

# **Project3 Report**

---

**Rongsheng Qian 301449387**

**Team: R.S Qian&Yuwen Jia&Isaac Ding**

**Names of group members: Rongsheng Qian, Isaac Ding, Yuwen Jia SFU**

**Using 3 free late day**

---

**Best accuracy: 0.42208**

# Part 1

## 1.1 List of the configs and modifications that you used.

### 1.1.1 Configs

```
1  cfg = get_cfg()
2  cfg.OUTPUT_DIR = "{}/output_custom{}".format(BASE_DIR)
3
4  cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/retinanet_R_101_FPN_3x.yaml"))
5  cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/retinanet_R_101_FPN_3x.yaml")
6
7  cfg.DATASETS.TRAIN = ("custom_train",)
8  cfg.DATASETS.TEST = ("custom_val",)
9  cfg.DATALOADER.NUM_WORKERS = 2
10
11 cfg.SOLVERIMS_PER_BATCH = 2
12 cfg.SOLVER.BASE_LR = 0.0001
13 cfg.SOLVER.MAX_ITER = 1000
14
15 cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
16 cfg.MODEL.ROI_HEADS.NUM_CLASSES = 5
```

### 1.1.2 Create 3 Dataloader (Data augmentation) (train 3 times using 3 CustomTrainer3)

```
1  class CustomDatasetMapper1(DatasetMapper):
2      def __init__(self, cfg, is_train=True):
3          super().__init__(cfg, is_train)
4          augmentations = []
5          if is_train:
6              augmentations.append(T.Resize((800, 800)))
7              augmentations.append(T.RandomRotation([-5, 5]))
8          self.augmentations = T.AugmentationList(augmentations)
9
10     def __call__(self, dataset_dict):
11         dataset_dict = super().__call__(dataset_dict)
12         return dataset_dict
13
14
15     class CustomDatasetMapper2(DatasetMapper):
16         def __init__(self, cfg, is_train=True):
17             super().__init__(cfg, is_train)
18             augmentations = []
19             if is_train:
20                 augmentations.append(T.Resize((800, 800)))
21                 augmentations.append(T.RandomFlip(prob=1))
22             self.augmentations = T.AugmentationList(augmentations)
```

```

23
24     def __call__(self, dataset_dict):
25         dataset_dict = super().__call__(dataset_dict)
26         return dataset_dict
27
28     class CustomDatasetMapper3(DatasetMapper):
29         def __init__(self, cfg, is_train=True):
30             super().__init__(cfg, is_train)
31             augmentations = []
32             if is_train:
33                 augmentations.append(T.Resize((800, 800)))
34
35             self.augmentations = T.AugmentationList(augmentations)
36
37         def __call__(self, dataset_dict):
38             dataset_dict = super().__call__(dataset_dict)
39             return dataset_dict
40
41
42     class CustomTrainer1(DefaultTrainer):
43         @classmethod
44         def build_train_loader(cls, cfg):
45             return build_detection_train_loader(cfg, mapper=CustomDatasetMapper1(cfg,
46 is_train=True))
47
48     class CustomTrainer2(DefaultTrainer):
49         @classmethod
50         def build_train_loader(cls, cfg):
51             return build_detection_train_loader(cfg, mapper=CustomDatasetMapper2(cfg,
52 is_train=True))
53
54     class CustomTrainer3(DefaultTrainer):
55         @classmethod
56         def build_train_loader(cls, cfg):
57             return build_detection_train_loader(cfg, mapper=CustomDatasetMapper3(cfg,
58 is_train=True))

```

### 1.1.3 Data Processing.

Divide an image into smaller blocks (800\*800) if the bounding box is less than 50% of its total area in this block, I drop it

```

1  def is_bbox_in_area(bbox_xywh, area_xyxy):
2      bbox_x1 = bbox_xywh[0]
3      bbox_y1 = bbox_xywh[1]
4      bbox_x2 = bbox_xywh[0] + bbox_xywh[2]
5      bbox_y2 = bbox_xywh[1] + bbox_xywh[3]
6
7      bbox = box(bbox_x1, bbox_y1, bbox_x2, bbox_y2)
8      area = box(area_xyxy[0], area_xyxy[1], area_xyxy[2], area_xyxy[3])

```

```

9     intersect = bbox.intersection(area)
10
11     intersect_area = intersect.area
12     bbox_area = bbox.area
13     threshold_area = 0.5 * bbox_area
14
15     if threshold_area > intersect_area:
16         return []
17
18     if not intersect.is_empty:
19         x1, y1, x2, y2 = intersect.bounds
20         w = x2 - x1
21         h = y2 - y1
22         return [x1,y1,w,h]
23     else:
24         return []

```

## 1.1.4 Add validation set

Set 19 images as validation set which is 10% of whole training set

## 1.2 Factors which helped improve the performance.

### 1.2.1 Increase the size of val set from 8 images to 19 images

The val accuracy increased from baseline 23% to 28%

```

+-----+-----+-----+-----+-----+-----+
| AP   | AP50  | AP75  | APs   | APm   | APl   |
+-----+-----+-----+-----+-----+-----+
| 28.494 | 51.242 | 29.399 | 13.506 | 37.452 | 53.231 |
[11/09 21:05:59 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category | AP      | category | AP      | category | AP      |
+-----+-----+-----+-----+-----+-----+
|          | nan    |          | nan    |          | nan    |
|          | nan    | plane   | 28.494 |          |          |
OrderedDict([('bbox',
              {'AP': 28.49429523609384,
               'AP50': 51.24230960124314,
               'AP75': 29.398519989052847,
               'APs': 13.505629791280432,
               'APm': 37.45163273304162,
               'APl': 53.23101583029035,
               'AP-': nan,
               'AP-plane': 28.49429523609384})))

```

## 1.2.2 Data Processing.

The val accuracy increased from baseline 29% to 43%

**1.2.2.1 if the bounding box is less than 50% of its total area in this block, I drop it. Comparing with the bounding box is less than 30% of its total area in this block**

less than 50%

 pred.csv  
Complete · 2d ago 0.31508

less than 30%

 pred.csv  
Complete · 2d ago 0.26631

This factor can increase the accuracy from 26% to 31%.

