Object Detection in an Urban Environment

1 Project Overview

Understanding the context of the environment that the Self-Driving Car is operating in is critical to its function. There are sensors that can help detect presence of objects in the environment but the primary job of classification of the objects rests with the cameras enable by Computer Vision or Convoluted Neural Networks. Self-Driving Cars need object detection to determine the location of traffic signs, pedestrians, and vehicles arounds it. Without knowing this information, the self-driving vehicle cannot make any critical driving decisions.

In this project, the aim is to study the data available the Waymo Open Dataset and create a convoluted neural network to detect and classify objects. A pre-trained model is going to be used for detection. Object detection is done using TensorFlow Object Detection API.

2 Set up

For completing this project, I have used the Udacity classroom workspace. So, the necessary data and libraries were already available. This project can also be completed on local machine.

2.1 Local Setup

2.1.1 Docker Setup

For local setup if you have your own Nvidia GPU, you can use the provided Dockerfile and requirements in the `build` directory of the starter code.

Build the image with:

```
docker build -t project-dev -f Dockerfile .
```

Create a container with:

```
docker run --gpus all -v <PATH TO LOCAL PROJECT FOLDER>:/app/project/ --network=host -ti project-dev bash
```

and any other flag you find useful to your system. eg, `--shm-size`

Once in container, you will need to install gsutil, which you can easily do by running:

```
curl https://sdk.cloud.google.com | bash
```

Once gsutil is installed and added to your path, you can auth using:

```
gcloud auth login
```

2.2 Execution Steps

2.2.1 Launch and run Jupyter notebook

Jupyter Notebook is used for data analysis. Use

`jupyter notebook Exploratory Data Analysis.ipynb` to start the notebook.

2.2.2 Model Training and Evaluation

Training process:

```
python experiments/model_main_tf2.py --model_dir=experiments/reference/ --pipeline_co
nfig_path=experiments/reference/pipeline_new.config
```

To monitor the training, you can launch a tensorboard instance by running `python -m tensorboard.main --logdir experiments/reference/`.

Evaluation Process:

```
python experiments/model_main_tf2.py --model_dir=experiments/reference/ --pipeline_co
nfig_path=experiments/reference/pipeline_new.config --checkpoint_dir=experiments/refe
rence/
```

