Object Detection in an Urban Environment

Self-Driving Car Engineer Nanodegree

Submitted by:
Abhishek U H

1. Project Overview:

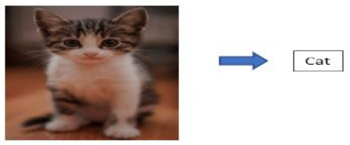
In project- object detection in urban environment, we will be using the existed pre-trained model from the tensorflow object detection API. We will be using Single Shot Detector-SSD Resnet 50 as initial pretrained model for our project and train it with the camera data present in waymo-open-dataset. The purpose of the project is to get the model classify and localize the object correctly and also be try not to overfit to the training data set by keeping the bias-loss tradeoff to be optimum value.

Importance of Object Detections in Self Driving Cars:

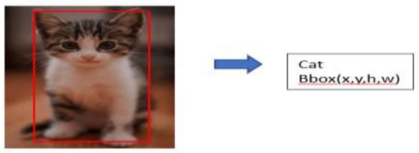
Object detections is a computer vision technique that is used for detecting and locating objects on the image or video. Usually, when we train the classification model, we train it with the images that contains only one object which gets classified. But in the real world example when the camera captures the images of the surrounding, it will not necessarily have an environment where there is only one object in the environment, there will multiple objects in the environment in same image and we might need to separate it out from the image and classify these objects.

Consider an example, if we have a classifier model that can classify image into dog , cat and human, given an image that has all three objects in it and passed to the classifier model to predict, the model can predict any one of them or unpredicted behavior can occur, so it is important that we separate out objects in image and classify those object.

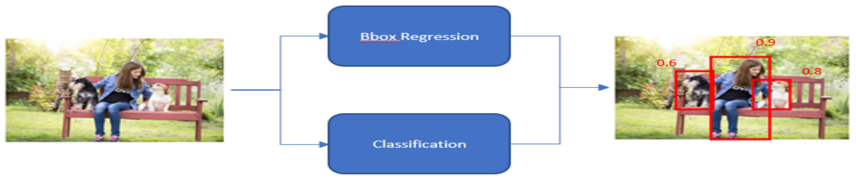
- Object Classification: Object classification refers to classifying the image with single object to one of the class.



- Object Localization: Object Localization refers to identifying the location of the object in the image.



- Object Detection: Objects detections refers to classification and localization of multi-objects in the given image.



For object detection we have tensorflow object detection API Pool that has several CNN Architecture which is capable of separating out objects in the complex image (in this example – cat, dog, human) and classify these objects to one of the defined class. There are many CNN architecture that detects objects in the image like R-CNN, Fast R-CNN , YOLO and SSD. This project makes use of pretrained SSD model for object detections.

2. Set Up:

In the Udacity Virtual Machine.

1. Install Chromium browser in the Udacity VM.

```
• sudo apt-get update
• sudo apt-get install chromium-browser
```

2. To launch the Jupyter Notebook open terminal box from the VM

```
• cd /home/workspace/
• jupyter notebook --port 3002 –ip 0:0:0:0 –allow-root
```

3. Open Chromium Browser (from new terminal type command -sudo chromium-browser --no-sandbox) paste the link displayed in step2 terminal. (127.0.0.1:3002). Run the Script to visualize image data
4. From new terminal window- type below command

```
• cd /home/workspace/experiments/pretrained_model/
• wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
• tar -xvzf ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
• rm -rf ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
```

5. For Training the SSD – Model , we need to first edit the model config file and the run the training process

```
• cd /home/workspace/
```

- python edit_config.py --train_dir /home/workspace/data/train/ --eval_dir /home/workspace/data/val/ --batch_size 2 –checkpoint /home/workspace/experiments/pretrained_model/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint/ckpt-0 --label_map /home/workspace/experiments/label_map.pbtxt
- python experiments/model_main_tf2.py --model_dir=experiments/reference/ --pipeline_config_path=experiments/reference/pipeline_new.config

6. For Evaluating the trained Model

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

Note: Above command needs to be run for number ckpt files by changing the checkpoint file parameter “model_checkpoint_path: ckpt-6” from 1 to N(number of ckpt files) so that we get a proper plot on the eval tensorboard else the evaluation will end up using the last ckpt and a dot appears on the mAP and AR plots.

