Project - Write Up

Project overview

In this project, a model is to be trained and its configuration adjusted so that it can better localize and classify objects in the provided datasets. Three classes should be recognized: cars, pedestrians, and cyclists.

Set up

The project was executed in the provided workspace, which means that all dependencies that are already pre-installed in the workspace are not known. Therefore, it is recommended to clone the repository in the provided workspace and run it there.

The workspace is root of the repository

Pretrained model

The pretrained model "ssd_resnet50_v1_fpn_640x640_coco17_tpu-8" was downloaded from the tensorflow webside "http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz"

It was unziped within the workspace at "/experiments/pretrained_model/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/"

The containing pipeline.config file was used as inputfor the "edit_config.py" at the workspace rood location to generate a new pipeline configuration file "pipeline_new.config". This file was moved to the folder "/experiments/reference/" to be used as training configuration input.

In arguments for the "edit_config.py" containing the training dir at the workspace "/data/train/", eval_dir "/data/eval/", batch_size, the checkpoint to the pretrained model "/experiments/pretrained_model/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint/ckpt-0" and the provided label_map "/experiments/label_map.pbtxt"

Train the model

To train the model the "model_main_tf2.py" was used with arguments for the model_dir at "experiments/reference/" and the pipeline_config_path at "experiments/reference/pipeline_new.config".

After the number of steps defined in the pipeline_new.config file the traning is finished and can be evaluated.

The training progress can be displayed in the tensorboard.

Evaluate the model

To evaluate the model the "model_main_tf2.py" with same arguments as for training the model was executed. Additionally the argument to the checkpoint dir is needed "experiments/reference/". This will output metrics in the terminal as well as a eval log for the tensorboard.

Tensorboard

To start the tensorboard server the following command has to be executed "python -m tensorboard.main --logdir experiments/reference/" within this board the metric will be shown over the training steps.

Custom Augmentation Function

A custom augmentation file "custom_augmentations.py" was created at "/experiments/" in which all different augmentation function can be implemented. It was implemented a function fo a random_cutout of pixel boxes. This function was used in the "model_main_tf2.py" to preprocess the loaded images which will be used for training.

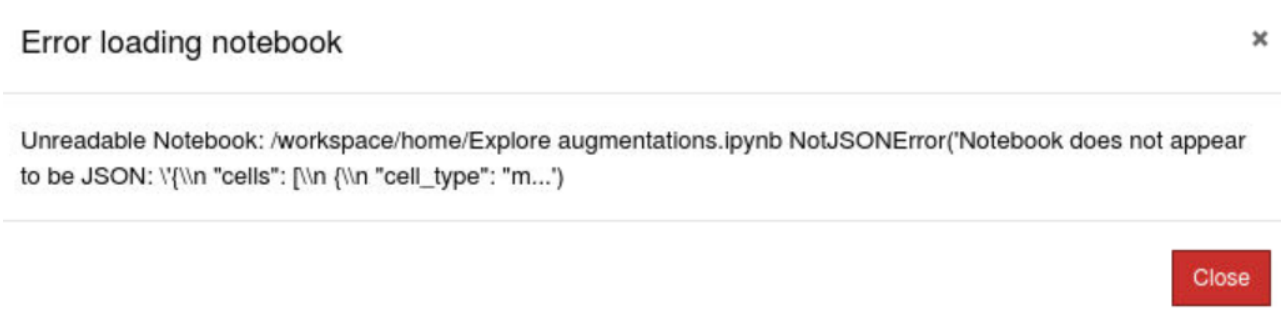
Exploratory data analysis

This is a jupiter notebook and needs to be opend within a jupyter notebook server by "jupyter notebook --port 3002 --ip=0.0.0.0 --allow-root"

In this notebook the training data will be evaluated by graping random 10 images to have a insight of the different training data or rather the weather conditions to set up the correct augmentations

