Object Detection in an Urban Environment

Zihan Zhang

Project Overview

This project is aiming for students to practice all the knowledge learnt from the first part of the self-driving car nano-degree program. In the current self-driving car technology stack, object detection on the video caught from the camera is a critical part of the perception. The result can help self-driving cars to be aware of the environment around them and make corresponding actions according to motion planning.

Setup

Due to some issues related to the bounding boxes in Udacity pre-downloaded data. It implicitly becomes a requirement to run the starter code in the provided docker container. A computer with a GPU is used to successfully run all the analysis depending on tensorflow object detection API.

Exploratory Data Analysis

By exploring the images randomly. I have the following observations:

1. Some images contain lots of vehicles, while some images contain no detected objects.
2. Some images are from night time while most others are from daytime.
3. I did not notice any cyclists.
4. All images seem in high quality, rarely blur or rotated.

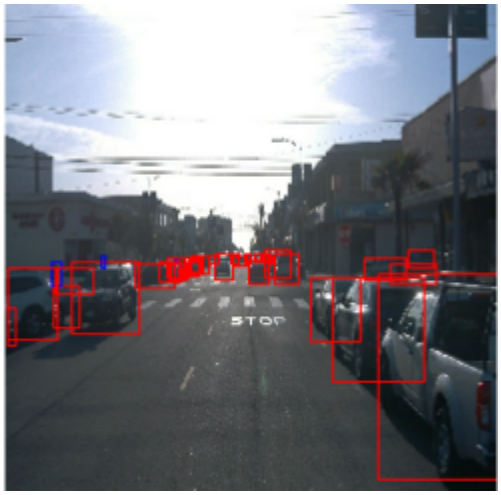

Figure 1

FIgure 2

I also performed additional analysis, and found the following points:

1. Almost no cyclists are detected
2. The 500 images have a balanced distribution of the total count of detected objects. But the majority of detected objects in most crowd images are vehicles.
3. According to the brightness analysis, it seems that the majority of images were taken during daytime, but non-neglectable amount of images were taken when the light was pretty low, at night.

Cross Validation

Following the experience, I choose to give 70% of the dataset to training data, 20% for validation, 10% for testing.

Training

Reference Experiment

During the warm up period, the training process started with a very low learning rate, and the loss function result is very unstable, fluctuates a lot. The training process then keeps increasing the learning rate until it peaks short after the ending of warm up. After warming up, everything seems on the right track, it comes with decreasing loss and decreasing learning rate as it progresses. Eventually come to the point where the learning rate reaches zero. The tensor board monitored graph can be found below:

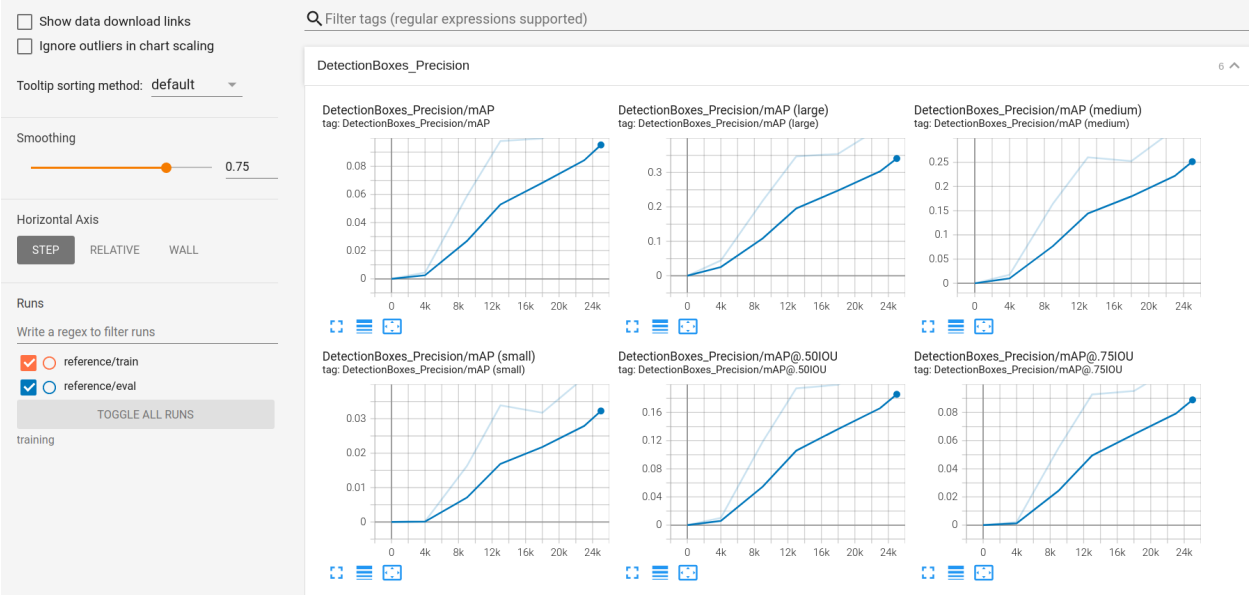

Figure 3

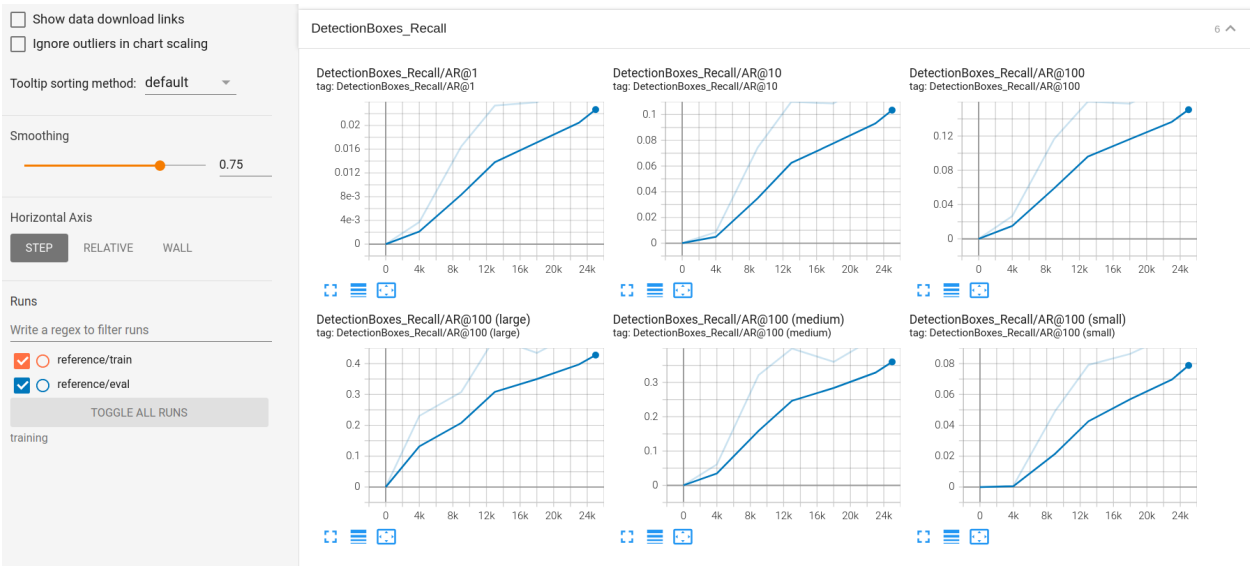

Figure 4

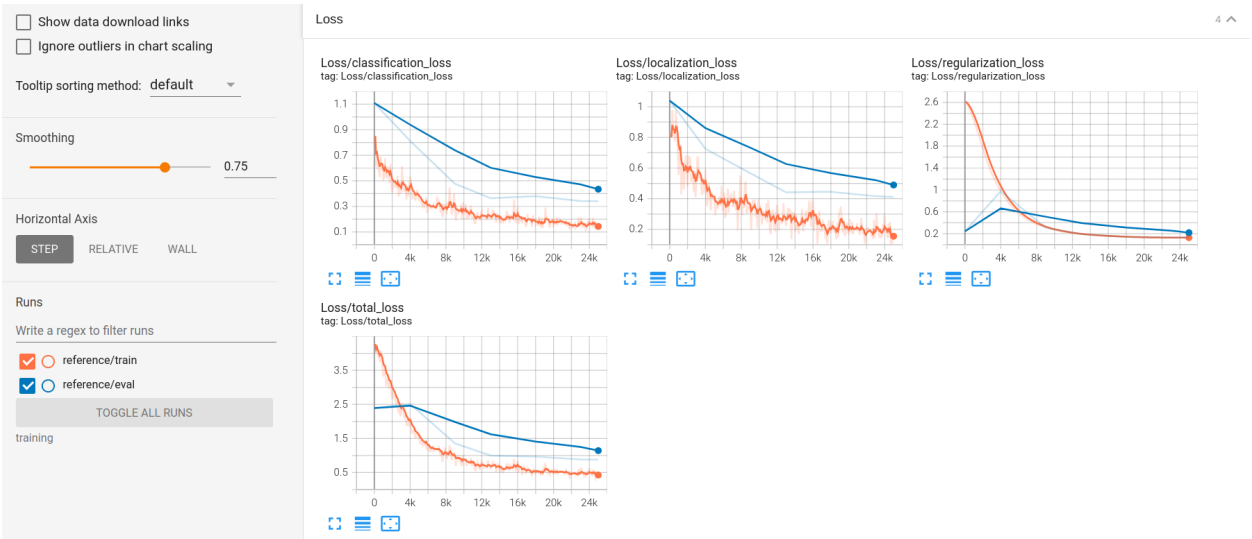

Figure 5

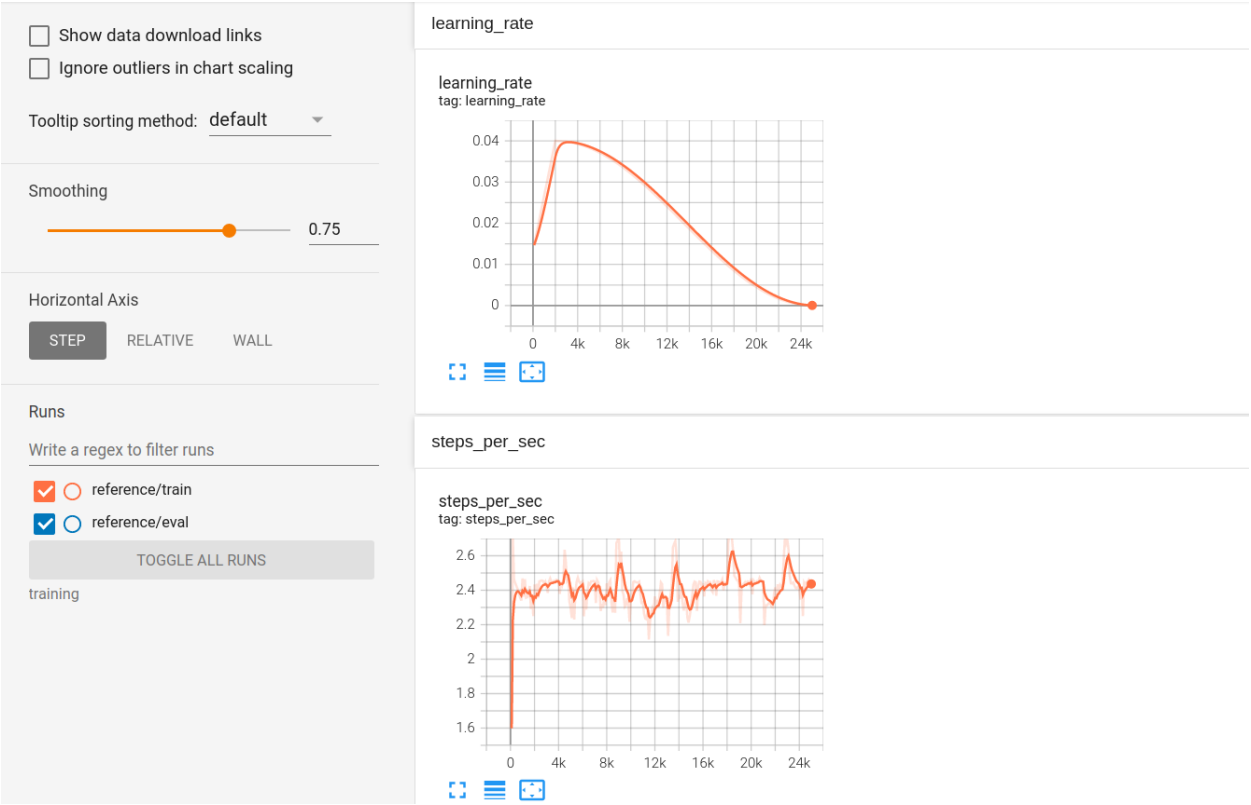

FIgure 6

From the result, we can see that the shape of the loss curve and learning rate curve is great. I noticed that the evaluation's loss is higher than the training session's loss. That is expected as the training may be a little overfitted. And another issue is about the learning rate. In my opinion, the initial learning rate in the warming up period is too low.