7. Visualization the training loss/step size and evalulating model – mAP,AR , tensorboard can be used

- cd /home/workspace
- python -m tensorboard.main --logdir experiments/reference/

8. For exporting the model.

- cd /home/workspace
- python experiments/exporter_main_v2.py --input_type image_tensor --pipeline_config_path experiments/reference/pipeline_new.config --trained_checkpoint_dir experiments/reference/ --output_directory experiments/reference/exported/

9. For testing the saved model.

- cd /home/workspace
- python inference_video.py --labelmap_path label_map.pbtxt --model_path experiments/reference/exported/saved_model --tf_record_path data/test/segment-12200383401366682847_2552_140_2572_140_with_camera_labels.tfrecord --config_path experiments/reference/pipeline_new.config --output_path animation.gif

3. Dataset:

The dataset for training the ssd_resnet50_v1_fpn_keras model contains camera data that contains road scenario -vehicle , pedestrian, bicycle. This camera dataset was taken from waymo open dataset.

- **Dataset Analysis:** For training the SSD model, 100 samples of camera data were taken from waymo open dataset that contained a good mix of zero moving objects -only road, only 1-2 moving objects , highway and urban scenario (too many objects) and also the dataset contained sample dataset contains a good mixture of data taken in daylight and night.
- **Cross-validation:** The downloaded sample camera data from waymo open dataset was divide into
  1. Training Dataset: 86 downloaded camera data samples were used to train the model.
  2. Validation/evaluation Dataset: 10 downloaded v samples were used for validation of the model. These dataset can be reused for evaluation after tuning model hyperparameters. The evaluation becomes more biased as skill on the validation/_evaluation dataset is incorporated into the model configuration. It becomes necessary to have one more dataset so that when the model is said to be performing well we can use test dataset
  3. Test Dataset: 3 downloaded camera data were used to test a final model which will be used to create an animation video at the last stage of the project.

4. Training:

Reference experiment: In the reference experiment, the pretrained model available in website: http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz was downloaded. The path to ssd_resnet50_v1_fpn_keras images for training and evaluating the model was added to model config file. Some of the highlight of the reference model config file

