Riya Roopesh
301360795

CMPT 412 ASSIGNMENT 3

Late Days Used: 2 days used

Submitted on: 10th November 2023

Kaggle Submission: riri

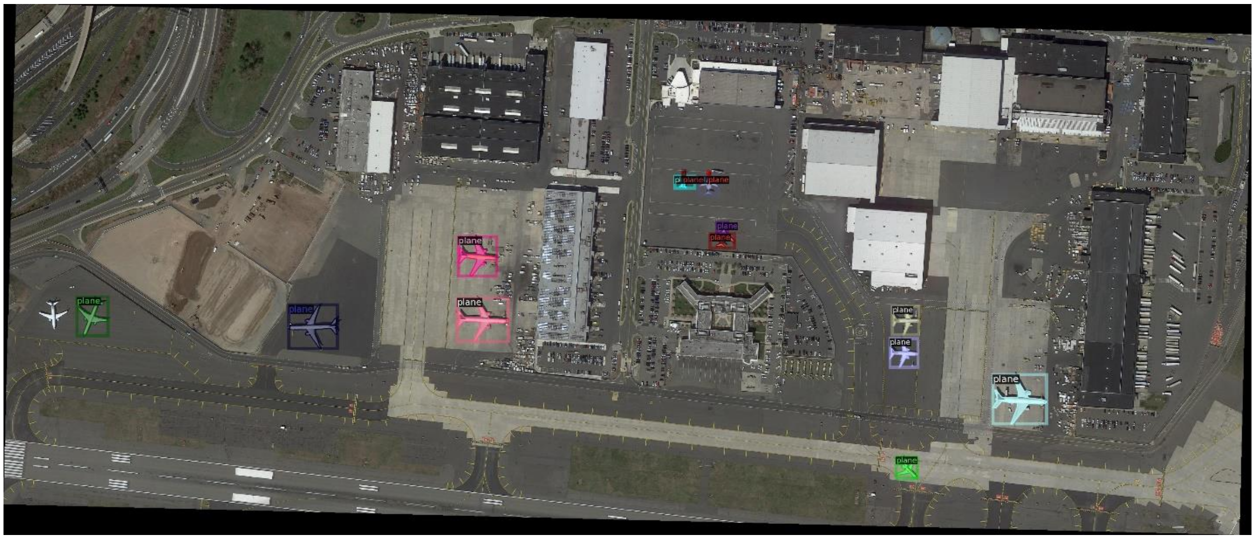
PART1: OBJECT DETECTION

For part1 I loaded the data from train.json file and created the datasets for training and testing. The training data contained all the essential information such as image_height, image_width and annotations etc. However, for the test dataset, annotations is set to 0. The dataset is then registered with DatasetCatalog with appropriate names to be utilized for training and testing purposes. Then I visualized the following images from the train dataset.

Visualizations:

These enable us to ensure that the functions are working well.

Image1:



Riya Roopesh
301360795

Image 2:

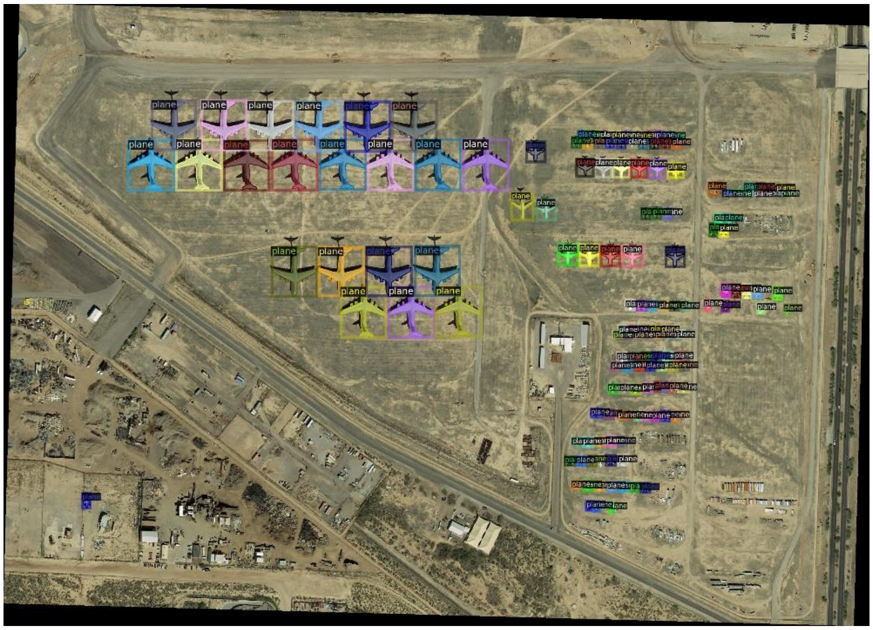
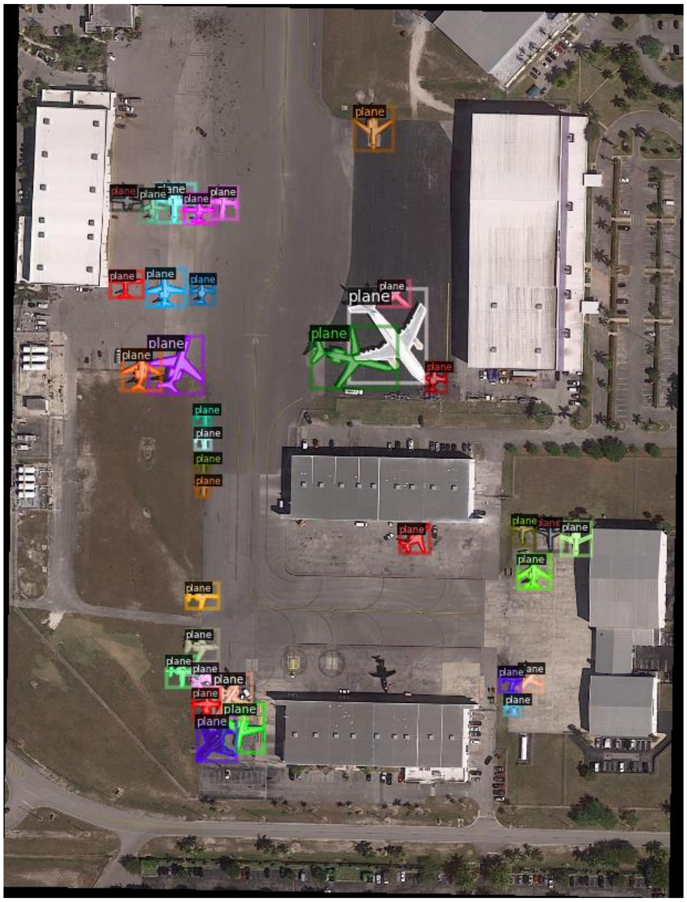
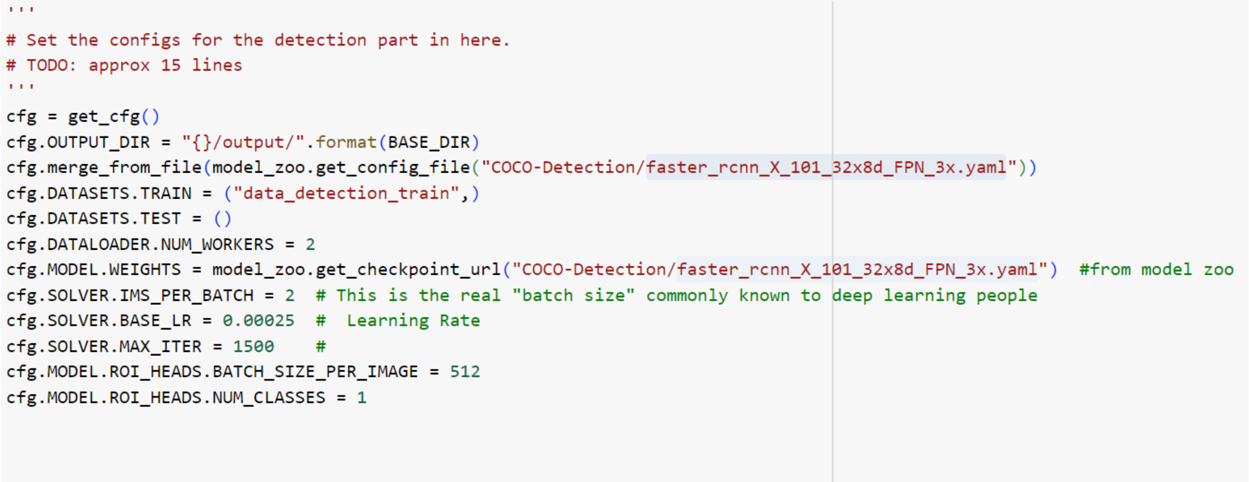


