HOMEWORK 6

CSCI 677

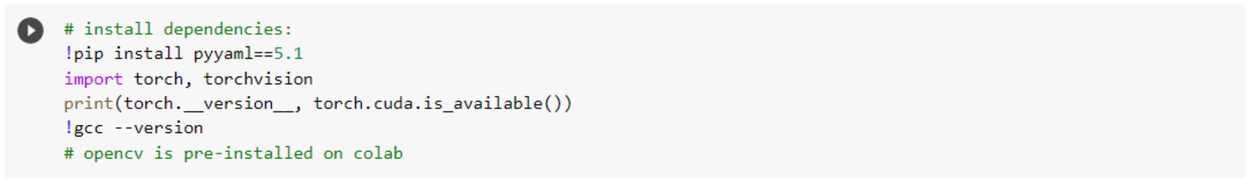
CHARU SINGH

Detectron2:

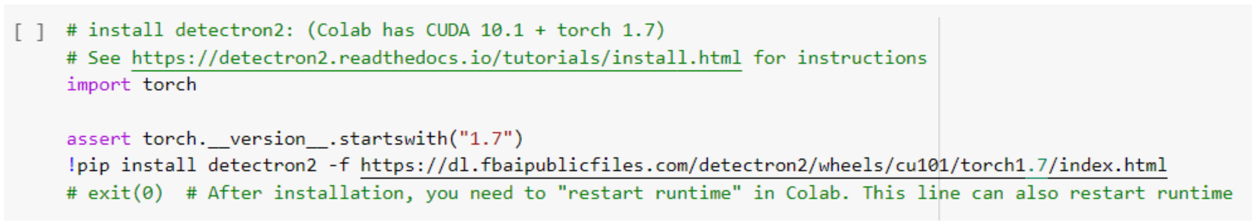
In this assignment we are using this software for object detection. All the models which are there in model zoo of the detectron 2 library are pre trained on coco dataset. Our goal is to fine tune our dataset on the pre trained model.

SUMMARY OF THE CODE:

1. **Installing Detectron 2:**
   Install dependencies and check whether CUDA is available. It helps in keeping track of the currently selected GPU and then install Detectron 2.

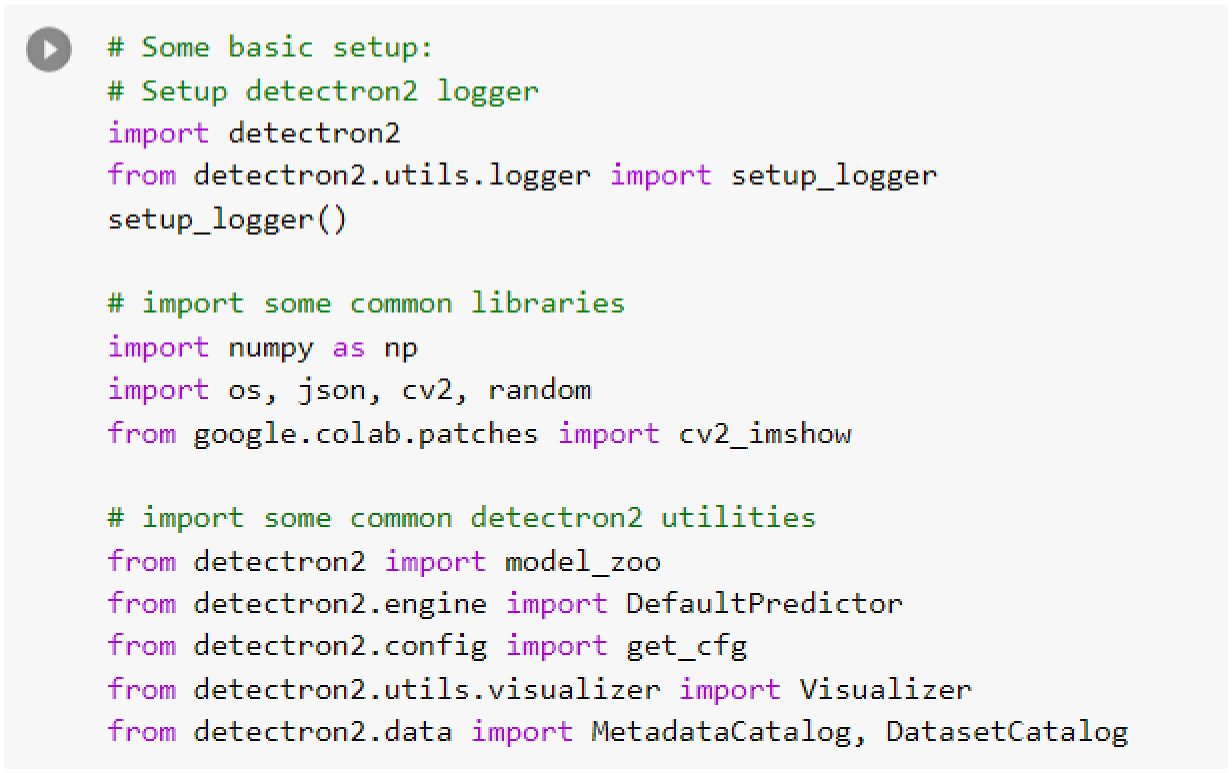


```
# install dependencies:
!pip install pyyaml==5.1
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
# opencv is pre-installed on colab
```



