**CMPT 412**

**Project 3**

**Object Detection, Semantic Segmentation, and**

**Instance Segmentation**

Instructor : Yasutaka Furukawa

Name: Kaikun Fang

Student ID: 301416542

**Note: Use 3 free late-day**

**Note: Use 3 free late-day**

**Part 1: Object Detection**

1. **Configs and Modifications:**

The learning rate and the number of iterations have been modified:

文本

描述已自动生成

**2. Factors:**

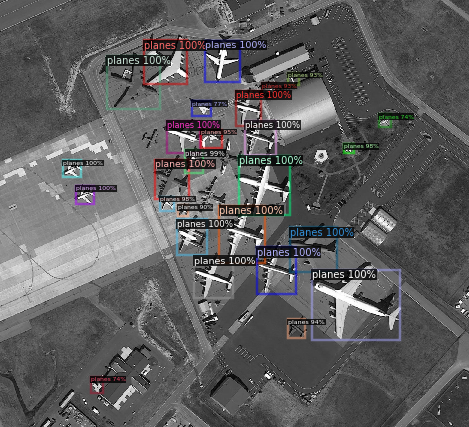
I first set the number of iterations at 500, and then set the learning rate at 0.0003 through several tests. Then I adjusted the number of iterations and found that the best accuracy could be achieved at 3000 without wasting training cost. At the end, get AP50 = 62.564

图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成**3. Final plot**

**4. Visualization**



**5. Ablation study:**

I performed ablation study on the number of iterations. I leave all other parameters unchanged and adjust only the number of iterations.

* MAX\_ITER = 500

| AP | AP50 | AP75 | APs | APm | APl |

|:------:|:------:|:------:|:------:|:------:|:------:|

| 25.631 | 47.240 | 25.758 | 17.370 | 33.062 | 52.314 |

* MAX\_ITER = 2000

| AP | AP50 | AP75 | APs | APm | APl |

|:------:|:------:|:------:|:------:|:------:|:------:|

| 37.804 | 58.323 | 42.507 | 26.755 | 45.778 | 71.471 |

* MAX\_ITER = 3000

| AP | AP50 | AP75 | APs | APm | APl |

|:------:|:------:|:------:|:------:|:------:|:------:|

| 39.560 | 62.564 | 44.099 | 29.556 | 47.392 | 71.334 |

* MAX\_ITER = 3500

| AP | AP50 | AP75 | APs | APm | APl |

|:------:|:------:|:------:|:------:|:------:|:------:|

| 40.053 | 62.700 | 44.476 | 29.529 | 48.045 | 72.047 |

图片包含 户外, 男人, 跳, 空气

描述已自动生成跳在空中

低可信度描述已自动生成跳在空中

低可信度描述已自动生成雪地上的飞机

低可信度描述已自动生成

Figure 1 Iter 500

Figure 2 Iter 2000

Figure 3 Iter 3000

Figure 4 Iter 3500

The data andthe visualization results show that the increase in the number of iterations has a positive effect on the training effect. Each of Figures 1, 2, and 4 has a part of the body of the plane that is also recognized as a separate plane. In contrast, Figure 3, which used a number of 3000 iterations, identified exactly the best results.

**Part 2: Semantic Segmentation**

1. **Hyperparameter settings:**

After a series of adjustments, my parameters were set as follows:

*batch\_size = 4*

*learning\_rate = 0.003*

*num\_epochs = 150*

**2. Final architecture:**

I have made changes to the default “down” class and “up” class:

For “down” class: To increase the degree of training, I added an extra layer of “conv”.

For “up” class: To increase efficiency, I added an extra layer of “nn.BatchNorm2d”.