- batch_size=2

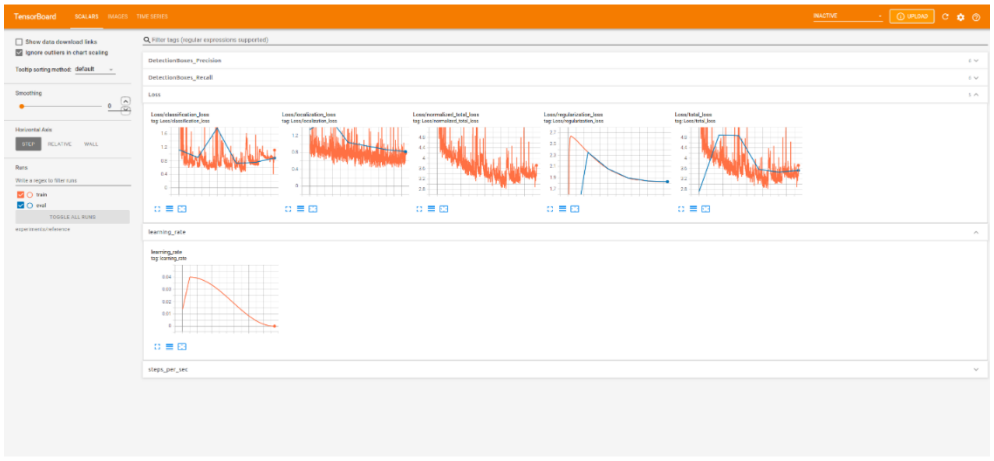
- epoch=2500

- momentum_optimizer

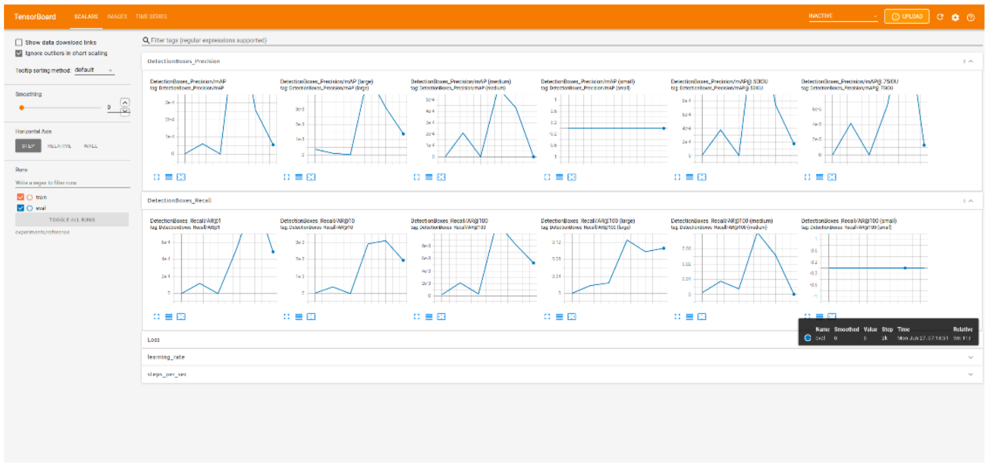
- cosine_decay_learning_rate with initial learning rate as 0.04

- 200 warm-up steps

Below are the tensorboard charts for the training and evaluation of the SSD model. The total loss of the model during the training was ~4 which is should be closer to zero



The evaluation of model gives mean Average Precision(mAP) and Average recall (AR) which are low(~0). However for AR for Large object was some non-zero value which is again considerably low, it only means that the model was picking up large objects correctly occasionally.

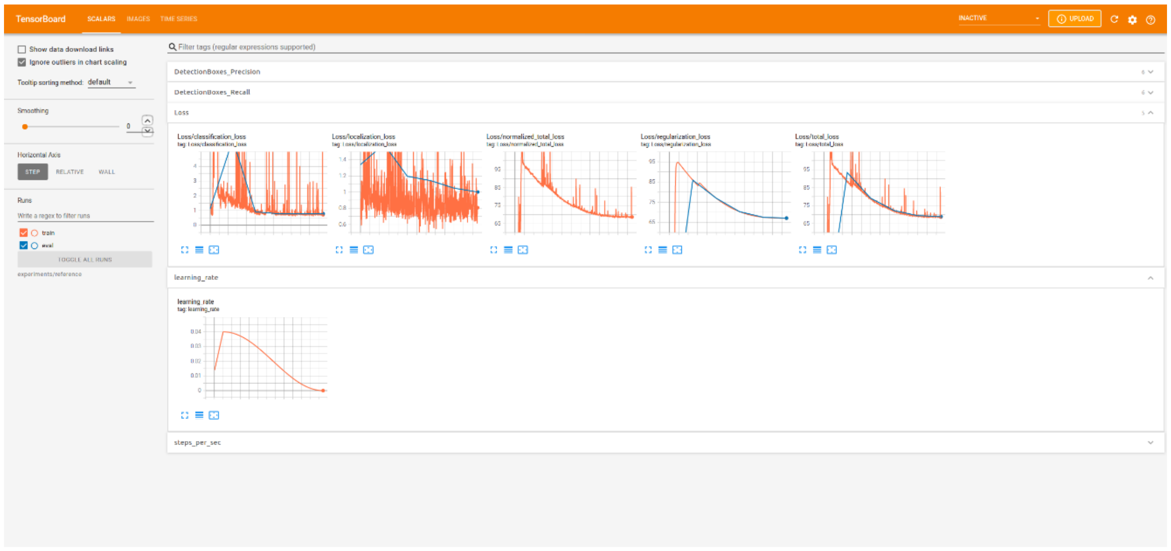

TensorBoard
SCALARS
IMAGES
TIME SERIES
INACTIVE
UPLOAD
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
Horizontal Axis
STEP
RELATIVE
WALL
Runs
train
eval
TOGGLE ALL RUNS
DetectionBoxes_Precision
DetectionBoxes_Precision/mAP
DetectionBoxes_Precision/mAP (large)
DetectionBoxes_Precision/mAP (medium)
DetectionBoxes_Precision/mAP (small)
DetectionBoxes_Precision/mAP@.50IOU
DetectionBoxes_Precision/mAP@.75IOU
DetectionBoxes_Recall
DetectionBoxes_Recall/AR@1
DetectionBoxes_Recall/AR@10
DetectionBoxes_Recall/AR@100
DetectionBoxes_Recall/AR@100 (large)
DetectionBoxes_Recall/AR@100 (medium)
DetectionBoxes_Recall/AR@100 (small)
Loss
learning_rate
steps_per_sec

For successful training of the model the mAP and AR should be closer to 1. The model need to be improved by using some data augmentation or fine tuning other hyperparameter of the model config.

