Shashank N S Homework 6 CSCI677 | Fall-2020

Objective: To fine tune the Faster RCNN object detection model using detectorn-2 software and evaluate the performance on PASCAL-VOC dataset.

1. **Introduction:** Regions with CNN features (RCNN) is a popular object detection technique proposed by Girishik et.al in the paper *Rich feature hierarchies for accurate for object detection and semantic segmentation.* The main objective of this network is to extend CNNs to detection tasks by the aid of externally generated proposals like selective search or edge box. RCNN takes each proposal, warps it to a fixed size and passes it to the CNN. The improvised version of RCNN called Fast RCNN removes the redundancy of recomputing the features for different proposals. Further, Faster RCNN uses the computed features to predict the proposals by adding a region proposal network. Figure 1 shows the block diagram of Faster-RCNN.

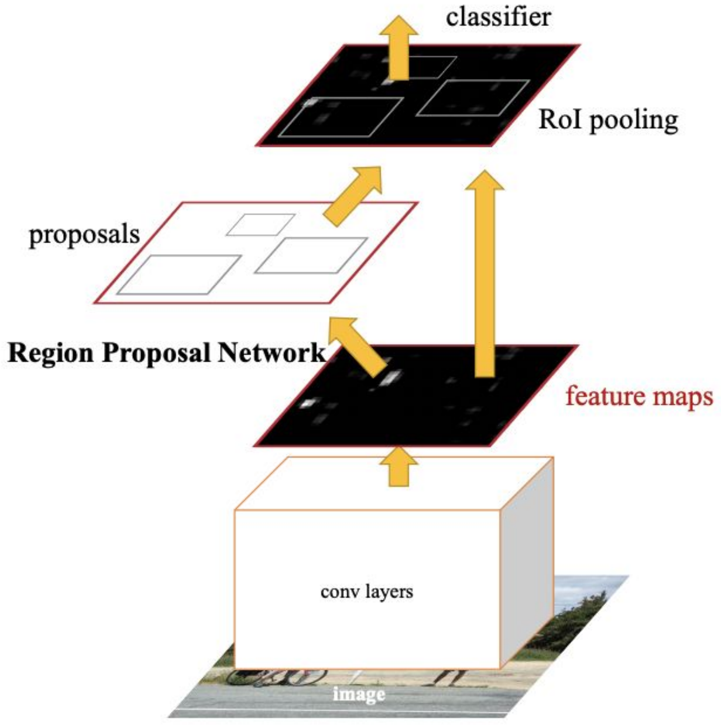


Figure 1: Faster RCNN network

2. Summary of the Program:

a. Training information:

i. **Network architecture:** We use detectron-2 software to implement Faster-RCNN with and without Feature Pyramid network (FPN).

ii. **Loss function:** Combined loss function for classification and box regression is used,

$$L(\{p_i\},\ \{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,\ p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,\ t_i^*)$$

iii. **Evaluation metric:** We use mean average precision at IoU 50%, 75% are used to quantitatively evaluate the network. Also, we show some qualitative results with success and failures.

iv. **Training parameters:** We use default detectron-2 training configuration with a learning rate of 0.00025, batch size 128, and the model is trained for 3000 iterations.

b. Program and dataset information:

The scope of this assignment is to build a pipeline to train Faster RCNN using existing APIs of detectron-2 on PASCAL-VOC dataset for the task of object detection. The code for dataset processing, training and result evaluation is organized in different cells of ***colab notebook (HW6-Object_detection.ipynb)***

c. Source listing:

i. **Training configuration:**

```
#Get the pretrained R-CNN model
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detec
tion/faster_rcnn_R_50_C4_3x.yaml"))

cfg.OUTPUT_DIR = 'MyVOCTraining'
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")

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

#Train the model
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

ii. **Visualize the results:**

```
from detectron2.data import MetadataCatalog
def parse_rec(filename):
   """Parse a PASCAL VOC xml file."""
   with PathManager.open(filename) as f:
       tree = ET.parse(f)
   objects = []
   for obj in tree.findall("object"):
       obj_struct = {}
       bbox = obj.find("bndbox")
       obj_struct["bbox"] = [
           int(bbox.find("xmin").text),
           int(bbox.find("ymin").text),
           int(bbox.find("xmax").text),
           int(bbox.find("ymax").text),
       ]
       objects.append(obj_struct)
       obj_struct["name"] = obj.find("name").text
       obj_struct["pose"] = obj.find("pose").text
                                  obj_struct["truncated"]     =
int(obj.find("truncated").text)
```

```
                              obj_struct["difficult"]    =
int(obj.find("difficult").text)
   return objects
im                                                       =
cv2.imread("./datasets/VOC2007/JPEGImages/000363.jpg")
cv2_imshow(im)
predictor = DefaultPredictor(cfg)
outputs = predictor(im)

v        =        Visualizer(im[:,        :,        ::-1],
MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
out                                                      =
v.draw_instance_predictions(outputs["instances"].to("cpu"
))
cv2_imshow(out.get_image()[:, :, ::-1])

obj                                                      =
parse_rec('./datasets/VOC2007/Annotations/000363.xml')
print(len(obj))
for i in range(0, len(obj)):
 b_box1 = obj[i]['bbox']
  xmin1,  ymin1,  xmax1,  ymax1  =  b_box1[0],  b_box1[1],
b_box1[2], b_box1[3]
cv2.rectangle(im,(xmin1,ymin1),(xmax1,ymax1),(0,255,0),2)
cv2_imshow(im)
```

iii. Performance evaluation:

```
#Quantitative performance evaluation.
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))
```

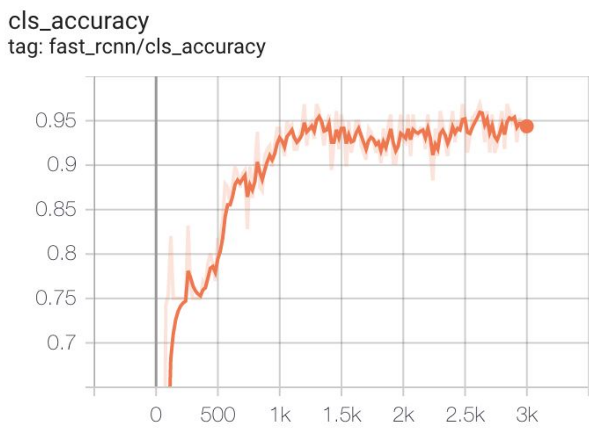
3. Experimental Results:

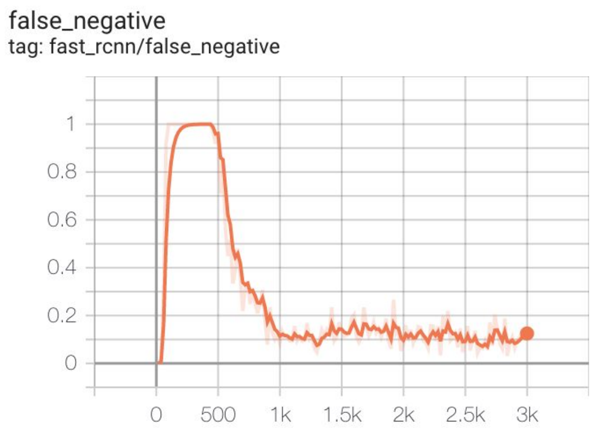
The Faster RCNN model is trained with two variations - one with Feature Pyramid Network (FPN) and the other without FPN.

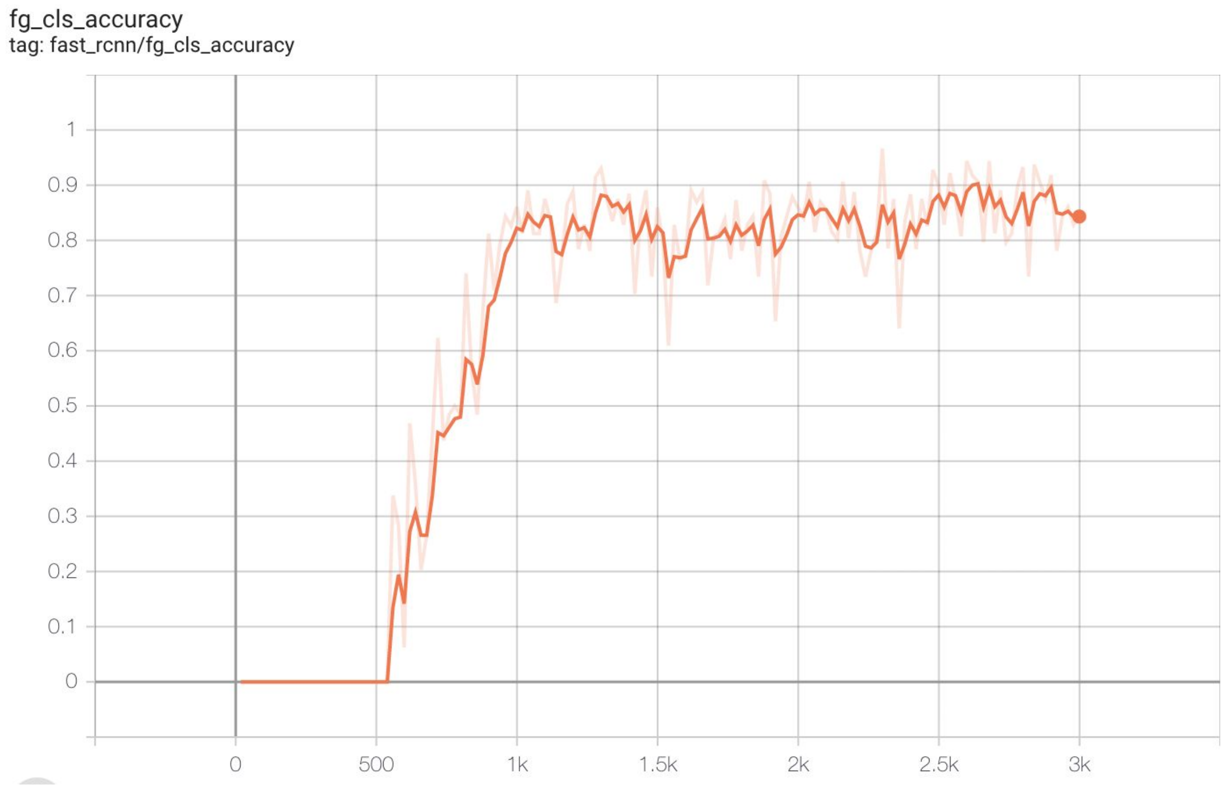
RCNN: Model-1 with FPN

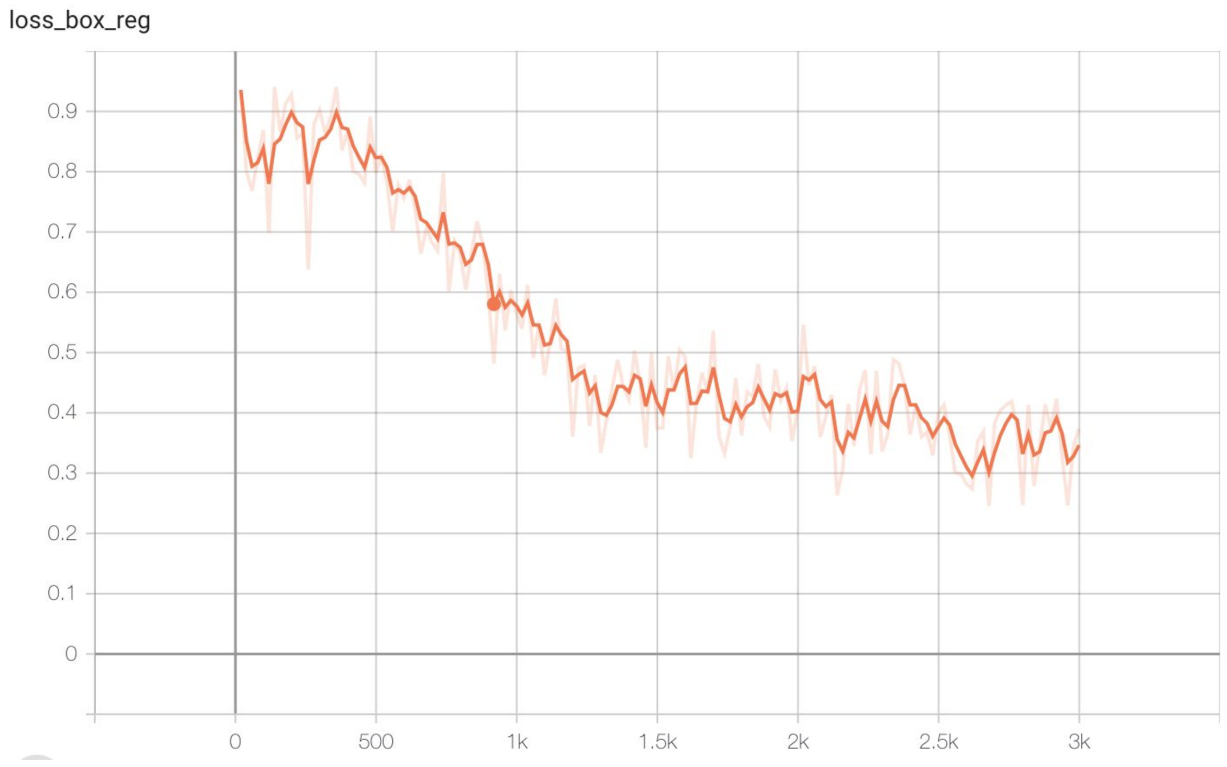
AP:	48.33
AP50:	78.48
AP75:	53.33

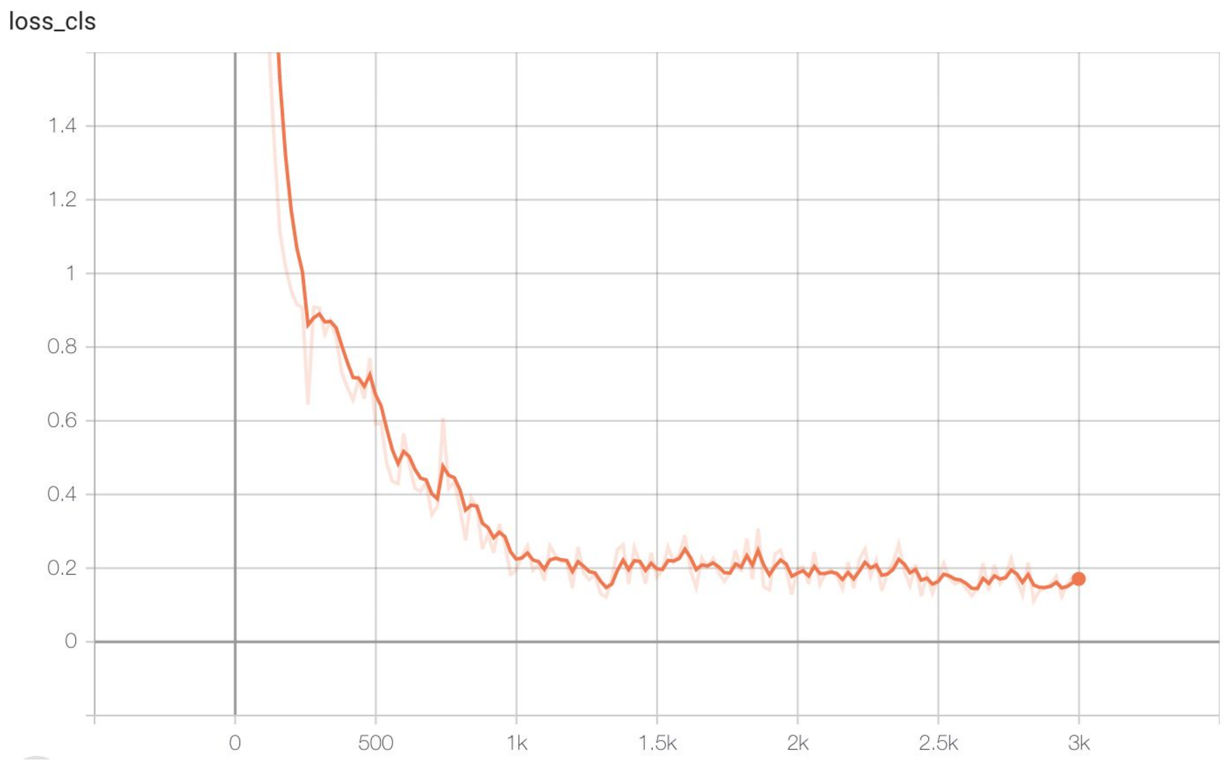
Plots:

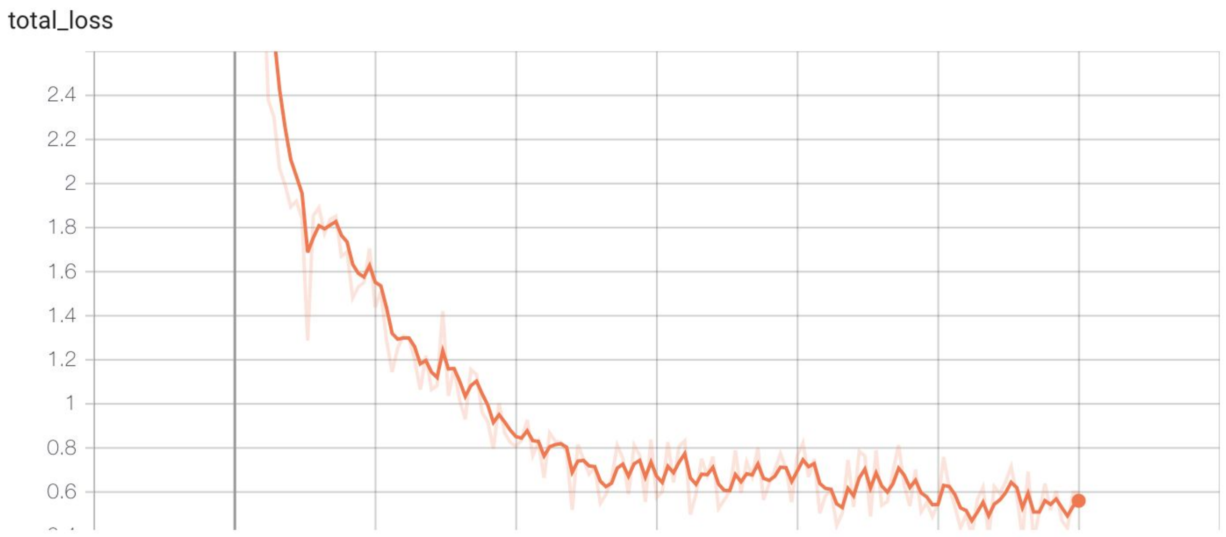





fg_cls_accuracy
tag: fast_rcnn/fg_cls_accuracy
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1k