Note that due to some errors, the evaluation script cannot be run successfully. So I use a workaround, and run the evaluation without GPU, the speed is kind of slow, that is the reason why only few checkpoints were evaluated.

Improved Experiment

Augmentation exploration

One way to reduce overfitting is to increase the dataset through augmentation. In the config file, I tried the several options like "random_adjust_brightness" and "random_jitter_boxes", when the property value is at maximum value, the result looks not good, like the image in daylight looks too bright, and the image in night time looks almost completely dark. Use those options, but keeping the property value within a mild level seems reasonable.

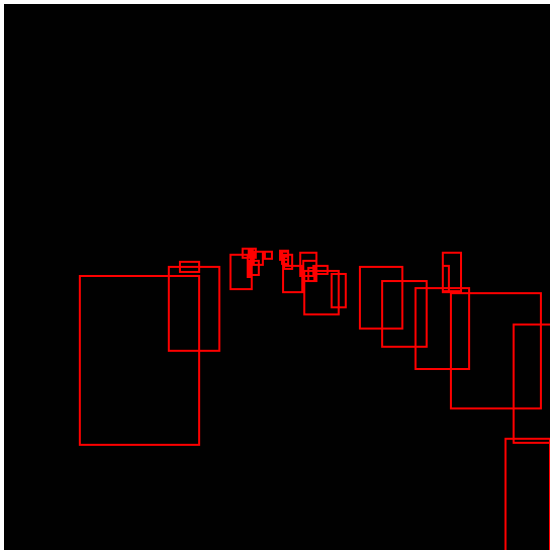

