Object Detection in an Urban Environment

Project overview

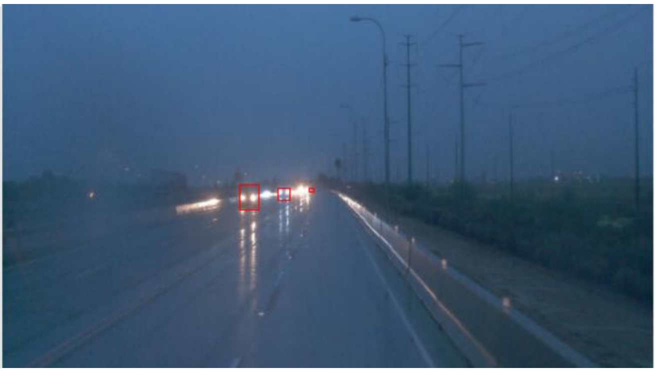
Object detection is a primary requirement for self-driving vehicles as it will make them sentient of the driving environment, the road conditions, and the obstacles they need to navigate. In this project we explore object detection in an urban environment as it is the most relevant scenario for use of self-driving vehicles.

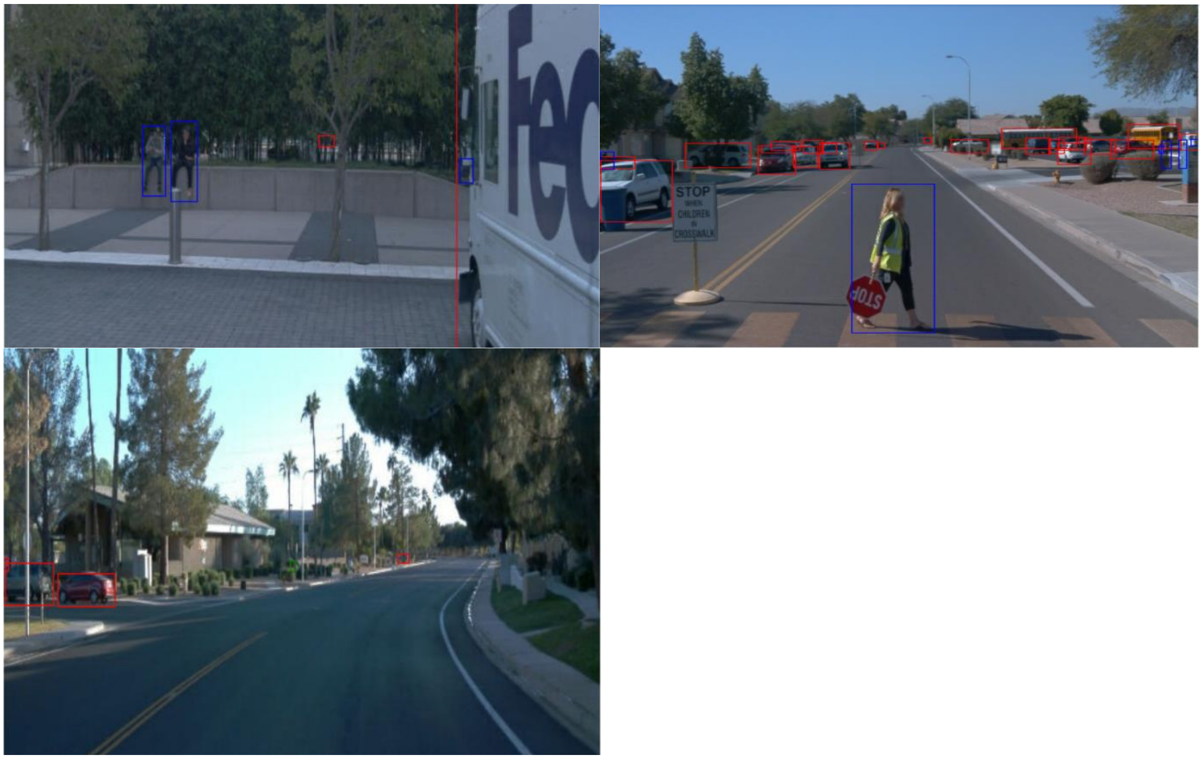
Dataset analysis

The exploratory data analysis notebook gives a good preview of the images in the dataset, here are some of my observations:

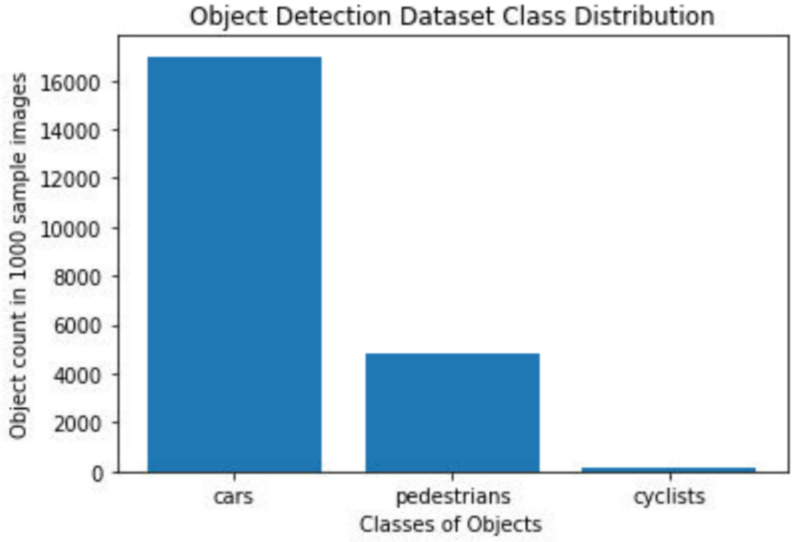
- It’s a city dataset
- The number plates are blurred
- There are night and daytime images
- Images of rainy days are included which slightly blurred
- Traffic lights are also in view
- There are several very small vehicles detected
- There are overlaps in bounding boxes and occlusions
- Some scenes have more number of objects than the others, for example busy intersections vs freeways
- Number of cyclists in the dataset is very small and hence, the dataset is skewed. The class distribution indicates very high number of vehicles, few pedestrians and very few cyclists.
- There are some hazy images due to camera flash
- The dataset includes parked vehicles as well as vehicles in motion

Here are some images from the dataset with incorrect detections:





Class distribution in 1000 sample images:



Cross Validation

In the latest code version, the splits have already been created with following number of images:

Training – 86

Validation – 10

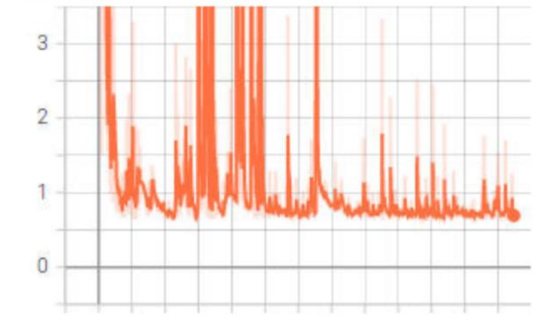
Testing – 3

Training

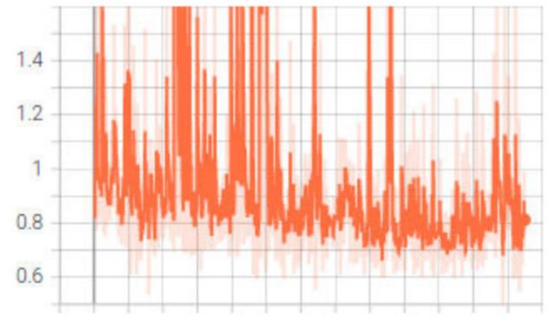
Reference experiment

For the reference training experiment, the loss was quite high with total loss = 5.646 with number of steps = 2500

