# Object-Detection-in-an-Urban-Environment

Zifeng Wu

Github link: <https://github.com/zwu620/Object-Detection-in-an-Urban-Environment.git>

## Project Overview

This project aims to have students to train a convolutional neural network (CNN) based on a pre-trained network to detect and classify objects using data from the Waymo Open Dataset. The CNN model is used to identify cars, pedestrians, and cyclists from the provided dataset containing images of urban environments with annotated cyclists, pedestrians, and vehicles. The object detection is an important component of self driving car systems because this is the "eye" of a self-driving car so it can “see” the surrounding and therefore take the correct action. For example, slow down if there is a preceding vehicle or stop when there is a pedestrian crossing the road.

## Set up

Had some issues to set up on my local computer. This project was done in the workspace provided by Udacity. Due to the space constraints and resource exhausted issues. The results are limited in terms of 1) the number of steps for the training and 2) the number of checkpoints can be included in the evaluation.

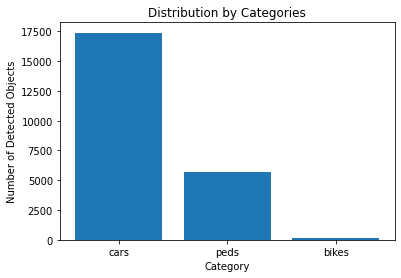
# Dataset

## Dataset Analysis

The provided Waymo dataset includes a variety of environment. Examples include 1) very dense downtown including heavy vehicle and pedestrian volumes, 2) low-volume minor arterial or local collectors, 3) high-volume freeway. Below are example images corresponding to the three environments. A total of 10 images were saved to the folder: workspace/data/10\_example\_images. The weather conditions range from strong sunlight to cloudy or rainy conditions. Images were taken both in day and night.

|  |  |  |
| --- | --- | --- |
| 1. Image(1)   Image(1) | 1. Image(2)   Image(2) | 1. Image(6)   Image(6) |

Cyclists are not as commonly observed as vehicles and pedestrians, which is confirmed by the bar graph below:



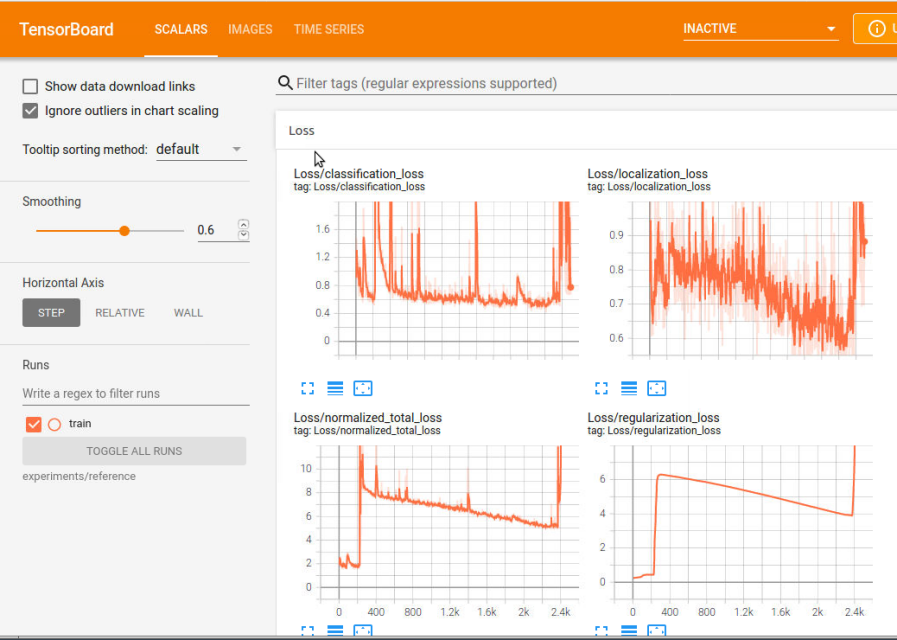
# Cross Validation

The provided Waymo dataset has a total of 100 tfrecord files. The percentages used to split the dataset into training, validation and testing sets are 80%, 15%, and 5%, respectively.