Figure 7

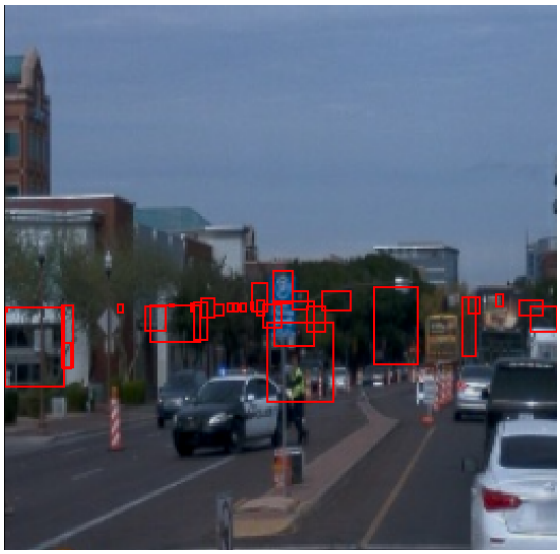

Figure 8

Like in figure 7, the randomly reduced brightness is too much, making the image completely dark; and in figure 8, the randomly shifted bounding box is too far away from the ground truth object.

Also tried the **albumentations** library for the image augmentation. It does provide more freedom from the engineer.

Other parameters

Regarding the other available options, I tried to use adam optimizer with exponential learning rate decay instead of momentum optimizer with cosine decay. The learning rate in that run is constant until the end, and the corresponding loss function chart does not look good.