Explore augmentations

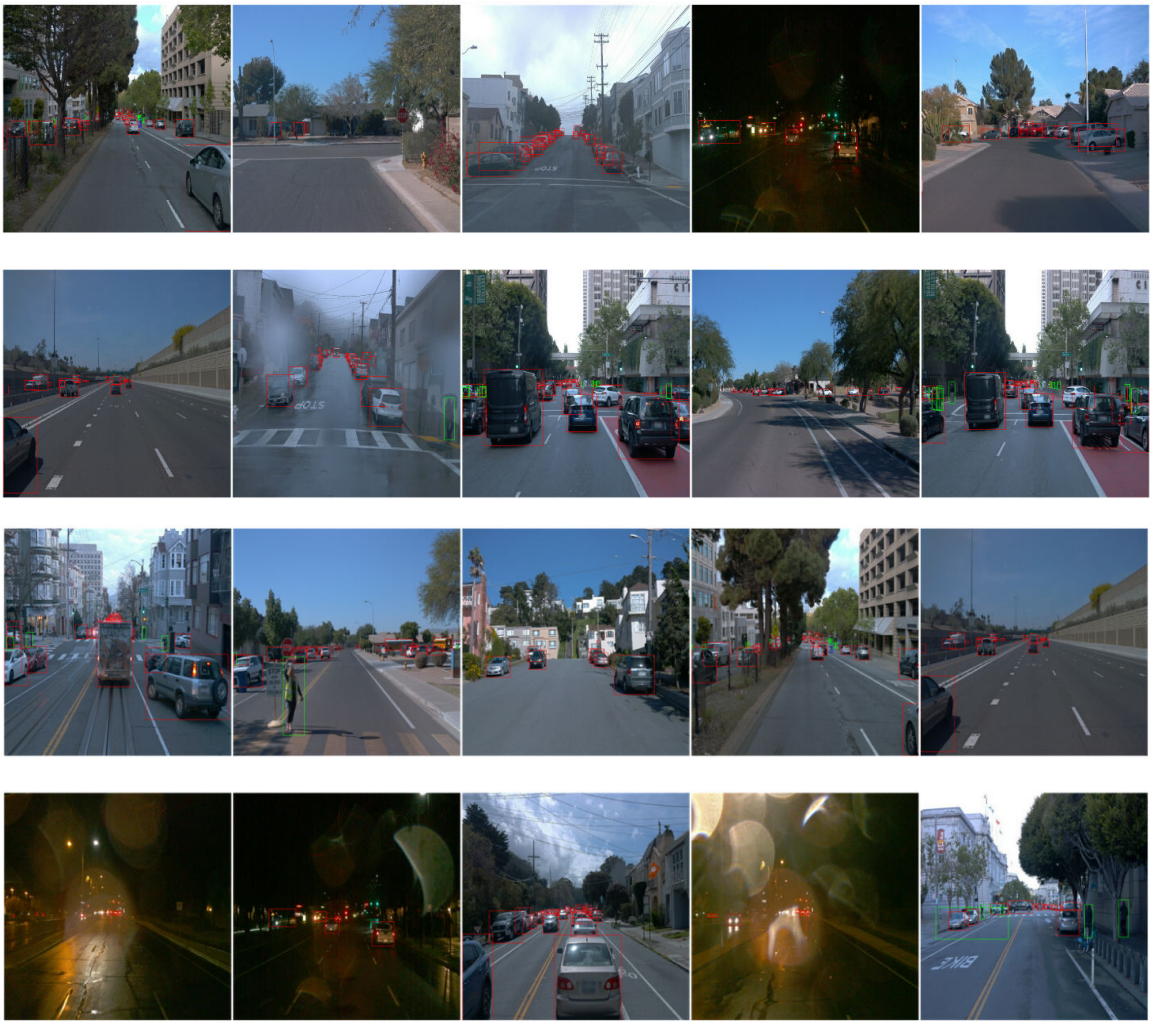
The jupiter notebook "exploer_augmentations" didn't work so the selection of augmentations was done just by description of those



Dataset

Dataset analyses

All datasets are randomly shuffled. Then, the first 10 datasets are used, and the first image from each is taken.



Here you can see a anlysis of a set of images according to the included classes:

```
Number of Images: 86
Number of Images with padestrians: 45
Number of Images with cyclists: 10
Number of Images with cars: 83
```

The images feature a variety of conditions, including sunny and rainy days, as well as nighttime and rainy scenes. Therefore, these different conditions need to be appropriately split between the training and validation sets. Additionally, the images contain varying numbers of detected classes, meaning not every image includes a cyclist or a pedestrian. To ensure a sufficient representation of these rare classes in both the training and validation sets, the data should be split according to the cross-validation strategy.

Cross validation

The cross-validation strategy involves using 80% of the available data for the training set and the remaining 20% for the validation set. However, the split will not be a simple 80/20 division. First, the images need to be categorized into the following categories:

- contains at least on car:
- contains at least on cyclist:

- contains at least on padestrian:
- contains no class

Based on the category with the fewest entries, this category will first be split 80/20. Then, for the next smallest category, the remaining images will also be split 80/20, continuing this process until all images are divided. This ensures that cyclists are learned in the training set but never validated.

