Object Detection in an Urban Environment

1. *Project overview*

The main objective is to implement detection of object in an urban environment. The dataset used here is the Waymo Open Dataset. Tensor board is used for displaying the training and evaluation results. Parameters tuning has also been done. The main steps along with implementation details are mentioned below.

2. *Setup*

Step 1 - Exploratory Data Analysis (EDA)

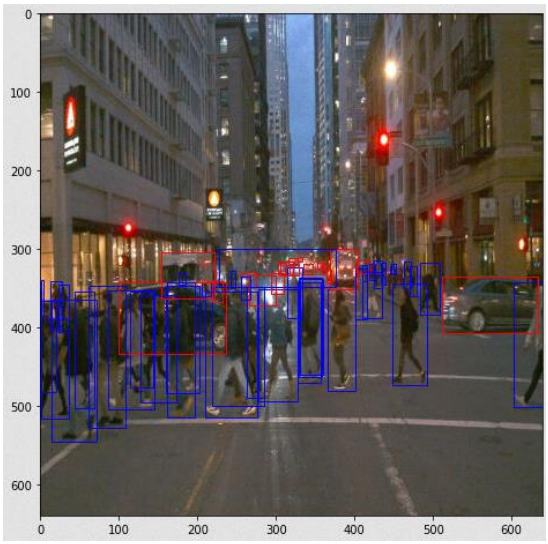
File: Explore data analysis.ipynb

Several steps are performed in EDA on the processes Waymo dataset.

Major steps include

a. Importing libraries for plotting
b. Writing the directory of the source dataset
c. Writing a function to display image and the bounding boxes. For reference, following two images are mentioned here. (Fig 1 and Fig 2)
   (Red bounding box: Vehicles; Blue bounding box: Pedestrian; Green bounding box: Green)
d. Addition EDA is performed to show Object count with respect to Class ID (Fig. 3)
   (Class 1: Vehicles; Class 2: Pedestrian; Class 4: Bicyclist)
   It is observed that the dataset is skewed, i.e. there is an imbalance of classes.

Dataset analysis: The data is divided in three folders namely Train, Val and test. The data is present in TF record format. Different types of images with objects, different weather conditions, dark, bright, different sizes are present.



