CMPT412 (Fall 2021): Project 3
Object Detection, Semantic Segmentation, and Instance Segmentation
Keenan Byun 301381767

Part1

1.configs and modifications used:
Pre-Trained Model = Faster R-CNN R 101 FPN
LR = 0.025
Iteration = 500
LR decay = 0.1 after 200 iterations
Images per Batch = 2
Batch Size per Image = 128
AP50 = 41.429 with validation data set

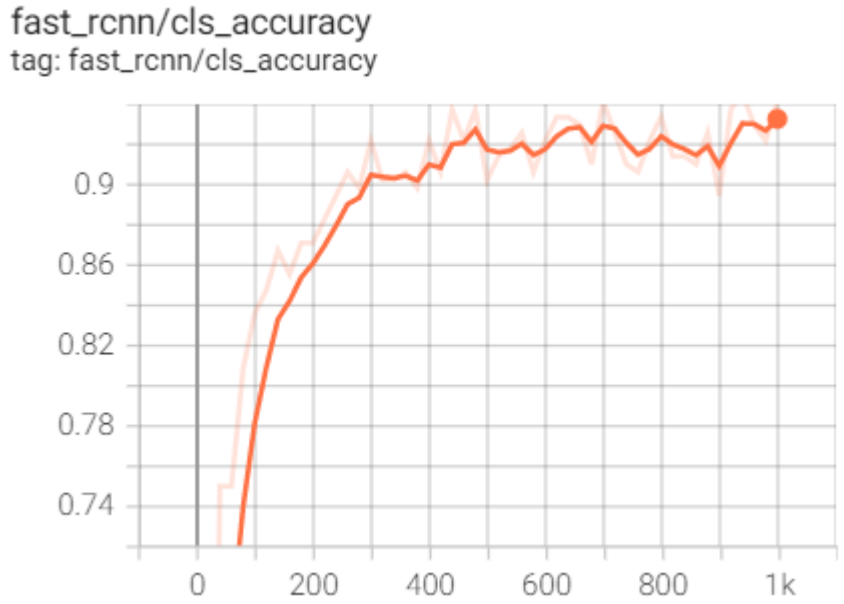
2.factors(improvement)
Picking a good learning rate significantly increases the AP50 because too low learning rate is too slow and too high learning rate converges its loss at high level.

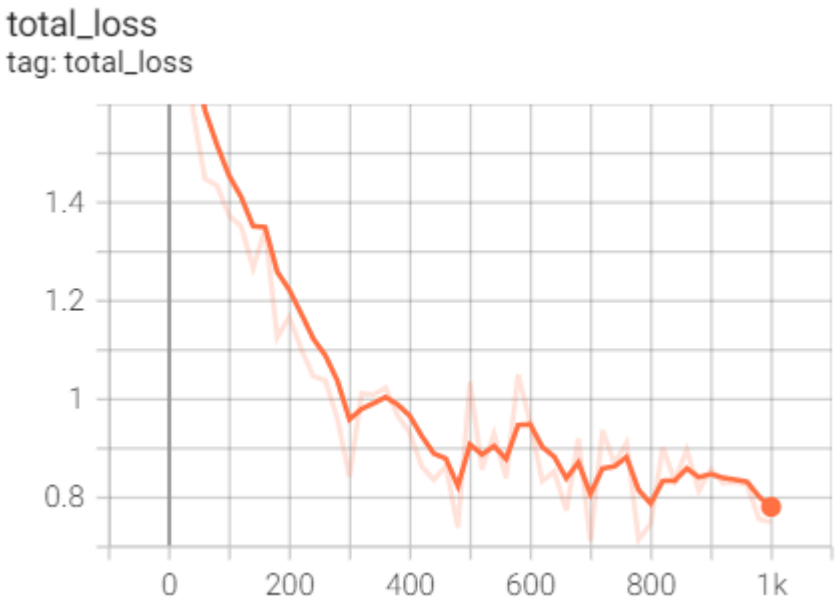
Reducing batch size has the effects of regularization as adding noise into the data set, which means it reduces the overfitting.
https://towardsdatascience.com/exploit-your-hyperparameters-batch-size-and-learning-rate-as-regularization-9094c1c99b55

Setting learning rate decay improves the model in speed aspect and accuracy aspect at the same time because with high learning rate in initial time, it makes model converge fast to some point, and decay adjusts the high learning rate to moderate one.