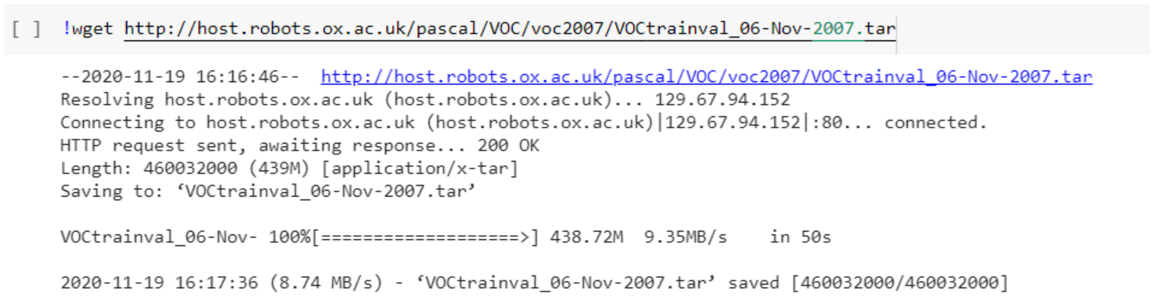
```
[ ] # install detectron2: (Colab has CUDA 10.1 + torch 1.7)
    # See https://detectron2.readthedocs.io/tutorials/install.html for instructions
    import torch

    assert torch.__version__.startswith("1.7")
    !pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.7/index.html
    # exit(0)  # After installation, you need to "restart runtime" in Colab. This line can also restart runtime
```

2. **Prepare and Register Dataset:**

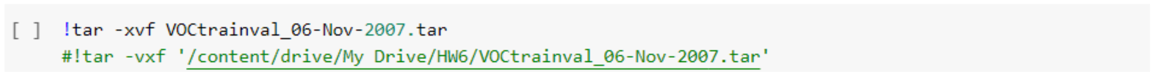


```
# Some basic setup:
# Setup detectron2 logger
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# import some common libraries
import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow

# import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
```

3. **Run pretrained Detectron 2 model:**



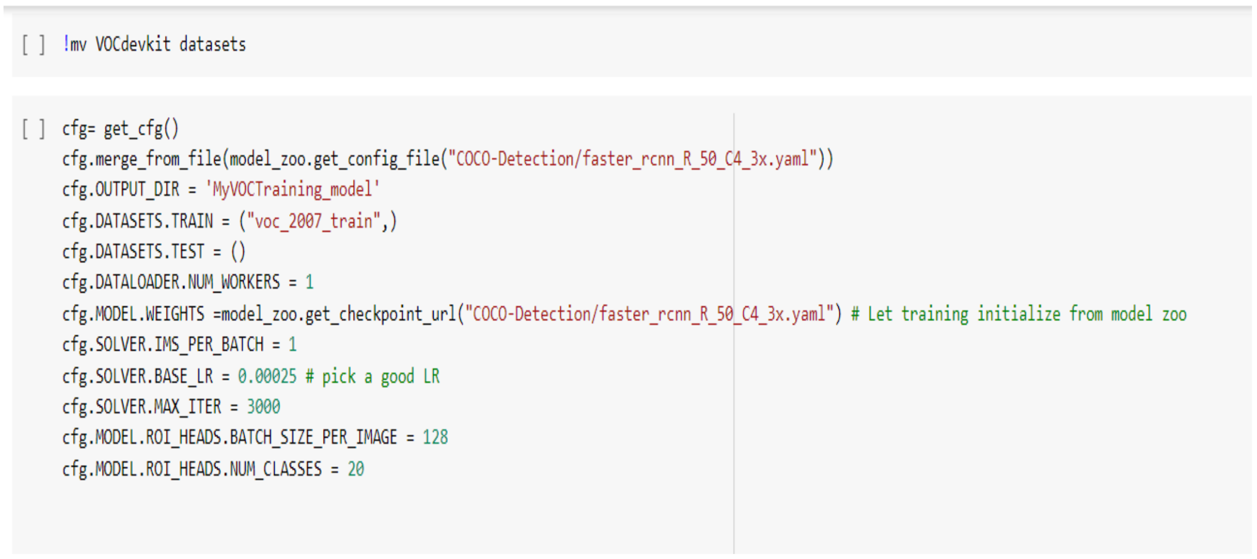
```
[ ] !wget http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar

    --2020-11-19 16:16:46--  http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar
    Resolving host.robots.ox.ac.uk (host.robots.ox.ac.uk)... 129.67.94.152
    Connecting to host.robots.ox.ac.uk (host.robots.ox.ac.uk)|129.67.94.152|:80... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 460032000 (439M) [application/x-tar]
    Saving to: ‘VOCtrainval_06-Nov-2007.tar’

    VOCtrainval_06-Nov- 100%[===================>] 438.72M  9.35MB/s    in 50s

    2020-11-19 16:17:36 (8.74 MB/s) - ‘VOCtrainval_06-Nov-2007.tar’ saved [460032000/460032000]
```



```
[ ] !tar -xvf VOCtrainval_06-Nov-2007.tar
    #!tar -xvf '/content/drive/My Drive/HW6/VOCtrainval_06-Nov-2007.tar'
```

4. **Create detectron 2 config and Detectron 2 *DefaultPredictor* to run inference on the images.**
   -Choose the model Faster RCNN from model zoo:

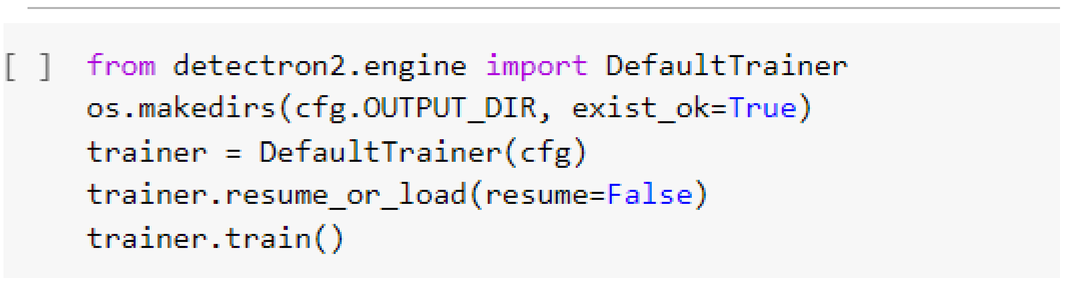
-Choose another model Faster RCNN without FPN from model zoo:



```
[ ] !mv VOCdevkit datasets
```