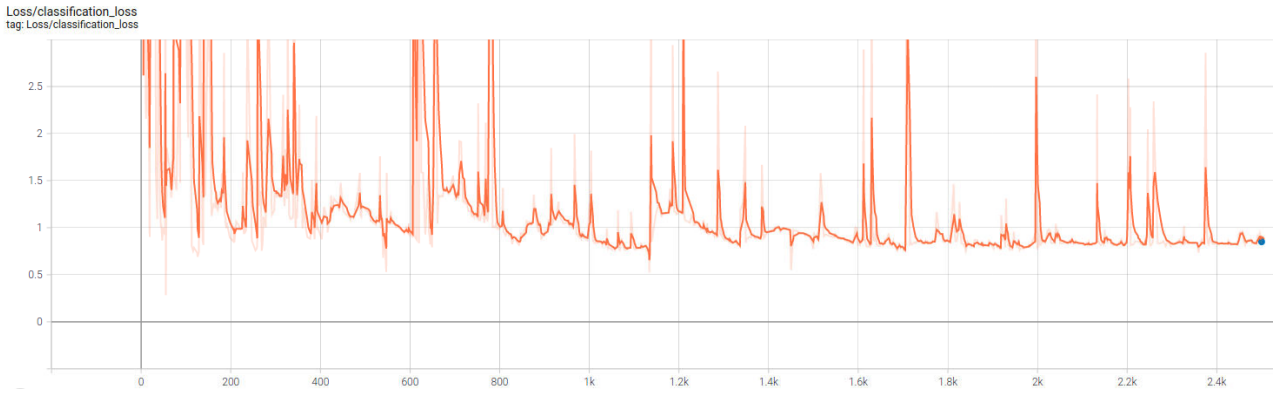
Next, it must be checked whether the weather conditions are appropriately distributed within the 80/20 split for the corresponding categories. This means that cars at night and in the rain must be both trained and validated. To achieve this, images from the training set can be swapped into the validation set as needed. This ensures that the ratio for the categories remains consistent.

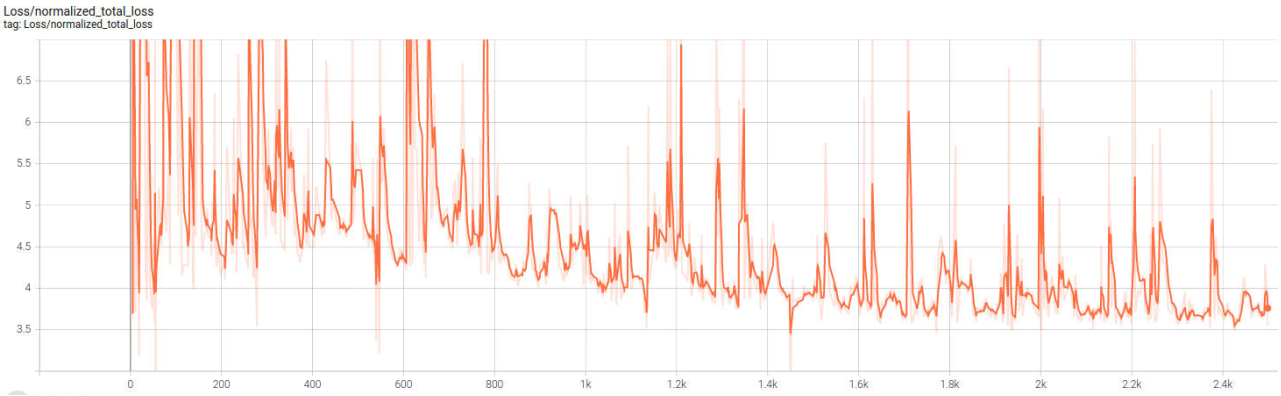
With this strategy, all possible scenarios should be evenly trained and validated.