**1.2.3 Using `retinanet_R_101_FPN_3x.yaml` model and pre-train instead of `faster_rcnn_R_101_FPN_3x.yaml`**

This factors can increase val accuracy from 60% to 63%

**1.2.4 Create 3 Dataloader (Data augmentation) (original images, filping images, rotation images)**

This factors can increase val accuracy from 63% to 67%

```
AP | AP50 | AP75 | APs | APm | APl |
:---:|:---:|:---:|:---:|:---:|:---:
63.779 | 88.482 | 77.008 | 69.716 | nan | nan

[11/11 06:37:59 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[11/11 06:37:59 d2.evaluation.coco_evaluation]: Per-category bbox AP:
category | AP | category | AP | category | AP |
:-----:|:---:|:-----:|:---:|:-----:|:---:
nan | nan | nan | nan | nan | nan
nan | plane | 63.779 | | | 

OrderedDict([('bbox',
    {'AP': 63.77916105122618,
     'AP50': 88.48155756625302,
     'AP75': 77.0081022631427,
     'APs': 69.71618395818514,
     'APm': nan,
     'APl': nan,
     'AP-': nan,
     'AP-plane': 63.77916105122618})))
```

```

| AP | AP50 | AP75 | APs | APm | AP1 |
|-----|-----|-----|-----|-----|-----|
| 66.923 | 91.258 | 82.316 | 72.591 | nan | nan |
[11/09 22:49:24 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[11/09 22:49:24 d2.evaluation.coco_evaluation]: Per-category bbox AP:
category | AP | category | AP | category | AP |
|-----|-----|-----|-----|-----|-----|
| nan | nan | nan |
| nan | plane | 66.923 | nan |
OrderedDict([('bbox',
    {'AP': 66.92288550535133,
     'AP50': 91.2584717138306,
     'AP75': 82.31615013389118,
     'APs': 72.59099451522125,
     'APm': nan,
     'AP1': nan,
     'AP-': nan,
     'AP-plane': 66.92288550535133})))

```

## 1.3 Final plot for total training loss and accuracy.

### 1.3.1 final accuracy:

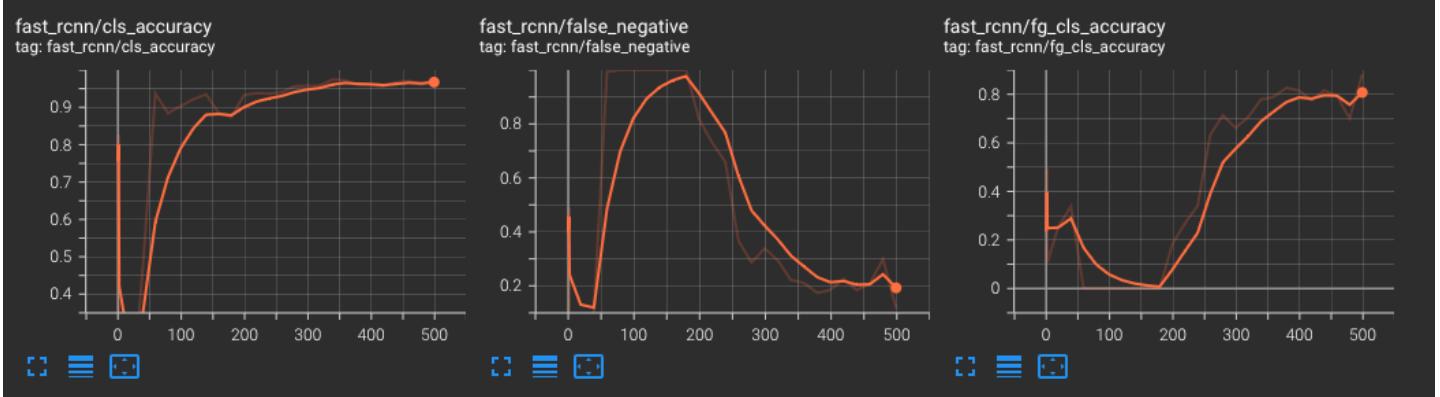
```

| AP | AP50 | AP75 | APs | APm | AP1 |
|-----|-----|-----|-----|-----|-----|
| 66.923 | 91.258 | 82.316 | 72.591 | nan | nan |
[11/09 22:49:24 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[11/09 22:49:24 d2.evaluation.coco_evaluation]: Per-category bbox AP:
category | AP | category | AP | category | AP |
|-----|-----|-----|-----|-----|-----|
| nan | nan | nan |
| nan | plane | 66.923 | nan |
OrderedDict([('bbox',
    {'AP': 66.92288550535133,
     'AP50': 91.2584717138306,
     'AP75': 82.31615013389118,
     'APs': 72.59099451522125,
     'APm': nan,
     'AP1': nan,
     'AP-': nan,
     'AP-plane': 66.92288550535133})))

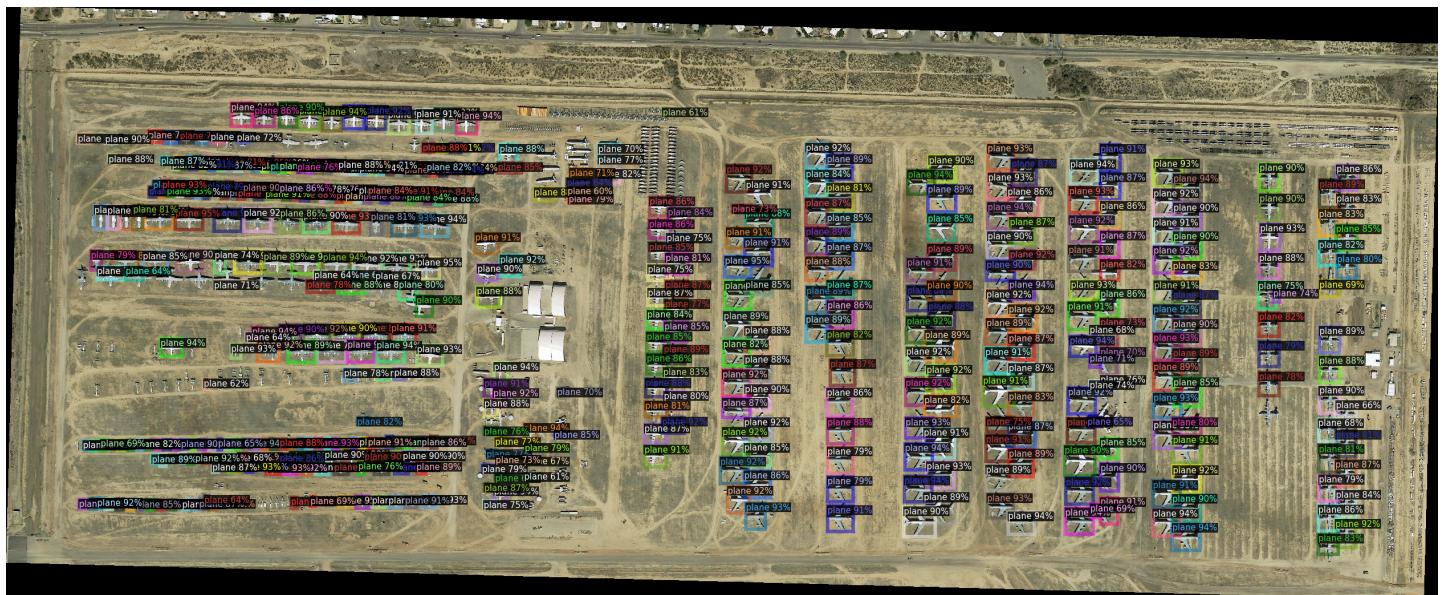
```

### 1.3.2 Plot





## 1.4 The visualization of 3 samples from the test set and the predicted results.





## 1.5 Ablation Study

---

### 1.5.1 Divide an image into smaller blocks (800\*800) if the bounding box is less than 50% of its total area in this block drop it

This factor can increase the accuracy from 28% to 64%.

The small plane can be detected more easily. The visualization is shown below. The origin model can not detect the small plane in this sample, however the model after adding this factor can detect it with a acceptable precision.

Dropping the bounding box which is less than 50% of its total area can help model to recognize how the plane looks like rather than predicted a building as a plane.

Without the Data Processing:



AP	AP50	AP75	APs	APm	APl
28.494	51.242	29.399	13.506	37.452	53.231

[11/09 21:05:59 d2.evaluation.coco\_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
	nan		nan		nan
	nan	plane	28.494		

OrderedDict([('bbox',

```

{'AP': 28.49429523609384,
 'AP50': 51.24230960124314,
 'AP75': 29.398519989052847,
 'APs': 13.505629791280432,
 'APm': 37.45163273304162,
 'APl': 53.23101583029035,
 'AP-': nan,
 'AP-plane': 28.49429523609384})))
```

With the Data Processing:



AP	AP50	AP75	APs	APm	APl
63.779	88.482	77.008	69.716	nan	nan

[11/11 06:37:59 d2.evaluation.coco\_evaluation]: Some metrics cannot be computed and is shown as NaN.

[11/11 06:37:59 d2.evaluation.coco\_evaluation]: Per-category bbox AP:

category	AP	category	AP	category	AP
nan	nan	nan	nan	nan	nan
nan	plane	63.779			

```
OrderedDict([('bbox',
    {'AP': 63.77916105122618,
     'AP50': 88.48155756625302,
     'AP75': 77.0081022631427,
     'APs': 69.71618395818514,
     'APm': nan,
     'APl': nan,
     'AP-': nan,
     'AP-plane': 63.77916105122618})))
```

## 1.5.2 Data augmentation

The image visualization result doesn't have the obvious difference.

However this factors can increase val accuracy from 63% to 67%

Data augmentation can increase the training set and let the network learn more features, which can increase the subtability of model in different kinds of images.

```

| AP | AP50 | AP75 | APs | APm | APl |
|-----|-----|-----|-----|-----|-----|
| 63.779 | 88.482 | 77.008 | 69.716 | nan | nan |
[11/11 06:37:59 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[11/11 06:37:59 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category | AP | category | AP | category | AP |
|-----|-----|-----|-----|-----|-----|
| nan | nan | nan | nan | nan |
| nan | plane | 63.779 | | |
OrderedDict([('bbox',
    {'AP': 63.77916105122618,
     'AP50': 88.48155756625302,
     'AP75': 77.0081022631427,
     'APs': 69.71618395818514,
     'APm': nan,
     'APl': nan,
     'AP-': nan,
     'AP-plane': 63.77916105122618})))

```

```

| AP | AP50 | AP75 | APs | APm | APl |
|-----|-----|-----|-----|-----|-----|
| 66.923 | 91.258 | 82.316 | 72.591 | nan | nan |
[11/09 22:49:24 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[11/09 22:49:24 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category | AP | category | AP | category | AP |
|-----|-----|-----|-----|-----|-----|
| nan | nan | nan | nan | nan |
| nan | plane | 66.923 | | |
OrderedDict([('bbox',
    {'AP': 66.92288550535133,
     'AP50': 91.2584717138306,
     'AP75': 82.31615013389118,
     'APs': 72.59099451522125,
     'APm': nan,
     'APl': nan,
     'AP-': nan,
     'AP-plane': 66.92288550535133})))

```

## Part 2

### 2.1 Any hyperparameter settings I used

```

1 num_epochs = 70
2 batch_size = 4
3 learning_rate = 0.01
4 weight_decay = 1e-5

```

### 2.2 The final architecture of my network

#### 2.2.1 Modification

Increasing channels layers using conv2d: (3,16), (16, 32), (64,128), (128,256), (256,512)

Repeated layers (in\_channel == out\_channel) using conv2d after (16,32), (32,64), (64,128), (128,256), there isn't repeated layer after (3,16)

Added fc layer between the encoder and decoder

## 2.2.2 explain the reason for each modification

### Repeating the CONV with the same number of input channels and output channels

Each repeating convolutional layer can capture different features, which can let network learn more complex features. It works when the train data is limited, which can mitigate overfitting issues

It can increase the depth of the network and add more non-linear part in network.

