der Daten				
	Model für das neuronale Netz entwerfen			5	1
	Lernprozess initiieren			1	
	Grafische Darstellung der Ergebnisse			0,5	
	Verschiedene Kleinigkeiten(Skripte, Hilfe usw.)		0,5	1	2
	Sprint Review	0,5	0,5	0,5	1
Zweiter Sprint	Product Backlog			0,5	1
	Interview mit Kunden führen	2	2	2	2
	Batch-Normalization		3		
	Hyperparameter optimieren			6	
	Verschiedene Inputgrößen testen	10			1
	Generator benutzen			1	
	Augmentation				8
	Bilder Ordnerstruktur				2
	Ausschneiden der Parkplätze überarbeiten			2	0,5
	Skript zum Ermitteln der Größe der Bilder				0,5
	Collab einarbeiten	0,5	0,5	0,5	0,5
	Verschiedene Kleinigkeiten(Skripte, Hilfe usw.)			2	2
	Sprint Review	0,5	0,5	0,5	1
Dritter Sprint	Product Backlog			0,5	1
	Interview mit Kunden führen	2	2	2	2
	Batch-Normalization		10		
	Hyperparameter optimieren			6	
	Verschiedene Inputgrößen testen	10			
	Regularization		15		
	Augmentation				8
	Verschiedene Kleinigkeiten(Skripte, Hilfe usw.)			2	2
	Sprint Review	0,5	1	1	1
	Interview mit Kunden führen	2	2	2	2
Abgabe	Systemdokumentation anfertigen			4	6
	Summe	36	44	53	53

## [Team 15]

---

Frederik Rieß Pit-Aurel Ehlers Jascha Schmidt Felix Willrich

---

# **[Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]**

## **Anforderungsdokument**

1.	Ziel und Zweck des Dokumentes .....	2
1.1.1.	Projektbeschreibung .....	2
1.1.2.	Kurzbeschreibung des Projekts .....	2
1.1.3.	Zweck des Projekts.....	2
1.1.4.	Hintergrund, Problemstellung, Motivation für das Projekt .....	3
1.1.5.	Ziele des Projekts .....	3
1.1.6.	Erfolgskriterien .....	3
2.	User Stories / Use Cases und Szenarien /Software Stories .....	4
3.	Anforderungen .....	5
4.	Anhang .....	14
4.1.	Abkürzungen.....	14
4.2.	Glossar .....	14
4.2.1.	Definition der Objekte .....	14
4.3.	Bildverzeichnis.....	14
4.4.	Tabellenverzeichnis.....	14

Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	14.03.2019	Felix Willrich	1. Entwurf + Eintragen aller Informationen	Abgeschlossen
0.2	19.03.2019	Felix Willrich	Beschreibung des Projekts eingetragen	Abgeschlossen
0.3	21.03.2019	Jascha Schmidt	Anforderung & Use-Case	Abgeschlossen
0.4	21.03.2019	Frederik Rieß	Anforderung & Use-Case	Abgeschlossen
0.5	21.03.2019	Pit-Aurel Ehlers	Anforderung & Use-Case	Abgeschlossen
1.0	21.03.2019	Felix Willrich	Letzter Review	Abgeschlossen

## 1. Ziel und Zweck des Dokumentes

Dieses Dokument beschreibt die Anforderungen der T-Systems on site services GmbH. Es handelt sich hierbei um die Systemdefinition, die der Auftragnehmer für den Auftraggeber (Kunde) erstellt, sodass der Kunde versteht und validieren kann, was das System leisten wird.

### 1.1.1. Projektbeschreibung

Dieses Projekt wird im Rahmen des Modules „Teamprojekt“ durchgeführt, welches von Herr Kircher & Frau Schiering doziert wird. Kunde für dieses Projekt ist Herr Philip May, welcher Angestellter bei der T-Systems on site GmbH ist und gleichzeitig die Rolle des Projektfansprechpartners einnimmt.

### 1.1.2. Kurzbeschreibung des Projekts

Das Projekt folgt einem gewissen Ablauf. Die Daten werden eingelesen, verarbeitet und ausgegeben. Diese drei Schritte werden anhand von folgendem Bild verständlich.



Abbildung 1: Produkt-Zyklus

Eine Kamera überträgt ein regelmäßig aufgezeichnetes Bild von einem Parkplatz an einen Mikrocontroller. Dieser verarbeitet die Daten, indem er das Bild in das konzipierte neuronale Netzwerk schickt. Anhand der Aufnahme soll bestimmt werden, wie viele Parkplätze frei sind. Diese Information wird an eine zentrale Stelle geschickt und weiterverteilt an eine Smartphone-App, damit die Nutzer zu jeder Zeit abrufen können, wohin sie fahren sollten. Die Bilder werden nicht vom Mikrocontroller weitergeschickt. Da aufgrund der Zeit Abstriche gemacht werden müssen, werden wir uns in diesem Projekt auf die Erkennung von freien Parkplätzen konzentrieren. Das bedeutet, dass wir keine Live-Daten aus der Kamera bekommen werden, bzw. auch keine App erstellen werden.

### 1.1.3. Zweck des Projekts

Das Projekt soll in erster Instanz zur Erkennung von freien Flächen eingesetzt werden und im Rahmen von Parkplätzen und Parkhäusern genutzt werden. Die Technik kann mit verschiedenen Inputdaten auf verschiedenste Felder ausgeweitet werden. Beispiele wären, Lagerbestände oder den Füllstand von verschiedenen Containern erkennen.

#### **1.1.4. Hintergrund, Problemstellung, Motivation für das Projekt**

Die T-Systems on site services GmbH in Person von Philip May benutzt im produktiven Sektor verschiedene Machine Learning/Deep Learning Applikationen und möchten durch dieses Produkt in weitere Felder stoßen bzw. weitere Erkenntnisse darüber gewinnen.

Für die Gruppe ergibt sich aufgrund von wenig Vorkenntnissen folgende Probleme:

- Neues Umfeld kennen lernen
- Geeignete Tools und Umgebung finden
- Datenbeschaffung zum Anlernen
- Prototypen erschaffen
- Genaue Erkennung implementieren
- Testumgebung

Die meisten Probleme werden oder wurden mit unserem Ansprechpartner besprochen und teilweise aufgearbeitet.

Die Motivation zu diesem Projekt ergibt sich aus dem ersten Stichpunkt der Probleme. Die Gruppe möchte in ein neues, aufstrebendes und sehr interessantes Thema einsteigen und dabei gleichzeitig Praxiserfahrung sammeln.

#### **1.1.5. Ziele des Projekts**

Das Projekt wird zuerst bis zu dem Schritt entwickelt bis das Netz angelernt ist und verschiedene Parkplatzsituationen erkannt werden. Eine Genauigkeit von 99% wird angestrebt.

#### **1.1.6. Erfolgskriterien**

Sollte die gewünschte Genauigkeit erlangt worden sein, wird das Projekt als Erfolg bezeichnet.

Weiterhin hinzukommen würden verschiedene Umgebungen, wie Schnee, Regen und andere Hindernisse wie Baulöcher oder Belegung von zwei Parkplätzen gleichzeitig. Sollten diese zusätzlichen Kriterien erfüllt werden, wird das Projekt in vollem Umfang als Erfolg gewertet. Es wird eine möglichst genaue Erkennung mit allen unterschiedlichen Faktoren angestrebt.

## 2. User Stories / Use Cases und Szenarien /Software Stories

Name	Beschreibung
Geeignete Projektstruktur	Die Daten sollen in einer geeigneten Projektstruktur zu finden sein.
Vorbereiten der Bilder	Um die Erkennung zu ermöglichen müssen aus den aufgenommenen Bildern die Parkplätze ausgeschnitten werden und als Einzelbilder gespeichert werden
Bearbeiten der Bilder	Für die Verarbeitung im CNN müssen die Bilder vorher in ein lesbares Format gewandelt werden.
Augmentation der Bilder	Die Genauigkeit des CNN soll durch eine anfängliche Augmentation der Bilder gesteigert werden.
Einlesen der Bilder	Das CNN muss die bearbeiteten Bilder einlesen, um sie dann zu verarbeiten.
Hyperparameter optimieren	Es sollen verschiedene Hyperparameter getestet werden, um ein optimales Ergebnis zu erzielen.
Auswahl der Layer	Eine unterschiedliche Anzahl und Größe der Layer erzeugen verschiedene Ergebnisse. Für ein optimales Ergebnis müssen alternative Ansätze ausprobiert werden.
Evaluation der Aktivierungsfunktion	Eine geeignete Aktivierungsfunktion sorgt für die korrekte Weitergabe der Informationen im CNN.
Zuverlässige Ausgaben	Das CNN soll für die Benutzer des Parkplatzes zuverlässige Aussagen treffen.
Ergebnisse überprüfen	Die Ergebnisse aus dem CNN sollen mit den echten Daten verglichen werden.
Grafische Darstellung	Die Genauigkeit verschiedener Testläufe soll zur Übersicht grafisch dargestellt werden.
Datenschutz einhalten	Daten sollen während des ganzen Projektes nur zur Auswertung benutzt werden. Keine Sicherung der Bilder.

Tabelle 1: User Stories

### 3. Anforderungen

ID	Projekt-01
Anforderungstyp	Strukturelle Anforderung
User Story/Use Case:	Geeignete Projektstruktur
Anforderung:	Die Projektordner, -Dateien und -Daten sollen in einer geeigneten Struktur zu finden sein.
Begründung:	Für die Übersicht über das Projekt müssen diese Dateien ordnungsgemäß angelegt werden. Möglichst zu Beginn sollte sich jeder im Klaren sein, wo was zu finden ist.
Abnahmekriterium:	Einigkeit im Team
Anforderer:	Team
Kundenzufriedenheit:	niedrig
Priorität:	mittel
Konflikte:	
Weiteres:	
Historie:	14.03.2019 FW GitHub Repository angelegt

Tabelle 2: Projekt-01 Anforderung

ID	Datenerhebung-01
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Vorbereiten der Bilder
Anforderung:	Ausschneiden der Parkplätze
Begründung:	Die Parkplätze müssen aus den Bildern ausgeschnitten werden
Abnahmekriterium:	Siehe Weiteres.
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	keine
Konflikte:	
Weiteres:	In den Testdaten sind die Parkplatzbilder schon bearbeitet.
Historie:	

Tabelle 3: Datenerhebung-01 Anforderung

ID	Datenerhebung-02
Anforderungstyp	Performanz
User Story/Use Case:	Augmentation der Bilder
Anforderung:	Die eingelesenen Bilder müssen durch Augmentation bearbeitet werden.
Begründung:	Durch Augmentation sind größere Datensätze mit einer großen Variation gegeben.
Abnahmekriterium:	Ein Bild soll in mehreren Variationen vorhanden sein
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 4: Datenerhebung-02 Anforderung

ID	CNN-01
Anforderungstyp	Performanz
User Story/Use Case:	Auswahl der Layer
Anforderung:	Die Anzahl der Layer mit Knoten im CNN muss bestimmt werden.
Begründung:	Durch unterschiedliche Strukturen können unterschiedliche Ergebnisse auftreten.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 5: CNN-01 Anforderung

ID	CNN-02
Anforderungstyp	Performanz
User Story/Use Case:	Hyperparameter optimieren
Anforderung:	Die Hyperparameter werden vor dem Trainieren des neuronalen Netzes gesetzt und müssen getestet werden.
Begründung:	Verschiedene Hyperparameter sorgen für eine unterschiedliche Genauigkeit und Output.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 6: CNN-02 Anforderung

ID	CNN-03
Anforderungstyp	Performanz
User Story/Use Case:	Evaluation der Aktivierungsfunktion
Anforderung:	Es muss eine passende Aktivierungsfunktion für den Problemfall gefunden werden.
Begründung:	Die Aktivierungsfunktion ist wichtig für den Output und die Weitergabe der Daten.
Abnahmekriterium:	
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 7: CNN-03 Anforderung

ID	CNN-04
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Ergebnisse überprüfen
Anforderung:	Die Daten aus dem CNN müssen mit den vorher errechneten Daten übereinstimmen.
Begründung:	Dies gewährt die Korrektheit des Systems.
Abnahmekriterium:	
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 8: CNN-04 Anforderung

ID	CNN-05
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Grafische Darstellung
Anforderung:	Die Ergebnisse und Testdaten sollen grafisch dargestellt werden.
Begründung:	Durch die grafische Darstellung sind verschiedene Testergebnisse und -Verläufe besser zu erkennen.
Abnahmekriterium:	Erkennen der Ergebnisse
Anforderer:	Team
Kundenzufriedenheit:	niedrig
Priorität:	niedrig
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 9: CNN-05 Anforderung

ID	CNN-06
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Weiterleiten von Daten
Anforderung:	Es dürfen keine privaten Daten weitergegeben werden wie zum Beispiel Bilder, sondern nur die ausgewerteten Daten.
Begründung:	Datenschutz
Abnahmekriterium:	Nur die Infos wie viele Parkplätze frei sind dürfen weitergegeben werden.
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	

Tabelle 10: CNN-06 Anforderung

## 4. Anhang

### 4.1. Abkürzungen

Abk.      Abkürzung  
CNN      Convolutional Neural Network

### 4.2. Glossar

#### 4.2.1. Definition der Objekte

Layer                Schichten im CNN (Convolutional Layer, Pooling Layer)  
Augmentation      Variation eines Bildes (gespiegelt, rotiert, verzerrt...)  
Hyperparameter    Vom Entwickler gesetzter Parameter, um Einfluss auf das CNN zu nehmen