In my final run, I ended up using several more augmentation options with default value, and continued to use the momentum optimizer, but with a higher warming up learning rate. The chart of the final run is below:

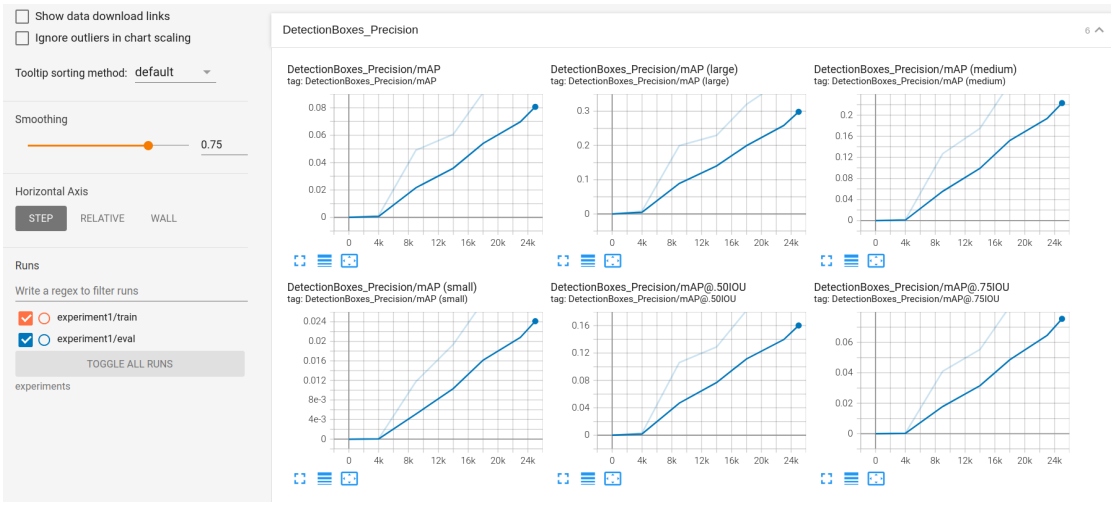

Figure 9

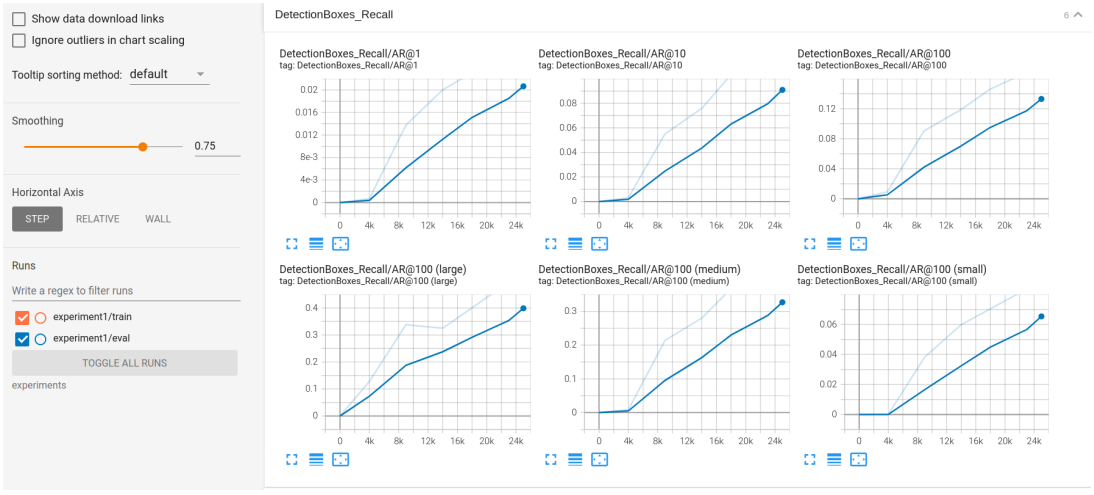