1.5k
2k
2.5k
3k


loss_box_reg
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1k
1.5k
2k
2.5k
3k


loss_cls
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1k
1.5k
2k
2.5k
3k

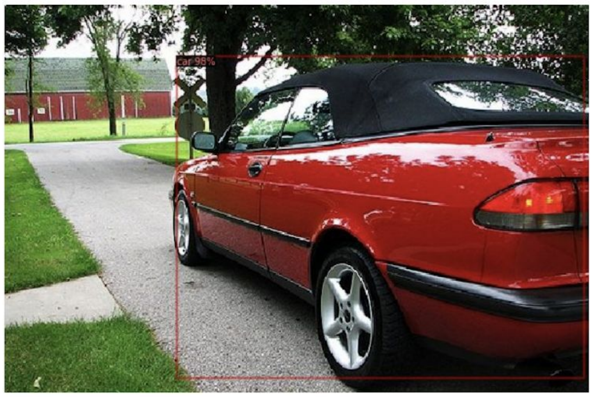

total_loss
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6

Visual Results:

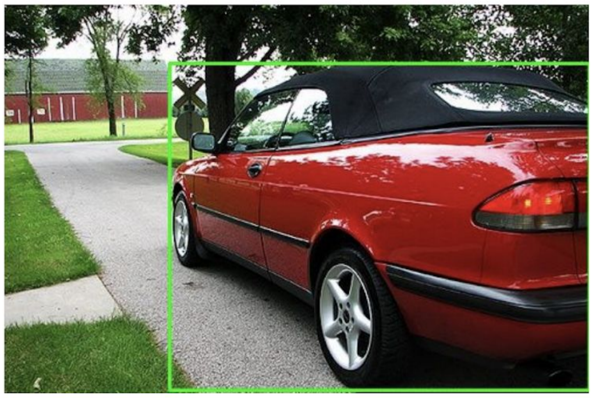
1.

Input Image

Predicted bounding box



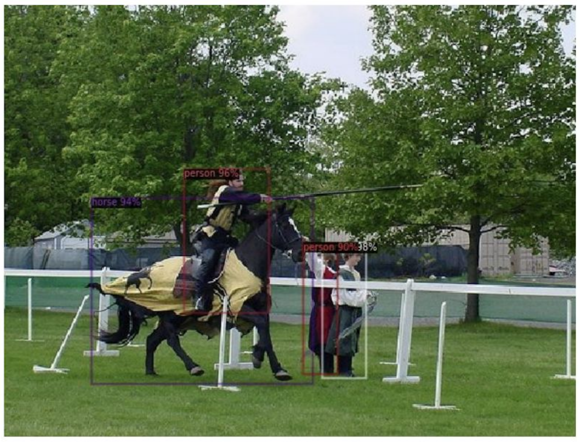
Ground truth bounding box



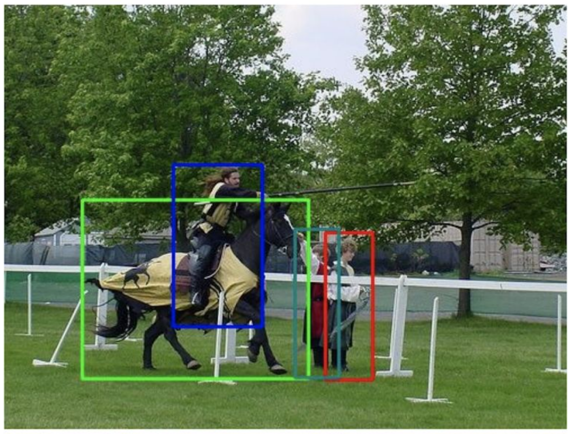
Success: This is a simple example where the prediction is 98% accurate.

Input Image

Predicted bounding box