### 4.3. Bildverzeichnis

Abbildung 1: Produkt-Zyklus ..... 2

### 4.4. Tabellenverzeichnis

Tabelle 1: User Stories .....	4
Tabelle 2: Projekt-01 Anforderung.....	5
Tabelle 3: Datenerhebung-01 Anforderung .....	6
Tabelle 4: Datenerhebung-02 Anforderung .....	7
Tabelle 5: CNN-01 Anforderung .....	8
Tabelle 6: CNN-02 Anforderung .....	9
Tabelle 7: CNN-03 Anforderung .....	10
Tabelle 8: CNN-04 Anforderung .....	11
Tabelle 9: CNN-05 Anforderung .....	12
Tabelle 10: CNN-06 Anforderung .....	13



## [Team 15]

Frederik Rieß Pit-Aurel Ehlers Jascha Schmidt Felix Willrich

---

# [Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]

## Projektmanagement-Plan

1.	Projektorganisation .....	2
1.1.	Rollen & Zuständigkeiten.....	2
1.2.	Projektstrukturplan .....	2
2.	Projektführung.....	3
2.1.	Rahmen- und Projektplan .....	3
2.2.	Vorbereitung.....	3
2.3.	Datenerhebung.....	4
2.4.	Programmierung.....	5
2.5.	Projektspezifische Aufgaben.....	6
3.	Projektunterstützung .....	7
3.1.	Tools für Entwicklung .....	7
3.2.	Konfigurationsmanagement .....	7
3.3.	Tools zur Projektkommunikation und zum Dokumentenaustausch .....	7
4.	Qualitäts- und Testplan .....	8
4.1.	Definition of Ready und Definition of Done .....	8
4.2.	Testdesign & Testautomatisierung.....	8
	Anhänge .....	9

### Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	14.03.2019	Felix Willrich	1. Entwurf + Eintragen aller Informationen	Abgeschlossen
0.2	20.03.2019	Felix Willrich	Erste Bearbeitung der Rollen	Abgeschlossen
0.3	27.03.2019	Jascha Schmidt	Testfälle beschrieben	Abgeschlossen
0.4	28.03.2019	Frederik Rieß	Arbeitspakete eingetragen	Abgeschlossen
0.5	28.03.2019	Felix Willrich	Gantt Diagramm	Abgeschlossen
1.0	31.03.2019	Felix Willrich	Vorläufig fertige Version	Abgeschlossen

## 1. Projektorganisation

### 1.1. Rollen & Zuständigkeiten

Zu Beginn des Projektes wurden die Stärken eines jeden Gruppenmitgliedes erörtert. Nachdem die Fähigkeiten, welche innerhalb der Gruppe zu Verfügung stehen, erfasst wurden, wurde ein Projektziel mit einem entsprechenden Produkt definiert. Aufbauend auf die Ziele dieses Projektes, wurde folgende Rollenverteilung innerhalb des Projektes vorgenommen:

Name	Rolle(n)	Beschreibung
<b>Felix Willrich</b>	Teamleiter, Projektmanagement, Entwickler	Projektleiter vom Projekt, Ansprechpartner vom Kunden, Entwickler für Teilbereiche
<b>Frederik Rieß</b>	Co-Teamleiter, Chefentwickler	Co-Teamleiter vom Projekt, Koordinierung der Entwicklung
<b>Pit-Aurel Ehlers</b>	Entwickler	Entwickler für Teilbereiche
<b>Jascha Schmidt</b>	Entwickler	Entwickler für Teilbereiche

### 1.2. Projektstrukturplan

Das Projekt wurde in vier Stadien eingeteilt. Die Vorbereitungsphase und die drei Sprints. Der Abschluss des Projekts wird nach dem dritten Sprint durchgeführt. Alle Aufgaben und Arbeitspakete werden in einem GANTT Diagramm dargestellt.

Siehe Anhang 1. Gantt Diagramm

## 2. Projektführung

### 2.1. Rahmen- und Projektplan

### 2.2. Vorbereitung

Zunächst wurden innerhalb des Teams wichtige Aspekte bezüglich der Struktur des Projektes und die Anforderungen mit dem Kunden besprochen. Zudem wurde festgelegt, wie untereinander kommuniziert wird, um gegebenenfalls Missverständnisse zu vermeiden.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
Interview mit Kunden führen	Team	Für die genauen Anforderungen an das Projekt wurde mit dem Kunden ein Interview geführt.	Vereinbarter Zeitpunkt	Kundenwünsche
Datenstruktur wählen	Team	Es wurde ein GitHub-Repository und verschiedene Ordner für das Projekt angelegt.	GitHub-Account	Geführtes Kundeninterview
Kommunikationsstruktur festlegen	Team	Das Team hat sich intern auf eine Kommunikation über WhatsApp geeinigt. Mit dem Kunden wird über Slack kommuniziert.	Smartphone mit WhatsApp	keine

### 2.3. Datenerhebung

In die Datenerhebung fließt das eigentliche Erfassen der Daten mit ein, sowie das Verarbeiten. Es müssen geeignete Quellen für Trainings- und Testdaten gefunden werden. Zudem soll sich mit der Augmentation der Daten beschäftigt und eingeschätzt werden, inwiefern und ob dies umgesetzt werden soll.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
<b>Erfassen der Lern- und Testdaten</b>	Jascha Schmidt	Um das CNN lernen zu lassen und das Gelernte zu testen, müssen geeignete Daten benutzt werden.	Internetzugang	Datensatz muss zu finden sein.
<b>Bearbeiten und Augmentation der Daten</b>	Felix Willrich	Die Daten müssen obduziert und auf Varianz überprüft bzw. dementsprechend bearbeitet werden.	Vorhandene Daten	Die Daten müssen vorhanden sein.
<b>Auslesen der XML-Datei</b>	Pit Ehlers	Zu komplett abgebildeten Parkplätzen sind XML-Dateien mit der Angabe vorhanden, ob die Parkplätze belegt sind. Es muss ein geeignetes Programm entwickelt werden, um diese Daten auszulesen.	Vorhandene Daten	Die Daten müssen vorhanden sein.
<b>Label für die Daten</b>	Felix Willrich	Die Daten (Parkplätze) müssen durch geeignete Label gekennzeichnet sein.	Vorhandene Daten	Der Datensatz muss vorhanden sein.

## 2.4. Programmierung

In der Teilaufgabe Programmierung werden die Daten zunächst richtig verarbeitet. Anschließend wird ein geeignetes Modell entworfen, um diese Daten für das Training des neuronalen Netzes zu benutzen. Dabei handelt es sich zunächst um ein initiales Modell, bei dem die Hyperparameter in den folgenden Sprints noch optimiert werden sollen. Eine grafische Darstellung soll zudem die Ergebnisse präsentieren.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
<b>Einlesen der Daten</b>	Frederik Rieß, Felix Willrich	Es ist eine geeignete Methode zu finden, die Daten einzulesen.	Vorhandene Daten, Jupyter Notebook mit Frameworks	Die Daten müssen vorhanden und vernünftig abgelegt sein.
<b>Skalieren der Daten</b>	Frederik Rieß, Jascha Schmidt	Die Daten müssen für Keras entsprechend skaliert bzw. geparsed werden.	Vorhandene Daten, Jupyter Notebook mit Frameworks	Die Daten müssen vorher eingelesen sein.
<b>Modell für das neuronale Netz entwerfen</b>	Frederik Rieß, Pit Ehlers	Für das neuronale Netz muss ein geeignetes Modell entworfen werden. Dabei ist die Auswahl der Layer von großer Bedeutung.	Skalierte Daten, Jupyter Notebook mit Frameworks	Die Daten müssen skaliert sein.
<b>Lernprozess initiieren</b>	Team	Für die Layer sind initiale Optimizer und eine Loss Function auszuwählen.	Skalierte Daten, Jupyter Notebook mit Frameworks	Ein Modell muss vorhanden sein.
<b>Grafische Darstellung der Ergebnisse</b>	Frederik Rieß	Die Ergebnisse sollen grafisch dargestellt werden können.	Fertiges neuronales Netz, Jupyter Notebook mit Frameworks	Das neuronale Netz sollte fertig programmiert sein.

## 2.5. Projektspezifische Aufgaben

Unter dieser Teilaufgabe wird die gesamte Dokumentation während der einzelnen Aufgaben und Arbeitspakete verstanden. Zudem sollen nach dem ersten Sprint ein Review gehalten und das neue Sprint Backlog entworfen werden.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
Dokumente erstellen	Team	Strukturen, Verfahren und Ergebnisse müssen dokumentiert werden.	Schreibprogramm auf dem PC	keine
Sprint-Review	Team	Review des vergangenen Sprints	Ergebnisse des Sprints	Fertige Dokumente
Sprint Backlog	Team	Sprint Backlog des neuen Sprints erstellen	Schreibprogramm auf dem PC	Vorheriges Sprint-Review

### 3. Projektunterstützung

#### 3.1. Tools für Entwicklung

Während des ersten Projektmeetings wurden folgende Tools für die Entwicklung beschlossen:

Name	Beschreibung
Python 3.X	Grundlage jeglicher Programmierung
Keras	Paket für maschinelles Lernen in Python geschrieben
NumPy	Paket, um das Rechnen mit Matrizen und Vektoren zu vereinfachen, Geschrieben in NumPy und wird passiv mitgenutzt
Matplotlib	Library für die Darstellung von Diagrammen
Tensorflow	Keras stützt sich auf Tensorflow
Jupyter Notebook	Sozusagen die IDE
Foto Datenbank	<a href="https://web.inf.ufpr.br/vri/databases/parking-lot-database/">https://web.inf.ufpr.br/vri/databases/parking-lot-database/</a> Grundlage zum Anlernen

#### 3.2. Konfigurationsmanagement

Im initialen Projektmeeting wurde festgelegt, dass keine Konfigurationen einheitlich eingesetzt werden. Es wird versucht eine möglichst breite Masse an Einstellungen auszutesten um das bestmögliche Ergebnis zu erreichen.

#### 3.3. Tools zur Projektkommunikation und zum Dokumentenaustausch

Während des ersten Projektmeetings wurden folgende Medien beschlossen, die zur Kommunikation und zum Austausch dienen sollen:

Name	Link(ggf.)	Beschreibung
Github	<a href="https://github.com/Scantraxx123/Parkplatzerkennung">https://github.com/Scantraxx123/Parkplatzerkennung</a>	Austausch aller Dokumente und Code
Slack		Kommunikation mit Kunden
Whatsapp		Kommunikation untereinander

## 4. Qualitäts- und Testplan

### 4.1. Definition of Ready und Definition of Done

Definition of Ready (DoR): Was muss ein Backlog Item mindestens erfüllen, damit es im nächsten Sprint Planning ausgewählt werden kann?

- Alle Teammitglieder müssen die Aufgabe verstanden haben
- Die Aufgabe wird als sinnvoll betrachtet
- Die Aufgabe unterstützt das Sprintziel
- Aufwand der Aufgabe wurde verhältnismäßig kalkuliert
- Ziel ist realistisch

Definition of Done (DoD): Was muss ein Backlog Item erfüllen, damit Sie am Ende eines Sprints vom Product Owner als fertig deklariert werden kann?

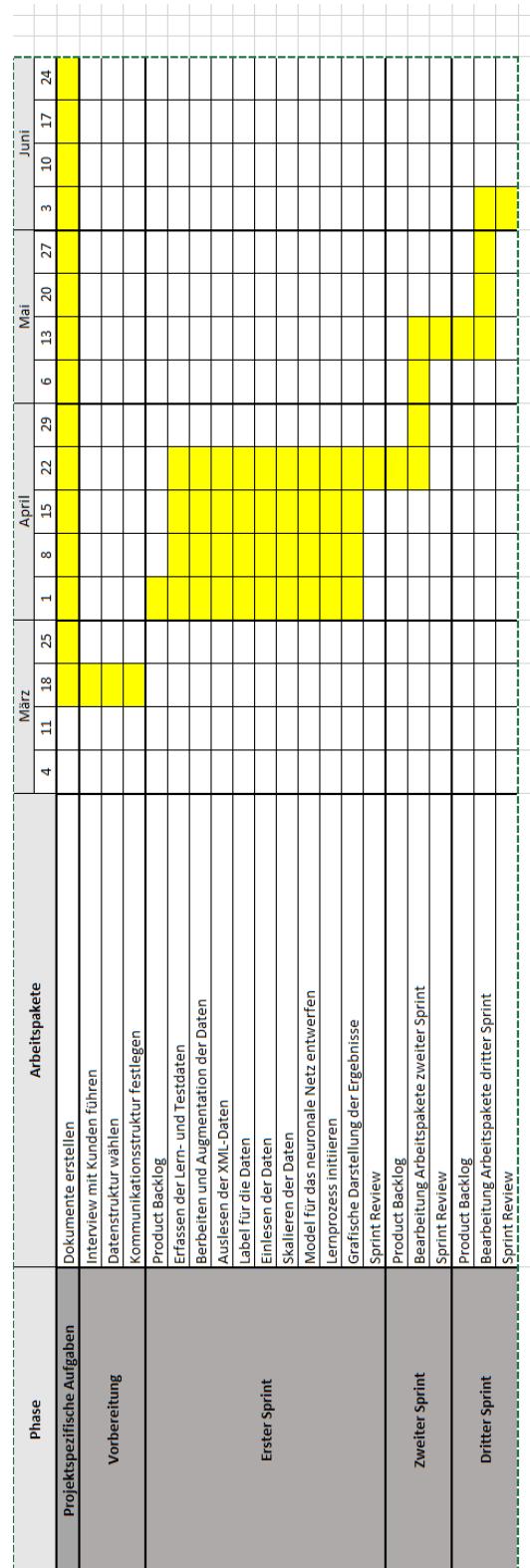
- Alle Teammitglieder sehen die Aufgabe als erledigt
- Item wurde getestet
- Ergebnis wird als hinreichend betrachtet
- Das Ergebnis ist in der Versionsverwaltung (Github) hochgeladen
- Ergebnis wurde von anderen Entwicklern überprüft
- Ergebnis ist im Gesamtkonzept mit eingebaut

### 4.2. Testdesign & Testautomatisierung

Das Projekt umfasst ein mit Hilfe des Frameworks Keras erstelltes neuronales Netzwerk (CNN). Keras bietet zusammen mit der Library Matplotlib die Möglichkeit, diverse Diagramme, wie zum Beispiel die Erkennungsrate, darzustellen. Dies wird die Grundlage der Tests sein, da somit nachvollzogen werden kann, wie genau die Parkplatzerkennung funktioniert, um unter anderem Overfitting zu vermeiden. Weitere Tests werden mit einzelnen Bildern vollzogen, die in den Testdaten liegen. Diese Testdaten zeigen eine Aufschlüsselung, wie viele Parkplätze frei sind. Diese werden mit einem Python-Skript ausgelesen und gegengeprüft durch das CNN.

## Anhänge

### 1. Gantt Diagramm



# **[Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]**

## **Sprint 1 Review**

1.	Arbeitspakete.....	2
1.1.	Datenerhebung .....	2
1.1.1.	Erfassen der Lern- und Testdaten.....	2
1.1.2.	Bearbeiten und Augmentation der Daten .....	2
1.1.3.	Auslesen der XML-Daten .....	2
1.1.4.	Label für die Daten.....	2
1.2.	Programmierung .....	3
2.	Arbeitspakete.....	4
3.	Entwicklerreview.....	9
3.1.	Frederik Rieß .....	9
3.1.1.	Datenerfassung .....	9
3.1.2.	CNN.....	9
3.2.	Pit Ehlers.....	10
3.3.	Jascha Schmidt .....	10
3.4.	Felix Willrich.....	11

**Versionen:**

Rev.	Datum	Autor	Bemerkungen	Status
0.1	22.04.2019	Felix Willrich	Erstellen des Sprint Reviews	Abgeschlossen
0.2	22.04.2019	Frederik Rieß	Persönlicher Kommentar erstellt	Abgeschlossen
0.3	22.04.2019	Pit Ehlers	Persönlicher Kommentar erstellt	Abgeschlossen
0.4	22.04.2019	Jascha Schmidt	Persönlicher Kommentar erstellt	Abgeschlossen
0.5	24.04.2019	Felix Willrich	Beta Version	Abgeschlossen
1.0	24.04.2019	Frederik Rieß	Finale Version	Abgeschlossen

## 1. Arbeitspakete

Der erste Sprint sah vor, sich in die Themen einzuarbeiten und das erste neuronale Netz aufzubauen. Dazu wurden diverse Arbeitspakete angelegt.

### 1.1. Datenerhebung

Die Datenerhebung hatte den Zweck Daten zu analysieren, Skripte zu schreiben und ggf. die Daten zu bearbeiten.

#### 1.1.1. Erfassen der Lern- und Testdaten

Verantwortlicher: Jascha Schmidt

Zum Start des Projektes wurde vom Product Owner eine Datenbank von Bildern bereitgestellt (<https://web.inf.ufpr.br/vri/databases/parking-lot-database/>). Da weitere Daten benötigt worden sind, sollte recherchiert werden, ob es geeignete Daten gibt. Es wurde eine weitere Datenbank an Bildern gefunden, die in Zukunft zum Anlernen bzw. Testen genutzt wird. <http://cnrspark.it/>

Vorteil bei beiden Datensätzen ist, dass Informationen über die einzelnen Parkplätze im CSV oder XML Format vorliegen. Dazugehörige Skripte wurden in einem anderen Arbeitspaket realisiert.

Arbeitspaket wird als abgeschlossen betrachtet.

#### 1.1.2. Bearbeiten und Augmentation der Daten

Verantwortlicher: Felix Willrich

Es wurde eine Möglichkeit gesucht, die existierenden Daten noch zu erweitern. Da in der Bearbeitung dem vorangegangen Arbeitspaket herausgekommen ist, dass zunächst keine bearbeitenden Bilder benötigt werden, wurde es nach anfänglicher Einarbeitung zunächst stillgelegt. Das vorläufige Ergebnis ist, dass eine Libary <https://github.com/aleju/imgaug> gefunden worden ist, die in Zukunft die Entwicklung vereinfacht.

Arbeitspaket wird zunächst als abgeschlossen betrachtet.

#### 1.1.3. Auslesen der XML-Daten

Verantwortlicher: Pit Ehlers

Die vorhandenen Bilder besitzen eine XML-Datei mit den jeweiligen Parkplätzen. Diese sagt aus, ob die einzelnen Plätze besetzt sind. Es wurde ein Skript entwickelt, welches diese Daten in einer extra Datei gesammelt ausgibt.

Dies liegt im Repository unter «Skripte»/«Cut\_images\_xml.py» und «cut\_images\_csv.py». Arbeitspaket wird als abgeschlossen betrachtet.

#### 1.1.4. Label für die Daten

Verantwortlicher: Felix Willrich (nachträglich Frederik Rieß)

Jeder Parkplatz braucht zum Einlesen ein Label, welches dem Netz sagt, ob der Platz besetzt ist oder nicht. Es wurde ein Notebook entwickelt, welches diese Label erstellt. Die Daten dazu sind im Repository unter «Notebooks» mit dem Namen „Datenerfassung\_XXX.ipnb“ zu finden

Arbeitspaket wird als abgeschlossen betrachtet.

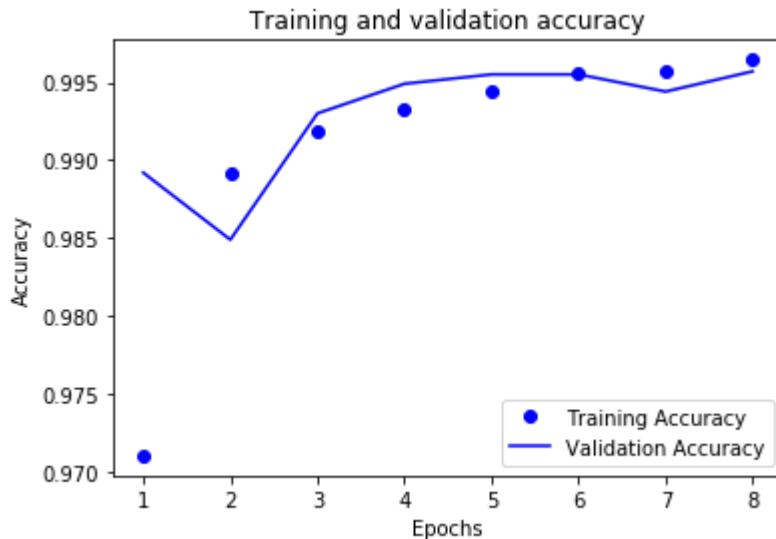
## 1.2. Programmierung

Verantwortlicher: Frederik Rieß

In diesem Schritt des ersten Sprints wurden folgende Pakete bearbeitet:

- Einlesen der Daten
- Skalieren der Daten
- Modell für das neuronale Netz entwerfen
- Lernprozess initiieren
- Grafische Darstellung der Ergebnisse

Im Teil der eigentlichen Programmierung des neuronalen Netzes sind die Arbeitspakete auf verschiedene Jupyter Notebooks aufgeteilt. Diese liegen in dem Verzeichnis <Notebooks> mit den Namen <CNN.ipynb> und <Datenerfassung.ipynb>. Die skalierten und mit Label versehenen Daten werden als Numpy-Array in einer separaten Datei gespeichert, damit dieser Prozess nicht immer wieder initiiert werden muss. In dem Notebook CNN.ipynb werden diese separaten Dateien anschließend eingelesen und weiterverarbeitet. Als Anfang ist hier ein Diagramm aufgeführt, um die aktuelle Validation Accuracy von 10000 Samples gegenüber 40000 gelernten Samples zu zeigen. Genaueres ist unter Punkt 3.1 zu entnehmen



Folgend ist noch die derzeitige Genauigkeit der selbst ausgeschnittenen Parkplätze zu sehen (100 Samples). Die Genauigkeit beträgt hier zurzeit 73% und der Loss liegt bei ca 2,54%.

```
100/100 [=====] - 0s 899us/step
2.546104352651164 0.73
```

Alle Arbeitspakete in diesem Schritt werden als abgeschlossen betrachtet. Die Details und das Einbauen von Skripten werden im nächsten Sprint verbessert bzw. bearbeitet.

## 2. Arbeitspakete

Folgende Arbeitspakete wurden aktualisiert:

ID	Datenerhebung-01
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Vorbereiten der Bilder
Anforderung:	Ausschneiden der Parkplätze
Begründung:	Die Parkplätze müssen aus den Bildern ausgeschnitten werden
Abnahmekriterium:	Siehe Weiteres.
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	keine
Konflikte:	
Weiteres:	In den Testdaten sind die Parkplatzbilder schon bearbeitet.
Historie:	17.04.2018 PA, FW: Erstellen der Skripte zum Ausschneiden der Bilder

ID	Datenerhebung-02
Anforderungstyp	Performanz
User Story/Use Case:	Augmentation der Bilder
Anforderung:	Die eingelesenen Bilder müssen durch Augmentation bearbeitet werden.
Begründung:	Durch Augmentation sind größere Datensätze mit einer großen Variation gegeben.
Abnahmekriterium:	Ein Bild soll in mehreren Variationen vorhanden sein
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	11.04.2019 FW: Einarbeitung in die Libary „imgaug“, Entscheidung in der Gruppe getroffen zuerst keine Augmentation durchzuführen

ID	CNN-01
Anforderungstyp	Performanz
User Story/Use Case:	Auswahl der Layer
Anforderung:	Die Anzahl der Layer mit Knoten im CNN muss bestimmt werden.
Begründung:	Durch unterschiedliche Strukturen können unterschiedliche Ergebnisse auftreten.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen

ID	CNN-02
Anforderungstyp	Performanz
User Story/Use Case:	Hyperparameter optimieren
Anforderung:	Die Hyperparameter werden vor dem Trainieren des neuronalen Netzes gesetzt und müssen getestet werden.
Begründung:	Verschiedene Hyperparameter sorgen für eine unterschiedliche Genauigkeit und Output.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen

ID	CNN-03
Anforderungstyp	Performanz
User Story/Use Case:	Evaluation der Aktivierungsfunktion
Anforderung:	Es muss eine passende Aktivierungsfunktion für den Problemfall gefunden werden.
Begründung:	Die Aktivierungsfunktion ist wichtig für den Output und die Weitergabe der Daten.
Abnahmekriterium:	
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen

ID	CNN-04
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Ergebnisse überprüfen
Anforderung:	Die Daten aus dem CNN müssen mit den vorher errechneten Daten übereinstimmen.
Begründung:	Dies gewährt die Korrektheit des Systems.
Abnahmekriterium:	
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen, Ergebnisse sind zunächst in Ordnung, müssen verbessert werden

ID	CNN-05
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Grafische Darstellung
Anforderung:	Die Ergebnisse und Testdaten sollen grafisch dargestellt werden.
Begründung:	Durch die grafische Darstellung sind verschiedene Testergebnisse und -Verläufe besser zu erkennen.
Abnahmekriterium:	Erkennen der Ergebnisse
Anforderer:	Team
Kundenzufriedenheit:	niedrig
Priorität:	niedrig
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen, Ergebnisse sind zunächst in Ordnung

### 3. Entwicklerreview

Jedes einzelne Projektmitglied hat eine Zusammenfassung über seine Aufgaben und Probleme in diesem Sprint verfasst.

#### 3.1. Frederik Rieß

In diesem Sprint bestand meine Hauptaufgabe hauptsächlich darin, eine erste funktionierende Version des Projektes zu entwickeln. Dabei sollten die Daten möglichst sinnvoll verarbeitet und eingelesen werden. Zudem sollte ein initiales Neuronales Netz entwickelt werden. Da ich bereits selbst ein paar kleine Beispiele in Bezug auf Convolutional Neural Networks programmiert habe, gab es hier keine größeren Probleme. Da wir innerhalb des Teams relativ viele getrennte Arbeitspakete hatten, war keine größere Kommunikation von Nöten, sodass ein Treffen pro Woche ausreichend war. In Zukunft wird die Kommunikation jedoch wichtiger werden, da es für alle Mitglieder um die Optimierung des Models geht und alle Aspekte ineinander greifen sollten. Folgend werden meine fachlichen Erkenntnisse in diesem Sprint noch aufgeführt.

##### 3.1.1. Datenerfassung

Für die Datenerfassung war es wichtig, eine geeignete Struktur aufzubauen. Über ein erstelltes Python-Skript wurden die belegten und nicht belegten Parkplätze aus ihren vielen verschiedenen Ordnern in lediglich zwei Ordner kopiert (Occupied, Empty). Zudem muss am Anfang festgelegt werden, auf welche Größe die einzelnen Bilder skaliert werden sollen. Hier wurde zunächst die Größe 40x80 Pixel ausgewählt, da die einzelnen Parkplätze zumeist rechteckig dargestellt werden.

Bei dem Einlesen werden nun die Ordner Empty und Occupied durchsucht. Wenn ein Bild gefunden wird, wird über den Namen des Ordners und einer Liste herausgefunden, welches Label für ein Bild zu vergeben ist. Der Index des Ordnernamens in dieser Liste ist dann das Label für das entsprechende Bild (0=Empty, 1=Occupied). Ist das Bild eingelesen, so wird es auf die angegebene Größe skaliert und zusammen mit dem passenden Label als Tupel an eine Liste übergeben. Die Bilder werden dabei als ein mehrdimensionales Array aus RGB-Werten eingelesen (Vektorisierung).

Anschließend müssen die erfassten Samples noch durchgemischt werden. Die zurzeit sortierten Samples sorgen ansonsten dafür, dass das Model später zu Overfitting neigt und nicht gut generalisiert wird. Daher sollte es bei dem Lernen immer eine Varianz der Daten geben. Die Daten und die Label werden dann auf eigene Listen verteilt. Beachtet werden muss außerdem, dass die Daten als Numpy-Array gespeichert werden, da Keras sonst diese Struktur falsch interpretieren könnte. Über die Funktion numpy.save werden sowohl das Array mit den Daten als auch das mit den Labels in einer eigenen Datei gespeichert, um die Daten später nicht immer wieder einlesen und umwandeln zu müssen.

Ein festgestelltes Problem am Ende des Sprints ist die Umwandlung der skalierten Bilder zu Numpy-Arrays. Das Problem liegt sehr wahrscheinlich an der nicht quadratischen Größe der Bilder. Dies sollte im nächsten Sprint behoben werden.

##### 3.1.2. CNN

In dem Model müssen zunächst die gespeicherten Dateien eingelesen werden. Wichtig ist hier, dass die einzelnen RGB-Werte in den Numpy-Arrays durch 255.0 dividiert werden. Die einzelnen Features müssen zwischen 0 und 1 sein, da sonst große Gradientenveränderungen auftreten können und so das Konvergieren des Netzes verhindert wird. Zudem sollten die Values der Features alle in derselben Reichweite sein.

Für die initiale Funktionalität des Netzwerks wurde der Optimizer Adam mit einer Lernrate von 0.001 gewählt. Dies kann und sollte in dem weiteren Verlauf neben dem Netzwerk selbst weiter angepasst werden. Fest steht, dass das Model sequenziell arbeitet und aus

mehreren Convolution Layern mit jeweils einem MaxPooling Layer besteht. Als Aktivierungsfunktion wurde zunächst „relu“ (Rectified Linear Unit) gewählt. Nach dem Abflachen (Flatten) der Layer zu 1D, muss es noch einen Dense Layer (voll vernetzt) geben, der für die Ausgabe zwischen 0 und 1 sorgt. Als Aktivierungsfunktion bietet sich hier Sigmoid an. Der Wert zwischen 0 und 1 entscheidet dann, ob es sich um einen belegten Parkplatz handelt oder nicht.

Die Lossfunction sollte die „binary\_crossentropy“ bleiben, da es sich bei uns um ein binäres Problem handelt. Über wie viele Epochen trainiert werden soll, muss noch weiter getestet werden. Beim Trainieren von 40000 Samples und Validieren von 10000 Samples wird aktuell eine Validierungsgenauigkeit von 99% erreicht. Bei bisher nur einzeln getesteten Bildern von großen Parkplätzen beträgt die Genauigkeit ca. 74%. Das Netzwerk muss selbstverständlich weiter optimiert werden. Dies war nur der initiale Aufbau des CNN.

### 3.2. Pit Ehlers

Aufgaben:

- Ein Python Skript schreiben, welches aus einer XML-Datei auslesen kann, wie viele Parkplätze belegt sind und wie viele frei sind.
- Ein Python Skript schreiben, welches die Parkplätze aus einem JPG Bild ausschneidet. Die Koordinaten, wo sich die Parkplätze befinden werden aus einer XML-Datei ausgelesen.

Vorgehen:

- Das erste Python Skript wurde mit PyCharm entwickelt, da diese Entwicklungsumgebung schon bekannt war.
- Nach Empfehlung von Herrn Willrich wurde auf Wing umgestiegen.

Probleme:

- Bei der Interpretation der Koordinaten kam es zu einem Missverständnis. Lösung: Nach Absprache mit den anderen Teampartner konnte Klarheit geschaffen werden.

### 3.3. Jascha Schmidt

Aufgaben:

- Weitere Testdaten beschaffen.
- Ein Python Skript schreiben, damit die neuen Testdaten eingelesen werden können.

Vorgehen:

- Die Möglichkeit eigene Testdaten zu erstellen wurde überprüft.
- Recherche nach weiteren Testdaten
- Erstellen des Einlesescripts auf Basis des Scripts zu einlesen der alten Daten

Probleme:

- Die Erstellung eigener Testdaten ist sehr umfangreich da sehr viele Daten benötigt werden und diese auch bearbeitet werden müssten. Lösung: Zunächst als weitere Testdaten ein anderes Datenset benutzen.
- Das neue Script hatte auf meinem MacBook nicht funktioniert. Lösung: Auf einem Windows-Laptop funktionierte es problemlos. Die Ursache war vermutlich ein Problem mit dem Dateisystem auf dem MacBook

### 3.4. Felix Willrich

Ich habe mich in diesem Sprint um verschiedene Sachen gekümmert. Zuerst gab es die Organisation, welche ich mir mit Frederik Rieß geteilt habe. Ich habe die Kommunikation mit dem Kunden übernommen bzw. die jeweiligen wöchentlichen Treffen in die Wege geleitet.

Aus der entwicklungstechnischen Sicht habe ich mich vor allem um die Erstellung von unterstützenden Skripten gekümmert. Gleichermassen habe ich alle Teammitglieder unterstützt bei Fragen in der Entwicklung. Das CNN wurde hauptsächlich von Frederik Rieß aufgebaut. Ich habe dort eine beratende Rolle eingenommen.

Ich habe außerdem Pit Ehlers unterstützt in seinem Skript für das Auslesen der XML Daten und in dem Zuge für Jascha Schmidt das Skript zum Auslesen der CSV Datei geschrieben.

Weiterhin habe ich mich in die imgaug Libary zur Augmentation von Daten eingearbeitet. Dieses Wissen könnte in zukünftigen Sprints wertvoller werden.

Aus meiner Sicht gab es vor allem Probleme in der Kommunikation. Wir haben uns bis jetzt jede Woche getroffen, aber zumeist wurden während dieser Treffen nicht alle Sachen verstanden, was dazu führte, dass vieles über nicht physische Kommunikationswege geregelt werden musste. Ich werde versuchen dies in den nächsten Sprints zu verbessern, da daraus auch Probleme bei der Programmierung entstanden sind.

Alles in allem ist der Sprint bis jetzt gut verlaufen. Die Kommunikation mit dem Kunden ist einwandfrei und auch die Ziele wurden alle erreicht. Wir sollten anstreben dies mit kleinen Verbesserungen weiterzuführen.

# **[Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]**

## **Product Backlog Sprint 2**

1.	Ziel Sprint 2.....	2
2.	Arbeitspakete.....	3
2.1.	Tests .....	3
2.2.	Hilfsarbeiten .....	4
2.3.	Anhänge .....	5

### Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	24.04.2019	Felix Willrich	1. Entwurf + Eintragen aller Informationen	Abgeschlossen
0.2	24.04.2019	Frederik Rieß	Eintragen/Verbesserung der Beschreibungen	Abgeschlossen
1.0	25.04.2019	Felix Willrich	Gantt Diagramm erstellt + Finale Version	Abgeschlossen

## 1. Ziel Sprint 2

Am 23.04.2019 wurde ein weiteres Treffen mit dem Kunden vereinbart. Dies sollte dazu dienen den Sprint 1 abzuschließen und gleichzeitig den zweiten Sprint zu besprechen. Während dieser Besprechung wurde vom Kunden ausdrücklich geäußert, dass das Testen von verschiedenen Parametern und Einflüssen gewünscht ist. Dies ist auch der Hauptaspekt unseres Teamprojektes und soll dazu führen, das Programm zu optimieren und Erkenntnisse für zukünftige Projekte zu gewinnen.

Aus diesem Grund werden im Sprint 2 diverse Testreihen bzw. kleine Hilfsarbeiten durchgeführt. Die Arbeitspakete wurden darauf angepasst. Alle Testreihen sollen dokumentiert werden, da diese an den Kunden weitergereicht werden. Unter anderem sollen verschiedene Größen der Samples oder auch die Augmentation der Bilder getestet werden.

## 2. Arbeitspakete

Die Arbeitspakete werden diesmal unterteilt in Testreihen und in Hilfsarbeiten. Die Tests stehen dabei im Fokus.

### 2.1. Tests

Bei den Tests kommen alle Aufgaben zu tragen, die sich damit beschäftigen die Parameter bzw. Einflüsse des Programms zu verändern. Ziel dabei ist es, herauszufinden welche Einstellung das bestmögliche Ergebnis ergibt. Die Arbeitspakete in diesem Bereich sind größer formuliert, da es während der Arbeit zu verschiedenen Tests kommen kann, die im Nachhinein alle dokumentiert werden.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
<b>Batch-Normalization</b>	Jascha Schmidt	Die Auswirkungen der Batch Normalization auf unsere Ergebnisse soll überprüft und dokumentiert werden.	Jupyter Notebook mit Frameworks	keine
<b>Hyperparameter optimieren</b>	Frederik Rieß	Die Layergrößen des CNNs sollten auf die Umgebung angepasst werden. Eine Libary kann diese Arbeit unterstützen.	Jupyter Notebook mit Frameworks	<a href="https://github.com/hyperopt/hyperopt">https://github.com/hyperopt/hyperopt</a>
<b>Verschiedene Inputgrößen testen</b>	Pit Ehlers	Die Bilder können in verschiedenen Größen eingelesen werden. Dies gilt zu testen, welches die optimalste Art ist.	Jupyter Notebook mit Frameworks	Skalierte Bilder
<b>Generator benutzen</b>	Frederik Rieß	Generatoren könnten als alternative Methode zum Einlesen und Skalieren der Daten benutzt werden.	Jupyter Notebook mit Frameworks	
<b>Augmentation</b>	Felix Willrich	Die Bilder können verändert werden und dann zum Anlernen benutzt werden. Es soll geschaut werden, ob dies einen Vorteil bringt.	Jupyter Notebook mit Frameworks	<a href="https://github.com/aleju/imgaug">https://github.com/aleju/imgaug</a>
<b>Weitere Tests</b>	Felix Willrich, Frederik Rieß, Pit Ehlers, Jascha Schmidt	Während des Arbeitens werden weitere Tests möglich sein, bzw. überhaupt erst auffallen.	Jupyter Notebook mit Frameworks	Unterschiedlich

## 2.2. Hilfsarbeiten

Die Hilfsarbeiten haben den Zweck die eigentlichen Tests zu unterstützen. Hierbei werden Skripte geschrieben, Hilfsmittel verstanden bzw. Daten analysiert.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
<b>Bilder Ordnerstruktur</b>	Felix Willrich	Verschiedene Pakete zum Anlernen und Testen des Netzes werden erstellt. Möglichst breite Streuung soll angestrebt werden.	Onedrive, Bilder	keine
<b>Ausschneiden der Parkplätze überarbeiten</b>	Felix Willrich, Jascha Schmidt	Zurzeit werden bei dem Skript verschiedene Störfaktoren mit ausgeschnitten. Diese sollen beseitigt werden.	Jupyter Notebook mit Frameworks	Skript zum Ausschneiden der Bilder aus einem großen Parkplatz, Ausgeschnittene Bilder
<b>Skript zum Ermitteln der Größe der Bilder</b>	Pit Ehlers	Ein Skript zum Ermitteln der Größe der ausgeschnittenen Parkplätze soll geschrieben werden. Dies hilft danach, verschiedene Inputgrößen zu wählen.	Python, Bilder	keine
<b>Collab einarbeiten</b>	Felix Willrich, Frederik Rieß, Pit Ehlers, Jascha Schmidt	Da zurzeit keine geeignete Hardware bereitsteht, soll ein öffentliches Netz genutzt werden. Dies soll jeder verstanden haben.	Jupyter Notebook	keine