Image3:



Riya Roopesh
301360795

List of configs:

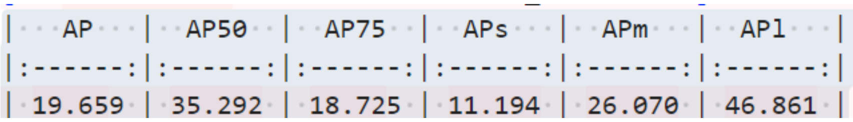


```
'''
# Set the configs for the detection part in here.
# TODO: approx 15 lines
'''
cfg = get_cfg()
cfg.OUTPUT_DIR = "{}/output/".format(BASE_DIR)
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("data_detection_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")  #from model zoo
cfg.SOLVER.IMS_PER_BATCH = 2  # This is the real "batch size" commonly known to deep learning people
cfg.SOLVER.BASE_LR = 0.00025  #  Learning Rate
cfg.SOLVER.MAX_ITER = 1500    #
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
```

Ablation Study:

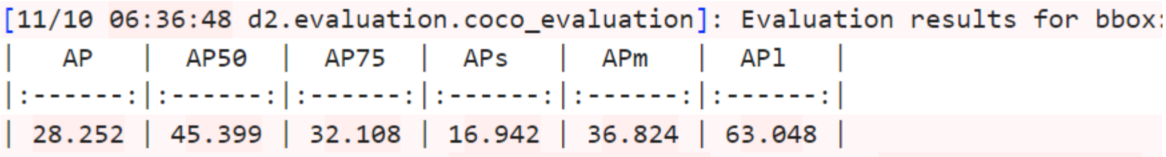
In the list of configurations, the main change I made was to the MAX_ITER. Initially it was at 500. But I obtained my best results after increasing it to **1500**. This was effective as it enabled the model to gradually improve its accuracy due to the small learning rate. Based on the evaluation done we obtained the following outcomes.

AP VALUES (At MAX_ITER =500):



```
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 19.659 | 35.292 | 18.725 | 11.194 | 26.070 | 46.861 |
```

AP VALUES (At MAX_ITER = 1500):



```
[11/10 06:36:48 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 28.252 | 45.399 | 32.108 | 16.942 | 36.824 | 63.048 |
```

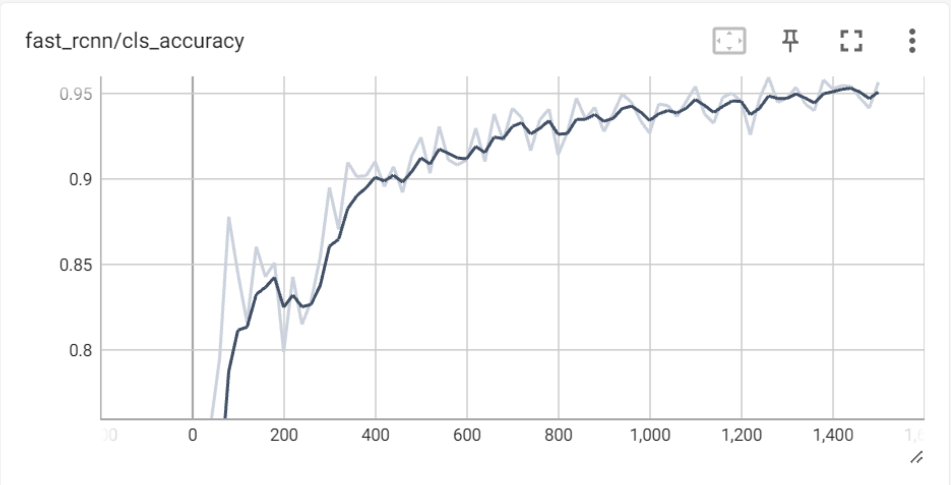
We can notice that increasing the MAX_ITER has resulted in a better evaluation result as shown in the snip provided above. And upon changing the model from **faster_rcnn_R_101_FPN_3x.yaml** to **faster_rcnn_X_101_32x8d_FPN_3x.yaml** I was able to achieve an even higher AP value on evaluation. It is a deeper network which is more complex. It demands for higher computational resources but results in a better result as compared to the baseline network.

Riya Roopesh
301360795

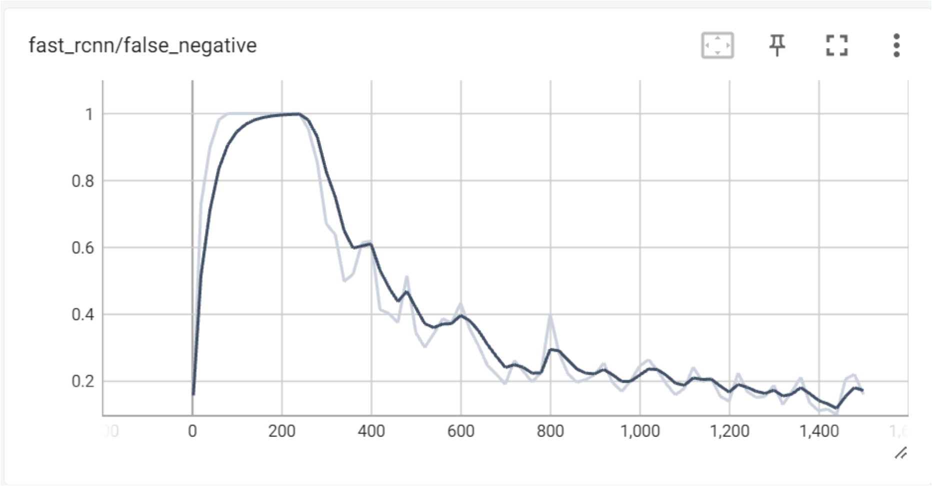
AP values after changing to faster_rcnn_X_101_32x8d_FPN_3x.yaml:

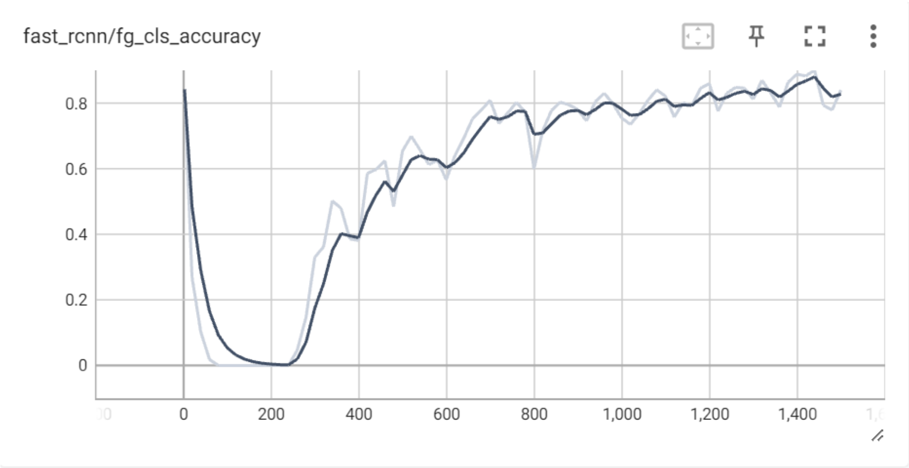
```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.309
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.476
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.360
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.190
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.399
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.693
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.016
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.133
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.332
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.190
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.424
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.761
[11/10 18:08:44 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 30.923 | 47.627 | 35.965 | 18.960 | 39.867 | 69.318 |
OrderedDict([('bbox', {'AP': 30.922649666646834, 'AP50': 47.62689955957731, 'AP75':
35.965267332872116, 'APs': 18.960130170370224, 'APm': 39.86656579860756, 'APl':
69.31791654481027})])
```

Plots for training accuracy:

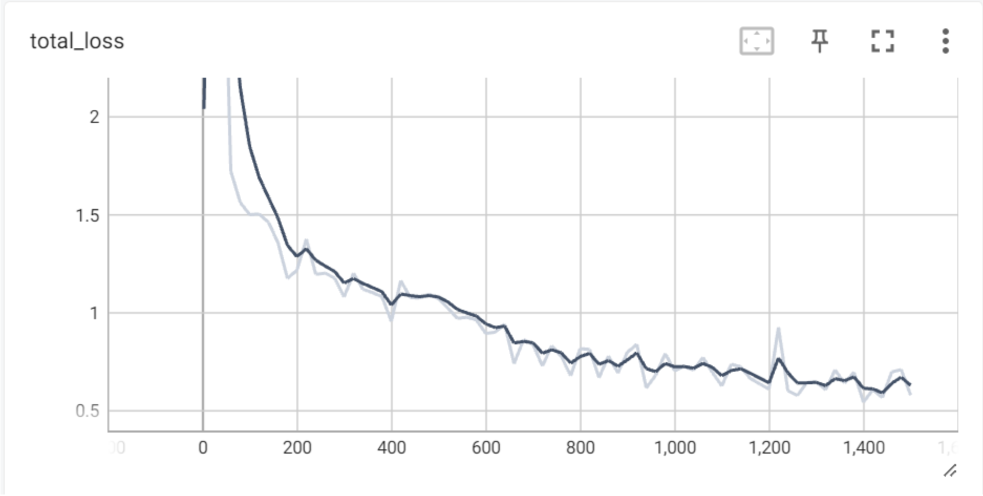


Riya Roopesh
301360795





Plot for Total Loss:



Riya Roopesh
301360795

Visualizations of test samples with the prediction results:

Image1:

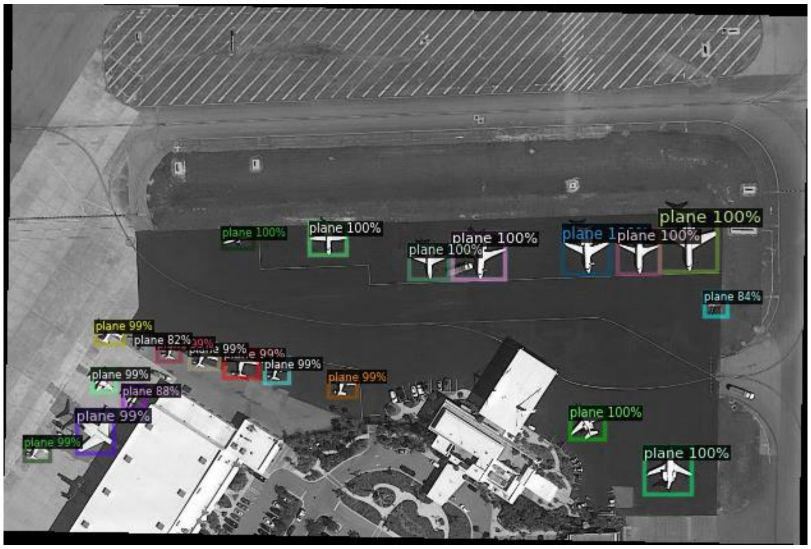
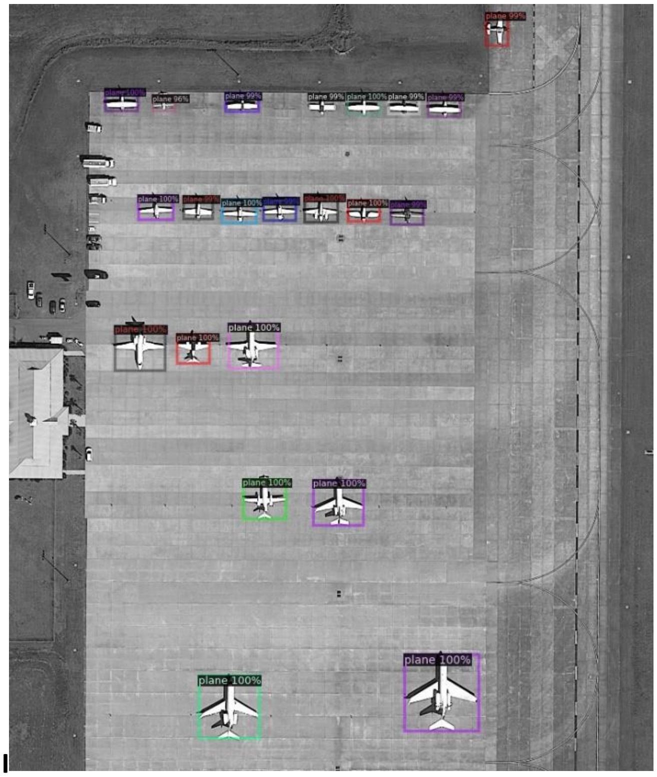
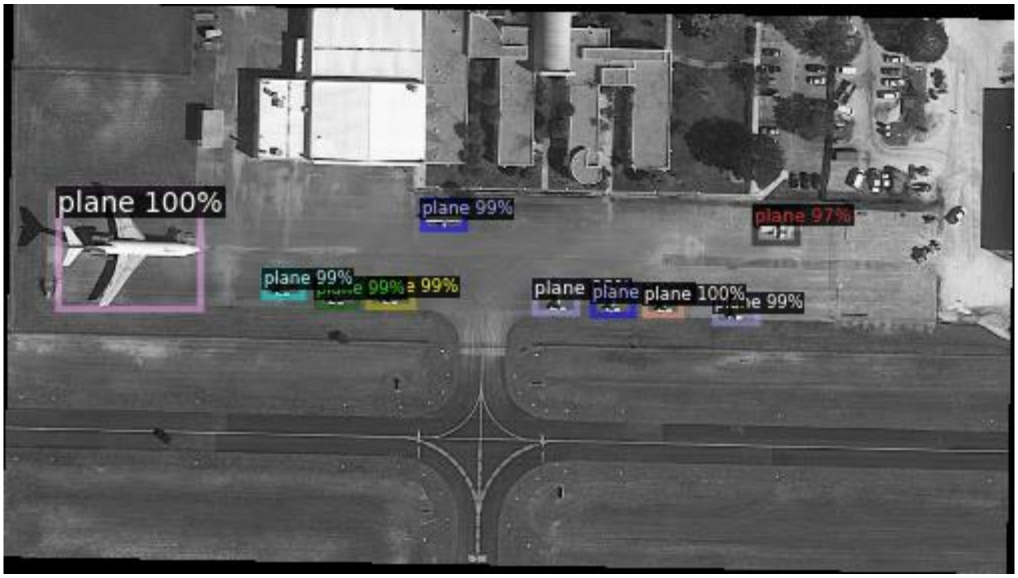


Image 2:



Riya Roopesh
301360795

Image 3:



Based on the above images we notice that the model did pretty well in predicting the planes among the given images.