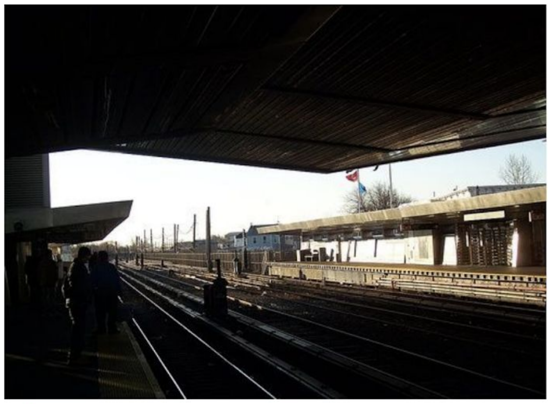


Ground truth bounding box

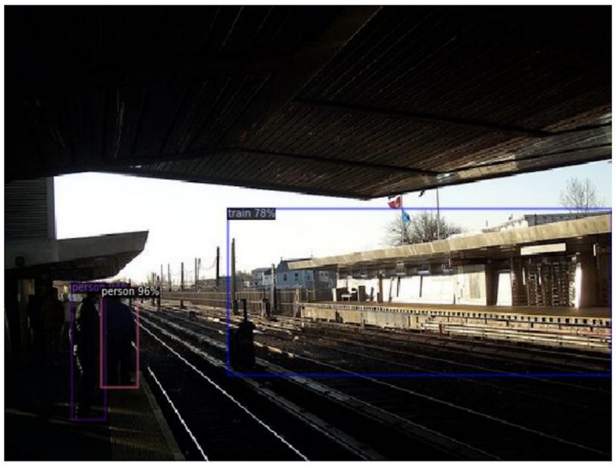


Success: This is a slightly difficult one and the network is able to detect each object correctly with accurate bounding boxes.

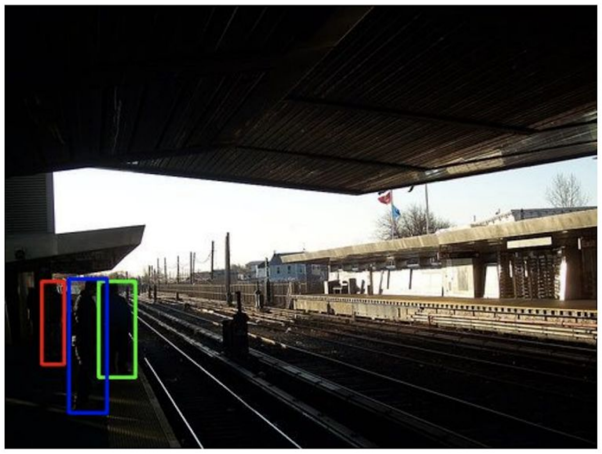
Input Image



Predicted bounding box



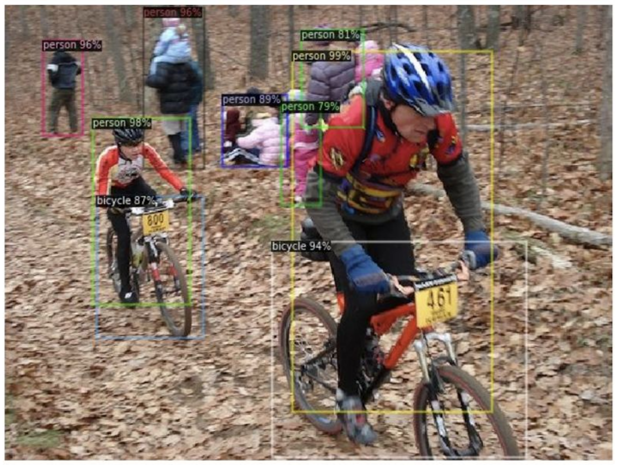
Ground truth bounding box



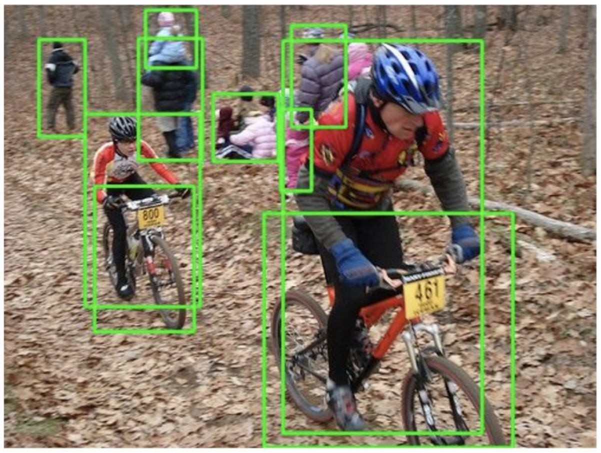
Failure: In this example, the ground truth annotations have 3 objects (person). However, the detector wrongly drew a bounding box near the rail track as a train with 78% confidence. So, the detection has one false positive.