## 2.3. Anhänge

### 1. Gantt Diagramm

Phase	Arbeitspakete																									
		März				April				Mai				Juni												
		4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24								
Projektspezifische Aufgaben	Dokumente erstellen																									
Vorbereitung	Interview mit Kunden führen																									
	Kommunikationsstruktur festlegen																									
	Erfassen der Lern- und Testdaten																									
	Bearbeiten und Augmentation der Daten																									
	Auslesen der XML-Daten																									
	Label für die Daten																									
	Einlesen der Daten																									
	Skalieren der Daten																									
	Modell für das neuronale Netz entwerfen																									
	Lernprozess initialisieren																									
	Grafische Darstellung der Ergebnisse																									
	Sprint Review																									
	Product Backlog																									
	Batch-Normalization																									
	Hyperparameter optimieren																									
	Verschiedene Inputgrößen testen																									
	Generator benutzen																									
	Augmentation																									
Zweiter Sprint	Weitere Tests																									
	Bilder Ordnerstruktur																									
	Ausschneiden der Parkplätze überarbeiten																									
	Skript zum Ermitteln der Größe der Bilder																									
	Collab einarbeiten																									
	Sprint Review																									
	Product Backlog																									
Dritter Sprint	Bearbeitung Arbeitspakete dritter Sprint																									
	Sprint Review																									

# [Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]

## Sprint 2 Review

1.	Arbeitspakete.....	2
1.1.	Tests.....	2
1.1.1.	Batch-Normalization .....	2
1.1.2.	Hyperparameter optimieren.....	5
1.1.3.	Verschiedene Inputgrößen testen.....	5
1.1.4.	Generator nutzen .....	5
1.1.5.	Augmentation .....	6
1.1.6.	Weitere Tests .....	7
1.2.	Hilfsarbeiten.....	8
1.2.1.	Bilder Ordnerstruktur .....	8
1.2.2.	Ausschneiden der Parkplätze überarbeiten .....	9
1.2.3.	Skript zum Ermitteln der Größe der Bilder .....	9
1.2.4.	Collab einarbeiten .....	9
2.	Use-Cases .....	10
3.	Entwicklerreview.....	15
3.1.	Frederik Rieß .....	15
3.2.	Pit Ehlers.....	15
3.3.	Jascha Schmidt .....	16
3.4.	Felix Willrich.....	17

Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	14.05.2019	Felix Willrich	Erstellen des Sprint Reviews	Abgeschlossen
0.2	15.05.2019	Frederik Rieß	Arbeitspakete beschrieben	Abgeschlossen
0.3	15.05.2019	Felix Willrich	Use Cases erweitert	Abgeschlossen
0.4	15.05.2019	Jascha Schmidt	Entwicklerreview eingetragen	Abgeschlossen
0.5	15.05.2019	Pit Ehlers	Entwicklerreview eingetragen	Abgeschlossen
1.0	16.05.2019	Felix Willrich	Finale Version	Abgeschlossen

## 1. Arbeitspakete

Der zweite Sprint sah vor, diverse Tests durchzuführen, um Erkenntnisse über die Grundlagen bzw. Eigenheiten des Netzes zu erkennen. Dazu wurden im Product Backlog diverse Pakete konzipiert, um Tests durchführen zu können.

Grundlegend sind die Pakete nicht alle vollständig abgeschlossen worden und im Gespräch mit dem Kunden am 14.05.2019, welches zur Nachbereitung des Sprint 2 und Vorbereitung von Sprint 3 dienen sollte, wurde festgelegt, dass die Pakete weitergeführt werden. Somit werden diesen Sprint nur Zwischenergebnisse dokumentiert.

### 1.1. Tests

Die Tests wurden konzipiert um Erkenntnisse über das Netz gewinnen, bzw. die optimalen Einstellungen zu finden.

#### 1.1.1. Batch-Normalization

Verantwortlicher: Jascha Schmidt

Batch Normalization ist ein Normalisierungsverfahren, bei dem die Ausgabe eines Layers vor der Aktivierungsfunktion normalisiert wird, so dass der Mittelwert nahe an 0 liegt und die Standardabweichung nahe bei 1. Die Normalisierung wird während des Trainings Batch-Weise berechnet und später werden laufende Mittelwerte, die während des Trainings bestimmt worden sind, verwendet.

Unser Standartnetz wurde erweitert, um die Auswirkung von Batch-Normalization zu testen. Folgende Ergebnisse sind zu erkennen:

```
# Das CNN
# Das Model arbeitet sequentiell
# Aktivierungsfunktion kann ersetzt werden
# Die Anzahl der Neuronen der Layer kann verändert werden
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D(16, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Flatten()) # "Abflachen" der Layer zu 1D
model.add(layers.Dense(16, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
# Auswahl der Loss-Function und des Optimizers
# binary_crossentropy muss bestehen bleiben, da binäres Problem
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Found 144965 images belonging to 2 classes.  
[0.3055513478815556, 0.9050000011920929]

```

# Das CNN
# Das Model arbeitet sequentiell
# Aktivierungsfunktion kann ersetzt werden
# Die Anzahl der Neuronen der Layer kann verändert werden
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D(16, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Flatten()) # "Abflachen" der Layer zu 1D
model.add(layers.Dense(16, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
# Auswahl der Loss-Function und des Optimizers
# binary_crossentropy muss bestehen bleiben, da binäres Problem
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

Found 144965 images belonging to 2 classes.  
[0.5577460393309593, 0.8500000059604644]

```

# Das CNN
# Das Model arbeitet sequentiell
# Aktivierungsfunktion kann ersetzt werden
# Die Anzahl der Neuronen der Layer kann verändert werden
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D(16, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Flatten()) # "Abflachen" der Layer zu 1D
model.add(layers.Dense(16, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
# Auswahl der Loss-Function und des Optimizers
# binary_crossentropy muss bestehen bleiben, da binäres Problem
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

Found 144965 images belonging to 2 classes.  
[0.6564721271395684, 0.8152500033378601]

```

# Das CNN
# Das Model arbeitet sequentiell
# Aktivierungsfunktion kann ersetzt werden
# Die Anzahl der Neuronen der Layer kann verändert werden
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Conv2D(16, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

model.add(layers.Flatten()) # "Abflachen" der Layer zu 1D
model.add(layers.Dense(16, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
# Auswahl der Loss-Function und des Optimizers
# binary_crossentropy muss bestehen bleiben, da binäres Problem
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

Found 144965 images belonging to 2 classes.  
[0.4100228056311607, 0.8829999983310699]

Das Standardnetz besteht zurzeit aus zwei «Convolutional Layern» und einem «Dens-Layer», um die Ergebnisse am Ende auszugeben. Aus diesem Grund ist es zurzeit nur möglich zwei Mal die Batch-Normalization einzubauen. Wie man an diesen Ergebnissen sieht, werden die besten Ergebnisse bei keiner Normalization erreicht bzw. wenn beide Layer damit bearbeitet werden. In Sprint 3 wird das Netz nochmal erweitert, deshalb dienen diese Ergebnisse als richtungsweisend. Das Arbeitspaket ist noch nicht abgeschlossen und weitere Tests werden durchgeführt.

### 1.1.2. Hyperparameter optimieren

Verantwortlicher: Frederik Rieß

Für die Optimierung der Hyperparameter sollte die Library «Hyperopt» hinzugezogen werden. Dadurch können verschiedene Ranges (zum Beispiel die Anzahl der Filter) eingegeben werden und das Model testet dieses, sodass am Ende die optimalen Werte genommen werden können.

Allerdings ist dies mit Google Colab aufgrund der großen Datenmenge nicht möglich gewesen. Auch deshalb wird der Datensatz für das Model reduziert, um dann im Sprint 3 diese Optimierungen durchzuführen. Stattdessen konnte in diesem Sprint festgestellt werden, dass bereits ein wenig komplexes Model eine Validierungsgenauigkeit von 99% erreicht. Um «Overfitting» zu vermeiden, muss das Model also nochmal überarbeitet werden.

Das Paket wird in Sprint 3 weitergeführt.

### 1.1.3. Verschiedene Inputgrößen testen

Verantwortlicher: Pit Ehlers

Das Netz beinhaltet die Funktion die eingelesenen Bilder in verschiedenen Größen einzulesen. Das Standartnetz liest die Bilder in der Größe 60x60 ein. Es sollte getestet werden, ob das Netz bessere Ergebnisse liefert, wenn verschiedene Größen zum Einlesen benutzt werden. Dazu wurde ein Skript erstellt, welches die Größen der Bilder analysiert. Dies wird unter dem Punkt 1.2.3. erklärt. Aufgrund von diversen Verzögerungen konnten noch keine aussagekräftigen Ergebnisse erstellt werden. Aus diesem Grund wird das Paket in Sprint 3 weitergeführt.

### 1.1.4. Generator nutzen

Verantwortlicher: Frederik Rieß

Durch das Ersetzen der ursprünglichen Datenerfassung mit einem Generator wurde das Labeln der Daten stark vereinfacht. Zudem kann die spätere Augmentation dadurch besser implementiert werden, als wenn externe Bibliotheken importiert werden müssten.

Unter anderem muss nur das Directory der Daten, die Größe der Bilder und die Batch-Size angegeben werden. Zudem sollte der Generator den Modus «binary» haben, da dieser auf unsere Problemstellung zutrifft. Dieser «ImageDataGenerator» kann nun noch weitere Parameter erhalten, um die Bilder mit Augmentation zu bearbeiten.

Das Paket wird als abgeschlossen betrachtet.

### 1.1.5. Augmentation

Verantwortlicher: Felix Willrich

Das Keras Framework bietet die Möglichkeit die Bilder zu bearbeiten, um verschiedene Abwandlungen der Bilder einzulesen. Dies hat den Vorteil das Netz auf mehrere Möglichkeiten von Bildern vorzubereiten. Dazu gibt es den «Imagedatagenerator» mit verschiedenen Einstellungen. Die Ergebnisse aus Sprint 2 sind geben zurzeit nur eine Richtlinie über erste Funktionen, da aufgrund von anderen Arbeiten die Zeit gefehlt hat alle Funktionen zu testen.

Konfiguration	Ergebnis
Standard	[0.39261420667171476, 0.8827500015497207]
featurewise_center=True	[0.45597286745905874, 0.8722500026226043]
samplewise_center=True	[1.7466542184352876, 0.7592500030994416]
featurewise_std_normalization=True	[0.4122985728085041, 0.8830000013113022]
samplewise_std_normalization=True	[1.0387932986021042, 0.5770000040531158]
zca_whitening=True	[0.35725187510252, 0.8905000001192093]
horizontal_flip=True	[0.5051708981394768, 0.8517500042915345]
vertical_flip=True	[0.6087541967630387, 0.8272500038146973]
rotation_range=45	[0.4142412066459656, 0.8805000007152557]
rotation_range=90	[0.6669619396328926, 0.8270000010728836]
rotation_range=135	[0.6810234829783439, 0.835999995470047]
rotation_range=180	[0.8304314732551574, 0.7987499982118607]
width_shift_range = 6	[0.45458429902791975, 0.8712500035762787]

Alle Funktionen hatten einen Durchlauf mit dem alten Netz «Notebooks/CNNwithGen.ipynb». Die Ergebnisse sind in zwei Werte aufgeteilt. Der erste Wert zeigt den «Loss» an, wie weit die Daten vom eigentlichen Ergebnis entfernt sind. Der zweite Wert zeigt die Genauigkeit der Gesamtdaten an. Angestrebt wird ein möglichst geringer Loss und eine möglichst hohe Genauigkeit.

Aufgrund der geringen Zeit werden die Daten hier auch als richtungsweisend angesehen und mit dem neuen Netz im dritten Sprint weitergeführt.

#### **1.1.6. Weitere Tests**

Verantwortliche: Alle

Das Arbeitspaket konnte diesen Schritt nicht ausgeführt werden, da während der Arbeiten am Programm diverse andere Baustellen aufgetaucht sind.

Das Paket wird weiter geführt in Sprint 3.

## 1.2. Hilfsarbeiten

Die Hilfsarbeiten wurden konzipiert, um die eigentlichen Tests zu unterstützen.

### 1.2.1. Bilder Ordnerstruktur

Verantwortlicher: Felix Willrich

Damit das Netz getestet werden konnte, mussten verschiedene Pakete mit Bildern erstellt werden. Die Struktur wurde in «train» und «test» eingeteilt. Jeder dieser Ordner beinhaltete zwei weitere Ordner «Empt» und «Occupied». Dies hatte den Vorteil, dass das Netz auf diese Struktur angepasst wurde und gleichzeitig als Labels angesehen wurden.

Damit die Pakete für jeden erreichbar waren, wurden diese auf OneDrive hochgeladen. Eine Anleitung wie auf diese Pakete zugegriffen werden kann, ist unter «Notebooks\README\_Download.txt» zu finden.

Zum Schluss dieses Sprints wurde ein Paket mit ca. 700.000 Bildern in einer 50/50 Aufteilung für «test» und «train» genutzt zum Trainieren. Dieses Paket bestand aus zwei Parkplätzen, woraus ein Parkplatz aus zwei unterschiedlichen Winkeln aufgenommen worden ist.

Um einen fremden Parkplatz zu testen wurde aus dem ersten Sprint der CNR Parkplatz benutzt

(<http://cnrspark.it/>)

Folgende Pakete wurden angelegt:

Name	Beschreibung
CNR_TEST	Daten von <a href="http://cnrspark.it/">http://cnrspark.it/</a> Nur «test» Ordner vorhanden zum Gegentesten des Netzes
PUC	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Trainingsdaten: 80% Validation 20%
PUC_50_50	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Trainingsdaten: 50% Validation 50%
PUC_UFPR05_04_50_50	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Kombination aus drei Parkplätzen Trainingsdaten: 50% Validation 50%
UFPR04	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Trainingsdaten: 80% Validation 20%
UFPR04_05	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Kombination aus zwei Parkplätzen Trainingsdaten: 80% Validation 20%
UFPR04_50_50	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Trainingsdaten: 50% Validation 50%
UFPR05	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Trainingsdaten: 80% Validation 20%
UFPR05_50_50	Daten von <a href="https://web.inf.ufpr.br">https://web.inf.ufpr.br</a> Trainingsdaten: 50% Validation 50%

Das Paket wird weitergeführt in Sprint 3.