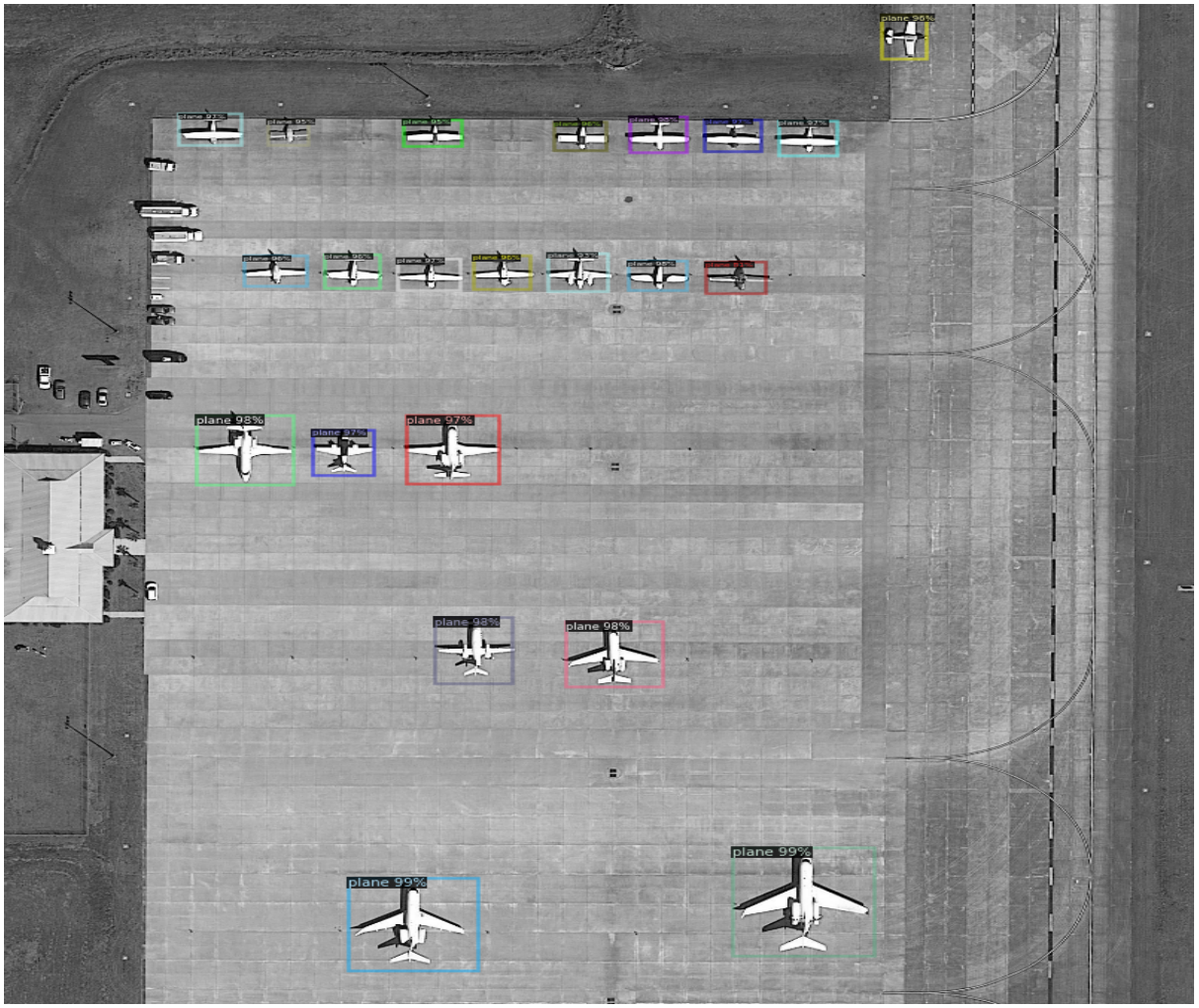
Choosing the best model was important too. RetinaNet and Faster R-CNN had similar processing time, but Faster R-CNN had better accuracy while RPN had the fastest processing time with the lowest accuracy.