Input Image

Predicted bounding box



Ground truth bounding box

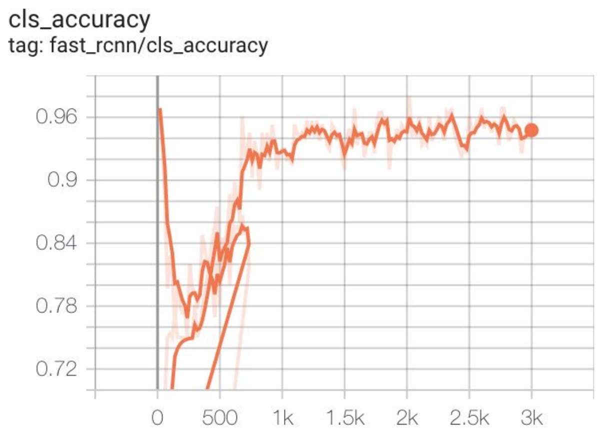


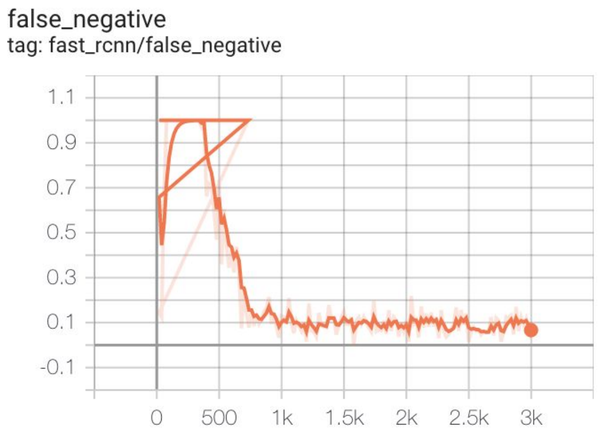
Success: This is slightly tough one, but the detector is able to detect all the objects correctly with accurate bounding boxes around the object.

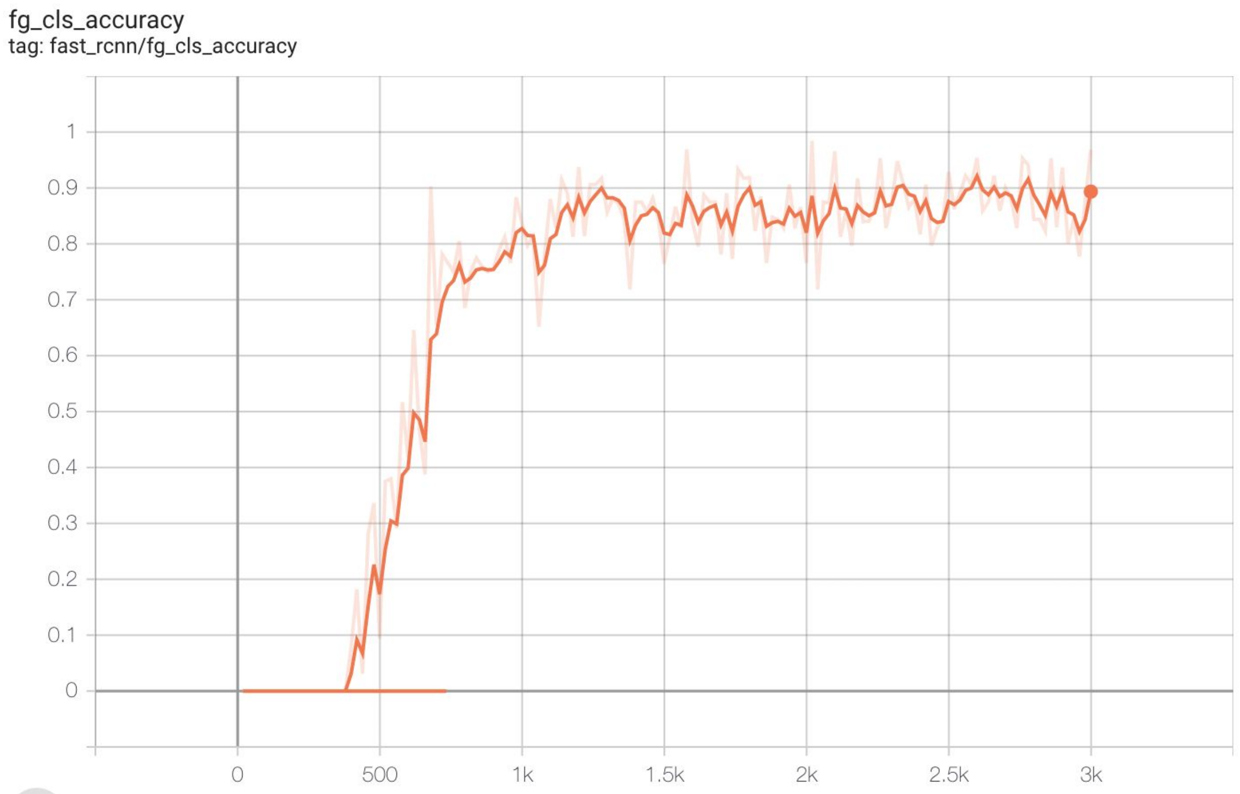
RCNN: Model-1 without FPN

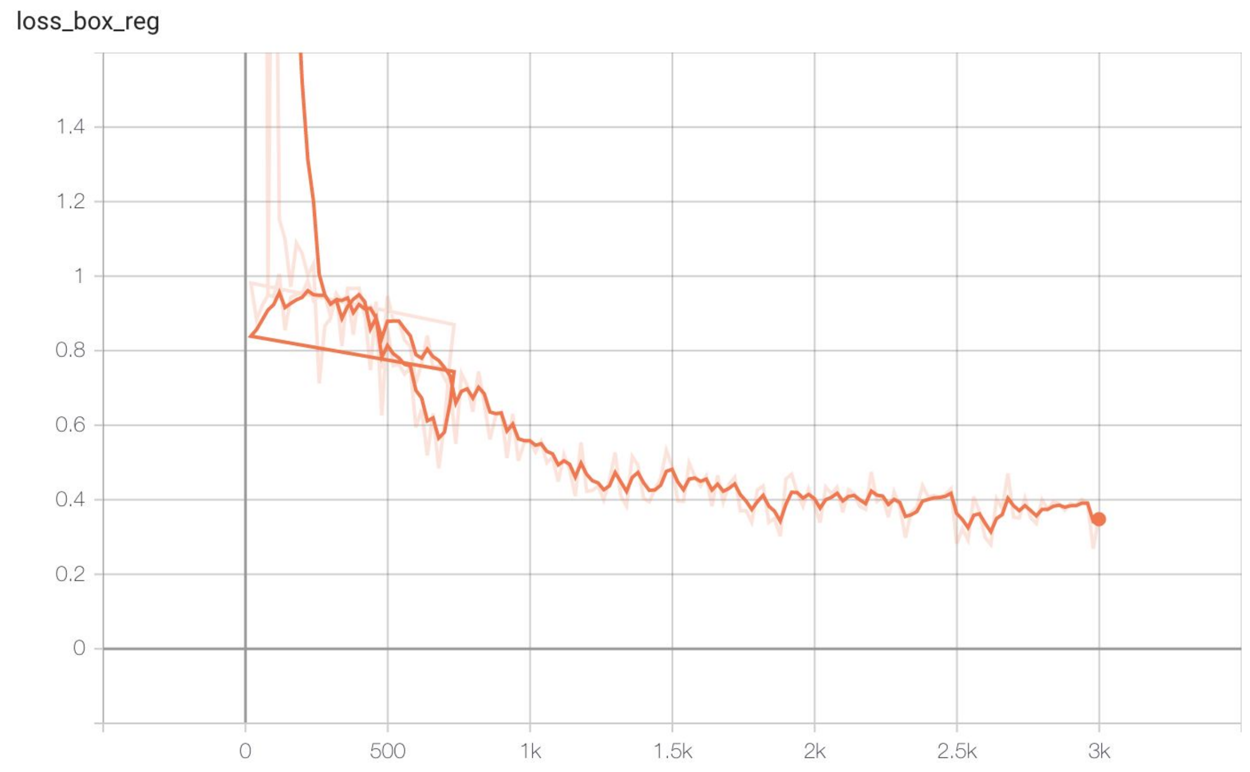
AP: 48.02
AP50: 76.43
AP75: 52.46

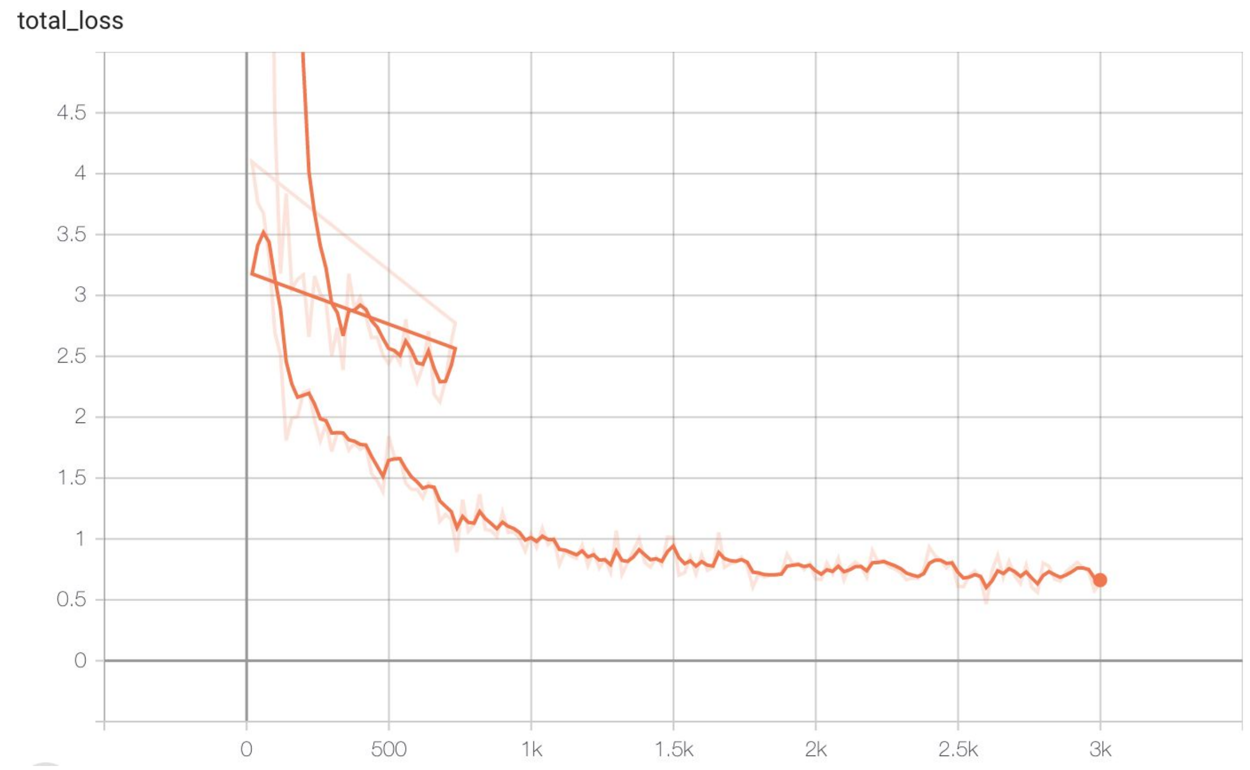
Plots:



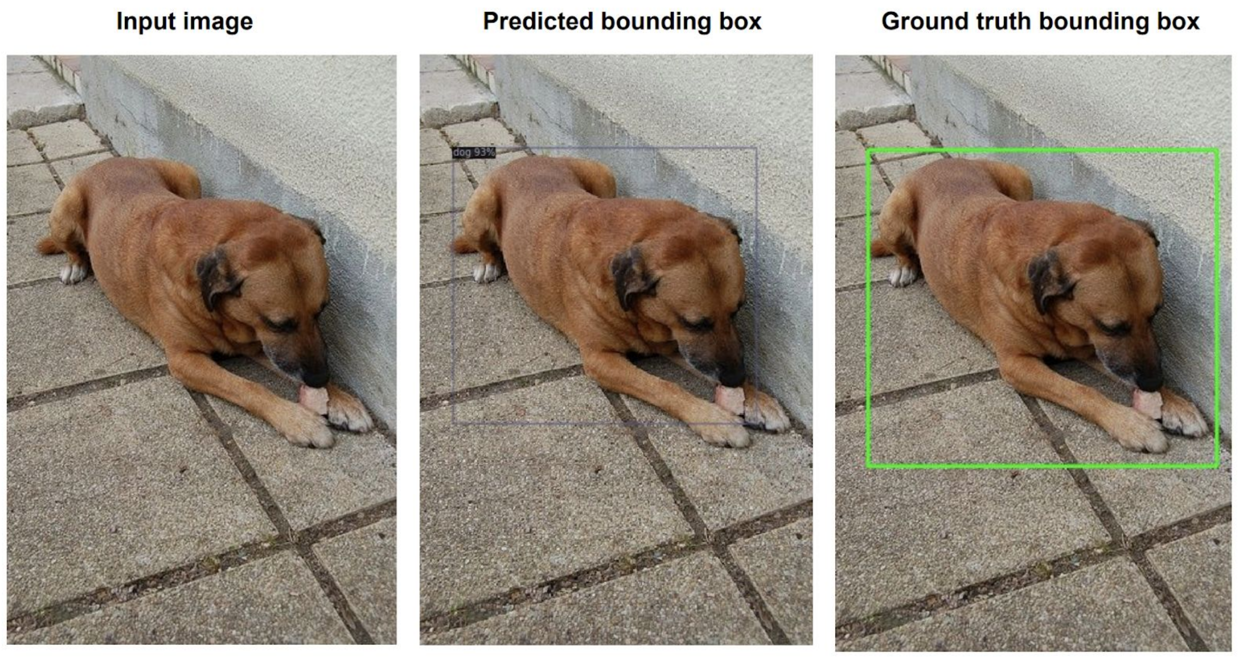





loss_box_reg
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1k
1.5k
2k
2.5k
3k


total_loss