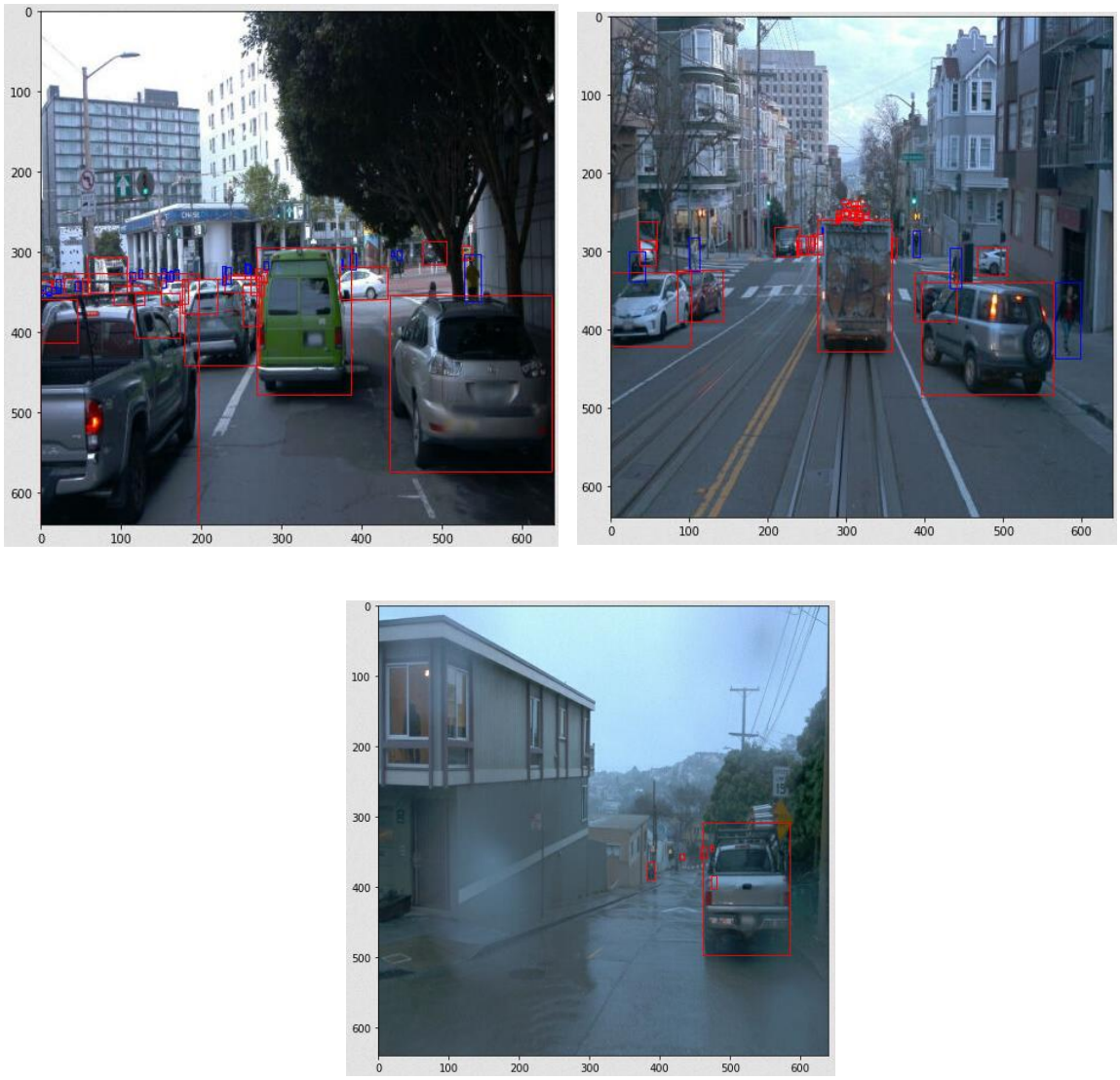


Fig. 1 Images to show the data set (environment, weather and classes distribution)

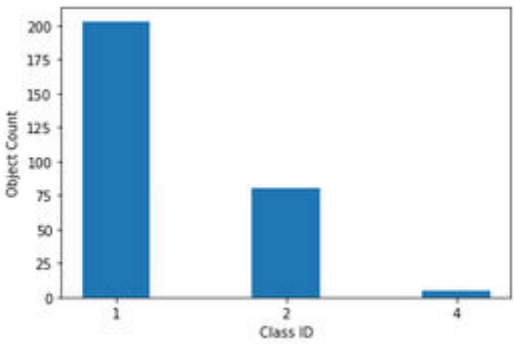


Fig. 2

Step 2 - Edit the config file

File: new_pipeline.config

Before the training, we need to change the config file to change the location of the training and validation files, as well as the location of the label_map file and pretrained weights. A new config file called pipeline_new.config is created and following modifications are done

a. Addition of data augmentation options (brightness, contrast, hue, saturation and color)
b. Changing learning rate base (Old: 0.04, New: 0.0005)
c. Changing Total steps (Old: 2500, New: 4500)
d. Changing warm up learning rate (Old: 0.01, New: 0.0005)
e. Changing warm up steps (Old: 200, New: 300)

Step 3 - Model Training and Evaluation

The training process was done and Tensor board was used to visualize the training and evaluation graphs. The training and evaluation graphs are mentioned below

