UE 5 – Durchführung

Detektion mit Transfer Learning Methoden

Überblick

In der Vorbereitungsaufgabe habt ihr den Aufbau von Deep Learning basierten Detektionsmodellen und den Transfer Learning Ansatz kennengelernt. Zudem habt ihr euch mit der Einrichtung der Google Colab Umgebung vertraut gemacht.

Das Ziel der praktischen Aufgabe ist die Implementierung eines Deep Learning basierten Detektionsmodells zur Detektion von Verkehrszeichen.

Dafür werdet ihr das TensorFlow Object Detection API (Applikation Programming Interface) verwenden. Das Object Detection API befindet sich im **research**-Ordner des TensorFlow Repository (siehe Abbildung 1). Wichtige Informationen und Hinweise zur Benutzung dieses API sind in dem Unterordner **g3doc** hinterlegt. Die Sammlung der vortrainierten Modelle mit den Angaben zur Laufzeit, mAp-Werten und Ausgaben für die einzelnen Modellarchitekturen (detection_model_zoo.md) befindet sich ebenfalls in diesem Unterordner. Zum Training dieser Modelle wurden folgende Datensätze verwendet: der COCO-Datensatz, der KITTI -Datensatz, der Open-Images-Datensatz, der AVA -v2.1-Datensatz, der iNaturalist-Species-Detection-Datensatz und der Snapshot Serengeti Datensatz.

```
tensorflow
├── models
│   ├── official
│   │   ├── ...
│   └── research
├── ...   ├── object_detection
│         │   ├── data
│         │   ├── models
│         │   ├── samples
│         │   ├── g3doc
│         │   │   ├── detection_model_zoo.md
│         │   │   ├── installation.md
│         │   │   ├── running_locally.md
│         │   │   ├── preparing_inputs.md
│         │   │   ├── evaluation_protocols.md
│         │   ├── model_main.py
│         │   ├── object_detection_tutorial.ipynb
│         │
│         ├── ...
├── ...
```

Abbildung 1: TensorFlow Object Detection API: Ordnerstruktur mit Auflistung von relevanten Ordnern und Dateien

Das Object Detection API hat folgende Vorteile:

- Es ermöglicht die Verwendung von bereits vortrainierten Modellen zur Objektdetektion sowie den Aufbau und das Training von eigenen Modellen.
- Als Detektoren werden State of the Art Region-Proposal-Systeme wie Faster R-CNN und R-FCN, Context RCNN sowie Single Shot Detector (SSD) benutzt.
- Als Merkmalsextraktoren kommen VGG-16, Resnet-101, Resnet-50, Inception v3, Inception v2, MobileNet zum Einsatz.
- Bedingt durch eine einheitliche Implementierung können Detektoren und Merkmalsextraktoren beliebig kombiniert werden. Dadurch kann die Performance der einzelnen Modelle aufgabenspezifisch gesteuert werden.

Aufgabe 1

Schaut euch das Object Detection API Demo, die sich im Unterordner colab_tutorials befindet. Führt dieses Notebook in Google Colab aus.

Aufgabe 2

In dieser Aufgabe solltet ihr die Daten für das Training vorbereiten. Die Daten sollen in TFRecord-Format konvertiert werden. TFRecord-Format ist ein standardisiertes TensorFlow Format zur Datenserialisierung. Unter Datenserialisierung wird das Speichern von Daten in Form von binären Zeichenfolgen verstanden. Zum Konvertieren der Daten ins TFRecord-Format ist eine spezifische Datenstruktur mit Informationen über den relativen Bildpfad, die Klasse und die x- und y-Koordinaten der Begrenzungsboxen erforderlich. Eine Anleitung zum Vorbereiten der Daten findet ihr in den Dateien using_your_own_dataset und preparing_inputs.

Bevor ihr TFRecords generiert, sollt ihr eine label_map.pbtxt-Datei anlegen und in dem **data**-Ordner:

$$tensorflow/models/research/object\_detection/data$$

speichern. Wie so eine pbtxt-Datei aufgebaut ist, könnt ihr hier schauen.

Aufgabe 3

In dieser Aufgabe sollt ihr ein vortrainiertes Modell auswählen und seine Konfigurationsdatei konfigurieren. Zur Auswahl eines passenden vortrainierten Modells können die Modelllaufzeit und mAp-Parameter der zur Verfügung gestellten vortrainierten Modelle aus **detection_model_zoo** herangezogen werden.

Die Konfigurationsdateien sind im configs-Ordner abgelegt:

$$tensorflow/models/research/object\_detection/samples/configs$$

Konfigurationsdateien enthalten fünf wesentliche Komponenten: *model, train_config, eval_config, train_input_config* und *eval_input_config*. Grundsätzlich könnt ihr alle Parameter konfigurieren und der Aufgabenstellung entsprechend anpassen. Für folgende Parameter sind Anpassungen erforderlich:

- num_classes
- num_steps
- input_path
- label_map_path
- num_examples

Aufgabe 4

Nachdem die config-Datei konfiguriert wurde, sollt ihr die Trainingspipeline konfigurieren. Hinweise dafür findet ihr hier.

OPTIONAL: Zur Überwachung des Trainings könnt ihr das Visualisierungstool TensorBoard nutzen.

Verwendet für diese Aufgabe die Daten der German Traffic Sign Detection Benchmark, die ihr bereits heruntergeladen habt.

Zusatzinformationen zu Google Colab

Notebook Willkommen bei Colaboratory