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
500
1k
1.5k
2k
2.5k
3k

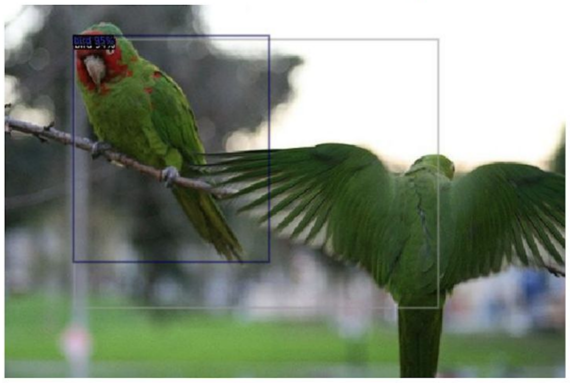
Visual results:



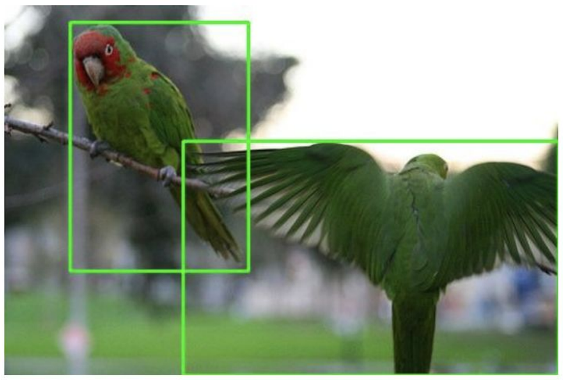
Success: This is a very simple example where the detector found the object with 93% accuracy.

Input image

Predicted bounding box



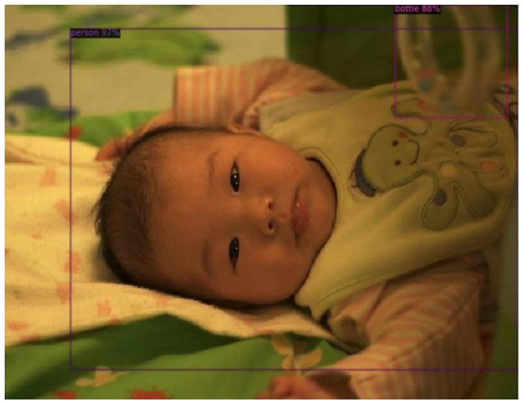
Ground truth bounding box



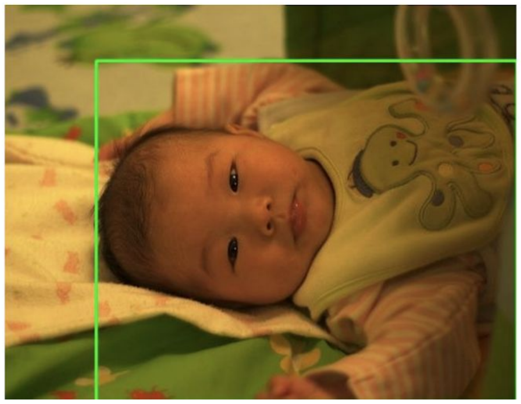
Failure: This example clearly shows the effect of Feature Pyramid Network (FPN) on detection. Without the detector enhancement, the detector marked the bounding box in the wrong position (lesser IoU), even though the object is detected as bird.