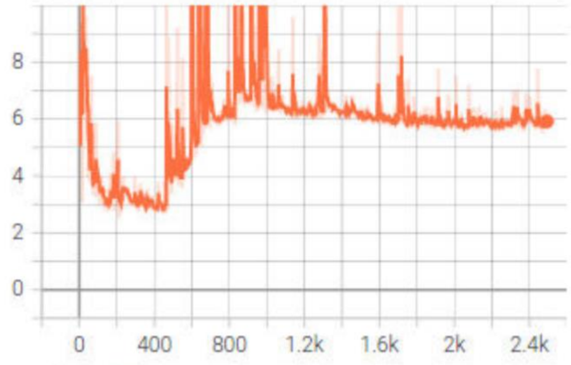
Loss/classification_loss
tag: Loss/classification_loss



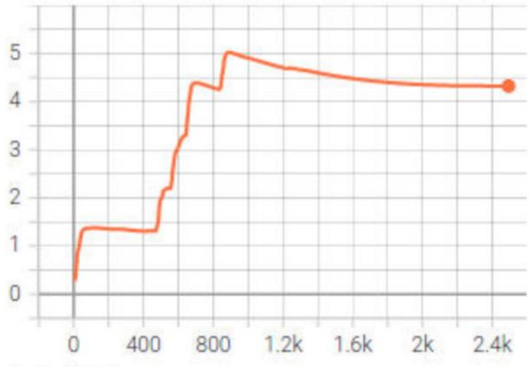
Loss/localization_loss
tag: Loss/localization_loss



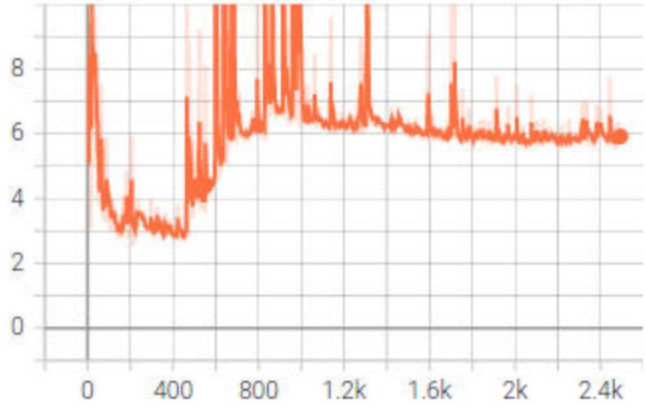
Loss/normalized_total_loss
tag: Loss/normalized_total_loss