### 1.2.2. Ausschneiden der Parkplätze überarbeiten

Verantwortlicher: Felix Willrich (nachträglich Frederik Rieß)

Da das vorherige Ausschneiden der einzelnen Parkplätze nicht optimal gelungen war, musste dieses durch ein Skript angepasst werden. Als neuer Ansatz wurde dabei die Bibliothek «OpenCV» genutzt, womit die Konturen besser ausgeschnitten werden konnten. Dabei wird innerhalb der 4 Koordinaten des Parkplatzes das größtmögliche Rechteck gewählt und ausgeschnitten. Anschließend wird das ausgeschnittene Bild noch entsprechend gedreht.

Dadurch entsteht ein kleiner Verlust des Bildes an den Seiten. Diese mögliche Variante könnte zukünftig mit anderen Verfahren verglichen werden.

Das Arbeitspaket wird als abgeschlossen betrachtet. Das Skript liegt unter «Skripte/cutImages.py».

### 1.2.3. Skript zum Ermitteln der Größe der Bilder

Verantwortlicher: Pit Ehlers (nachträglich Felix Willrich)

Um den Test «Verschiedene Inputgrößen testen» zu unterstützen wurde ein Skript geschrieben, welches die Bilder der ausgeschnittenen Parkplätze durchiteriert und die verschiedenen Größen in eine Txt-Datei schreibt. Dies hilft dabei, die Bilder besser zu analysieren und Vorgaben für das Input-Shape im Netz zu haben.

Das erstellte Skript liegt im Repository unter «Skripte/pictureSize.py». Das Ergebnis unter «Notebooks/avg\_picture\_size.txt».

Das Paket wird als abgeschlossen betrachtet.

### 1.2.4. Collab einarbeiten

Verantwortliche: Alle

Da zurzeit keine Hardware der Gruppe zur Verfügung steht, musste auf eine Alternative zurückgegriffen werden. Google stellt eine Umgebung mit potenter Hardware zur Verfügung, die mit einem Google Konto benutzt werden konnte. Jeder der Mitglieder der Gruppe hatte während des Sprint 2 damit gearbeitet, damit sollte die Einarbeitung abgeschlossen sein. Die einzige Einstellung, die getroffen werden musste, bezog sich auf die Nutzung der GPU anstatt der CPU.

Probleme traten vor allem bei der längeren Nutzung auf, da die Session die Verbindung unterbrochen hat. Es konnte ein reconnect durchgeführt werden, aber hierzu musste der Rechner immer im Auge behalten werden. Die Umgebung wird weiterhin genutzt, da Anpassungen an den Trainingsdaten durchgeführt worden sind. Sollte einmal ein längerer Test beabsichtigt werden, kann vom Kunden eine Umgebung bereitgestellt werden.

Das Paket wird als abgeschlossen betrachtet.

## 2. Use-Cases

Folgende Use-Cases wurden aktualisiert:

ID	Projekt-01
Anforderungstyp	Strukturelle Anforderung
User Story/Use Case:	Geeignete Projektstruktur
Anforderung:	Die Projektordner, -Dateien und -Daten sollen in einer geeigneten Struktur zu finden sein.
Begründung:	Für die Übersicht über das Projekt müssen diese Dateien ordnungsgemäß angelegt werden. Möglichst zu Beginn sollte sich jeder im Klaren sein, wo was zu finden ist.
Abnahmekriterium:	Einigkeit im Team
Anforderer:	Team
Kundenzufriedenheit:	niedrig
Priorität:	mittel
Konflikte:	
Weiteres:	
Historie:	14.03.2019 FW: GitHub Repository angelegt Sprint 2 FW: Diverse Bildpakete im OneDrive angelegt

ID	Datenerhebung-01
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Vorbereiten der Bilder
Anforderung:	Ausschneiden der Parkplätze
Begründung:	Die Parkplätze müssen aus den Bildern ausgeschnitten werden
Abnahmekriterium:	Siehe Weiteres.
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	keine
Konflikte:	
Weiteres:	In den Testdaten sind die Parkplatzbilder schon bearbeitet.
Historie:	17.04.2019 PA, FW: Erstellen der Skripte zum Ausschneiden der Bilder 03.05.2019 FW, FR_ Überarbeitetes Skript hochgeladen, bessere Ergebnisse

ID	Datenerhebung-02
Anforderungstyp	Performanz
User Story/Use Case:	Augmentation der Bilder
Anforderung:	Die eingelesenen Bilder müssen durch Augmentation bearbeitet werden.
Begründung:	Durch Augmentation sind größere Datensätze mit einer großen Variation gegeben.
Abnahmekriterium:	Ein Bild soll in mehreren Variationen vorhanden sein
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	11.04.2019 FW: Einarbeitung in die Libary „imgaug“, Entscheidung in der Gruppe getroffen zuerst keine Augmentation durchzuführen  Sprint 2 FW: Einarbeiten in den ImageDataGenerator, Erste Tests durchgeführt

ID	CNN-01
Anforderungstyp	Performanz
User Story/Use Case:	Auswahl der Layer
Anforderung:	Die Anzahl der Layer mit Knoten im CNN muss bestimmt werden.
Begründung:	Durch unterschiedliche Strukturen können unterschiedliche Ergebnisse auftreten.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen 13.05.2019 FR: Umbauen des Netzes, kleinere Netze ergeben zurzeit bessere Erfolge

ID	CNN-02
Anforderungstyp	Performanz
User Story/Use Case:	Hyperparameter optimieren
Anforderung:	Die Hyperparameter werden vor dem Trainieren des neuronalen Netzes gesetzt und müssen getestet werden.
Begründung:	Verschiedene Hyperparameter sorgen für eine unterschiedliche Genauigkeit und Output.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen Sprint 2 FR: Einarbeiten in die Hyperopt Library und erste Tests

### 3. Entwicklerreview

#### 3.1. Frederik Rieß

Zu meinen Aufgaben in diesem Sprint gehörte es zunächst, das Ausschneiden der Bilder zu optimieren. Mit der Bibliothek «OpenCV» gab es hier keine Probleme, allerdings kann das Ausschneiden zukünftig mit anderen Verfahren noch verglichen werden, um das bestmögliche Ergebnis zu erzielen.

Außerdem musste ein Generator implementiert werden, der die Datenerfassung erleichtert. Auch hier gab es keinerlei Probleme, da ich mich an Vorgaben gehalten habe.

Das größte Problem bestand in dem Testen des neuronalen Netzes. Da Google Colab den Client nach einer gewissen Nutzungsdauer disconnected, musste teilweise das Model über 5 Stunden kontrolliert werden. Auf Dauer erwies sich dies nicht als praktikabel, vor allem da das Testen mit der Bibliothek «HyperOpt» noch mehr Zeit in Anspruch nimmt. Daher wird zukünftig auf ein kleineren Datensatz zurückgegriffen.

Nach den Absprachen im Team war relativ schnell klar, wer was zu erledigen hatte. Das unabhängige Testen erwies sich nun leider als relativ unnötig, da wir letzten Endes auf einen kleineren Datensatz umsteigen. Allerdings konnte durch die hohe Validation-Accuracy gleich zu Beginn des Trainierens gesehen werden, dass die Aufteilung der Trainings- und Testdaten nicht gut gewählt war und die Variation an Daten deutlich erhöht werden muss. Die Arbeit untereinander empfand ich als etwas zu ruhig, da unter anderem die Ergebnisse der jeweiligen Tests erst gegen Ende des Sprints besprochen wurden. So konnten keine aktuellen Zwischenstände bzw. Erkenntnisse mitgeteilt werden.

#### 3.2. Pit Ehlers

Aufgaben:

- Tests durchführen, um festzustellen, welche Inputgröße der Parkplatzbilder zur höchsten Genauigkeit führen.

Vorgehen:

- Da mit sehr vielen Testdaten gearbeitet wurde, wurde Google Colab genutzt, um die Tests möglichst schnell durchzuführen.
- Durch ein Python Skript wurde eine Txt-Datei erstellt, in der alle Inputgrößen stehen, die wir als Daten haben.
- Die Inputgrößen aus dieser Txt-Datei werden mit Google Colab getestet.

Probleme:

- Die Tests dauerten zu lange. Lösung: Im nächsten Sprint werden weniger Trainingsdaten etc. genutzt, um mehr Tests durchzuführen.

Ergebnis:

- Es wurden zu wenig Tests durchgeführt, um zu einem aussagekräftigen Ergebnis zu kommen.

### **3.3. Jascha Schmidt**

Aufgaben:

- Den Einfluss von einer Batchnormalisierung der Layer auf unser CNN ermitteln

Vorgehen:

- Die Batchnormalisierung einbauen.
- Durch aktivieren/deaktivieren der Batchnormalisierung in einem oder beiden Layers und anschließendes dokumentieren der Ergebnisse den Einfluss auf das CNN bestimmen

Probleme:

- In diesem Sprint sind bei mir keine größeren Probleme aufgetreten

### 3.4. Felix Willrich

Ich habe diesen Sprint wieder die Koordination der Arbeit übernommen, Hilfsskripte geschrieben und die Augmentation der Bilder getestet.

Zuerst musste die Aufteilung der Arbeiten in Sprint 2 durchgeführt werden. Dort habe ich mich mit Frederik beraten inwieweit welche Aufteilung sinnvoll ist. Dies haben wir im Anschluss an das zweite Treffen mit dem Kunden gemacht, welches Frederik und ich durchgeführt haben.

Nach der Aufteilung habe ich diverse Bildpakete gepackt, um verschiedenste Möglichkeiten zu haben das Netz zu trainieren. Darauf aufbauend habe ich ein Skript erstellt, welches die Bildgrößen der jeweiligen Bilder erkennt und speichert. Dieses Skript soll Pit die Arbeit bei seinen Tests erleichtern. Als letztes habe ich mich um die Augmentation der Bilder gekümmert, Die Ergebnisse sind in den jeweiligen Bereichen dieses Dokumentes beschrieben.

Es gab diesen Sprint ein paar Probleme in der Koordination bzw. es wurden falsche Annahmen getroffen. Die Aufteilung funktionierte in Ordnung nur wurden die Arbeitsschritte nicht qualitativ oder zu spät umgesetzt. Zum nächsten Sprint versuche ich deshalb ein wenig mehr Druck zu machen, bzw. die Treffen besser zu gestalten.

Die Tests konnten durch falsche Annahmen in der Durchführung ungenügend durchgeführt werden und die Testergebnisse sind weniger wert als angenommen. Auch dies werden wir zum nächsten Sprint verbessern, da der nächste Sprint komplett aufs Testen ausgelegt sein wird.



## **[Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]**

### **Product Backlog Sprint 3**

1. Ziel Sprint 3.....	2
2. Arbeitspakete .....	3
2.1. Tests .....	3
2.2. Anhänge.....	4

Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	15.05.2019	Felix Willrich	1. Entwurf + Eintragen aller Informationen	Abgeschlossen
1.0	16.05.2019	Felix Willrich	Finale Version	Abgeschlossen

## 1. Ziel Sprint 3

Am 14.05.2019 wurde ein weiteres Treffen mit dem Kunden vereinbart. Dies sollte dazu dienen den Sprint 2 abzuschließen und den dritten Sprint zu besprechen.

Während der Besprechung von Sprint 2 und 3 wurden die Komplikationen von Sprint 2 beschrieben und dass die Tests nicht zufriedenstellend erfüllt worden sind. Gleichzeitig wurden Mängel in dem Standardnetz festgestellt, die während der Besprechung korrigiert worden sind. Aus diesem Grund werden alle Tests vom vorherigen Sprint in den dritten Sprint mit übernommen. Es gelten die gleichen Ziele wie in Sprint 2. Es sollen möglichst viele Testdaten akquiriert werden. Diese Testdaten sollen zum Ende des Sprints alle dem Kunden zur Verfügung gestellt werden. Erkenntnisse über das Netz sind wichtiger als ein vollends perfektes System.

## 2. Arbeitspakete

Die Arbeitspakete bestehen nur aus Tests, da die Hilfsarbeiten in diesem Sprint soweit abgeschlossen sein sollten.

### 2.1. Tests

Bei den Tests kommen alle Aufgaben zu tragen, die sich damit beschäftigen die Parameter bzw. Einflüsse des Programms zu verändern. Ziel dabei ist es, herauszufinden welche Einstellung das bestmögliche Ergebnis ergibt. Die Arbeitspakete in diesem Bereich sind größer formuliert, da es während der Arbeit zu verschiedenen Tests kommen kann, die im Nachhinein alle dokumentiert werden.

