Object Detection in an Urban Environment

Project overview

A self driving car needs to be aware of its surroundings at all weather conditions , driving conditions and lighting conditions during the course of diving. It can come across other stationary objects like side poles , traffic signs garbage bins which needs to be avoided. Also other moving objects like pedestrians , bicyclists and other vehicles must be detected and tracked with high accuracy for successful self driving.

In this project we are doing transfer learning using a pretrained SSD Resnet 50 640x640 model. First s step we are feeding with a set of training images from Waymo open dataset to retrain the network to detect and classify 3 different classes of objects cars , pedestrians and cyclists. After initial training and validation , the goal is to adjust some of the hyper parameters to improve the model performance.

Set up

```
git clone https://github.com/BlueUnicorn7777/nd013-c1-vision-starter.git
```

```
cd nd013-c1-vision-starter
```

Local Setup and Data

I am using local docker setup for this project. Build the docker image.

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

For this project, I am using the pre processed data from the classroom workspace. Open your classroom workspace , right click on the data folder and download it as a zip file. Extract the zip file to your local data folder. Create a data folder and save this data in training_and_validation folder.

```
mkdir -p data/waymo/training_and_validation
```

Run the docker image using below command form host computer.

```
source dockerrun.sh
```

Dataset

Dataset analysis

Exploratory Data Analysis

Run the following commands within the docker container. Copy the URI from the container and open it on a browser on host computer. You should see the juypter notebook. Open and run the notebook Exploratory Data Analysis.ipynb.

```
cd /app/project
source jupyterrun.sh
```

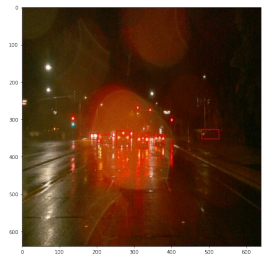
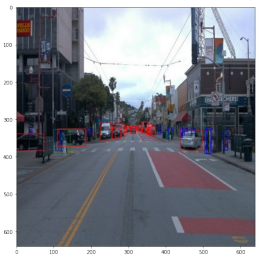
Image Quality - Looking at the images , we can see that there are varying light conditions – bright day light , dark night with street lights and head lamps. Also the images are taken during different weather condition like rainy night , foggy day . There are images distorted with weather conditions , and poor visibility. The images could be in a crowded city environment or highway driving with less clutter of buildings and pedestrians.

Class Distribution - Looking at the histogram of classes over some random 20000 images we can see that we have most labels associated with cars and fair amount of labels are for pedestrians. There are very few number of bicycles present in these images.

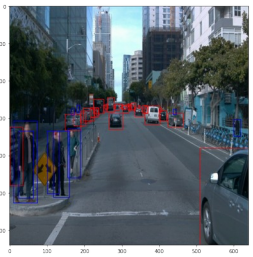
There are ~ 2200 or so images with no cars at all. Quite large number of images (~1500 each) have 10,20 or 30 cars present. Few images have very large >60 no of cars present.

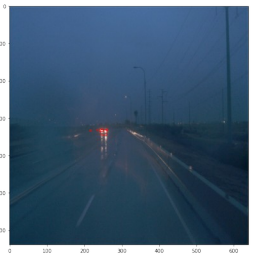
Most images >12000 or so images there are no pedestrians present. But fairly low number of images have between 10 and 25 pedestrians. Some images the count can go as high as > 40 or so.

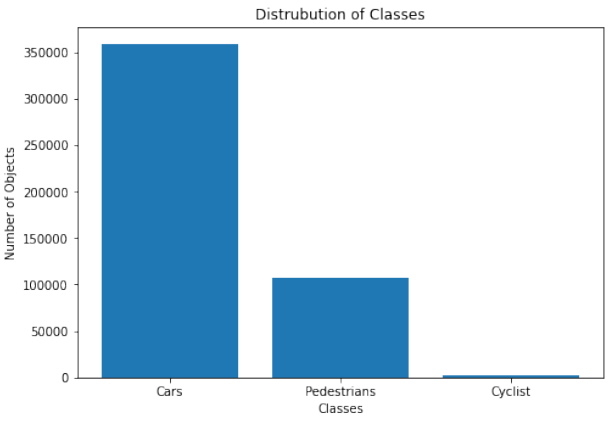
The bicycles distribution looks similar to pedestrians , with majority of (~17000) images without bicycles. A very few images with 1 to 6 bicycles present.