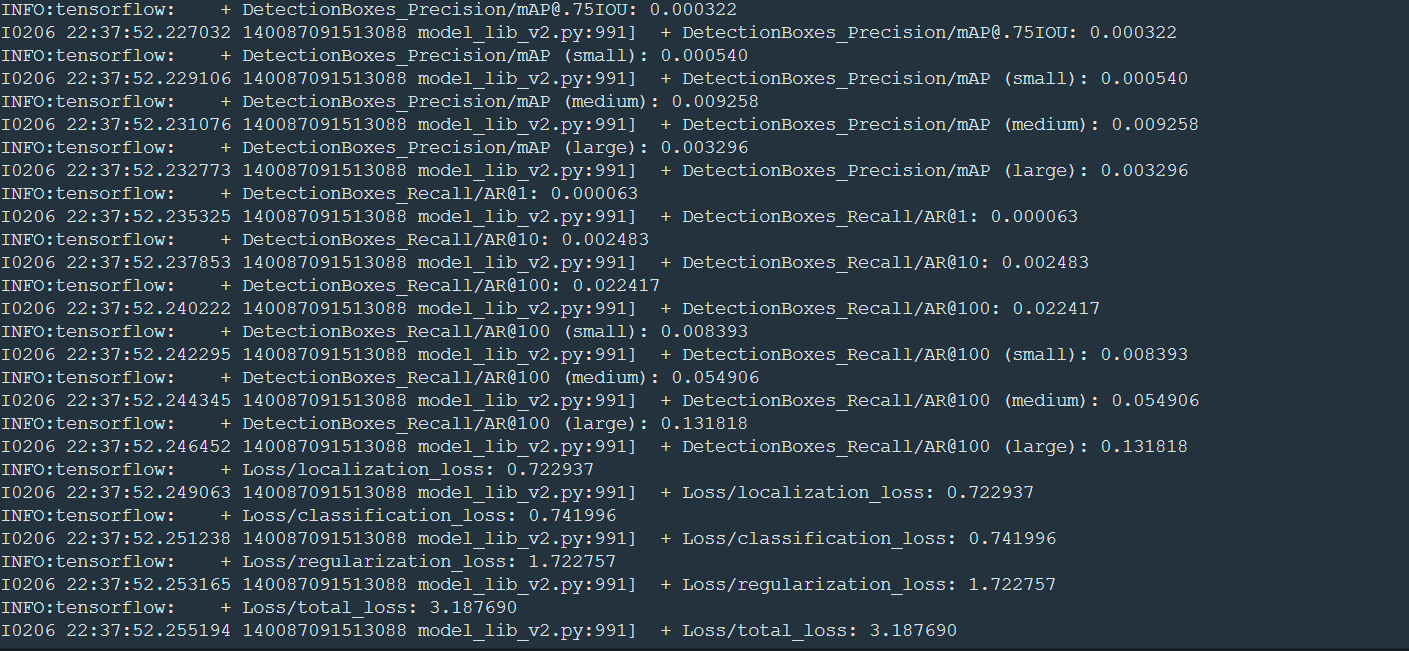
# Training

## Reference Experiment

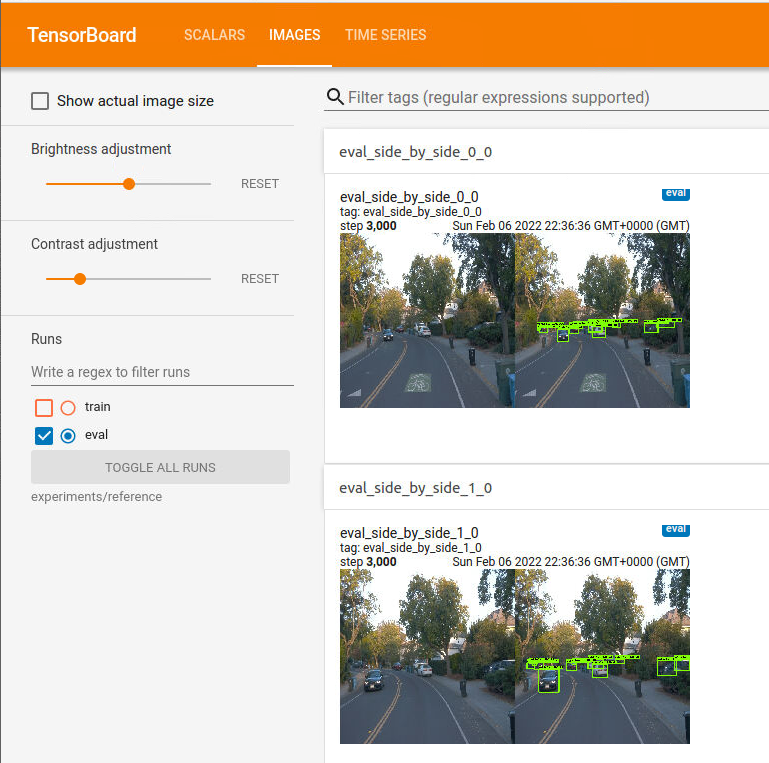
The reference experiment was conducted using the config file generated by directly running edit\_config.py. The total\_steps and warmup\_steps were set as 2500 and 200. The training loss graph is as below:



The orange line represents the training loss and the total loss for training sets appear to decrease but is very unstable and doesn’t seem to converge to an acceptable level. The spike toward the end of the graph is due to a broken run. When I restarted the run, it started over from ckpt-1 and resulted in a weird data point. I went back to check the validation run using a previously stored check point file but it didn’t result in anything valuable. The mAP metrics are almost 0, as shown below:



This is consistent with the resulting image as show below. Even when the step gets to 3000, the image on the left is still showing 0 box, indicating the model is not well trained.



