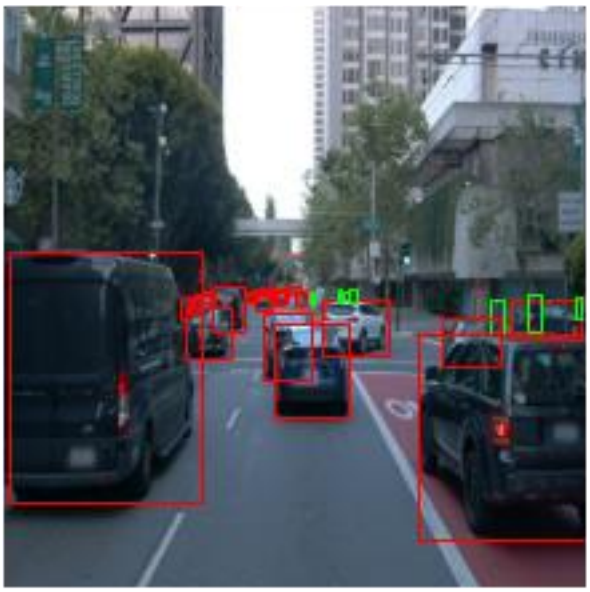
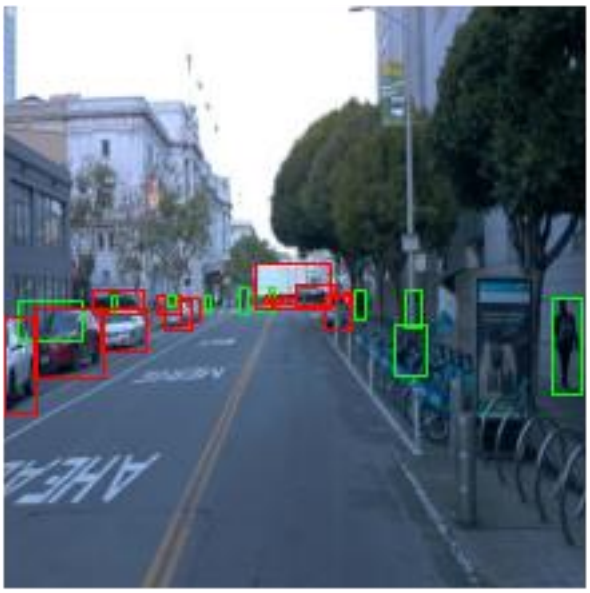
Object Detection in an Urban Environment

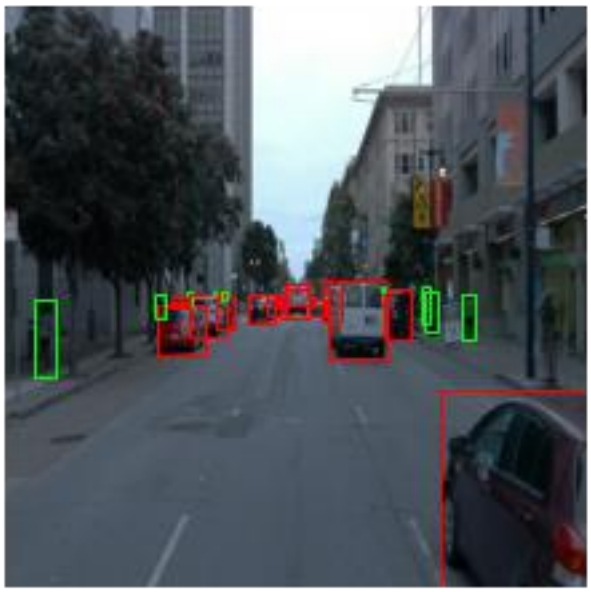
This project performs a convolutional neural network to detect and classify objects using Waymo Data set. The project is fulfilled in a 4 steps, where in step 1, the data set is investigated, and in step 2-4, the model is trained, evaluated and performance is improved iteratively to obtain a satisfactory result.

