Report Project 3

Object Detection, Semantic Segmentation, and Instance Segmentation

I am using my 3 late days for assignment 3 (Project 2)

Name: *Jashanraj Singh Gosain*

Student Number: *301435386*

Late Days*: 3 Used*

Part 1:

The following configuration was used in part1.

```
'''
# Set the configs for the detection part in here.
# TODO: approx 15 lines
'''
from detectron2.engine import DefaultTrainer

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 = ("data_detection_test",)
cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 2  # "batch size"
cfg.SOLVER.BASE_LR = 0.00025  # LR
cfg.SOLVER.MAX_ITER = 1500    # iterations
cfg.SOLVER.STEPS = []        # decay learning rate
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512  # "RoIHead batch size"
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1  # only has one class (Plane)
```

Modifications:

- Max Iterations: Increasing the number of iterations helped lower the loss. With 500 iterations, there was more scope of fine tuning to lower the loss further. Increasing iterations helped reduce the loss.
- Low learning rate has little scope of improvement. It converges to the final accuracy in fewer iterations. High learning rate starts with good accuracy but the accuracy fluctuates and loss starts to increase after convergence. Ideally 0.00025 shows steady improvement with little fluctuations.
- “faster_rcnn_X_101_32x8d_FPN_3x.yaml” is supposedly more accurate model to train.
- ***cfg.DATALOADER.NUM_WORKERS = 4***

  It helps speed up the training by employing 4 process on separate GPU core in parallel.

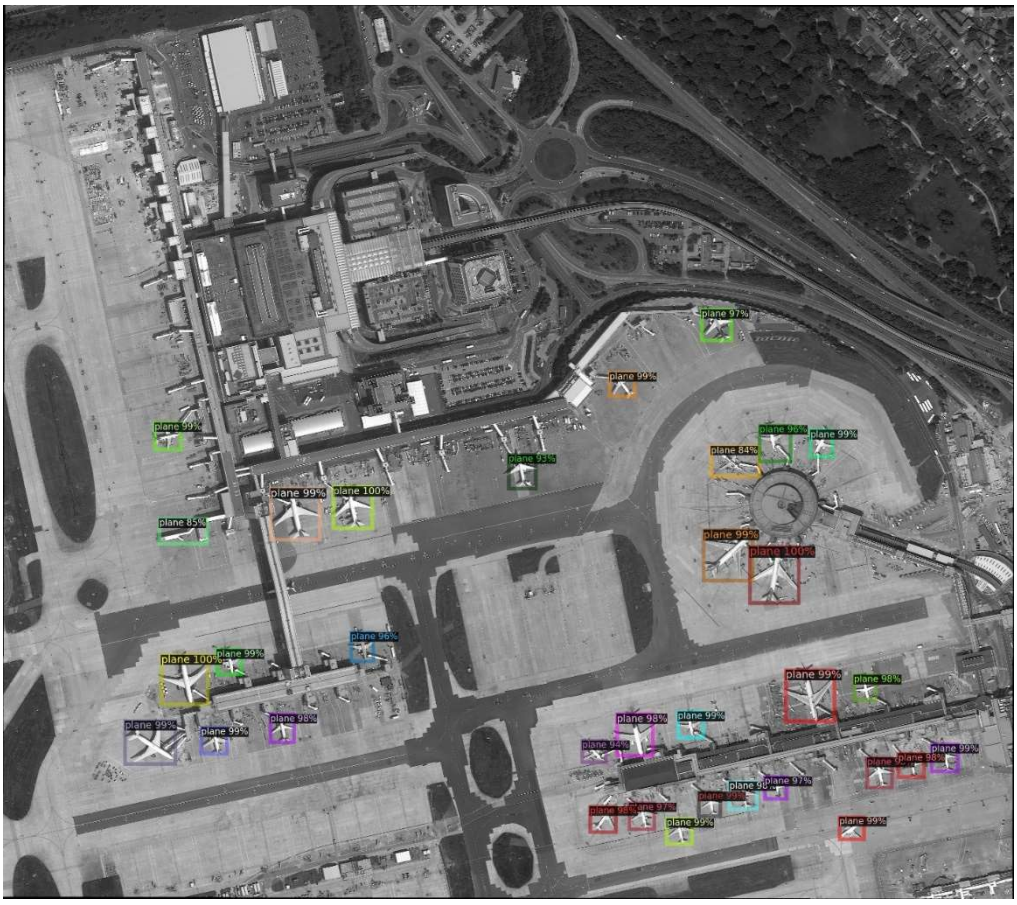
Final Accuracy: 68.668

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.366
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.569
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.420
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.263
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.453
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.687
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.017
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.141
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.400
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.263
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.493
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.781
[11/11 23:26:55 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 36.606 | 56.865 | 41.969 | 26.251 | 45.258 | 68.668 |
OrderedDict([('bbox', {'AP': 36.6059251323455, 'AP50': 56.865410003200445, 'AP75': 41.96888714629425, 'APs': 26.250909023515234, 'APm': 45.25848723794878, 'APl': 68.66845198902169})])
```

Visualization of test samples

Sample 1

Sample 2



Sample 3


plane 100%
plane 98%
plane 99%
plane 69%
plane 100%
plane 100%
plane 99%
plane 99%
plane 98%
plane 100%
plane 100%
plane 79%
plane 100%
plane 76%
plane 89%
plane 100%
plane 100%
plane 89%
plane 98%
plane 94%
plane 94%
plane 100%
plane 80%
plane 87%
plane 100%
plane 88%
plane 100%
plane 82%
plane 100%
plane 100%

Ablation Study:

Increasing **cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE** made the convergence faster:

- For 1024, it converged faster but it keeps fluctuating and loss doesn't reduce further. Converge to total_loss of around 1.

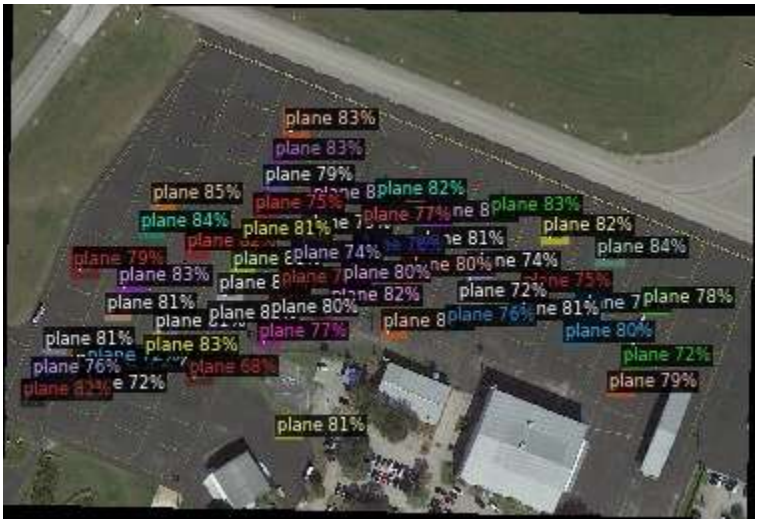
```
[11/11 21:56:23 d2.engine.train_loop]: Starting training from iteration 0
[11/11 21:57:13 d2.utils.events]:  eta: 0:49:40  iter: 19  total_loss: 4.372  loss_cls: 0.6757  loss_box_reg: 0.1734  loss_rpn_cls: 2.953  loss_rpn_loc: 0.5968    time: 2.3204  last_time: 1.7986  data_time:
[11/11 21:57:55 d2.utils.events]:  eta: 0:46:14  iter: 39  total_loss: 2.061  loss_cls: 0.6328  loss_box_reg: 0.2277  loss_rpn_cls: 0.6847  loss_rpn_loc: 0.392    time: 2.2033  last_time: 1.7849  data_time:
[11/11 21:58:39 d2.utils.events]:  eta: 0:45:57  iter: 59  total_loss: 1.453  loss_cls: 0.5575  loss_box_reg: 0.2259  loss_rpn_cls: 0.2968  loss_rpn_loc: 0.2953    time: 2.2052  last_time: 1.8540  data_time
[11/11 21:59:17 d2.utils.events]:  eta: 0:45:04  iter: 79  total_loss: 1.284  loss_cls: 0.4525  loss_box_reg: 0.1779  loss_rpn_cls: 0.2197  loss_rpn_loc: 0.3423    time: 2.1335  last_time: 1.8873  data_time
[11/11 22:00:00 d2.utils.events]:  eta: 0:45:19  iter: 99  total_loss: 1.178  loss_cls: 0.3733  loss_box_reg: 0.1898  loss_rpn_cls: 0.2865  loss_rpn_loc: 0.293    time: 2.1286  last_time: 1.5080  data_time:
[11/11 22:00:41 d2.utils.events]:  eta: 0:45:39  iter: 119  total_loss: 0.9994  loss_cls: 0.326  loss_box_reg: 0.2075  loss_rpn_cls: 0.1971  loss_rpn_loc: 0.2059    time: 2.1221  last_time: 1.3430  data_tim
[11/11 22:01:21 d2.utils.events]:  eta: 0:44:57  iter: 139  total_loss: 1.159  loss_cls: 0.3193  loss_box_reg: 0.2471  loss_rpn_cls: 0.2113  loss_rpn_loc: 0.3417    time: 2.1022  last_time: 1.3939  data_tim
[11/11 22:02:02 d2.utils.events]:  eta: 0:44:12  iter: 159  total_loss: 1.107  loss_cls: 0.2695  loss_box_reg: 0.255  loss_rpn_cls: 0.2339  loss_rpn_loc: 0.269    time: 2.0940  last_time: 1.6294  data_time:
[11/11 22:02:49 d2.utils.events]:  eta: 0:43:40  iter: 179  total_loss: 1.091  loss_cls: 0.2701  loss_box_reg: 0.2088  loss_rpn_cls: 0.206  loss_rpn_loc: 0.2952    time: 2.1218  last_time: 1.7894  data_time
[11/11 22:03:33 d2.utils.events]:  eta: 0:43:49  iter: 199  total_loss: 0.8985  loss_cls: 0.2118  loss_box_reg: 0.2224  loss_rpn_cls: 0.1865  loss_rpn_loc: 0.2643    time: 2.1307  last_time: 1.6979  data_ti
[11/11 22:04:12 d2.utils.events]:  eta: 0:42:52  iter: 219  total_loss: 1.045  loss_cls: 0.2401  loss_box_reg: 0.1883  loss_rpn_cls: 0.1696  loss_rpn_loc: 0.2777    time: 2.1159  last_time: 1.6931  data_tim
[11/11 22:04:53 d2.utils.events]:  eta: 0:42:13  iter: 239  total_loss: 1.043  loss_cls: 0.2789  loss_box_reg: 0.2342  loss_rpn_cls: 0.1817  loss_rpn_loc: 0.2386    time: 2.1099  last_time: 1.6793  data_tim
[11/11 22:05:36 d2.utils.events]:  eta: 0:41:32  iter: 259  total_loss: 0.9977  loss_cls: 0.2717  loss_box_reg: 0.2624  loss_rpn_cls: 0.1773  loss_rpn_loc: 0.2364    time: 2.1102  last_time: 1.5884  data_ti
[11/11 22:06:17 d2.utils.events]:  eta: 0:40:52  iter: 279  total_loss: 0.9007  loss_cls: 0.2625  loss_box_reg: 0.2995  loss_rpn_cls: 0.1686  loss_rpn_loc: 0.1743    time: 2.1063  last_time: 2.0552  data_ti
[11/11 22:07:05 d2.utils.events]:  eta: 0:40:35  iter: 299  total_loss: 1.061  loss_cls: 0.2716  loss_box_reg: 0.312  loss_rpn_cls: 0.1757  loss_rpn_loc: 0.2165    time: 2.1255  last_time: 2.1453  data_time
[11/11 22:07:44 d2.utils.events]:  eta: 0:39:47  iter: 319  total_loss: 1.099  loss_cls: 0.2666  loss_box_reg: 0.3316  loss_rpn_cls: 0.1616  loss_rpn_loc: 0.258    time: 2.1150  last_time: 2.1518  data_time
[11/11 22:08:26 d2.utils.events]:  eta: 0:39:14  iter: 339  total_loss: 0.7895  loss_cls: 0.1759  loss_box_reg: 0.2521  loss_rpn_cls: 0.1537  loss_rpn_loc: 0.213    time: 2.1156  last_time: 2.5653  data_tim
[11/11 22:09:14 d2.utils.events]:  eta: 0:38:49  iter: 359  total_loss: 0.9343  loss_cls: 0.2964  loss_box_reg: 0.369  loss_rpn_cls: 0.1259  loss_rpn_loc: 0.1829    time: 2.1319  last_time: 2.6413  data_tim
[11/11 22:09:56 d2.utils.events]:  eta: 0:38:12  iter: 379  total_loss: 0.8501  loss_cls: 0.2152  loss_box_reg: 0.2273  loss_rpn_cls: 0.145  loss_rpn_loc: 0.2228    time: 2.1285  last_time: 1.7980  data_tim
[11/11 22:10:37 d2.utils.events]:  eta: 0:37:28  iter: 399  total_loss: 0.9636  loss_cls: 0.2667  loss_box_reg: 0.3844  loss_rpn_cls: 0.126  loss_rpn_loc: 0.2001    time: 2.1254  last_time: 1.5370  data_tim
[11/11 22:11:18 d2.utils.events]:  eta: 0:36:44  iter: 419  total_loss: 0.9925  loss_cls: 0.2428  loss_box_reg: 0.3107  loss_rpn_cls: 0.1323  loss_rpn_loc: 0.1827    time: 2.1222  last_time: 1.9124  data_ti
[11/11 22:12:06 d2.utils.events]:  eta: 0:36:10  iter: 439  total_loss: 1.02  loss_cls: 0.2044  loss_box_reg: 0.2899  loss_rpn_cls: 0.1372  loss_rpn_loc: 0.2331    time: 2.1335  last_time: 2.2892  data_time
[11/11 22:12:50 d2.utils.events]:  eta: 0:35:25  iter: 459  total_loss: 0.8739  loss_cls: 0.2137  loss_box_reg: 0.3273  loss_rpn_cls: 0.1204  loss_rpn_loc: 0.1864    time: 2.1358  last_time: 1.7495  data_ti
[11/11 22:13:36 d2.utils.events]:  eta: 0:34:48  iter: 479  total_loss: 0.9252  loss_cls: 0.2576  loss_box_reg: 0.2901  loss_rpn_cls: 0.1102  loss_rpn_loc: 0.2052    time: 2.1431  last_time: 2.5612  data_ti
[11/11 22:14:19 d2.utils.events]:  eta: 0:34:07  iter: 499  total_loss: 0.9243  loss_cls: 0.2151  loss_box_reg: 0.324  loss_rpn_cls: 0.1311  loss_rpn_loc: 0.197    time: 2.1431  last_time: 1.7906  data_time
```

- Similarly for 700, it converges to 0.9.

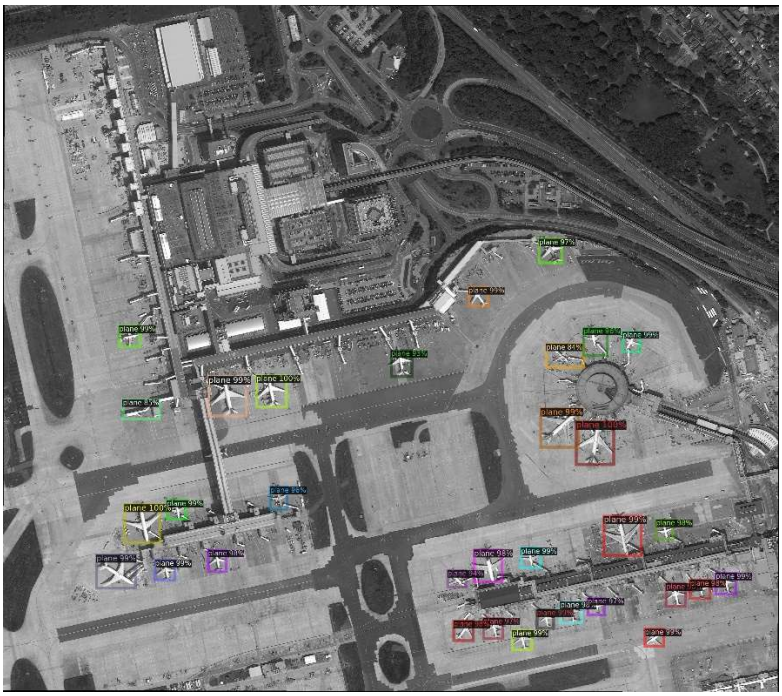
Low number of iterations: 1000

It was still converging, and the results were not promising with 1000 iterations. Loss was still reducing and could be further reduced with more iterations.

Visualization with 1000 iterations



Visualization with 1500 iterations



Part 2

Hyperparameter setting

- Number of epochs: 70
- Batch size: 8
- Learning rate: 0.01

Loss of first few iterations:

```
Epoch: 0, Loss: 0.4026719927787781
Epoch: 1, Loss: 0.2648690938949585
Epoch: 2, Loss: 0.23141880333423615
Epoch: 3, Loss: 0.21204398572444916
Epoch: 4, Loss: 0.19880929589271545
Epoch: 5, Loss: 0.18895624577999115
```

Loss of last few iterations:

```
Epoch: 64, Loss: 0.07312179356813431
Epoch: 65, Loss: 0.07243578881025314
Epoch: 66, Loss: 0.07232359796762466
Epoch: 67, Loss: 0.07169823348522186
Epoch: 68, Loss: 0.0712670236825943
Epoch: 69, Loss: 0.07036665081977844
```

Final Architecture:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 8, 128, 128]             224
       BatchNorm2d-2          [-1, 8, 128, 128]              16
              ReLU-3          [-1, 8, 128, 128]               0
              conv-4          [-1, 8, 128, 128]               0
            Conv2d-5         [-1, 16, 128, 128]           1,168
       BatchNorm2d-6         [-1, 16, 128, 128]              32
              ReLU-7         [-1, 16, 128, 128]               0
              conv-8         [-1, 16, 128, 128]               0
         MaxPool2d-9           [-1, 16, 64, 64]               0
             down-10           [-1, 16, 64, 64]               0
           Conv2d-11           [-1, 32, 64, 64]           4,640
      BatchNorm2d-12           [-1, 32, 64, 64]              64
             ReLU-13           [-1, 32, 64, 64]               0
             conv-14           [-1, 32, 64, 64]               0
```

```
         MaxPool2d-15          [-1, 32, 32, 32]               0
               down-16          [-1, 32, 32, 32]               0
              Conv2d-17          [-1, 64, 32, 32]          18,496
         BatchNorm2d-18          [-1, 64, 32, 32]             128
                ReLU-19          [-1, 64, 32, 32]               0
                conv-20          [-1, 64, 32, 32]               0
         MaxPool2d-21          [-1, 64, 16, 16]               0
               down-22          [-1, 64, 16, 16]               0
              Conv2d-23         [-1, 128, 16, 16]          73,856
         BatchNorm2d-24         [-1, 128, 16, 16]             256
                ReLU-25         [-1, 128, 16, 16]               0
                conv-26         [-1, 128, 16, 16]               0
         MaxPool2d-27           [-1, 128, 8, 8]               0
               down-28           [-1, 128, 8, 8]               0
              Conv2d-29           [-1, 256, 8, 8]         295,168
         BatchNorm2d-30           [-1, 256, 8, 8]             512
                ReLU-31           [-1, 256, 8, 8]               0
                conv-32           [-1, 256, 8, 8]               0
         MaxPool2d-33           [-1, 256, 4, 4]               0
               down-34           [-1, 256, 4, 4]               0
              Conv2d-35           [-1, 512, 4, 4]       1,180,160
         BatchNorm2d-36           [-1, 512, 4, 4]           1,024
                ReLU-37           [-1, 512, 4, 4]               0
                conv-38           [-1, 512, 4, 4]               0
         MaxPool2d-39           [-1, 512, 2, 2]               0
               down-40           [-1, 512, 2, 2]               0
  ConvTranspose2d-41           [-1, 512, 4, 4]       1,049,088
              Conv2d-42           [-1, 256, 4, 4]       1,179,904
         BatchNorm2d-43           [-1, 256, 4, 4]             512
                ReLU-44           [-1, 256, 4, 4]               0
                conv-45           [-1, 256, 4, 4]               0
                  up-46           [-1, 256, 4, 4]               0
  ConvTranspose2d-47           [-1, 256, 8, 8]         262,400
              Conv2d-48           [-1, 128, 8, 8]         295,040
         BatchNorm2d-49           [-1, 128, 8, 8]             256
                ReLU-50           [-1, 128, 8, 8]               0
                conv-51           [-1, 128, 8, 8]               0
                  up-52           [-1, 128, 8, 8]               0
  ConvTranspose2d-53         [-1, 128, 16, 16]          65,664
              Conv2d-54          [-1, 64, 16, 16]          73,792
         BatchNorm2d-55          [-1, 64, 16, 16]             128
                ReLU-56          [-1, 64, 16, 16]               0
                conv-57          [-1, 64, 16, 16]               0
                  up-58          [-1, 64, 16, 16]               0
  ConvTranspose2d-59          [-1, 64, 32, 32]          16,448
              Conv2d-60          [-1, 32, 32, 32]          18,464
         BatchNorm2d-61          [-1, 32, 32, 32]              64
                ReLU-62          [-1, 32, 32, 32]               0
                conv-63          [-1, 32, 32, 32]               0
                  up-64          [-1, 32, 32, 32]               0
  ConvTranspose2d-65          [-1, 32, 64, 64]           4,128
              Conv2d-66          [-1, 16, 64, 64]           4,624
         BatchNorm2d-67          [-1, 16, 64, 64]              32
                ReLU-68          [-1, 16, 64, 64]               0
                conv-69          [-1, 16, 64, 64]               0
                  up-70          [-1, 16, 64, 64]               0
  ConvTranspose2d-71        [-1, 16, 128, 128]           1,040
```