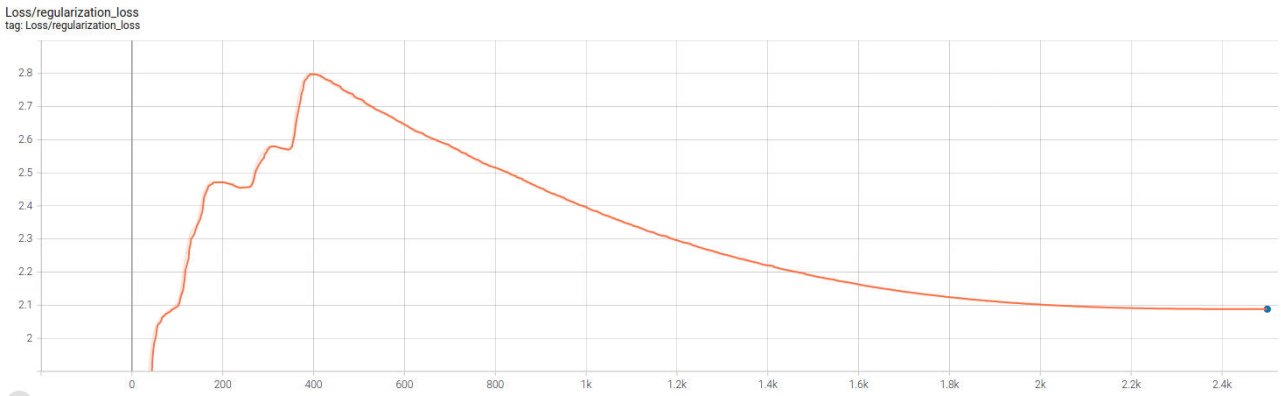
Model Training

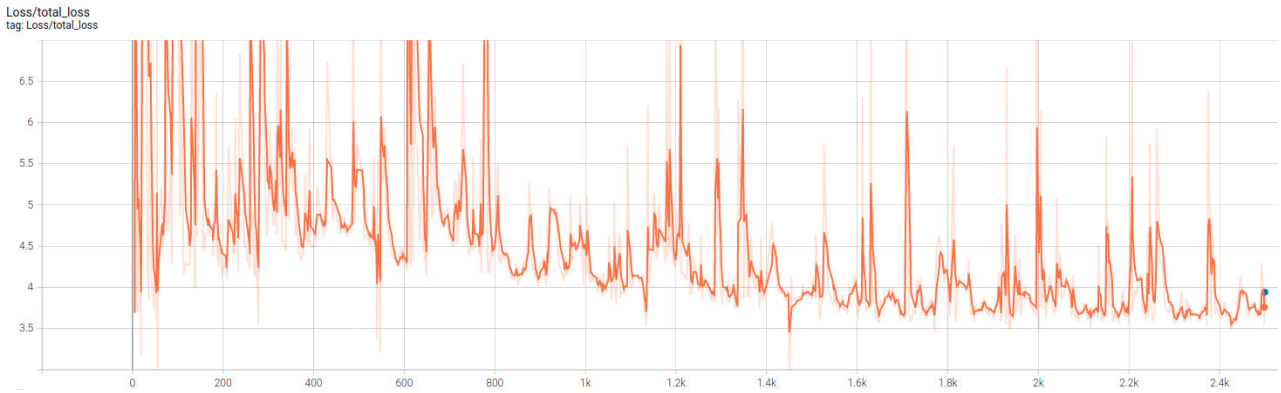
Reference experiment

Loss



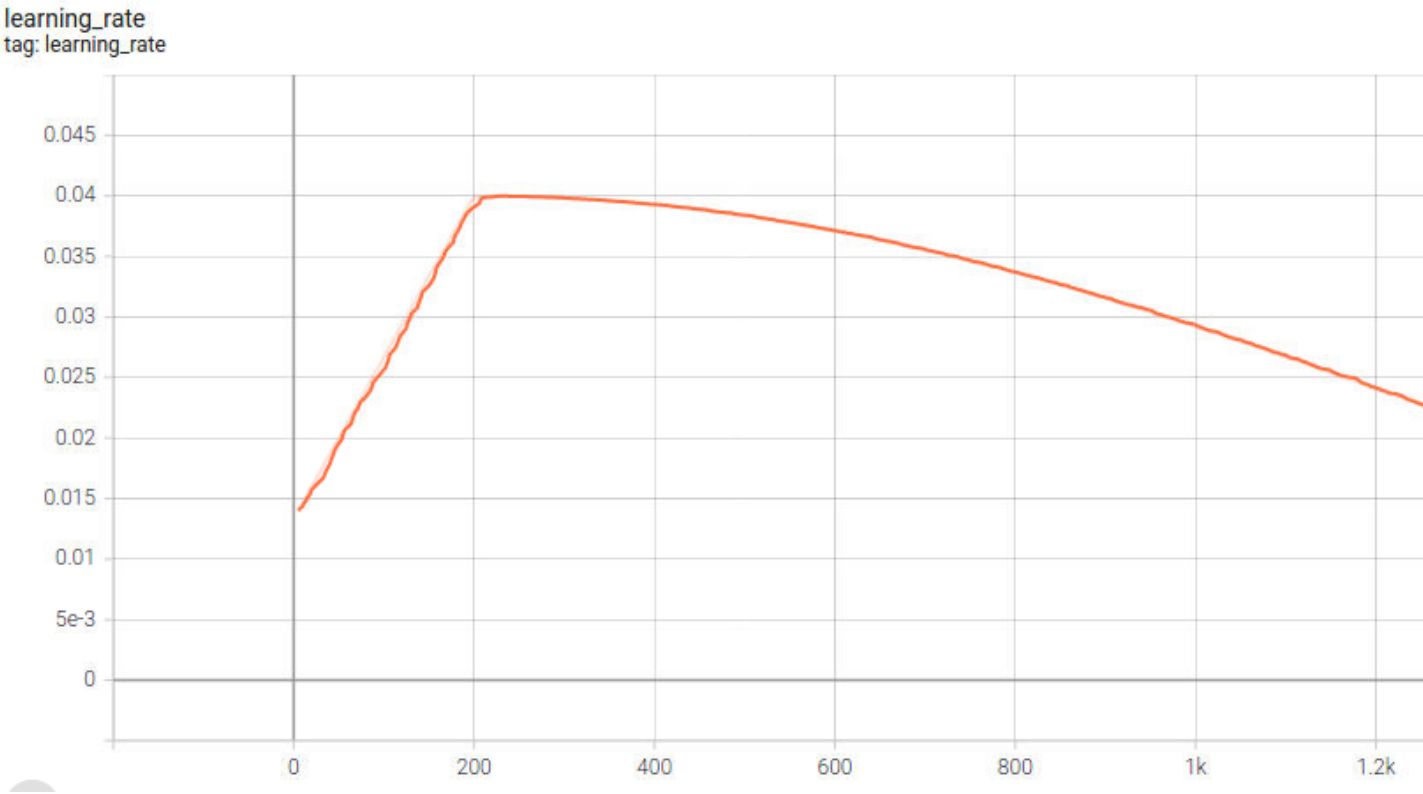






The classification, localization, and regularization loss are approaching a plateau, which means that the model is overfitting. This suggests that the loss on the validation data is higher than on the training data, which has also been observed.