## 2.2.3 Final architecture

```
1 class MyModel(nn.Module):
2     def __init__(self):
3         super(MyModel, self).__init__()
4
5         # Encoder
6
7         self.input_conv = conv(3, 16)
8         self.down = nn.Sequential(
9             down(16, 32),
10            conv(32, 32),
11            conv(32, 32),
12
13            down(32, 64),
14            conv(64, 64),
15            conv(64, 64),
16
17            down(64,128),
18            conv(128, 128),
19            conv(128, 128),
20
21            down(128,256),
22            conv(256, 256),
23            conv(256, 256),
24
25            down(256,512)
26        )
27
28        self.fc_net = nn.Sequential(
29            nn.Linear(512*4*4, 512*4*4),
30            nn.BatchNorm1d(512*4*4),
31            nn.ReLU(inplace=True)
32        )
33
```

```

34     # Decoder
35
36     self.up = nn.Sequential(
37         up(512, 256),
38         conv(256, 256),
39         conv(256, 256),
40
41         up(256, 128),
42         conv(128, 128),
43         conv(128, 128),
44
45         up(128, 64),
46         conv(64, 64),
47         conv(64, 64),
48
49         up(64, 32),
50         conv(32, 32),
51         conv(32, 32),
52
53         up(32, 16)
54     )
55     self.output_conv = conv(16, 1, False) # ReLu activation is removed to keep the logits
for the loss function
56
57
58     def forward(self, input):
59         y = self.input_conv(input)
60         y = self.down(y)
61
62         b,c,h,w = y.size(0),y.size(1),y.size(2),y.size(3)
63         y = y.view(b, -1)
64         y = self.fc_net(y)
65         y = y.view(b,c,h,w)
66
67         y = self.up(y)
68         output = self.output_conv(y)
# print(output.shape)
69         return output
70

```

## 2.3 Report the loss functions that I used and the plot the total training loss of the training procedure

---

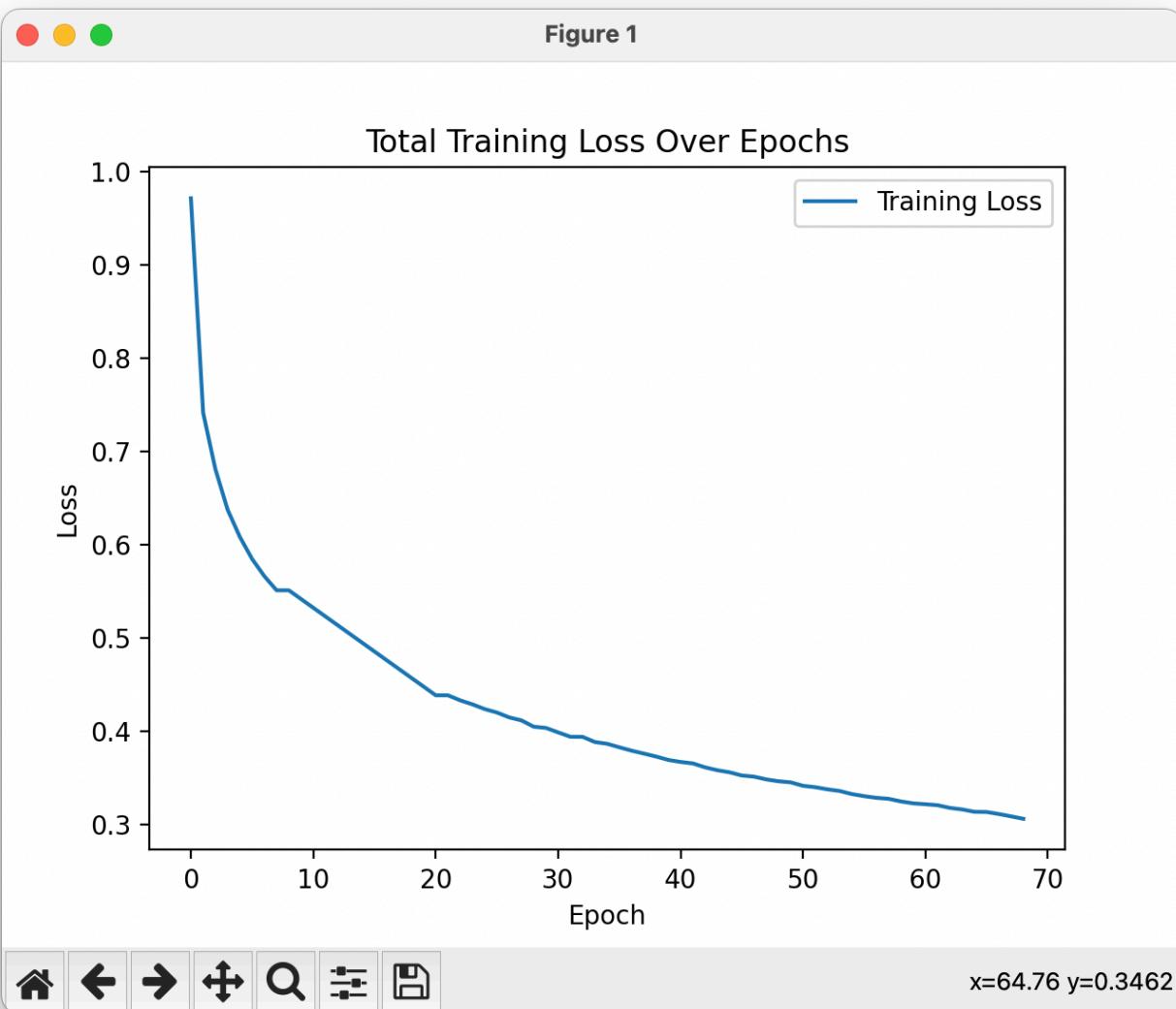
### 2.3.1 Loss function I used

```
1 crit = nn.BCEWithLogitsLoss() # Define the loss function
2 optim = torch.optim.SGD(model.parameters(), lr=learning_rate, weight_decay=weight_decay)
```