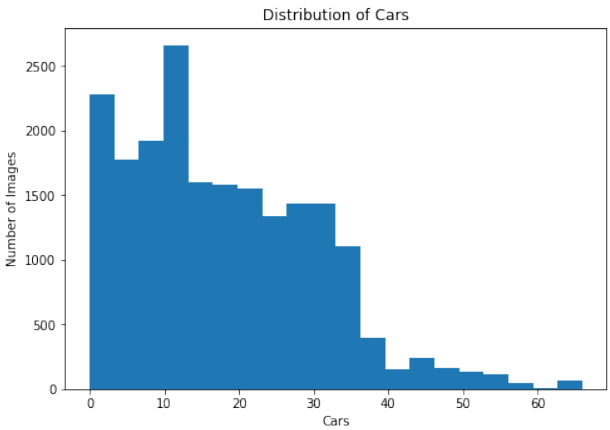



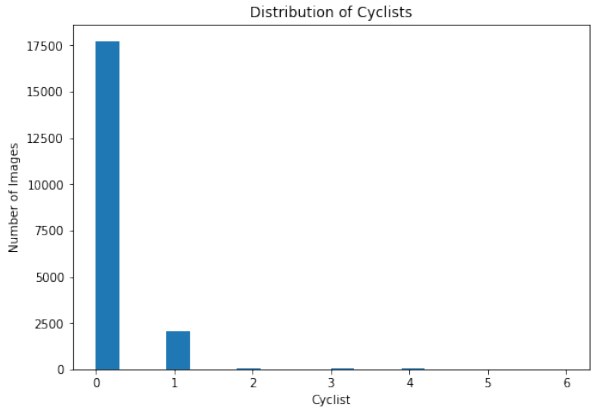


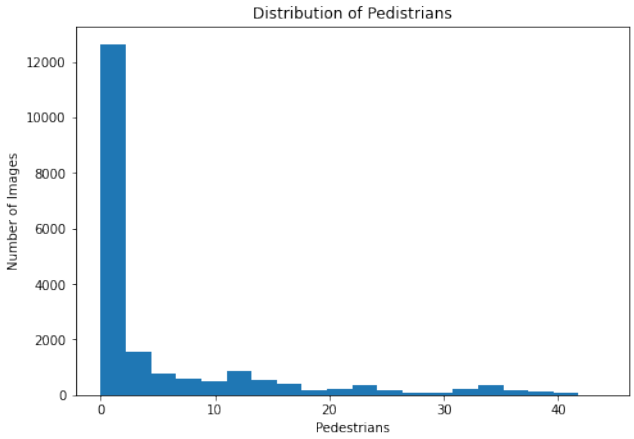




Distrubution of Classes
Number of Objects
350000
300000
250000
200000
150000
100000
50000
0
Cars
Pedestrians
Cyclist
Classes


Distribution of Cars
Number of Images
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
Cars





Cross validation

Create the splits

To create data splits please run the below command in docker container specifying the --source and --destination directory arguments to the create_splits.py.

```
cd /app/project

source create_split.sh

2022-02-26 18:10:44,289 INFO     Creating splits...

72 15 10

Creating symlink for train files

Creating symlink for validation files

Creating symlink for testing files
```

….

This script will first do a random shuffle on the filenames and split the data set into training = 75% , validation = 15% and testing = 10% of the data set files. The respective splits are then iterated over to create symlinks in the train , val and test folders under data folder.

We can see that there is a great class imbalance of cars vs bicycles vs pedestrians. The random sampling will even out this distribution. Also we would want the train and validation data set to have sample of images with varying conditions of light ,weather and driving scenarios.

I am creating symlinks here instead of moving or copying the files. Also downloaded 97 pre-processed files form classroom workspace waymo/training_and_validation folder. When using the classroom workspace unprocessed test dataset , we could split the data into 90% train and 10 % validate as an alternative.

Note : - there are some helpful commands in Support_Functions.ipynb.

Note: - edit the file /usr/local/lib/python3.8/dist-packages/numpy/core/function_base.py in your docker image change line no 120 as shown below.

num = operator.index(int(num))

Training

Reference experiment

For reference experiment a new pipeline_new.config is created as per the project instructions. The hyper parameters are left unchanged.