Learn Rate



Output

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.001
 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 ] = 0.003
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.001
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.003
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.006
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.001
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.121
INFO:tensorflow:Eval metrics at step 2500
I1016 08:44:58.695125 139935232083712 model_lib_v2.py:988] Eval metrics at step 2500
INFO:tensorflow:    + DetectionBoxes_Precision/mAP: 0.000186
I1016 08:44:58.704513 139935232083712 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP: 0.000186
INFO:tensorflow:    + DetectionBoxes_Precision/mAP@.50IOU: 0.000664
I1016 08:44:58.706021 139935232083712 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP@.50IOU: 0.000664
INFO:tensorflow:    + DetectionBoxes_Precision/mAP@.75IOU: 0.000071
I1016 08:44:58.710376 139935232083712 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP@.75IOU: 0.000071
INFO:tensorflow:    + DetectionBoxes_Precision/mAP (small): 0.000000
```

```
I1016 08:44:58.711563 139935232083712 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP (small): 0.000000
INFO:tensorflow:    + DetectionBoxes_Precision/mAP (medium): 0.000009
I1016 08:44:58.712855 139935232083712 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP (medium): 0.000009
INFO:tensorflow:    + DetectionBoxes_Precision/mAP (large): 0.003045
I1016 08:44:58.714114 139935232083712 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP (large): 0.003045
INFO:tensorflow:    + DetectionBoxes_Recall/AR@1: 0.000619
I1016 08:44:58.715449 139935232083712 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@1: 0.000619
INFO:tensorflow:    + DetectionBoxes_Recall/AR@10: 0.002615
I1016 08:44:58.716647 139935232083712 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@10: 0.002615
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100: 0.006159
I1016 08:44:58.717833 139935232083712 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100: 0.006159
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100 (small): 0.000000
I1016 08:44:58.718954 139935232083712 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100 (small): 0.000000
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100 (medium): 0.000896
I1016 08:44:58.720100 139935232083712 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100 (medium): 0.000896
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100 (large): 0.121000
