

# Installation

---

(nur für lokales training)

## 1. Tensorflow-gpu

```
conda install -c conda-forge tensorflow-gpu==1.14
```

### 1.2. Cuda Version

Cuda Version überprüfen mit:

```
conda list | grep cud  
nvidia-smi
```

wenn nicht wie in [Cuda Support Matrix](#) entweder nvidia driver oder cuda versionen anpassen.

## 2. Tensorflow Object Detection Api

wie in [Documentation/Tutorial](#) beschrieben:

### 2.1. Tensorflow Models

Tensorflow Model Repository clonen (enthält object\_detection)

```
git clone https://github.com/tensorflow/models.git
```

### 2.2 Protobuf

- Python Version von [Protobuf](#) herunterladen:
- Protobuf installieren:

```
sudo ./configure  
sudo make check  
sudo make install  
sudo ldconfig  
cd models/research  
protoc object_detection/protos/*.proto --python_out=.
```

- Environment Variablen im .bashrc (in letzter Zeile) hinzufügen:

```
export
PYTHONPATH=$PYTHONPATH:/path/to/TensorFlow/models/research:/path/to/Tensor
Flow/models/research/slim:/path/to/TensorFlow/models/research/object_detec
tion
```

## Training (lokal)

---

### 1. vortrainierten Model herunterladen

[Tensorflow Object Detection Model Zoo](#)

### 2. Config File

Das für das Model richtige [Config File](#) herunterladen und folgende Stellen anpasse:

```
num_classes # anzahl der Klassen
```

```
fine_tune_checkpoint "path/to/model_folder/model.ckpt" # pfad zum
heruntergeladenen Model
```

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/label_map.pbtxt"
}
# selbe für eval_input_reader
```

```
eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: N_TEST # entspricht Zeilenzahl aus test.csv
}
```

je nach Model Type können weitere anpassungen vorgenommen werden.

Dann in *data* ordner verschieben.

### 3. Training starten

```
python3 path/to/models/research/object_detection/model_main.py \
--pipeline_config_path=data/MODEL_CONFIG_FILE.config \
--model_dir=OUTPUT_DIR \
--alsologtostderr \
--num_train_steps=N_STEPS_TO_TRAIN
```

## 4. Training Visualisierung

```
tensorboard --logdir OUTPUT_DIR
```

## 5. Trainierten Tensorflow Graph Exportieren

```
python3 path/to/models/research/object_detectionexport_inference_graph.py \
--input_type image_tensor \
--pipeline_config_path data/MODEL_CONFIG_FILE.config \
--trained_checkpoint_prefix OUTPUT_DIR/model.ckpt-NR \
--output_directory DATASET_DIR/exported/
```

erzeugt einen Ordner *exported* der *frozen\_inference\_graph.pb* enthält.

Kann jetzt mit *OpenVino Model Optimizer* für *InferenceEngine* konvertiert werden

## Mit Google Colab trainieren

---

- Das [train\\_object\\_detection\\_in\\_colab.ipynb](#) Notebook in Google Drive hochladen und ausführen.
- *data* ordner mit *train.record*, *test.record* und *label\_map.pbtxt* entweder:
  - in google drive hochladen
  - in remote git repository (z.B. GitLab) hochladen
- die zellen des notebooks schrittweise ausführen.

## TensorFlow Graph für OpenVino in IR Format konvertieren

---

entweder Lokal...

mit ModelOptimizer aus OpenVino Toolkit

```
python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo_tf.py \
--input_model path/to/frozen_inference_graph.pb \
```

```
--output_dir path/to/export/model \
--tensorflow_use_custom_operations_config model_support.json \
--tensorflow_object_detection_api_pipeline_config path/to/pipeline.config \
--input_shape [1,h,w,3] \
--data_type FP16 \
--input_image_tensor \
--output=detection_classes,detection_scores,detection_boxes,num_detections \
--reverse_input_channels
```

- Je nach model das entsprechende model\_support.json file aus */opt/intel/openvino/deployment\_tools/model\_optimizer/extensions/front/tf/* verwenden.  
Bsp ssd:
  - Für Modelle die in Tensorflow v1.14 trainiert wurden *ssd\_support\_api\_v1.14.json*
  - nor konvertieren (ohne training) *ssd\_support.json* verwenden
- *input\_shape* aus config file übernehmen.
- *data\_type* für NCS2 immer FP16

erzeugt *frozen\_inference\_graph.bin* und *frozen\_inference\_graph.xml*, welche für die Inferenz mit OpenVino's InferenceEngine verwendet werden können.

## ... oder mit JupyterNotebook

[openvino\\_convert\\_tf\\_object\\_detection.ipynb](#) öffnen und zellen schrittweise ausführen