Project overview

One of the main perception tasks in autonomous driving is object detection, because it helps to figure out what is around the car, by identifying and differentiate other cars, motorcycle, cyclists, pedestrians and traffic controls from traffic light color to stop signs.

In this project, we are going to use an SSD Resnet 50 640x640 model to detect and classify cars, pedestrians and cyclists with training and validation data from Waymo Open dataset. We will walk through the complete pipeline, including Exploratory Data Analysis, training and validation split, model training and performance improvements.

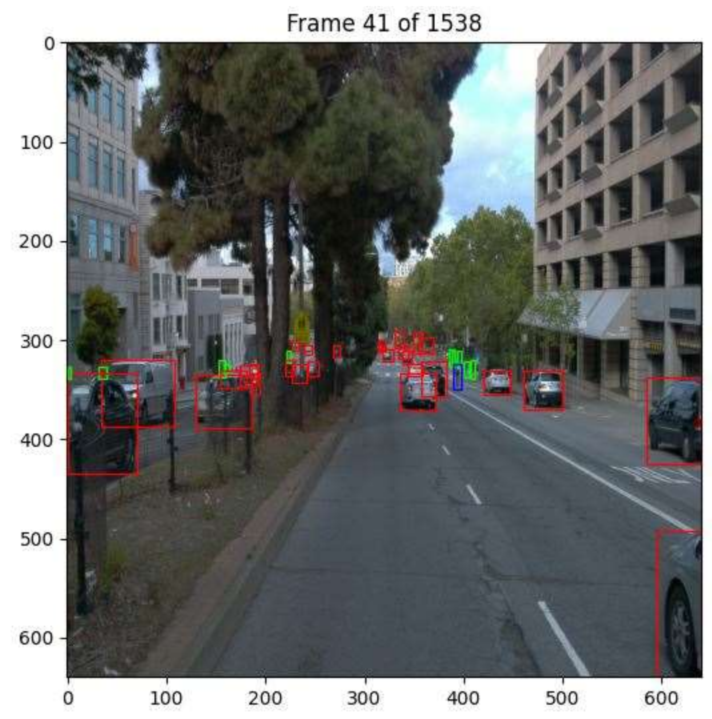
Set up

The code has been structure as requested. So it does not require any particular setup. I've just added the function "get_dataset2" in the "utils.py", which is slightly different respect to the original "get_dataset".
I will provide the modified "utils.py".