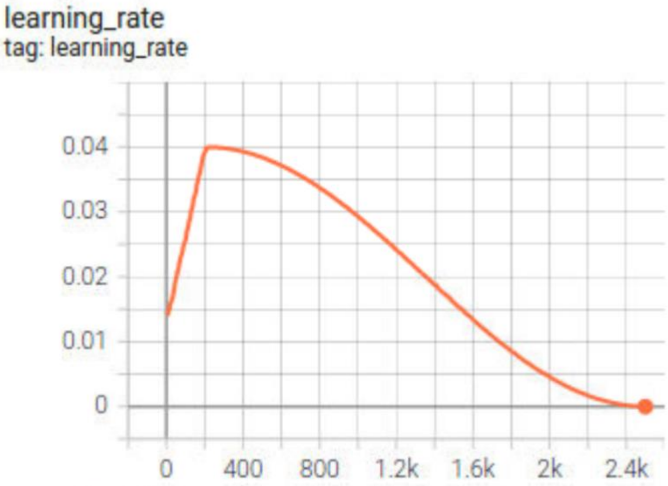


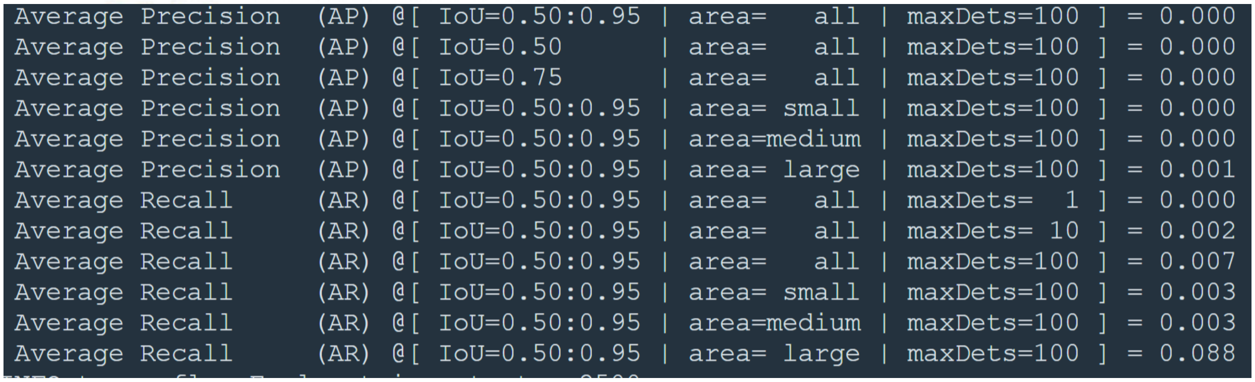
Loss/regularization_loss
tag: Loss/regularization_loss



Loss/total_loss
tag: Loss/total_loss





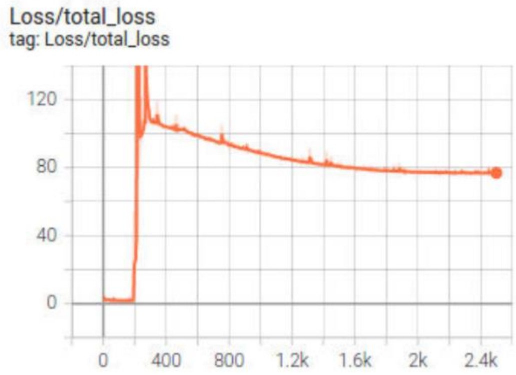


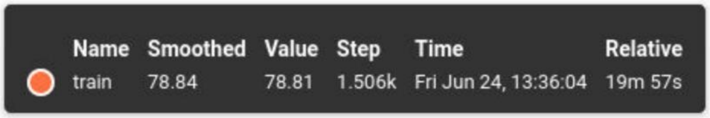
```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.001
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.002
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.007
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.003
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.003
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.088
```

Improve on the reference

1. Experiment1 – Adding augmentations to the reference image.
   Augmentations used : random_rgb_to_gray
   random_adjust_brightness
   random_adjust_contrast
   random_patch_gaussian

   The augmentations were selected to try and simulate night time and blurred images which are much lesser in the dataset.
   This led to an extremely high jump in the loss to 78