Evaluation:

We utilized to COCOEvaluator to evaluate this model and the following are the results we achieved based on the configuration as shown above.

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.309
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.476
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.360
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.190
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.399
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.693
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.016
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.133
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.332
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.190
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.424
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.761
[11/10 18:08:44 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 30.923 | 47.627 | 35.965 | 18.960 | 39.867 | 69.318 |
OrderedDict([('bbox', {'AP': 30.922649666646834, 'AP50': 47.62689955957731, 'AP75':
35.965267332872116, 'APs': 18.960130170370224, 'APm': 39.86656579860756,'APl':
69.31791654481027})])
```

Riya Roopesh
301360795

PART2 SEMANTIC SEGMENTATION:

Hyperparameters:

Num_epochs: 80

Batch_size : 15

Learning_rate:0.015

Optimizer :SGD optimizer

Loss function : nn.BCEWithLogitsLoss()

Network architecture:

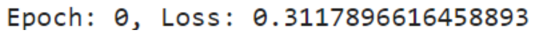
The model consists of four down sampling and four up sampling layers. It also consists of an input_conv layer and an output conv layer. And a additional convolution layer right before the output conv layer. The initial input_conv layer was given (3,32) the following input and output channels respectively. The following down sampling layers increased the output channels to twice the number of input channels. And the up sampling layers in turn decreased the output channels to half the number of input channels. The following is a layer by layer representation of my model.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 32, 128, 128]             896
       BatchNorm2d-2        [-1, 32, 128, 128]              64
              ReLU-3        [-1, 32, 128, 128]               0
              conv-4        [-1, 32, 128, 128]               0
            Conv2d-5        [-1, 64, 128, 128]          18,496
       BatchNorm2d-6        [-1, 64, 128, 128]             128
              ReLU-7        [-1, 64, 128, 128]               0
              conv-8        [-1, 64, 128, 128]               0
         MaxPool2d-9          [-1, 64, 64, 64]               0
           down-10          [-1, 64, 64, 64]               0
          Conv2d-11         [-1, 128, 64, 64]          73,856
     BatchNorm2d-12         [-1, 128, 64, 64]             256
            ReLU-13         [-1, 128, 64, 64]               0
            conv-14         [-1, 128, 64, 64]               0
       MaxPool2d-15         [-1, 128, 32, 32]               0
            down-16         [-1, 128, 32, 32]               0
          Conv2d-17         [-1, 256, 32, 32]         295,168
     BatchNorm2d-18         [-1, 256, 32, 32]             512
            ReLU-19         [-1, 256, 32, 32]               0
            conv-20         [-1, 256, 32, 32]               0
       MaxPool2d-21         [-1, 256, 16, 16]               0
            down-22         [-1, 256, 16, 16]               0
          Conv2d-23         [-1, 512, 16, 16]       1,180,160
```

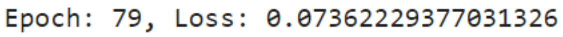
```
      BatchNorm2d-24         [-1, 512, 16, 16]           1,024
             ReLU-25         [-1, 512, 16, 16]               0
             conv-26         [-1, 512, 16, 16]               0
        MaxPool2d-27           [-1, 512, 8, 8]               0
             down-28           [-1, 512, 8, 8]               0
  ConvTranspose2d-29         [-1, 512, 16, 16]       1,049,088
           Conv2d-30         [-1, 256, 16, 16]       1,179,904
      BatchNorm2d-31         [-1, 256, 16, 16]             512
             ReLU-32         [-1, 256, 16, 16]               0
             conv-33         [-1, 256, 16, 16]               0
               up-34         [-1, 256, 16, 16]               0
  ConvTranspose2d-35         [-1, 256, 32, 32]         262,400
           Conv2d-36         [-1, 128, 32, 32]         295,040
      BatchNorm2d-37         [-1, 128, 32, 32]             256
             ReLU-38         [-1, 128, 32, 32]               0
             conv-39         [-1, 128, 32, 32]               0
               up-40         [-1, 128, 32, 32]               0
  ConvTranspose2d-41         [-1, 128, 64, 64]          65,664
           Conv2d-42          [-1, 64, 64, 64]          73,792
      BatchNorm2d-43          [-1, 64, 64, 64]             128
             ReLU-44          [-1, 64, 64, 64]               0
             conv-45          [-1, 64, 64, 64]               0
               up-46          [-1, 64, 64, 64]               0
  ConvTranspose2d-47        [-1, 64, 128, 128]          16,448
           Conv2d-48        [-1, 32, 128, 128]          18,464
      BatchNorm2d-49        [-1, 32, 128, 128]              64
             ReLU-50        [-1, 32, 128, 128]               0
             conv-51        [-1, 32, 128, 128]               0
               up-52        [-1, 32, 128, 128]               0
           Conv2d-53         [-1, 4, 128, 128]           1,156
      BatchNorm2d-54         [-1, 4, 128, 128]               8
             ReLU-55         [-1, 4, 128, 128]               0
             conv-56         [-1, 4, 128, 128]               0
           Conv2d-57         [-1, 1, 128, 128]              37
             conv-58         [-1, 1, 128, 128]               0
================================================================
Total params: 4,533,521
Trainable params: 4,533,521
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 138.25
Params size (MB): 17.29
Estimated Total Size (MB): 155.73
----------------------------------------------------------------
```

In the following case the number of epochs was increased to **80** and the optimizer remained the same. The main reason to increase the epochs was to ensure the loss value at this point began to stabilize and plateau. The loss at the first epoch is **0.311** and the last epoch we were able to achieve a loss of **0.073**. Increasing the batch size to **15** helped speed up the training process by a small amount. And increasing the learning rate to **0.015** helped in achieving much lower loss towards the end of training and a better prediction. The loss functions was kept the same.

Riya Roopesh
301360795

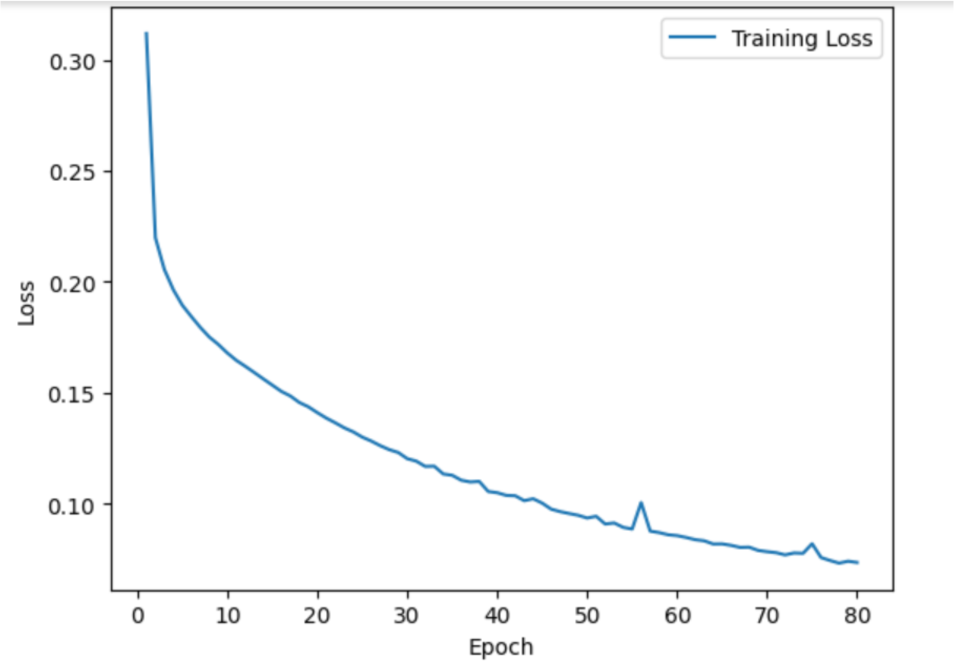


```
Epoch: 0, Loss: 0.3117896616458893
```



```
Epoch: 79, Loss: 0.07362229377031326
```

Plot of training loss to epochs:

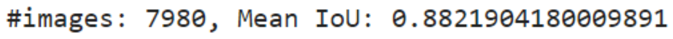


The plot represents that the loss gradually keeps decreasing.