MyModel code:

*def forward(self, input):*

*y = self.input\_conv(input)*

*y = self.down1(y)*

*y = self.down2(y)*

*y = self.down3(y)*

*y = self.down4(y)*

*y = self.down5(y)*

*y = self.down6(y)*

*y = self.up1(y)*

*y = self.up2(y)*

*y = self.up3(y)*

*y = self.up4(y)*

*y = self.up5(y)*

*y = self.up6(y)*

*output = self.output\_conv(y)*

Torchsummary:

图片包含 表格

描述已自动生成 表格

描述已自动生成

**3. Loss functions:**

The loss functions I used is the default loss function:

*crit = nn.BCEWithLogitsLoss() # Define the loss function*

|  |  |  |  |
| --- | --- | --- | --- |
| *Epoch* | *Loss* | *Epoch* | *Loss* |
| *0* | 0.548725962638855 | *74* | 0.10606835782527924 |
| *1* | 0.35738807916641235 | *75* | 0.10531891882419586 |
| *2* | 0.29464346170425415 | *76* | 0.10455069690942764 |
| *3* | 0.2662748098373413 | *77* | 0.10415859520435333 |
| *4* | 0.24825114011764526 | *78* | 0.10370544344186783 |
| *…* | *…* | *…* | *…* |
| *…* | *…* | *…* | *…* |
| *…* | *…* | *…* | *…* |
| *69* | 0.10856733471155167 | *145* | 0.08330231159925461 |
| *70* | 0.10829269886016846 | *146* | 0.08313968032598495 |
| *71* | 0.10713459551334381 | *147* | 0.08315932005643845 |
| *72* | 0.10693884640932083 | *148* | 0.08290217071771622 |
| *73* | 0.10650912672281265 | *149* | 0.08267544955015182 |

**4. IoU:**

The final mean IoU of my model is:

*Mean IoU: 0.8916377923312807*

**5. Visualize:**

Img Prediction

图片包含 游戏机, 桌子, 台子

描述已自动生成图标

描述已自动生成

** 图标

描述已自动生成**

**图片包含 游戏机, 刀

描述已自动生成 **

**Part 3: Instance Segmentation**

**1. Kaggle Group Name: ChengH.**

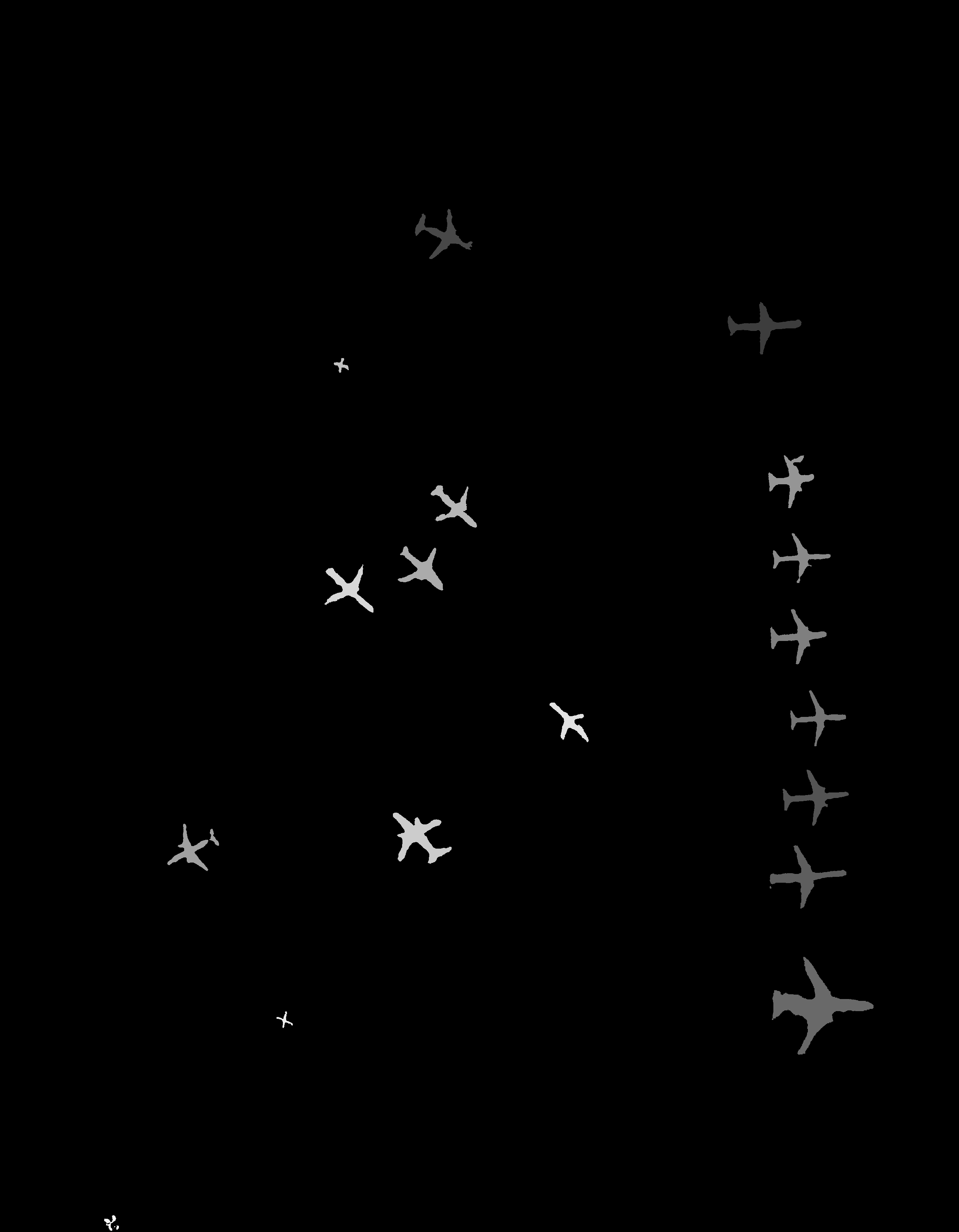
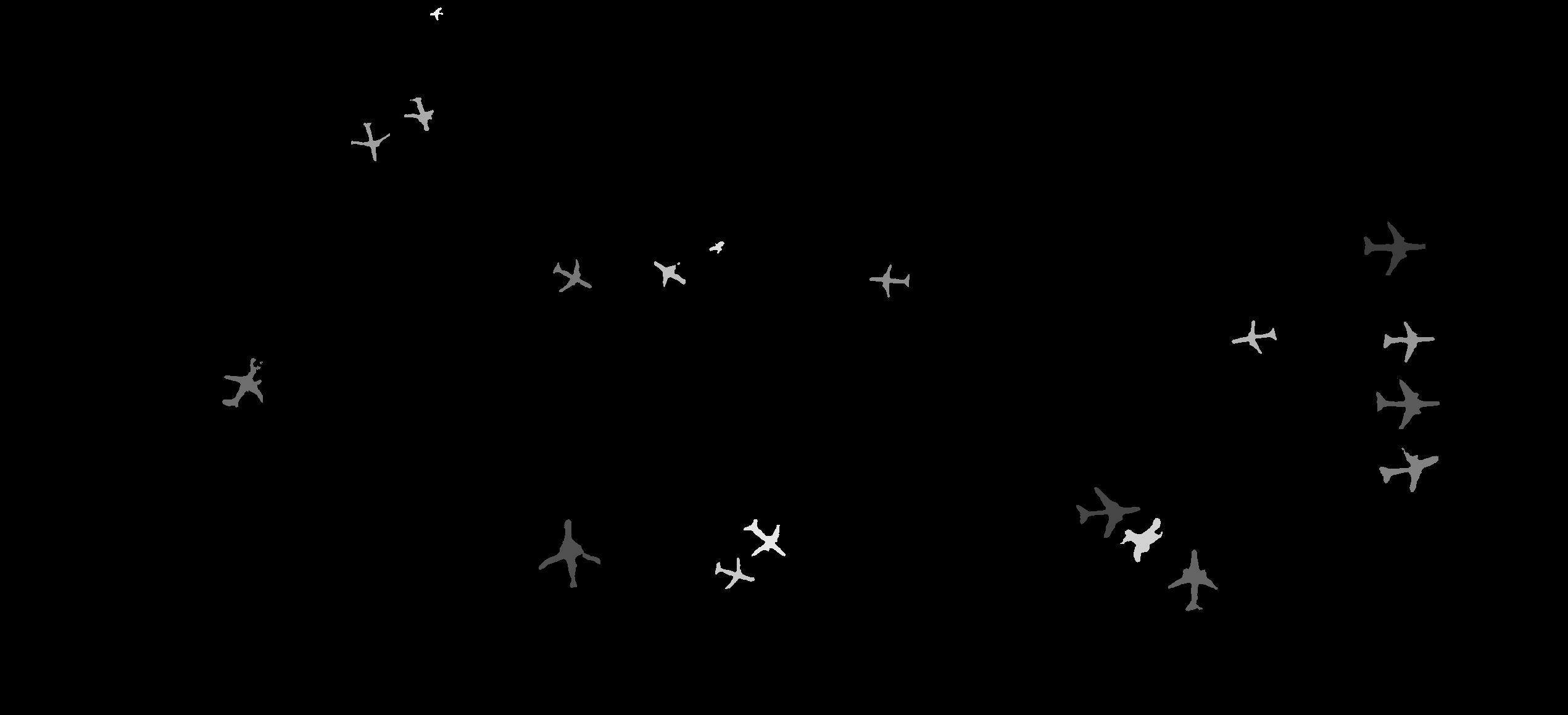
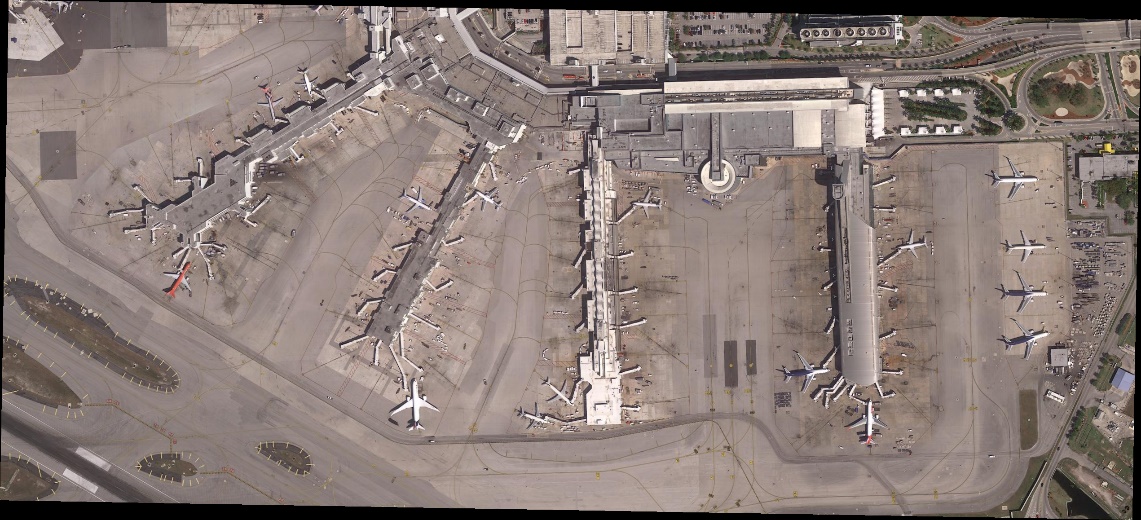
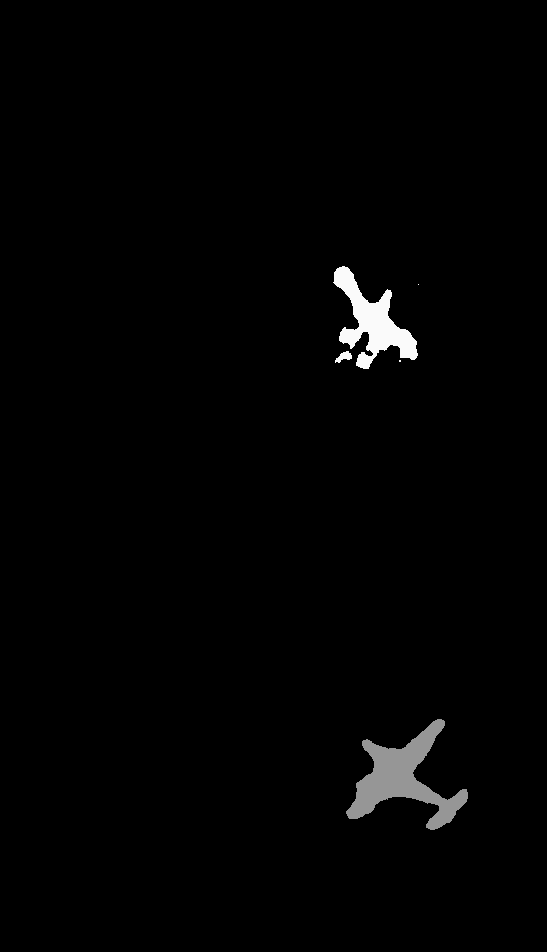
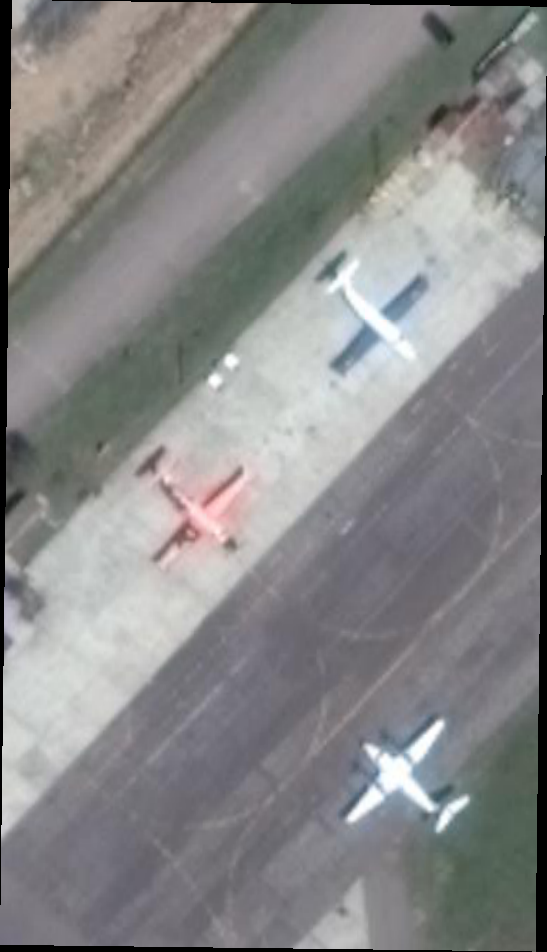
**Members:** Cheng Hu (301435966), Kaikun Fang (301416542)

**2. Best accuracy:**

The highest score of our group on Kaggle is 0.83098

**3.Visualization:**

My best result:

****

**Part4: Mask R-CNN**

1. **Config:**

文本

描述已自动生成

**2.Visualization:**



Figure 5



Figure 6

地图

描述已自动生成

Figure 7

1. **Evaluation:**

**Con:** The segmentation of the planes fuselage is not accurate enough. In some results, other objects next to the plane are incorrectly counted as part of the plane. And In Figure 5, the number of ships incorrectly identified as planes is relatively high, this is not as good as part1's performance.

**Pro:** The trained model is able to find the vast majority of planes in the picture. And faster.

1. **Difference between Part 3 and Part 4:**

The comparison of the results revealed that. In the detection of the number of planes, part4 is higher than the number detected by part3. But in the partitioning of the airplane fuselage and background, part3 is more accurate in partitioning the fuselage. part4 will incorrectly partition some things near the fuselage as part of the fuselage as well.