I also did a test run - using the model to generate the animation for a night-time condition (i.e.,

segment-12200383401366682847\_2552\_140\_2572\_140\_with\_camera\_labels.tfrecord).

The animation shows no detected vehicles at all for the dark environment.

## Improvements Over Reference

Based on the findings from the reference experiment. The problems need to solve include: 1) the total loss for training is not converged after 3000 steps, 2) the model didn’t work under dark environment.

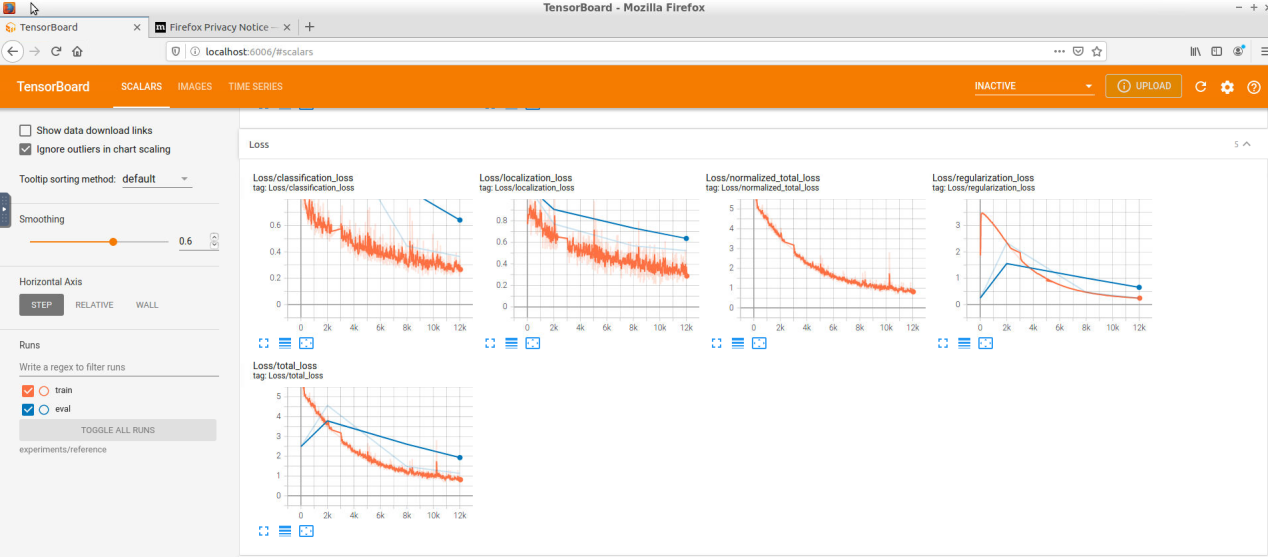
To address the problem 1): the batch\_size was increased from 2 to 4, and the total steps total\_steps and warmup\_steps were increased to 25,000 and 2,000. (Please note the results presented here is for steps ended at 12,000 because of an error indicating the resources and spaces on the workspace are not sufficient to run more steps. Also, the batch\_size can’t be further increased due to similar errors.)

To address the problem 2), additional augmentation options were applied to the model to improve the performance. Two augmentations were added in the config file: random\_adjust\_brightness and random\_adjust \_contrast.

