***Please note project was done as an individual mainly due to my extensive travel schedule. It would have been unfair of me to join a group and not contribute on time and be un-responsive. ***

This is the project repo for the final project of the Udacity Self-Driving Car Nanodegree: Programming a Real Self-Driving Car. For more information about the project, see the project introduction here.

Native Installation

- Be sure that your workstation is running Ubuntu 16.04 Xenial Xerus or Ubuntu 14.04 Trusty Tahir. Ubuntu downloads can be found here.
- If using a Virtual Machine to install Ubuntu, use the following configuration as minimum:
    - 2 CPU
    - 2 GB system memory
    - 25 GB of free hard drive space

  The Udacity provided virtual machine has ROS and Dataspeed DBW already installed, so you can skip the next two steps if you are using this.
- Follow these instructions to install ROS
    - ROS Kinetic if you have Ubuntu 16.04.
    - ROS Indigo if you have Ubuntu 14.04.
- Dataspeed DBW
    - Use this option to install the SDK on a workstation that already has ROS installed: One Line SDK Install (binary)
- Download the Udacity Simulator.

Docker Installation

Install Docker

Build the docker container

```
docker build . -t capstone
```

Run the docker file

```
docker run -p 4567:4567 -v $PWD:/capstone -v /tmp/log:/root/.ros/ --rm -it capstone
```

Usage

1. Clone the project repository

```
git clone https://github.com/udacity/CarND-Capstone.git
```

2. Install python dependencies

```
cd CarND-Capstone
pip install -r requirements.txt
```

3. Make and run styx

```
cd ros
catkin_make
source devel/setup.sh
```

```
roslaunch launch/styx.launch
```

4. Run the simulator

Real world testing

1. Download training bag that was recorded on the Udacity self-driving car (a bag demonstraing the correct predictions in autonomous mode can be found here)
2. Unzip the file

```
unzip traffic_light_bag_files.zip
```

3. Play the bag file

```
rosbag play -l traffic_light_bag_files/loop_with_traffic_light.bag
```

4. Launch your project in site mode

```
cd CarND-Capstone/ros
roslaunch launch/site.launch
```

5. Confirm that traffic light detection works on real life images

Project Details:

Project was implemented in the following steps (and as advised in the project details https://classroom.udacity.com/nanodegrees/nd013/parts/6047fe34-d93c-4f50-8336-b70ef10cb4b2/modules/e1a23b06-329a-4684-a717-ad476f0d8dff/lessons/462c933d-9f24-42d3-8bdc-a08a5fc866e4/concepts/455f33f0-2c2d-489d-9ab2-201698fbf21a)

1. Waypoint Updater Node (Partial): Complete a partial waypoint updater which subscribes to `/base_waypoints` and `/current_pose` and publishes to `/final_waypoints`.
2. DBW Node: Once your waypoint updater is publishing `/final_waypoints`, the `waypoint_follower` node will start publishing messages to the `/twist_cmd` topic. At this point, you have everything needed to build the `dbw_node`. After completing this step, the car should drive in the simulator, ignoring the traffic lights.

   >> After the above steps the car moved along the track but not responding to any traffic lights.

3. Traffic Light Detection: This can be split into 2 parts:
   1. Detection: Detect the traffic light and its color from the `/image_color`. The topic `/vehicle/traffic_lights` contains the exact location and status of all traffic lights in simulator, so you can test your output.
   2. Waypoint publishing: Once you have correctly identified the traffic light and determined its position, you can convert it to a waypoint index and publish it.
4. Waypoint Updater (Full): Use `/traffic_waypoint` to change the waypoint target velocities before publishing to `/final_waypoints`. Your car should now stop at red traffic lights and move when they are green.

Design:

Based on research there were many possibilities in implementation. In the simulator mode, one can even implement the traffic light detection without DL. However, I wanted to keep both the simulator and the site implementations based on the same approach.

a. Simulator mode:

Mobilened+SSD classifier has been used for the model training. The SSD mobile net coco frozen graph has been used as initial graph for transfer learning. This is based on the learning provided in https://classroom.udacity.com/nanodegrees/nd013/parts/6047fe34-d93c-4f50-8336-b70ef10cb4b2/modules/595f35e6-b940-400f-afb2-2015319aa640/lessons/69fe4a9c-656e-46c8-bc32-aee9e60b8984/concepts/4ab7a82d-2280-4e44-8b75-9a88b82fa8bb

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

Udacity AMI was used as starting point for retraining the dataset. Simu images were annotated and used for training and validation.

Frozen graph output was saved as frozen_inference_graph_sim.pb

b. Site mode:

Several more accurate classification methods were researched and finally mobilenet was chosen for its speed and familiarity. In site mode rosbag images were exported and tagged for training. The starting point for the retraining (https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/image_retraining) was based on mobilenet. Bosch dataset was downloaded and augumented with the rosbag data.

Testing:

1. Testing with simulator

   Works correctly with little bit of latency. Unfortunate

   Please see link https://youtu.be/VtYj4Jv_OhA

2. Testing with Carla:

Correctly identifies traffic lights.

Log output sample:

[rosout][INFO] 2017-10-22 11:45:24,842: Traffic Light: RED

[rosout][INFO] 2017-10-22 11:45:24,843: Traffic light detected, waypoint number: 43

[rosout][INFO] 2017-10-22 11:45:24,942: Traffic lights classes scores: [1.02071150e-03 8.96993931e-03 9.90009308e-01 4.67998751e-09]

[rosout][INFO] 2017-10-22 11:45:24,943: Traffic Light: RED

[rosout][INFO] 2017-10-22 11:45:24,943: Traffic light detected, waypoint number: 43

[rosout][INFO] 2017-10-22 11:45:25,041: Traffic lights classes scores: [1.10378151e-03 7.43961185e-02 9.24473286e-01 2.66935986e-05]

[rosout][INFO] 2017-10-22 11:45:25,041: Traffic Light: RED

[rosout][INFO] 2017-10-22 11:45:25,042: Traffic light detected, waypoint number: 43

[rosout][INFO] 2017-10-22 11:45:25,158: Traffic lights classes scores: [3.13197486e-02 2.06825603e-02 9.47986484e-01 1.13088736e-05]

[rosout][INFO] 2017-10-22 11:54:17,516: Traffic lights classes scores: [9.84171212e-01 6.00112653e-05 1.11606102e-02 4.60807886e-03]

[rosout][INFO] 2017-10-22 11:54:17,518: Traffic Light: GREEN

[rosout][INFO] 2017-10-22 11:54:17,519: Traffic light detected, waypoint number: 43

Credits:

1. https://medium.com/@anthony_sarkis/self-driving-cars-implementing-real-time-traffic-light-detection-and-classification-in-2017-7d9ae8df1c58
2. https://hci.iwr.uni-heidelberg.de/node/6132
3. https://discussions.udacity.com/t/survey-traffic-lights/342249?u=drivewell
4. https://github.com/bosch-ros-pkg/bstld
5. https://github.com/Kairos-Automotive/TL-detection-segmentation