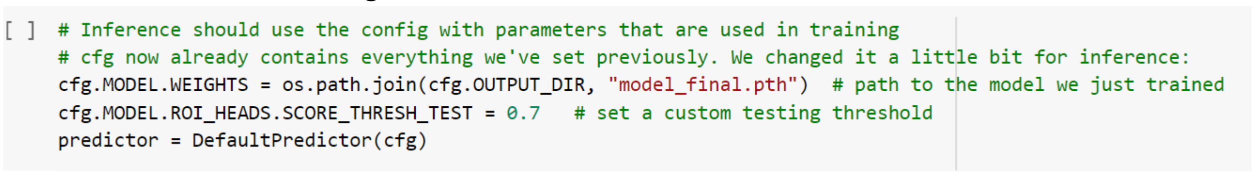
```
[ ] cfg= get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_C4_3x.yaml"))
    cfg.OUTPUT_DIR = 'MyVOCTraining_model'
    cfg.DATASETS.TRAIN = ("voc_2007_train",)
    cfg.DATASETS.TEST = ()
    cfg.DATALOADER.NUM_WORKERS = 1
    cfg.MODEL.WEIGHTS =model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_C4_3x.yaml") # Let training initialize from model zoo
    cfg.SOLVER.IMS_PER_BATCH = 1
    cfg.SOLVER.BASE_LR = 0.00025 # pick a good LR
    cfg.SOLVER.MAX_ITER = 3000
    cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 20
```

5. Train on custom dataset (PASCAL VOC dataset)



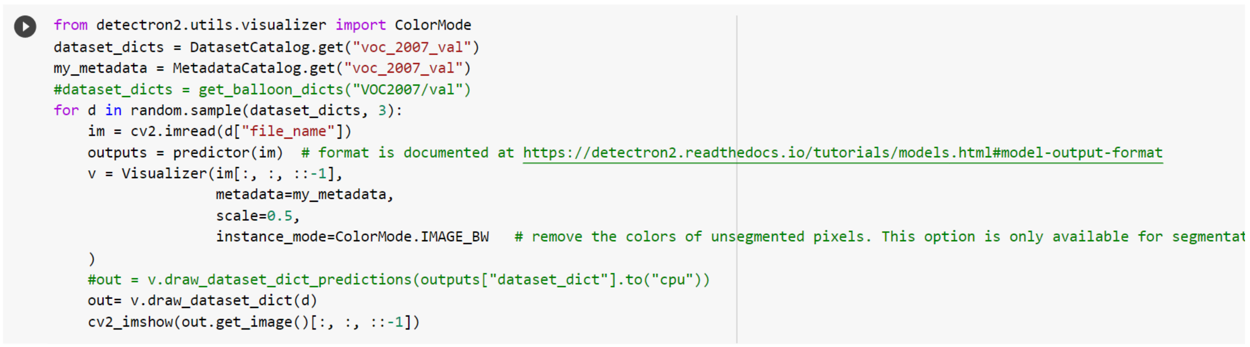
```
[ ] from detectron2.engine import DefaultTrainer
    os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
    trainer = DefaultTrainer(cfg)
    trainer.resume_or_load(resume=False)
    trainer.train()
```

6. Inference and Evaluation using the trained model:



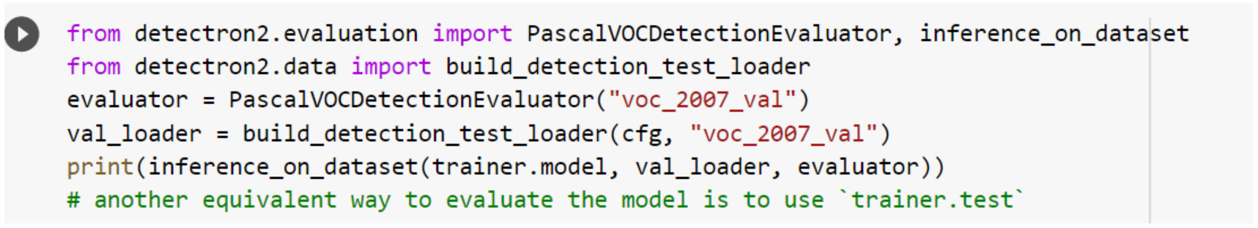
```
[ ] # Inference should use the config with parameters that are used in training
    # cfg now already contains everything we've set previously. We changed it a little bit for inference:
    cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")  # path to the model we just trained
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set a custom testing threshold
    predictor = DefaultPredictor(cfg)
```

7. Visualize some randomly chosen images:



```
from detectron2.utils.visualizer import ColorMode
dataset_dicts = DatasetCatalog.get("voc_2007_val")
my_metadata = MetadataCatalog.get("voc_2007_val")
#dataset_dicts = get_balloon_dicts("VOC2007/val")
for d in random.sample(dataset_dicts, 3):
    im = cv2.imread(d["file_name"])
    outputs = predictor(im)  # format is documented at https://detectron2.readthedocs.io/tutorials/models.html#model-output-format
    v = Visualizer(im[:, :, ::-1],
                   metadata=my_metadata,
                   scale=0.5,
                   instance_mode=ColorMode.IMAGE_BW   # remove the colors of unsegmented pixels. This option is only available for segmentat
    )
    #out = v.draw_dataset_dict_predictions(outputs["dataset_dict"].to("cpu"))
    out= v.draw_dataset_dict(d)
    cv2_imshow(out.get_image()[:, :, ::-1])
```

8. **Evaluate performance by finding out AP using Pascal VOC Detection Evaluator.**

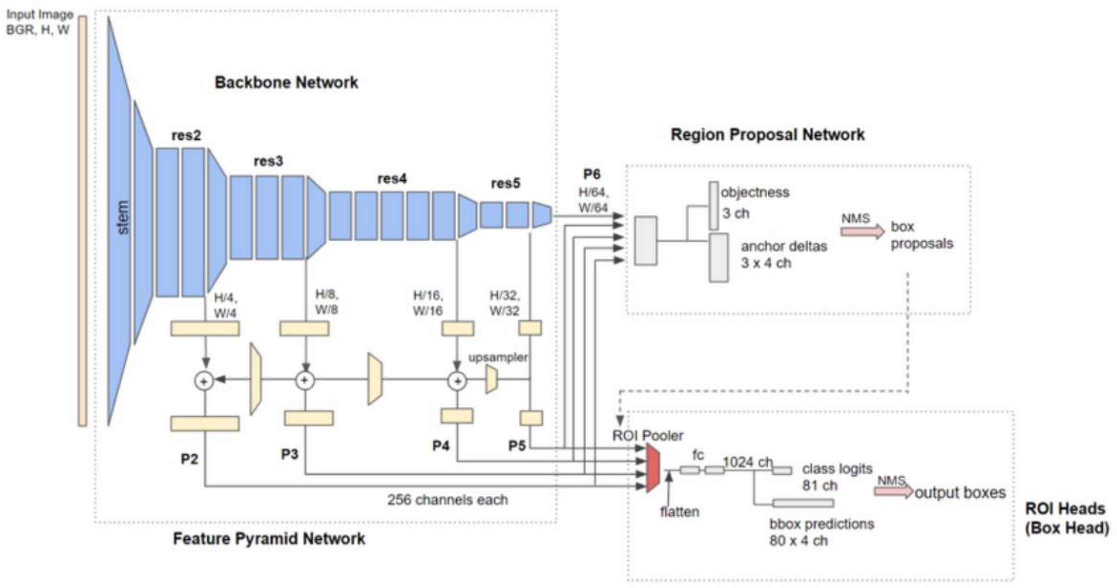


```
from detectron2.evaluation import PascalVOCDetectionEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader
evaluator = PascalVOCDetectionEvaluator("voc_2007_val")
val_loader = build_detection_test_loader(cfg, "voc_2007_val")
print(inference_on_dataset(trainer.model, val_loader, evaluator))
# another equivalent way to evaluate the model is to use `trainer.test`
```

MODEL USED:

1. **faster_rcnn_R_50_FPN_3x:**

This model has a FPN (Feature Pyramid Network) backbone which is a multiscale detector that gives higher accuracy for detecting objects of various sizes.



Here the backbone network is Resnet which will extract features from the image followed by Region Proposal Network and Box head for tightening the bounding box.

1. **Backbone Network**: extracts feature maps from the input image at different scales. Base-RCNN-FPN's output features are called P2 (1/4 scale), P3 (1/8), P4 (1/16), P5 (1/32) and P6 (1/64) whereas in non-FPN ('C4') architecture's output feature is only from the 1/16 scale.

2. **Region Proposal Network**: detects object regions from the multi-scale features. 1000 box proposals (by default) with confidence scores are obtained.

3. **Box Head**: crops and warps feature maps using proposal boxes into multiple fixed-size features and obtains fine-tuned box locations and classification results via fully connected layers. Finally, 100 boxes (by default) in maximum are filtered out using non-maximum suppression (NMS). The box head is one of the sub-classes of **ROI Heads**.

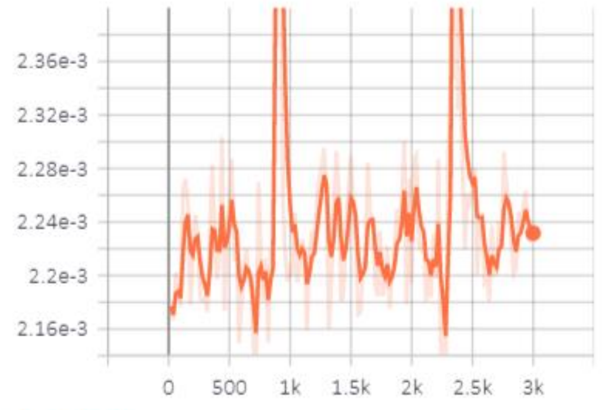
Results and Discussion:

Qualitative Results:

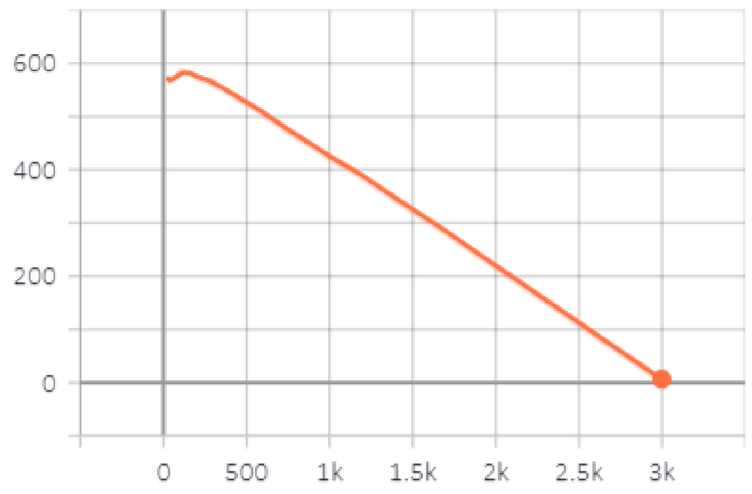
1. **Model 1:**
   Parameters:
   Default:

```
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 3000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
```

