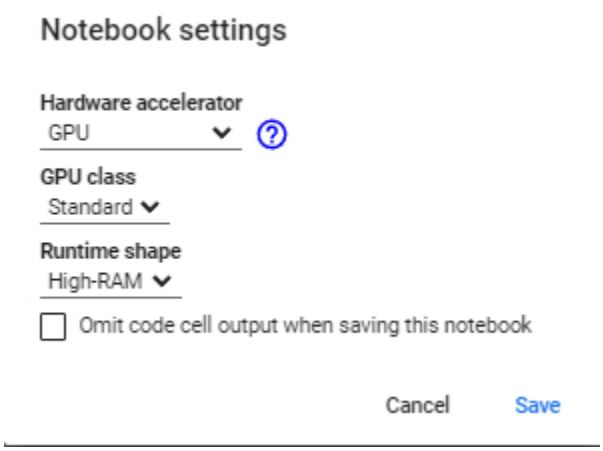
Cmpt 412

Project 3

Hao Ran Wei
301354701

Prereqs:
Turn on the following setting on google collab



Part1:

List of final configs and modifications used:

Max iteration: 800
Base LR: 0.0007
Batch size: 512
COCO model: faster_rcnn_X_101_32x8d_FPN_3x.yaml

Factors to help improve performance:

The reason why I chose the model faster_rcnn_X_101_32x8d_FPN_3x.yaml cause on the coco documentation the box AP had a higher accuracy than the default as shown in the ablation study. Also played mostly with base LR and max iteration to improve AP, due to both having the most impact on a total loss. Which is a good indicator of over fitting. I also opted to use the default data loader which had ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333, sample_style='choice') as its augmentation.

ablation study

Default

Config:

```
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS =
model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml")  #
Let training initialize 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  # pick a good LR
cfg.SOLVER.MAX_ITER = 500    # 300 iterations seems good enough for this toy
dataset; you will need to train longer for a practical dataset
cfg.SOLVER.STEPS = []        # do not decay learning rate
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512   # The "RoIHead batch size".
128 is faster, and good enough for this toy dataset (default: 512)
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
```

results:

```
creating index...
index created!
[02/25 19:38:59 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[02/25 19:39:00 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.10 seconds.
[02/25 19:39:00 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[02/25 19:39:00 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.190
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.317
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.206
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.103
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.252
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.013
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.106
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.213
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.101
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.275
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.630
[02/25 19:39:00 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 19.031 | 31.710 | 20.650 | 10.302 | 25.232 | 49.989 |
OrderedDict([('bbox', {'AP': 19.030548659393396, 'AP50': 31.709733096818766, 'AP75': 20.649587962667823, 'APs': 10.30200518759837, 'APm': 25.23187301024759, 'APl': 49.988805599044994})])
```

List of the configs and modifications that you used.

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download

Changed the default: `faster_rcnn_R_101_FPN_3x.yaml`

R101-FPN	3x	0.286	0.051	4.1	42.0	137851257	model | metrics

To: faster_rcnn_X_101_32x8d_FPN_3x.yaml

X101-FPN	3x	0.638	0.098	6.7	43.0	139173657	model | metrics

Reason: better box AP ratio

Results:

```
creating index...
index created!
[02/25 20:17:44 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[02/25 20:17:44 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.13 seconds.
[02/25 20:17:44 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[02/25 20:17:44 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.240
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.393
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.272
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.138
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.317
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.565
 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 ] = 0.117
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.264
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.136
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.344
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.669
[02/25 20:17:44 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 23.953 | 39.337 | 27.188 | 13.770 | 31.740 | 56.542 |
OrderedDict([('bbox', {'AP': 23.952565285759814, 'AP50': 39.33731045946842, 'AP75': 27.187713550379446, 'APs': 13.770489019991519, 'APm': 31.740071153988225, 'APl': 56.541637913252984})])
```

A much better results with an AP of 23.95 compared to 19.03.

Increase BASE_LR to 0.0007

Results:

```
creating index...
index created!
[02/25 21:05:46 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[02/25 21:05:46 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.16 seconds.
[02/25 21:05:46 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[02/25 21:05:46 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.314
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.510
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.353
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.204
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.403
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.617
 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 ] = 0.129
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.342
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.205
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.435
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.722
[02/25 21:05:46 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 31.425 | 51.022 | 35.323 | 20.351 | 40.336 | 61.746 |
OrderedDict([('bbox', {'AP': 31.424904330687873, 'AP50': 51.021593938926245, 'AP75': 35.323079223315396, 'APs': 20.351324180987163, 'APm': 40.33559901856648, 'APl': 61.746205269859104})])
```

Significant increase from AP 23.95 to 31.

Change batch size from 512 to 128 as suggested in the tutorial to speed up training speed.
Result:

```
creating index...
index created!
[02/25 21:39:04 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[02/25 21:39:04 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.13 seconds.
[02/25 21:39:04 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[02/25 21:39:04 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.292
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.456
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.346
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.178
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.383
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.617
 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 ] = 0.126
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.317
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.178
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.410
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.720
[02/25 21:39:04 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 29.244 | 45.565 | 34.620 | 17.757 | 38.267 | 61.746 |
OrderedDict([('bbox', {'AP': 29.24359955514254, 'AP50': 45.56526556606061, 'AP75': 34.61980749098994, 'APs': 17.757068940214214, 'APm': 38.267113536313616, 'APl': 61.746205269859104})])
```

Notice how a lower batch size actually decreased AP, but barely.

Changing batch size from 512 to 768 to try to test AP.
Results:

```
DONE (t=0.01s)
creating index...
index created!
[02/25 22:28:27 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[02/25 22:28:27 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.13 seconds.
[02/25 22:28:27 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[02/25 22:28:27 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.301
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.483
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.345
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.195
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.383
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.602
 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 ] = 0.128
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.330
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.195
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.419
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.721
[02/25 22:28:27 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 30.085 | 48.270 | 34.523 | 19.471 | 38.331 | 60.163 |
OrderedDict([('bbox', {'AP': 30.085400610910295, 'AP50': 48.26964491340521, 'AP75': 34.523160825362396, 'APs': 19.47108024072006, 'APm': 38.330874125800044, 'APl': 60.16264288815803})])
```

This seems not to affect the “AP” that much so I kept it as 512

Next increase the epoch from 500 to 800 since that’s where the total loss seems to stagnate:

Results:

```
[02/27 20:24:15 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 31.278 | 50.289 | 35.591 | 19.593 | 40.661 | 66.012 |
OrderedDict([('bbox', {'AP': 31.27763852613224, 'AP50': 50.28872537761709, 'AP75': 35.59114212561016, 'APs': 19.592516332434627, 'APm': 40.660917296157365, 'APl': 66.01196322710581})])
```

Very slight difference

Note: some optimization that I used is to save the get_detection_data list as a global directory. This made training much faster, as it takes a lot longer to load the images once instead of checking the list every time in the lambda function. I also set the SCORE_THRESH_TEST to be 0.75

Example test prediction visualization

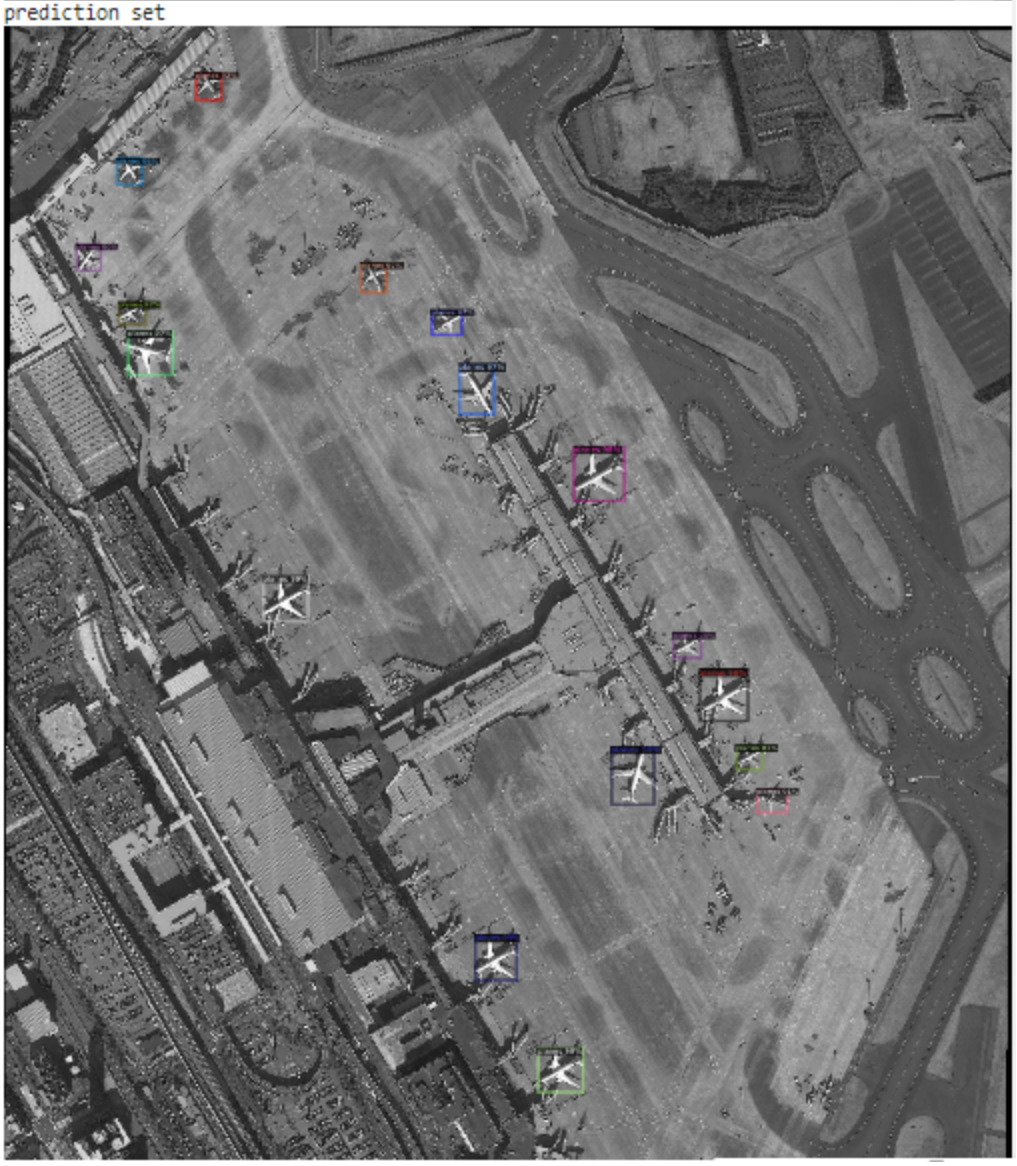
Config:

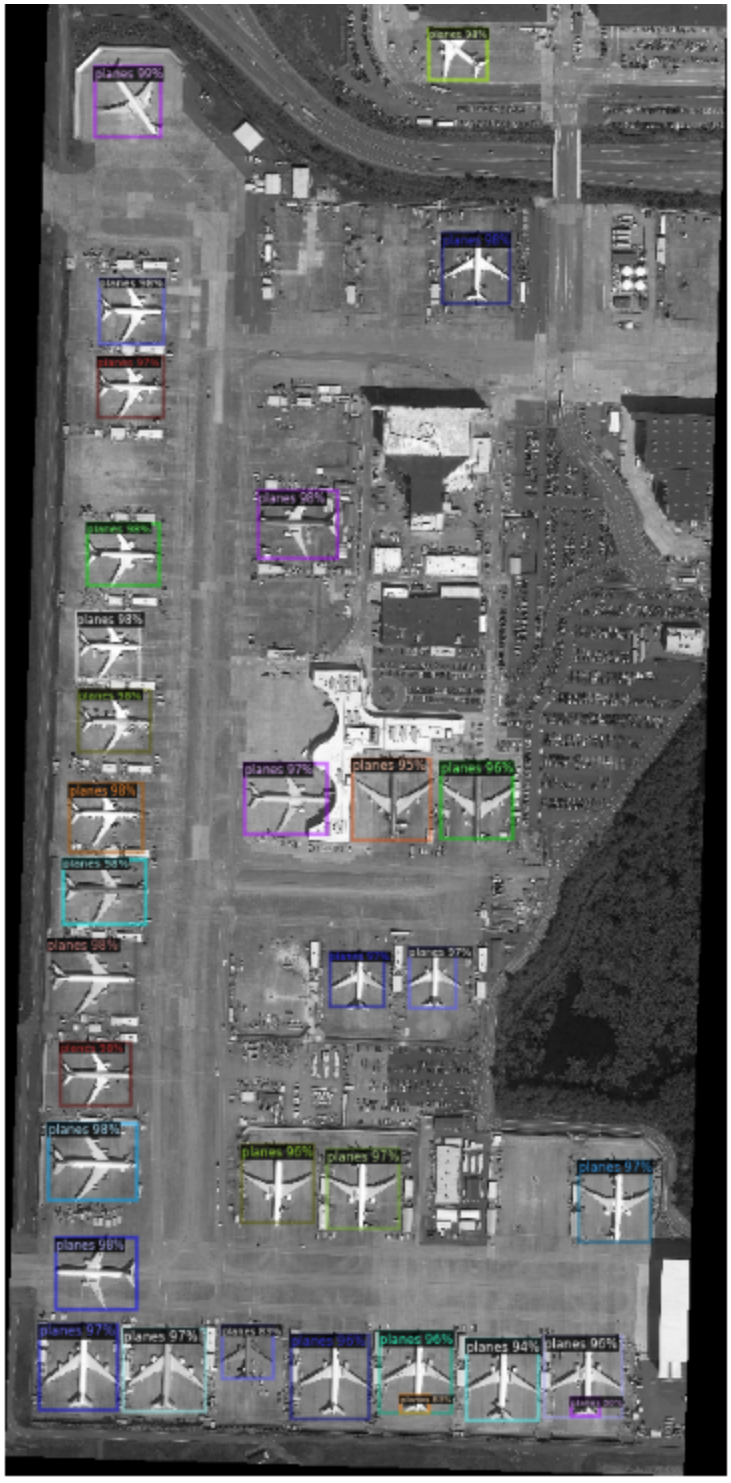
```
cfg.SOLVER.IMS_PER_BATCH = 2  # This is the real "batch size" commonly known to deep learning people
cfg.SOLVER.BASE_LR = 0.0009  # pick a good LR
cfg.SOLVER.MAX_ITER = 300
```

Visualization:

prediction set

prediction set





Visualization with the following config:

```
cfg.SOLVER.IMS_PER_BATCH = 2  # This is the real "batch size" commonly known to deep learning people
cfg.SOLVER.BASE_LR = 0.000000001  # pick a good LR
```

```
cfg.SOLVER.MAX_ITER = 300
```

Visualization:

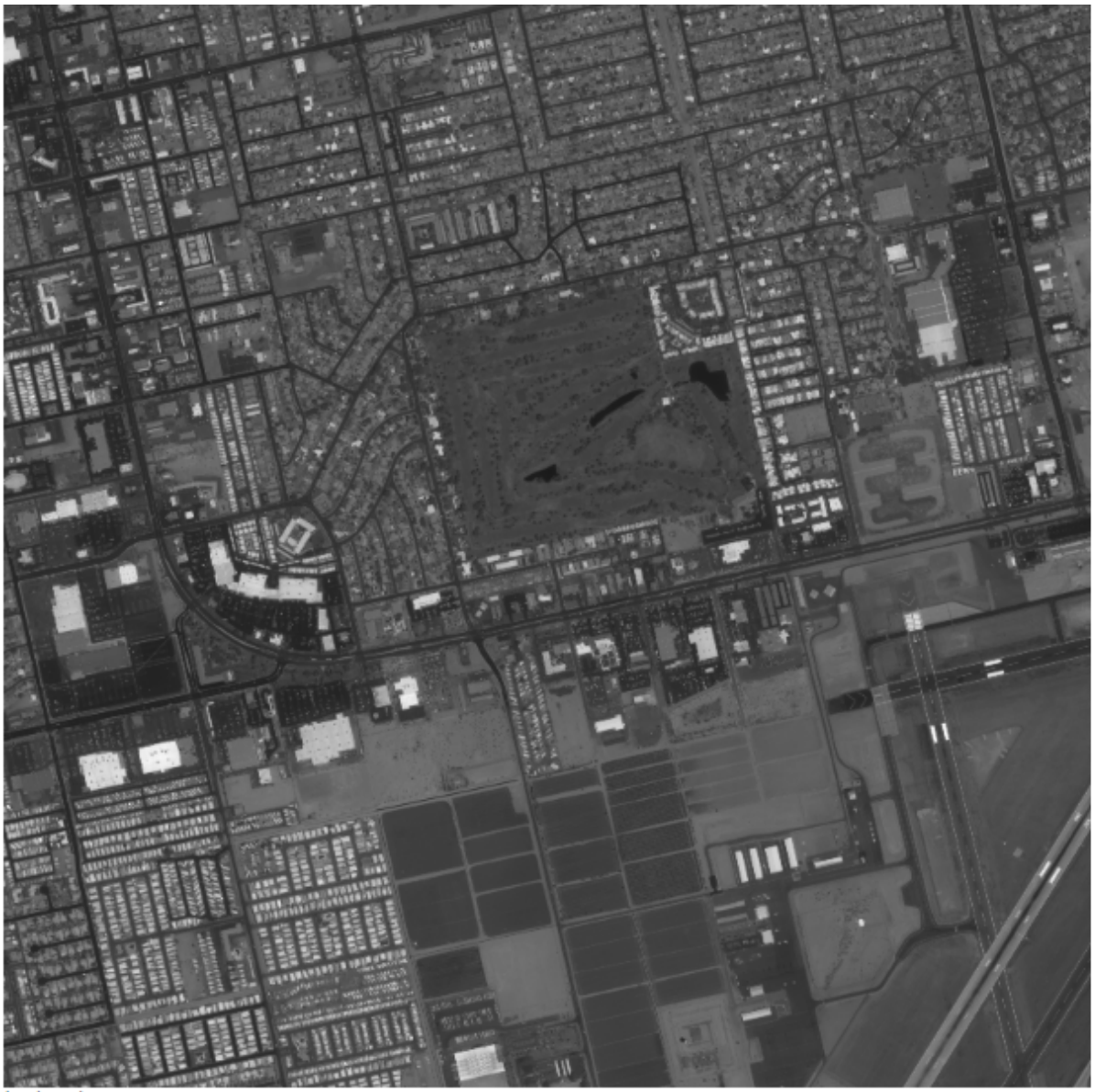
prediction set

prediction set

Final Visualization:

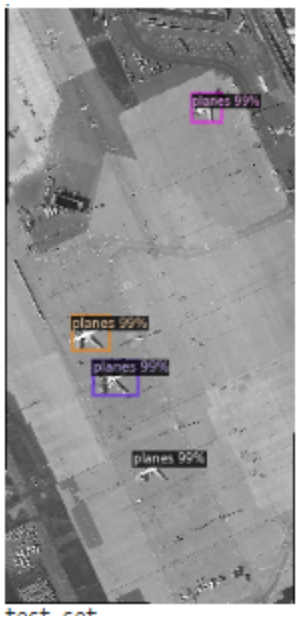
Test set:

Prediction set



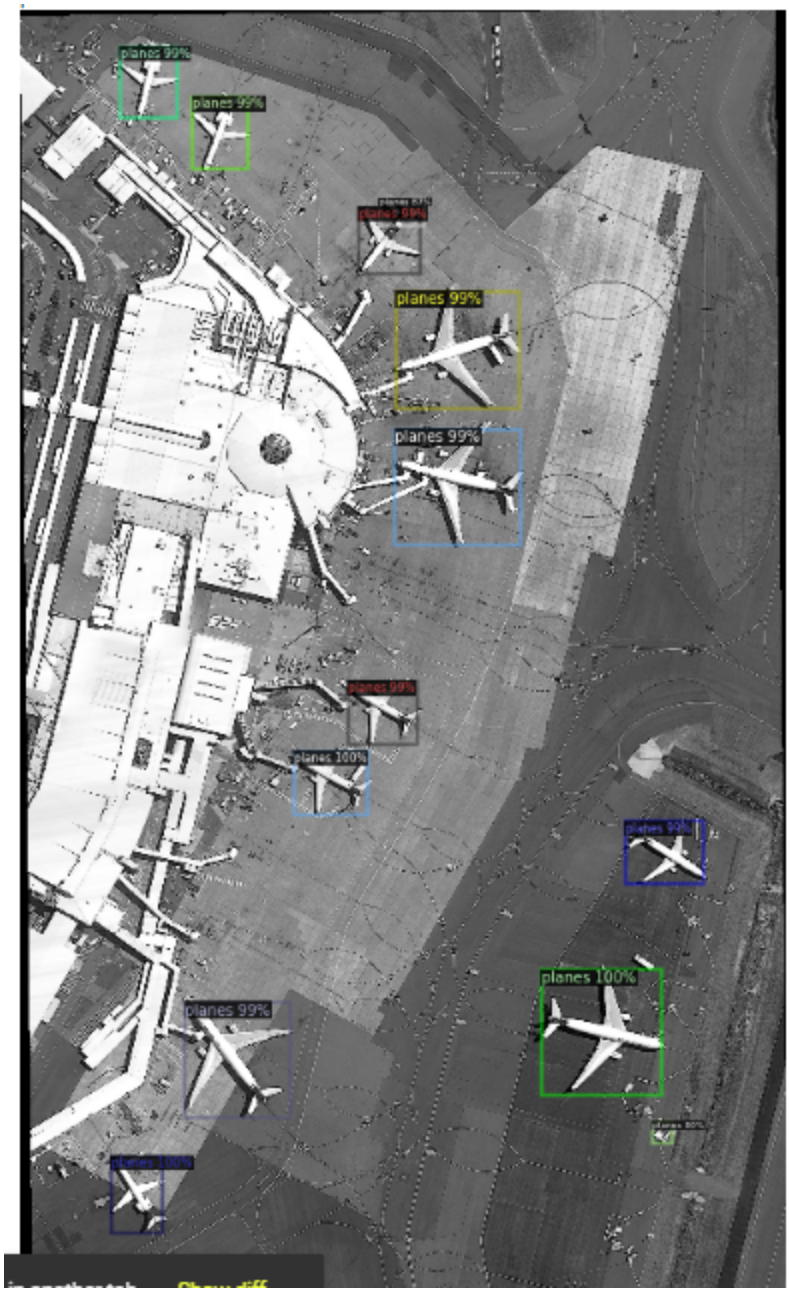
test set

Prediction set

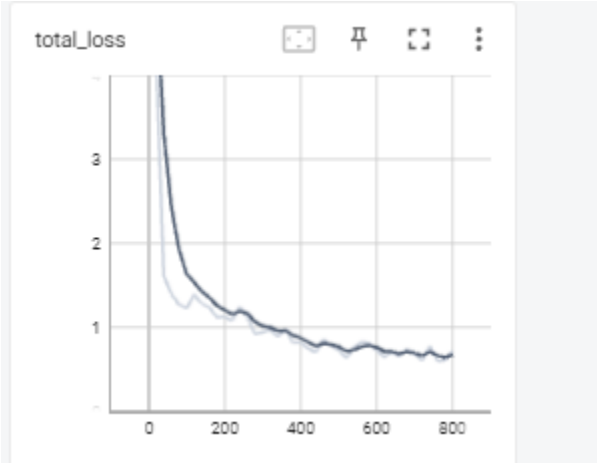


Test set:

Prediction set:


planes 99%
planes 99%
planes 99%
planes 99%
planes 99%
planes 100%
planes 99%
planes 100%
planes 99%
planes 100%

Total loss graph:



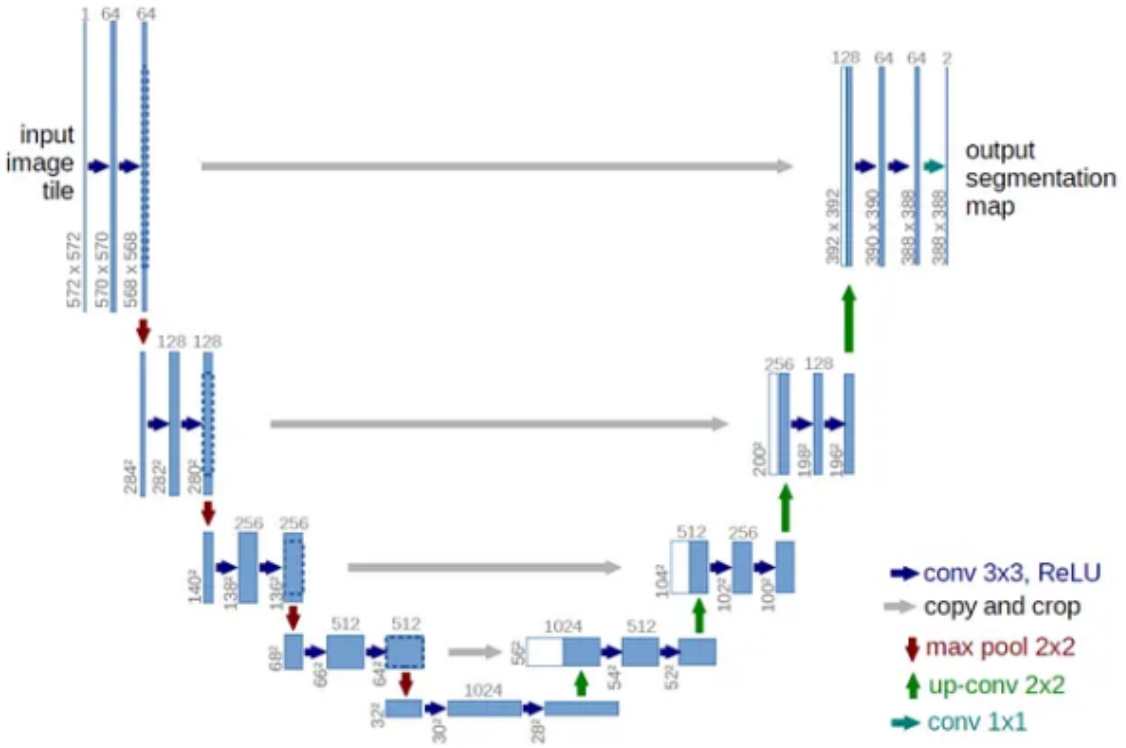
Part 2

Edits and changes in base code:

Changing the function `get_instance_sample` to load all file names at once instead of by idx individually significantly increased the training time of the Unet model and allowed the function to train up to 50 epochs in 1 hr instead of 1 epoch per hour. This method is memoization. I also pre-loaded all the files as a dict to save on RAM.

Unet model with 1024x1024 as lowest channel visual results:

- note that the prediction is significantly lower ei as shown in the model below:



My modified version of Unet:

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
BatchNorm2d-47	[-1, 256, 4, 4]	512
ConvTranspose2d-48	[-1, 256, 8, 8]	262,400
Conv2d-49	[-1, 128, 8, 8]	295,040
BatchNorm2d-50	[-1, 128, 8, 8]	256
ReLU-51	[-1, 128, 8, 8]	0
conv-52	[-1, 128, 8, 8]	0
up-53	[-1, 128, 8, 8]	0
BatchNorm2d-54	[-1, 128, 8, 8]	256
ConvTranspose2d-55	[-1, 128, 16, 16]	65,664

```
        Conv2d-56          [-1, 64, 16, 16]          73,792
   BatchNorm2d-57          [-1, 64, 16, 16]             128
          ReLU-58          [-1, 64, 16, 16]               0
          conv-59          [-1, 64, 16, 16]               0
            up-60          [-1, 64, 16, 16]               0
   BatchNorm2d-61          [-1, 64, 16, 16]             128
onvTranspose2d-62          [-1, 64, 32, 32]          16,448
        Conv2d-63          [-1, 32, 32, 32]          18,464
   BatchNorm2d-64          [-1, 32, 32, 32]              64
          ReLU-65          [-1, 32, 32, 32]               0
          conv-66          [-1, 32, 32, 32]               0
            up-67          [-1, 32, 32, 32]               0
   BatchNorm2d-68          [-1, 32, 32, 32]              64
onvTranspose2d-69          [-1, 32, 64, 64]           4,128
        Conv2d-70          [-1, 16, 64, 64]           4,624
   BatchNorm2d-71          [-1, 16, 64, 64]              32
          ReLU-72          [-1, 16, 64, 64]               0
          conv-73          [-1, 16, 64, 64]               0
            up-74          [-1, 16, 64, 64]               0
   BatchNorm2d-75          [-1, 16, 64, 64]              32
onvTranspose2d-76         [-1, 16, 128, 128]           1,040
        Conv2d-77          [-1, 8, 128, 128]           1,160
   BatchNorm2d-78          [-1, 8, 128, 128]              16
          ReLU-79          [-1, 8, 128, 128]               0
          conv-80          [-1, 8, 128, 128]               0
            up-81          [-1, 8, 128, 128]               0
   BatchNorm2d-82          [-1, 8, 128, 128]              16
        Conv2d-83          [-1, 3, 128, 128]             219
   BatchNorm2d-84          [-1, 3, 128, 128]               6
          ReLU-85          [-1, 3, 128, 128]               0
          conv-86          [-1, 3, 128, 128]               0
        Conv2d-87          [-1, 1, 128, 128]              28
          conv-88          [-1, 1, 128, 128]               0
================================================================
al params: 4,549,765
inable params: 4,549,765
-trainable params: 0
----------------------------------------------------------------
ut size (MB): 0.19
ward/backward pass size (MB): 39.22
ams size (MB): 17.36
imated Total Size (MB): 56.76
----------------------------------------------------------------
```

Did not include unsample cause it slowed down the model too much. Also did not go up to 1024x1024 since it causes too much loss of information, as shown in the section above. I also normalized the output of the copy and crop section to eliminate more noise.

Current cong:

```
num_epochs = 60
batch_size = 8
learning_rate = 0.0001
weight_decay = 1e-5
```

Note: in visualizatoin the order of the photos goes from original image then prediction than true mask.
Resulting visualization:

Mean IOU:

```
#images: 7980, Mean IoU: 0.7003960934377673
```

Change learning rate to 0.1
Final results in mean IOU:

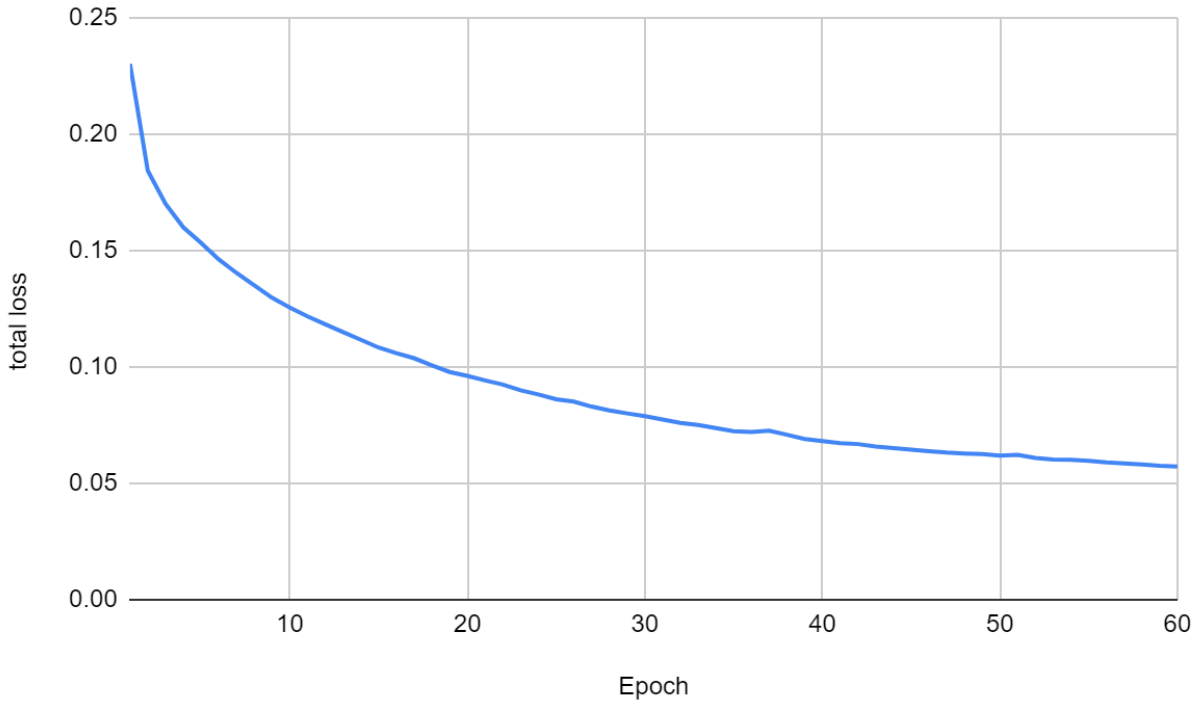
```
100% 998/998 [00:11<00:00, 84.22it/s]

#images: 7980, Mean IoU: 0.9068593285289236
```

Final Visualization:

Loss function: `BCEWithLogitsLoss()`

Loss graph:



Part 3:

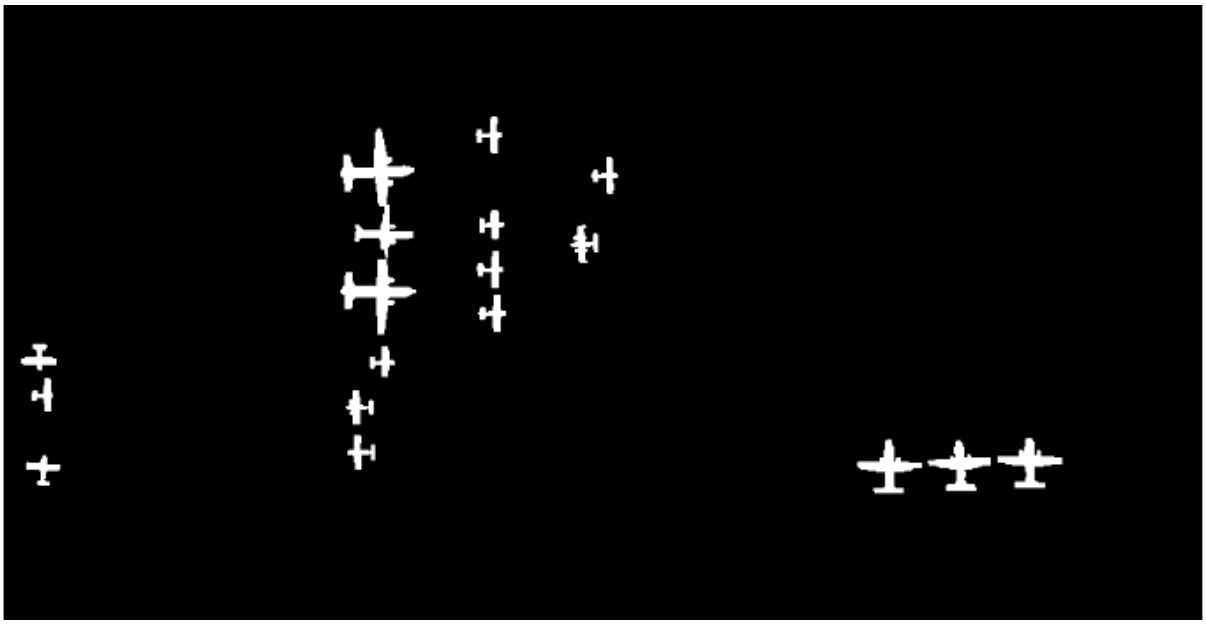
Group name on kaggle: hrwei_sfu

Group members: Justin Lin, Hao Ran Wei

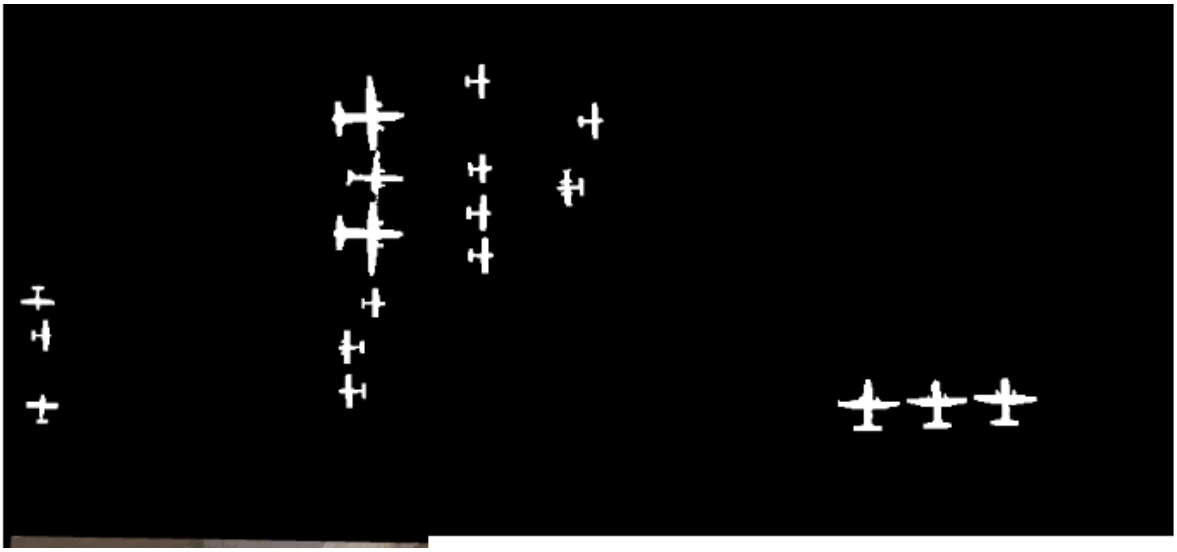
Current best score on kaggle: 0.63761

Note: due to the size of the image, the screen shots may not cover the entire image.
Visualization:
original:

True mask



prediction:



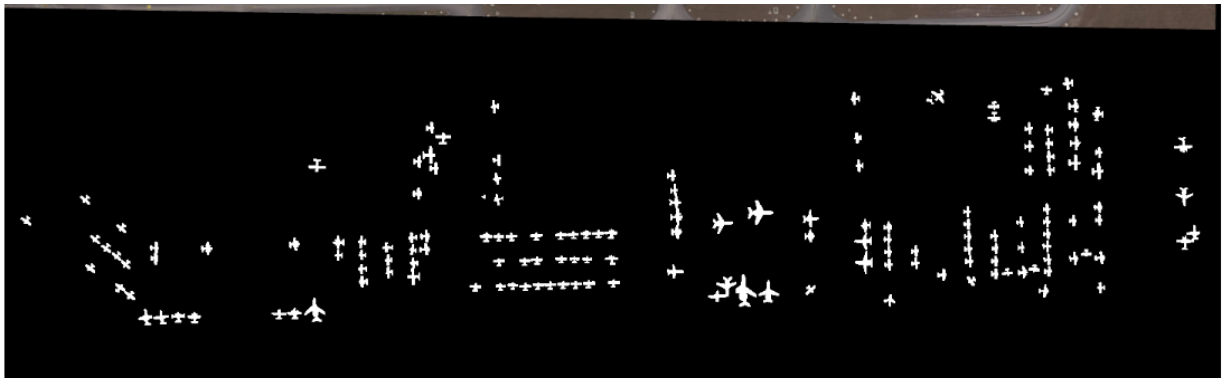
original

True mask

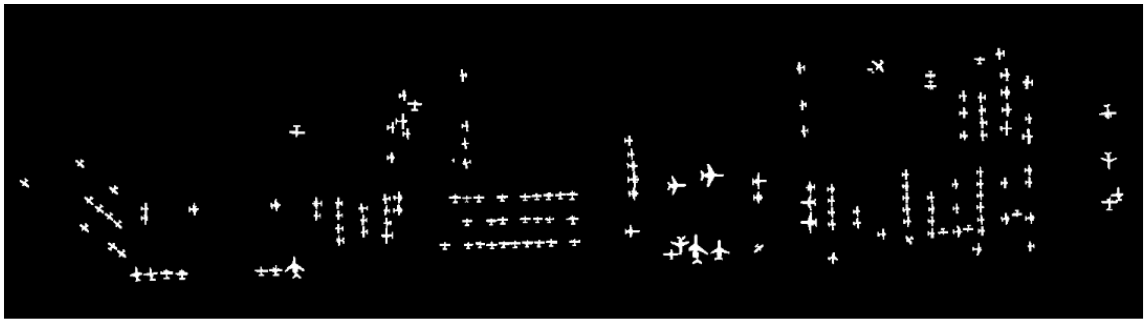
Prediction mask

original:

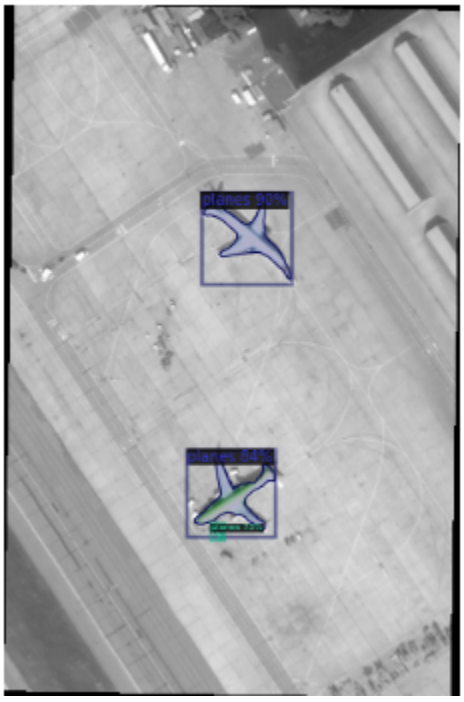
True mask

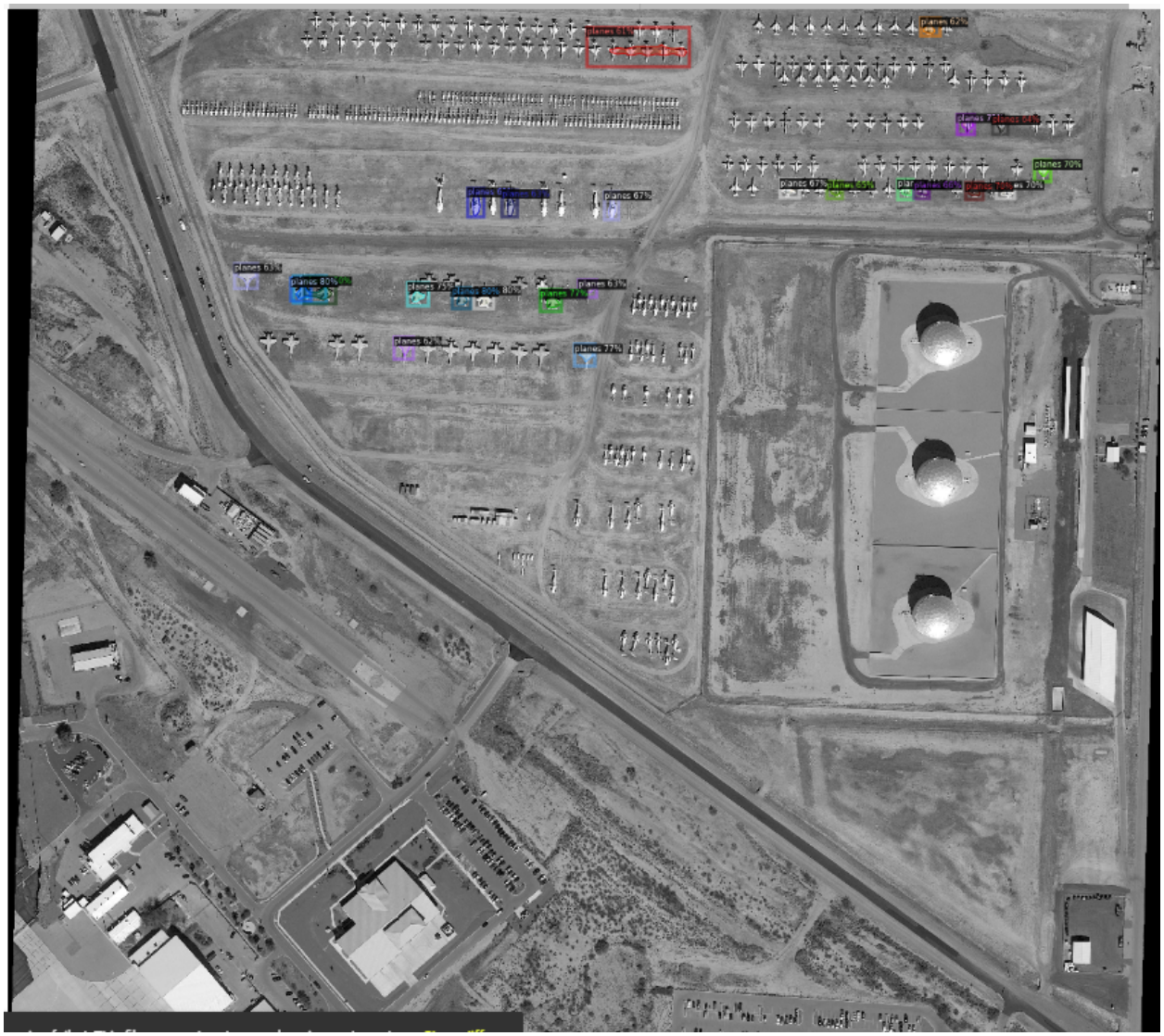


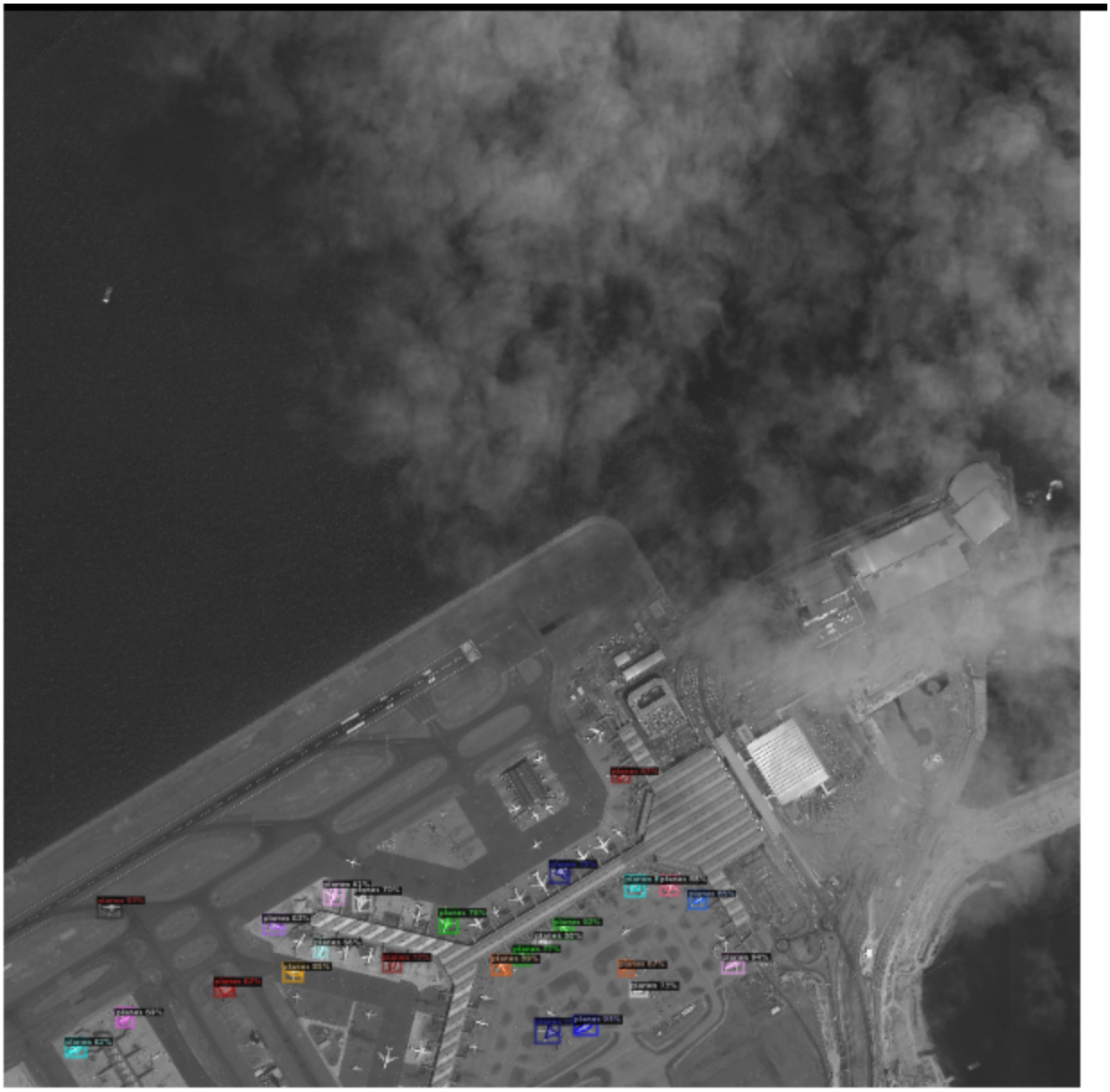
Prediction mask:



Part 4:
visualization:







This model is way worse than the one used in part 1. The Ap score for this model is 0.119 where as in part 1 it’s around 0.31

```
[02/24 02:20:11 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[02/24 02:20:12 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 1.52 seconds.
[02/24 02:20:12 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[02/24 02:20:12 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.03 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.000
[02/24 02:20:12 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
|:-----:|:------:|:------:|:-----:|:-----:|:-----:|
| 0.000 | 0.001  | 0.000  | 0.000 | 0.000 | 0.000 |
OrderedDict([('bbox', {'AP': 0.11915358397089167, 'AP50': 0.532702566638666, 'AP75': 0.011977004152028105, 'APs': 0.04021587619959273, 'APm': 0.276
```