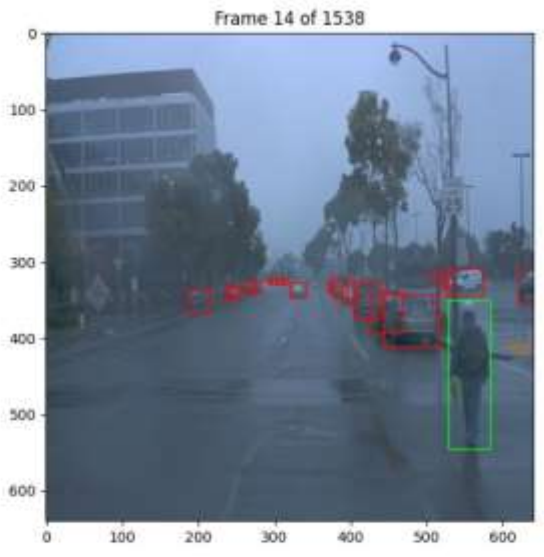
Dataset

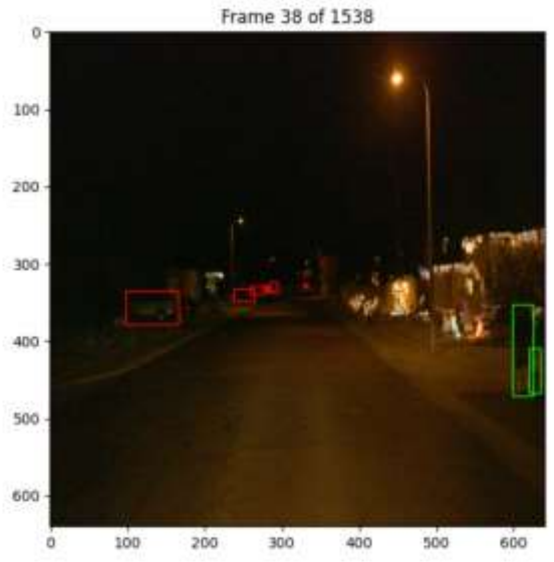
Dataset analysis

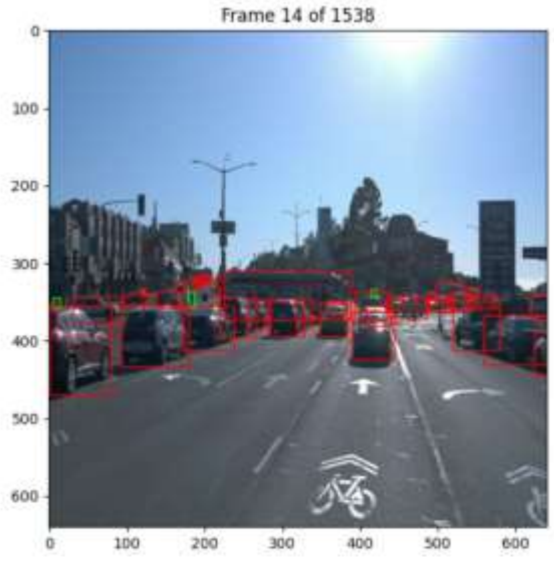
The dataset consists of 97 files, each of them containing 20 frames and labelled data with bounding boxes ground truth for 3 classes of objects: cars (red bounding box), pedestrians (green bounding box) and cyclists (blue bounding box).



All the images in the dataset seems to have the same resolution. However, they are quite unbalanced regarding the weather/light conditions; in fact, most of the data (around 75%) are sunny/good visibility images. The rest is rainy or/and nighttime data while very few images (less than 5%) exhibits a low-light environment.



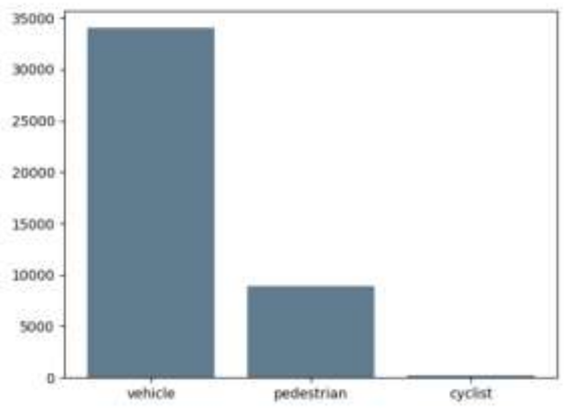




The distribution of the three classes of objects is unbalanced as well.

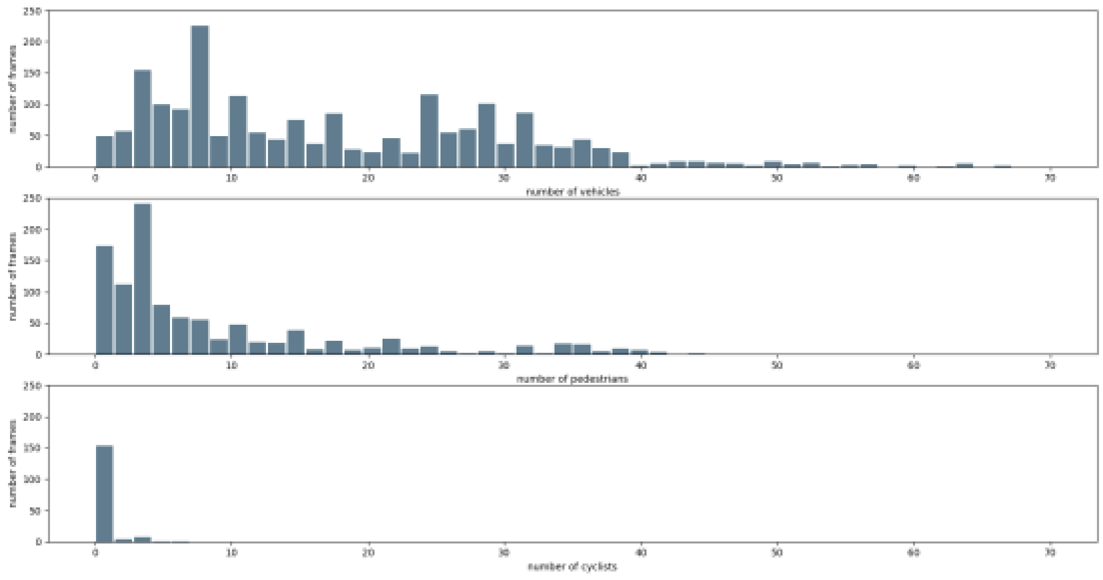
The number of objects in the dataset is as below:

- Vehicles - 34042
- Pedestrians - 8963
- Cyclists - 192



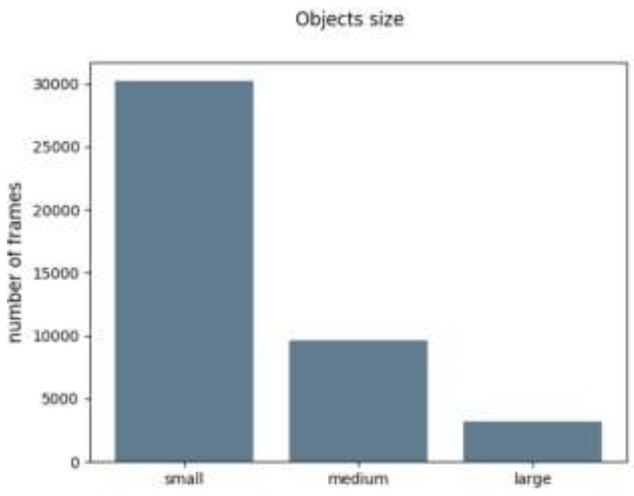
The number of vehicle objects in images are much larger than pedestrians and cyclists.

The class instances distribution in the dataset is as below:



Vehicle class has a very high incidence. A lot of frames contain vehicles objects (up to around 40 vehicles per frame). Therefore, the model should perform well with different intensity of traffic.

Regarding the object size, distribution in the dataset is as below:



Objects with an area with a number of pixel < 32^2 are considered small. Medium objects have 32^2 < area < 96^2, while large objects have area > 96^2 pixels

Cross validation

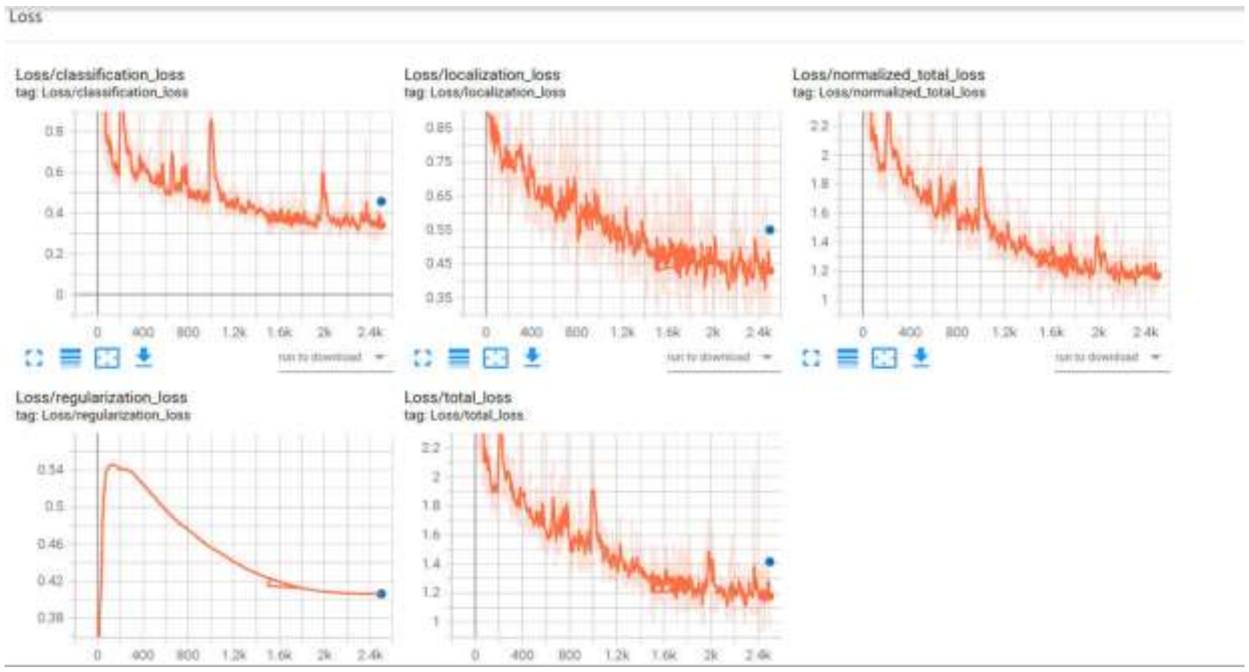
The dataset is split via "crate_splits.py" function into train and val sets – 80% and 20% respectively.