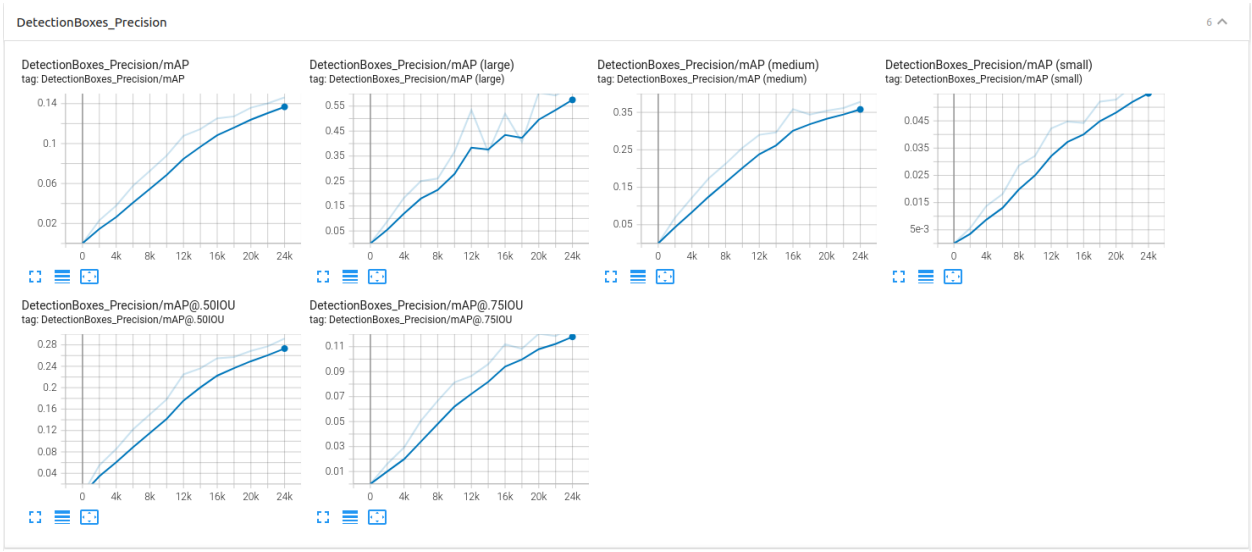
Start the training process and evaluation process in the docker image per project instruction.

```
python experiments/model_main_tf2.py --model_dir=training/reference/ --pipeline_config_path=training/reference/pipeline_new.config &
```

```
python experiments/model_main_tf2.py --model_dir=training/reference/ --pipeline_config_path=training/reference/pipeline_new.config --checkpoint_dir=training/reference &
```

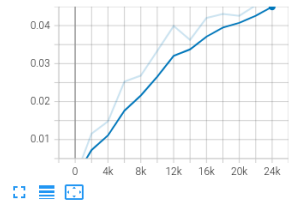
For monitoring the process with Tensorboard please open the showtensorboard.ipynb and run cells. Click to open the Tensorboard URL http://localhost:6006/

Below images show the tensorboard graphs.

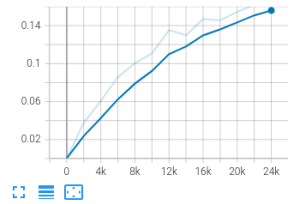


DetectionBoxes_Recall 6

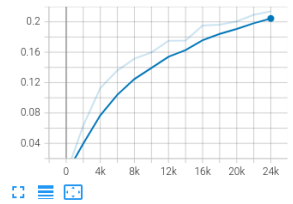
DetectionBoxes_Recall/AR@1
tag: DetectionBoxes_Recall/AR@1



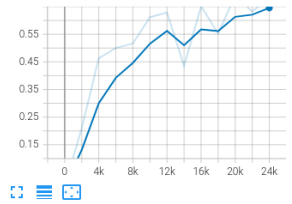
DetectionBoxes_Recall/AR@10
tag: DetectionBoxes_Recall/AR@10



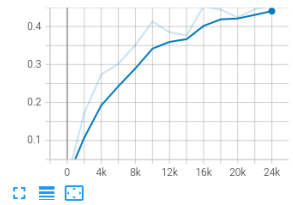
DetectionBoxes_Recall/AR@100
tag: DetectionBoxes_Recall/AR@100



