Ostfalia
Hochschule für angewandte Wissenschaften

Frederik Rieß Pit-Aurel Ehlers Jascha Schmidt Felix Willrich

[*Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken*]

Sprint 2 Review

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 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:

```
# 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 verschiedensten 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 «Occpuied». 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://cnrpark.it/)

Folgende Pakete wurden angelegt:

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