1. Exploratory data analysis

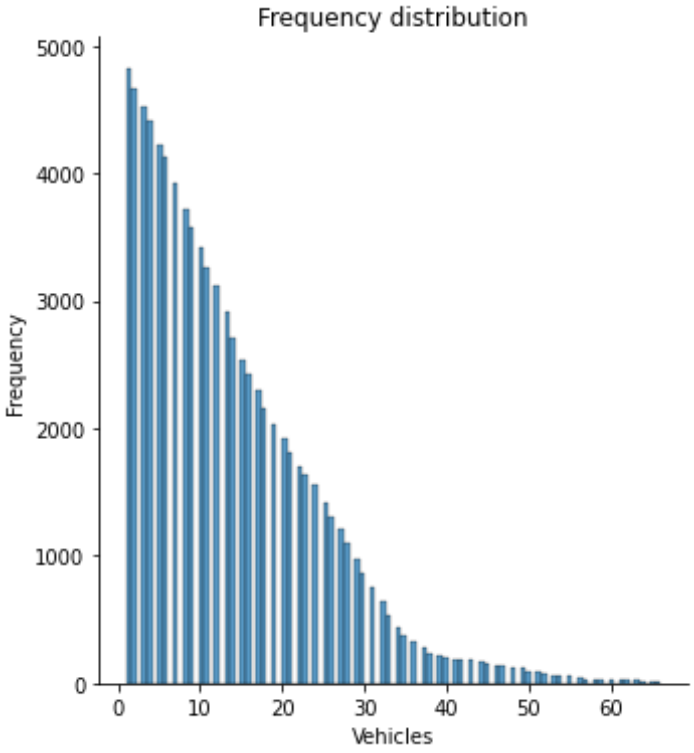
   In this section, the dataset is analyzed. In the following, 10 random images with bounding boxes are displayed. Also, some statistics of the images are extracted and shown below. Statistical analysis of the data. Brightness of images are also investigated in the next step. There are a lot of other process that can be performed on the images.