Final IoU value:

The final IoU value is 0.882 .

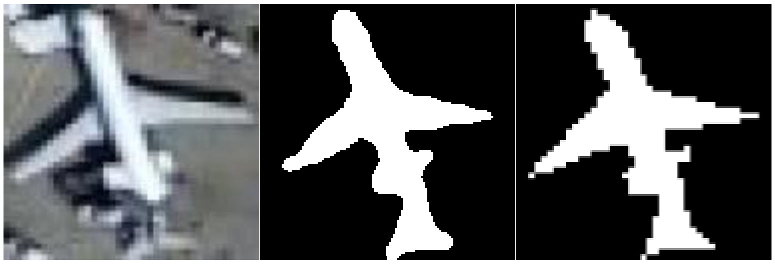
```
100% 1995/1995 [01:06<00:00, 30.52it/s]
```



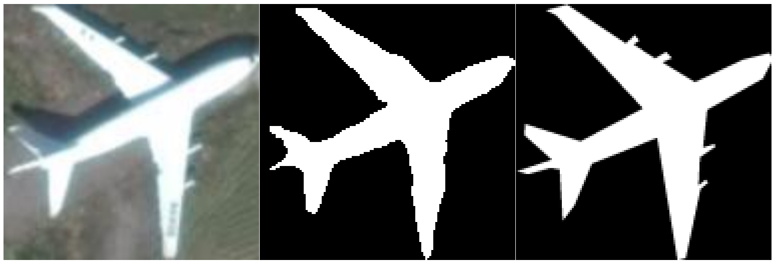
```
#images: 7980, Mean IoU: 0.8821904180009891
```

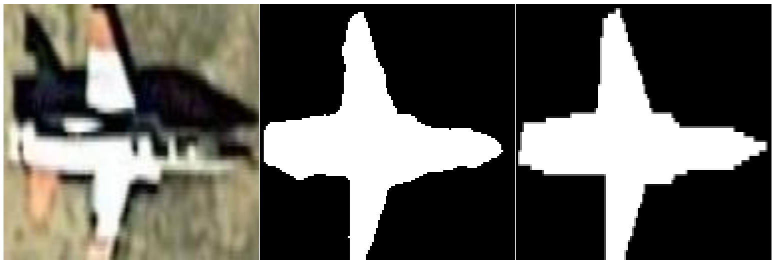
Visualizations:

These are visualizations of original plane images followed by the prediction and lastly the ground truth mask.



Riya Roopesh
301360795





PART3 Instance Segmentation:

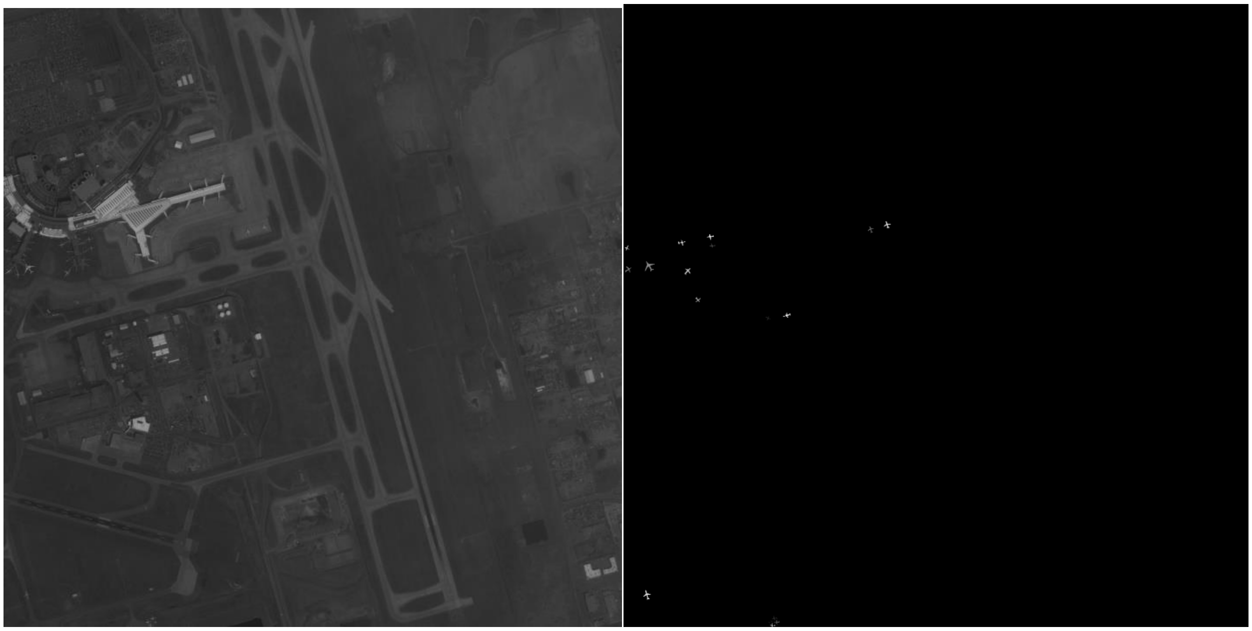
Kaggle submission:

Kaggle submission name: **riri**

Best Score on Kaggle: **0.62806**

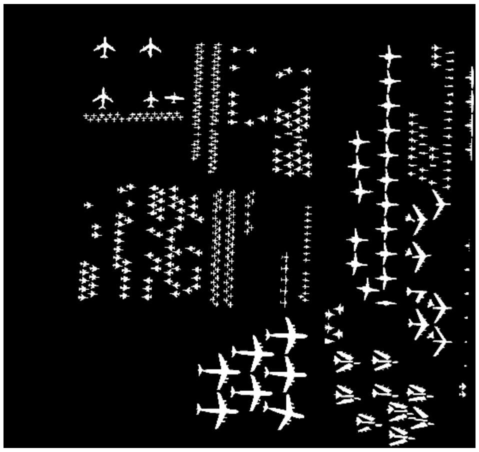
Visualizations :

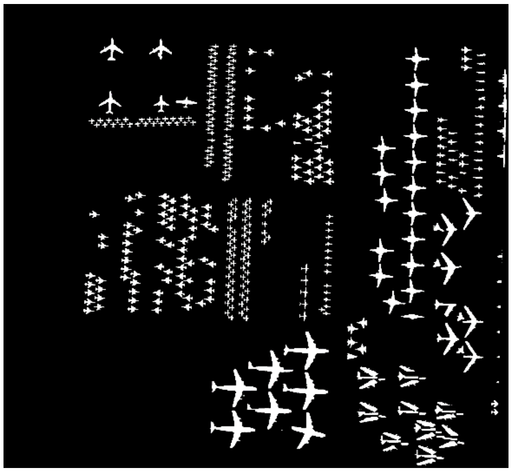
Image1:



Riya Roopesh
301360795

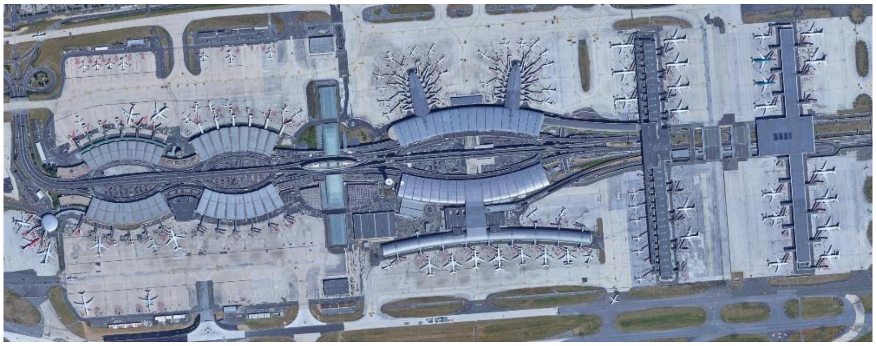
Image 2:

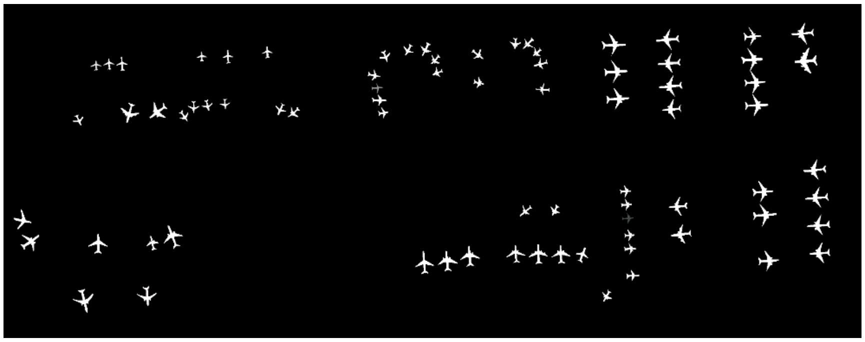


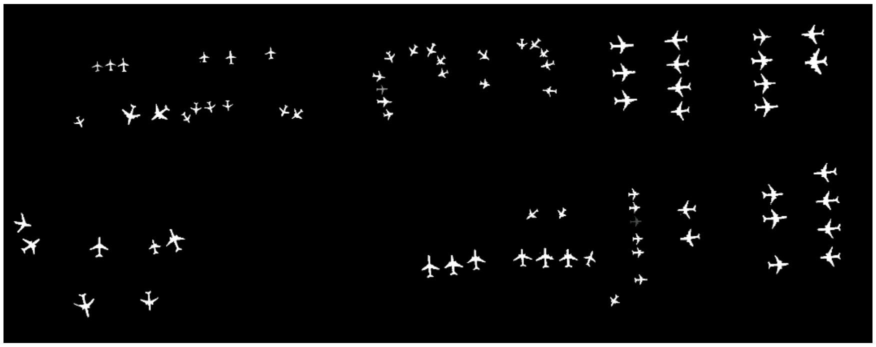


Riya Roopesh
301360795

Image 3:





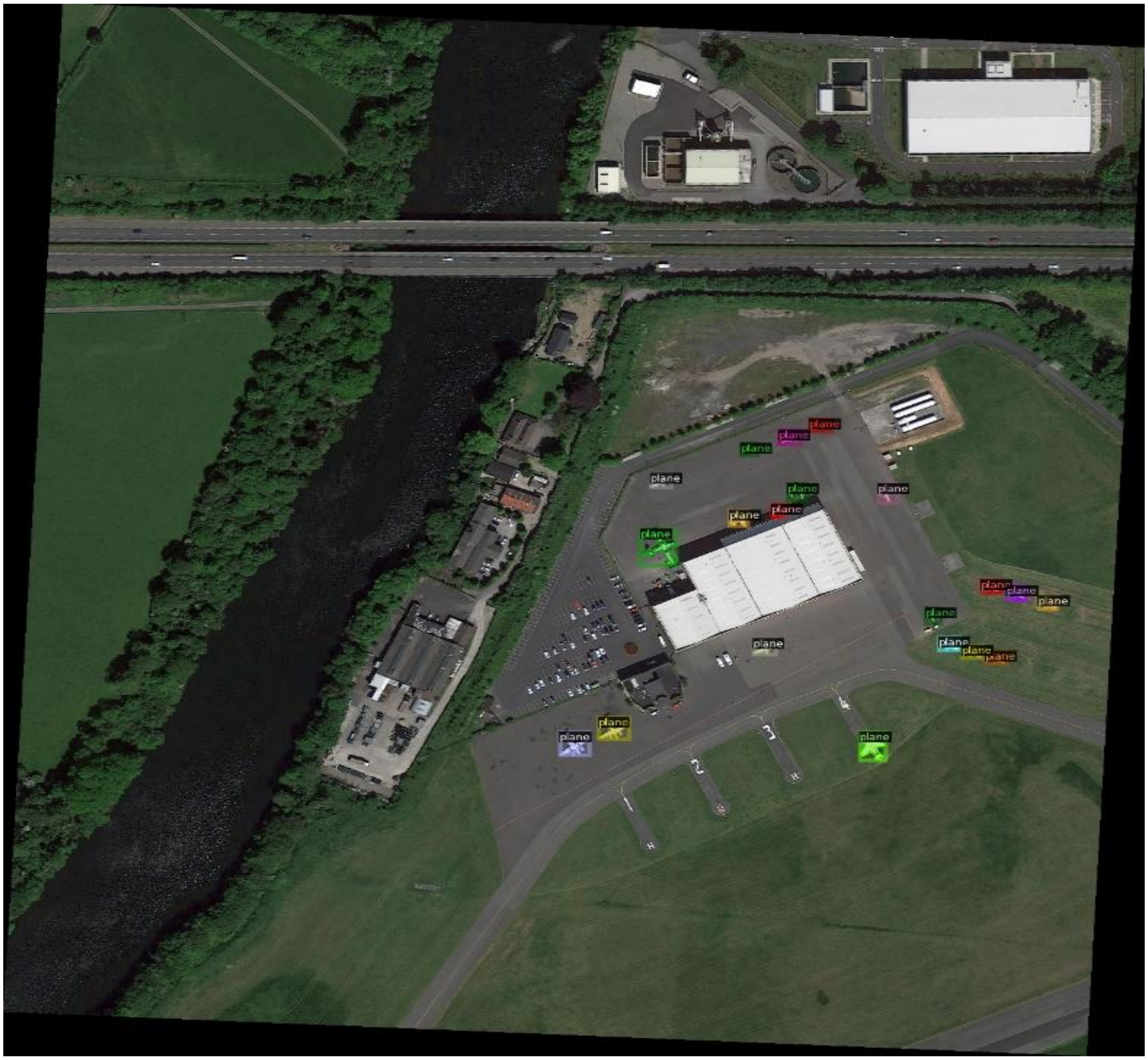


Riya Roopesh
301360795

PART4 : Mask R-CNN

Visualizations:

Image1:



Riya Roopesh
301360795

Image2:

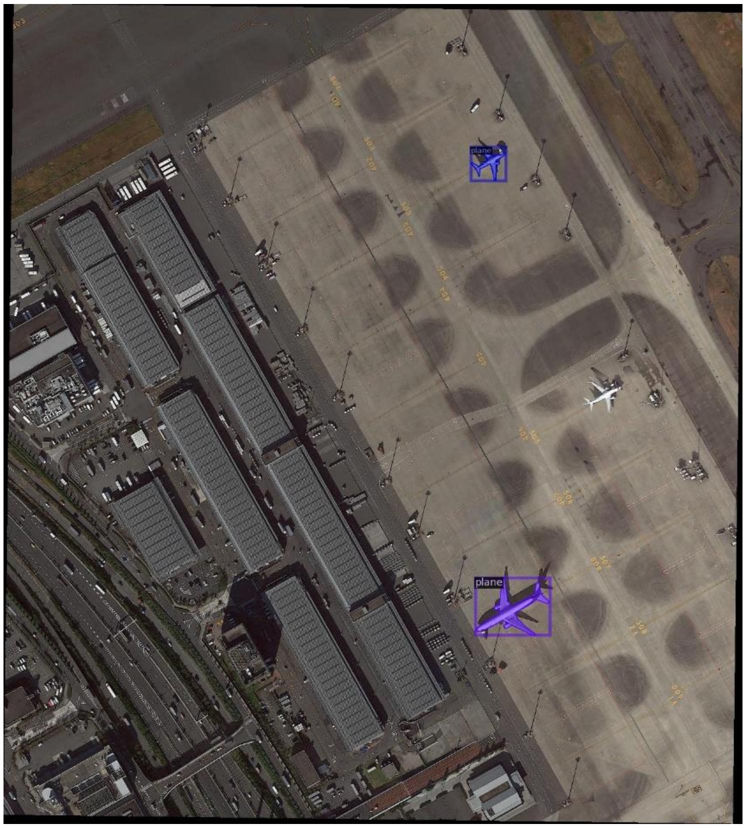
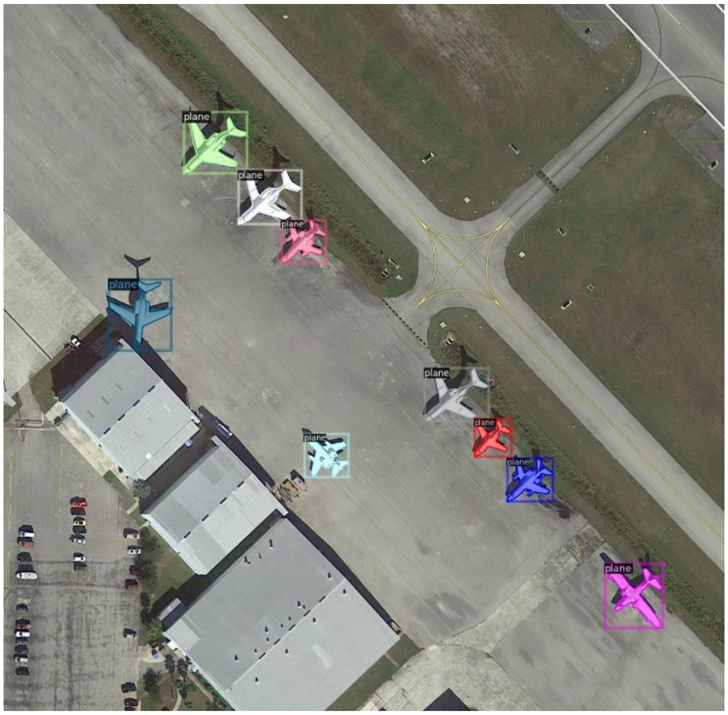


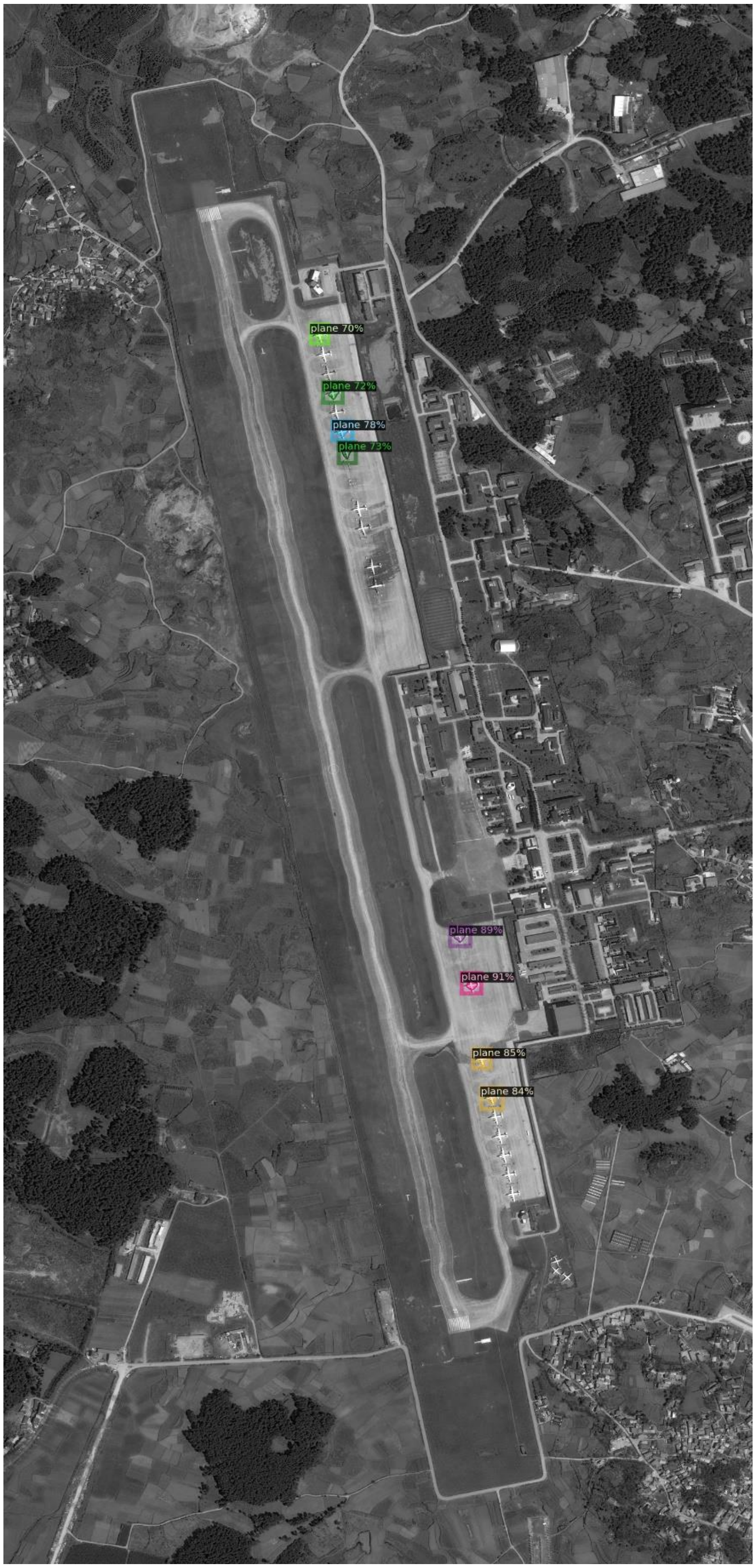
Image3:



Riya Roopesh
301360795

Visualizations with test samples and prediction results:

Image1:



Riya Roopesh
301360795

Image2:

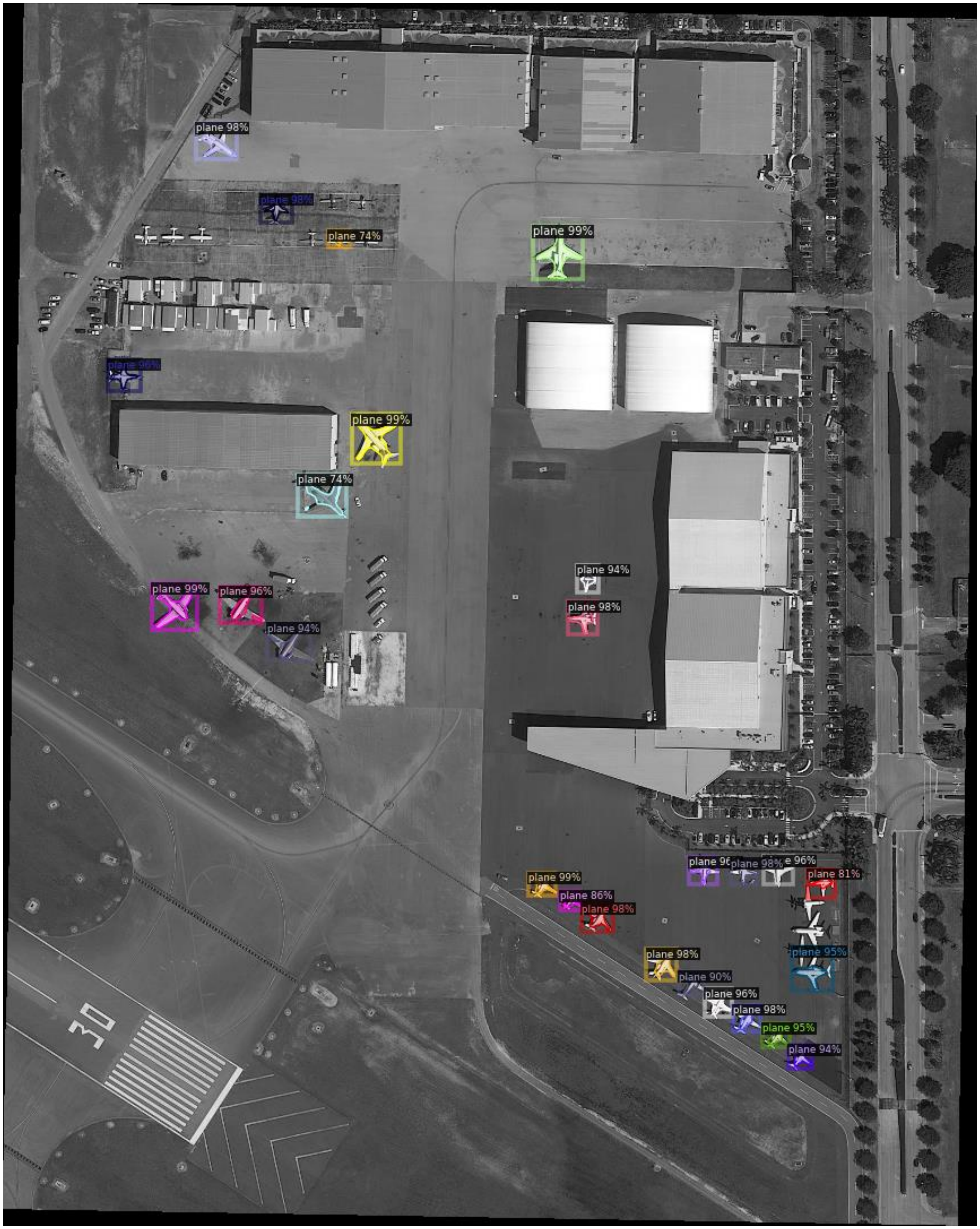
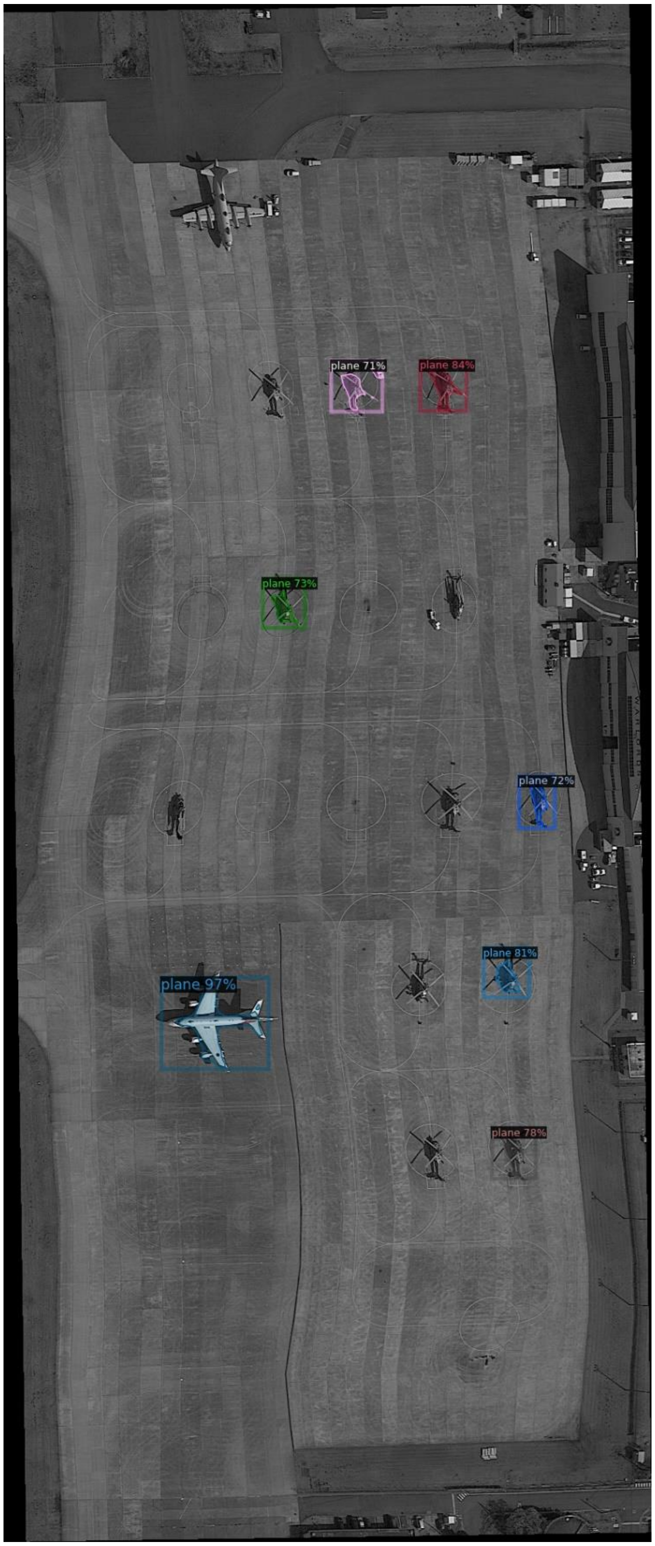


Image3:



From the above visualizations it is evident that this model did not do as well as part1. As its prediction values were not as accurate as part1. Moreover, part 3 has a mean IoU value of 0.88.

Riya Roopesh
301360795

Based on the COCOEvaluation part1 AP values are much higher than part4 as shown below. These values further indicate that part4 model is not as accurate as the previous parts.

AP values for part 1:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.309
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.476
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.360
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.190
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.399
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.693
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.016
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.133
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.332
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.190
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.424
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.761
[11/10 18:08:44 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 30.923 | 47.627 | 35.965 | 18.960 | 39.867 | 69.318 |
```

AP values for part 4:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.210
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.354
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.236
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.115
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.284
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.479
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.014
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.109
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.235
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.113
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.312
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.619
[11/10 23:08:17 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 20.983 | 35.369 | 23.553 | 11.534 | 28.442 | 47.898 |
```