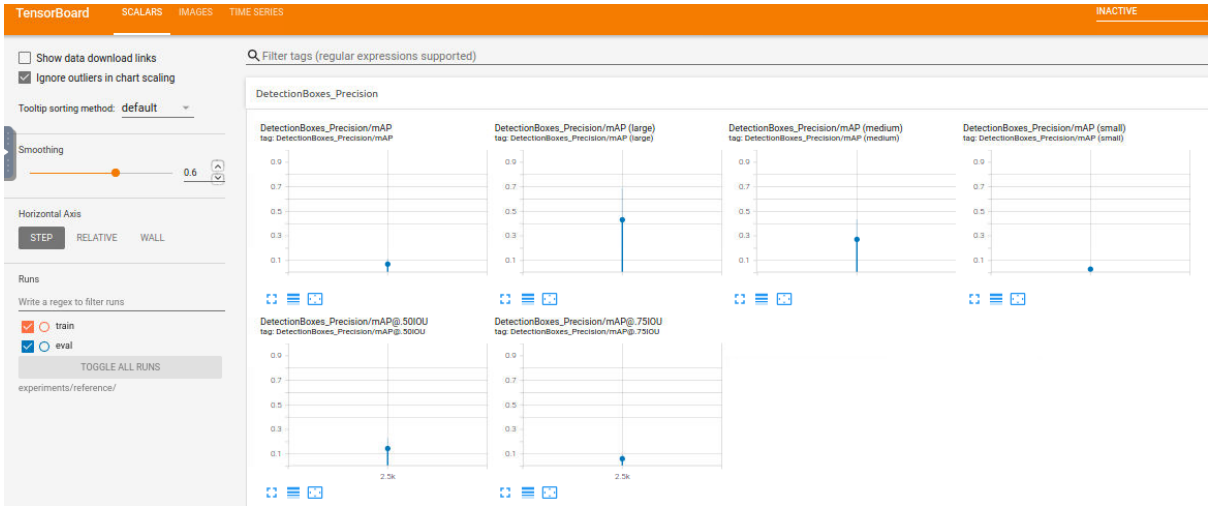


Fig. 4

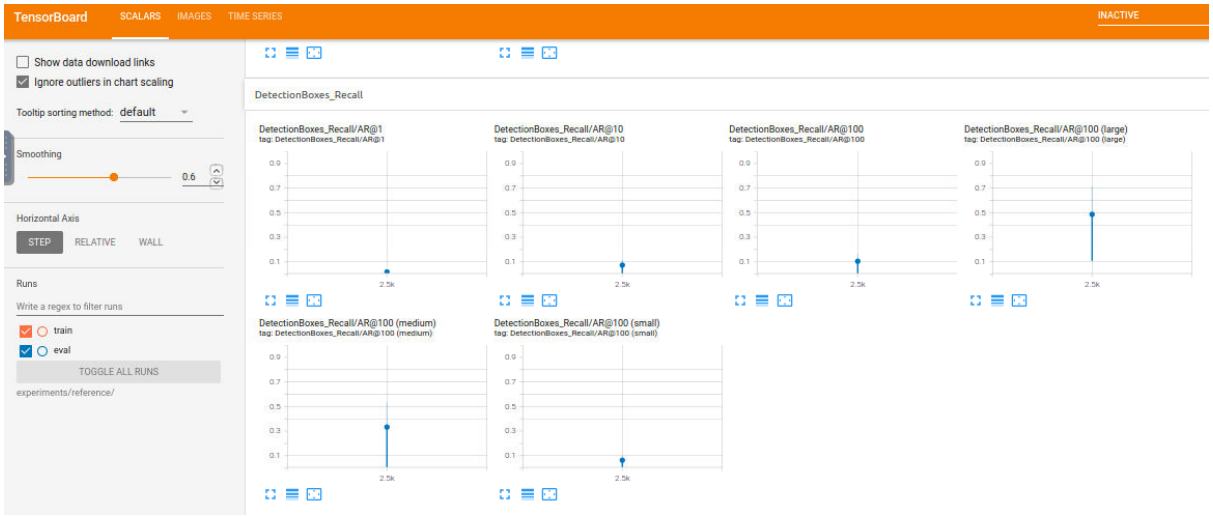

TensorBoard
SCALARS
IMAGES
TIME SERIES
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
eval
TOGGLE ALL RUNS
experiments/reference/
DetectionBoxes_Recall
DetectionBoxes_Recall/AR@1
DetectionBoxes_Recall/AR@10
DetectionBoxes_Recall/AR@100
DetectionBoxes_Recall/AR@100 (large)
DetectionBoxes_Recall/AR@100 (medium)
DetectionBoxes_Recall/AR@100 (small)