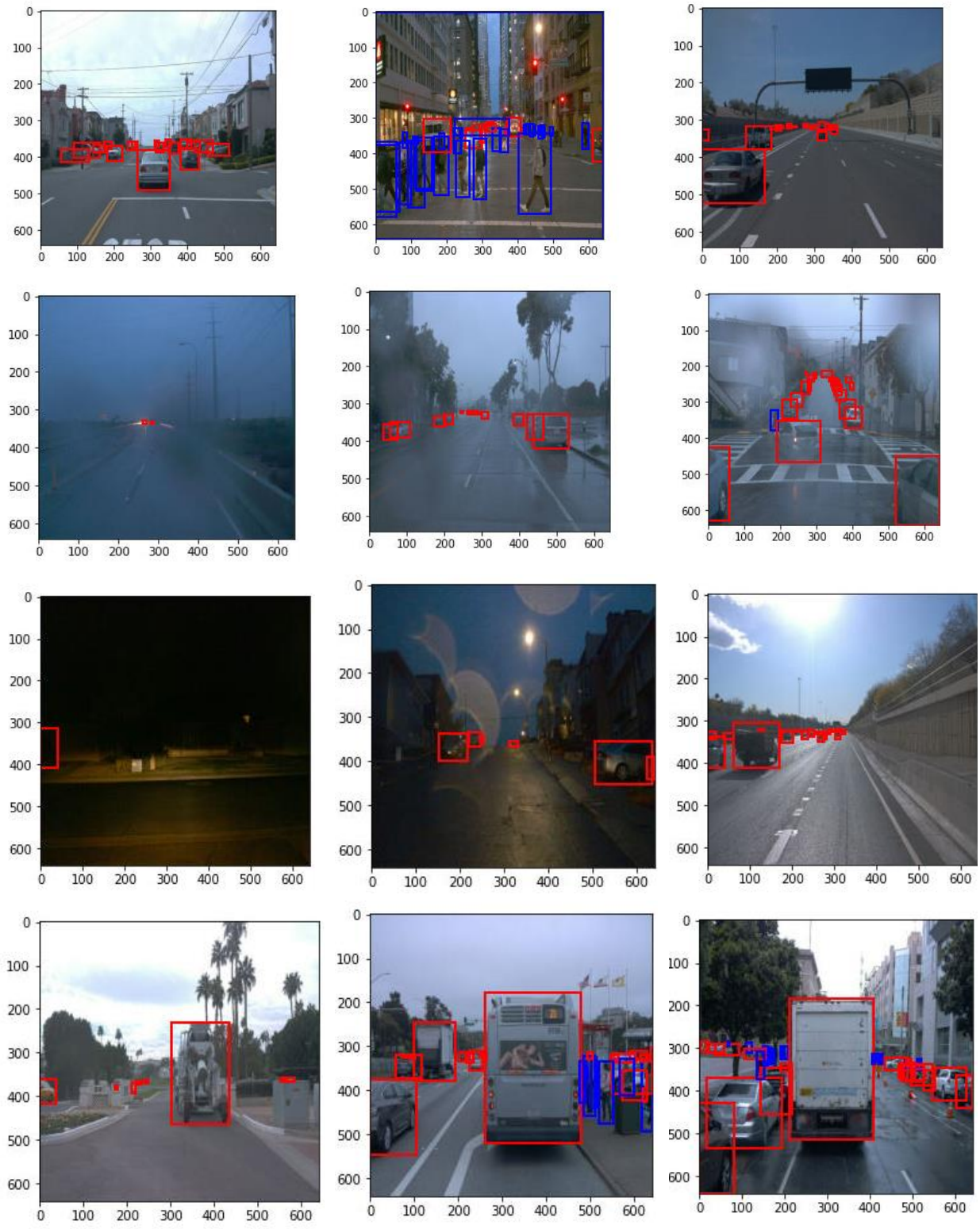
3 Dataset Analysis

Dataset includes variety of examples road types. There are some city centers and downtowns with high density of pedestrians. There are some cases of highway driving and then there are suburban cases as well where we could see a mix of both cars, pedestrians and cyclists.

The dataset also contains different levels of weather conditions. There cases of rain and fog observed in the dataset. Though it is hard to quantify what percentage of the dataset contains these cases.

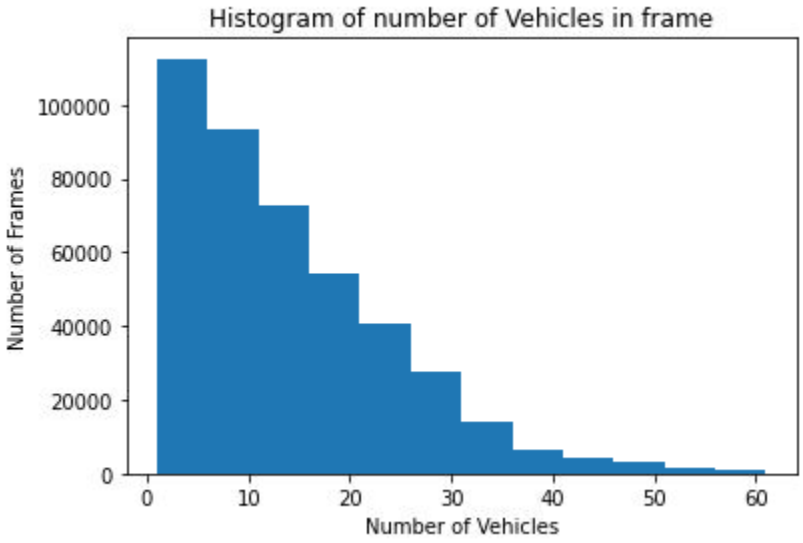
Data also contains examples from different lighting conditions. We see some nighttime data with the headlight being the only light source. There are nighttime example with street lights and there are cases with daytime driving with bright sunny condition. Again, it is difficult to estimate the percentage of data that contains these cases.

