Name: Parsa Alamzadeh
Computing ID: palamzad
Kaggle Team Name: **Papa Goose**

Date: Nov. 1st, 2020
Student number: 301316272

Project 3: Semantic Segmentation

Link to my colab notebook:
https://colab.research.google.com/drive/1cxwu0Fm2cvCTd4Mz-J5Td4SQWm9v8eq0?usp=sharing

Part 1:

For this section, I first cached the data into a variable so I don't have to read from the disk every time. This significantly reduced the time of training, 2-3 mins preloading as opposed to 2-3 minutes every time for training.

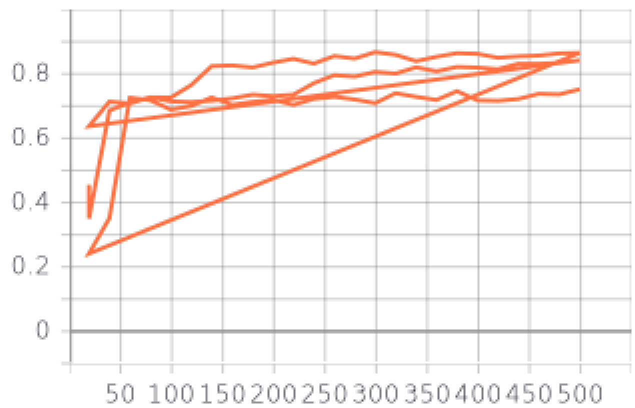
For configuration of the pretrained model I only modified the MAX_ITER and the BASE_LR. Here is the configuration i used that outperformed the other configurations:

BASE_LR: 0.00025 MAX_ITER: 1,000

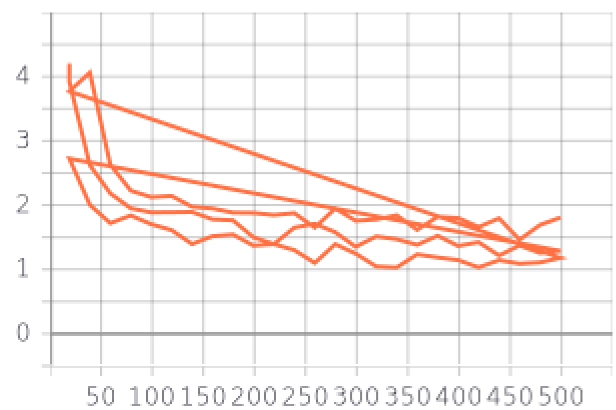
Different learning rates converged to their final accuracy in different rates. The one with 0.001 started off with around 65% accuracy and barely improved (converged around 72-73%). 0.0005 also had a similar issue where it started off with a high accuracy but failed to improve as much, therefore it was still a bit high and when I chose 0.00025 I saw that accuracy steadily improved. A higher BASE_LR had a lower accuracy while a lower learning struggled to converge. This also can be seen in the Total Loss graph as well. The reason for that is that too large of a learning rate cannot converge to local minima while too small of a learning rate can't find the global minima and gets stuck at the local minima.

Another modification that I made was using the faster_rcnn_X_101_32x8d_FPN_3x.yaml instead of the suggested one, since it had a better convergence compared to the provided one.

Here are the graphs from different configurations:
Accuracy of the models:

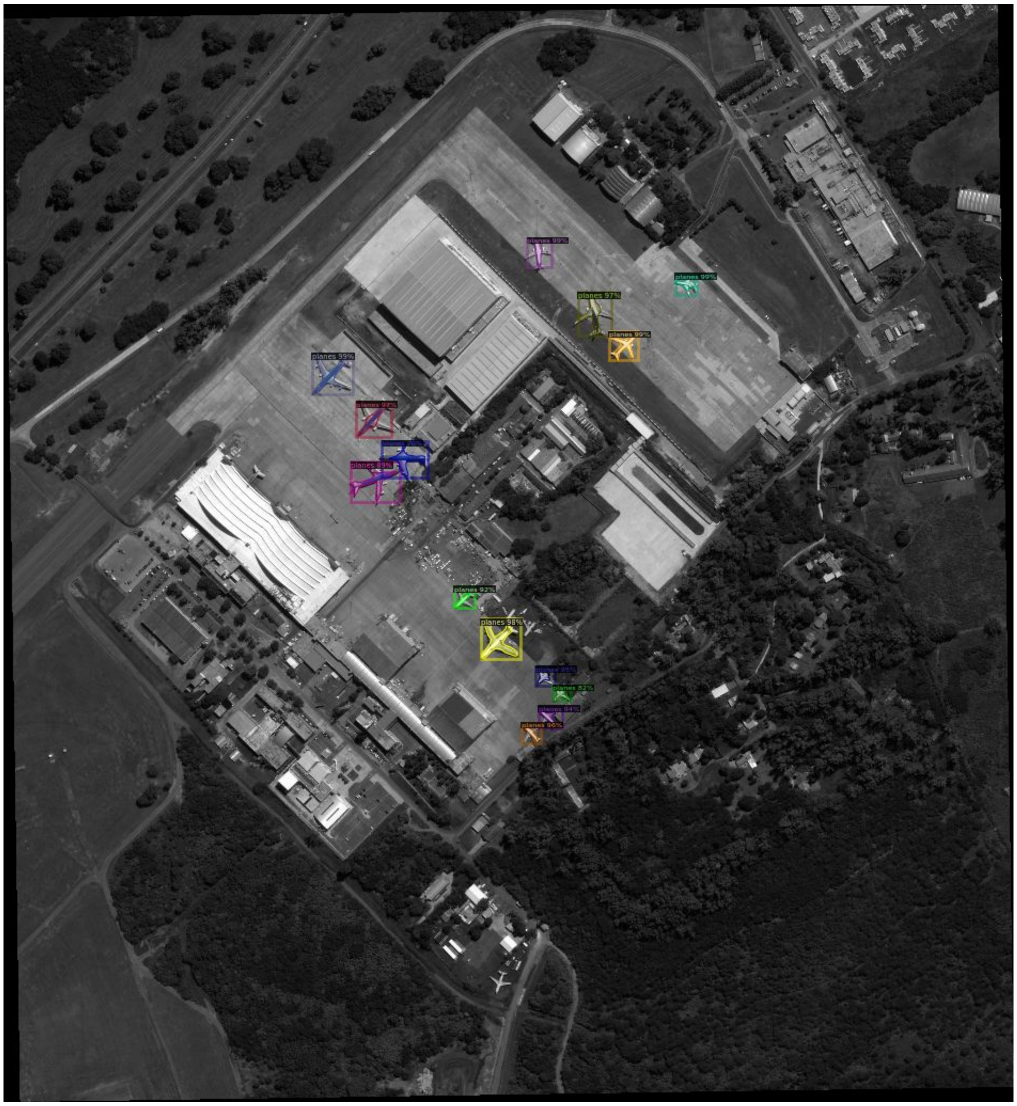


Total loss:



The straight lines are the linear regression automatically generated which I couldn’t remove or modify. The learning rates I tried were: 0.001, 0.0005, 0.00025 and 0.0001. The reason I used a higher MAX_ITER was to allow the lower learning rate to converge.

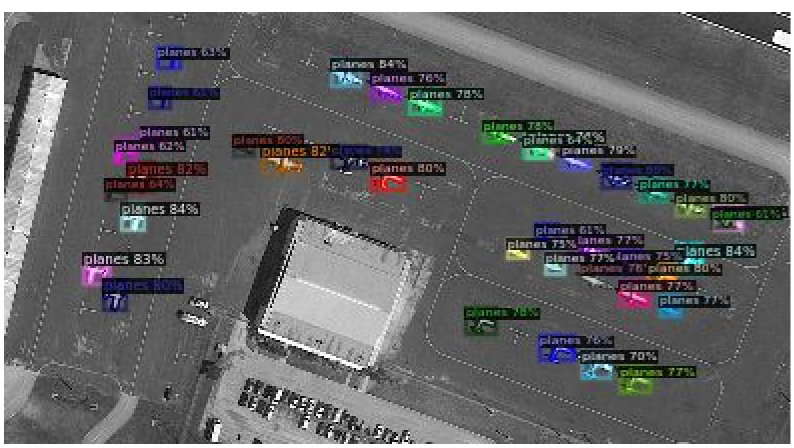
Here are some visualizations done on the test dataset.
Visualizations from the optimal model:

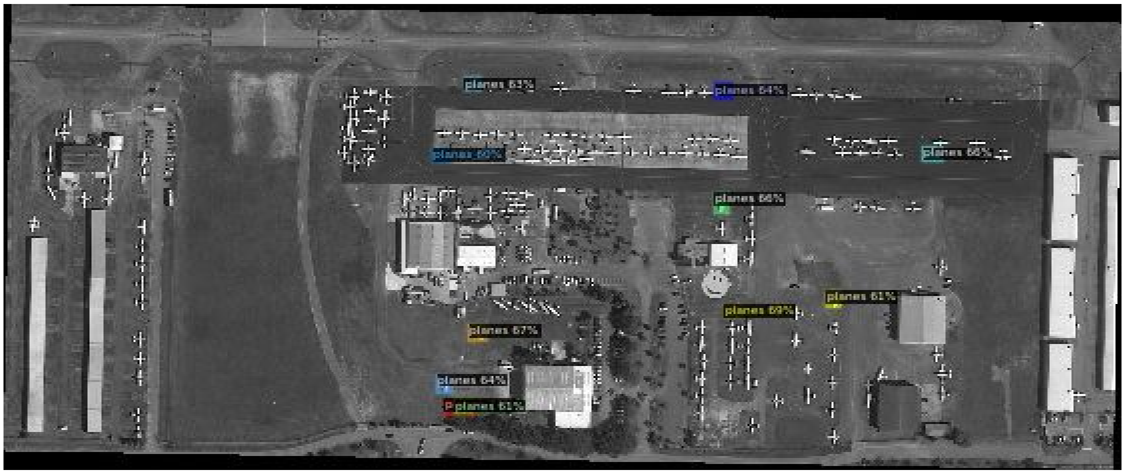



planes 98%
planes 99%
planes 98%
planes 99%
planes 99%
planes 98%
planes 99%
planes 97%
planes 99%


planes 97%
planes 92%
planes 98%
planes 97%
planes 98%
planes 99%
planes 98%
planes 99%
planes 92%
planes 99%

The results from a higher learning rate:





While the masks are not bad, it is missing a lot of planes.

Here is the results from a lower learning rate:

This model fails completely to make any predictions with a high confidence.

As for evaluation, I set aside 10 (~5% of the data) images from the training dataset for validation. Here is the results from COCOevaluator on the validation data:
BBox results:

```
Evaluate annotation type *bbox*
COCOeval_opt.evaluate() finished in 0.02 seconds.
Accumulating evaluation results...
COCOeval_opt.accumulate() finished in 0.00 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.312
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.550
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.329
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.359
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.227
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.735
 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 ] = 0.338
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.367
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.245
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.829
[11/01 08:18:20 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 31.216 | 54.958 | 32.930 | 35.947 | 22.717 | 73.536 |
```

Segmentation results:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.113
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.385
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.019
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.099
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.088
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.447
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.007
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.054
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.142
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.128
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.104
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.497
[11/01 08:18:20 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs  |  APm  |  APl   |
|:------:|:------:|:------:|:-----:|:-----:|:------:|
| 11.300 | 38.490 | 1.922  | 9.876 | 8.814 | 44.710 |
```

Part 2:

Similar to part 1, I cached the data needed for training the model. As for the hyperparameters, I had the epochs set to 50, with batch size of 32 with initial learning rate of 0.1 and weight decay of 0.00001. Every 15 epochs (after the 30th epoch), I decreased the learning rate by a factor of 10.

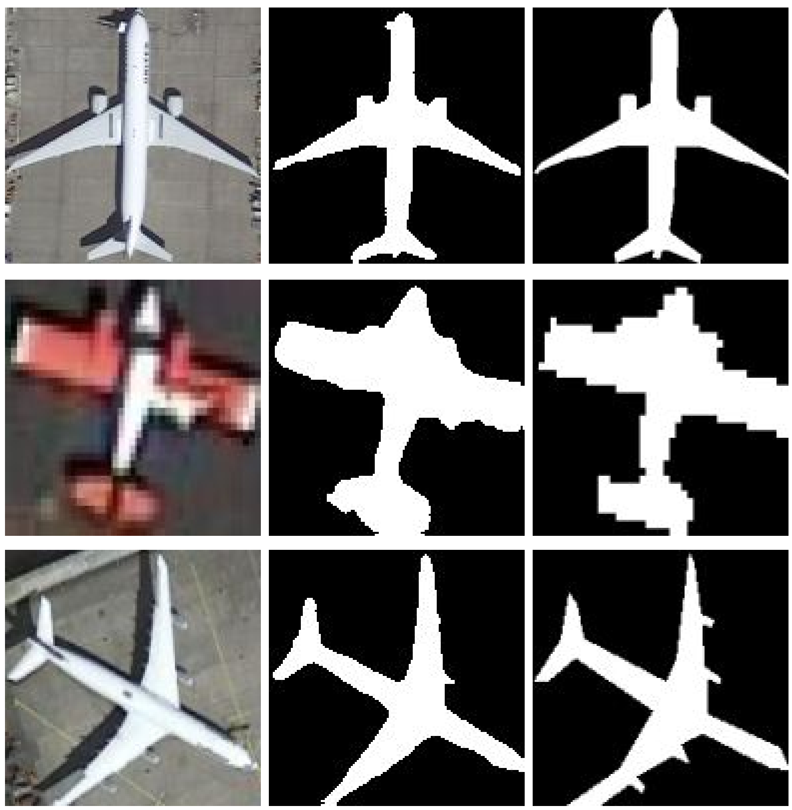
Here is the architecture of the network I used for my predictions:

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
ConvTranspose2d-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
ConvTranspose2d-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
ConvTranspose2d-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