Figure 10

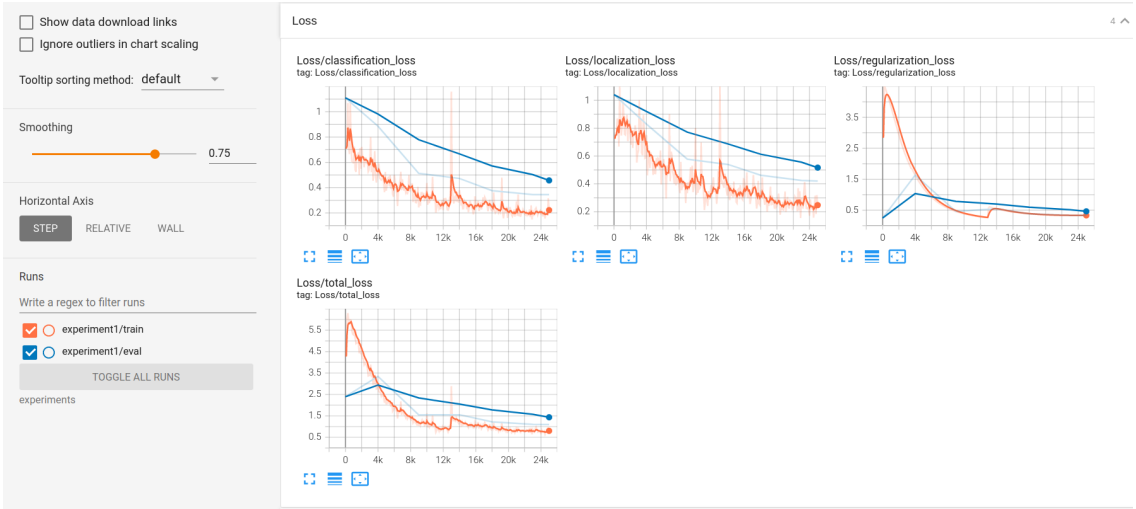

Figure 11

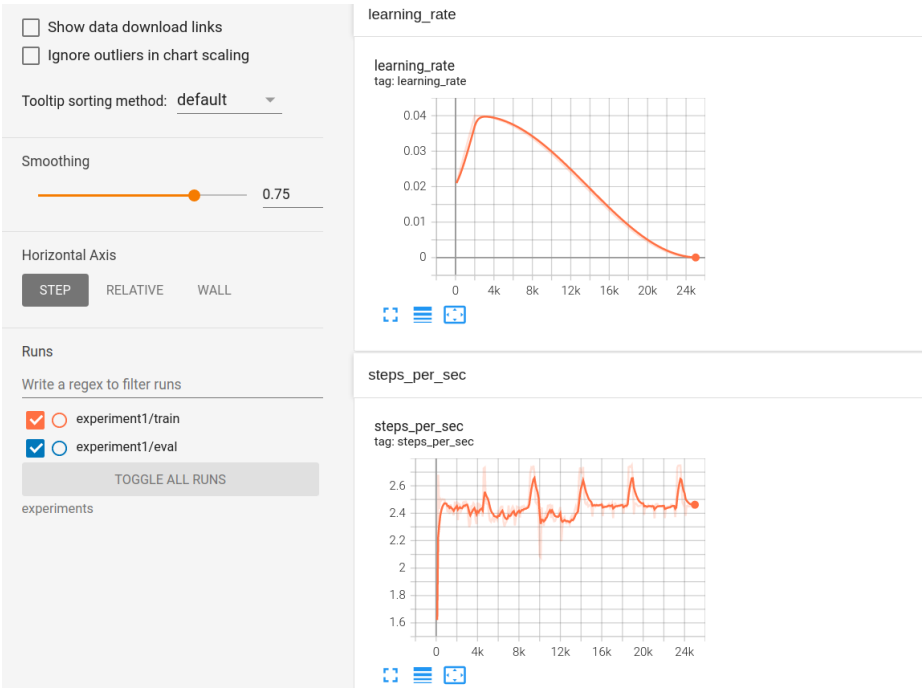

Figure 12

In the final run, the difference between training session and evaluation session did reduce, which is an improvement. And the final model was used to generate an animation video.