Training

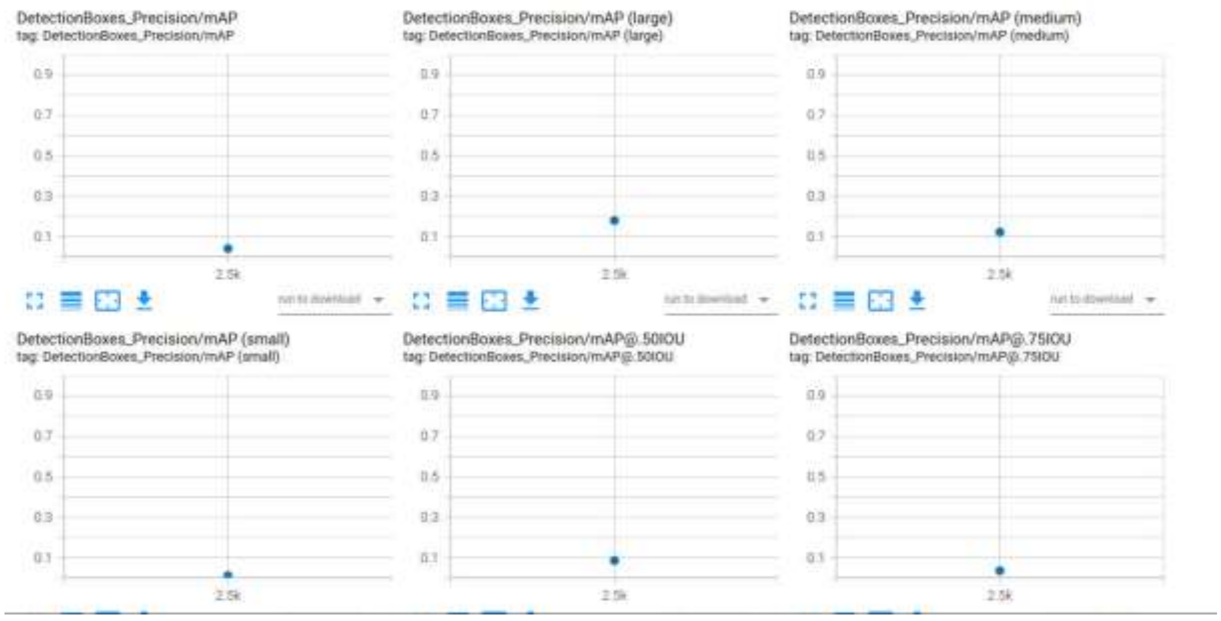
Reference experiment

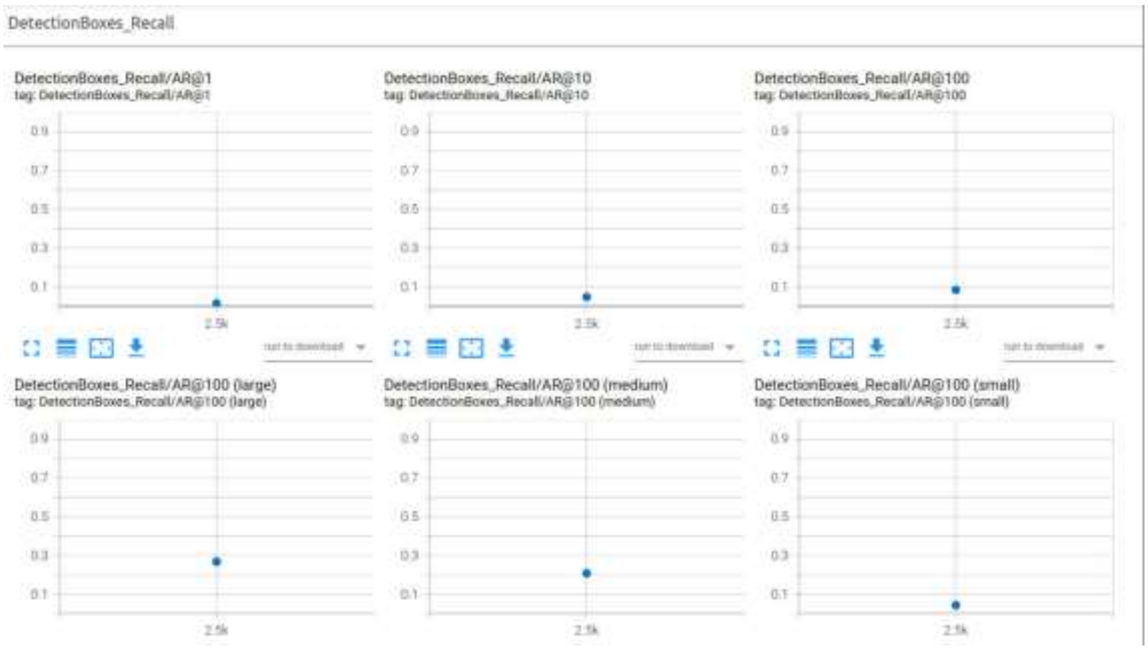
I started by training and evaluating the model with the reference configuration.

Loss – we can see that the training loss decrease while the training goes on and that the evaluation loss is higher than the training loss (at step 2500) as expected.



Precision and Recall - the values for precision and recall are very low. What we can observe is that regarding we obtained better performance on larger objects.



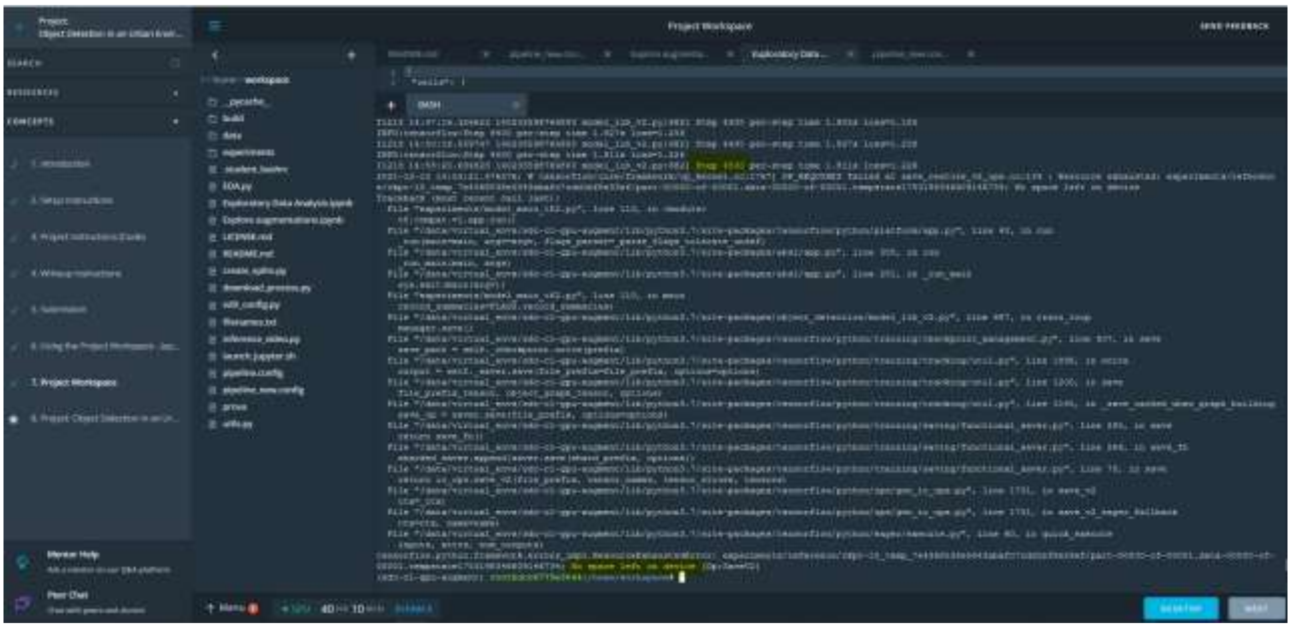


Improve on the reference

The reference experiment did not provide optimal results. So the first thing that I tried was to:

- **increase the number of training steps from 2.5k to 25k**

Unfortunately, I was not able to run the training since it stopped after 4.5k steps for memory constraints.



```
[illegible]
```

I've cleaned my workspace and try to run again the training, but I still had the same issue.

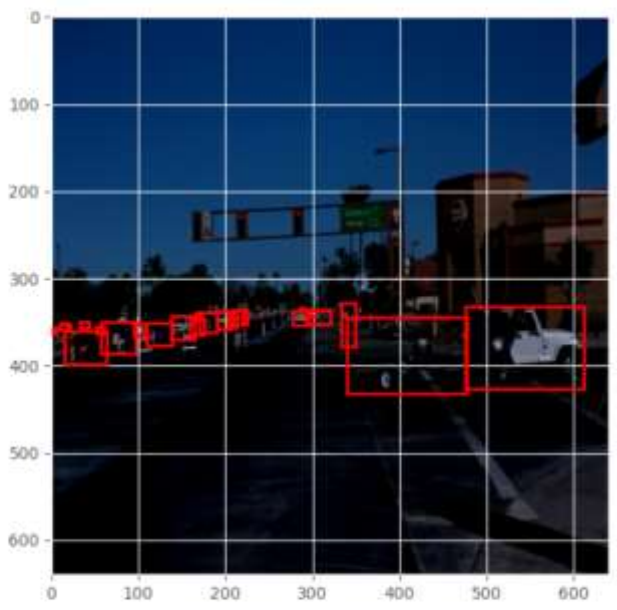
I considered here that the main goal of this project is to apply the competences acquired during the course, so I tried to go on by adopting the main strategies suggested to improve the model by keeping the number of steps at 2.5k.

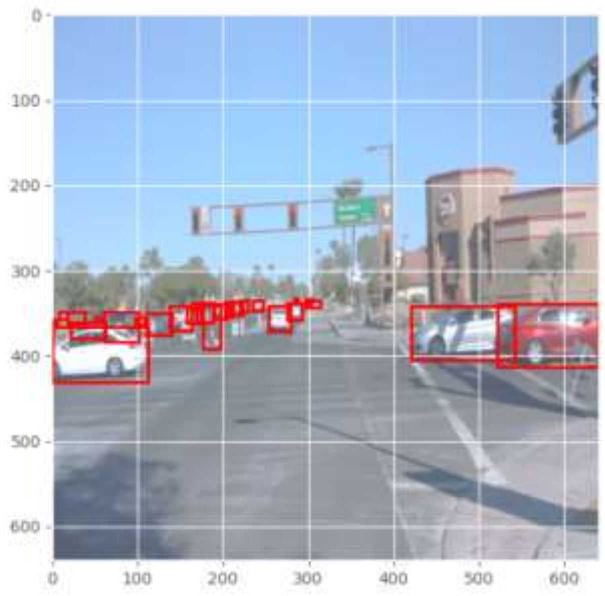
I tried to:

- **improve the data augmentation strategy (pipeline_new_v1.config)**

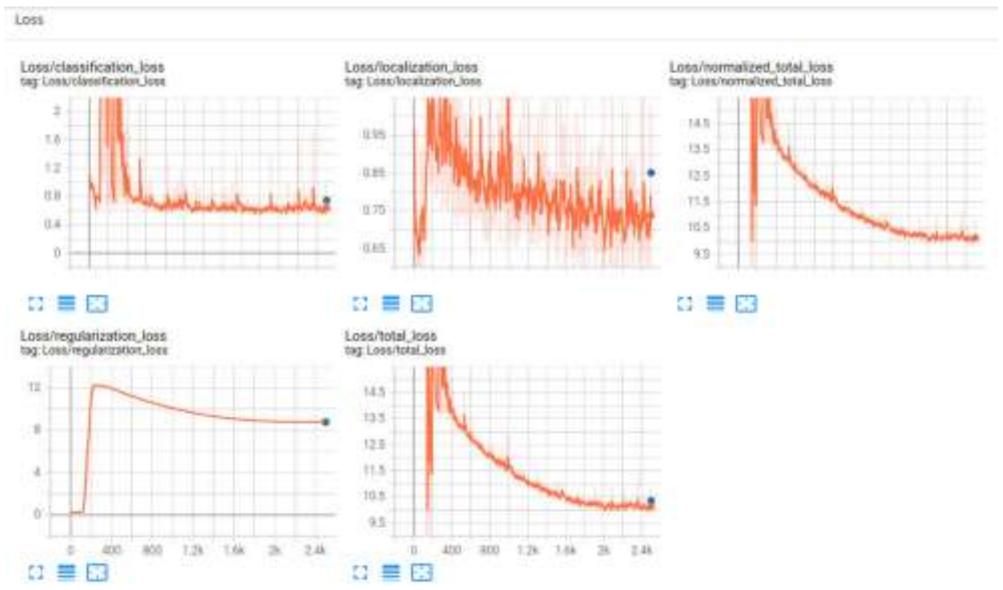
From the data analysis it turns out that the dataset is quite unbalanced regarding the weather/light conditions; in fact, most of the data (around 75%) contain sunny/good visibility images. The rest are rainy or/and nighttime images while very few images (less than 5%) exhibits a low-light environment.

In order to allow the model to generalize across images trained on different lighting levels, I applied the random brightness augmentation (with max_delta parameter set to 0.4) to expand the size of the training set by randomly darkening or brightening images.





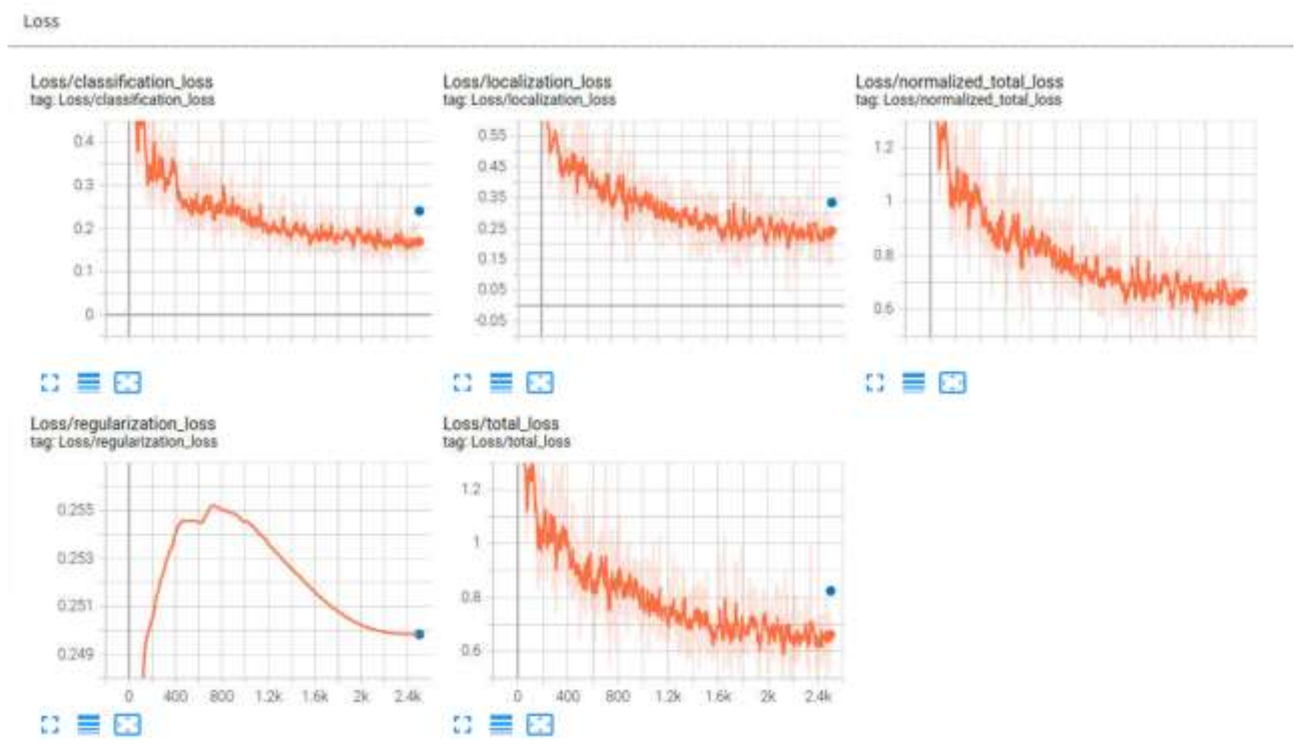
In the new loss obtained, we can see very “unstable” behavior with some overshoots.



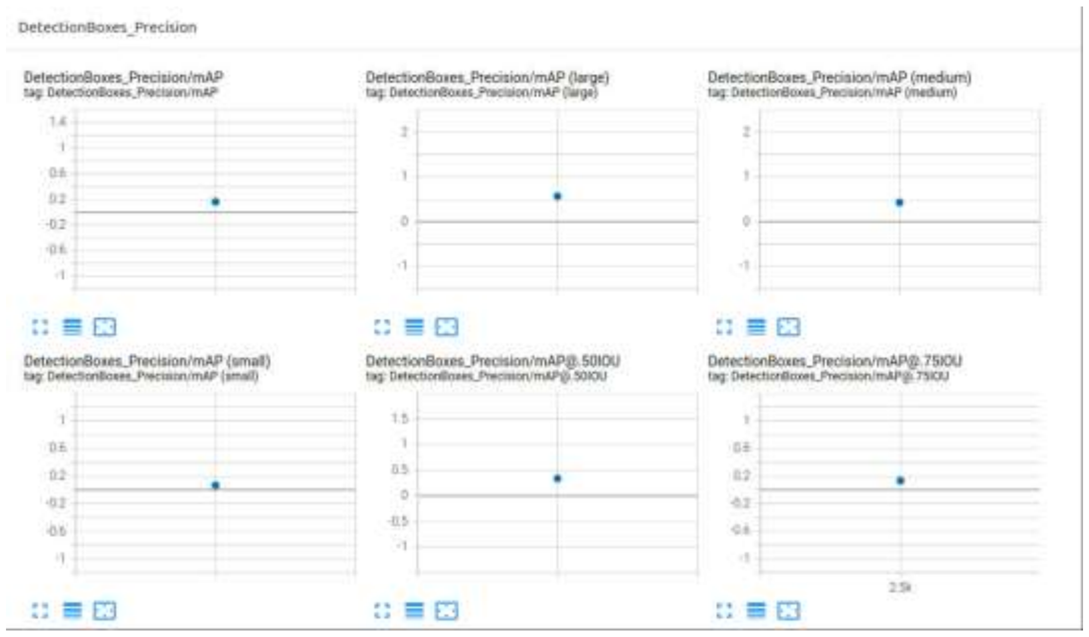
Finally, I’ve tried to:

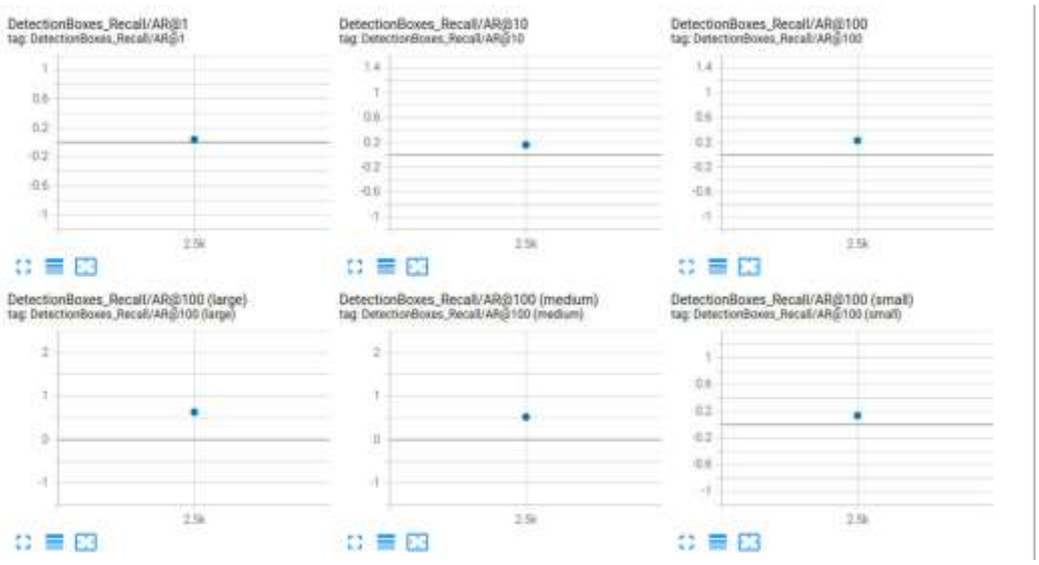
- **lower the learning rate from 0.04 to 0.1 (pipeline_new_v2.config)**

Loss – with a smaller learning rate it seems that the losses improve. The values after 2.5k are reduced by half respect to values obtained with the first reference experiment



Precision and Recall – they also show some improvements respect to the reference values, mainly for recall AR@100 for large and medium objects.




DetectionBoxes_Recall/AR@1
tag: DetectionBoxes_Recall/AR@1
1
0.6
0.2
-0.2
-0.6
-1
2.5k
DetectionBoxes_Recall/AR@10
tag: DetectionBoxes_Recall/AR@10
1.4
1
0.6
0.2
-0.2
-0.6
-1
2.5k
DetectionBoxes_Recall/AR@100
tag: DetectionBoxes_Recall/AR@100
1.4
1
0.6
0.2
-0.2
-0.6
-1
2.5k
DetectionBoxes_Recall/AR@100 (large)
tag: DetectionBoxes_Recall/AR@100 (large)
2
1
0
-1
2.5k
DetectionBoxes_Recall/AR@100 (medium)
tag: DetectionBoxes_Recall/AR@100 (medium)
2
1
0
-1
2.5k
DetectionBoxes_Recall/AR@100 (small)
tag: DetectionBoxes_Recall/AR@100 (small)
1
0.6
0.2
-0.2
-0.6
-1
2.5k