I1016 08:44:58.721437 139935232083712 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100 (large): 0.121000
INFO:tensorflow:    + Loss/localization_loss: 1.004677
I1016 08:44:58.722437 139935232083712 model_lib_v2.py:991]  + Loss/localization_loss:
1.004677
INFO:tensorflow:    + Loss/classification_loss: 0.849736
I1016 08:44:58.723536 139935232083712 model_lib_v2.py:991]  + Loss/
classification_loss: 0.849736
INFO:tensorflow:    + Loss/regularization_loss: 2.088152
I1016 08:44:58.724582 139935232083712 model_lib_v2.py:991]  + Loss/
regularization_loss: 2.088152
INFO:tensorflow:    + Loss/total_loss: 3.942566
I1016 08:44:58.725620 139935232083712 model_lib_v2.py:991]  + Loss/total_loss:
3.942566
```

Improve on the reference

First Iteration

Model Changes

Config File changes and expactations

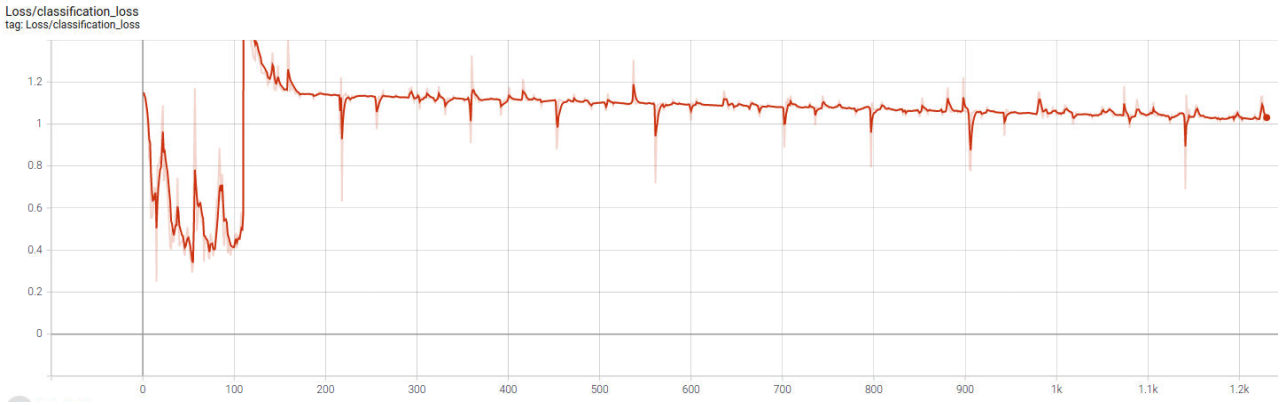
1. **Increasing the complexity:** The number of layers before the predictor was increased from 4 to 6. This is intended to improve the model's ability to localize and classify objects
2. **Increasing the batch size**: Increasing the batch size is intended to make the model learn more robustly and reduce fluctuations due to outliers.
3. **Change of learning schedule**r: Changing the learning schedule from cosine to exponential so that the model converges faster with fewer cycles.
4. **Adapting the learning rate:** Reduce learning rate get a more stable lerning progress

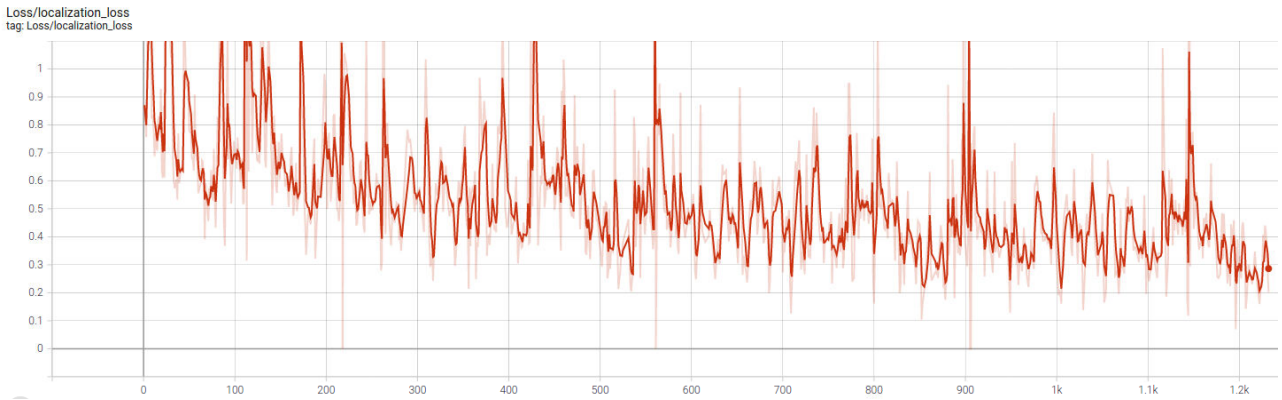
Reality Behvoir

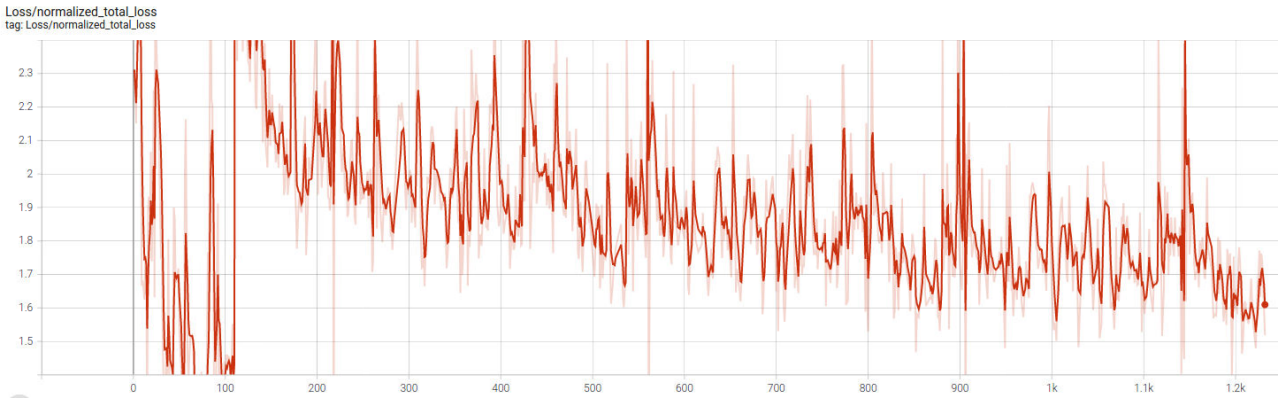
1. The localization has significantly improved due to the changes, but the classification has not improved or has even become slightly worse.
2. Increasing the batch size and reduction of learning rate led to more stable learning with fewer fluctuations.
3. The change of the sheduling has not lead to a faster converges but this could be because of the reduction of the learning rate

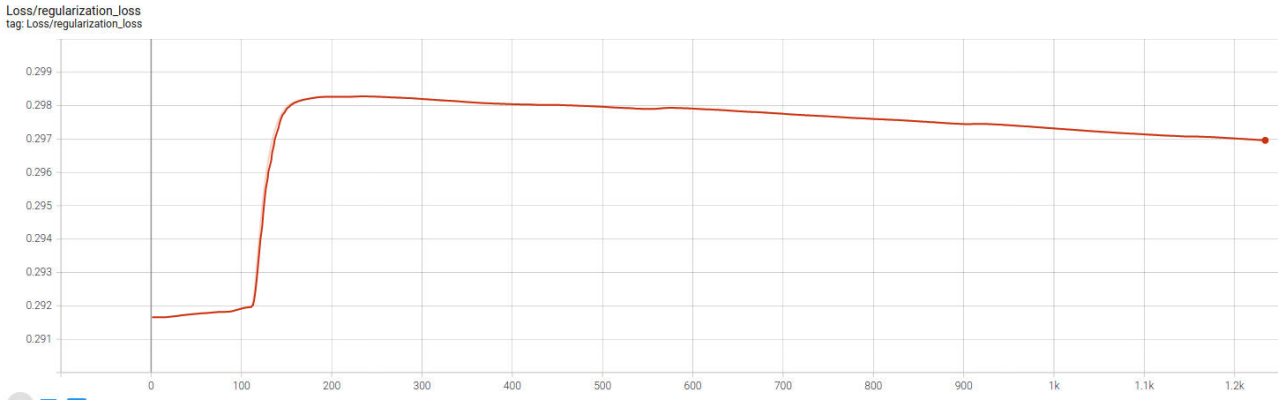
Tensorboard Diagrams

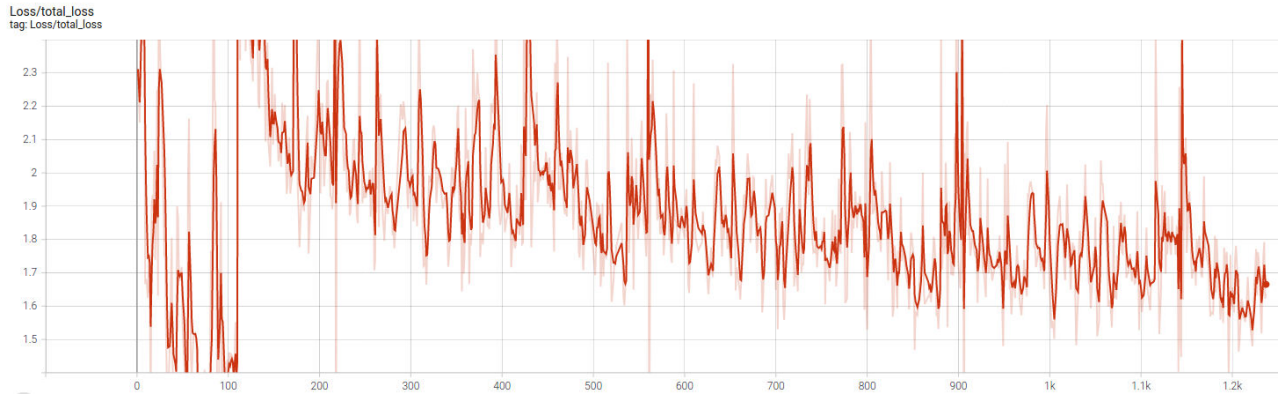
Loss



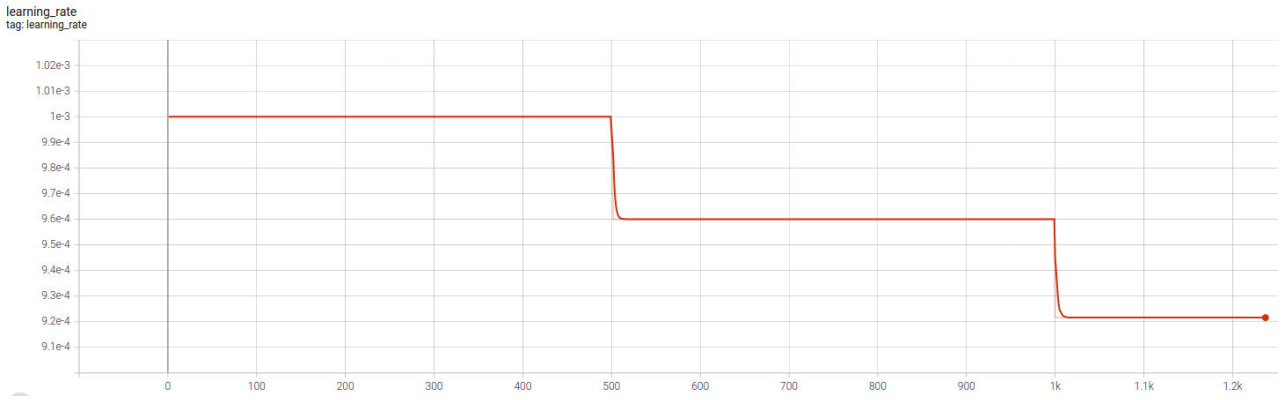








Learn Rate



Output

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.001
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 ] = 0.012
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 ] = 0.009
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.006
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.005
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.074
INFO:tensorflow:Eval metrics at step 2500
I1016 10:07:53.100537 140381778958080 model_lib_v2.py:988] Eval metrics at step 2500
INFO:tensorflow:    + DetectionBoxes_Precision/mAP: 0.000253
