Sprint Review

# 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.

# Das Convolutional Neural Network

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 sequentiell 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 (ausgeschnitten 100 einzelne Parkplätze) wird eine Genauigkeit von ungefähr 74% erreicht. Diesen Wert gilt es in zukünftigen Sprints zu optimieren (Augmentation, mehr Daten etc.). Das grundlegende Model steht jedoch bereits.