Finally, the dataset contains example of vehicles other than cars. We see an example of cement mixer. There are cases with trucks and then there are example of public transit buses. The challenge here is we don’t know the mix between cars and other motor vehicles.

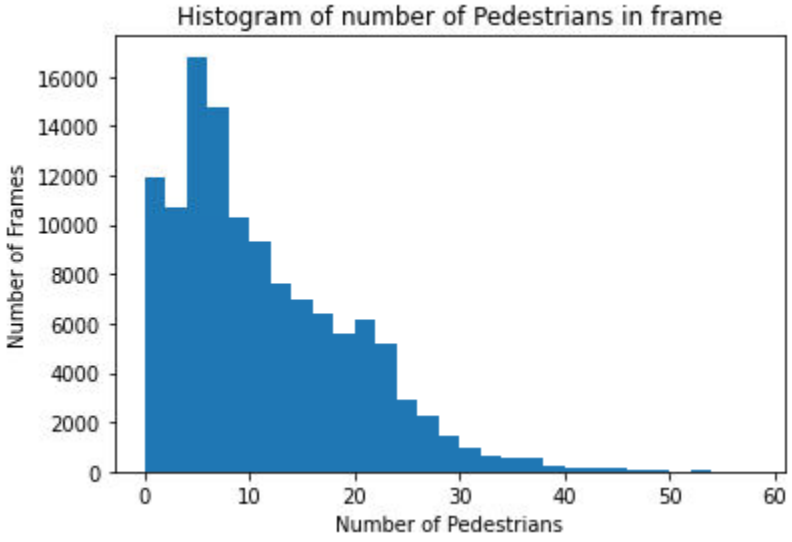


Additional data analysis was done on the dataset to analyze the distribution of labels within the dataset. In 25000 frames there are 431586 vehicles, 121718 pedestrians and 3119 cyclists

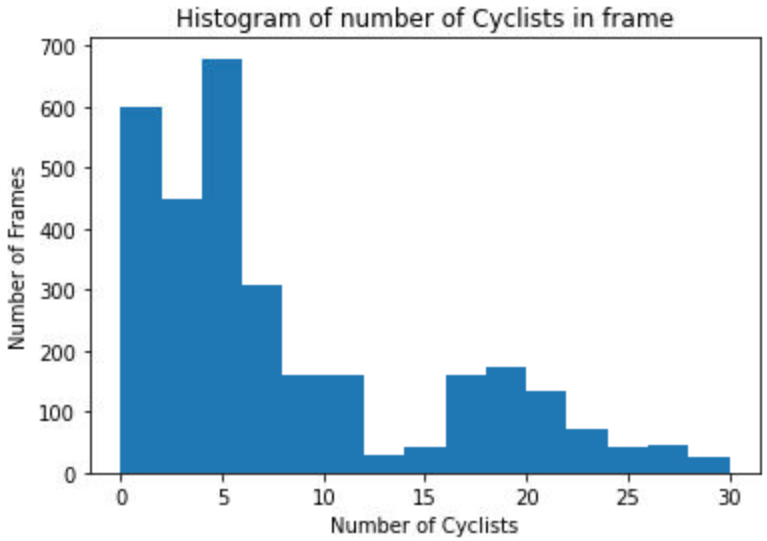
The histogram for number of vehicles in frames is below. It exhibits a smooth reduction in number of frames as the number of vehicles increases. This tells us that data won't show many scenarios with more 35-40 vehicles in a frame.



The histogram for number of pedestrians in frames is below. It has peak value around 6-7 pedestrians. This shows that significant number of frames usually have pedestrians in it and that the number of pedestrians per frame is not expected to be large.