### 2.3.2 training loss

My training loss is 3 times than the original one is because I did data augmentation inside the for loop, which have orginal image, h\_filp image and v\_filp image

```
1 loss_list =
2 [0.9716119170188904, 0.7413926720619202, 0.6809871792793274, 0.6373889446258545, 0.6082096695899963, 0
3 .5845683217048645, 0.5660731196403503, 0.5509427785873413,
4 0.55094278, 0.54155553, 0.53216827, 0.52278102, 0.51339377, 0.50400652, 0.49461927, 0.48523201, 0.475844
5 76, 0.46645751, 0.45707026, 0.447683, 0.43829575,
6 0.43829575181007385, 0.4328637719154358, 0.428557425737381, 0.42350563406944275, 0.4198061525821686,
7 0.41460368037223816, 0.41136375069618225, 0.4045928120613098,
8 0.4032733142375946, 0.39848583936691284, 0.39379405975341797, 0.3937068283557892, 0.3881424069404602
9 , 0.3862606883049011, 0.38249462842941284,
0.37887489795684814, 0.3757827877998352, 0.3725910782814026, 0.36895090341567993,
0.3667604923248291, 0.36521628499031067, 0.36099541187286377, 0.35789838433265686,
0.35567939281463623, 0.35227134823799133, 0.35101622343063354, 0.3480479121208191, 0.346030175685882
57, 0.3448733687400818, 0.3411721885204315, 0.3396941125392914,
0.3374038338661194, 0.33560723066329956, 0.33230581879615784,
0.33006858825683594, 0.3282719850540161, 0.3272158205509186, 0.32439732551574707,
0.32234859466552734, 0.3214151859283447, 0.3203503489494324,
0.3174724280834198, 0.31584280729293823, 0.3132993280887604, 0.31308993697166443,
0.31090065836906433, 0.3083840012550354, 0.3057791590690613]
```



