CMPT 412
Project 3

Object Detection, Semantic Segmentation, and Instance Segmentation

Late Days Used: 2

Submitted by: Paolo Sy-Quia

Part 1

Configs and Modifications

Initial Config

```
# THIS IS THE BAD CONFIG

cfg = get_cfg()
cfg.OUTPUT_DIR = "{}/output/badconfig/".format(BASE_DIR)

cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_101_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_R_101_FPN_3x.yaml")
cfg.SOLVER.BATCH_SIZE_PER_IMAGE = 128

cfg.SOLVER.IMS_PER_BATCH = 2  # batch size
cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
cfg.SOLVER.MAX_ITER = 500  # and a good number of iterations
cfg.SOLVER.STEPS = []        # do not decay learning rate
```

Final Config

```
# THIS IS THE FINAL CONFIG

cfg = get_cfg()
cfg.OUTPUT_DIR = "{}/output/".format(BASE_DIR)

cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_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_R_50_FPN_3x.yaml")
cfg.SOLVER.BATCH_SIZE_PER_IMAGE = 512
cfg.SOLVER.IMS_PER_BATCH = 4  # batch size
cfg.SOLVER.BASE_LR = 0.0005  # pick a good LR
cfg.SOLVER.MAX_ITER = 1000  # and a good number of iterations
cfg.SOLVER.STEPS = []        # do not decay learning rate
```

List of Modifications

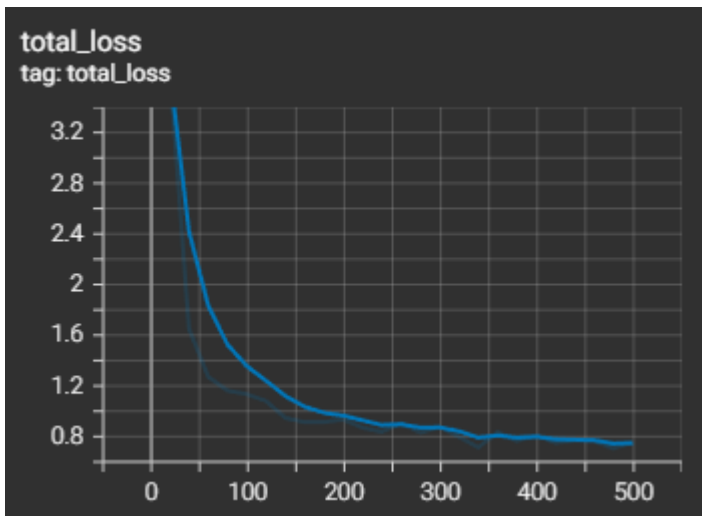
- Changed model from `faster_rcnn_R_101_FP` to `faster_rcnn_R_50_FPN`
- Increased `BATCH_SIZE_PER_IMAGE` to `512`
- Increased `IMS_PER_BATCH` to `4`
- Increased `BASE_LR` to `0.0005`
- Increased `MAX_ITER` to `1000`

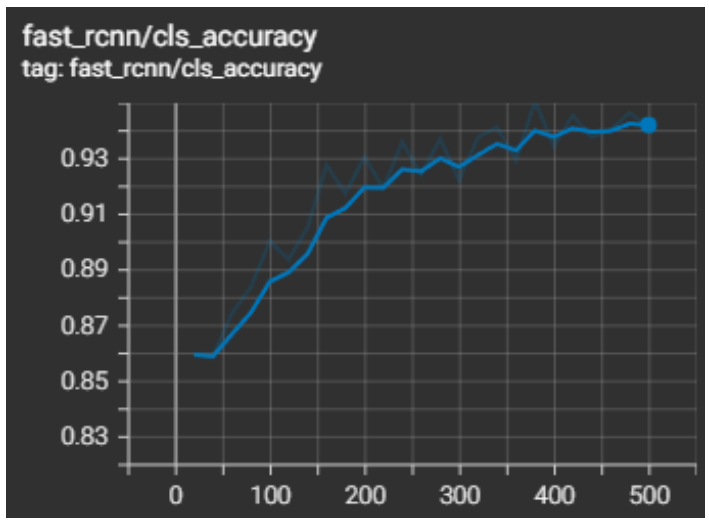
Factors That Improved Performance

- Increasing BATCH_SIZE_PER_IMAGE
    - Allows us to sample more RPN proposals when training
    - Increases the likelihood of getting better proposals
- Increasing IMS_PER_BATCH
    - Allows us to train with much more images per iteration (`config.NUM_GPUS * config.IMS_PER_BATCH images/iter`)
    - Using more images per iteration allows our model to learn more in less iterations
- Increasing BASE_LR
    - Through trial and error, I found that LR=0.0005 led to the best AP50 accuracy

Training Loss and Accuracy Plots

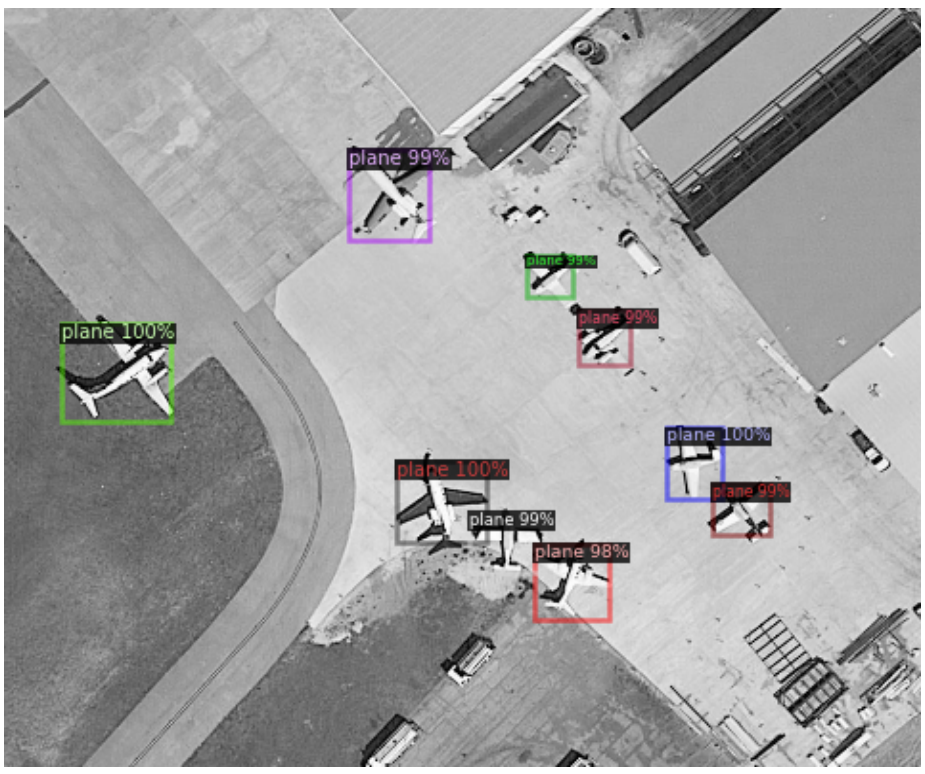
```
[11/06 21:17:23 d2.evaluation.coco_evaluation]: Evaluation resu
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 28.190 | 46.352 | 31.562 | 17.312 | 36.915 | 58.232 |
OrderedDict([('bbox', {'AP': 28.190410724730274, 'AP50': 46.352
```

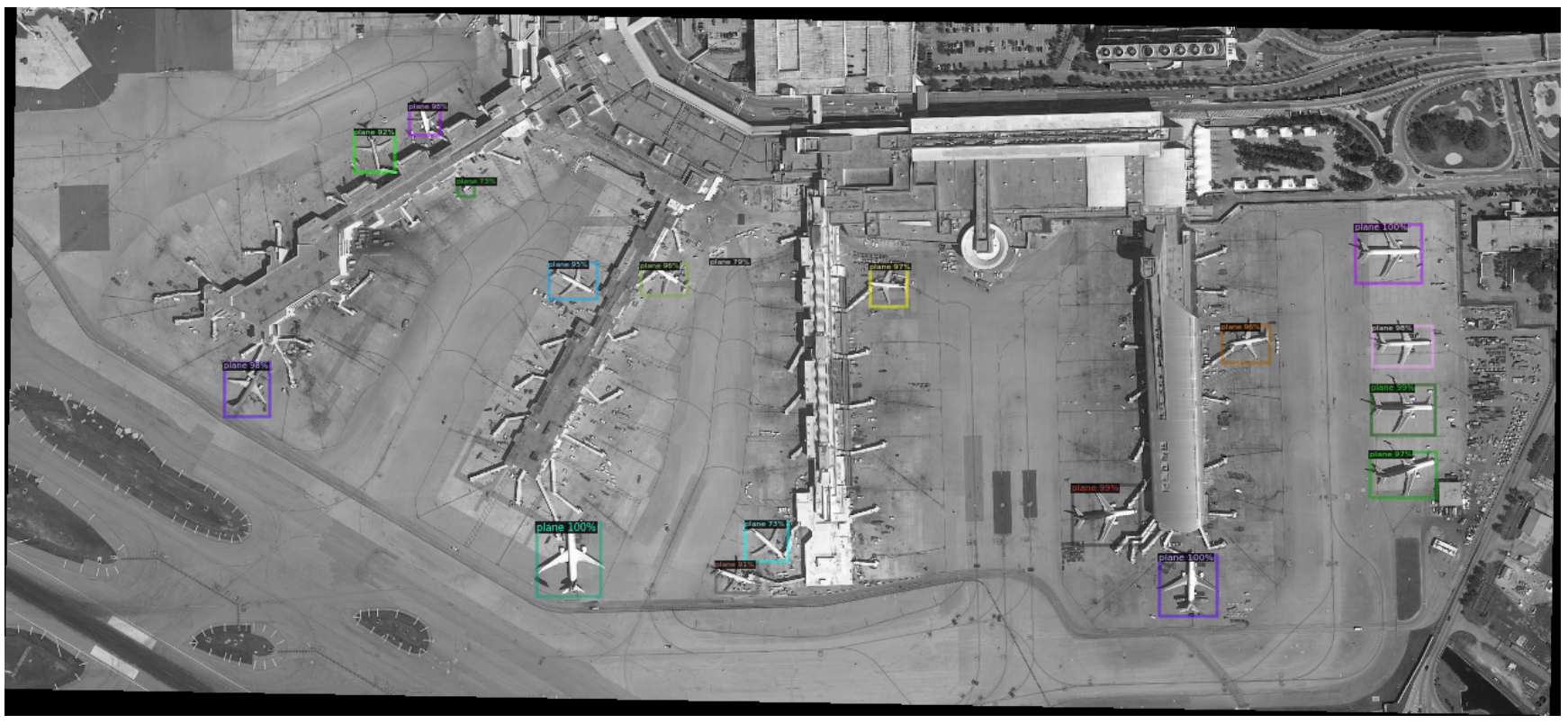




Visualization

Test set predictions





Ablation Study

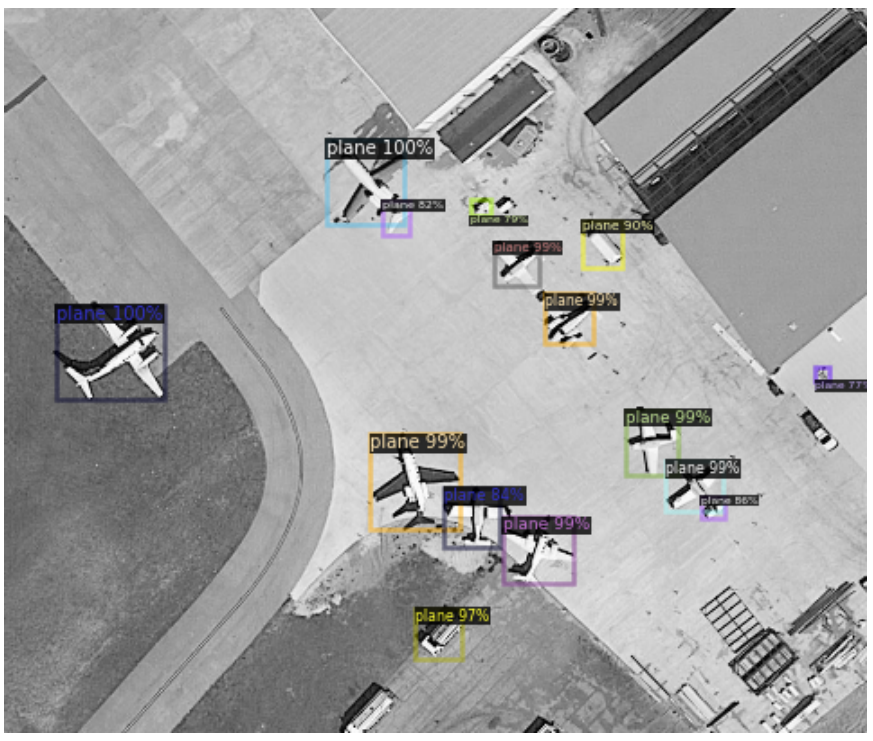
Config #1 (initial config)

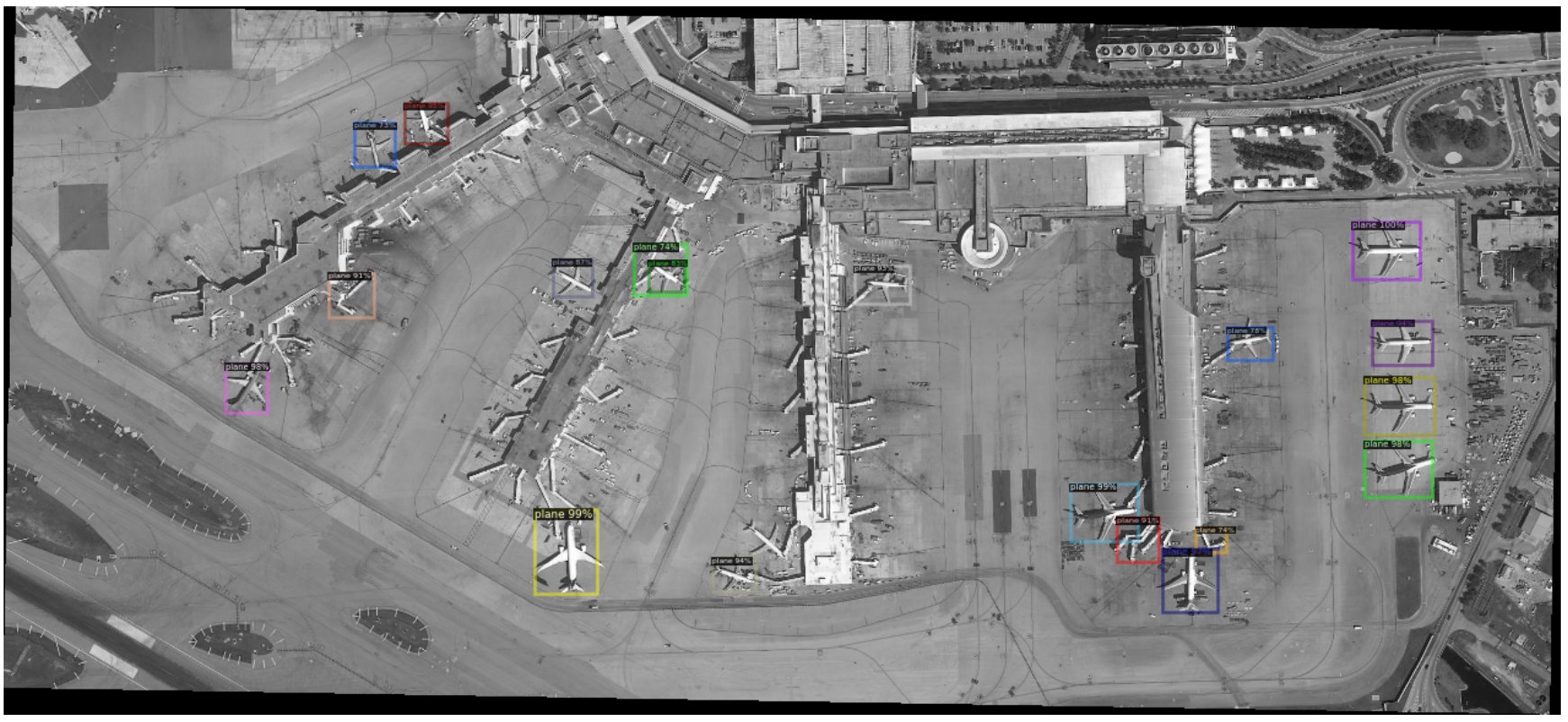
- See Configs and Modifications section above

AP50 Accuracy

```
[11/06 22:32:25 d2.evaluation.coco_evaluation]: Evaluation results
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 19.591 | 33.203 | 20.786 | 10.727 | 25.372 | 52.012 |
```

Sample Visualization





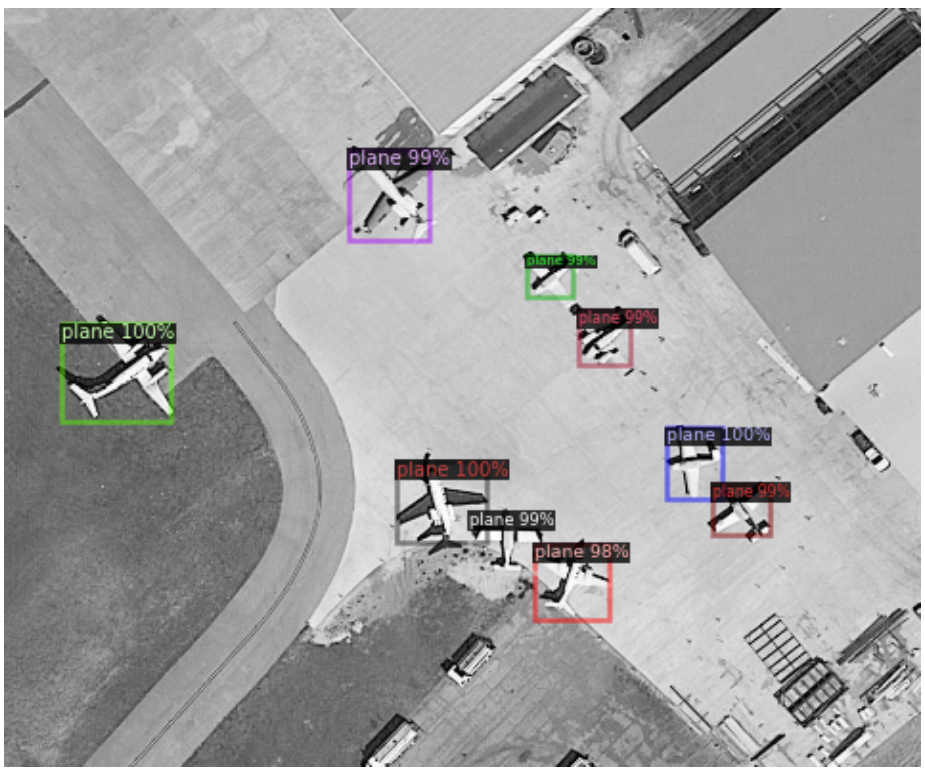
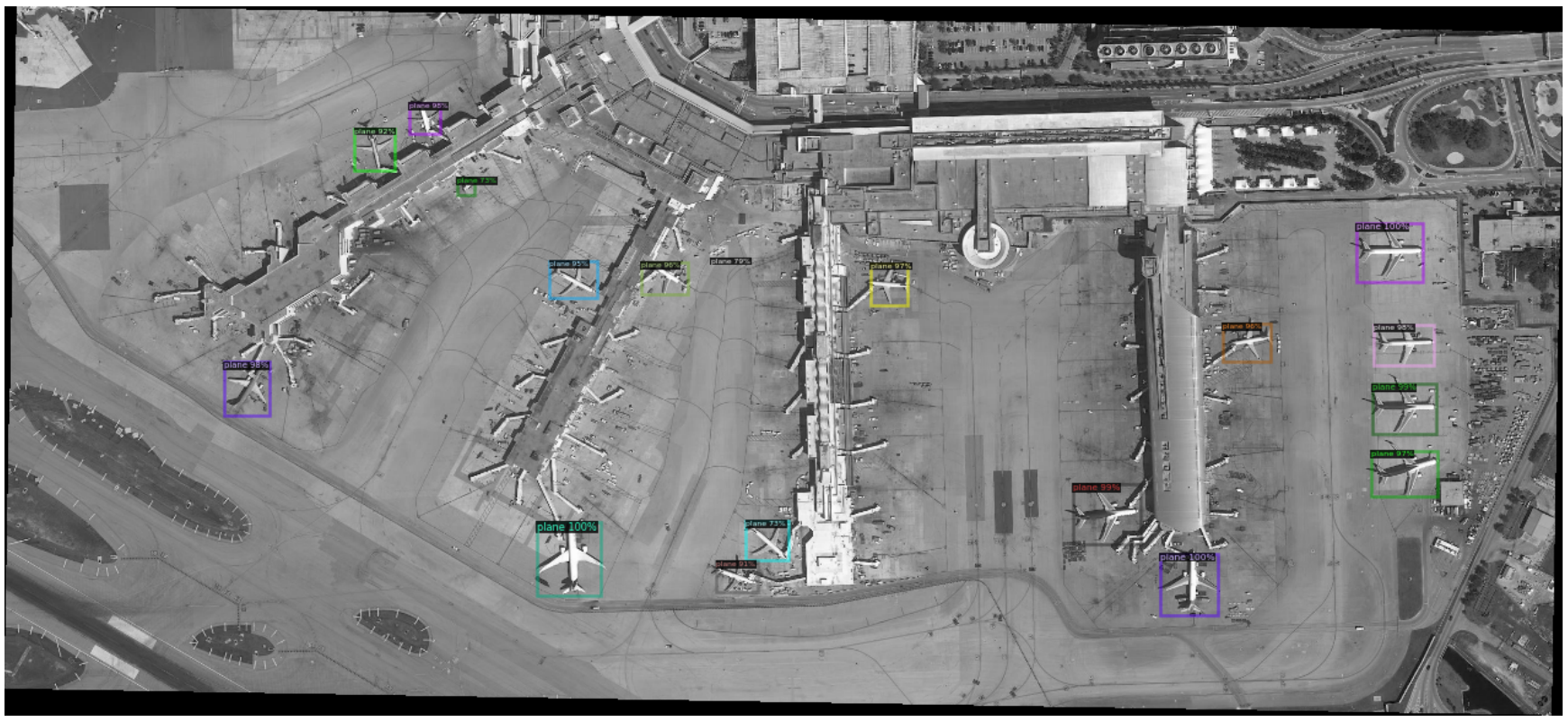
Config #2 (final config)

- See Configs and Modifications section above

AP50 Accuracy

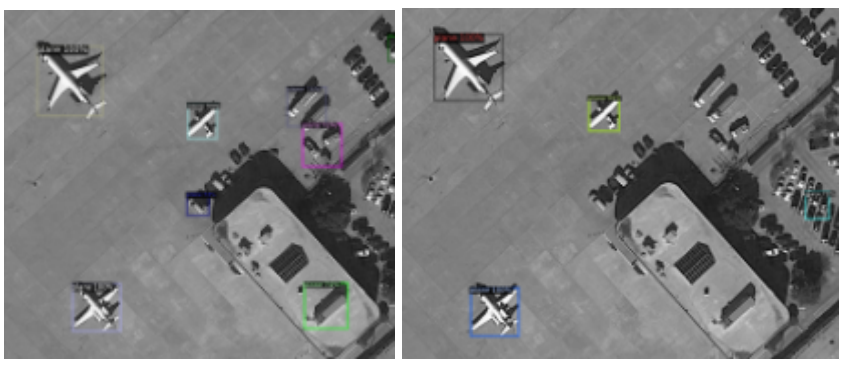
```
[11/06 21:17:23 d2.evaluation.coco_evaluation]: Evaluation resu
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 28.190 | 46.352 | 31.562 | 17.312 | 36.915 | 58.232 |
```

Sample Visualization

Summary

- AP50 improvement from 33.205 to 46.352
- Less False Positives



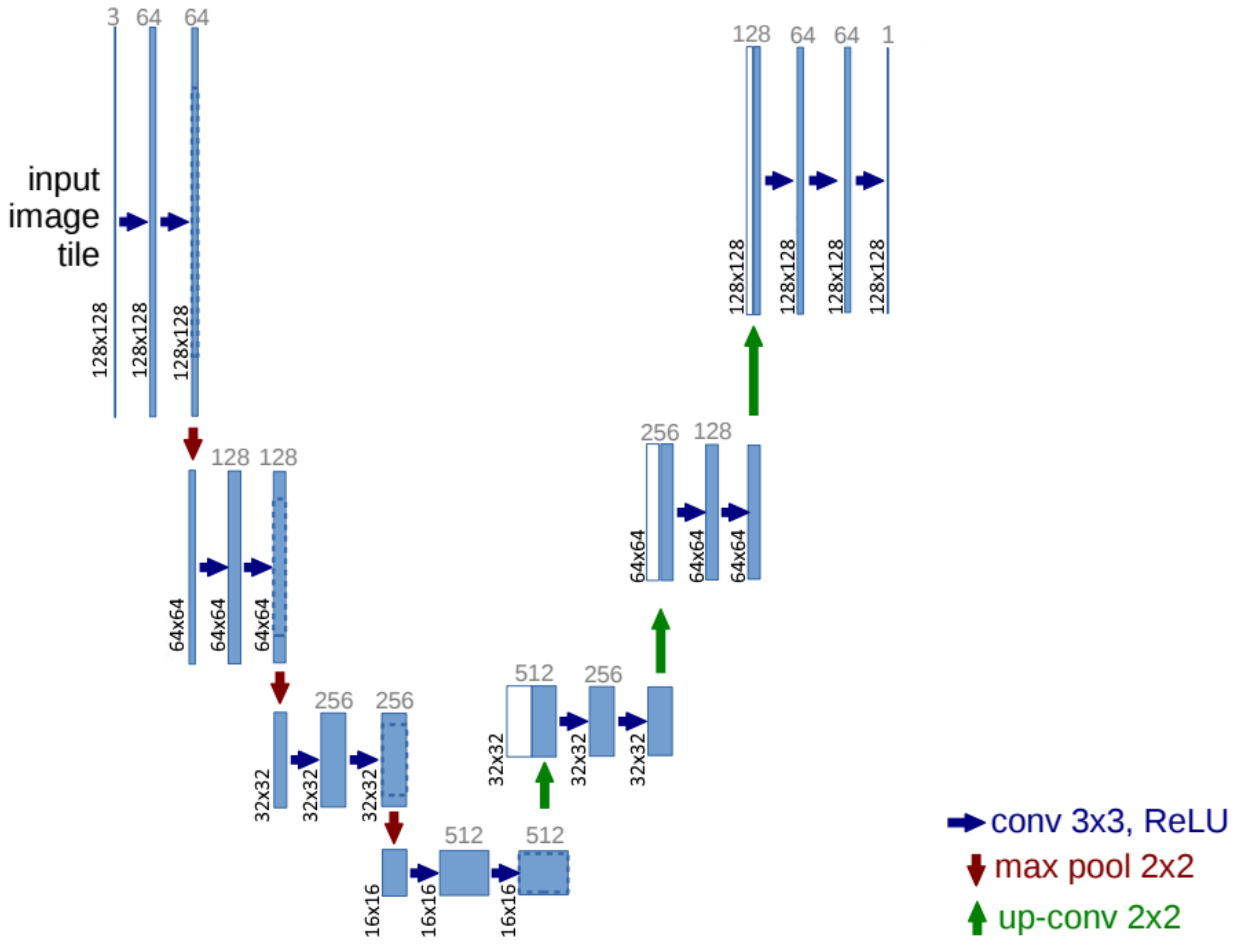
Part 2

Hyperparameters

- `num_epochs = 5`
- `batch_size = 4`
- `learning_rate = 0.02`
- `weight_decay = 1e-5`

Network

- The network is based on the U-Net model. It was heavily modified for this assignment to be simpler to speed up training.
- Changes to U-Net included
    - Reducing input image size to 128
    - Reducing the total number of Conv layers
    - Reducing the output channels of some layers
    - Removing the copy and crop operation



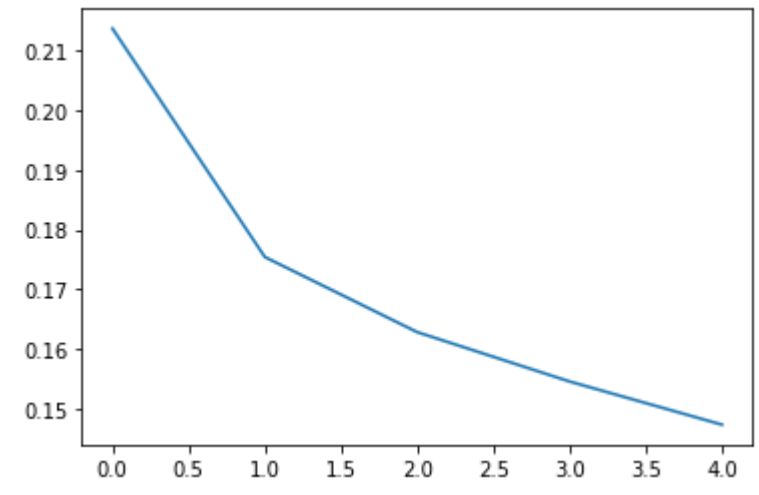
In Code

```
# Encoder

self.conv1 = conv(3, 64)
self.down1 = down(64, 64)
# Output size = 128//2 x 128//2 = 64 x 64
self.conv2 = conv(64, 128)
self.down2 = down(128, 128)
# Output size = 64//2 x 64//2 = 32 x 32
self.conv3 = conv(128, 256)
self.down3 = down(256, 256)
# Output size = 32//2 x 32//2 = 16 x 16
self.conv4 = conv(256, 512)
self.conv5 = conv(512, 512)
```

```
# Decoder

self.up1 = up(512, 512)
# Output size = 16*2 x 16*2 = 32 x 32
self.conv6 = conv(512, 256)
self.up2 = up(256, 256)
# Output size = 32*2 x 32*2 = 64 x 64
self.conv7 = conv(256, 128)
self.up3 = up(128, 128)
# Output size = 64*2 x 64*2 = 128 x 128
self.conv8 = conv(128, 64)
self.conv9 = conv(64, 64)
self.conv10 = conv(64, 64)
self.output_conv = conv(64, 1, False) # ReLu activa
```

Training Loss

```
100%|██████████| 1596/1596 [08:26<00:00, 3.17it/s]
Epoch: 0, Loss: 0.21362344920635223
100%|██████████| 1596/1596 [08:27<00:00, 3.17it/s]
Epoch: 1, Loss: 0.17539161443710327
100%|██████████| 1596/1596 [08:26<00:00, 3.19it/s]
Epoch: 2, Loss: 0.16282221674919128
100%|██████████| 1596/1596 [08:26<00:00, 3.16it/s]
Epoch: 3, Loss: 0.15456324815750122
100%|██████████| 1596/1596 [08:28<00:00, 3.16it/s]
Epoch: 4, Loss: 0.14737963676452637
```



Intersection of Union (IoU)

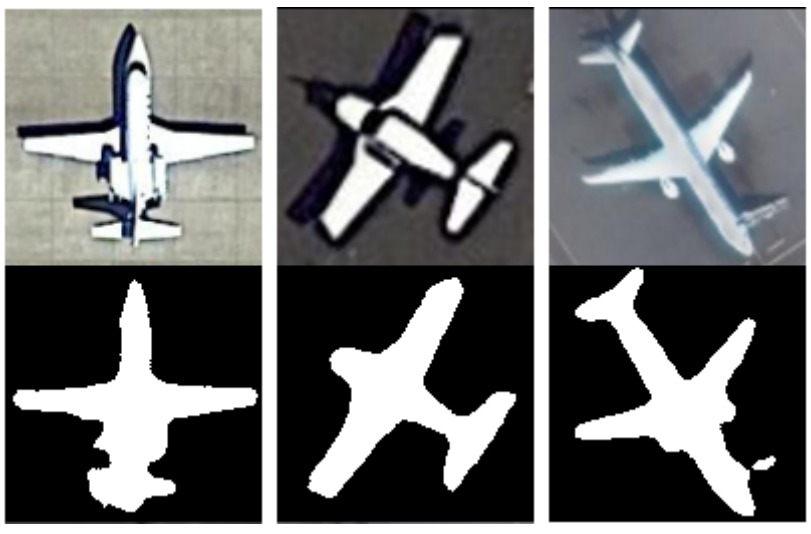
```
# calc IoU
inter = cv2.bitwise_and(a_thresh, b)
union = cv2.bitwise_or(a_thresh, b)
inter_c = cv2.countNonZero(inter)
union_c = cv2.countNonZero(union)
iou = inter_c/union_c
```

```
iter 1000, curr mean IoU: 0.8092125586659898
iter 2000, curr mean IoU: 0.807713742485899
iter 3000, curr mean IoU: 0.8046617101603795
iter 4000, curr mean IoU: 0.805206099385606
iter 5000, curr mean IoU: 0.8049648773877132
iter 6000, curr mean IoU: 0.8047193949801157

 #images: 6384, Mean IoU: 0.8040473510410518
```

Final Mean IoU:

0.8040473510410512

Test Set Predicted Masks



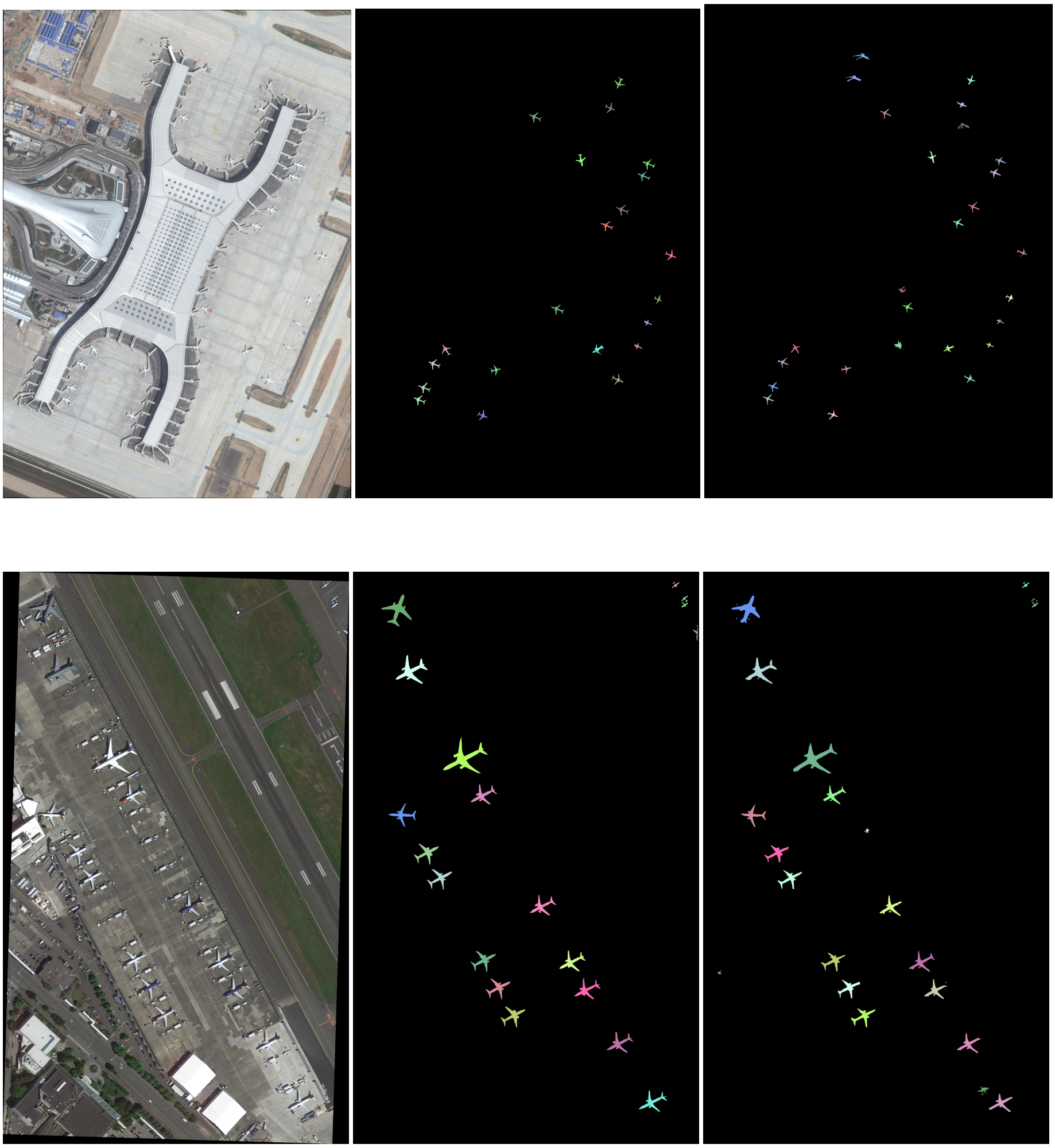
Part 3

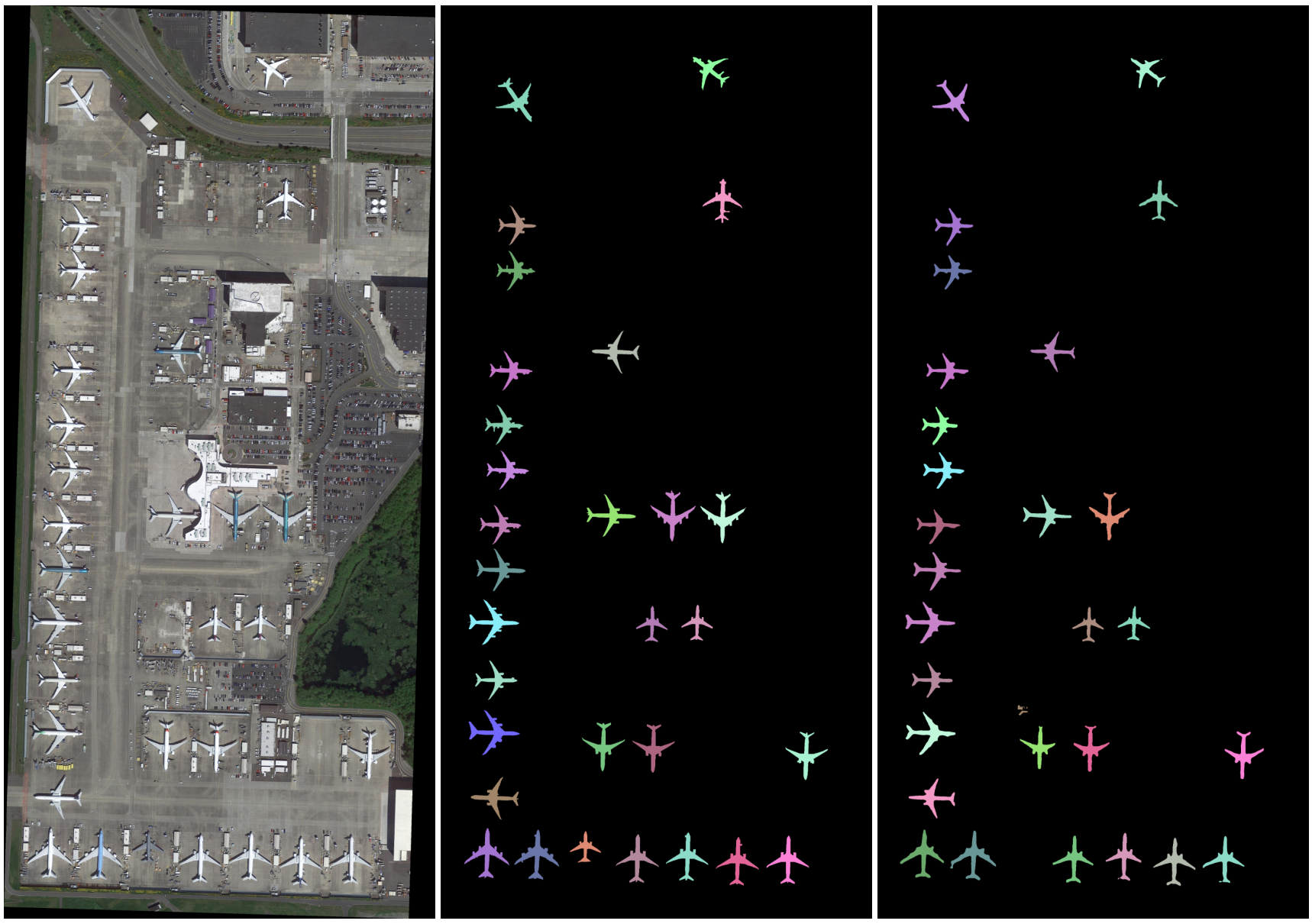
Kaggle Details

- Name: “my real score is .41”
- Best Score: **0.41221**
    - **NOTE:** On Kaggle, it shows my best score as **0.59572**. This score is wrong because I used the GT bounding boxes for the train set in one of my earlier submissions

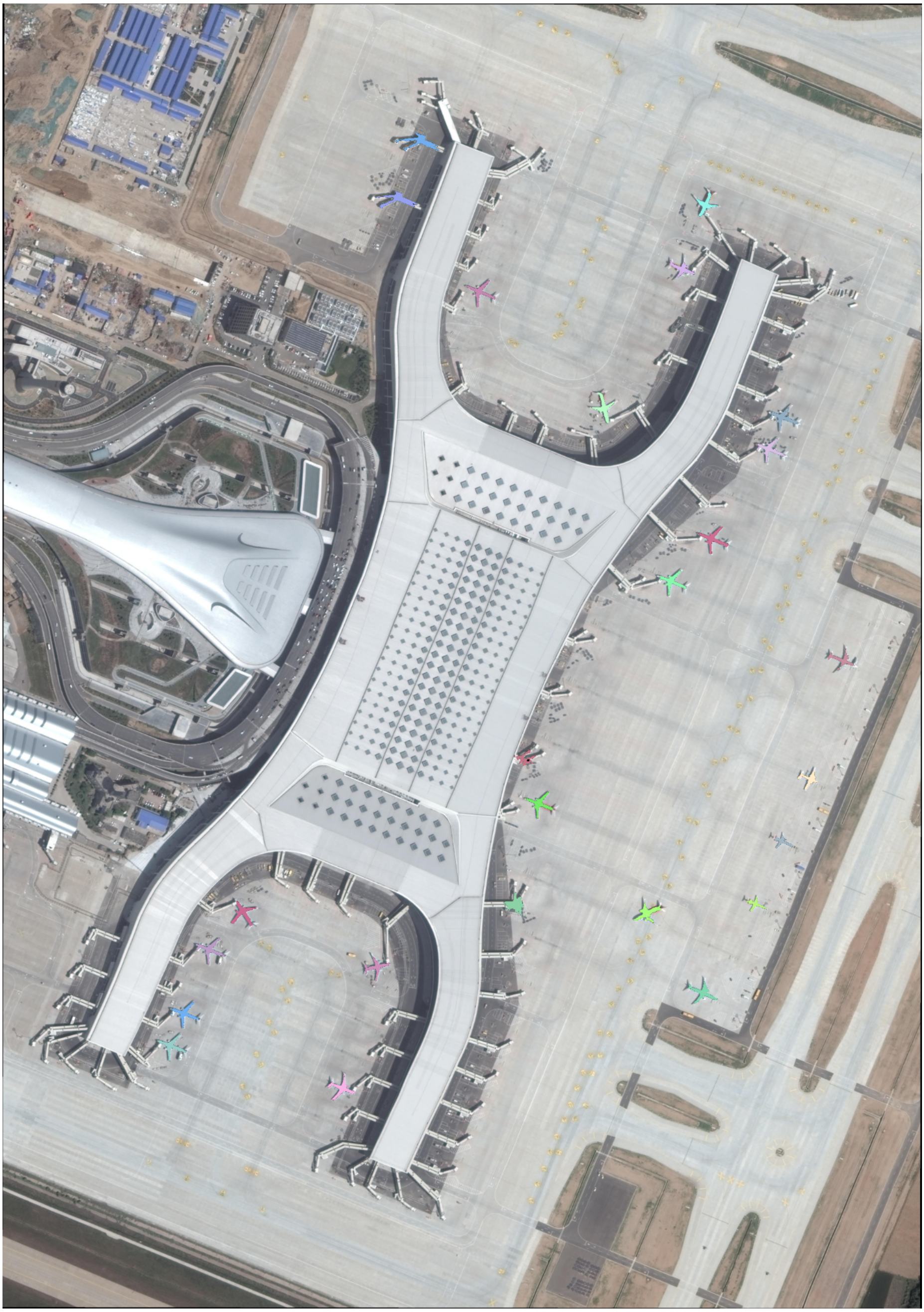
Test Set Visualization

Side-By-Side





Predictions Masks Over Image



Part 4

Evaluation

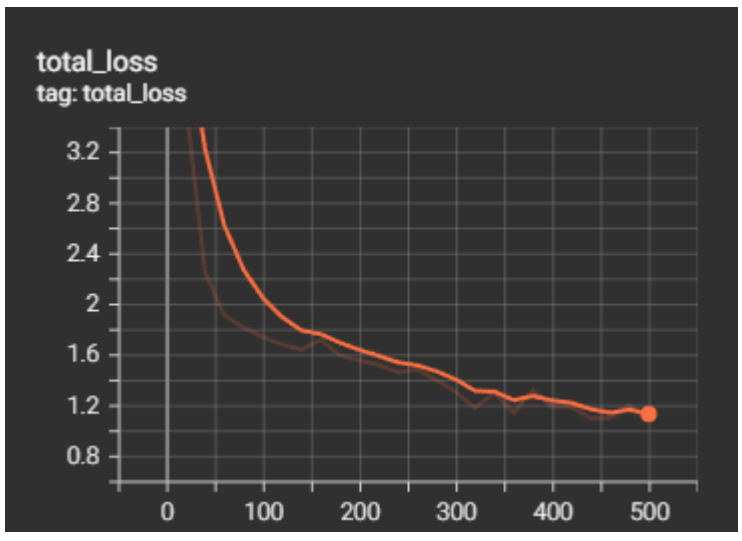
BBox

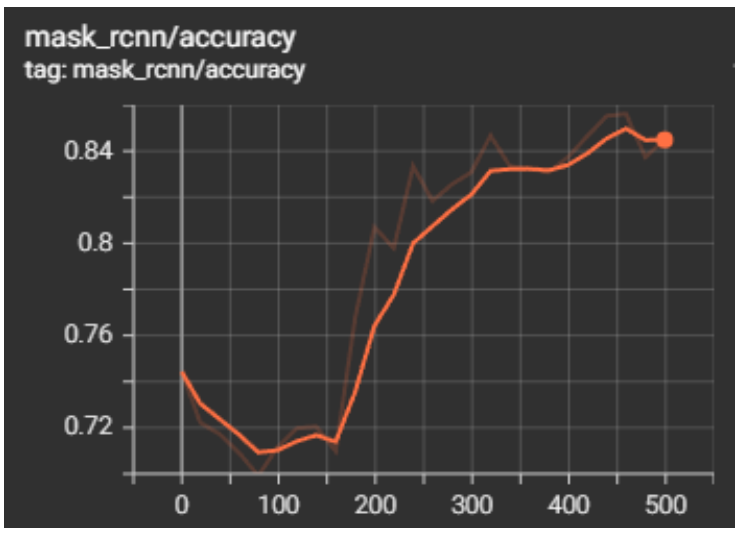
```
[11/06 23:17:51 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 27.543 | 46.081 | 30.162 | 17.496 | 36.567 | 54.836 |
```