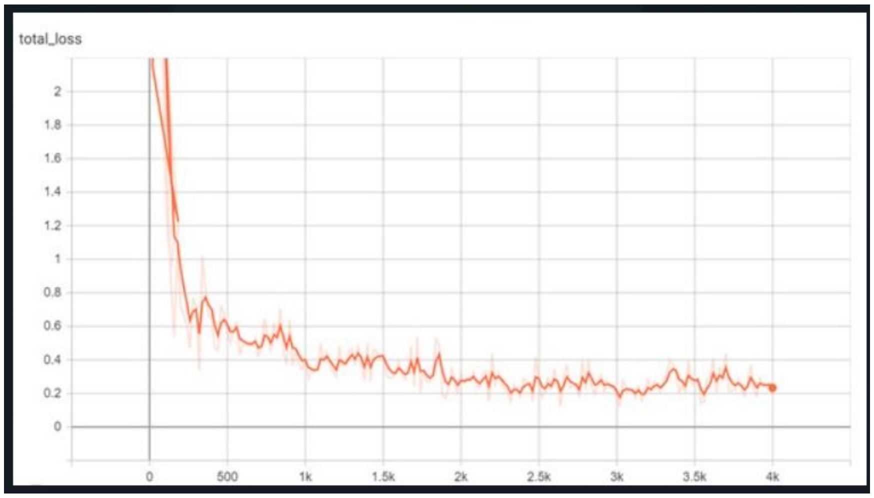
data_time

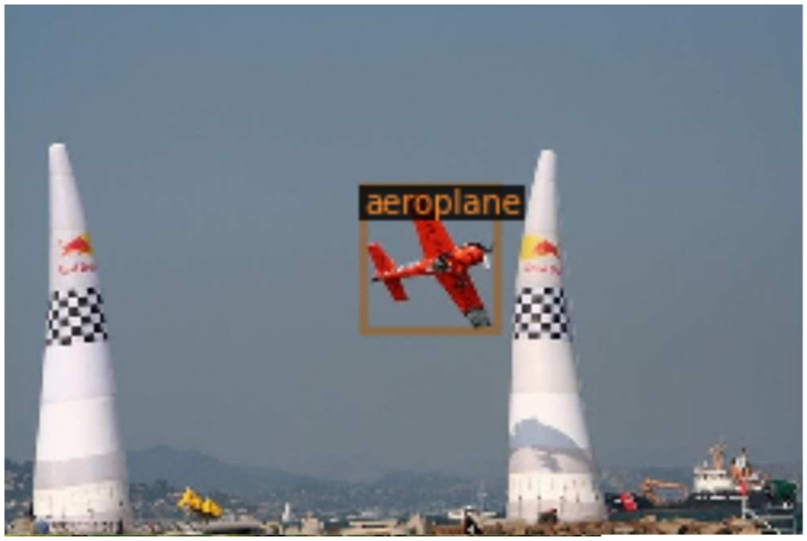


eta_seconds

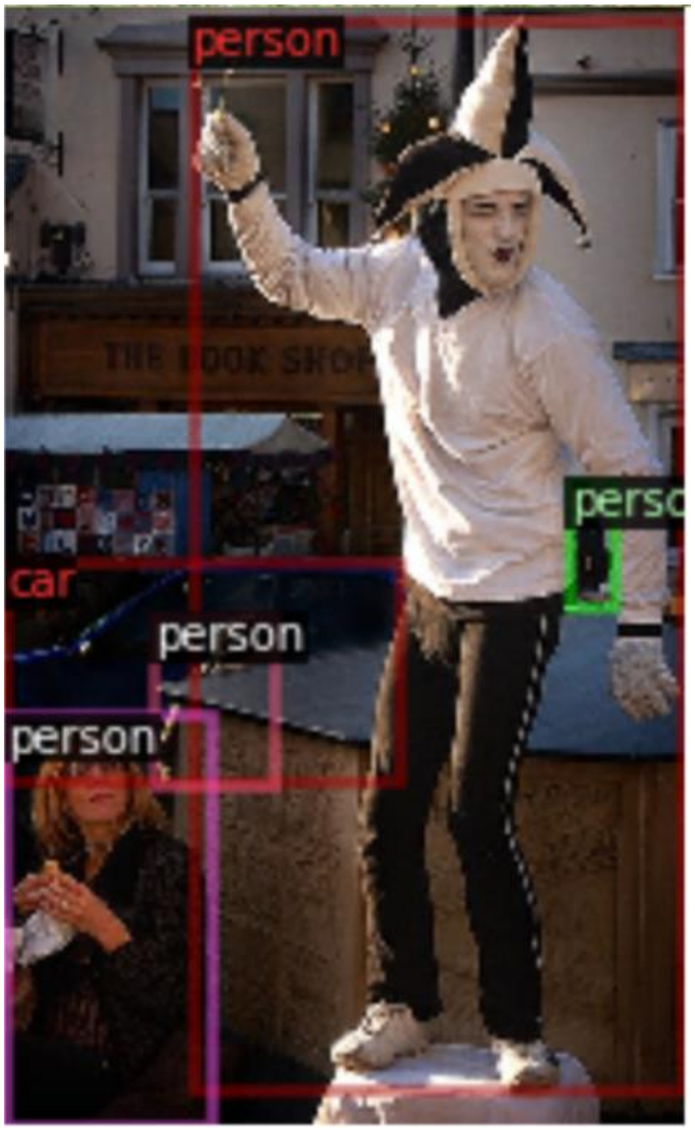


Loss function:





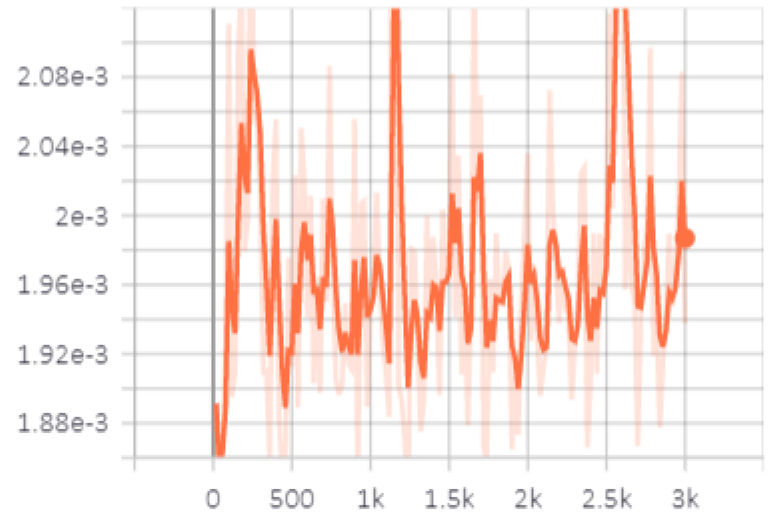
bicycle



2

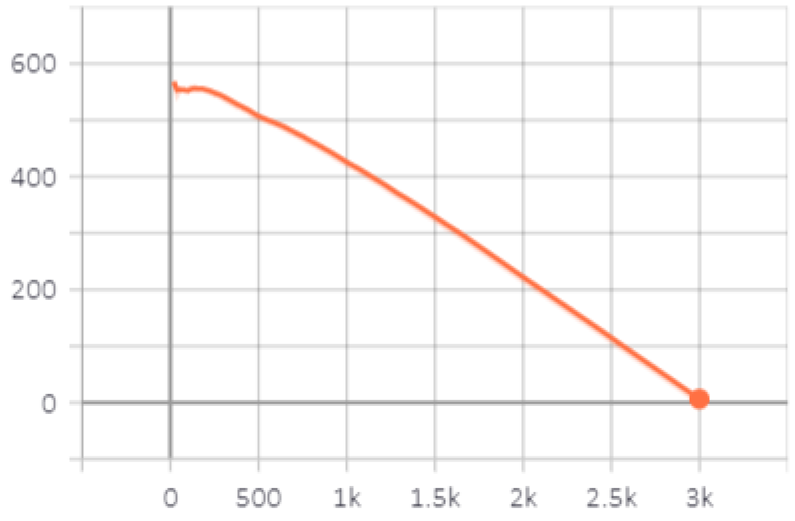
```
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 3000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE =64
```