I1016 10:07:53.111101 140381778958080 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP: 0.000253
INFO:tensorflow:    + DetectionBoxes_Precision/mAP@.50IOU: 0.000982
I1016 10:07:53.112813 140381778958080 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP@.50IOU: 0.000982
INFO:tensorflow:    + DetectionBoxes_Precision/mAP@.75IOU: 0.000054
I1016 10:07:53.114117 140381778958080 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP@.75IOU: 0.000054
INFO:tensorflow:    + DetectionBoxes_Precision/mAP (small): 0.000048
I1016 10:07:53.115391 140381778958080 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP (small): 0.000048
INFO:tensorflow:    + DetectionBoxes_Precision/mAP (medium): 0.000180
I1016 10:07:53.116673 140381778958080 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP (medium): 0.000180
INFO:tensorflow:    + DetectionBoxes_Precision/mAP (large): 0.012384
I1016 10:07:53.117954 140381778958080 model_lib_v2.py:991]  +
DetectionBoxes_Precision/mAP (large): 0.012384
INFO:tensorflow:    + DetectionBoxes_Recall/AR@1: 0.000120
I1016 10:07:53.119202 140381778958080 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@1: 0.000120
INFO:tensorflow:    + DetectionBoxes_Recall/AR@10: 0.002076
I1016 10:07:53.120432 140381778958080 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@10: 0.002076
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100: 0.009044
I1016 10:07:53.122040 140381778958080 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100: 0.009044
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100 (small): 0.005674
I1016 10:07:53.123340 140381778958080 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100 (small): 0.005674
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100 (medium): 0.005299
I1016 10:07:53.125269 140381778958080 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100 (medium): 0.005299
INFO:tensorflow:    + DetectionBoxes_Recall/AR@100 (large): 0.074200
I1016 10:07:53.126665 140381778958080 model_lib_v2.py:991]  + DetectionBoxes_Recall/
AR@100 (large): 0.074200
INFO:tensorflow:    + Loss/localization_loss: 0.490712
I1016 10:07:53.127790 140381778958080 model_lib_v2.py:991]  + Loss/localization_loss:
0.490712
INFO:tensorflow:    + Loss/classification_loss: 0.907277
I1016 10:07:53.128888 140381778958080 model_lib_v2.py:991]  + Loss/
classification_loss: 0.907277
INFO:tensorflow:    + Loss/regularization_loss: 0.295589
I1016 10:07:53.129986 140381778958080 model_lib_v2.py:991]  + Loss/
regularization_loss: 0.295589
INFO:tensorflow:    + Loss/total_loss: 1.693577
I1016 10:07:53.131019 140381778958080 model_lib_v2.py:991]  + Loss/total_loss:
1.693577
```