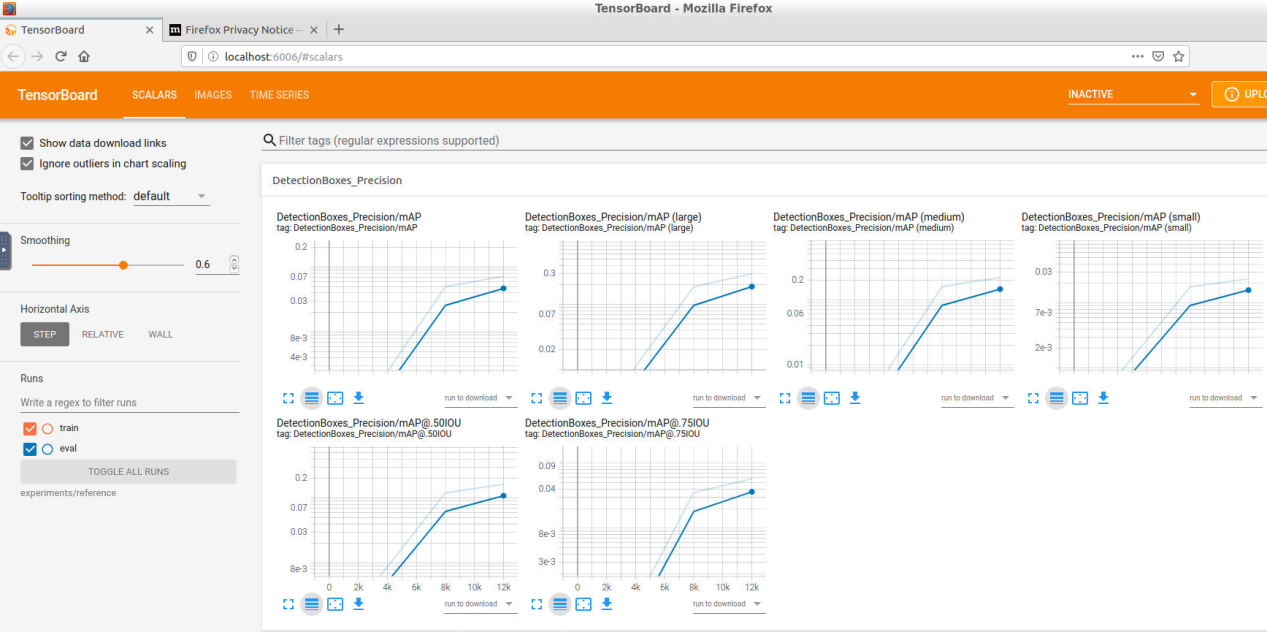
Due to the limited constraint, only the training and validation for the improved CNN model are presented in the figure below. The training loss is shown as an orange line and the validation loss is as an blue line. The total loss was lowered from 3.7 at step 2k to 0.8 at step 12k. This confirmed that the reference experiment has a total\_step of 2.5k is too low.

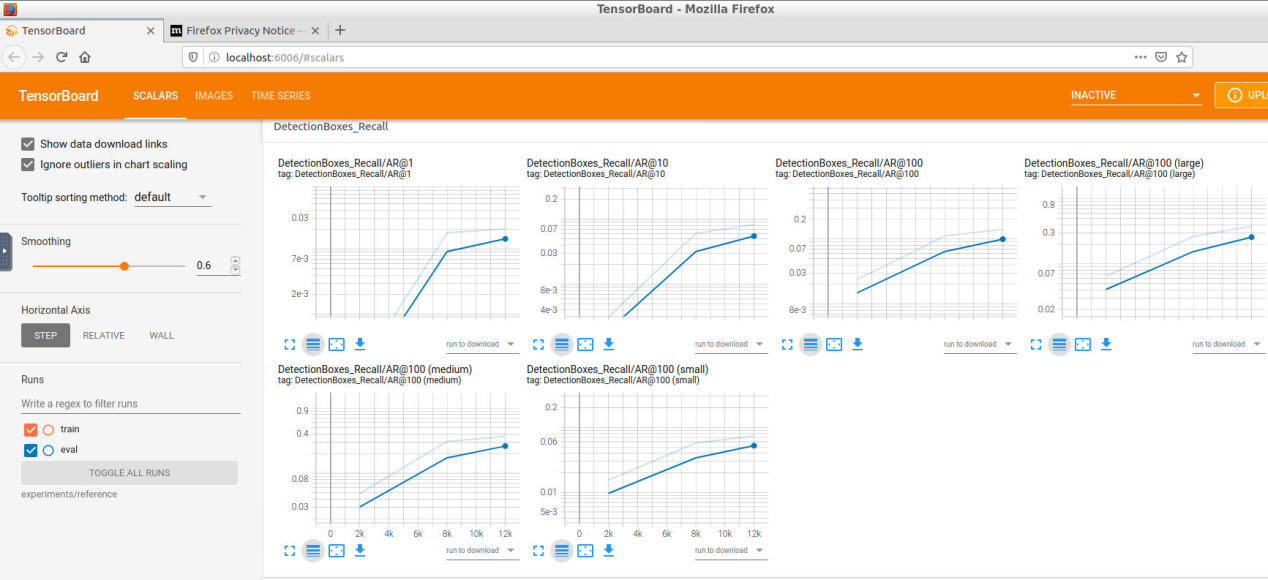
Due to the limited resources in the workspace, only four checkpoints can be included for the validation curve below.

Images of the improved model:



The resulting precision and recall charts of the improved model are shown below:





From the above graphs, the improved model is shown as dark blue lines. The precision of the improved model increased from 0.0 to 0.07 at step 12k, measured as mAP. The improved model increased the both precision and recall measures compared to reference model.

The same testing was performed and the improved model is able to detect vehicles in night-time environment. However, when it’s very dark, the model can only detect the wheels of vehicles because those reflect the front lights and are brighter than other parts of the vehicles. Also, the model can’t detect the vehicles from opposing vehicles with front light on.

In addition, a day-time condition was also tested and the improved model can detect vehicles in its surrounding but can’t capture the vehicle with with large distance from it.