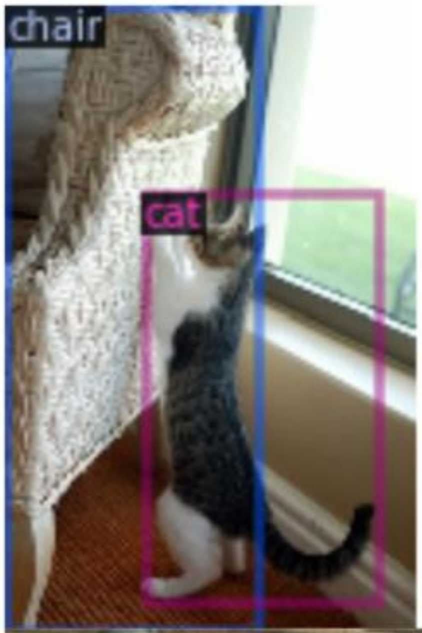
data_time

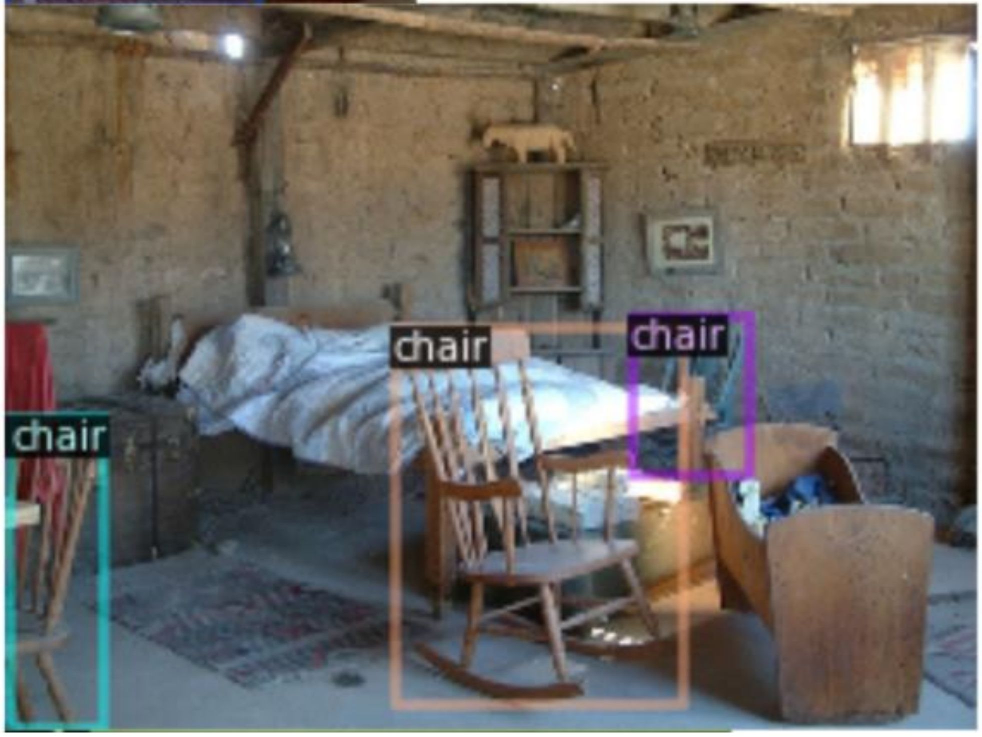


eta_seconds

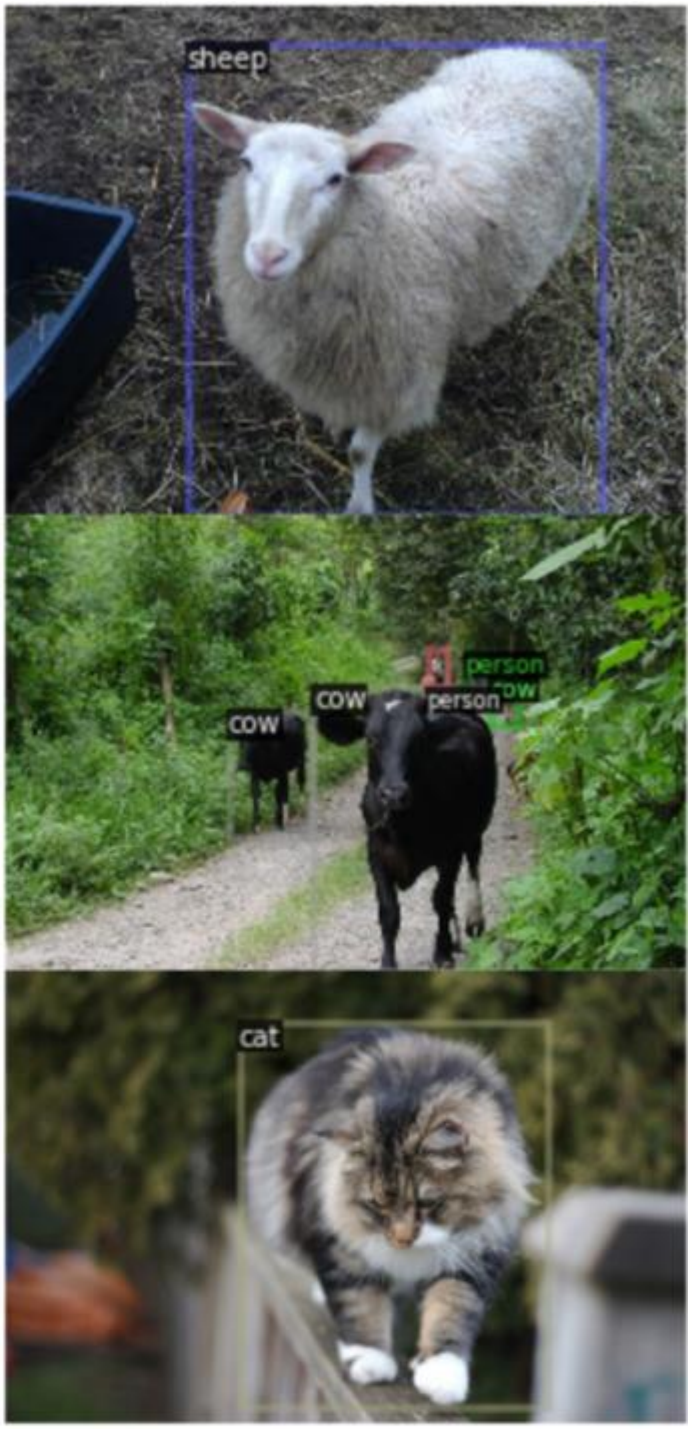
eta_seconds







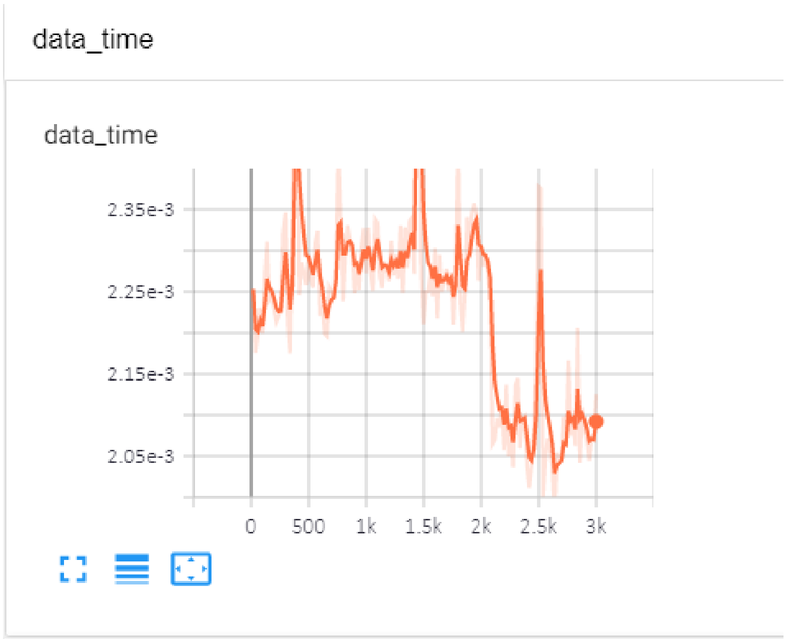
```
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 3000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE =512
```

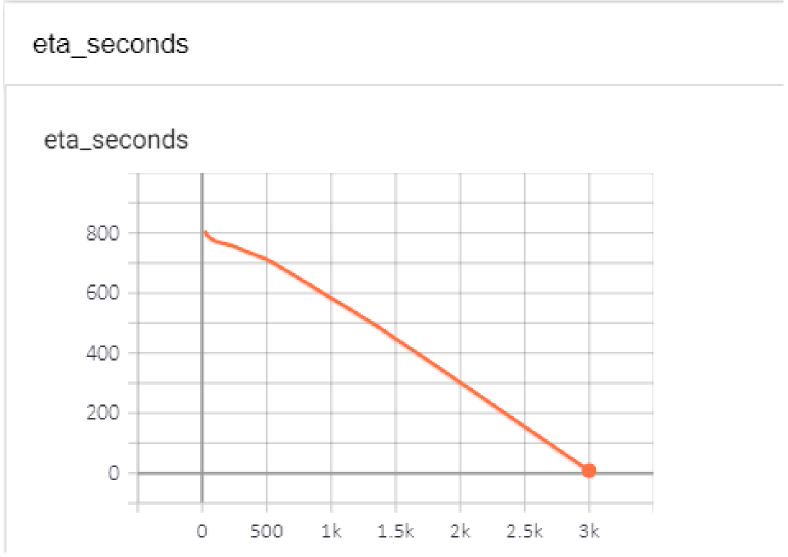


MODEL 2:

Parameters:
Default:

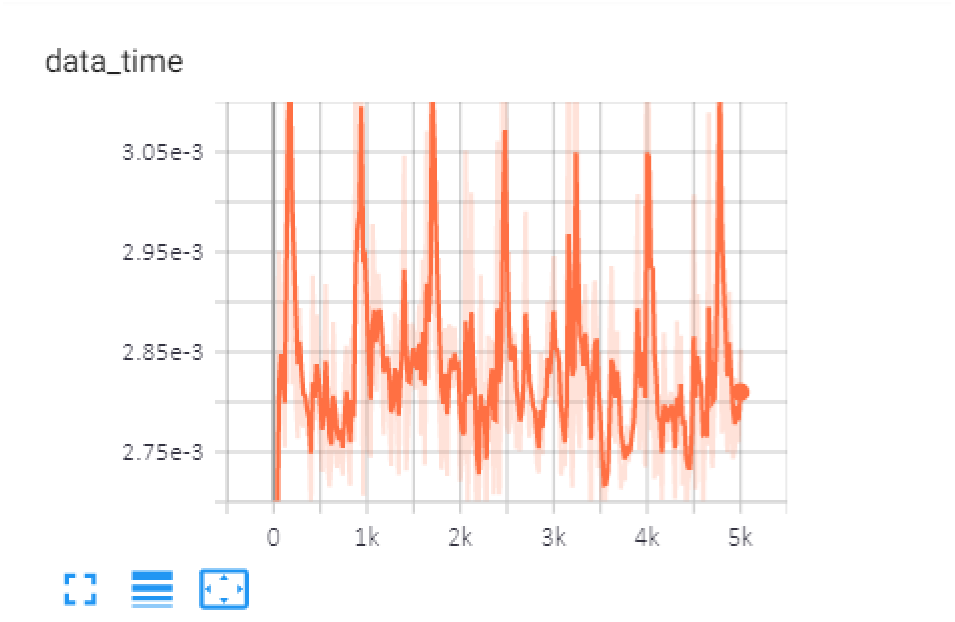
```
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 3000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
```