Second iteration

Augmentation Options

General benefits of the augmentations options:

Increase in Data Diversity: By rotating images in different directions, the number of unique training examples is increased, which helps the model generalize better.

Avoidance of Overfitting: By varying the training data, the model is less likely to learn specific features of the training data that are not present in the test data. This helps to avoid overfitting.

Roation

Robustness to Orientations: In many real-world applications, objects can appear in various orientations. Through rotation, the model learns to recognize objects regardless of their orientation.

```
data_augmentation_options {
  random_rotation {
    min_angle: -15
    max_angle: 15
  }
}
```

Brightness Adjustment

Robustness to Lighting Conditions: The dataset contains images taken during the day and in sunny conditions. By adjusting brightness, the model learns to recognize objects in both low and high light conditions.

```
data_augmentation_options {
  random_adjust_brightness {
    max_delta: 0.2
  }
}
```

Contrast Adjustment

Improved Visibility of Features: By adjusting the contrast, the differences between light and dark areas of an image are enhanced or reduced. This makes features more noticeable (during the day and in sunny conditions) or less noticeable (at night).

```
data_augmentation_options {
  random_adjust_contrast {
    min_delta: 0.8
    max_delta: 1.2
  }
```

```
}
```

Random Cutout

Robustness to Occlusions: In the training data, there are situations where raindrops on the camera partially obscure objects. Random Cutout helps the model handle such situations better by teaching it to make accurate predictions even when objects are partially obscured. This functionality is provided via a custom augmentation function.

Saturation Reduction

Robustness to Color Variations: In the training data, there are images with different weather and lighting conditions, causing the colors in the images to vary. By reducing saturation, the model becomes more robust to such variations.

```
data_augmentation_options {
  random_adjust_saturation {
    min_delta: 0.1
    max_delta: 0.5
  }
}
```

Hue Adjustement

Improved Color Differentiation: Due to varying lighting conditions, adjusting the hue can highlight subtle differences in colors, enhancing the model's ability to distinguish between different objects and scenes.

```
data_augmentation_options {
  random_adjust_hue {
    max_delta: 0.1
  }
}
```