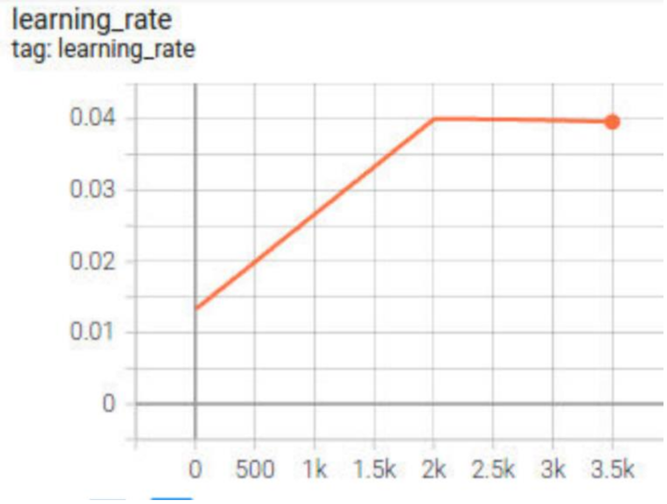


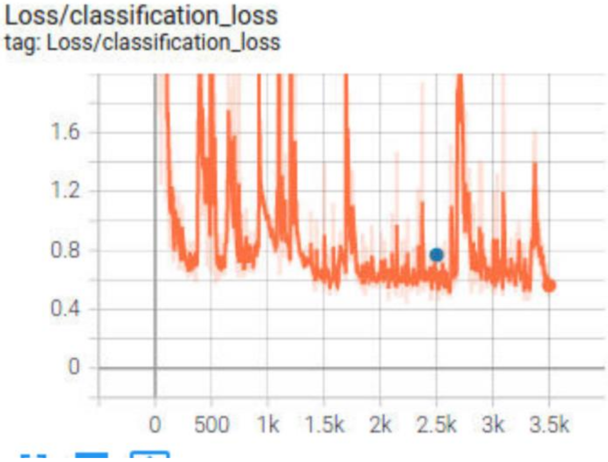


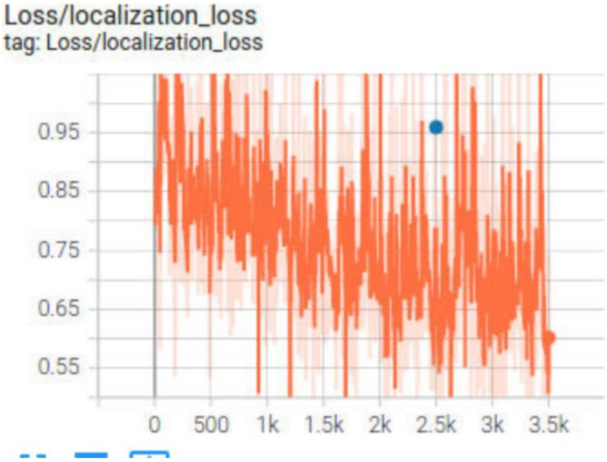
	Name	Smoothed	Value	Step	Time	Relative
●	train	78.84	78.81	1.506k	Fri Jun 24, 13:36:04	19m 57s

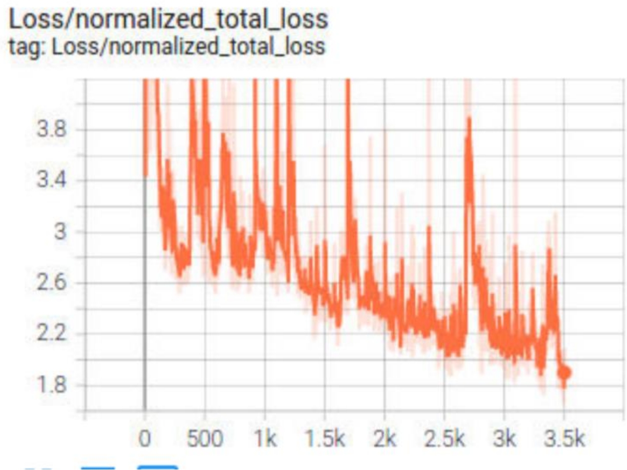
This loss was highly unexpected and after reading earlier posts on the Knowledge portal and comparing results, I found that maybe 2500 steps are very less for convergence of the loss. Also, after rerunning the experiment several times I concluded that the initial loss is the result of random selection of weights and bias and might be very high in some cases but should reduce as the training proceeds, further emphasizing that I needed to increase the number of steps.

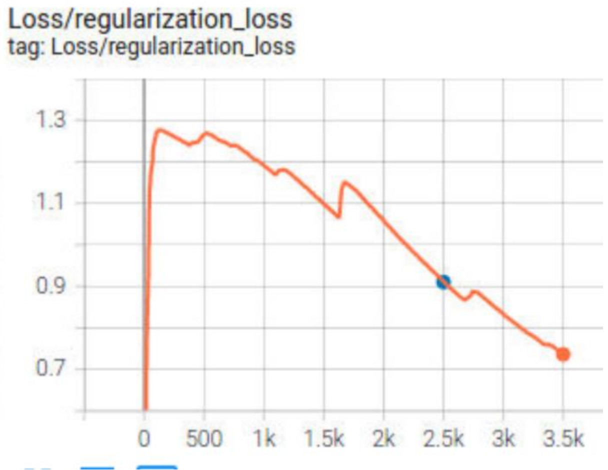
2. Experiment 2 – Increasing number of steps to 25000 in the optimizer and change of learning rate parameters to warm up for 2000 steps (no augmentation)