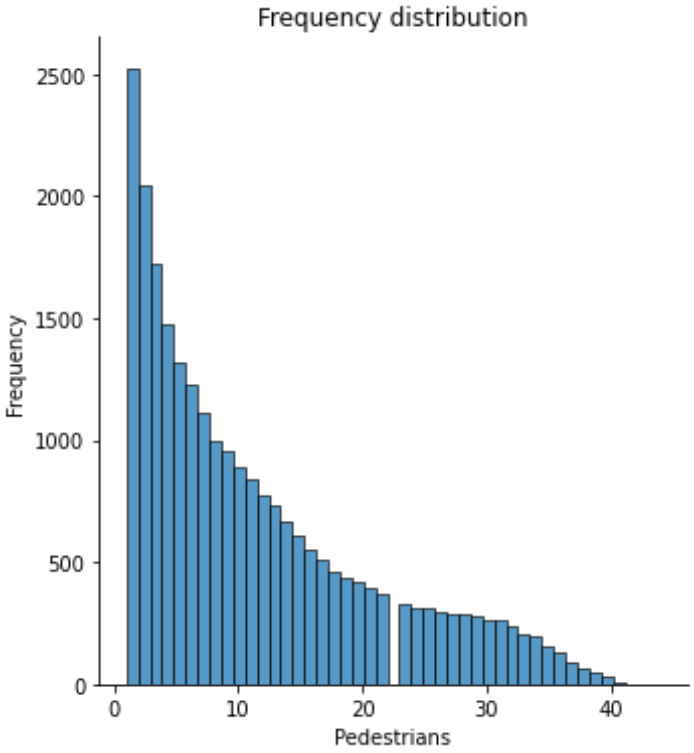


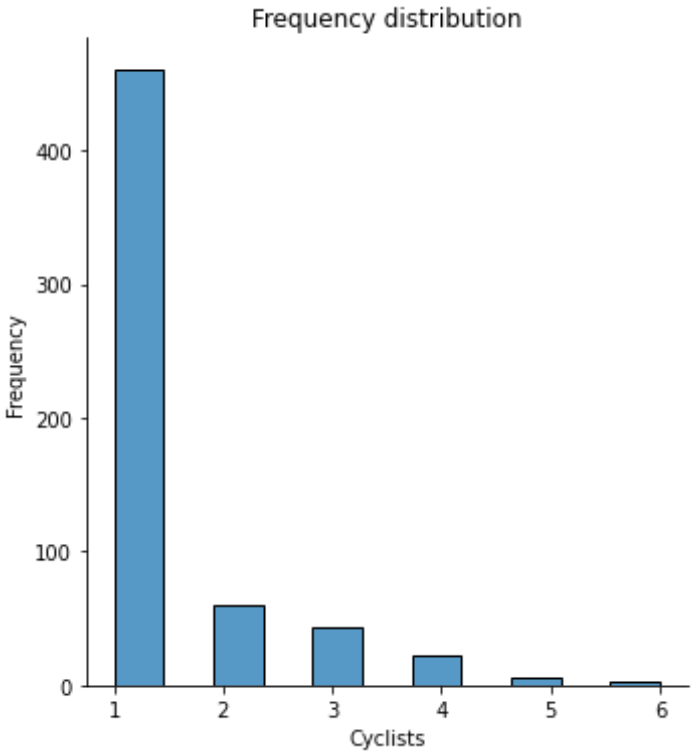




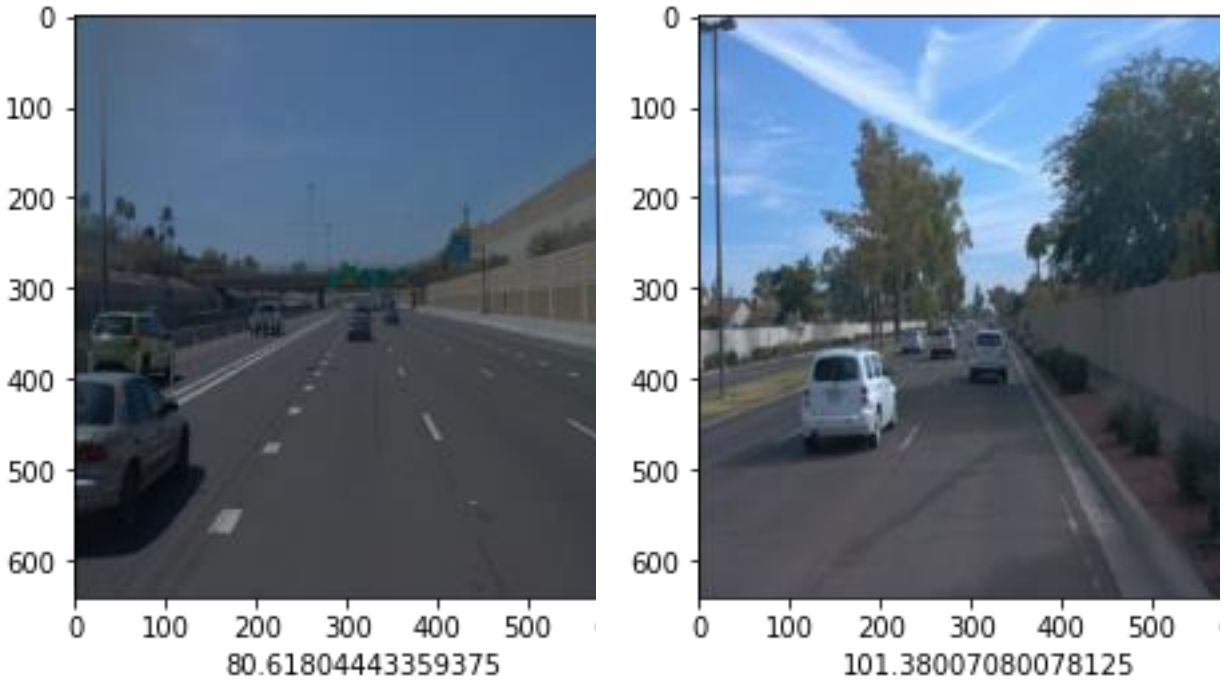
Frequency Distribution of objects in 5000 images:

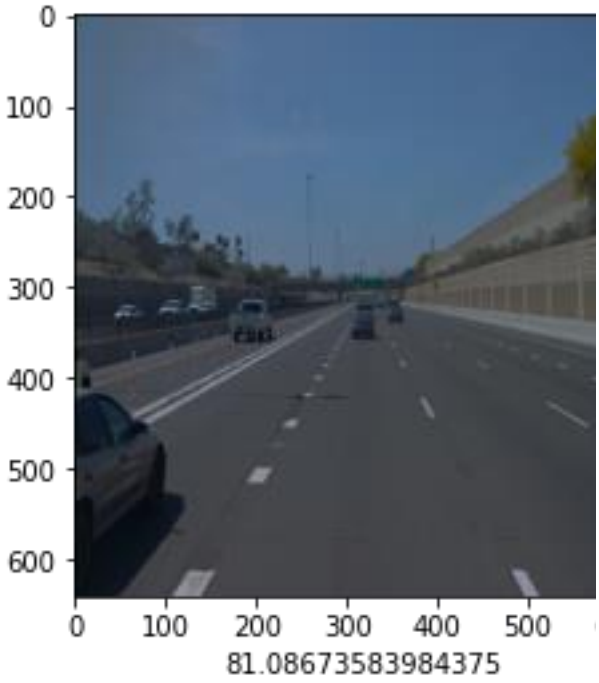






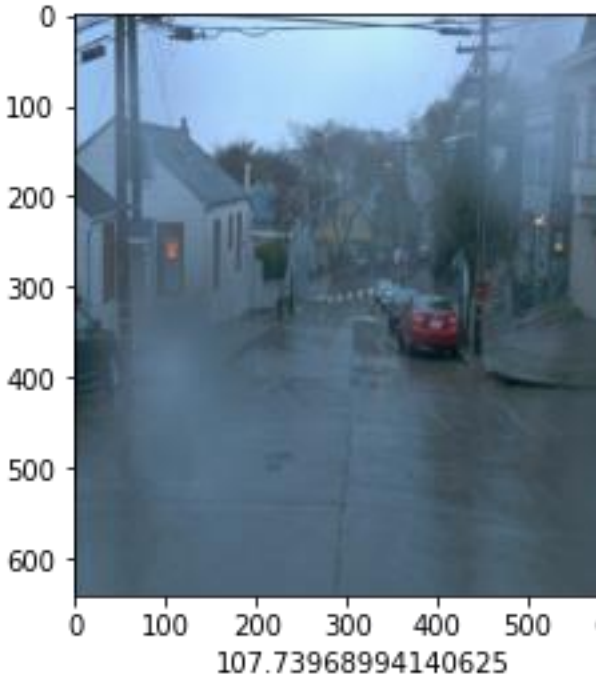
Brightness calculation of pictures (average brightness of each image is calculated). More statistics work can be done on this feature:





81.08673583984375

125.67002197265624



107.73968994140625

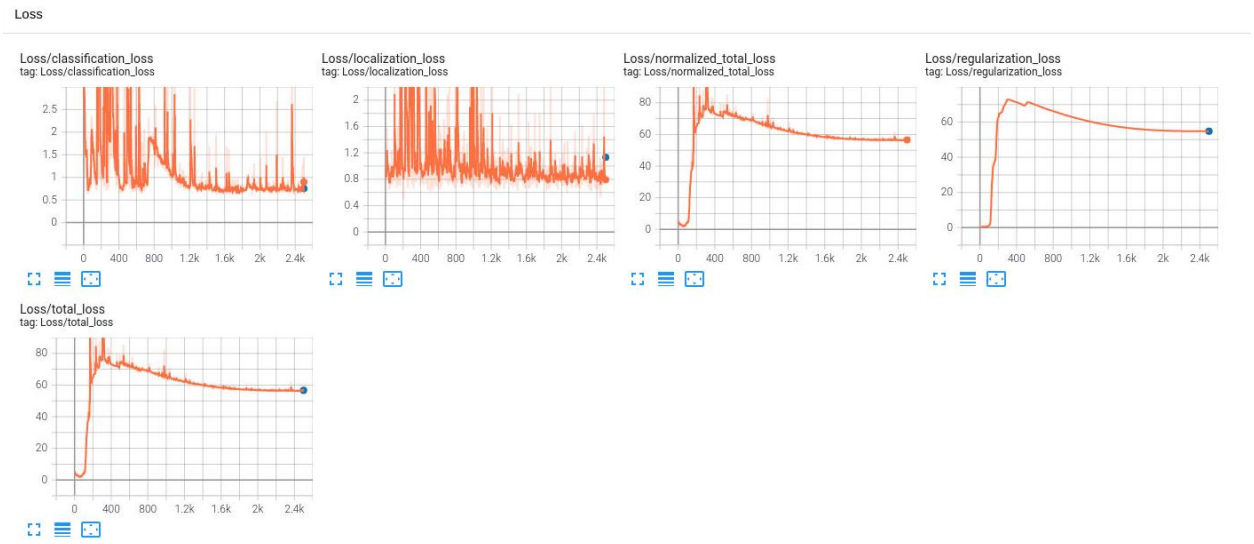
95.59715576171875

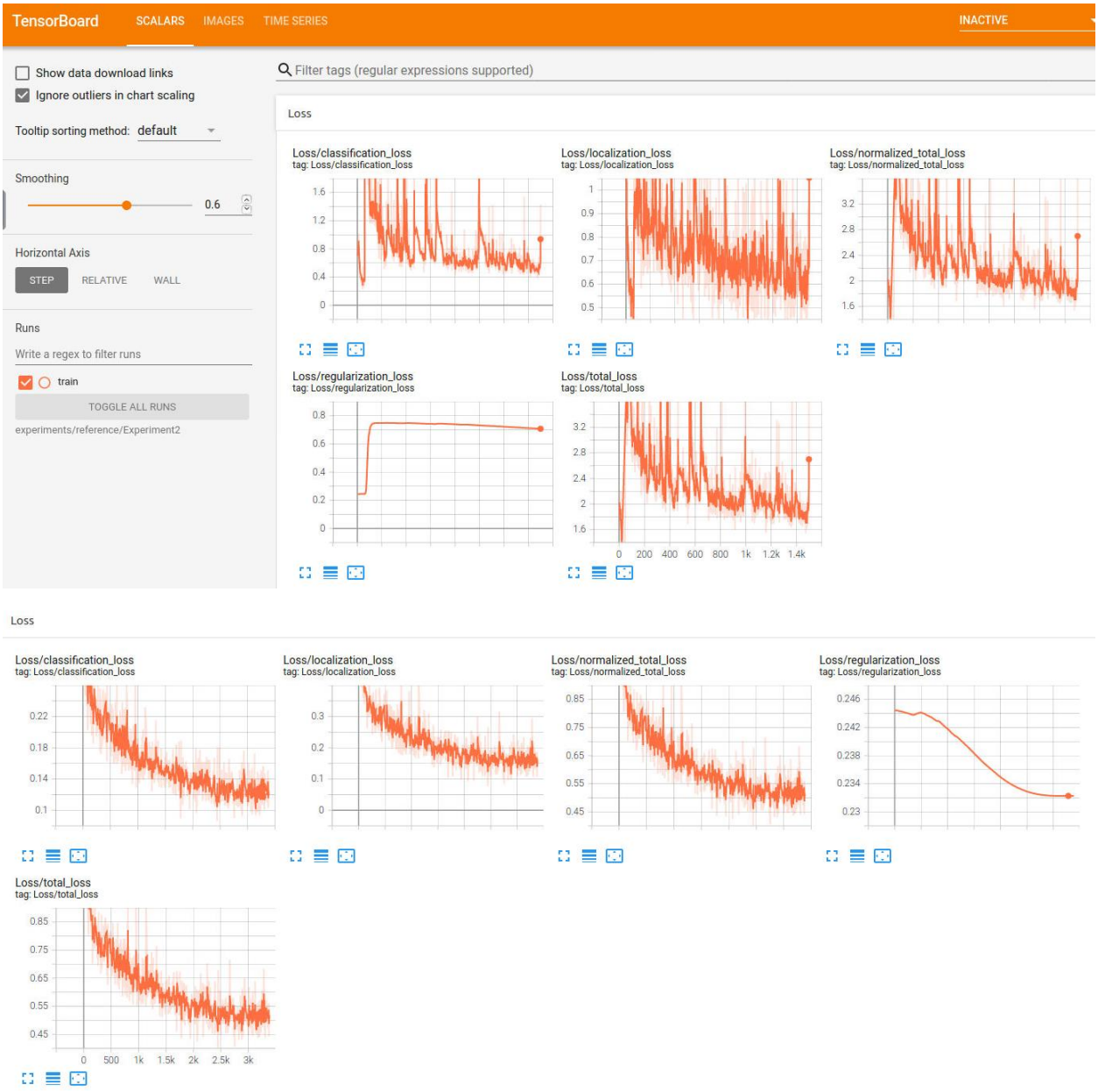
Step 2-4:

Edit Config File – Train Model and Evaluate – Improve Performance

As mentioned in this step, the goal is to train an appropriate CNN for object detection. For this purpose, different configuration parameters in config file are changed and tested to reach at better performance. Also, augmentation is applied to have a better reach set of data. Different augmentation strategies were also tested and tried. You can find the final config file in \experiments\reference. The number of batches is set to 8, and other parameters are also changed. In the following some of the results from tensor board is shown.

It is noted that running this process on virtual workspace is a tedious work, since for each training of a few thousand steps it was taking hours to train the model, and then to improve and re-iterate the process, all old data shall be deleted to find space. Tensorboard is also keep crashing. Therefore, with limited time of GPU available, this is the best result I could achieve. Nevertheless, it is possible to attain networks with better performance if a better environment was available for simulation.




TensorBoard
SCALARS
IMAGES
TIME SERIES
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
TOGGLE ALL RUNS
experiments/reference/Experiment2
Filter tags (regular expressions supported)
Loss
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
Loss
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

Final Result - Animation

The final results for two of test files is presented in the attached animation files. [Because of the limit on git, they are broken in 50M zip files. If you merge the files, you can watch the animation].