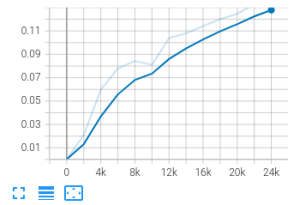
DetectionBoxes_Recall/AR@100 (large)
tag: DetectionBoxes_Recall/AR@100 (large)



DetectionBoxes_Recall/AR@100 (medium)
tag: DetectionBoxes_Recall/AR@100 (medium)

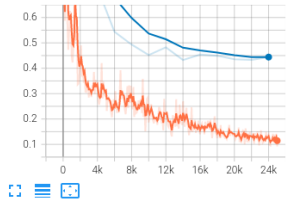


DetectionBoxes_Recall/AR@100 (small)
tag: DetectionBoxes_Recall/AR@100 (small)

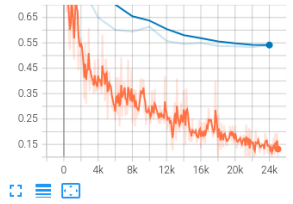


Loss 4

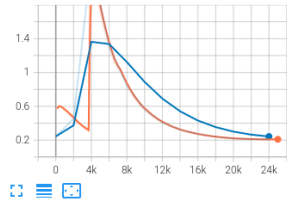
Loss/classification_loss
tag: Loss/classification_loss



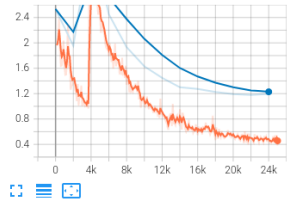
Loss/localization_loss
tag: Loss/localization_loss