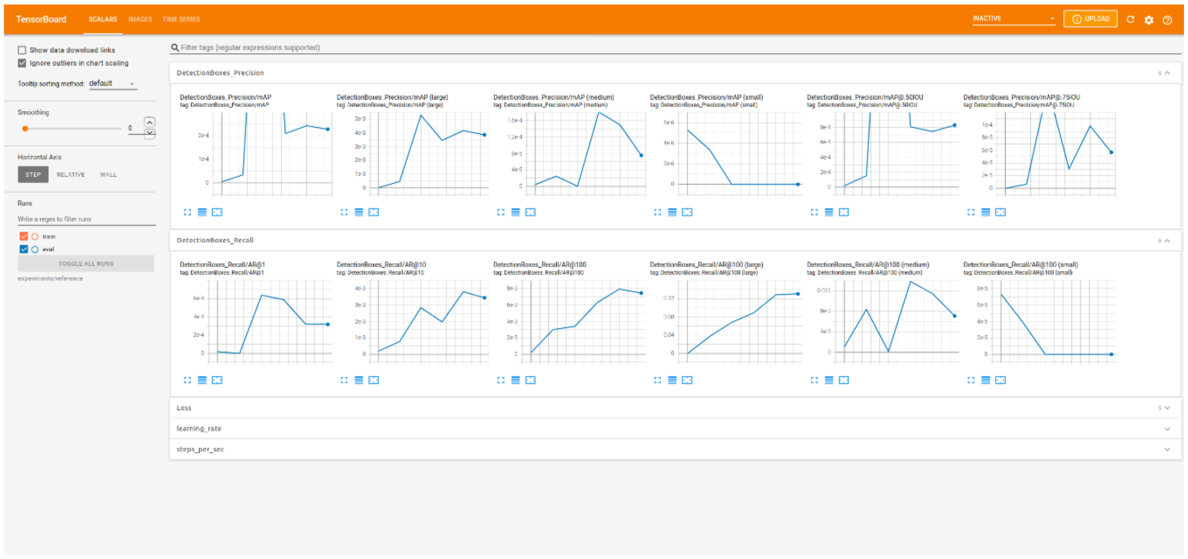
Experiment1:

In this experiment for training the SSD model , data augmentation of Random RGB to GRAY was added and below are the tensorboard charts

The training loss was pretty high ~75. Higher loss implies lower ability of model to predict the objects correctly.



The mAP and AR are very low. Only random RGB to Gray data augmentation did not add any improvement to the model.


TensorBoard
SCALARS
IMAGES
TIME SERIES
INACTIVE
UPLOAD
DetectionBoxes_Precision
DetectionBoxes_Precision/mAP
DetectionBoxes_Precision/mAP (large)
DetectionBoxes_Precision/mAP (medium)
DetectionBoxes_Precision/mAP (small)
DetectionBoxes_Precision/mAP@.50IOU
DetectionBoxes_Precision/mAP@.75IOU
DetectionBoxes_Recall
DetectionBoxes_Recall/AR@1
DetectionBoxes_Recall/AR@10
DetectionBoxes_Recall/AR@100
DetectionBoxes_Recall/AR@100 (large)
DetectionBoxes_Recall/AR@100 (medium)
DetectionBoxes_Recall/AR@100 (small)
Loss
learning_rate
steps_per_sec

Experiment2:

In this experiment for training the SSD model, there was zero warm up steps and initial learning rate was set to 0.001 and the training for model was run for 2500 steps with batch size of 2. Also there was few additional data-augmentation added to the training data sample for model performance improvement.