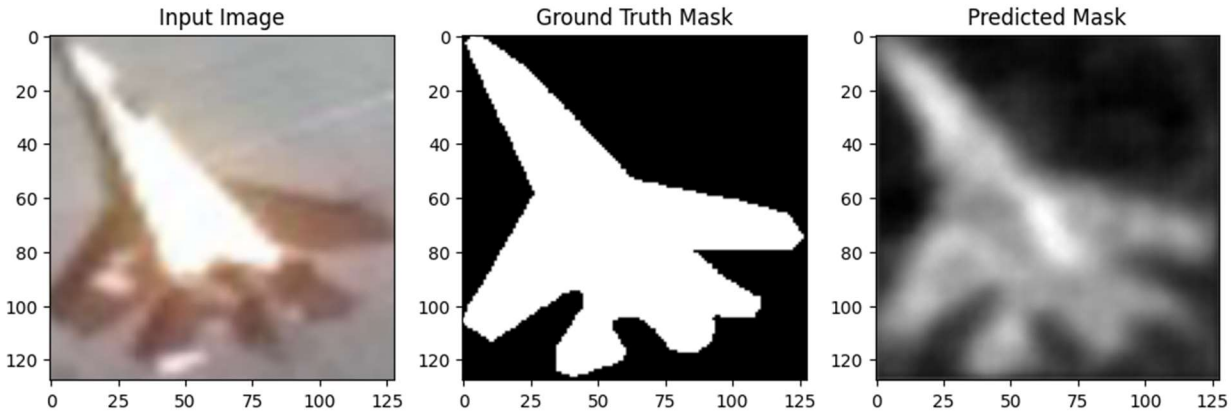
```
           Conv2d-72          [-1, 4, 128, 128]             580
      BatchNorm2d-73          [-1, 4, 128, 128]               8
             ReLU-74          [-1, 4, 128, 128]               0
             conv-75          [-1, 4, 128, 128]               0
               up-76          [-1, 4, 128, 128]               0
           Conv2d-77          [-1, 1, 128, 128]              37
             conv-78          [-1, 1, 128, 128]               0
================================================================
Total params: 4,547,953
Trainable params: 4,547,953
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 33.25
Params size (MB): 17.35
Estimated Total Size (MB): 50.79
----------------------------------------------------------------
```

Final Mean IOU: 0.86

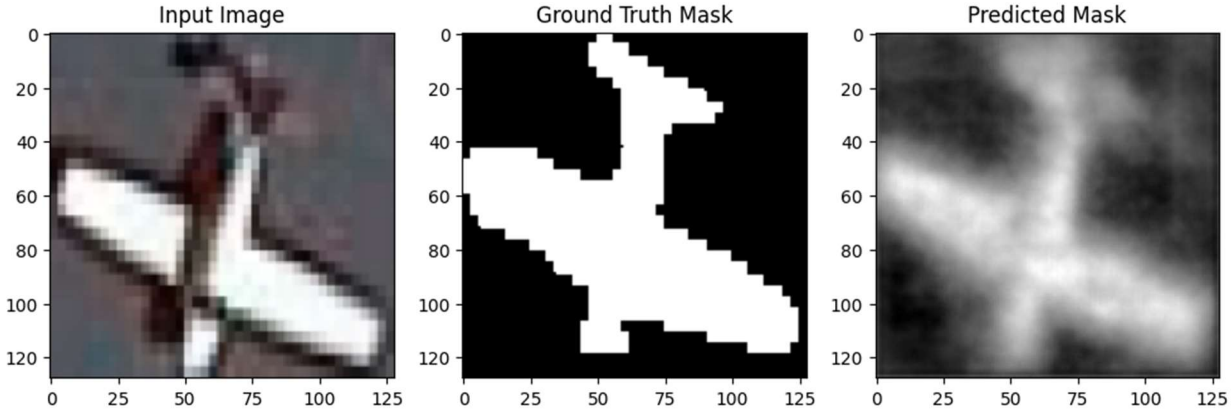
```
#images: 7980, Mean IoU: 0.9057676527897925
```

Test plane image and predicted masks:

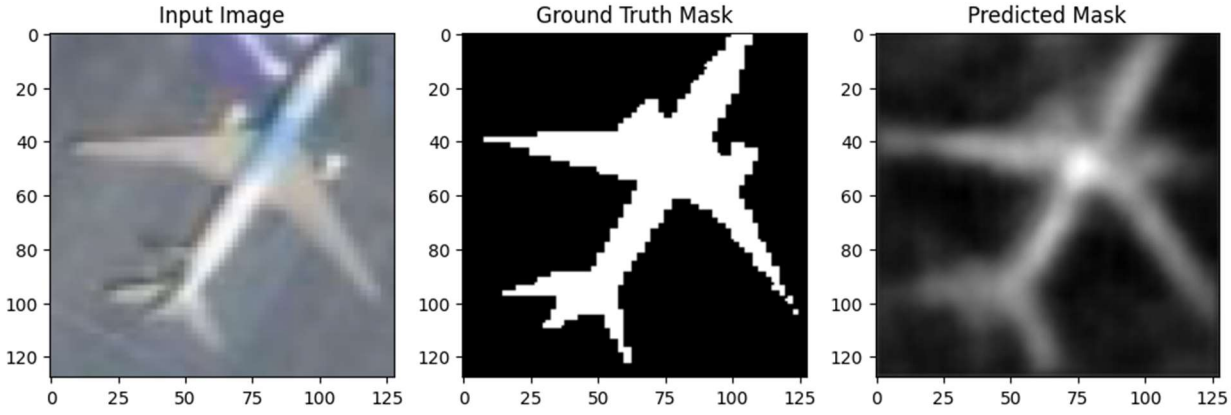
Sample 1



Sample 2



Sample 3



Part3

Group Member: Jashanraj Singh Gosain (301435386) and Archit Verma (301401441)

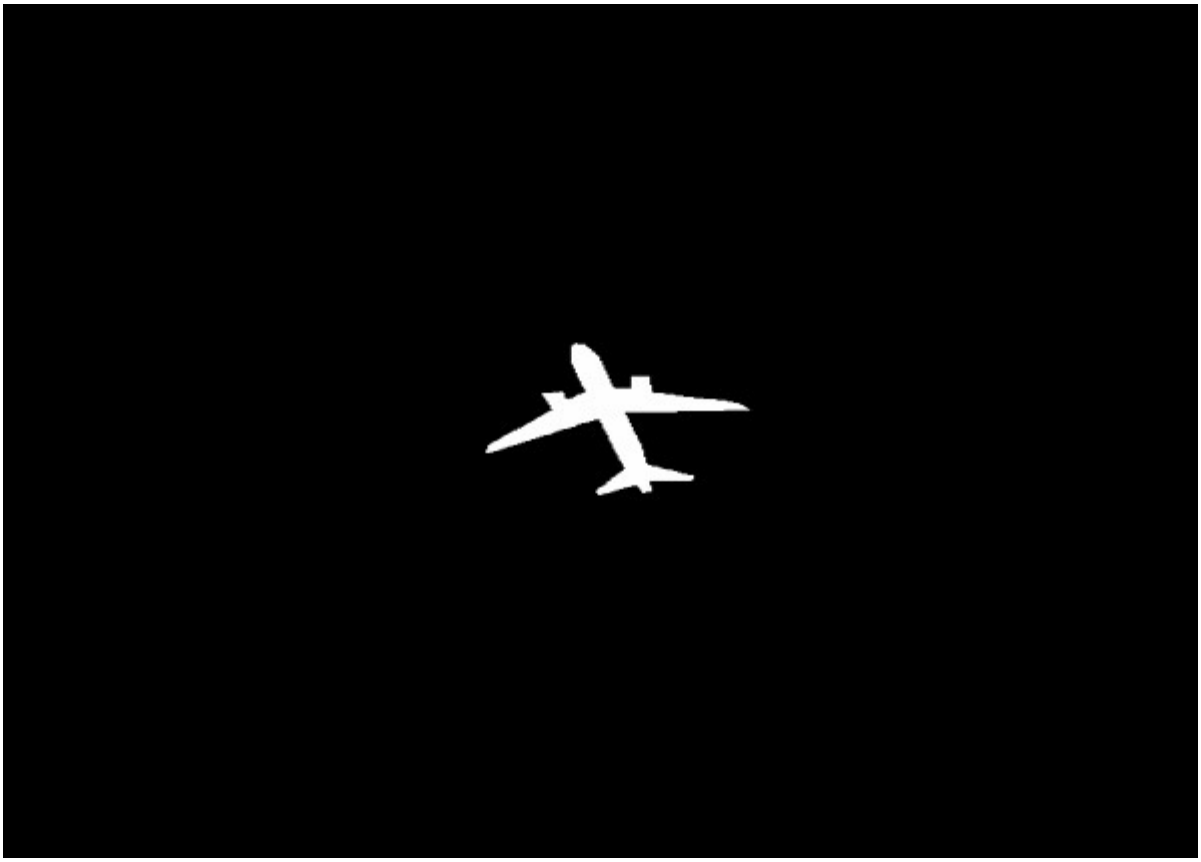
Kaggle Name : Jashanraj Singh Gosain, Archit