The histogram for number of cyclists in frames is below. Its peak value and number of frames are too low. This indicates that the there are very few cyclists in the dataset. So, we have to be careful that the detection doesn't bias away from them as identifying cyclists is an important part of autonomous vehicles function.

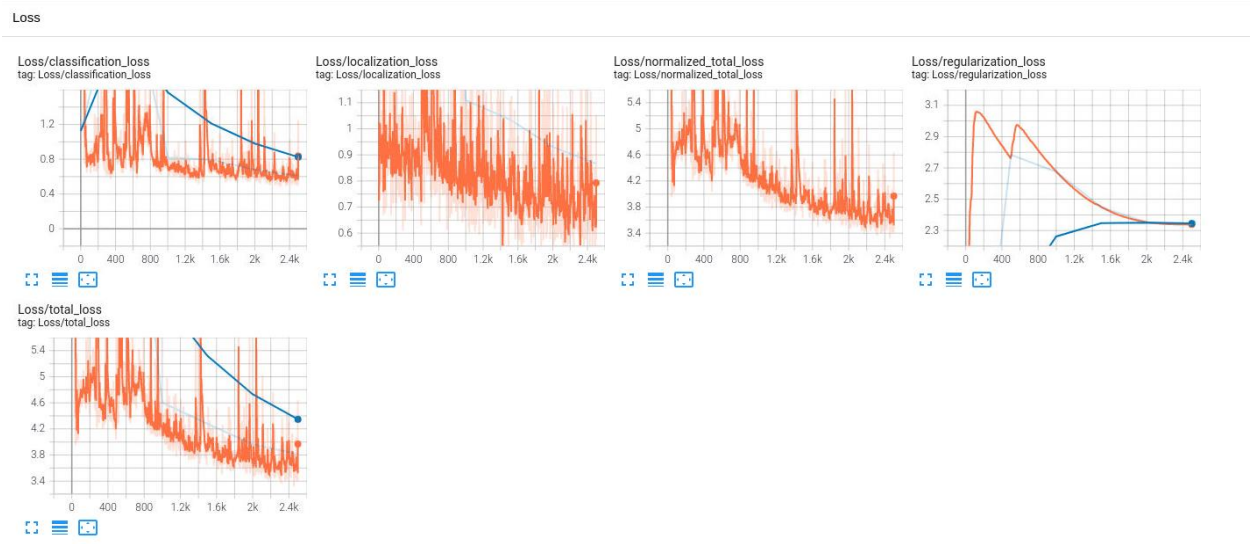


Cross validation is not needed with this project as the training and validation dataset have been split by Udacity.

