Ostfalia
Hochschule für angewandte Wissenschaften

[*Team* 15]

Frederik Rieß Pit-Aurel Ehlers Jascha Schmidt Felix Willrich

[*Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken*]

Sprint 3 Review

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 Standartabweichung 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 erweitertet, 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: 0,990126666666667

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

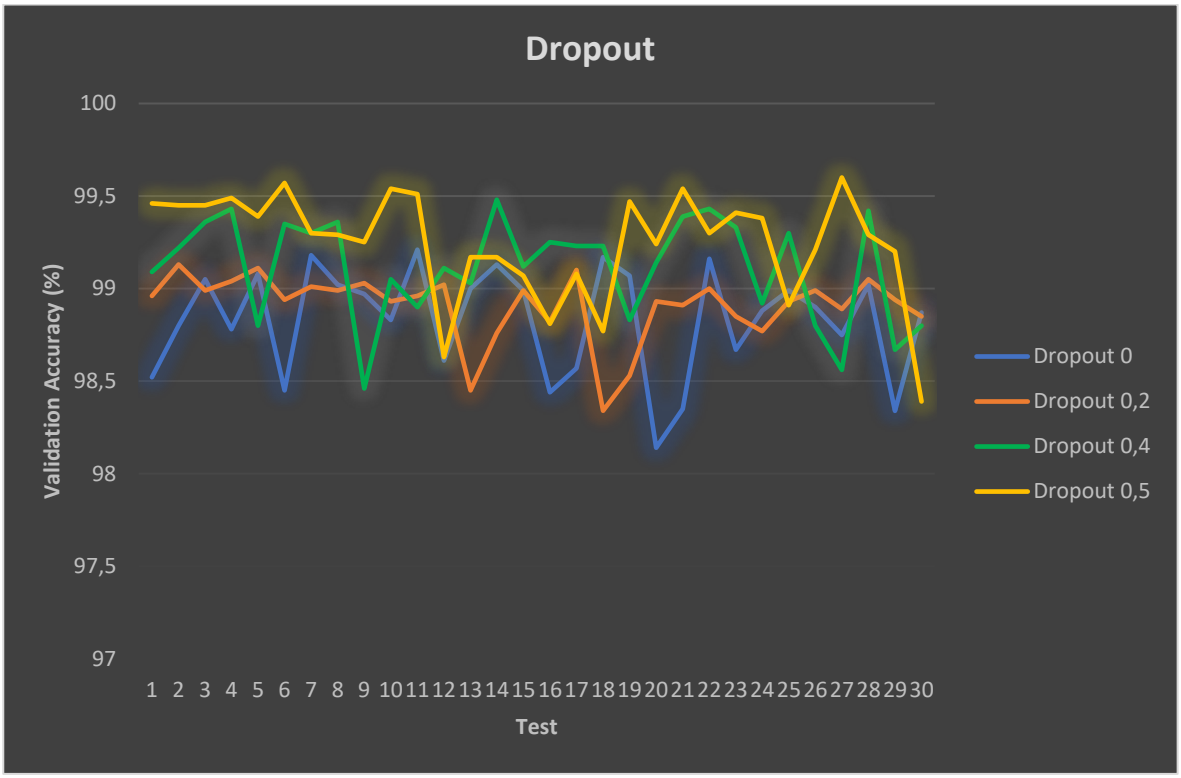
Anmerkung
Da die Anforderung für 30 Testläufen pro Konfiguration erst nachträglich kam, wurden für die Punkte „teilweise aktiv“ und „nicht aktiv“ nur jeweils zehn Testläufe durchgeführt.

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 treffe. 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: 0,896113333333333

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:	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 Sprint 3 FW: Tests im neuen Datenmodel 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 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:	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 Sprint 3 FW: Die Tests werden nun 30 Mal ausgeführt und in einer grafischen Darstellung angezeigt. Eigentliche Überprüfung der Ergebnisse wird vom Kunden durchgeführt.

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. Desweiteren 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 letzten Endes 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 genaueres 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.