The model's architecture is very similar to the existing one, but I also borrowed the skip layer component from resnet, which improved the accuracy of the model. Output of downsampling layers are copied and added to the upsampling ones. I have 6 downsampling layers and 6 up sampling. As for the optimizer and loss function, I used the default ones and achieved loss of 0.07.

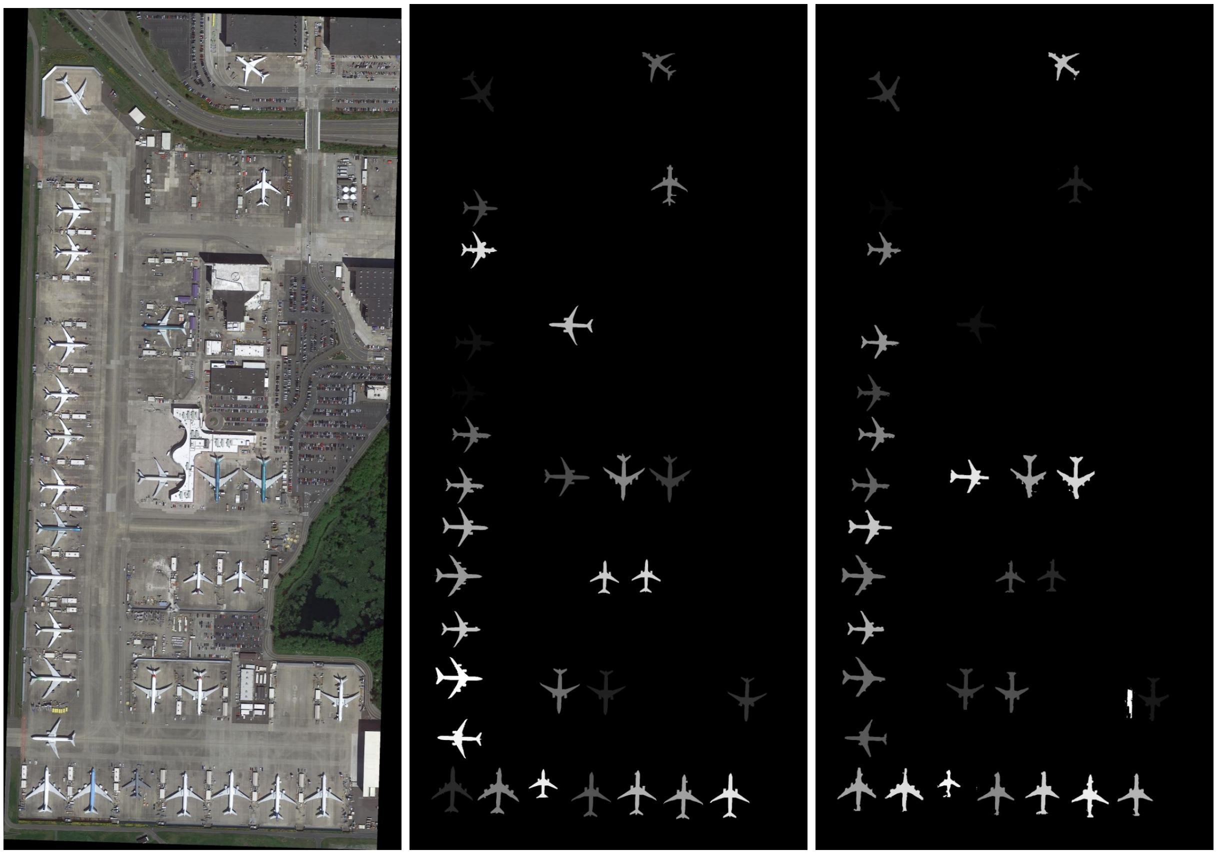
For IoU, I did an element-wise multiplication and counted the non-zero elements (for the intersection operation) . For the union I did element-wise addition and counted the non-zero elements. This is after setting all pixel values to 0 and 1 for both the predicted and ground truth. The average IoU for all instances was 0.90.

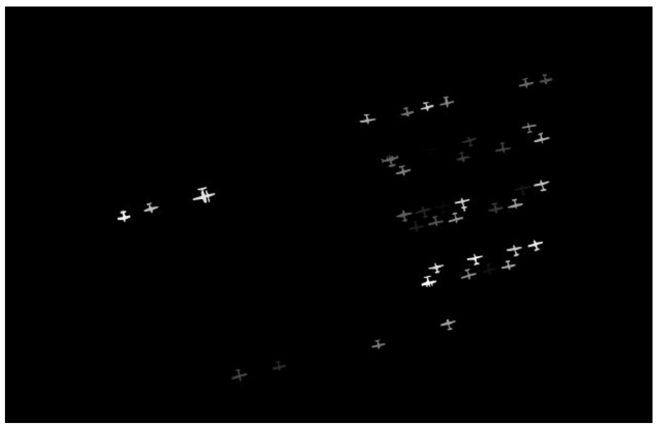
Below are samples of my models predictions on the training dataset, where the top image is the actual image, the second one is model's prediction and the third one is the ground truth mask:

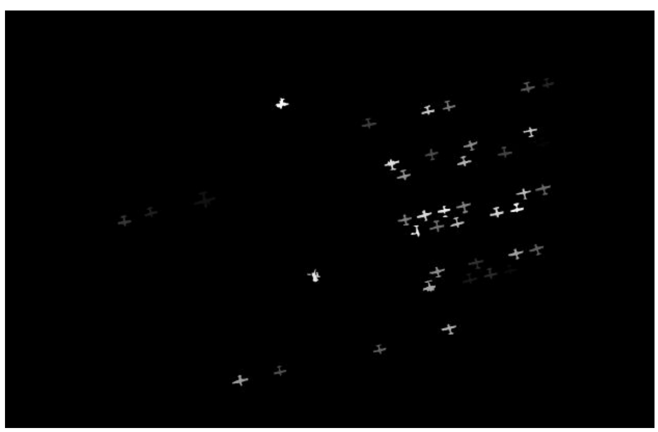


Part 3:

I submitted my results as **Papa Goose** on kaggle. My current score is 0.62312 on the public leaderboard. Here are some of the predicted masks from my model on the training dataset, the first one is the actual image, second one is the ground truth mask and the third one is the predicted mask:

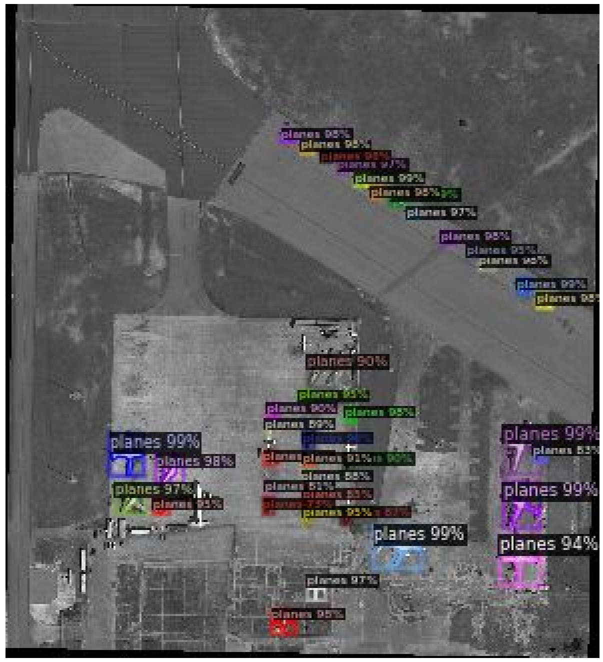


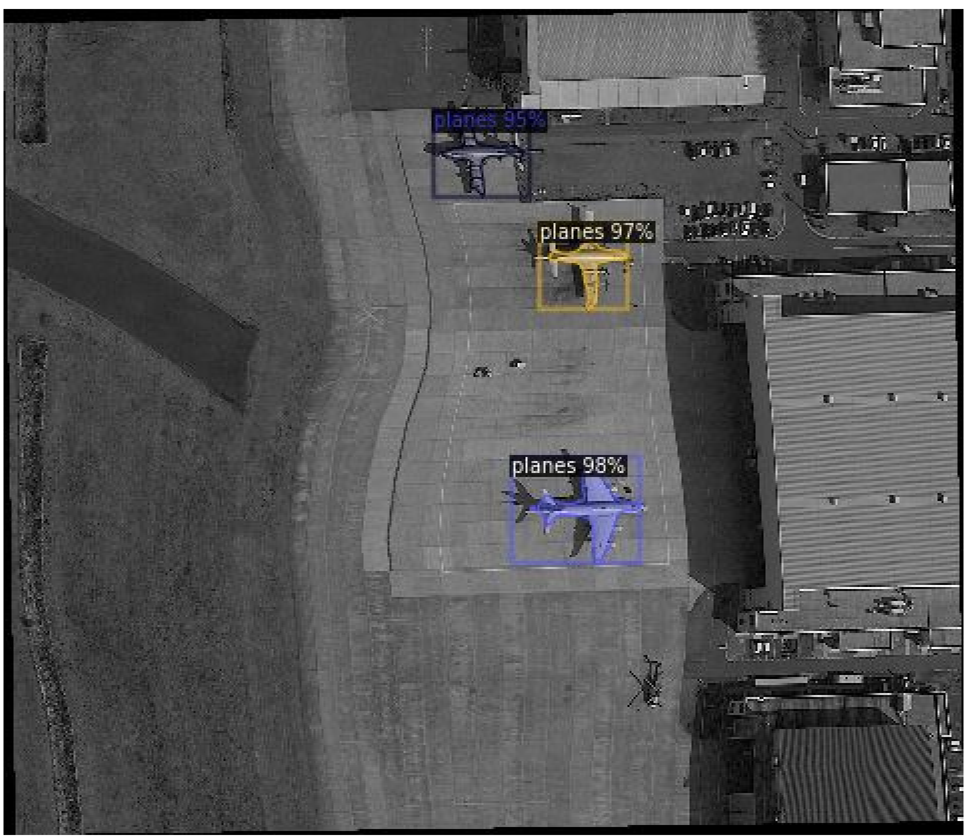




Part 4:

Visualizations:


planes 90%
planes 99%
planes 98%
planes 97%
planes 95%
planes 99%
planes 99%
planes 99%
planes 94%
planes 97%
planes 98%


planes 95%
planes 97%
planes 98%

This model performs way worse than the one used in part 1. Here are the AP scores for this model on the validation data:

BBox AP:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.236
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.476
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.211
 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 ] = 0.272
 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 ] = 0.014
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.110
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.292
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.215
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.336
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.649
[11/02 02:27:03 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 23.555 | 47.633 | 21.108 | 20.375 | 27.195 | 50.033 |
```

Segmentation AP:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.042
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.182
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.007
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.020
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.057
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.141
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.007
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.043
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.074
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.042
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.095
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.198
[11/02 02:27:04 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl   |
|:-----:|:------:|:------:|:-----:|:-----:|:------:|
| 4.175 | 18.161 | 0.721  | 2.049 | 5.745 | 14.067 |
```