Loss/regularization_loss
tag: Loss/regularization_loss

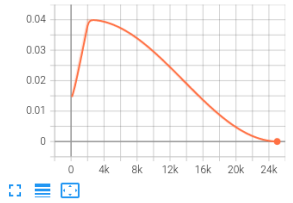


Loss/total_loss
tag: Loss/total_loss



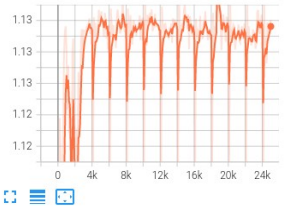
learning_rate

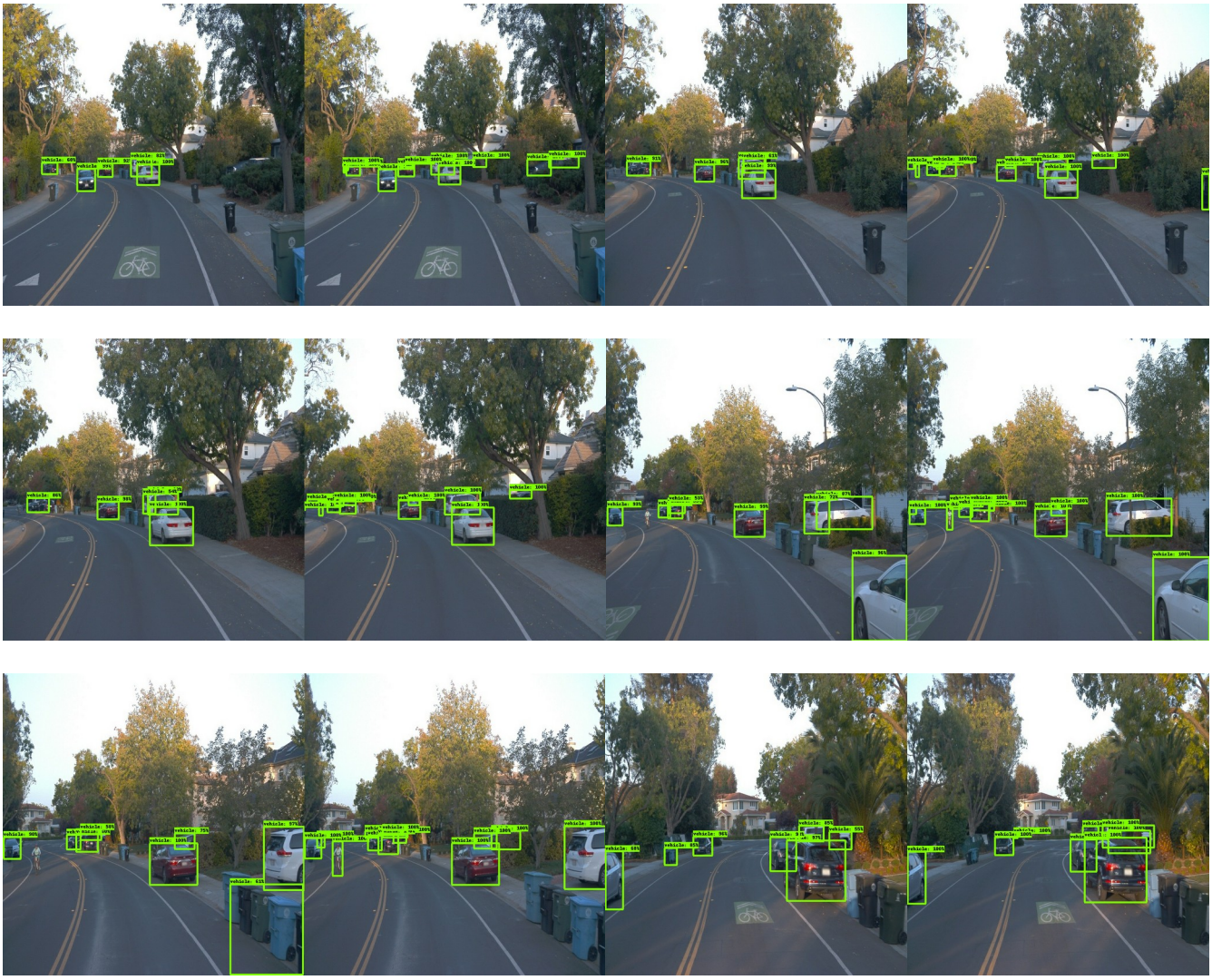
learning_rate
tag: learning_rate



steps_per_sec

steps_per_sec
tag: steps_per_sec



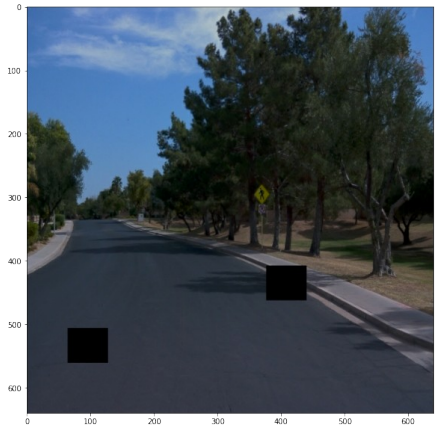


Improve on the reference

From the above graphs we can see that the model performance has room for improvement. The model is over fitting on the training data and we have higher validation loss. Looking at some of the images we can see the where the model is out putting FP and FN detection.
Highly occluded vehicles and very low visibility at the image border , the detection are missed by the model. Also our model is not very well detecting the bicycles class.
Some of the Garbage cans are miss classified as vehicles.
Since the data set selected has very low number of bicycles , we may need to look at adding more images with bicycles to train the model better.
Also adding a new class of garbage cans may help with reducing some of the FP.
For better detecting occluded vehicles , I am adding random black patch augmentation in second experiment.
Also adding random_adjust_contrast and random_adjust_brightness should help us with object detection with high brightness or darker driving conditions.

Data Augmentation Images



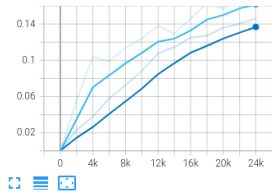
Improve on the reference

The model is run again through the training and evaluation process with the additional augmentations to the pipeline_new.config file. Below charts show slight improvements in performance across the board. We are able to reduce the validation loss although by small amount. The model is still over fitting and there is still room for improving the mAP , AR numbers. May be a different learning rate , scheduler or a different object detection model such as yolo could be investigated in further experiments.

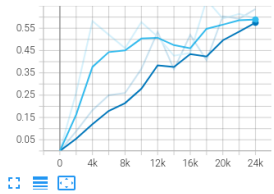
DetectionBoxes_Precision

6

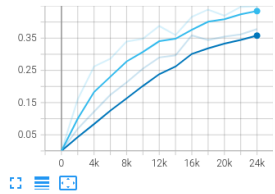
DetectionBoxes_Precision/mAP
tag: DetectionBoxes_Precision/mAP



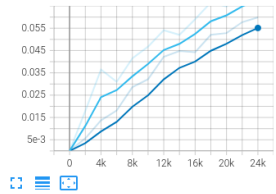
DetectionBoxes_Precision/mAP (large)
tag: DetectionBoxes_Precision/mAP (large)