Fig. 5

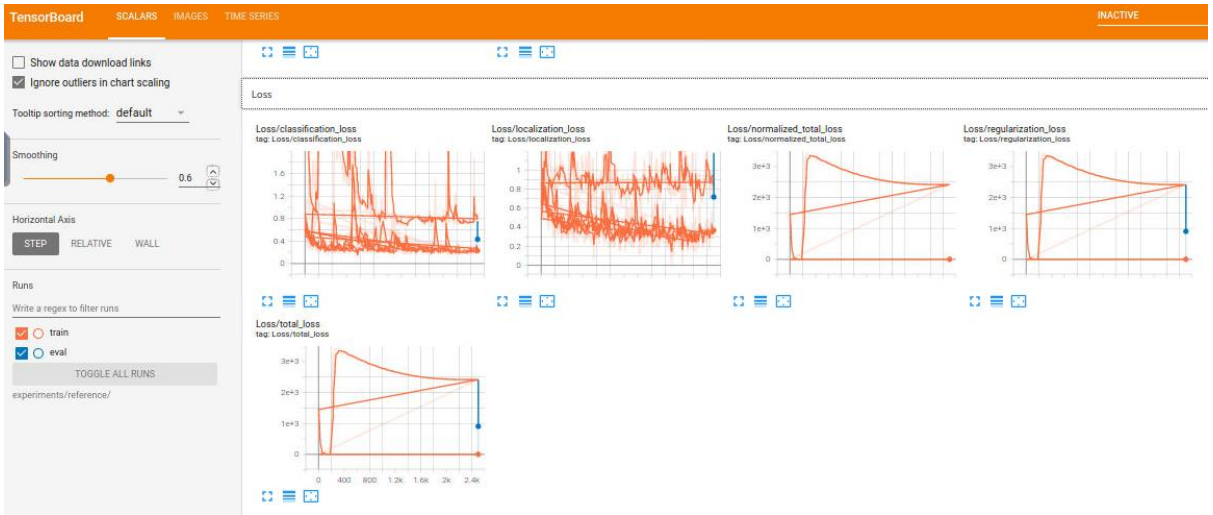

TensorBoard
SCALARS
IMAGES
TIME SERIES
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
eval
TOGGLE ALL RUNS
experiments/reference/
Loss
Loss/classification_loss
Loss/localization_loss
Loss/normalized_total_loss
Loss/regularization_loss
Loss/total_loss