Segm

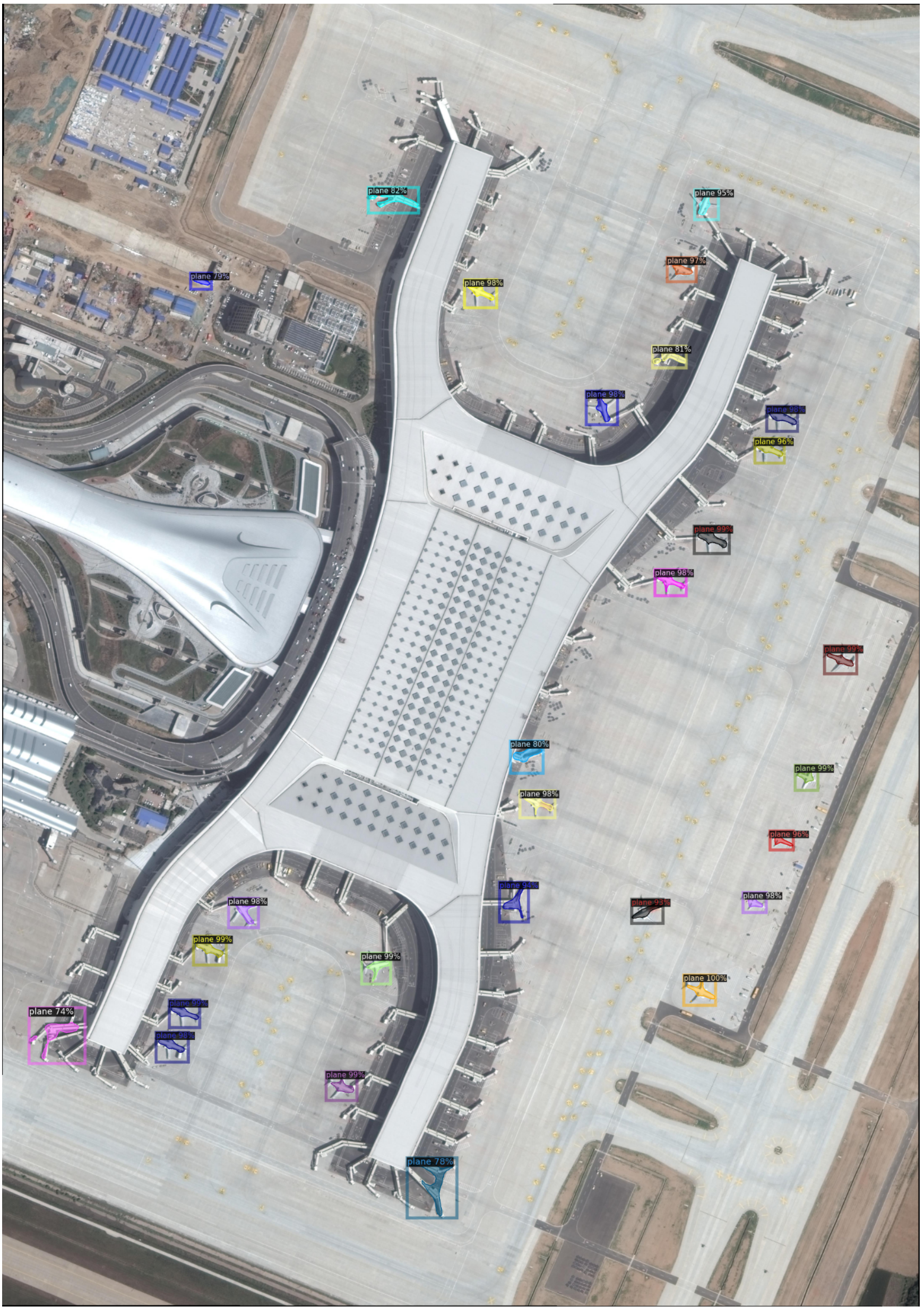
```
[11/06 23:17:53 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 6.962 | 24.832 | 1.688  | 3.594 | 7.694 | 27.617 |
```