Score: 0.38

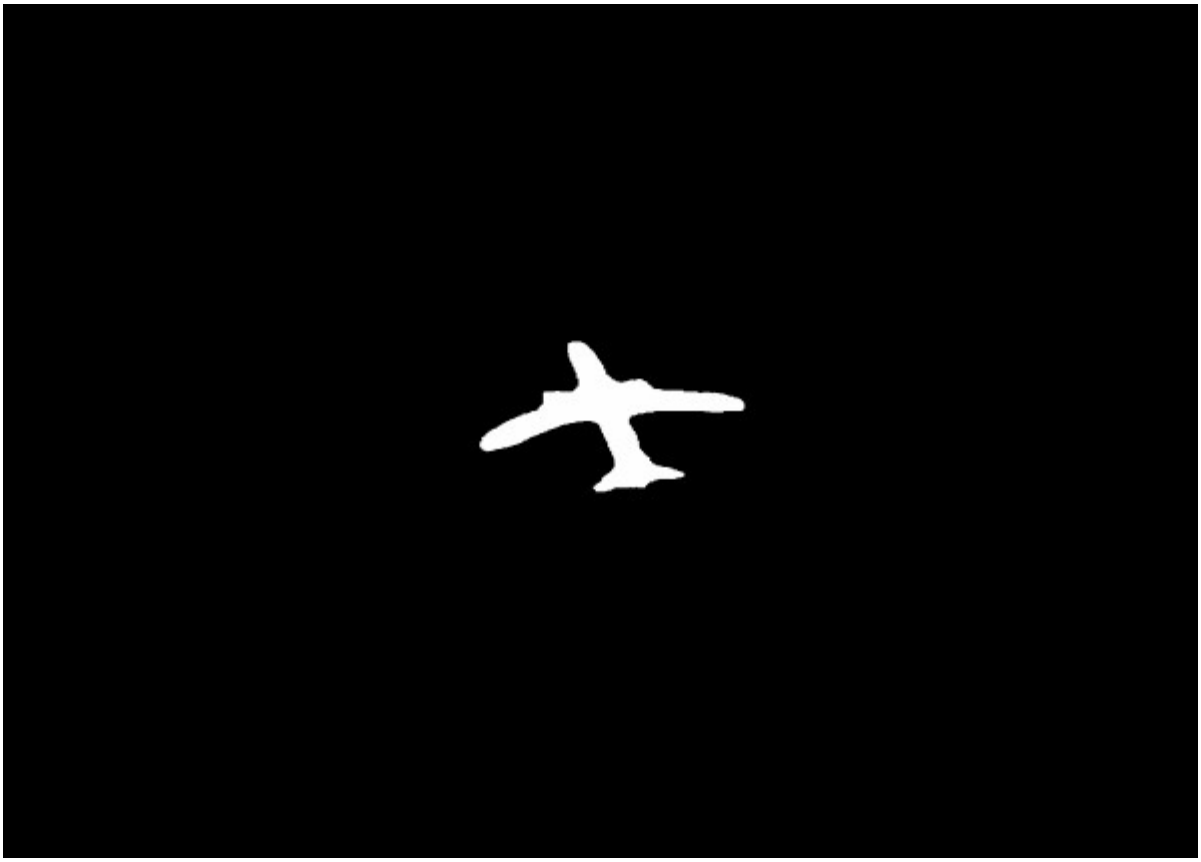
Sample1

Image

Ground Truth



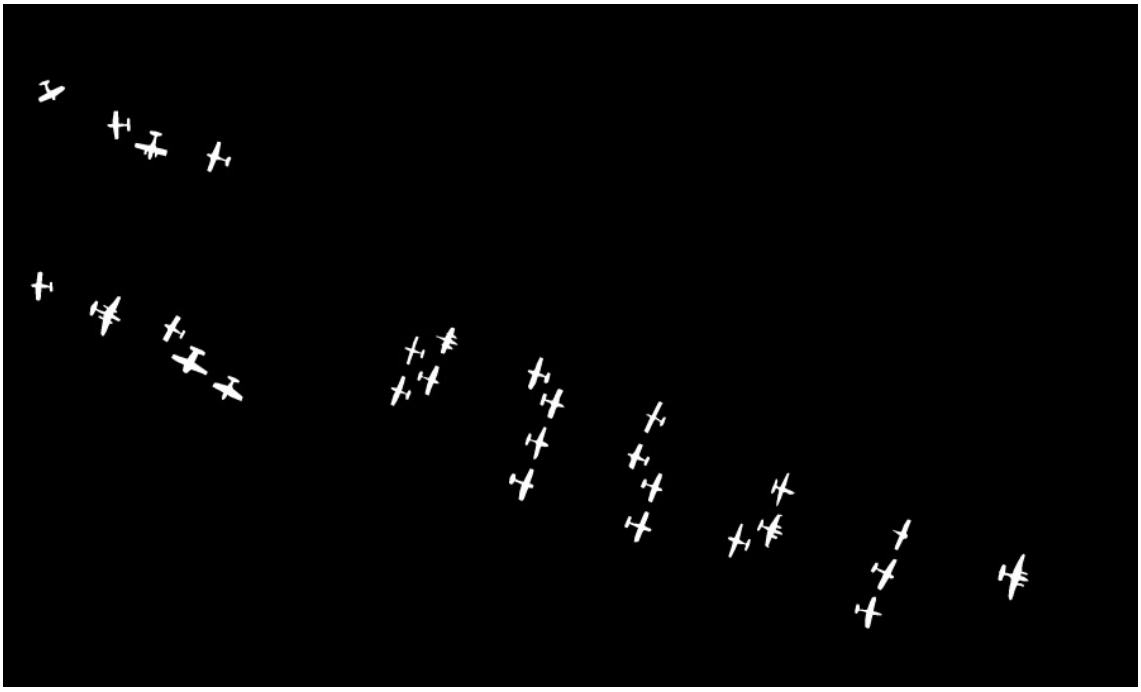
Prediction



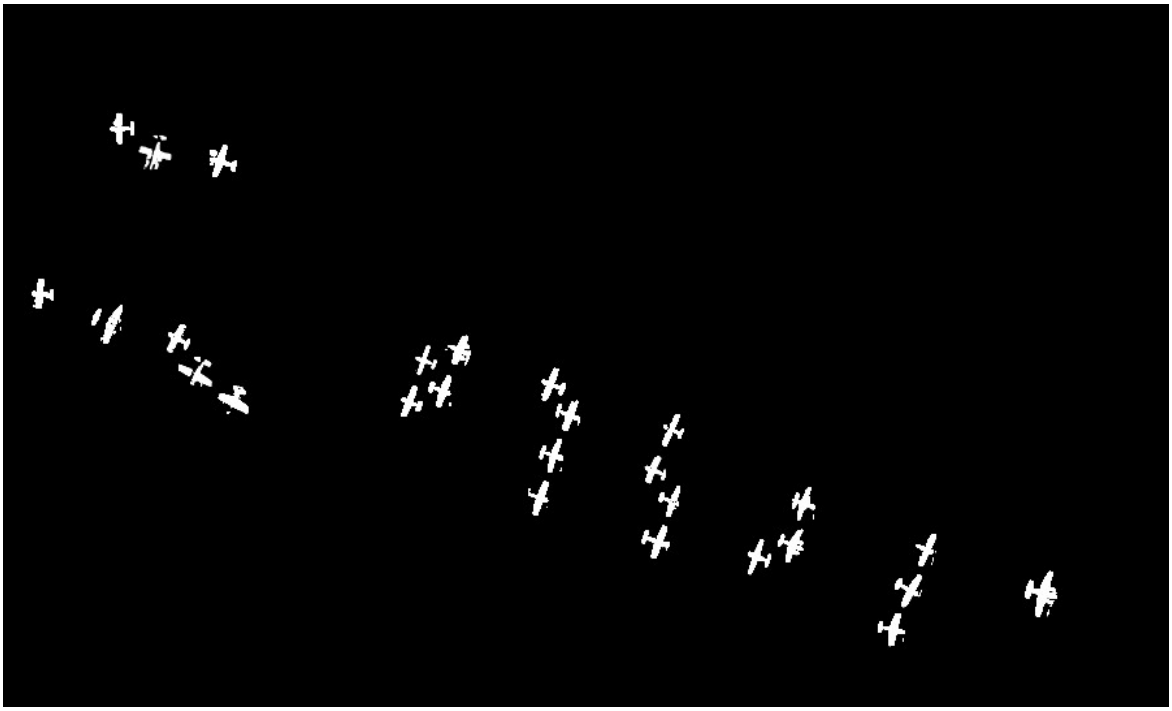
Sample 2

Image

Ground Truth

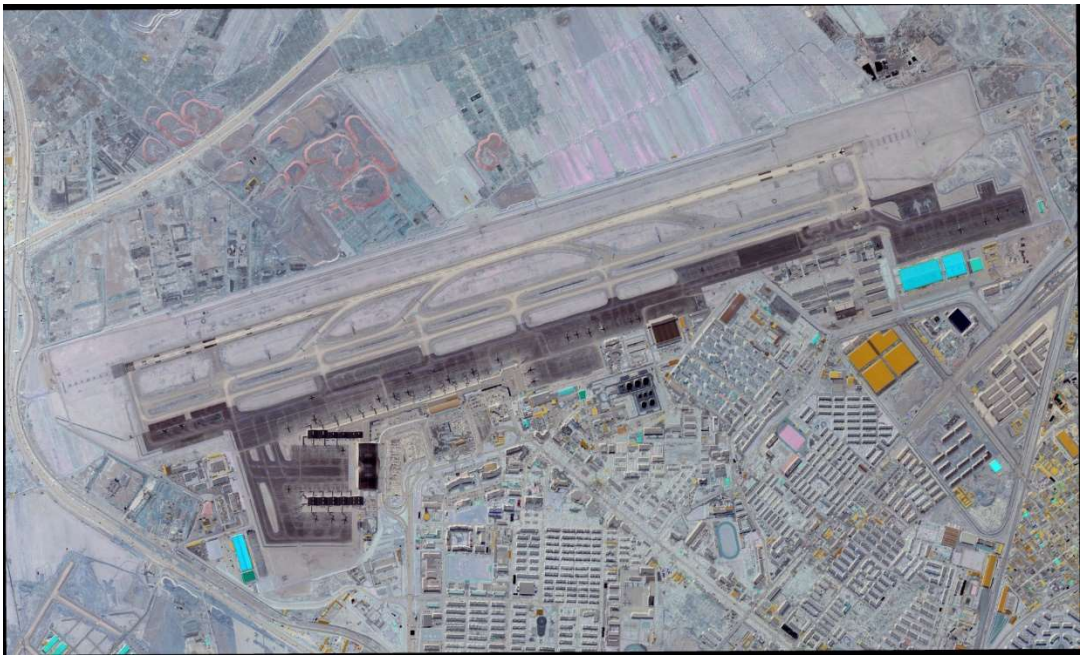


Prediction



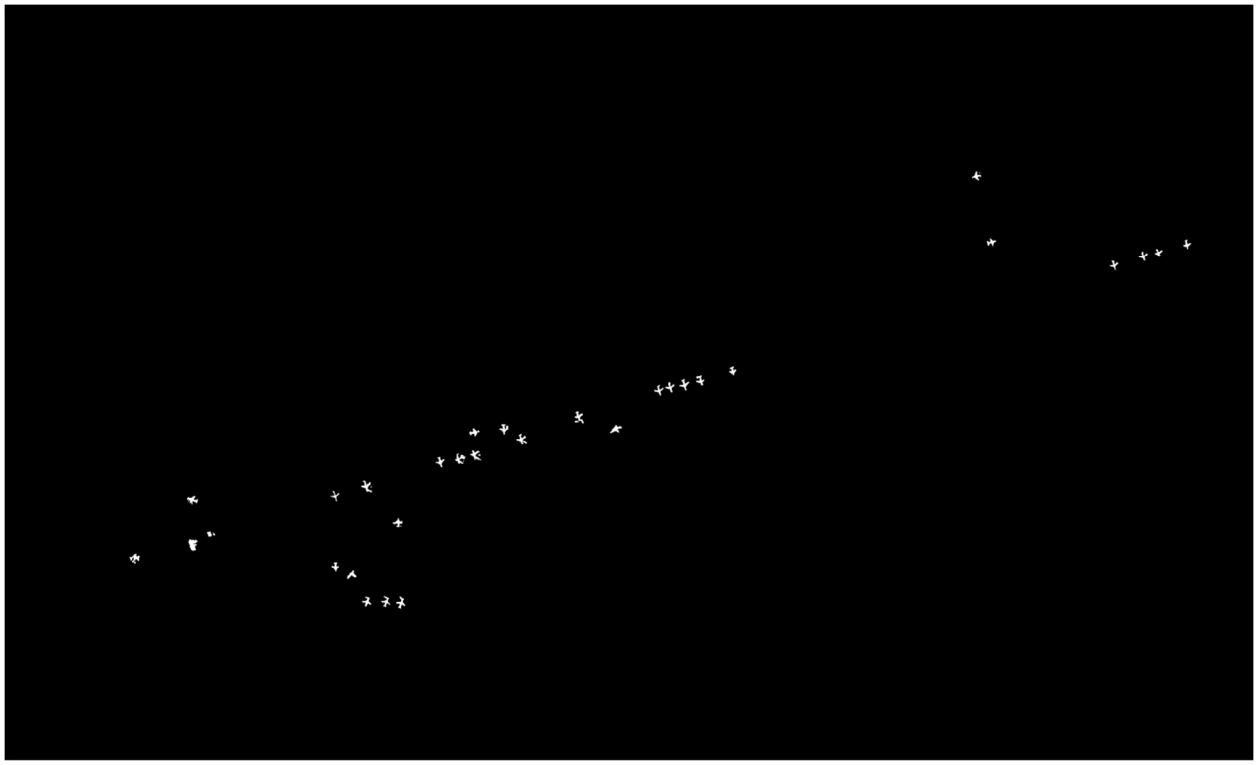
Sample 3

Image



Ground Truth

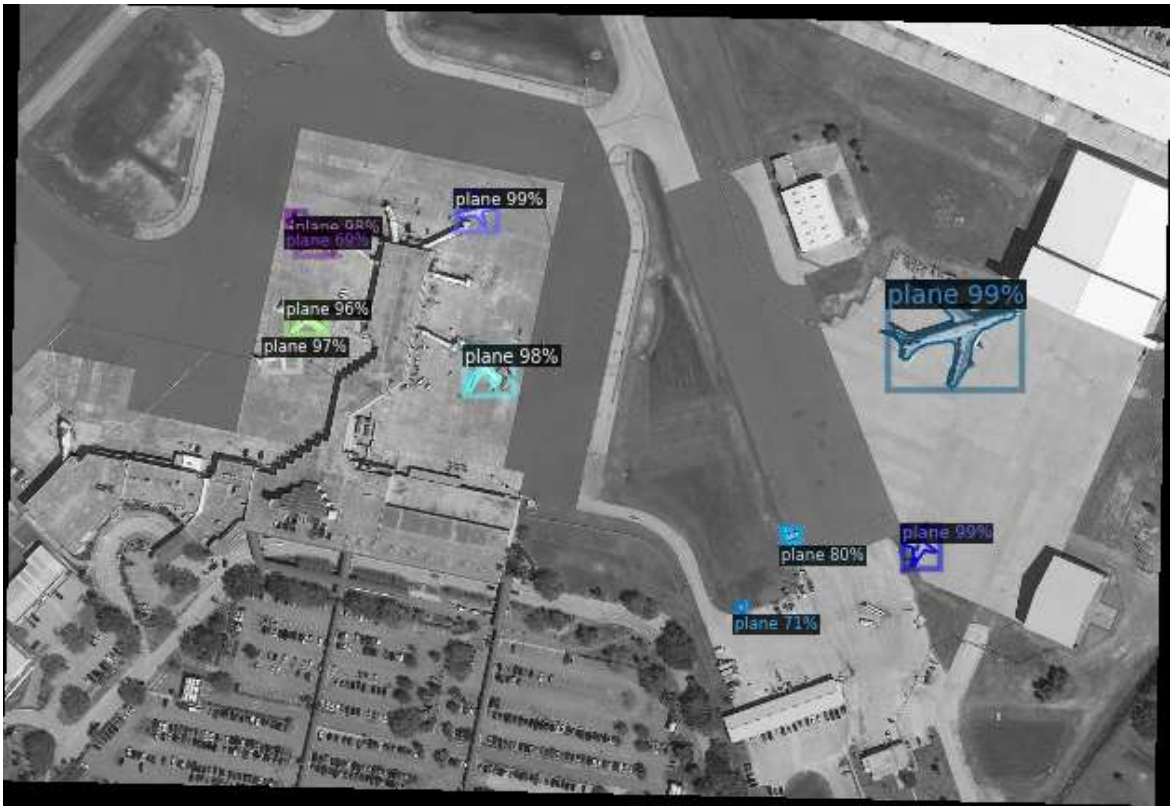
Prediction



Part 4

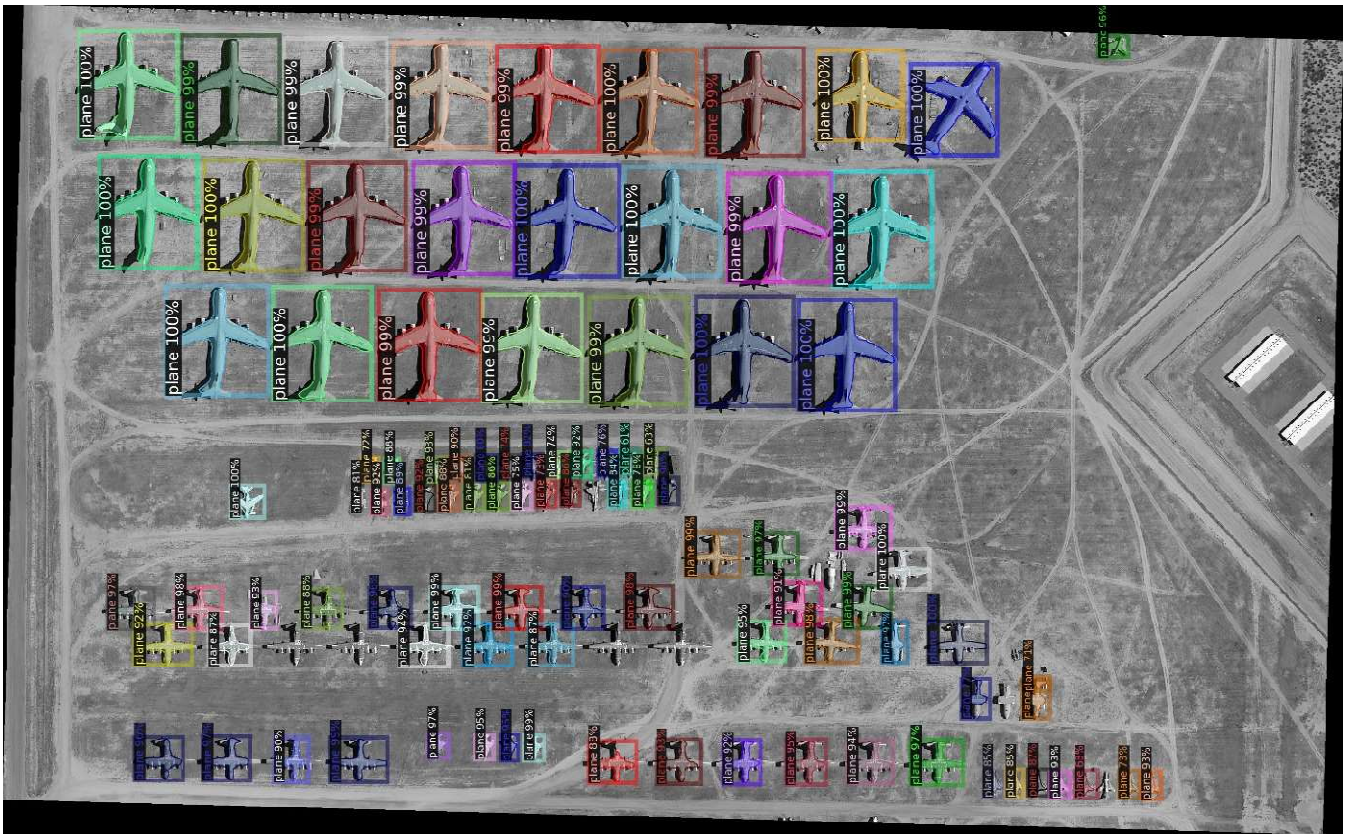
Visualization of test samples

Sample 1



Sample 2

Sample 3



Evaluation:

AP for bbox is 32 in part 4 which is lower than AP for bbox in part 2.

AP50 (**51**) ,AP75 (**35**), Aps (**21**), APm (**41**) and Apl (**62**) for Part 2 have higher value than it's corresponding values in part 4 (AP50 – 56, AP75 – 41, Aps – 26, APm – 45, Apl – 68) .

"mask_rcnn_R_50_FPN_3x.yaml":

- It is computationally efficient as compared to "faster_rcnn_X_101_32x8d_FPN_3x.yaml".
- It provides good results but with increased computational efficiency it's accuracy decreases.

"faster_rcnn_X_101_32x8d_FPN_3x.yaml"

- It is useful to calculate achieve high accuracy
- It's computational cost is higher than "mask_rcnn_R_50_FPN_3x.yaml". so, it will need more time and resources to train.

```
[11/12 02:23:17 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 32.011 | 51.126 | 35.685 | 21.095 | 41.038 | 62.045 |
Loading and preparing results...
DONE (t=0.04s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *segm*
DONE (t=9.94s).
Accumulating evaluation results...
DONE (t=0.11s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.091
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.300
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.025
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.047
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.101
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.348
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.008
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.058
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.121
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.062
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.145
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.401
[11/12 02:23:27 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:-----:|:------:|:------:|:-----:|:------:|:------:|
| 9.113 | 29.981 | 2.462  | 4.719 | 10.124 | 34.849 |
OrderedDict([('bbox', {'AP': 32.01118799298497, 'AP50': 51.12643736491969, 'AP75': 35.685389168440565,
```