Object Detection API – Anleitung

# Vorbereitung 1: Tensorflow für Python installieren

Falls noch nicht geschehen, mittels pip install tensorflow oder pip install tensorflow-gpu die nötigen Tensorflow-Dateien installieren.

# Vorbereitung 2: Object-Detection API via git clonen

Am Einfachsten ist es, die gesamten Tensorflow-Models von <https://github.com/tensorflow/models> zu klonen. Wir benötigen anschließend nur den Unterordner research/object\_detection. In diesen Ordner sollten alle unsere Dateien kopiert werden und von diesem aus alle Skripte ausgeführt werden.

# Vorbereitung 3: Die Object-Detection-Installationsanweisungen durchführen

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md>

# Schritt 1: Trainings- und Testdatensätze festlegen

Die Datensätze müssen händisch ausgewählt werden und am Besten in einzelne Dateien „train.json“ und „test.json“ aufgeteilt werden.

# Schritt 2: JSON in CSV umwandeln

Die von uns erstellten JSON-Dateien müssen zur Weiterverarbeitung in ein bestimmtes CSV-Format konvertiert werden. Hierzu habe ich das Script „generate\_csv\_from\_json.py“ geschrieben. Dieses verfügt über 2 Parameter:

# python generate\_csv\_from\_json.py --csv=train.csv --json=train.json

Die CSV-Datei wird dabei automatisch erstellt. Falls das Skript mehrmals ausgeführt wird, werden die zusätzlichen Daten einfach an das vorhanden CSV-File angefügt. Es ist also problemlos möglich, aus mehreren JSON-Dateien eine einzige CSV-Datei zu generieren.

# Schritt 3: Orderstruktur erstellen und Bilder kopieren

An dieser Stelle ist es nötig, einen Ordner „seil“ innerhalb des Ordners „object\_detection“ zu erstellen. In diesen neuen Ordner müssen nun alle Test- und Trainingsbilder sowie die beiden CSV-Dateien hinterlegt werden. Es sind keine weiteren Unterordner nötig, alle Dateien können einfach in „seil“ kopiert werden.

# Schritt 4: CSV in TFRECORD umwandeln

Google stellt ein Skript zum Erstellen von TFRECORD-Files zur Verfügung. Dieses habe ich bereits an unsere Anforderungen angepasst und mit neuem Namen versehen:

python generate\_tfrecord\_seil.py --csv\_input=seil/train.csv --output\_path=data/train.record

Das selbe Skript kann dann noch für test.csv und test.record ausgeführt werden.

# Schritt 5: Das eigentliche Training

Da wir nun endlich die beiden TFRecords erstellt haben, kann das Training beginnen. Die TFRecords enthalten dabei bereits alle Label und Bilder.

Für das Training habe ich die beiden Dateien „seil.pbtxt“ und „ssd\_mobilenet\_v1\_seil.config“ vorbereitet. Diese sollten am Besten in den Ordner „training“ verschoben werden. Nun kann train.py über folgenden Befehl ausgeführt werden:

python train.py --logtostderr --train\_dir=training --pipeline\_config\_path=training/ssd\_mobilenet\_v1\_seil.config

Je nachdem wie Leistungsfähig eure GPU oder CPU sind, könnt ihr im .config-File den Parameter batch\_size vor der Ausführung verändern (je höher desto mehr Leistung wird benötigt, aber desto mehr Bilder können in einem Trainingsschritt verarbeitet werden).

Zur Kontrolle des Trainings kann:

python tensorboard.py --logdir=./training

im Verzeichnis „object\_detection“ ausgeführt werden.

Zeitgleich zum Training kann eine Evaluation des Netzes angestartet werden:

python eval.py --logtostderr --checkpoint\_dir=training --eval\_dir=eval --pipeline\_config\_path=training/ssd\_mobilenet\_v1\_seil.config

Um die Daten der Evaluation auszuwerten ist es nötig tensorboard zu Starten und als logdir das eval-Verzeichnis anzugeben.

# Schritt 6: Ende des Trainings

Ein gutes Netz sollte bei dem Parameter total\_loss, welchen man mittels Tensorboard im Trainings-Verzeichnis begutachten kann einen Wert um 1.0 haben (am Besten unter 1.0).

Zudem sollte die Accuracy, wenn man Tensorboard im Eval-Verzeichnis startet möglichst nah an 100% sein.

# Weitere Schritte

In den nächsten Tagen werde ich noch mal eine Anleitung schreiben, wie man das so trainierte Netz dann mit eigenen Dateien am PC testet und wie man das Model in einer Android-App zum Laufen bekommt.