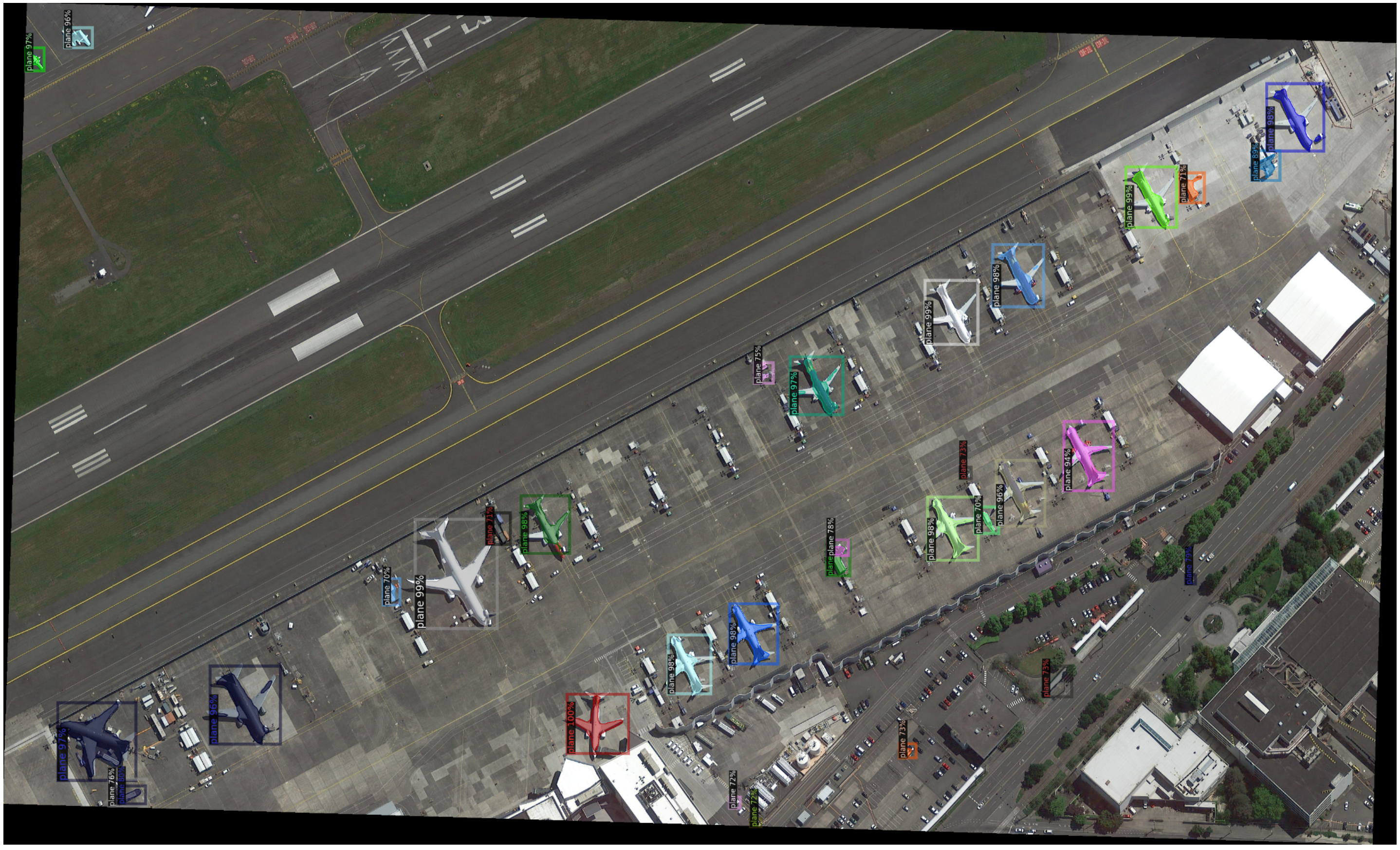
Loss and Accuracy Plots





Visualization

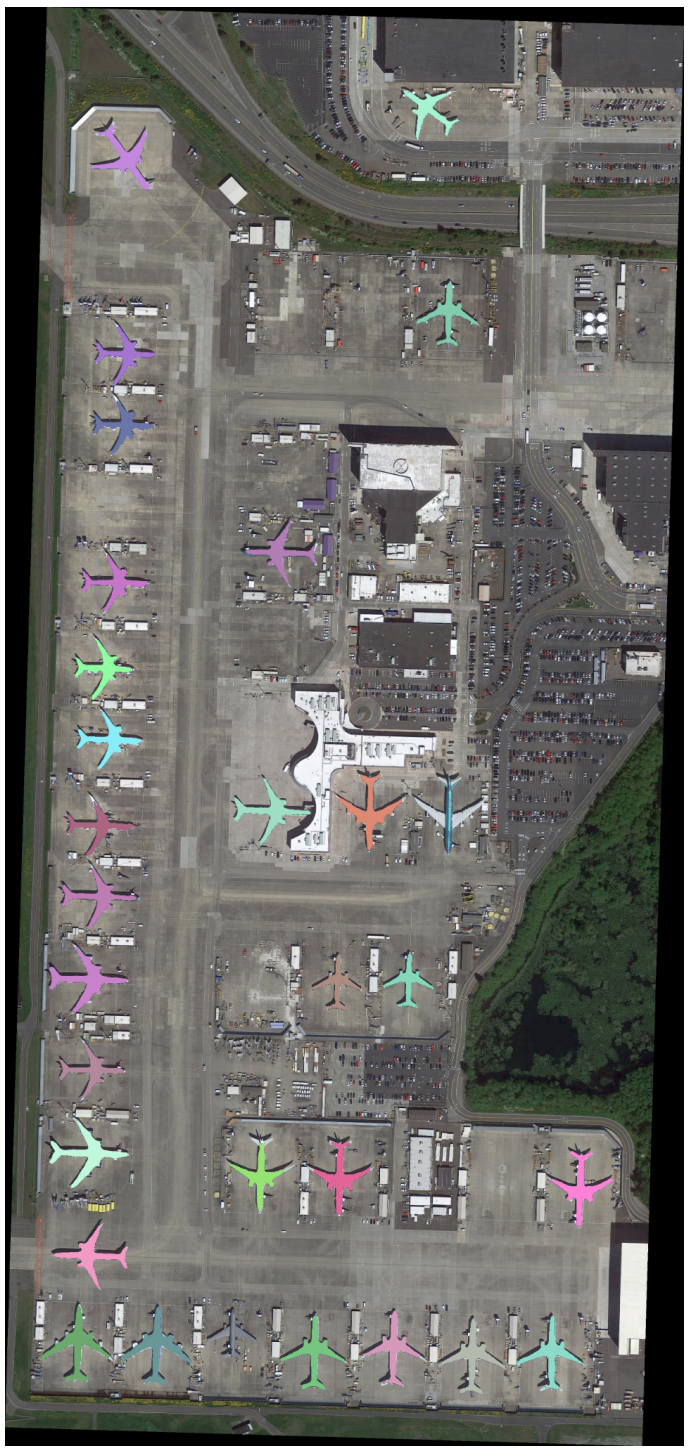



plane 97%
plane 96%
plane 75%
plane 97%
plane 99%
plane 98%
plane 71%
plane 88%
plane 96%
plane 99%
plane 71%
plane 70%
plane 99%
plane 98%
plane 98%
plane 100%
plane 98%
plane 78%
plane 98%
plane 70%
plane 96%
plane 94%
plane 79%
plane 73%
plane 73%
plane 72%
plane 72%
plane 97%
plane 76%
plane 96%


plane 99%
plane 99%
plane 71%
plane 97%
plane 93%
plane 98%
plane 99%
plane 99%
plane 98%
plane 99%
plane 99%
plane 100%
plane 99%
plane 99%
plane 98%
plane 99%
plane 100%
plane 99%
plane 100%
plane 99%
plane 98%
plane 92%
plane 94%
plane 96%
plane 97%
plane 76%
plane 97%
plane 97%
plane 98%
plane 97%

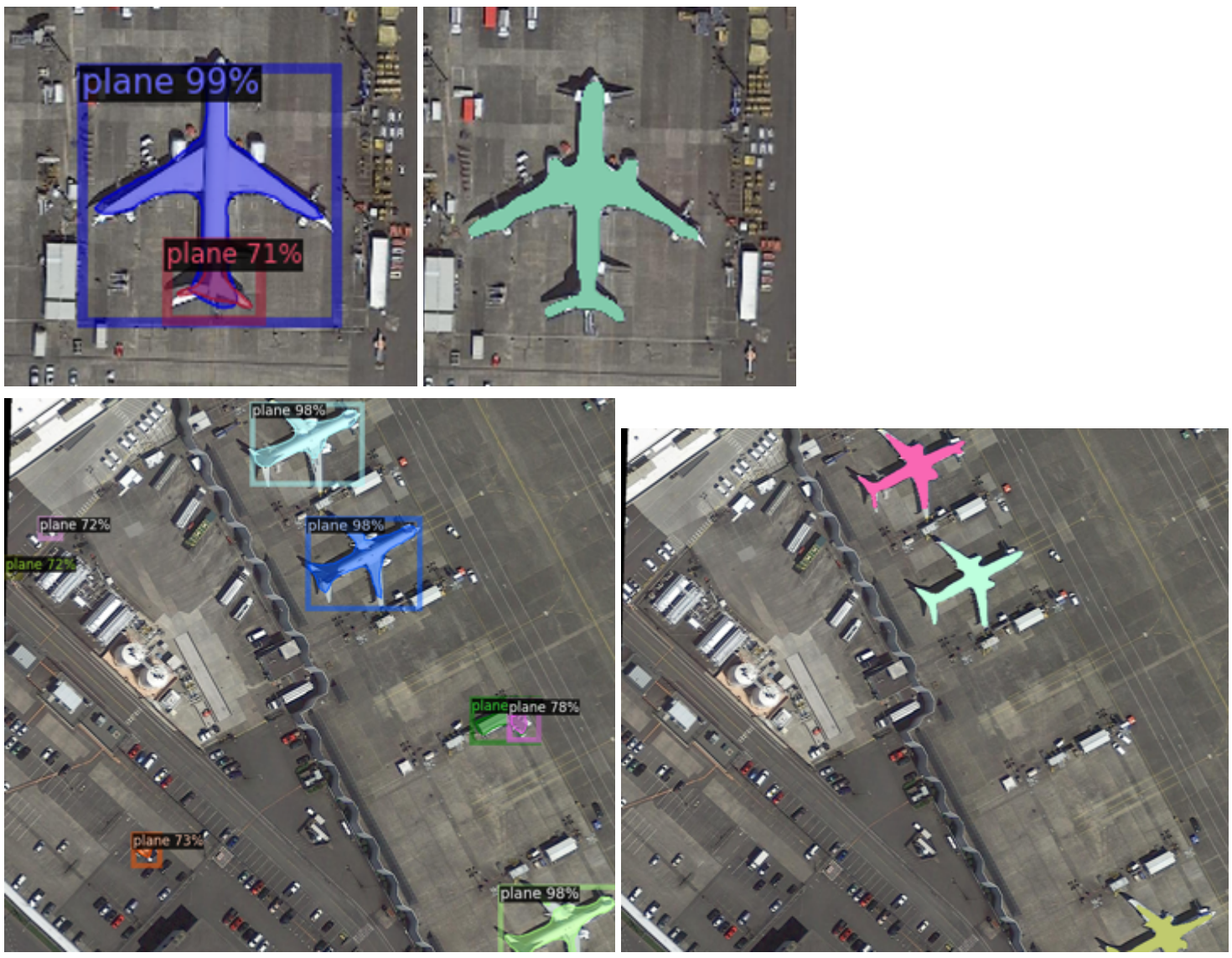
Comparison

Side-by-side comparison with the results from part 3



False Positives

With the Mask R-CNN model, I observed more false positives with high confidence (> 0.7).



Poor Masks

The Mask R-CNN model features more poorly shaped masks. While the model from part 3 had much more consistently “plane-shaped” masks.