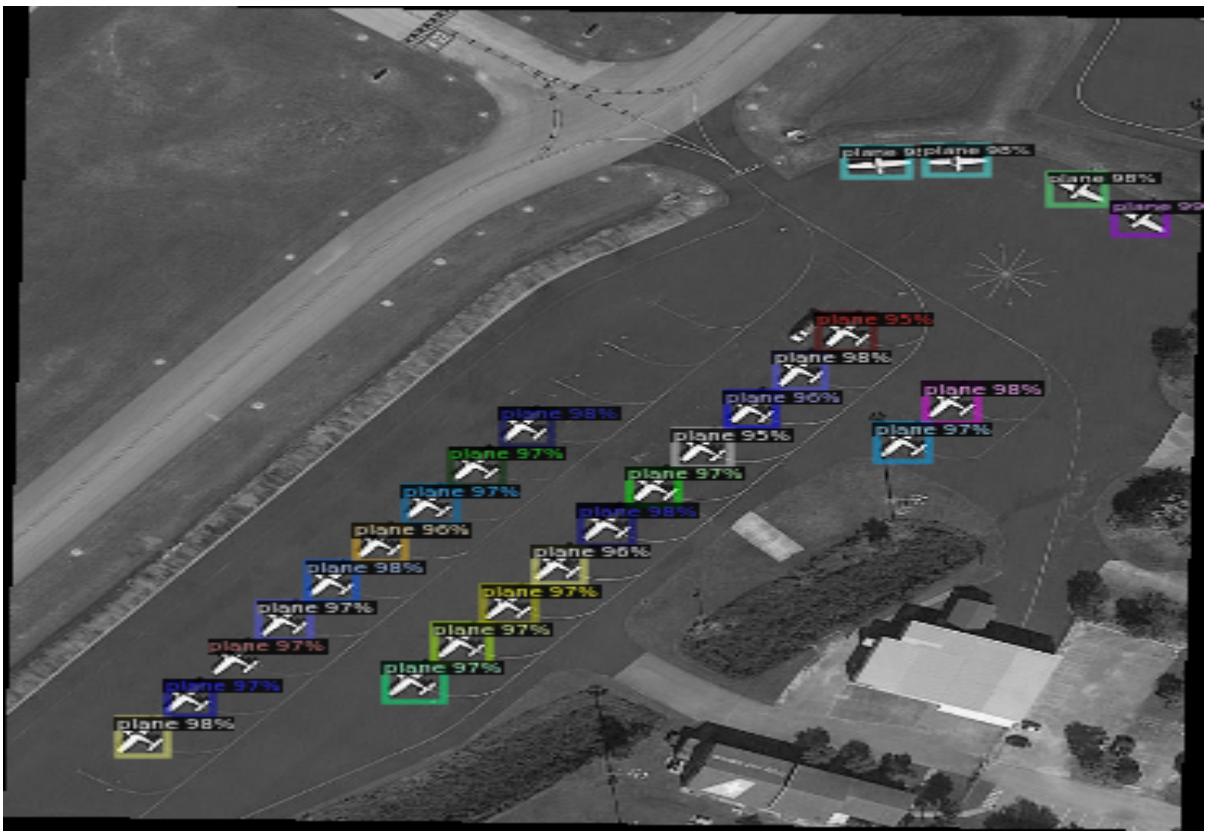
3.training loss and accuracy plot





4.visualization(test)




plane 95%
plane 98%
plane 96%
plane 98%
plane 95%
plane 97%
plane 98%
plane 97%
plane 97%
plane 97%
plane 98%
plane 96%
plane 98%
plane 96%
plane 97%
plane 97%
plane 97%
plane 97%
plane 97%
plane 98%

5.ablation study

RetinaNet vs Faster R-CNN

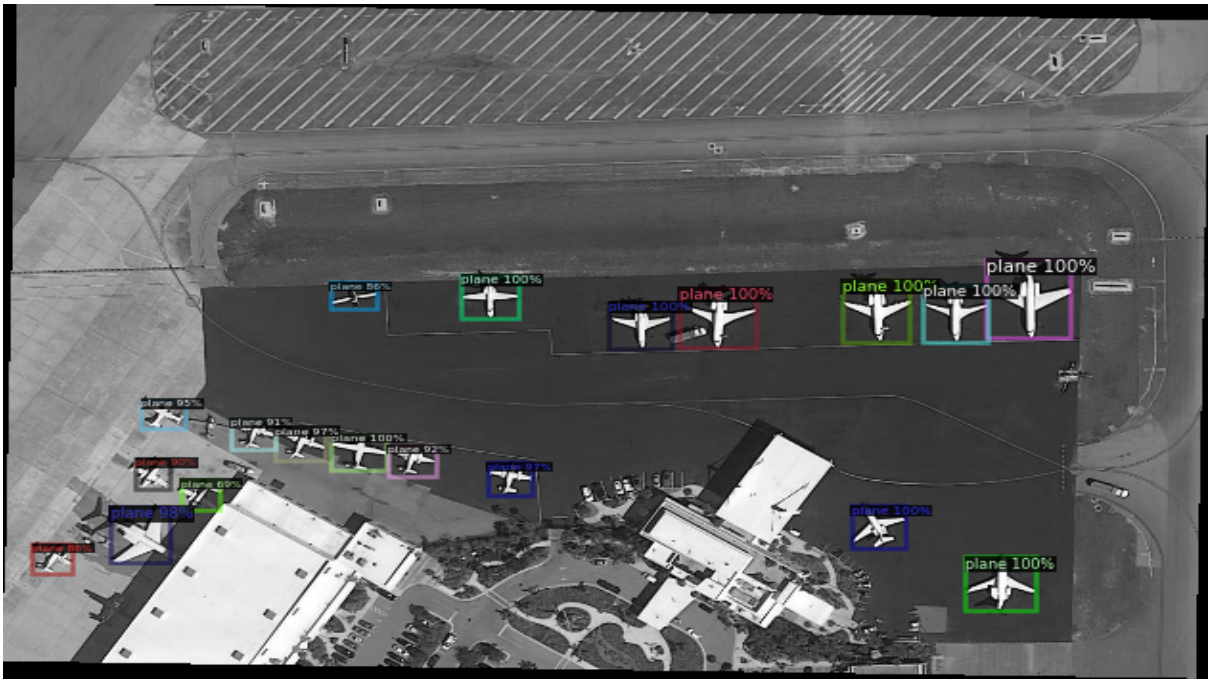
Performed benchmarking between the two models above. The models had the same configurations for fair comparison.

AP50(RetinaNet) = 19.375

AP50(Faster R-CNN) = 41.429

Visualized Comparison:

RetinaNet:



Faster R-CNN:

Analysis: Faster R-CNN tends to have more false positive detections than RetinaNet.

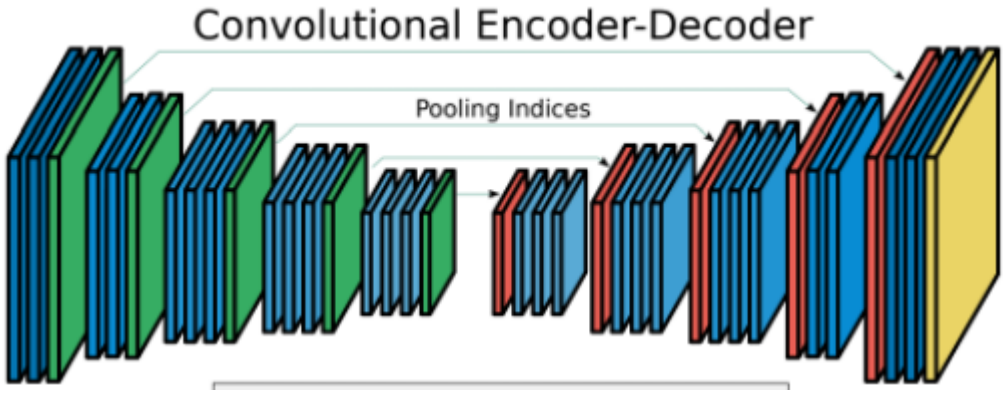
Part2

1.hyperparameter setting

Epoch = 15
Batch_Size = 4
LR = 0.007
Optimizer = AdamW(Weight_Decay = 0.00001)