Fig. 6

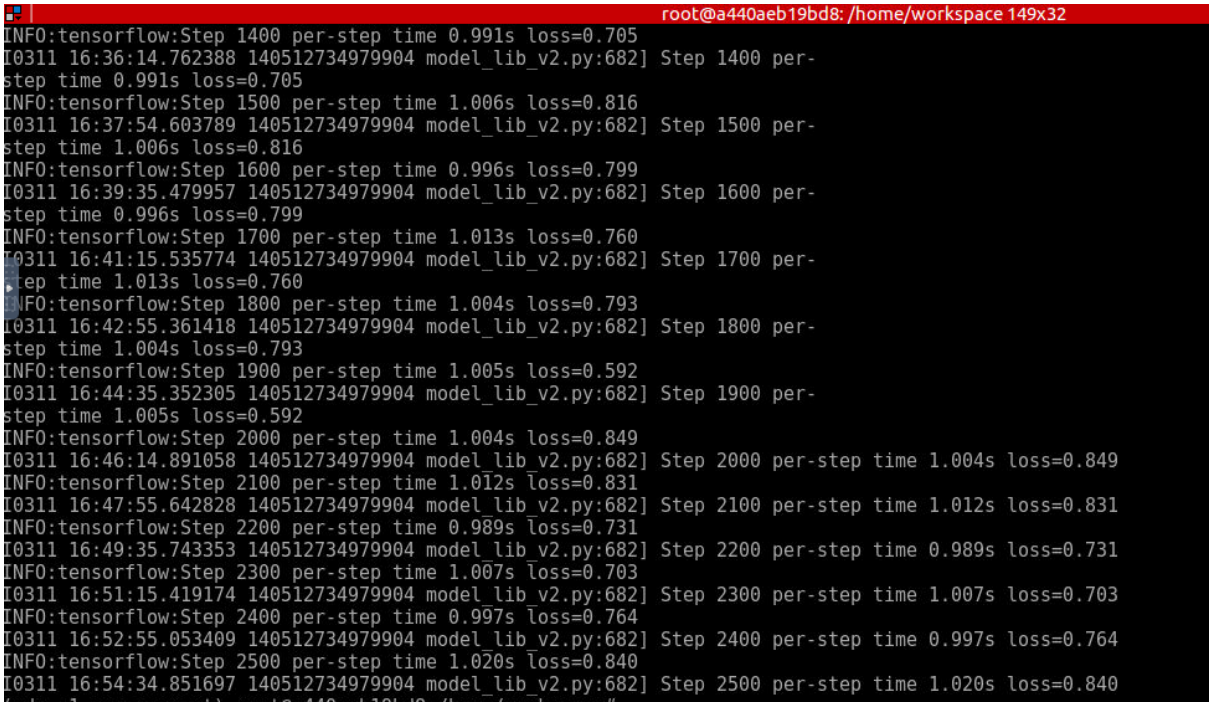


Fig. 7 Training

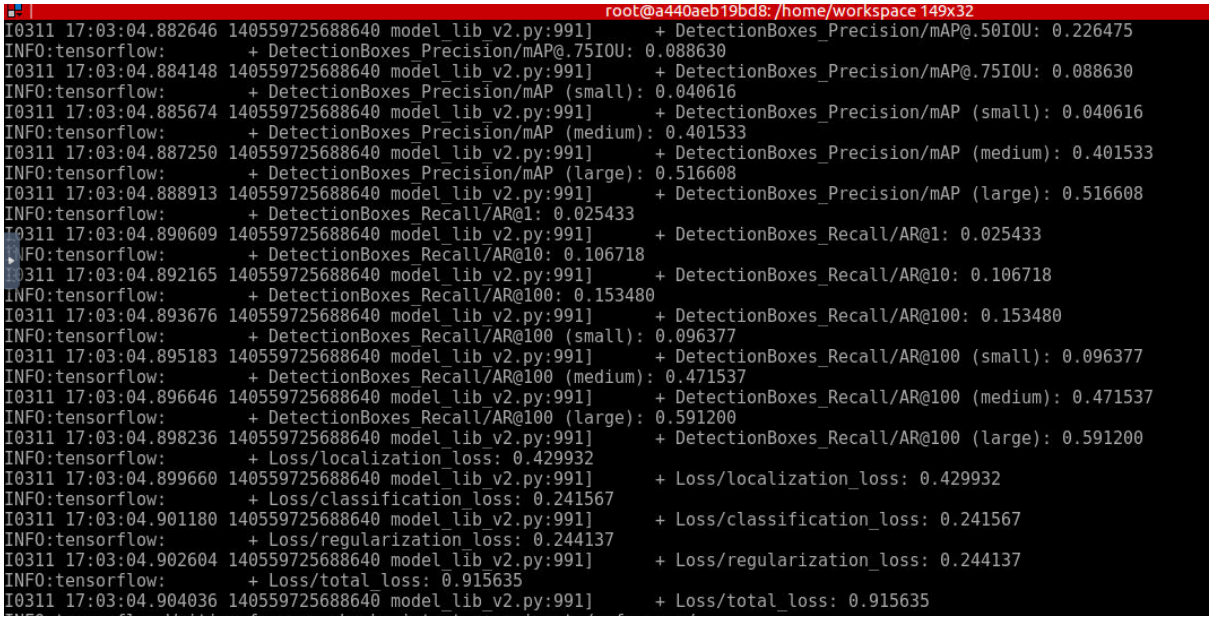


Fig. 8 Evaluation

Step 4 - Improve the performances

File: Augmentations.ipynb

- To improve the performances data augmentation is implemented as mentioned in Step 2
- Also, parameters in Explore augmentations.ipynb has been modified.
- Following augmentation techniques have been used
  a. Random crop image
  b. Random adjust contrast
  c. Random adjust saturation
  d. Random adjust hue
  e. Random adjust brightness
  f. Random distort color

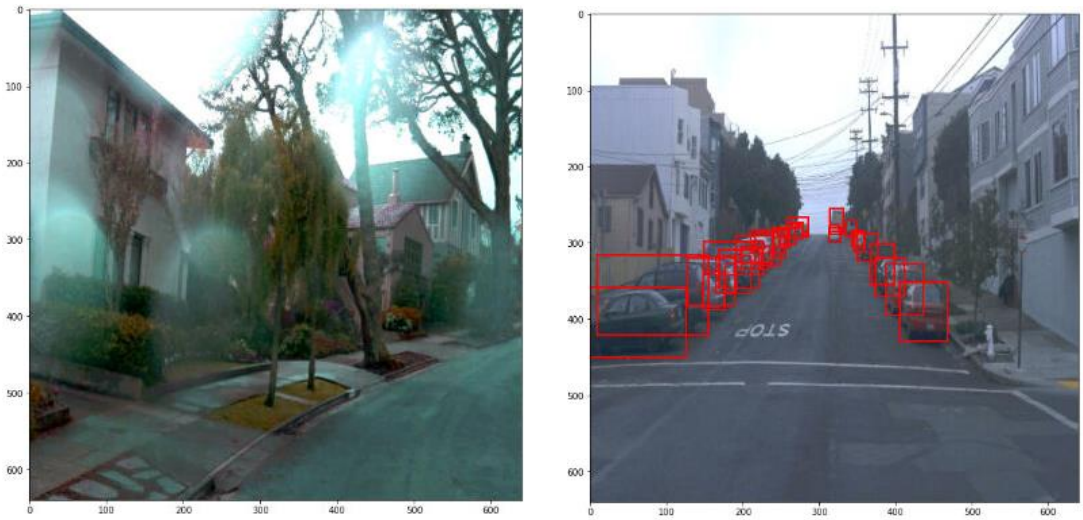


Fig. 9 After augmentation images