Project 1 - Object Detection in an Urban Environment

Project Overview

In this project, the main goal is detecting objects in urban environment using tensorflow object detection api in order to classify different classes.

Set up

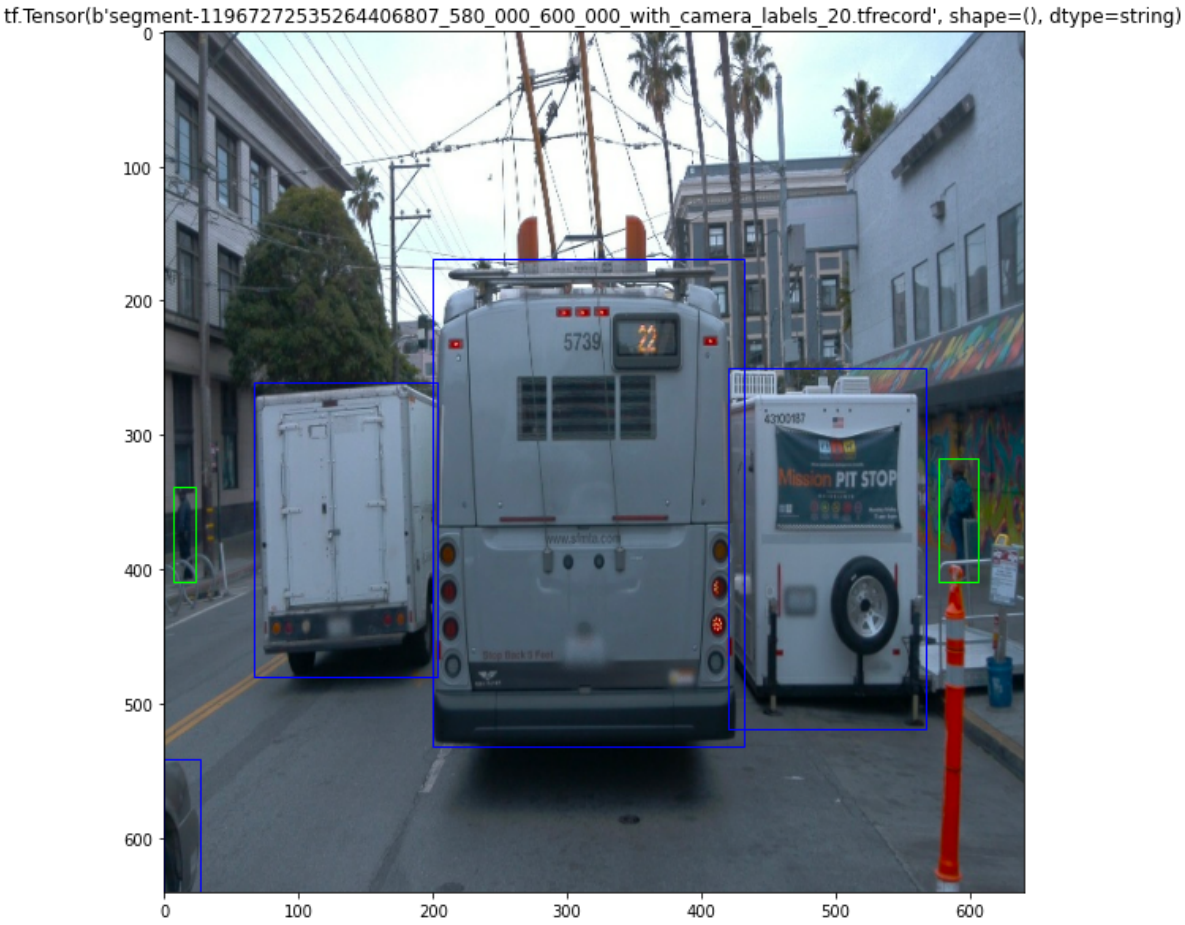
The VM provided by Udacity was used for this project. So there is no extra installation etc.

Dataset
Dataset Analysis
I've started project with the exploring data and labels.

Actually, the frames from day time are much more than the night time frames. Dataset includes frames from different weather conditions like foggy, rainy, days and nights etc.

Below I list some of the situations I have encountered.

- There are mislabeled parts in frames. The box on the right labeled as car, which I don't understand, is actually not a car.

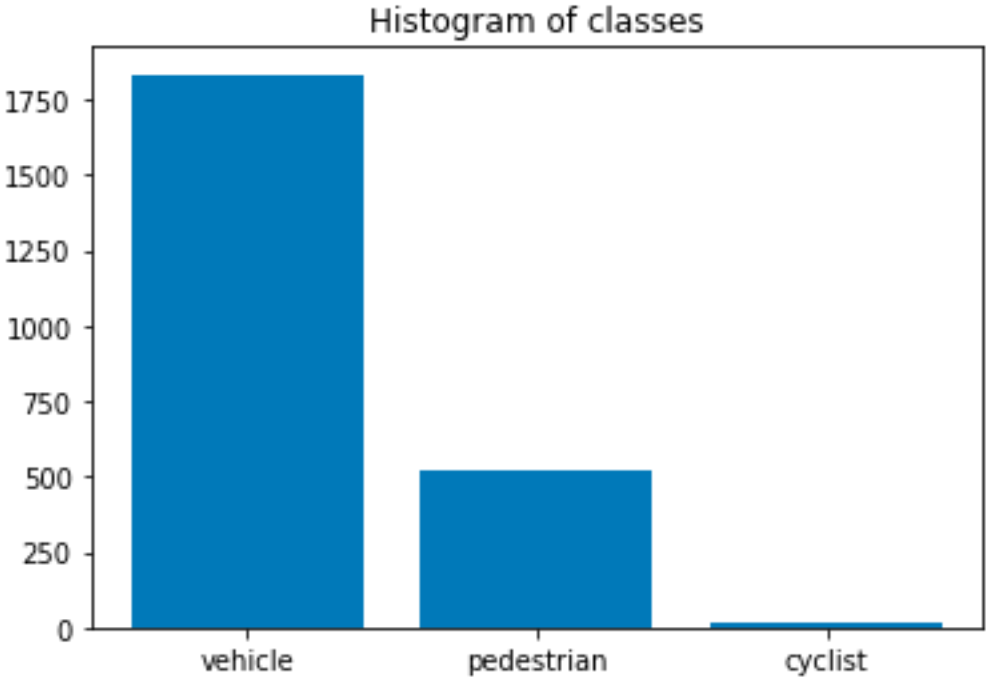


- In some frames, there is no cars, pedestrians or cyclists.

- In the image below, for gray car bounding boxes are displayed two times and probably with different confidince levels.

tf.Tensor(b'segment-10664823084372323928_4360_000_4380_000_with_camera_labels_140.tfrecord', shape=(), dtype=string)

The class imbalance can be shown clearly from the distribution of car, pedestrian and cyclist classes:




```
Total number of cars: 1833 Total number of pedestrian: 521 Total number of cyclists: 13
```

As you can see, cyclists rarely come across in frames. For this histogram 100 different random frame selected and the distribution of classes observed.

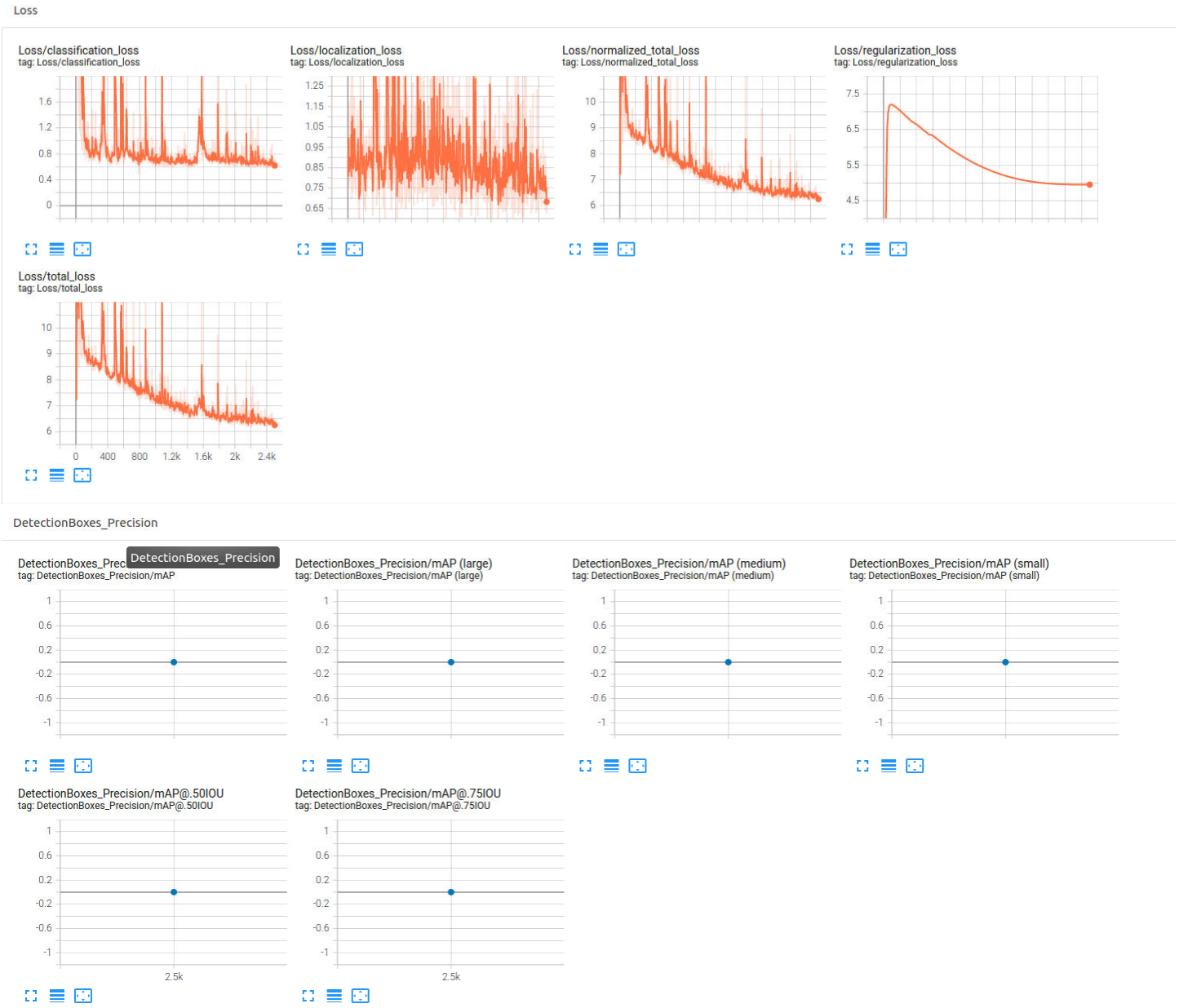
Cross Validation

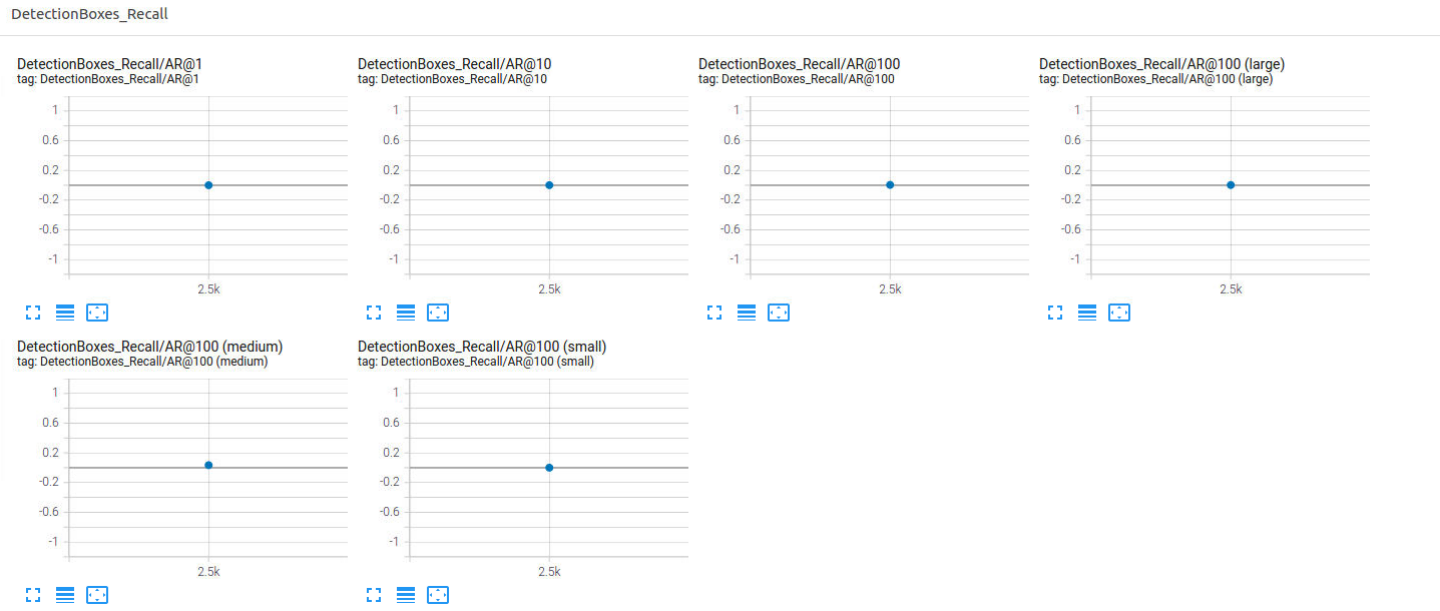
Training

Reference Experiment

First, config file was prepared for reference training. Pretrained model is downloaded and moved under pretrained_model folder. And then using edit_config.py file pipeline_new.config file was generated and moved under /home/workspace/experiments/reference/ directory.

For reference experiment the results from tensorboard visualization is given below:





For reference experiment, loss does not reach a plateau. End of the training total loss is too much to classify objects. And then evaluation process has been started, after evaluation mAP values are too low to identify classes. When I extract and generate gif with trained model I don't see any classification because of the too low average precision values.

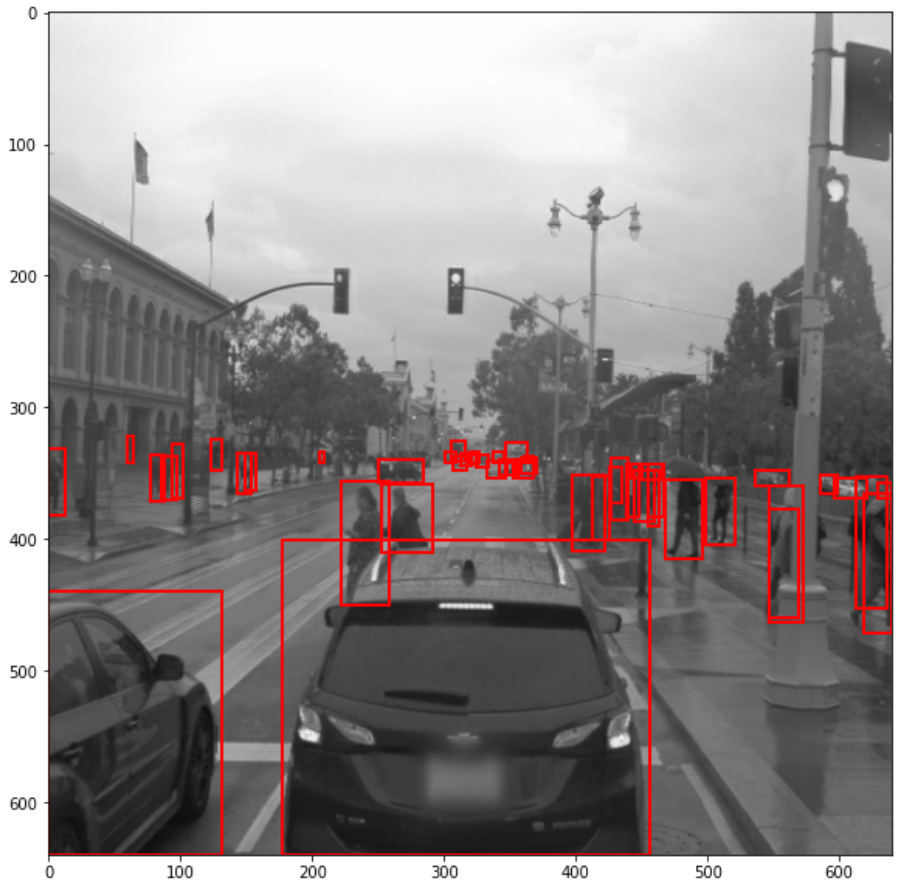
Improve on the reference

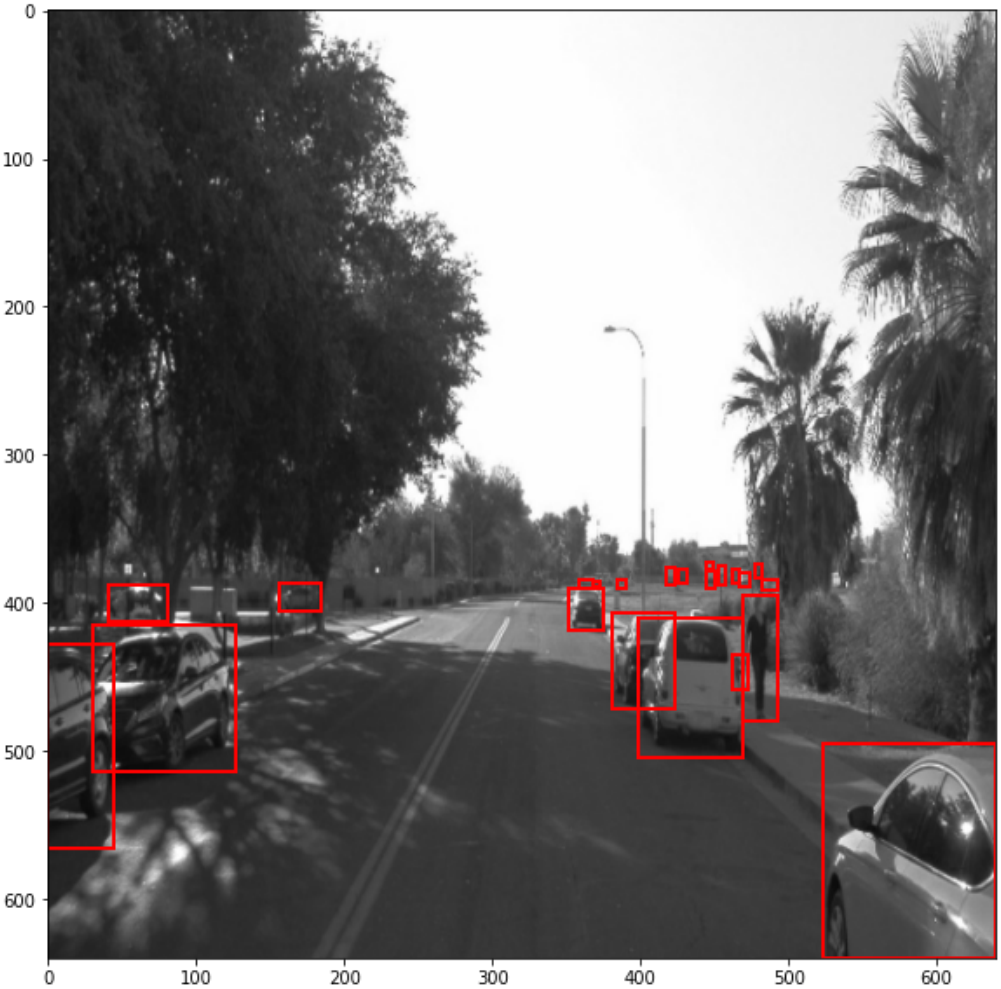
"**Data augmentation** is a way to increase the variability in the training dataset without having to capture more images. By applying pixel-level and geometric transformations during training, we can artificially simulate different environments. For example, a blur filter could be used to mimic motion blur. The augmentations should be carefully chosen."

Experiment1:
For the first experiment, I've tried some augmentations on pipeline given below:

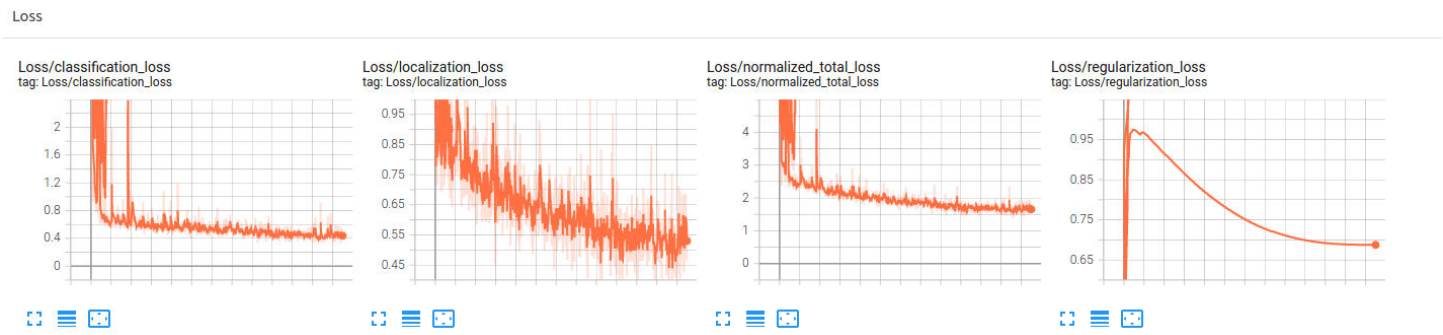
- Change batch size from 2 to 4
- Data augmentation random_rgb_to_gray with probability 0.6

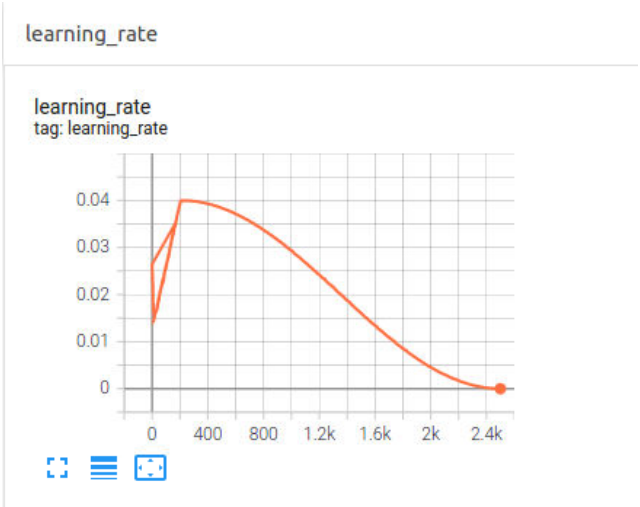
And I've checked augmentation on Explore_augmentations.ipynb. The followings are the augmented samples from dataset:


0
100
200
300
400
500
600
0
100
200
300
400
500
600


0
100
200
300
400
500
600
0
100
200
300
400
500
600

Then I started training for experiment1. The training results are as shown below:





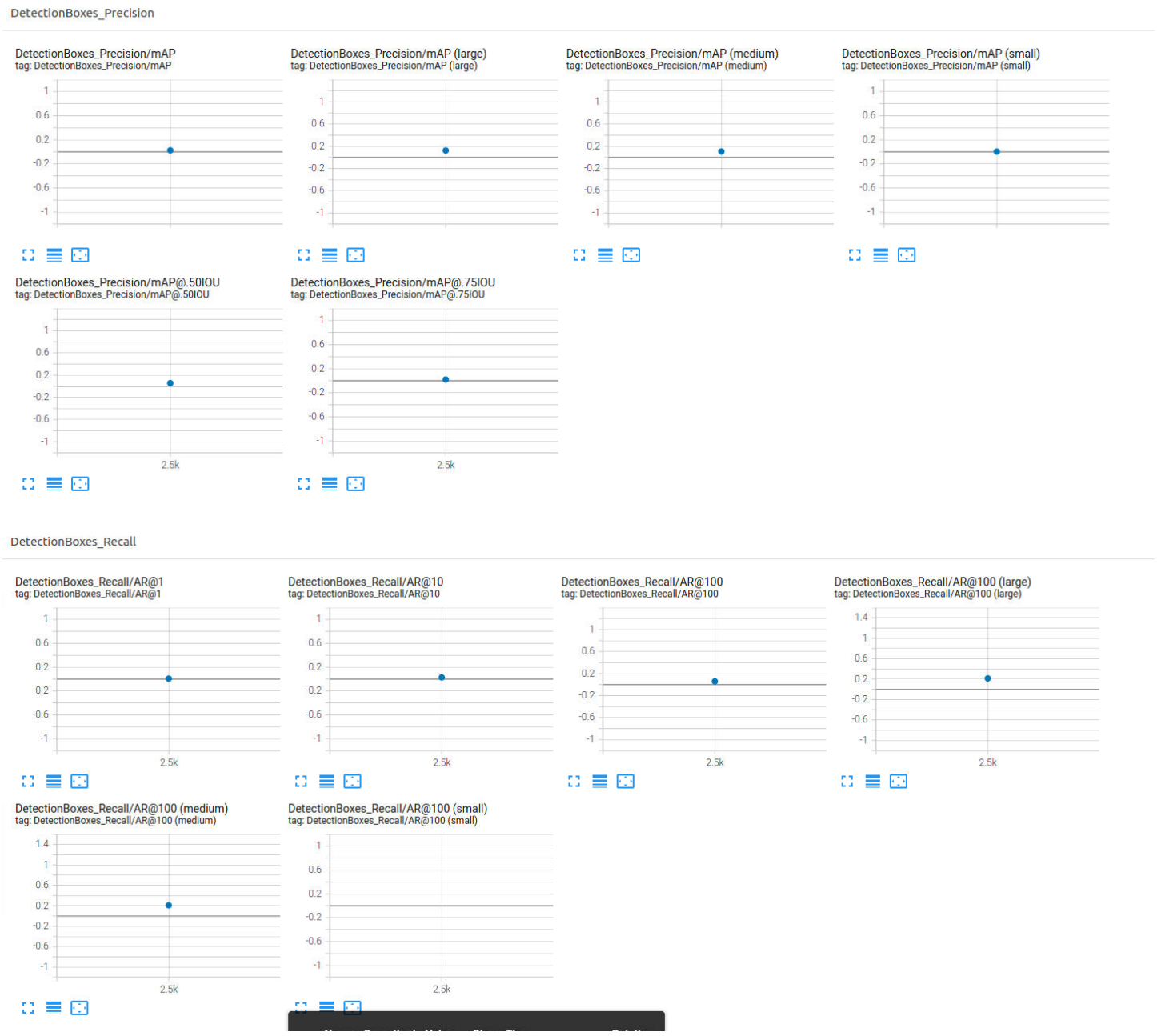
Still loss is too high to identify objects in frames. End of the training total loss is around 1.5 and classification loss is around 0.4.