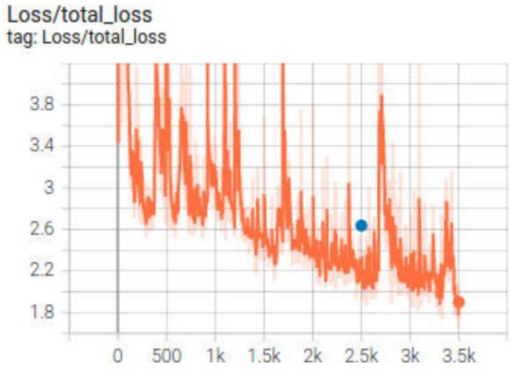

learning_rate
tag: learning_rate
0.04
0.03
0.02
0.01
0
0
500
1k
1.5k
2k
2.5k
3k
3.5k

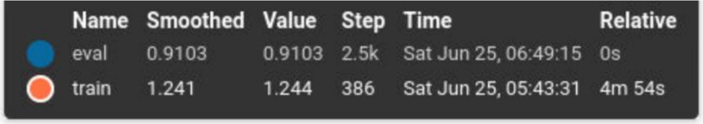

Loss/classification_loss
tag: Loss/classification_loss
1.6
1.2
0.8
0.4
0
0
500
1k
1.5k
2k
2.5k
3k
3.5k


Loss/localization_loss
tag: Loss/localization_loss
0.95
0.85
0.75
0.65
0.55
0
500
1k
1.5k
2k
2.5k
3k
3.5k