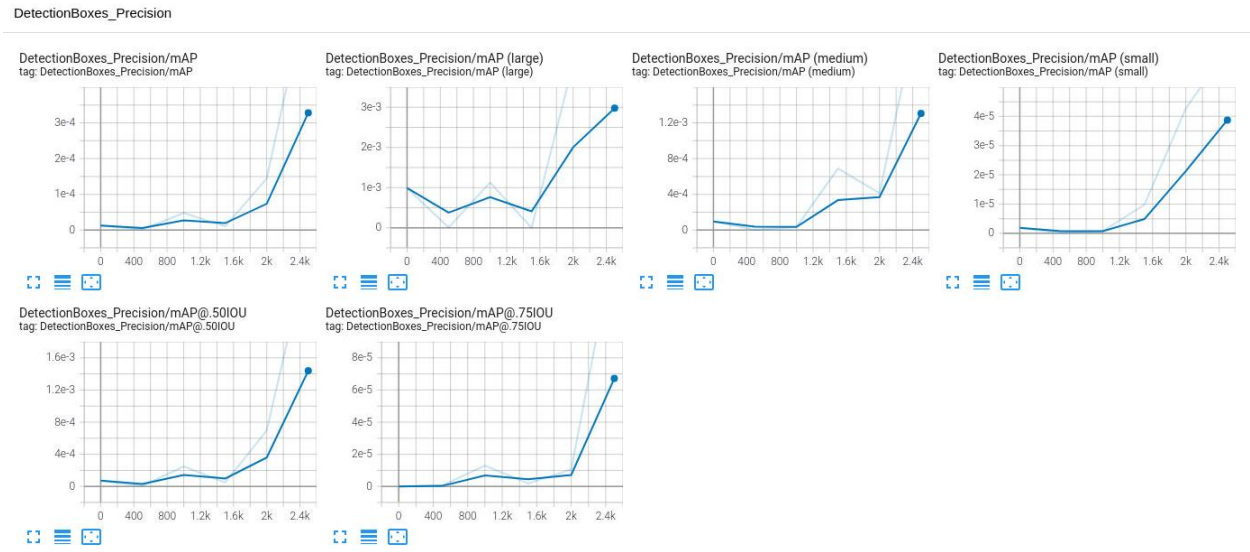
4 Training with Reference Experiment

Following are the results from the reference experiment run after config file was downloaded and moved to the reference folder. Nothing other than batch size was changed in the config file and a reference training was run.

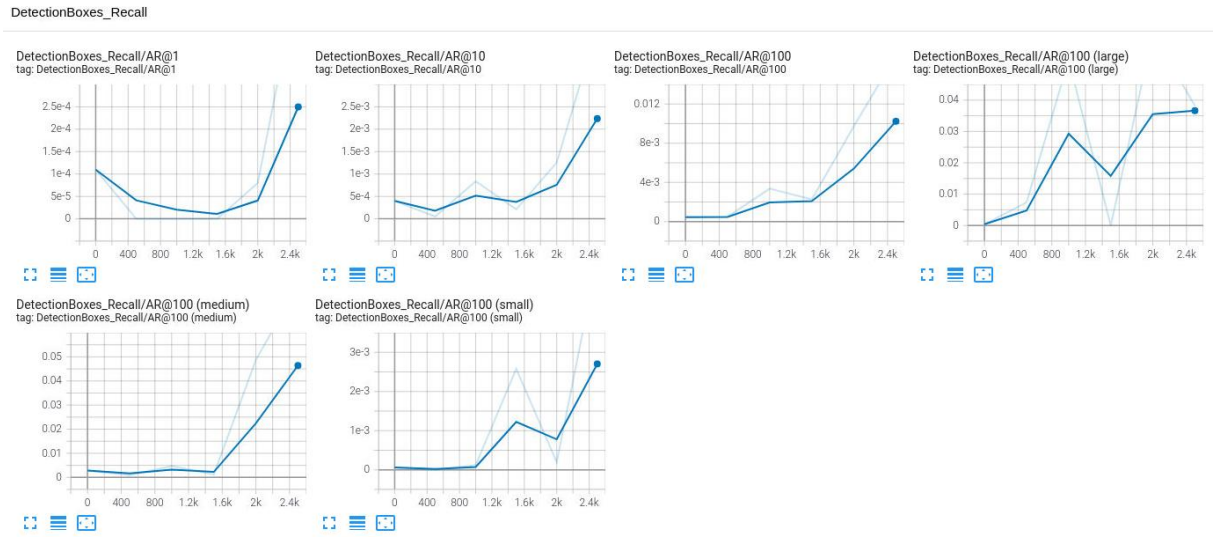
Loss:



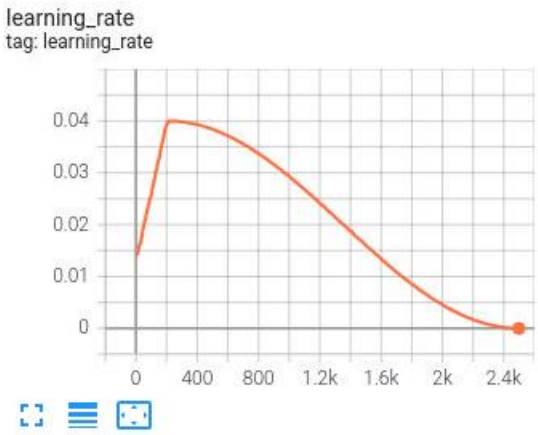
Precision:



Recall:



Learning Rate:



Looking at precision and recall charts it looks like the training could have used some more steps. The reference config file was run for 2500 steps. Next training should be run with more steps or lower learning rate so the precision, recall, loss and learning rate can be flatter before we stop training rather than showing signs of further potential movement in future steps.

Loss graph shows some signs of overfitting as there is divergence and gap between the training (orange) and validation (blue) data.

Final point to note is that precision for the training for reference was low. So, some augmentation should be attempted to improve that.

5 Improvement Strategy

In order to use the full potential of the training we will try different augmentations along with improvement to cosine decay learning rate and batch size.

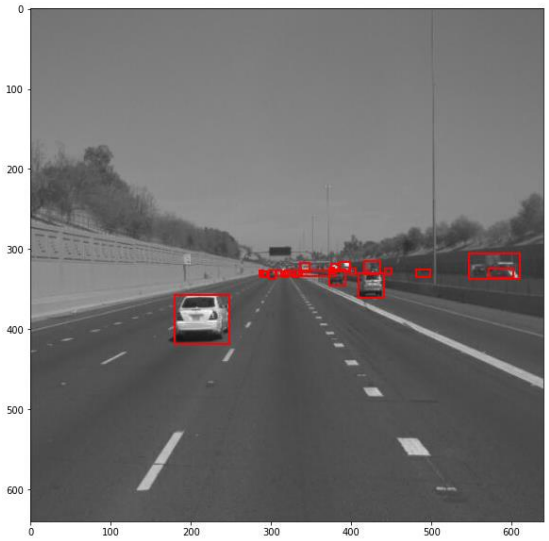
Augmentations are also applied to help the optimizer with different scenario which should improve the final detection model. Below is the list of augmentation applied,

- Random RGB to Gray
- Random Horizontal Flip
- Random Adjust Brightness
- Random Adjust Contrast

The learning rate was modified to 0.0004 and batch size to 4. Both these changes done to improve the optimizer under each step rather than it working over multiple steps to land to a solution that could have been achieved sooner.

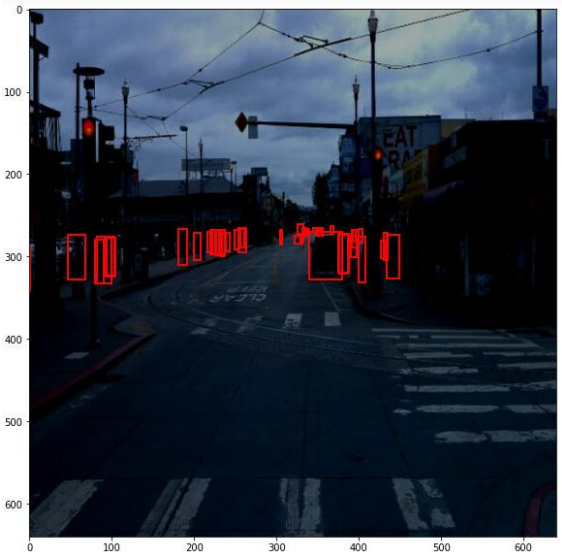
Samples of Augmentation:

Random RGB to Gray



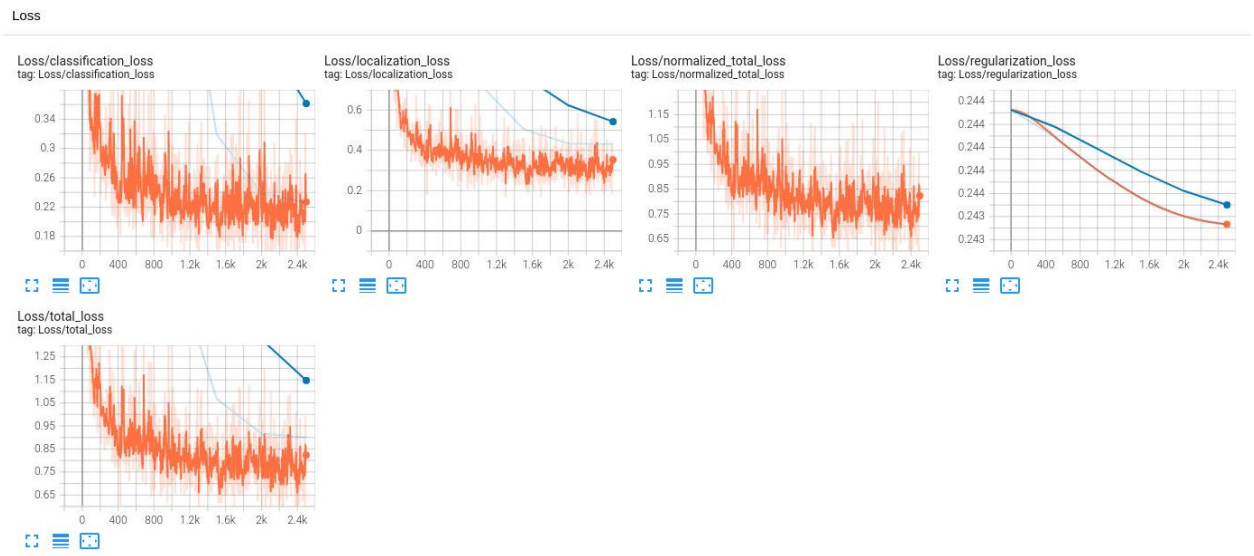
Random Horizontal Flip

Random Adjust Brightness

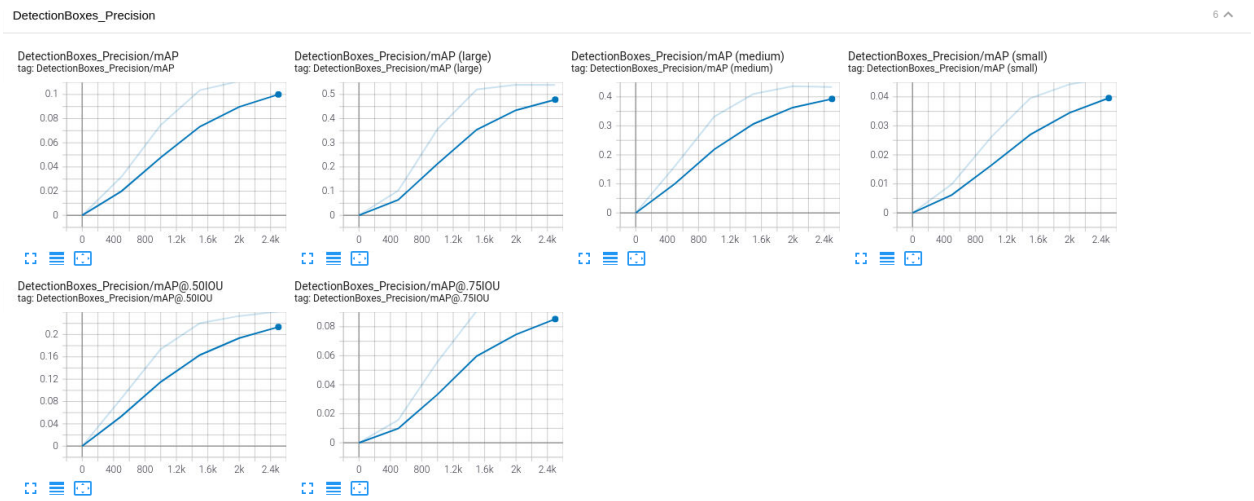


Random Adjust Contrast

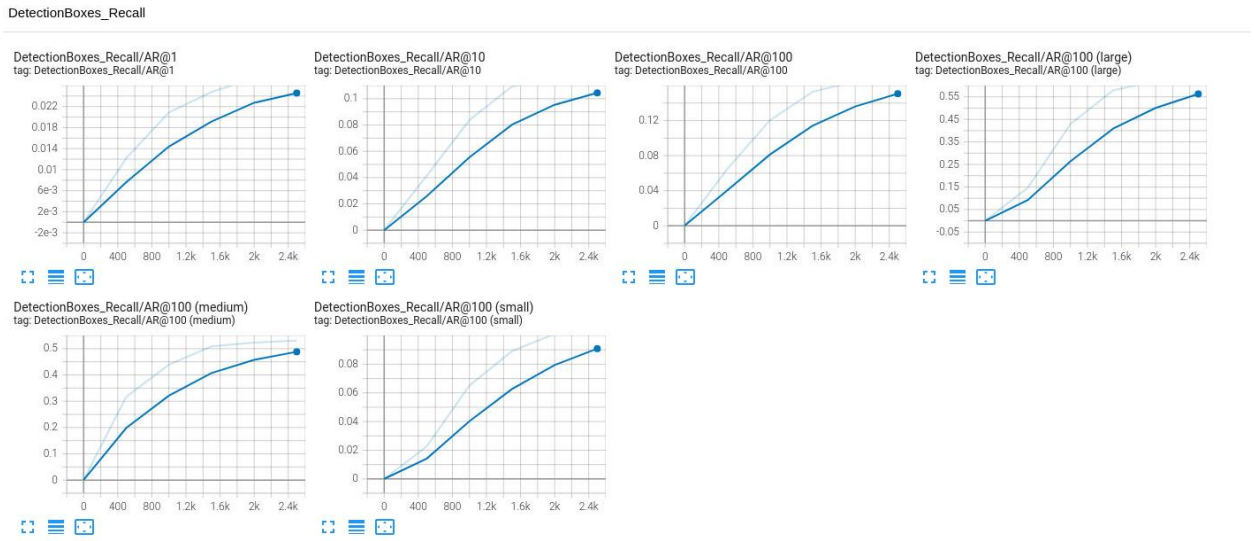
Loss:



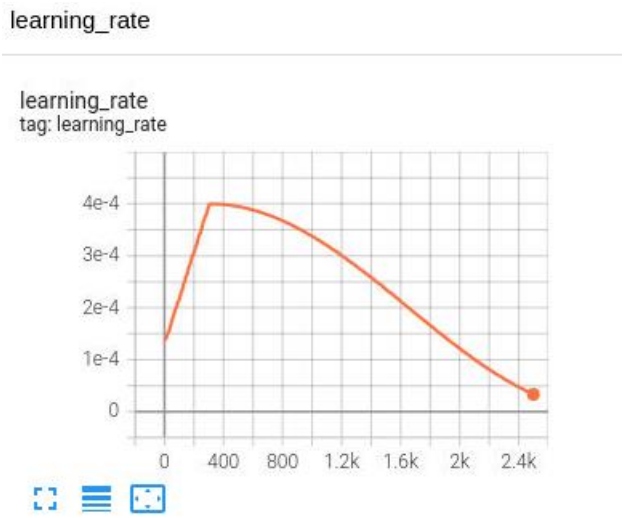
Precision:



Recall:



Learning Rate:



Loss shows a significant reduction in value compared to the reference training. Precision and Recall have improved as well when compared to reference training. This shows that the steps taken with this experiment would be the right direction to proceed further.

One potential room for improvement is the existing amount of overfitting that is happening with the model. Some of it is due to lack of data under class of pedestrians and cyclists other would-be due imperfect values of the hyperparameter of the optimizer.

This couldn't be explored further due to lack of GPU time available with the course.