And then I evaluated the model, after the evaluation process the average precision and average recall values and graphs are like below:

Evaluation of the experiment1:

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.024
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.056
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.019
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.004
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.106
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.124
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.008
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.028
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.059
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.025
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.209
```

```
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.214
```

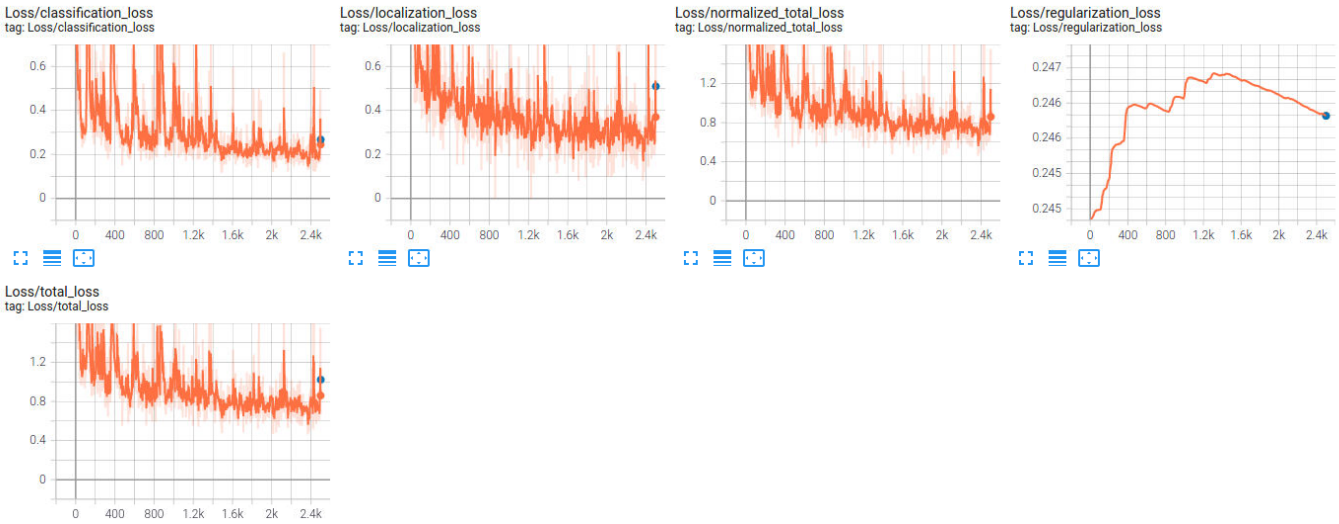


Mean average precision values are higher than reference experiment, but end of this experiment I can not able to see bounding boxes on gif also too. I also noticed that the precision values for small objects are lower than medium and large objects. And then I want to change some parameters and started to experiment2.