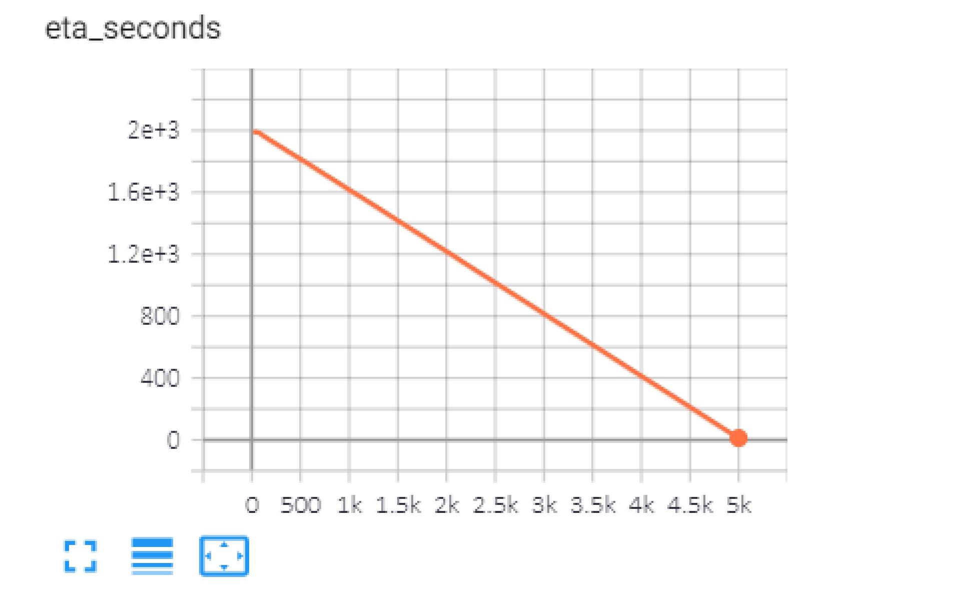


person
person
person
tvmonitor
person
person
bicycle

```
cfg.SOLVER.BASE_LR = 0.0001
cfg.SOLVER.MAX_ITER = 5000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
```



eta_seconds



PARAMETERS	**AP (training)**	**AP(validation)**

`cfg.SOLVER.BASE_LR = 0.00025`
`cfg.SOLVER.MAX_ITER = 3000`
`cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE =128`
	'AP50': 77.63984958966401	'AP50': 68.55142312360846

`cfg.SOLVER.BASE_LR = 0.00025`	'AP50': 76.8834361835558	'AP50': 64.1247

cfg.SOLVER.MAX_ITER = 3000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE =64		
cfg.SOLVER.BASE_LR = 0.00025 cfg.SOLVER.MAX_ITER = 4000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE =512	'AP50': 80.99124715777167	'AP50': 69.99124715777167
MODEL2 ***Faster RCNN R_50_C4***		
cfg.SOLVER.BASE_LR = 0.00025 cfg.SOLVER.MAX_ITER = 3000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE =128	'AP50': 78.65167319733654	'AP50': 67.8
cfg.SOLVER.BASE_LR = 0.0001 cfg.SOLVER.MAX_ITER = 5000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512	'AP50': 80.64260222944195	'AP50': 71.63484951771777
cfg.SOLVER.BASE_LR = 0.0001 cfg.SOLVER.MAX_ITER = 5000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64	'AP50': 77.66	'AP50': 75.64
cfg.SOLVER.BASE_LR = 0.0001 cfg.SOLVER.MAX_ITER = 5000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128	'AP50':80.33	'AP50': 77.38
cfg.SOLVER.BASE_LR = 0.0001 cfg.SOLVER.MAX_ITER = 4000 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 256	'AP50': 79.657	'AP50': 77.7

Inference:

Quantitative:

We fine tuned the above model and try to achieve best accuracy on the give dataset. I have tried various hyperparameters which are reported in the table above. I have tuned parameters such as Batch size, learning rate and number of iterations for both the models. Average Precision of 50% for every combination of hyper tuned parameters is shown above. I observed that increasing **number of iterations** is increasing the accuracy (I have tried 3000,4000 and 5000) and its possible that increasing number of iterations more than 5000 may increase the accuracy more. Decreasing **learning rate** is also increasing the accuracy as it finds local minima more accurately. The gradient step is less when we backpropagation as we decrease the learning rate.

By default, batch size is 128, if we decrease the batch size the accuracy is dropping and if we increase the batch size to 256 or 512, it is increasing the accuracy. However, increasing batch size from 256 to 512 is not increasing the accuracy by significant amount and batch size of 512 is taking more time than 256 so its better to go with batch size of 256.

If we compare the two models the accuracy achieved is almost same in both cases however the first model takes less amount of time than the second one so we would prefer the first one.

Qualitative:

We can see that in both the models the visuals are clear. Each of the bounding box is pretty much accurate and is identifying the objects correctly.

In model 2 the bounding box are more prominent.

REFERENCES:

1. https://research.fb.com/wp-content/uploads/2019/12/4.-detectron2.pdf
2. https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5?authuser=1#scrollTo=U5LhISJqWXgM