Arbeitspaket	(Haupt-) Verantwortlicher	Beschreibung	Benötigte Ressourcen	Abhängigkeiten
<b>Batch-Normalization</b>	Jascha Schmidt	Die Auswirkungen der Batch Normalization auf unsere Ergebnisse soll überprüft und dokumentiert werden.	Jupyter Notebook mit Frameworks	keine
<b>Hyperparameter optimieren</b>	Frederik Rieß	Die Layergrößen des CNNs sollten auf die Umgebung angepasst werden. Eine Libary kann diese Arbeit unterstützen.	Jupyter Notebook mit Frameworks	<a href="https://github.com/hyperopt/hyperopt">https://github.com/hyperopt/hyperopt</a>
<b>Verschiedene Inputgrößen testen</b>	Pit Ehlers	Die Bilder können in verschiedenen Größen eingelesen werden. Dies gilt zu testen, welches die optimalste Art ist.	Jupyter Notebook mit Frameworks	Skalierte Bilder
<b>Augmentation</b>	Felix Willrich	Die Bilder können verändert werden und dann zum Anlernen benutzt werden. Es soll geschaut werden, ob dies einen Vorteil bringt.	Jupyter Notebook mit Frameworks	<a href="https://github.com/aleju/imgaug">https://github.com/aleju/imgaug</a>
<b>Regularization</b>	Jascha Schmidt	Die Bilder können Overfitting erzeugen im Netz. Dies führt zu einem zu sensiblen Netz. Dies sollte verhindert werden,	Jupyter Notebook mit Frameworks	keine
<b>Weitere Tests</b>	Felix Willrich, Frederik Rieß, Pit Ehlers, Jascha Schmidt	Während des Arbeitens werden weitere Tests möglich sein, bzw. überhaupt erst auffallen.	Jupyter Notebook mit Frameworks	Unterschiedlich

## 2.2. Anhänge

### 1. Gantt Diagramm

Phase	Arbeitspakete	März				April				Mai				Juni			
		4	11	18	25	1	8	15	22	6	13	20	27	3	10	17	24
Projektspezifische Aufgaben	Dokumente erstellen																
	Interview mit Kunden führen																
	Datenstruktur wählen																
	Kommunikationsstruktur festlegen																
	Product Backlog																
	Erfassen der Lern- und Testdaten																
	Bearbeiten und Augmentation der Daten																
	Auslesen der XML-Daten																
	Label für die Daten																
	Einlesen der Daten																
Vorbereitung	Skalieren der Daten																
	Model für das neuronale Netz entwerfen																
	Lernprozess initialisieren																
	Grafische Darstellung der Ergebnisse																
	Sprint Review																
	Product Backlog																
	Batch-Normalization																
	Hyperparameter optimieren																
	Verschiedene Inputgrößen testen																
	Generator benutzen																
Erster Sprint	Augmentation																
	Weitere Tests																
	Bilderrahmenstruktur																
	Ausschneiden der Parkplätze überarbeiten																
	Skript zum Ermitteln der Größe der Bilder																
	Collab einarbeiten																
	Sprint Review																
	Product Backlog																
	Batch-Normalization																
	Hyperparameter optimieren																
Zweiter Sprint	Verschiedene Inputgrößen testen																
	Generator benutzen																
	Augmentation																
	Weitere Tests																
	Bilderrahmenstruktur																
	Ausschneiden der Parkplätze überarbeiten																
	Skript zum Ermitteln der Größe der Bilder																
	Collab einarbeiten																
	Sprint Review																
	Product Backlog																
Dritter Sprint	Batch-Normalization																
	Hyperparameter optimieren																
	Verschiedene Inputgrößen testen																
	Regularization																
	Augmentation																
	Weitere Tests																
	Sprint Review																
	Abgabe																
	Systemdokumentation anfertigen																



# **[Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken]**

## **Sprint 3 Review**

1.	Arbeitspakete.....	2
1.1.	Batch-Normalization .....	2
1.2.	Hyperparameter optimieren .....	6
1.3.	Verschiedene Inputgrößen testen .....	8
1.4.	Augmentation.....	9
1.5.	Regularization.....	11
1.6.	Weitere Tests.....	11
2.	Use-Cases .....	12
3.	Entwicklerreview.....	18
3.1.	Felix Willrich.....	18
3.2.	Frederik Rieß .....	18
3.3.	Pit Ehlers.....	19
3.4.	Jascha Schmidt .....	19

Versionen:

Rev.	Datum	Autor	Bemerkungen	Status
0.1	04.06.2019	Felix Willrich	Erstellen des Sprint Reviews	Abgeschlossen
0.2	06.06.2019	Pit Ehlers	Eintragen der Arbeitspakete & des Reviews	Abgeschlossen
0.3	06.06.2019	Jascha Schmidt	Eintragen der Arbeitspakete & des Reviews	Abgeschlossen
0.4	06.06.2019	Felix Willrich	Eintragen der Arbeitspakete & des Reviews	Abgeschlossen
0.5	06.06.2019	Frederik Rieß	Eintragen der Arbeitspakete & des Reviews + Verbesserungen durchgeführt	Abgeschlossen

## 1. Arbeitspakete

Der dritte Sprint sah vor die Tests aus Sprint 2 und die neu konzipierten Tests in Sprint 3 durchzuführen. Dazu wurden im Product Backlog diverse Pakete übernommen aus Sprint 2 und ein weiteres hinzugefügt.

In Abstimmung mit dem Kunden wird eine Enddokumentation angelegt, welche die Testergebnisse beinhaltet. Alle Ergebnisse werden an den Kunden übergeben, da dieser die Auswertung analysiert und verfeinert.

### 1.1. Batch-Normalization

Batch Normalization ist ein Normalisierungsverfahren, bei dem die Ausgabe eines Layers vor der Aktivierungsfunktion normalisiert wird, so dass der Mittelwert nahe an 0 liegt und die Standardabweichung nahe bei 1. Die Normalisierung wird während des Trainings batchweise berechnet und später werden laufende Mittelwerte, die während des Trainings bestimmt worden sind, verwendet. Unser Standartnetz wurde erweitert, um die Auswirkung von Batch-Normalization zu testen. Folgende Ergebnisse sind zu erkennen:

**Mit Batch-Normalization aktiv:**

```
model = models.Sequential()

model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())

model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
```

Durchschnittliche VAL\_ACC: 0,9901266666666667

**Mit Batch-Normalization teilweise aktiv:**

### Konfiguration 1

```

model = models.Sequential()

model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())

model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

```

Durchschnittliche VAL\_ACC: 0,9883

### Konfiguration 2

```

model = models.Sequential()

model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())

model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

```

Durchschnittliche VAL\_ACC: 0,98839

### Konfiguration 3

```

model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())

model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

```

Durchschnittliche VAL\_ACC: 0,982

### Konfiguration 4

```

model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())

model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

```

Durchschnittliche VAL\_ACC: 0,98966

### Mit Batch-Normalization nicht aktiv:

```

model = models.Sequential()

model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
#model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())

model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()

```

Durchschnittliche VAL\_ACC: 0,98613

### Anmerkung

Da die Anforderung für 30 Testläufen pro Konfiguration erst nachträglich kam, wurden für die Punkte „teilweise aktiv“ und „nicht aktiv“ nur jeweils zehn Testläufe durchgeführt.

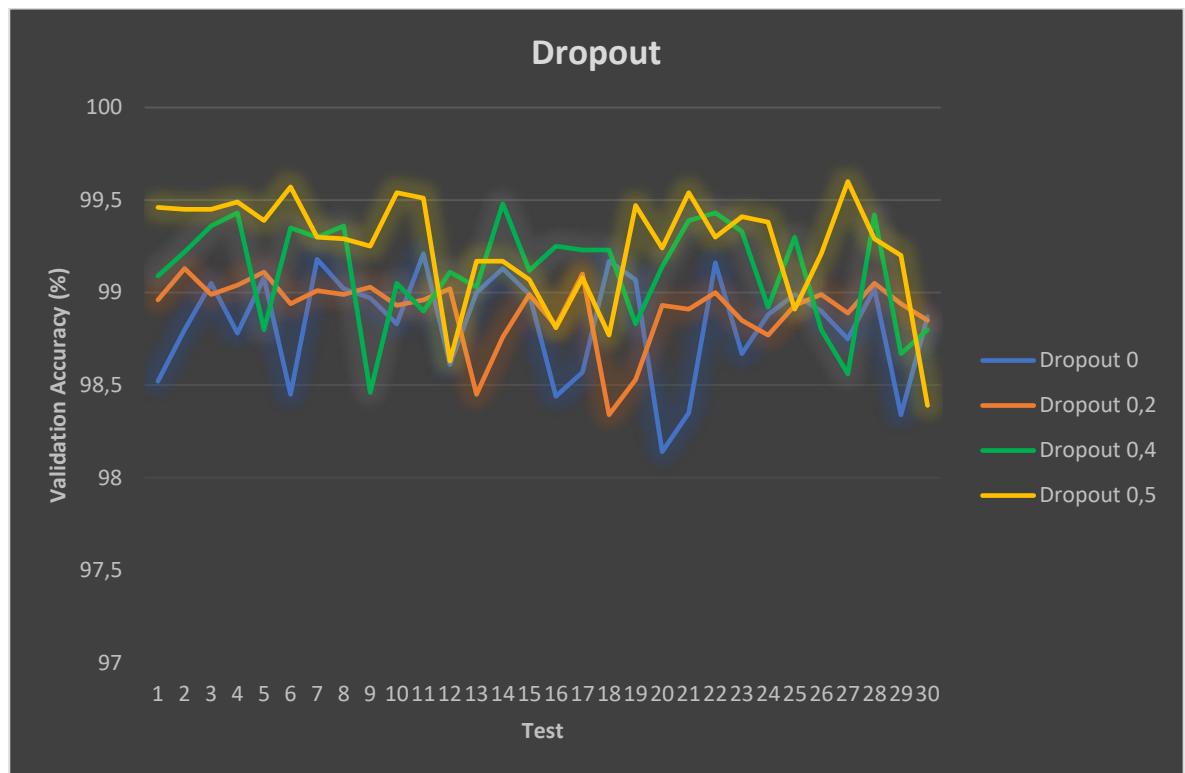
### Ergebnis

Das Aktivieren der Batch-Normalization in allen Layern verbessert die VAL\_ACC des CNN.

## 1.2. Hyperparameter optimieren

Da es zeitlich nicht möglich war, jegliche Hyperparameter zu testen, wurde sich an einem Model orientiert. Zwar wurde auch die Library Hyperopt genutzt, aber es ließen sich nach lediglich einem Testdurchlauf keine konkreten Aussagen treffen. Daher galt die Konzentration hier dem Dropout. Der Dropout setzt zufällig einige der Features eines Layers auf 0, um vor allem Overfitting zu vermeiden. Dabei wird häufig eine Rate zwischen 20% und 50% genommen, die angibt, wie viele Features zu Null werden sollen. Da wir uns auf 30 Durchläufe pro Test festgelegt haben, wurden hier die Werte 0, 0.2, 0.4 und 0.5 gewählt. Es konnte gezeigt werden, dass je kleiner der Dropout ist, desto schlechter ist auch die Validation Accuracy.

Testreihe	Dropout 0	Dropout 0,2	Dropout 0,4	Dropout 0,5
1	98,52	98,96	99,09	99,46
2	98,8	99,13	99,22	99,45
3	99,05	98,99	99,36	99,45
4	98,78	99,04	99,43	99,49
5	99,08	99,11	98,8	99,39
6	98,45	98,94	99,35	99,57
7	99,18	99,01	99,3	99,3
8	99,02	98,99	99,36	99,29
9	98,97	99,03	98,46	99,25
10	98,83	98,93	99,05	99,54
11	99,21	98,96	98,9	99,51
12	98,61	99,02	99,11	98,63
13	99	98,45	99,03	99,17
14	99,13	98,76	99,48	99,17
15	98,99	98,99	99,12	99,07
16	98,44	98,82	99,25	98,81
17	98,57	99,1	99,23	99,08
18	99,17	98,34	99,23	98,77
19	99,07	98,53	98,83	99,47
20	98,14	98,93	99,14	99,24
21	98,35	98,91	99,39	99,54
22	99,16	99	99,43	99,3
23	98,67	98,85	99,33	99,41
24	98,88	98,77	98,92	99,38
25	98,99	98,93	99,3	98,91
26	98,9	98,99	98,8	99,21
27	98,75	98,89	98,56	99,6
28	99,02	99,05	99,42	99,29
29	98,34	98,94	98,67	99,2
30	98,87	98,85	98,8	98,39



Dropout	Validation-Accuracy
0	98,83
0,2	98,91
0,4	99,11
0,5	99,24

### 1.3. Verschiedene Inputgrößen testen

Aufgrund der unterschiedlichen Größen der einzelnen ausgeschnittenen Parkplätze wird getestet, welche eingelesene Größe die größtmögliche Genauigkeit bringt. Da in Sprint 3 das Testverfahren geändert worden ist, wurden 3 Eigenschaften getestet in diesem Sprint. Es werden bis zur Abgabe der Systemdokumentation noch weitere Eigenschaften getestet. Die Analyse der Testergebnisse erfolgt in der Abschlussdokumentation

**Input-Shape:** 124x60

Testnummer	Ergebnis
1	0,9894
2	0,9951
3	0,9921
4	0,9854
5	0,9952
6	0,9943
7	0,9852
8	0,9961
9	0,9870
10	0,9889
11	0,9890
12	0,9896
13	0,9899
14	0,9941
15	0,9860
16	0,9926
17	0,9946
18	0,9867
19	0,9927
20	0,9943
21	0,9922
22	0,9909
23	0,9906
24	0,9864
25	0,9920
26	0,9883
27	0,9873
28	0,9936
29	0,9908
30	0,992

**Durchschnitt:** 0,99074333

**Input-Shape:** 60x54

Testnummer	Ergebnis
1	0,9943
2	0,9899
3	0,9932
4	0,9943
5	0,9827
6	0,9945
7	0,9971
8	0,9822
9	0,9978
10	0,9973
11	0,9989
12	0,9895
13	0,9913
14	0,9692
15	0,9932
16	0,9853
17	0,9924
18	0,9933
19	0,9909
20	0,9850
21	0,9892
22	0,9941
23	0,9902
24	0,9774
25	0,9846
26	0,9923
27	0,9941
28	0,9915
29	0,9879
30	0,9899

**Durchschnitt:** 0,99011667

#### 1.4. Augmentation

Die Augmentation dient dazu die Bilder programmatisch zu verändern und gleichzeitig den Einfluss auf das CNN zu testen. Da in Sprint 3 das Testverfahren geändert worden ist, wurden 3 Eigenschaften getestet in diesem Sprint. Es werden bis zur Abgabe der Systemdokumentation noch weitere Eigenschaften getestet.

Die Analyse der Testergebnisse erfolgt in der Abschlussdokumentation.

**rotation\_range: 45**

Testnummer	Ergebnis
1	0,9856
2	0,9873
3	0,9802
4	0,9867
5	0,9913
6	0,9842
7	0,9913
8	0,9779
9	0,9922
10	0,9871
11	0,9929
12	0,9895
13	0,9915
14	0,9934
15	0,9955
16	0,9899
17	0,9899
18	0,9947
19	0,9906
20	0,9849
21	0,9875
22	0,9907
23	0,984
24	0,9925
25	0,9876
26	0,9956
27	0,9873
28	0,9817
29	0,9828
30	0,9823

Durchschnitt: 0,9882867

**width\_shift\_range: 9**

Testnummer	Ergebnis
1	0,9913
2	0,9888
3	0,9922
4	0,9922
5	0,9908
6	0,9811
7	0,9907
8	0,9918
9	0,9954
10	0,9925
11	0,9888
12	0,994
13	0,9938
14	0,9877
15	0,9886
16	0,9935
17	0,9899
18	0,9929
19	0,9936
20	0,9851
21	0,9869
22	0,9935
23	0,99
24	0,9862
25	0,9909
26	0,9939
27	0,993
28	0,9903
29	0,9928
30	0,9909

Durchschnitt: 0,99077

**horizontal\_flip: True**

Testnummer	Ergebnis
1	0,9894
2	0,9853
3	0,9874
4	0,9882
5	0,9854
6	0,9851
7	0,9883
8	0,9873
9	0,991
10	0,9958
11	0,9861
12	0,9895
13	0,9901
14	0,9834
15	0,9938
16	0,9848
17	0,9871
18	0,9882
19	0,9839
20	0,9955
21	0,9845
22	0,9875
23	0,9906
24	0,9903
25	0,9903
26	0,9876
27	0,9847
28	0,9858
29	0,9878
30	0,9905

Durchschnitt: 0,9881733

Aufgrund von Zeitproblemen, konnten zwei weitere Eigenschaften nur auf 10 Tests durchgeführt werden.

**width\_shift\_range: 15**

Testnummer	Ergebnis
1	0,9929
2	0,9904
3	0,9938
4	0,9933
5	0,9839
6	0,9929
7	0,9905
8	0,9947
9	0,987
10	0,9939

**Durchschnitt: 0,99133**

**rotation\_range: 135**

Testnummer	Ergebnis
1	0,9889
2	0,984
3	0,9918
4	0,9938
5	0,9939
6	0,9862
7	0,9908
8	0,9898
9	0,9915
10	0,9888

**Durchschnitt: 0,98995**

## 1.5. Regularization

Regularization ist eine Funktion, um die Gewichtung der Loss-Funktion zu beeinflussen.

### Mit Regularization:

```
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), input_shape=(60, 60, 3), use_bias=False, kernel_regularizer=regularizers.l2(0.01),
                      activity_regularizer=regularizers.l1(0.01)))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Dropout(0.4))

model.add(layers.Conv2D(32, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(64, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), use_bias=False))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Flatten())
model.add(layers.Dropout(0.4))

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
```

Durchschnittliche VAL\_ACC: 0,8961133333333333

### Ergebnis

Bei der Verwendung der Regularization ist die VAL\_ACC gesunken, sodass die Verwendung nicht sinnvoll ist.

## 1.6. Weitere Tests

Das Arbeitspaket konnte diesen Schritt nicht ausgeführt werden, da die weiteren Tests die gesamte Zeit in Anspruch genommen haben.

Aufgrund der Projektabgabe wird das Arbeitspaket als abgeschlossen betrachtet.

## 2. Use-Cases

Folgende Use-Cases wurden aktualisiert:

ID	Projekt-01
Anforderungstyp	Strukturelle Anforderung
User Story/Use Case:	Geeignete Projektstruktur
Anforderung:	Die Projektordner, -Dateien und -Daten sollen in einer geeigneten Struktur zu finden sein.
Begründung:	Für die Übersicht über das Projekt müssen diese Dateien ordnungsgemäß angelegt werden. Möglichst zu Beginn sollte sich jeder im Klaren sein, wo was zu finden ist.
Abnahmekriterium:	Einigkeit im Team
Anforderer:	Team
Kundenzufriedenheit:	niedrig
Priorität:	mittel
Konflikte:	
Weiteres:	
Historie:	14.03.2019 FW: GitHub Repository angelegt Sprint 2 FW: Diverse Bildpakete im OneDrive angelegt Sprint 3 FW: Diverse Bildpakete im OneDrive angelegt, Testpaket angelegt, Testverzeichnis angelegt

ID	Datenerhebung-02
Anforderungstyp	Performanz
User Story/Use Case:	Augmentation der Bilder
Anforderung:	Die eingelesenen Bilder müssen durch Augmentation bearbeitet werden.
Begründung:	Durch Augmentation sind größere Datensätze mit einer großen Variation gegeben.
Abnahmekriterium:	Ein Bild soll in mehreren Variationen vorhanden sein
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	<p>11.04.2019 FW: Einarbeitung in die Libary „imgaug“, Entscheidung in der Gruppe getroffen zuerst keine Augmentation durchzuführen</p> <p>Sprint 2 FW: Einarbeiten in den ImageDataGenerator, Erste Tests durchgeführt</p> <p>Sprint 3 FW: Tests im neuen Datenmodel durchgeführt</p>

ID	CNN-01
Anforderungstyp	Performanz
User Story/Use Case:	Auswahl der Layer
Anforderung:	Die Anzahl der Layer mit Knoten im CNN muss bestimmt werden.
Begründung:	Durch unterschiedliche Strukturen können unterschiedliche Ergebnisse auftreten.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	normal
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen 13.05.2019 FR: Umbauen des Netzes, kleinere Netze ergeben zurzeit bessere Erfolge 06.06.2019 FR: Es wurde sich an einem Model orientiert, um weitere Tests durchzuführen

ID	CNN-02
Anforderungstyp	Performanz
User Story/Use Case:	Hyperparameter optimieren
Anforderung:	Die Hyperparameter werden vor dem Trainieren des neuronalen Netzes gesetzt und müssen getestet werden.
Begründung:	Verschiedene Hyperparameter sorgen für eine unterschiedliche Genauigkeit und Output.
Abnahmekriterium:	Bestmögliche Genauigkeit
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen Sprint 2 FR: Einarbeiten in die Hyperopt Libary und erste Tests Sprint 3 FR: Hyperopt war nicht nützlich für die Tests, da es zeitlich nicht möglich war. Stattdessen wurden manuell Tests durchgeführt.

ID	CNN-04
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Ergebnisse überprüfen
Anforderung:	Die Daten aus dem CNN müssen mit den vorher errechneten Daten übereinstimmen.
Begründung:	Dies gewährt die Korrektheit des Systems.
Abnahmekriterium:	
Anforderer:	T-Systems
Kundenzufriedenheit:	hoch
Priorität:	hoch
Konflikte:	--
Weiteres:	--
Historie:	<p>08.04-19.04.2019 FR: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen, Ergebnisse sind zunächst in Ordnung, müssen verbessert werden</p> <p>Sprint 3 FW: Die Tests werden nun 30 Mal ausgeführt und in einer grafischen Darstellung angezeigt. Eigentliche Überprüfung der Ergebnisse wird vom Kunden durchgeführt.</p>

ID	CNN-05
Anforderungstyp	Funktionale Anforderung
User Story/Use Case:	Grafische Darstellung
Anforderung:	Die Ergebnisse und Testdaten sollen grafisch dargestellt werden.
Begründung:	Durch die grafische Darstellung sind verschiedene Testergebnisse und -Verläufe besser zu erkennen.
Abnahmekriterium:	Erkennen der Ergebnisse
Anforderer:	Team
Kundenzufriedenheit:	niedrig
Priorität:	niedrig
Konflikte:	--
Weiteres:	--
Historie:	08.04-19.04.2019 FR, FW: Aufbauen des Netzes, mit Einarbeitung in die Theorie, erstes Modell erstellt zum Testen, Ergebnisse sind zunächst in Ordnung  Sprint 3 FW: Darstellung Grafisch in Form von Diagrammen und Graphen

### 3. Entwicklerreview

#### 3.1. Felix Willrich

Ich habe in diesem Sprint mich wieder um die Organisation gekümmert. Gleichzeitig habe ich das Testen der Augmentation der Bilder übernommen. Aufgrund des umgestellten Testverfahrens war es für mich sehr aufwändig alle Tests durchzuführen. Aus diesem Grund versuche ich bis zum Ende der Abgabe der Systemdokumentation weiter zu testen. Die Organisation verlief dieses Mal besser als bei den Sprints zuvor. Die Terminabsprache mit Herr May stellte keine Probleme da. Das Abschlussgespräch wird am 12.06.2019 durchgeführt.

Die Fragen, die aufgekommen sind, wurden innerhalb der regelmäßig stattfindenden Meetings geklärt.

Eine abschließende Bewertung des Projekts wird in der Systemdokumentation durchgeführt.

#### 3.2. Frederik Rieß

Meine Aufgabe in diesem Sprint war das Analysieren und Testen unseres Models in Bezug auf einen unterschiedlichen Dropout. Da das Testen mit verschiedenen Werten relativ viel Zeit in Anspruch nimmt, musste dies an vielen Tagen durchgeführt werden. Hier gab es allerdings keine weiteren Probleme und es konnte festgestellt werden, dass ein höherer Dropout für eine höhere Validation-Accuracy sorgt. Des Weiteren habe ich unseren Datensatz nochmal genauer untersucht und festgestellt, dass in einigen XML-Dateien die Parkplätze falsch angegeben sind. Das heißt, bei einigen XML-Dateien sind Parkplätze fälschlicherweise als "belegt" angegeben anstatt "nicht belegt". Dies wird bei unserer Validierung anschließend aber berücksichtigt.

Die Arbeit im Team verlief eigentlich reibungslos, da wir jeweils getrennt voneinander unsere Tests durchgeführt haben. Falls untereinander Fragen auftauchten, wurde eine Lösung gefunden. Zum Beispiel können unendlich viele Tests durchgeführt werden, die dann aber insgesamt keine zuverlässigen Ergebnisse lieferten. Wir entschieden uns aber, auch in Absprache mit unserem Kunden, für eine qualitative Aussage statt einer quantitativen. Ein weiterer Punkt war, dass unser Standard-Model Bilder nur bis zu einer bestimmten Größe trainieren lassen kann. Dies war ein Problem bei Pit Ehlers, dem ich dies dann erklärt hatte. Aber auch hier wurde letztendlich eine Lösung gefunden.

### 3.3. Pit Ehlers

Aufgaben:

- Mehr Tests (jeden Test 30x) durchführen, um festzustellen, welche Inputgröße der Parkplatzbilder zur höchsten Genauigkeit führen. Um ein genauereres Ergebnis zu erzielen

Vorgehen:

- Wie im Sprint zuvor wird mit Google Colab gearbeitet.
- Es werden nun nicht mehr so viele verschiedene Tests wie möglich durchgeführt, sondern die aus Sprint 2 anscheinend wichtigsten Tests mehrmals durchgeführt, um einen Durchschnittswert zu berechnen.

Ergebnis:

- Es ist Nahezu kein Unterschied, wenn man die Inputgröße ändert.

### 3.4. Jascha Schmidt

Aufgaben:

- Testen des Einflusses von Batch Normalisation auf das CNN
- Testen des Einflusses von Regularizers auf das CNN

Vorgehen:

- Das CNN mit verschiedenen Konfigurationen der Batch Normalisation mehrfach durchlaufen lassen und die Ergebnisse dokumentieren
- Das CNN mit verschiedenen Konfigurationen der Regularizer mehrfach durchlaufen lassen und die Ergebnisse dokumentieren

Probleme:

- da es im Verlauf des Sprints die Auflage gab, dass jede Konfiguration anstatt zehnmal 30x getestet werden sollte, dauerten die Testläufe sehr lange und es war mir zeitlich nicht mehr möglich alle Konfigurationen weitere 20x zu testen.