random_rgb_to_gray

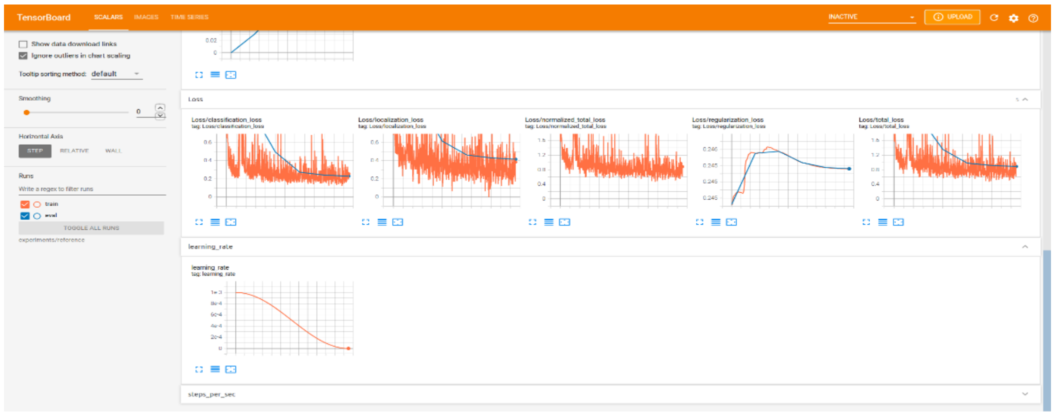
random_adjust_brightness

random_adjust_contrast

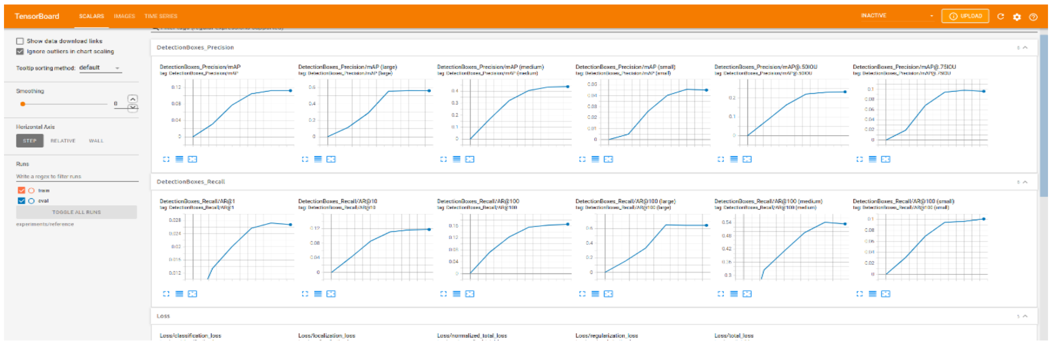
random_patch_gaussian

Note: explore augmentation.ipynb was used for visualizating data augmentation and then used in model

Below are the tensorboard charts for the training and evaluation of the SSD model after addition of changes mentioned above to model config file. The training loss and validation loss follows each other – that indicates that there was no overfitting of model, and has the losses are lesser than 1.



The mAP values for trained model performs better in detecing objects that are large object(~60%) and medium object(~50%) in the eval dataset but performs poor for smaller object. Similar behavior can be seen in the AR chart of tensorboard.


TensorBoard
SCALARS
IMAGES
TIME SERIES
DetectionBoxes_Precision
DetectionBoxes_Precision/mAP
DetectionBoxes_Precision/mAP (large)
DetectionBoxes_Precision/mAP (medium)
DetectionBoxes_Precision/mAP (small)
DetectionBoxes_Precision/mAP@.50IOU
DetectionBoxes_Precision/mAP@.75IOU
DetectionBoxes_Recall
DetectionBoxes_Recall/AR@1
DetectionBoxes_Recall/AR@10
DetectionBoxes_Recall/AR@100
DetectionBoxes_Recall/AR@100 (large)
DetectionBoxes_Recall/AR@100 (medium)
DetectionBoxes_Recall/AR@100 (small)
Loss
Loss/classification_loss
Loss/localization_loss
Loss/normalized_total_loss
Loss/regularization_loss
Loss/total_loss

5. Inference video snippet: