Traffic Light Detection using the TensorFlow Object Detection API

This is a quick guide and documentation for the Udacity Capstone project (Self Driving Car nanodegree) on how to train a traffic light classifier using the object detection API from TensorFlow.

The first step is to download the Faster R-CNN saved model weights file from here: http://storage.googleapis.com/download.tensorflow.org/models/object_detection/faster_rcnn_resnet101_coco_11_06_2017.tar.gz.

Other models from state of the art object detection neural networks can be found here: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

The next step is to install TensorFlow Object Detection API locally using the steps from here: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/installation.md

Download the models repo locally and run the commands listed in the guide above. My folder with the repository is named: “models-master”.

Dataset preparation:

The Bosch Traffic Light dataset was obtained from here: https://hci.iwr.uni-heidelberg.de/node/6132

I downloaded the RGB files and extracted them. Then I used a python code to convert the data into TFRecords format that is used for training. The python notebook is here: http://bit.ly/2Dsm16y (link to google drive with the notebook file), and it’s based on the official guideline from here: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/using_your_own_dataset.md .

To avoid the common “*No module named object_detection*” error, make sure you first run these commands from the models/research directory (as stated in the Object Detection API installation guide):

```
protoc object_detection/protos/*.proto --python_out=.
export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim
```

The change directory of your terminal to the python directory and launch Jupyter notebook from the same terminal.

Directory structure:

This is my own directory structure:

TrafficLightDetection/

- -- bosch/train/
  - -- pipeline.cfg
- -- data/
  - -- bosch_test.record
  - -- bosch_config.record
- -- label_map.pbtxt
- -- models-master/research/
  - -- train.py
- -- faster_rcnn_resnet101_coco_11_06_2017

It can be improved but that's how it ended up and how I made it to work in the end. I might change it in the future, but it works fine with the commands listed above the directory structure.

Start training:

Use the following command to start training the model:

python models-master/research/object_detection/train.py --logtostderr --pipeline_config=bosch/train/pipeline.config --train_dir=bosch/train/

TensorBoard:

View logs and training data in TensorBoard by running a new terminal and running this command:

tensorboard --logdir=bosch/train/

The pipeline config file that is given as input for the training file contains the necesarry settings for retraining the network. I've set the total number of steps to a maximum of 10000 steps due to time constrains: on my desktop machine I am able to train a step in 12-16 seconds, meaning that 10K steps would take roughly anywhere between: 33 - 44 hours. This file is based on the official guideline from Object Detection API: https://github.com/tensorflow/models/tree/master/research/object_detection/samples/configs. The config file also contains the path to the TFRecord files used in training and testing and model specific data such as image dimension, width / height stride, dropout or learning rate settings.

After training is completed, the network must be exported. This is done using (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/exporting_models.md):

```
python models-master/research/object_detection/export_inference_graph.py \
    --input_type image_tensor \
    --pipeline_config_path bosch/train/pipeline.config  \
    --trained_checkpoint_prefix bosch/train/model.ckpt-10000 \
    --output_directory output_inference_graph
```

I had errors with “layout_optimizer” when exporting the model. This was the error message: “*ValueError: Protocol message RewriterConfig has no "layout_optimizer*" field. “. It seems to be a common problem: https://github.com/tensorflow/models/issues/2861. The fix for me was to edit the file under: “models-master/research/object-detection/exporter.py” and change the following:

I commented out:
rewrite_options = rewriter_config_pb2.RewriterConfig(
layout_optimizer=rewriter_config_pb2.RewriterConfig.ON)

and changed it to:
rewrite_options = rewriter_config_pb2.RewriterConfig()

Prediction:

The final step is to test and make predictions by loading the saved network. A quick python notebook is available here where all the steps are described: https://drive.google.com/open?id=1QHGTAQsQqu5I_Lsb6l5pePX-PK45dnX5.

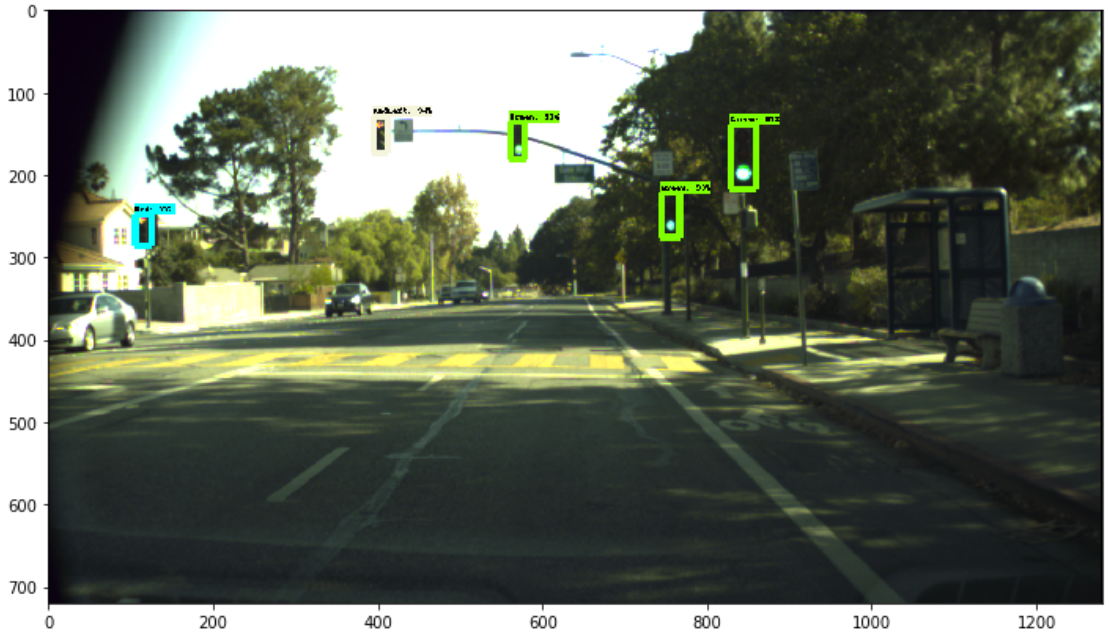
My python notebook is actually based on the official example: https://github.com/tensorflow/models/blob/master/research/object_detection/object_detection_tutorial.ipynb

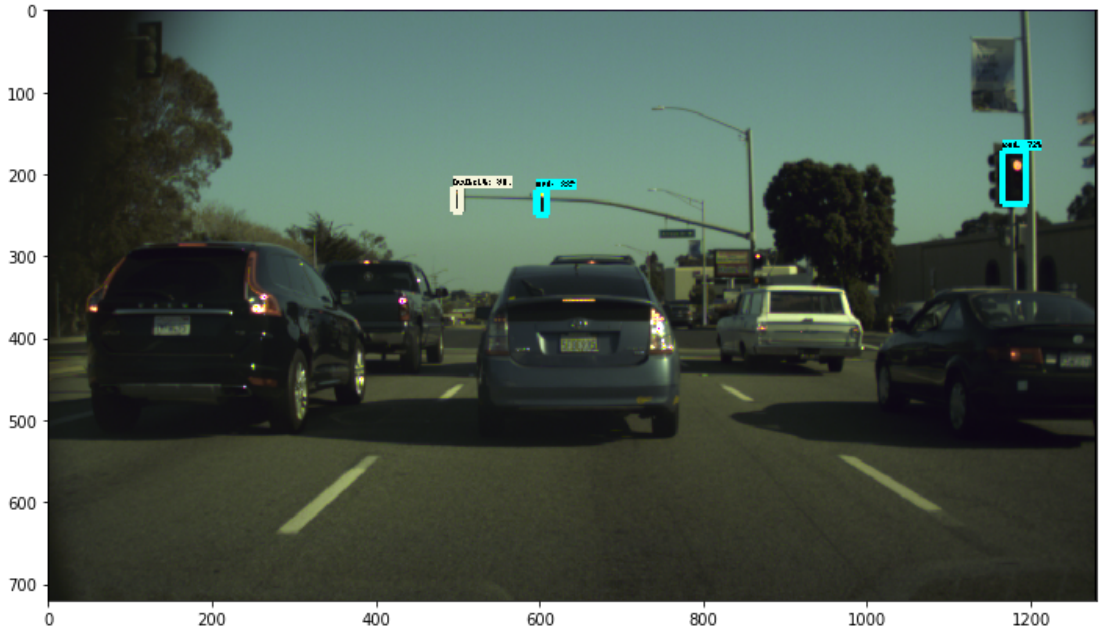
The label map text file is here: https://drive.google.com/open?id=1caCjKGlvs0usuwBH_EF1VAbL98wxDWZV

The frozen graph can be downloaded from here: https://drive.google.com/open?id=1NKorUTi1U2PzGwNncJAdw9gT33I9PRUn.

Test images are RGB: 1280 x 720. Example test images:
1 - https://drive.google.com/open?id=11E7qfvk5evHCjezOELwSnJt-J7soj4sf
2 - https://drive.google.com/open?id=1zobggTI_3rWKsZSUMNIT9Dqujt3BC70Q
2 - https://drive.google.com/open?id=1ctEQEy8-EQ93oPmSJdkUPdS8kZpcAmBq

Prediction results:





Bibliography:

1 - https://codeburst.io/self-driving-cars-implementing-real-time-traffic-light-detection-and-classification-in-2017-7d9ae8df1c58

2 - https://becominghuman.ai/traffic-light-detection-tensorflow-api-c75fdbadac62

3 - Udacity slack channel

Razvan Itu
Email: itu.razvan@gmail.com