Ostfalia
Hochschule für angewandte Wissenschaften

[*Team* 15]

Frederik Rieß Pit-Aurel Ehlers Jascha Schmidt Felix Willrich

[*Intelligente Parkplatzerkennung mit künstlichen neuronalen Netzwerken*]

Sprint 1 Review

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://cnrpark.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.

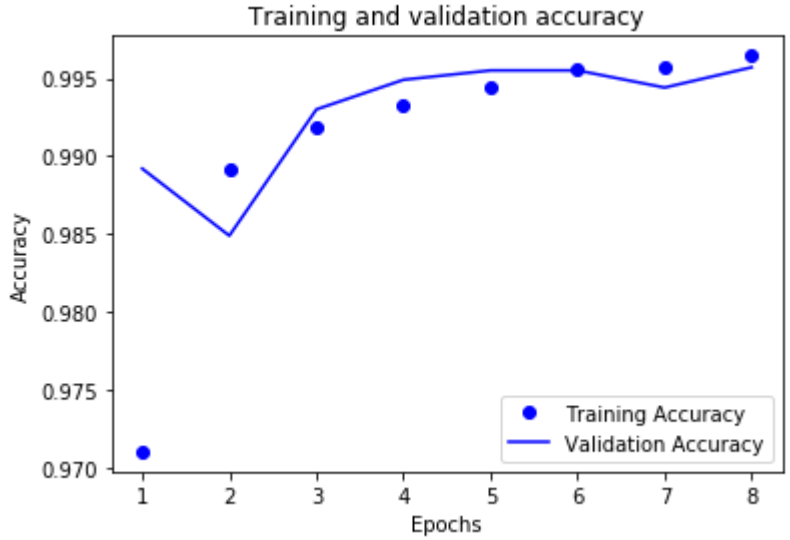
1.2. Programmierung

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 ineinandergreifen 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“ (Recitifed 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 eingebarbeitet. 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.