2.network architecture

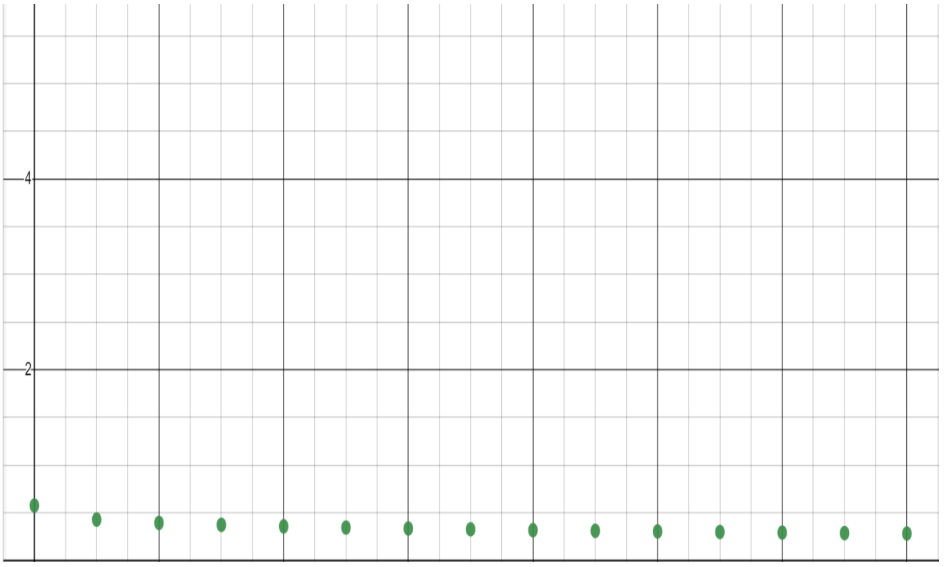
5 pooling layers(down) and 5 upsampling layers(up)
(conv down) + (conv down) + (conv conv down) + (conv conv down) + (conv conv down) + (up conv conv) + (up conv conv) + (up conv conv) + (up conv) + (up conv(no activation))
where:
conv = conv2d + Normalization + ReLu, down = conv2d + maxpool, up = conv2d + unpool
Deeper network could have better performance than too simple designs with high dimension channels; therefore, stacked more layers referencing the design of SegNet.



3.loss function and plot

For the loss function, the Dice_Loss(loss1) and Binary Cross Entropy Loss(loss2) are combined to evaluate total loss(loss = loss1 + loss2).

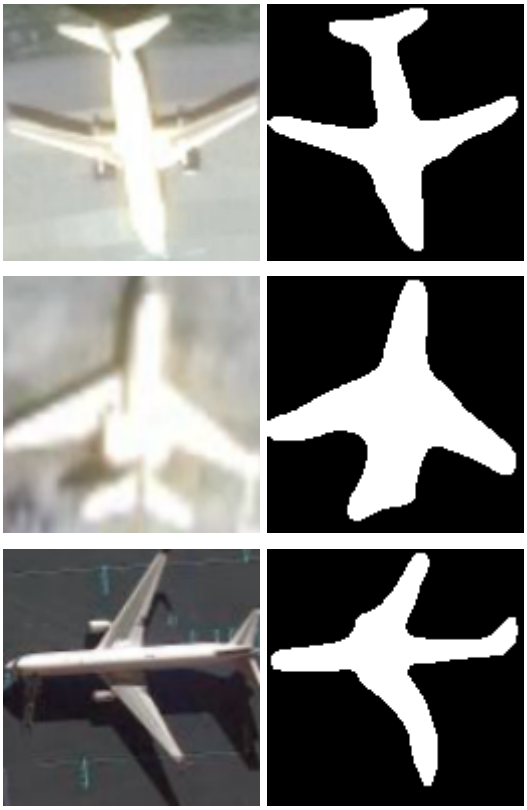
0	0.57417253
1	0.42750293
2	0.39216807
3	0.37241530
4	0.35643750
5	0.34375476
6	0.33301753
7	0.32473635
8	0.31597602
9	0.30849903
10	0.30271273
11	0.29685842
12	0.29078528
13	0.28552016
14	0.28019472