## 2.4 Report the final mean IoU of my model.

```
#images: 93, Mean IoU: 0.7536434492757244
```

## 2.5 Visualize 3 images from the test set and the corresponding predicted masks.



## Part3

---

### 3.1 Names of my group members

---

Team Member: Rongsheng Qian & Yuwen Jia & Isaac Ding

Names of group members: Rongsheng Qian, Isaac Ding, Yuwen Jia

### 3.2 Report the best score

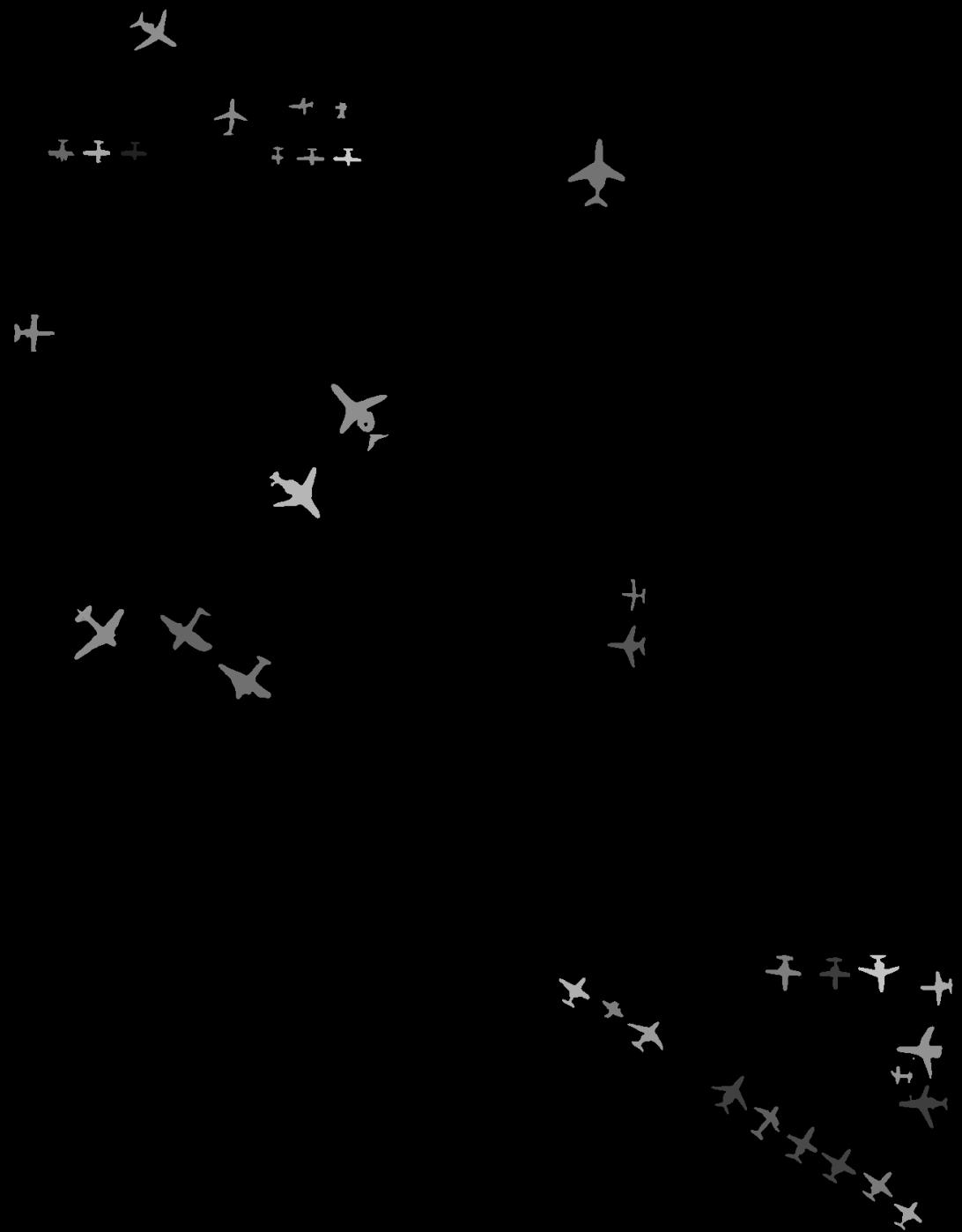
---

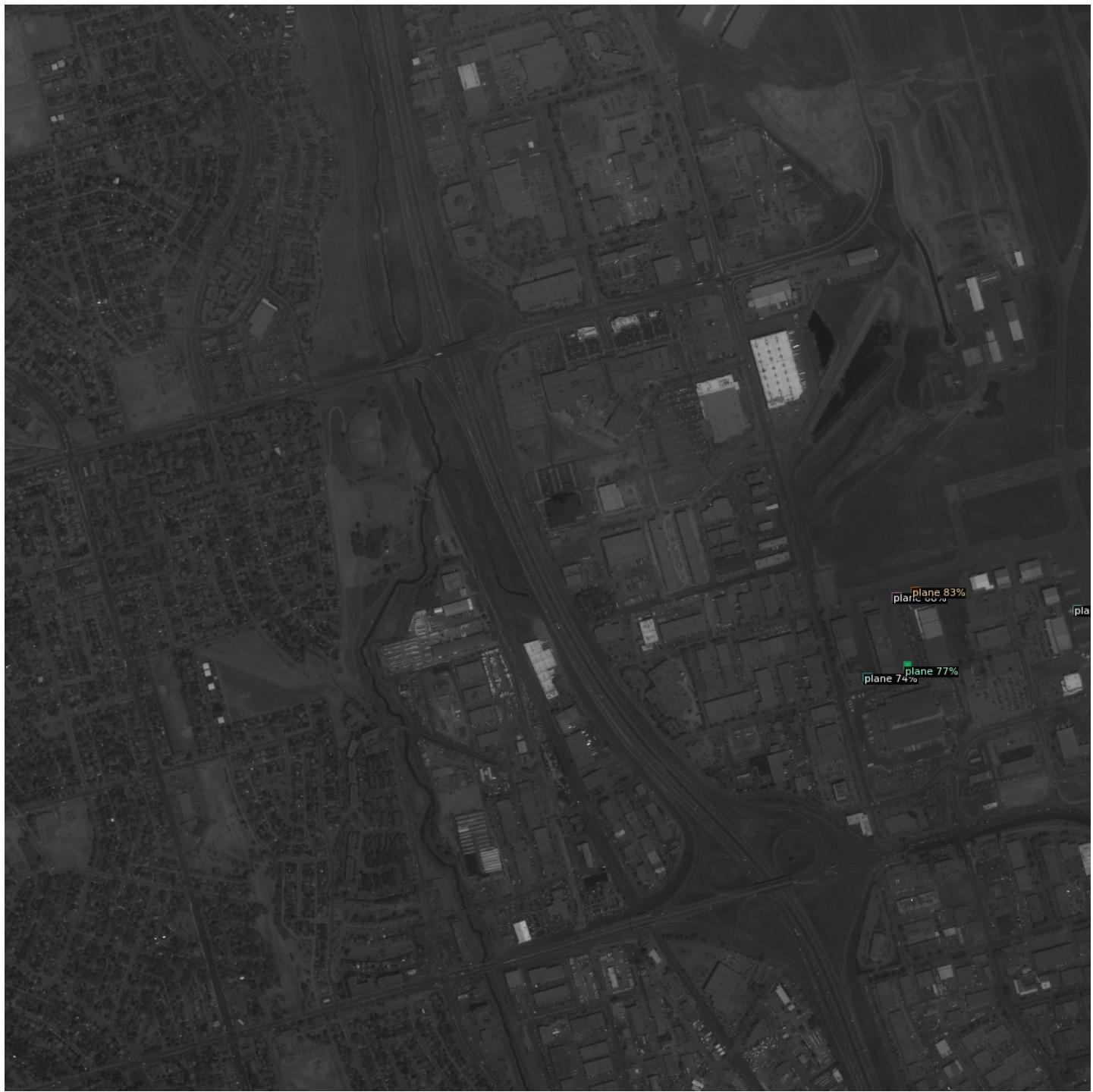
0.42208

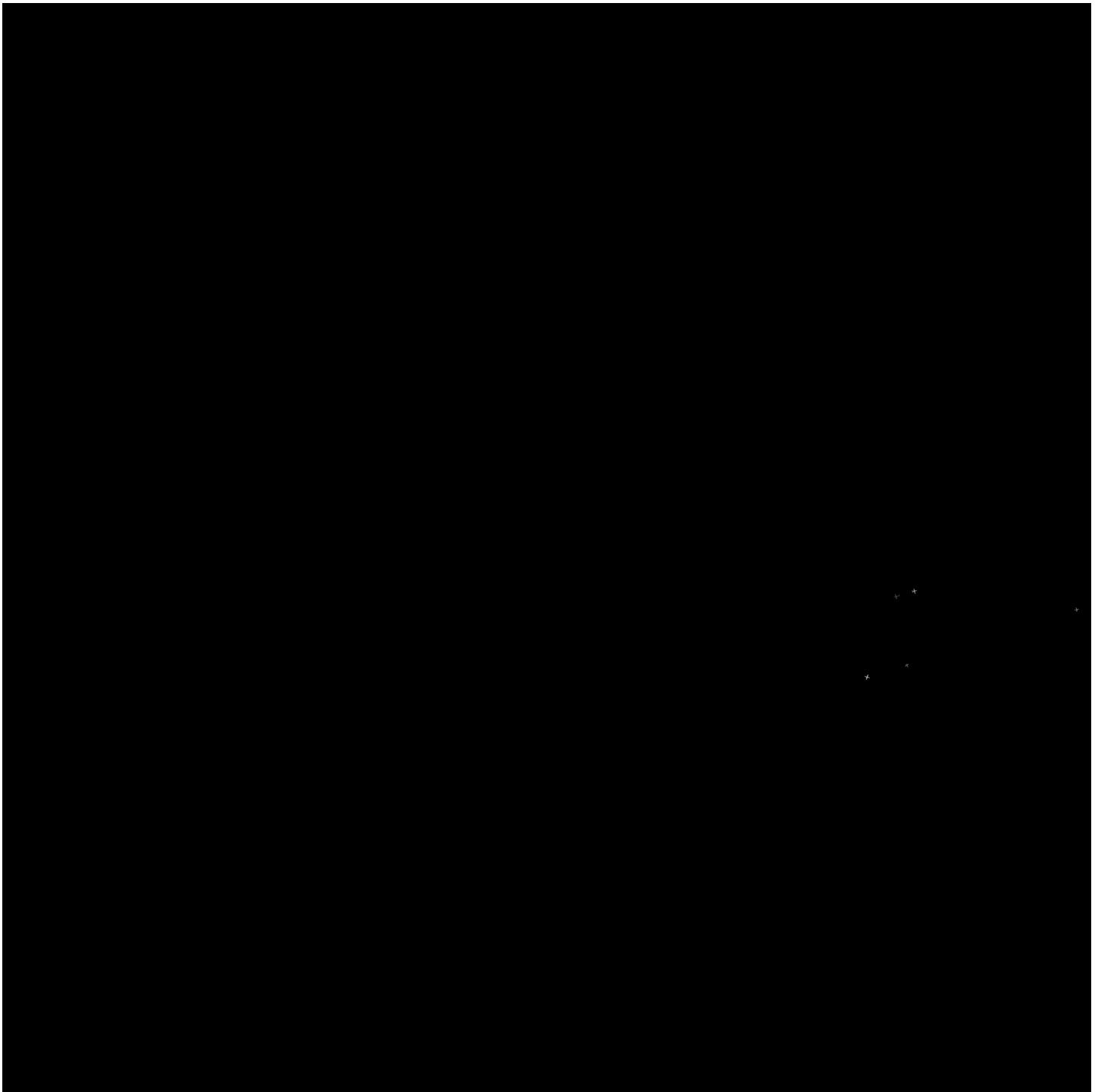
### 3.3 The visualization of results for 3 random samples from the test set.