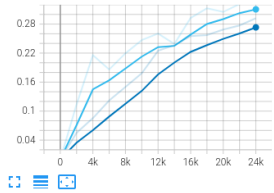
DetectionBoxes_Precision/mAP (medium)
tag: DetectionBoxes_Precision/mAP (medium)



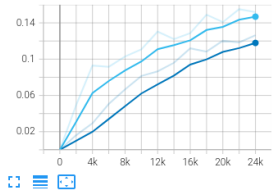
DetectionBoxes_Precision/mAP (small)
tag: DetectionBoxes_Precision/mAP (small)



DetectionBoxes_Precision/mAP@.50IOU
tag: DetectionBoxes_Precision/mAP@.50IOU



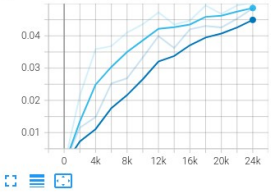
DetectionBoxes_Precision/mAP@.75IOU
tag: DetectionBoxes_Precision/mAP@.75IOU



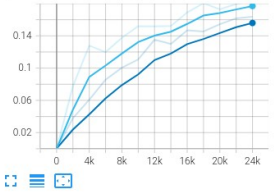
DetectionBoxes_Recall

6

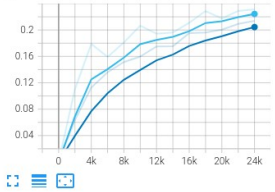
DetectionBoxes_Recall/AR@1
tag: DetectionBoxes_Recall/AR@1



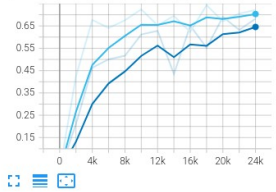
DetectionBoxes_Recall/AR@10
tag: DetectionBoxes_Recall/AR@10



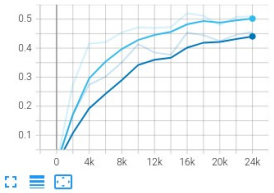
DetectionBoxes_Recall/AR@100
tag: DetectionBoxes_Recall/AR@100



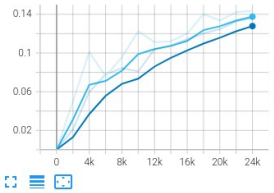
DetectionBoxes_Recall/AR@100 (large)
tag: DetectionBoxes_Recall/AR@100 (large)



DetectionBoxes_Recall/AR@100 (medium)
tag: DetectionBoxes_Recall/AR@100 (medium)

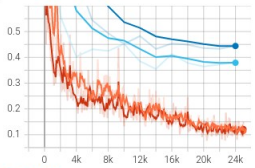


DetectionBoxes_Recall/AR@100 (small)
tag: DetectionBoxes_Recall/AR@100 (small)

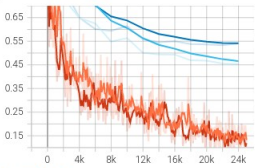


Loss

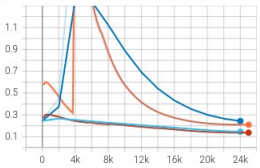
Loss/classification_loss
tag: Loss/classification_loss



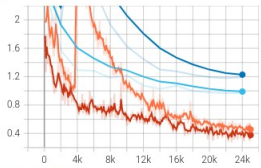
Loss/localization_loss
tag: Loss/localization_loss



Loss/regularization_loss
tag: Loss/regularization_loss



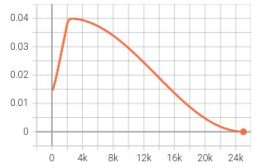
Loss/total_loss
tag: Loss/total_loss



	Name	Smoothed	Value	Step	Time	Relative
●	eval	0.4656	0.4528	24k	Sat Mar 12, 15:27:25	5h 36m 39s
●	exp0_eval	0.5415	0.5405	24k	Sat Mar 12, 03:37:11	5h 55m 10s
●	exp0_train	0.1573	0.1949	24.7k	Sat Mar 12, 03:45:32	6h 3m 39s
●	train	0.14	0.1262	24.7k	Sat Mar 12, 15:35:19	5h 47m 0s

learning_rate

learning_rate
tag: learning_rate



steps_per_sec
tag: steps_per_sec