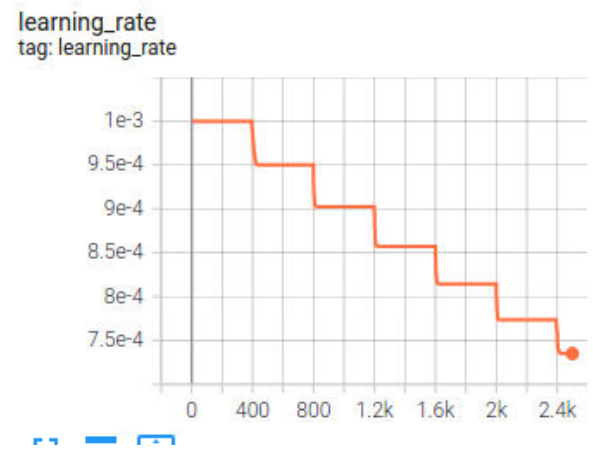
Experiment2:

For second experiment, I want to change optimizer from cosine decay to exponential decay and augment data with random rgb to gray with probability of 0.5.

And then I started training with this augmented data pipeline. The results for training are as shown below:

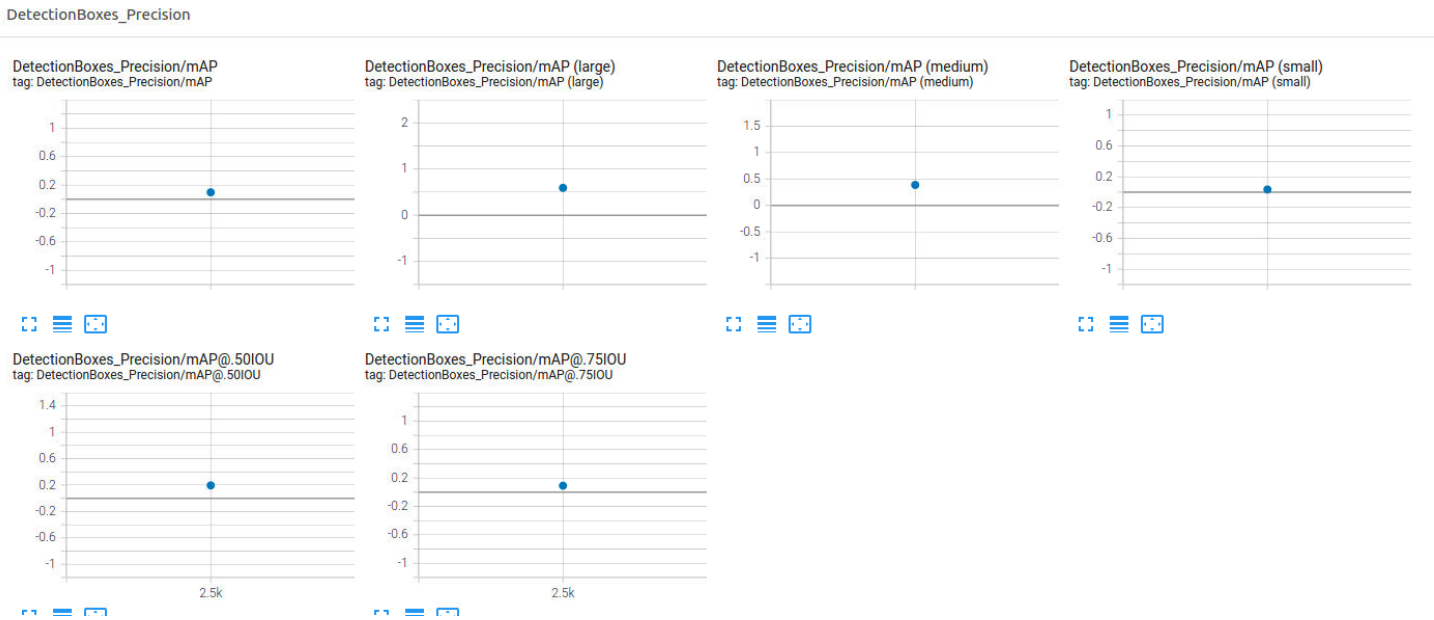


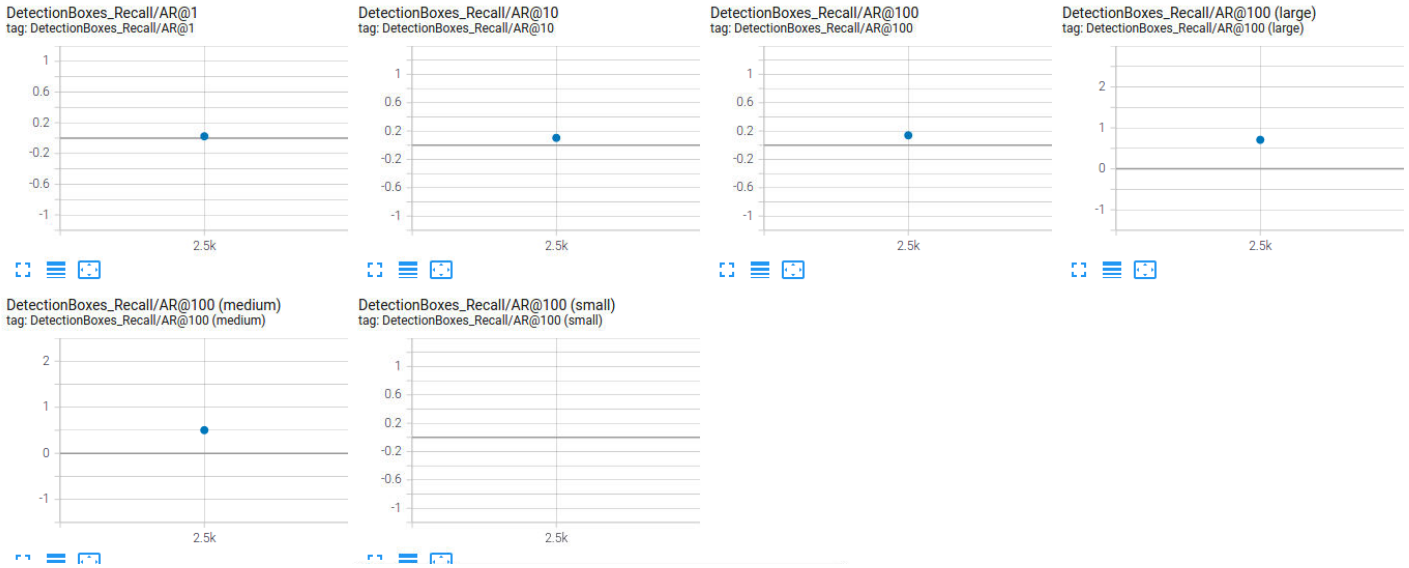
learning_rate



It can be seen on the graphs that the total loss is lower than 1. This is the best of the results in these experiments. End of the training also classification loss is around 0.2.

Next, I've evaluated this model and get the precision and recall values.





The graph shows that AP values are much more than previous experiments. And the detection of small objects still have low precision. After this process I exported the model and created a gif with this exported model. This time I am able to see the bounding boxes with precision values on gif.

One of the created animations with the last model can be seen under animations folder.