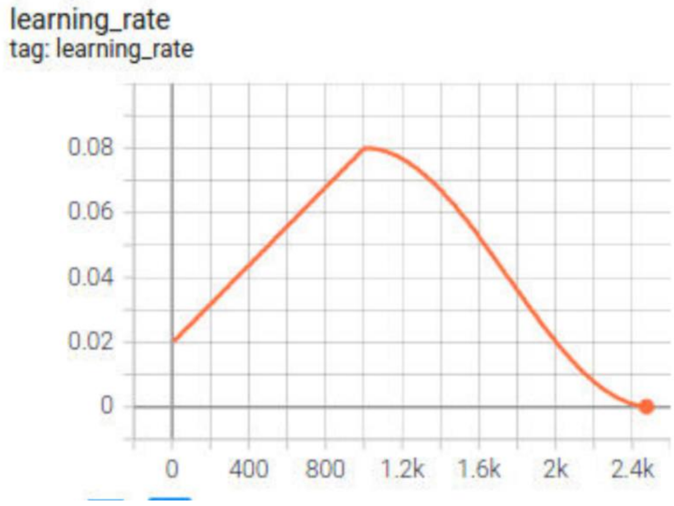


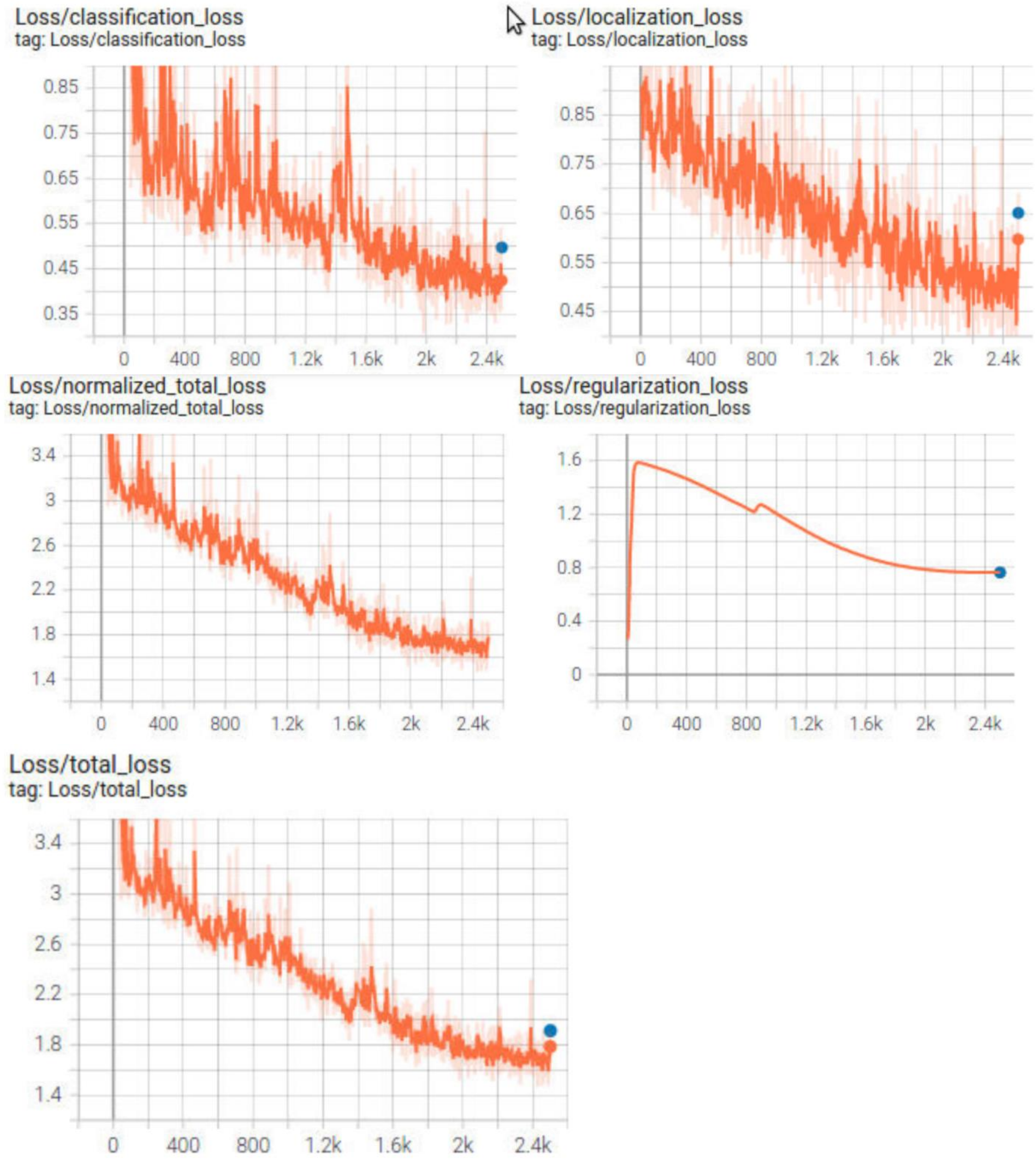


	Name	Smoothed	Value	Step	Time	Relative
●	eval	0.9103	0.9103	2.5k	Sat Jun 25, 06:49:15	0s
●	train	1.241	1.244	386	Sat Jun 25, 05:43:31	4m 54s

The results of this experiment showed a significant improvement, however, the experiment had to be interrupted because of lack of space in the Workspace which doesn't support more than 3500 steps at a time.

3. <u>Experiment 3</u> – In this experiment, I try to make the training converge faster in 2500 steps by using a higher learning rate with warm up in 1000 steps and batch size was increased to 4.





Loss/classification_loss
tag: Loss/classification_loss

Loss/localization_loss
tag: Loss/localization_loss

Loss/normalized_total_loss
tag: Loss/normalized_total_loss

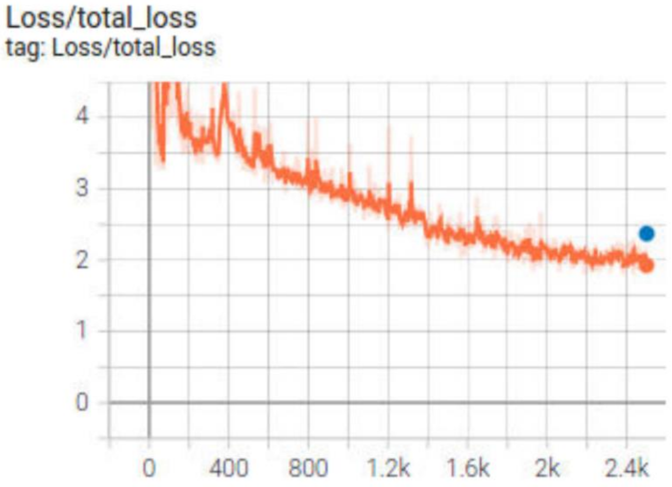
Loss/regularization_loss
tag: Loss/regularization_loss