4.final mean IOU

Images: 6617, Mean IoU: 0.8122304971171077

5.visualization(test)

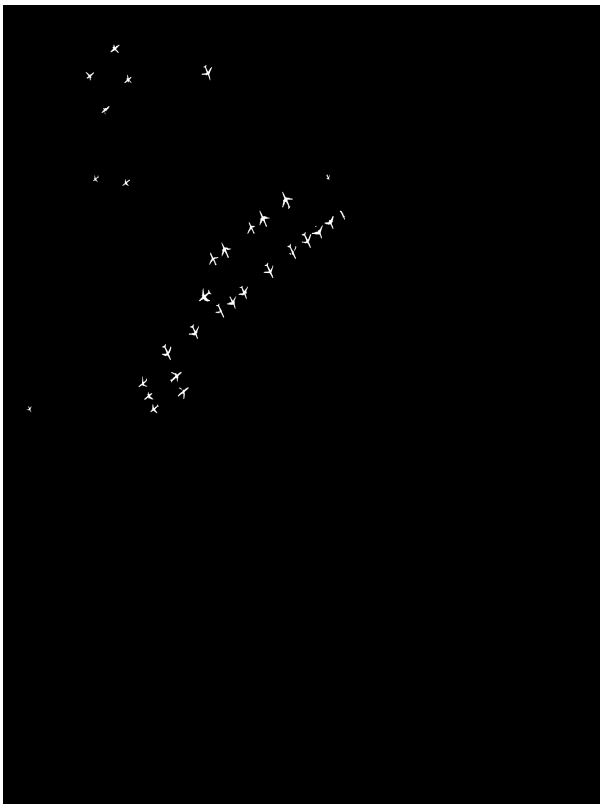


Part3

1.kaggle name and the best score

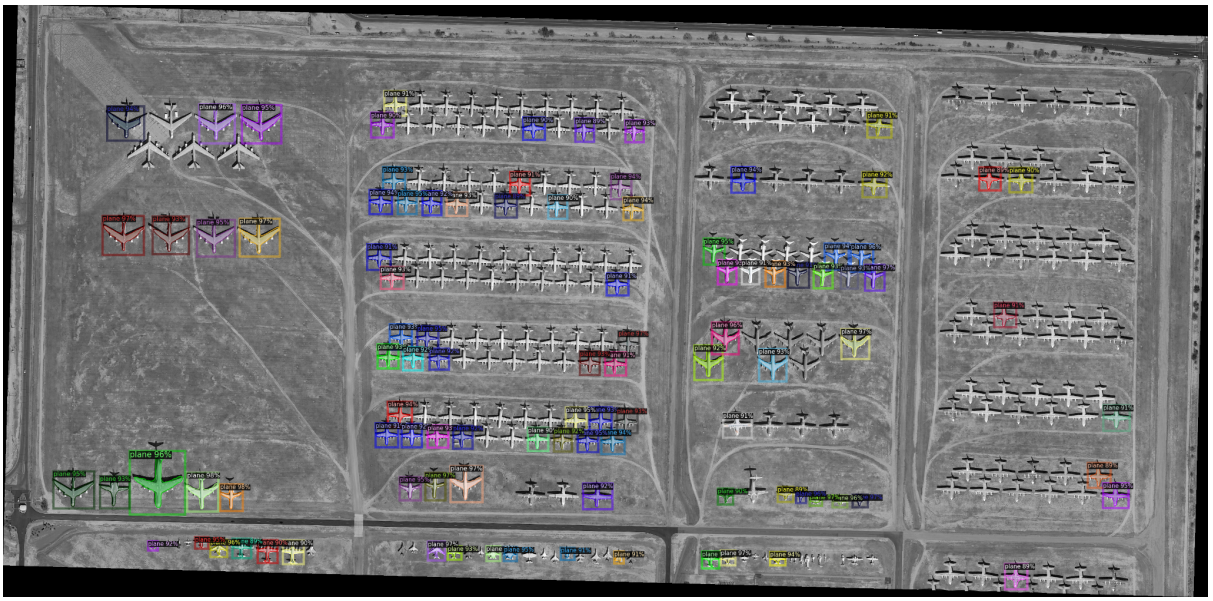
keenan byun: 0.32250, 23 submissions

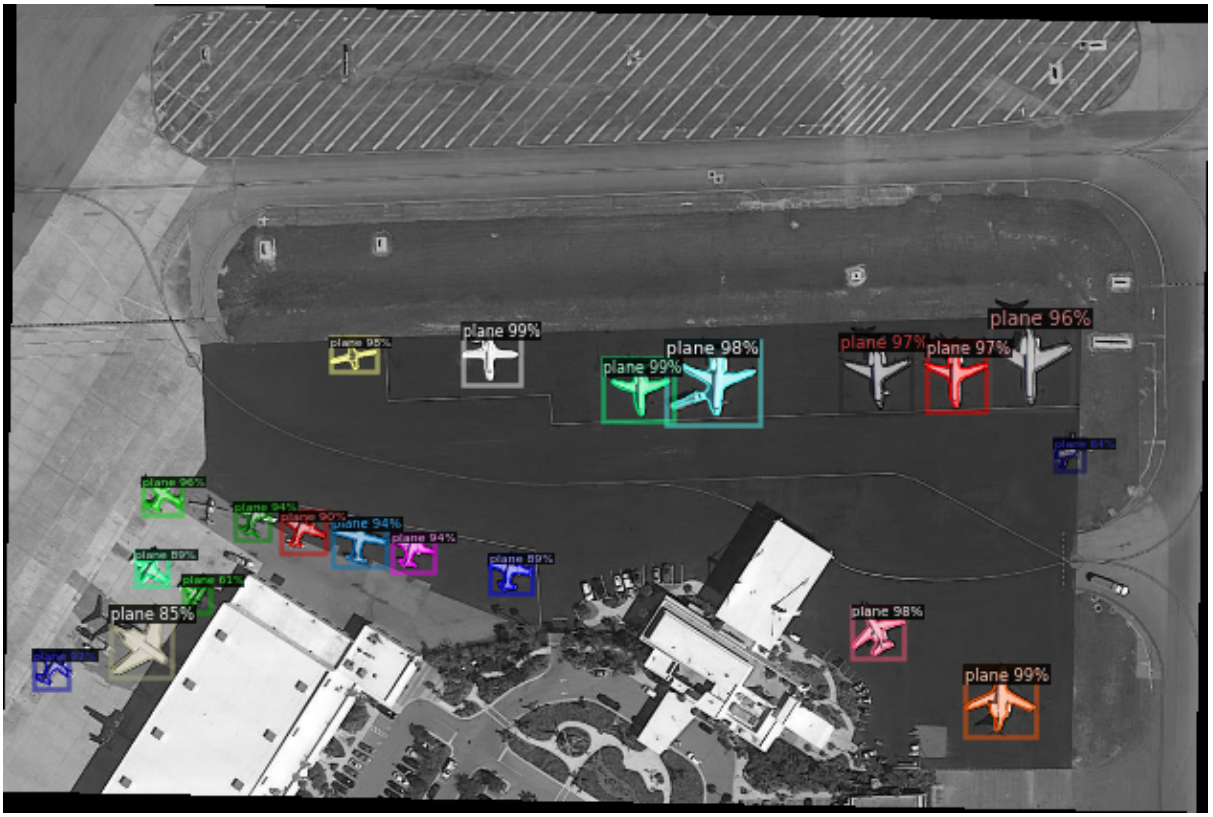
2.visualization(test)



Part4

1.visualization and evaluation





Evaluation:
Detection AP50 = 32.153
Segmentation AP50 = 16.035

2.differences from part3

My model had higher evaluation than the mask R-CNN base model(my model detection AP50 = 41.429 and segmentation = 0.32250). Probably, the configs like hyperparameters I used are more suitable to my model than the mask R-CNN. I guess I can get higher results with the mask R-CNN with different settings. However, for now, I think my model is better because of huge accuracy gaps.