Input image

Predicted bounding box



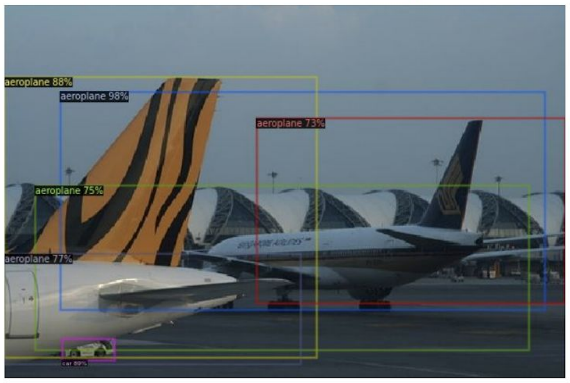
Ground truth bounding box



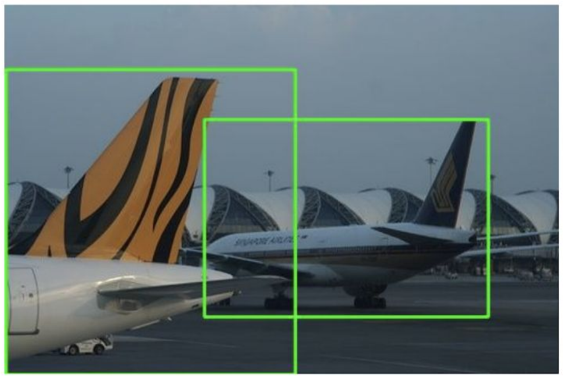
Failure: In this example, the ground truth annotations have only 1 object (person). However, the detector wrongly drew a bounding box near the toy as a bottle with 88% confidence. So, the detection has one false positive.

Input image

Predicted bounding box



Ground truth bounding box



Failure: In this example, the ground truth annotations have 2 objects (aeroplanes). However, the detector wrongly drew bounding boxes detecting 5 objects as aeroplanes. So, the detection has 3 false positives. This example shows why the average precision decreases for the trained network with FPN.

4. **Discussion:**

   In this assignment, we fine tuned the Faster RCNN detector using detectron-2 software with two variations: one with Feature Pyramid Network (FPN) and the other without FPN. It is evident from the results shown above that a network trained without FPN produces more false positives compared to the network with FPN. This is substantiated by the Average Precision at 50% IoU of both the networks. There is a 2% drop in the AP50 value for the network trained without FPN, which is a type of detector enhancer. x