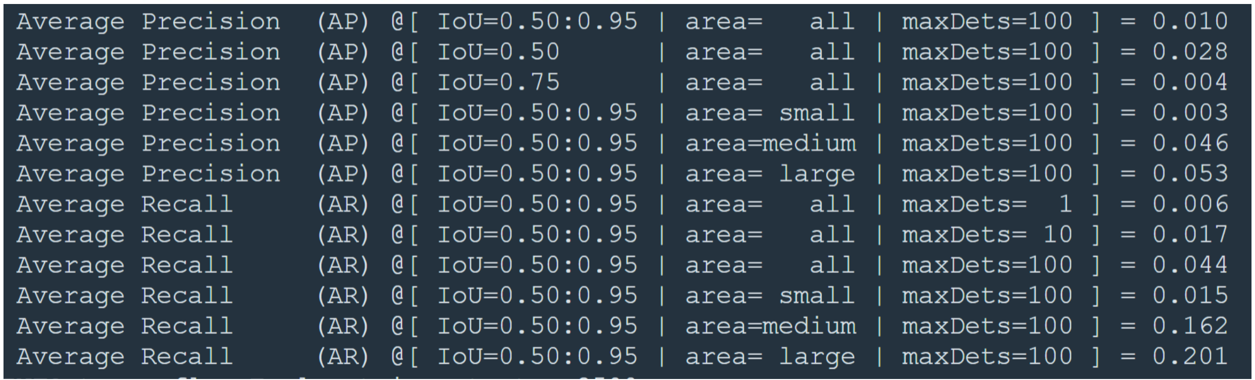
Loss/total_loss
tag: Loss/total_loss

The loss has decreased significantly to 1.8 with evaluation loss at 1.9

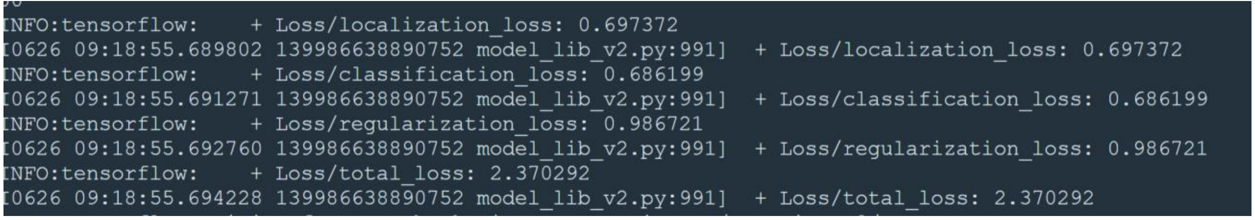
Several variations of the optimizer with different peak learning rates, warm up time and even type of learning rate decay (including exponential learning rate decay) were tried and this one was found to yield the best results consistently.

4. Experiment 4 - Added augmentations to this new optimizer:
    a. random_rgb_to_gray
    b. random_adjust_brightness





```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.010
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.028
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.004
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.003
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.046
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.053
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.006
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.017
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.044
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.015
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.162
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.201
```



```
INFO:tensorflow:    + Loss/localization_loss: 0.697372
I0626 09:18:55.689802 139986638890752 model_lib_v2.py:991]  + Loss/localization_loss: 0.697372
INFO:tensorflow:    + Loss/classification_loss: 0.686199
I0626 09:18:55.691271 139986638890752 model_lib_v2.py:991]  + Loss/classification_loss: 0.686199
INFO:tensorflow:    + Loss/regularization_loss: 0.986721
I0626 09:18:55.692760 139986638890752 model_lib_v2.py:991]  + Loss/regularization_loss: 0.986721
INFO:tensorflow:    + Loss/total_loss: 2.370292
I0626 09:18:55.694228 139986638890752 model_lib_v2.py:991]  + Loss/total_loss: 2.370292
```

We see that there is no significant improvement from augmentations, however, the metrics have improved significantly over the reference.