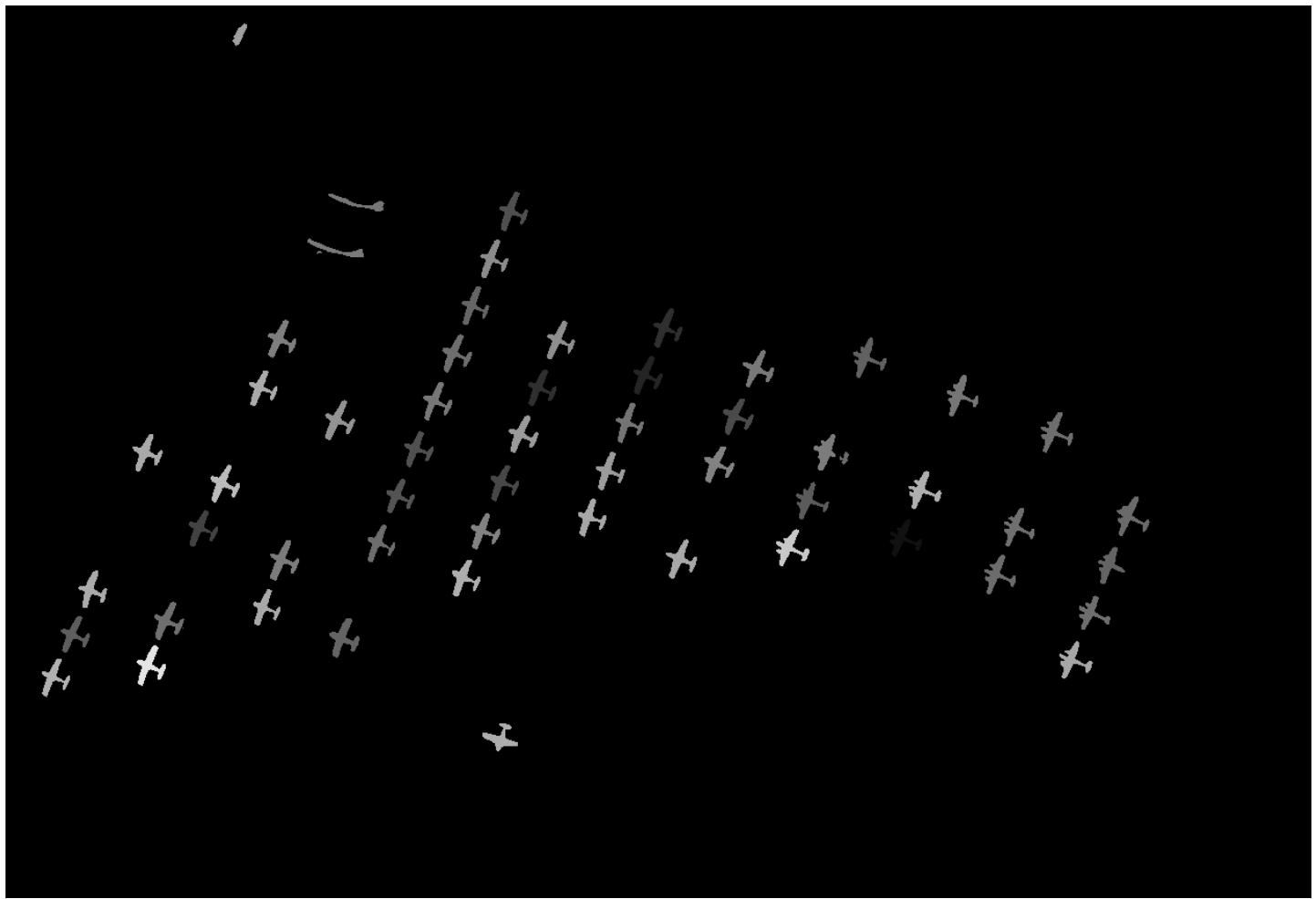
---





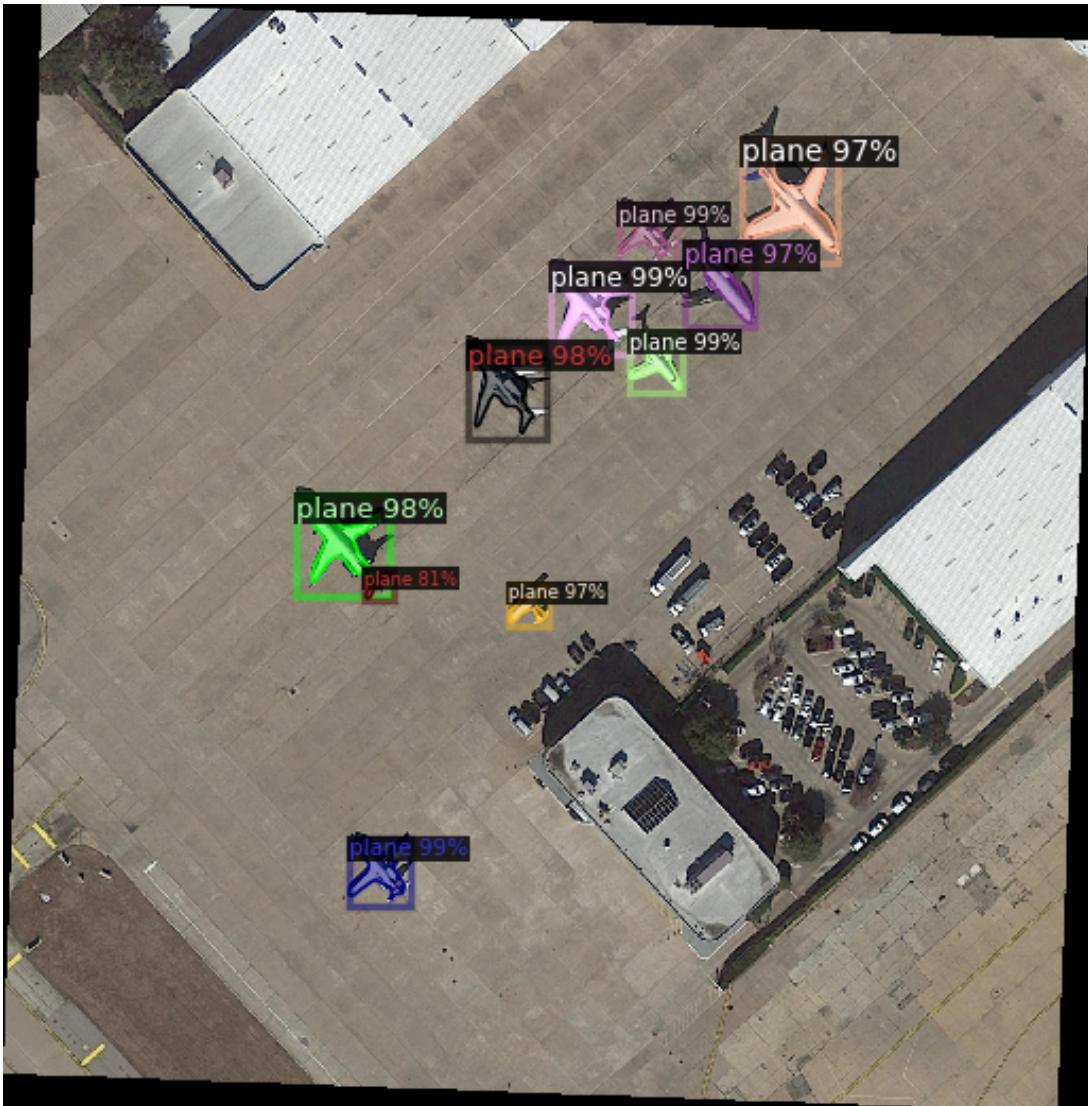






## Part4

4.1 The visualization and the evaluation results similar to Part 1.







**4.2 Explain the differences between the results of Part 3 and Part 4**

---

```
OrderedDict([('bbox',
    {'AP': 28.398225937281584,
     'AP50': 47.84622204366393,
     'AP75': 30.779133903503457,
     'APs': 11.893842234937116,
     'APm': 38.731526537111584,
     'APl': 50.17012741157468,
     'AP-': nan,
     'AP-plane': 28.398225937281584}),
   ('segm',
    {'AP': 8.453292716889878,
     'AP50': 30.151730837392794,
     'AP75': 0.3140860304517846,
     'APs': 3.0730235536909922,
     'APm': 10.329902541380156,
     'APl': 24.477706032394554,
     'AP-': nan,
     'AP-plane': 8.453292716889878})))
```

## 4.2.1 Training & inference time

Its training & inference time is potentially slower to converge than models focused solely on object detection due to its dual-task nature.

## 4.2.2 Flexibility

Part 3 model have more flexibility than part 4, which can adapt model to specific task requirements. For example, the mask-rcnn cannot find the plane in last sample but the combination of RetinaNet and FCN can do it by adding post-processing and pre-processing.

## 4.2.3 Segmentation

The result of segmentation in part4 seems worse than part3, which cannot recognize the whole plane. Its Ap of segm only have 8.4 which is lower than the part 3