

# 111550173\_HW3 Report

## Introduction.

In this project, I implemented an instance segmentation system for cell images using deep learning techniques.

With the experience gained from the previous two assignments, I realized that the final performance is heavily influenced by the choice of backbone model made at the beginning.

Fortunately, since the dataset for this task was relatively small, training times were short, allowing me to experiment with various backbones for comparison.

I tried ResNet-50[1], ResNeXt-50[2], and ResNet-50-FPN v2[3], and ultimately selected the Mask R-CNN[4] model with a ResNet-50-FPN v2 backbone to detect and classify different types of cells.

To support training, I designed a custom dataset loader capable of handling cell images and their corresponding masks.

During the evaluation phase, I used ground-truth masks to calculate the IoU and mAP between predictions and true masks, providing an accurate assessment of model performance.

## Method.

The entire code is divided into the following sections:

- import
- model loading(with pretrain weight)
  - Load the pretrained model
  - Extract the Backbone
  - Wrap it into a new Mask R-CNN model
- data process
  - Reads the image and its corresponding class masks (for 4 classes).
  - Converts the image to a tensor.
  - Iterates over each class and generates a binary mask for each label, separates the components and appends them to masks and stores corresponding labels.

- Converts the list of masks to a tensor and uses masks\_to\_boxes to calculate bounding boxes.
- Filters out invalid boxes (where width/height  $\leq 0$ ).
- Creates a target dictionary containing the boxes, labels, masks, area, and crowd information.
- training
  - batch size = 2
    - A big size will lead the gpu out of memory, after my testing, 2 is the limit.
  - Lr rate = 0.0005
  - Adam optimizer
  - epoch = 5
  - save checkpoint for every epoch
- evaluate
- prediction

## Results.

```

Average Precision (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.287
Average Precision (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.461
Average Precision (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.340
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.209
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.238
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.119
Average Recall    (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.042
Average Recall    (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.214
Average Recall    (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.385
Average Recall    (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.353
Average Recall    (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.331
Average Recall    (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.162

```

After 5 epoch of training it reach the AP as the above figure shows, and the training loss is no more descending, and it seems enough for the Codabench contest, so I suddenly move on the prediction part

```

100%|██████████| 105/105 [04:40<00:00, 2.67s/it]

Epoch [1/5] - Average Loss: 1.8835
✅ Model saved at epoch 1.

100%|██████████| 105/105 [04:20<00:00, 2.48s/it]

Epoch [2/5] - Average Loss: 1.4656
✅ Model saved at epoch 2.

100%|██████████| 105/105 [04:20<00:00, 2.48s/it]

Epoch [3/5] - Average Loss: 1.2757
✅ Model saved at epoch 3.

100%|██████████| 105/105 [04:23<00:00, 2.51s/it]

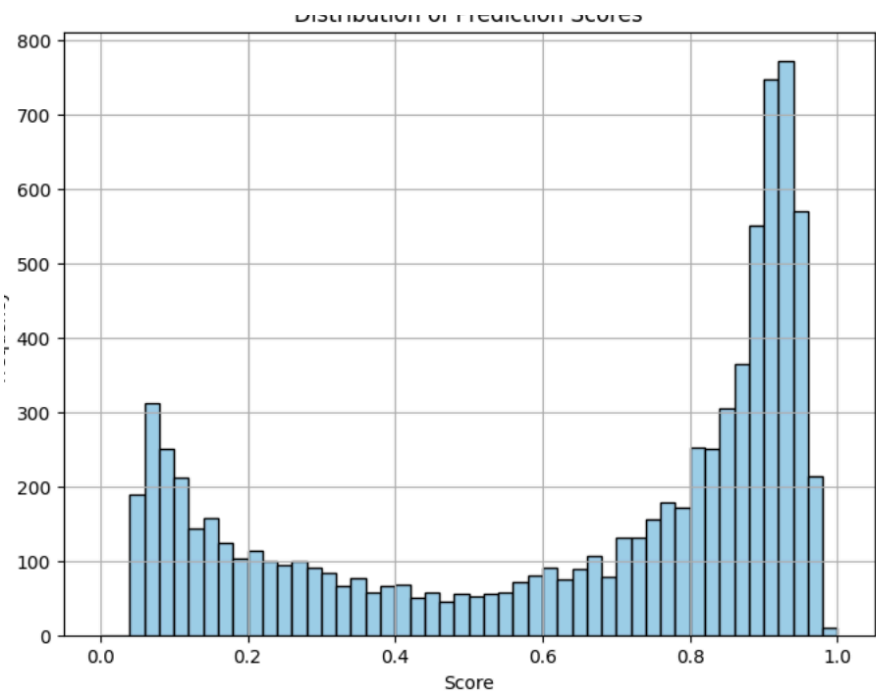
Epoch [4/5] - Average Loss: 1.2308
✅ Model saved at epoch 4.

100%|██████████| 105/105 [04:23<00:00, 2.51s/it]

Epoch [5/5] - Average Loss: 1.1274
✅ Model saved at epoch 5.

```

For choosing a better score threshold, I plot all the prediction's score into a histogram, and try a different dropping strategy, hoping to see a better result.



Based on the distribution in the chart, I tried several score thresholds:

- Cutting off the low-score peak (threshold = 0.2)
- Keeping only the high-score peak (threshold = 0.5)
- Accepting all predictions (threshold = 0)

Surprisingly, the version that accepted all predictions achieved the highest score. This suggests that the model's training may not have been very successful, as the correct and incorrect predictions have not been effectively differentiated in terms of their scores.

test-results_4.zip	2025-05-05 19:32	Finished	0.26	 
test-results_3.zip	2025-05-05 19:26	Finished	0.21	 
test-results_2.zip	2025-05-05 19:16	Finished	0.27	

**Reference.**

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in \*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)\*, Las Vegas, NV, USA, 2016, pp. 770–778.

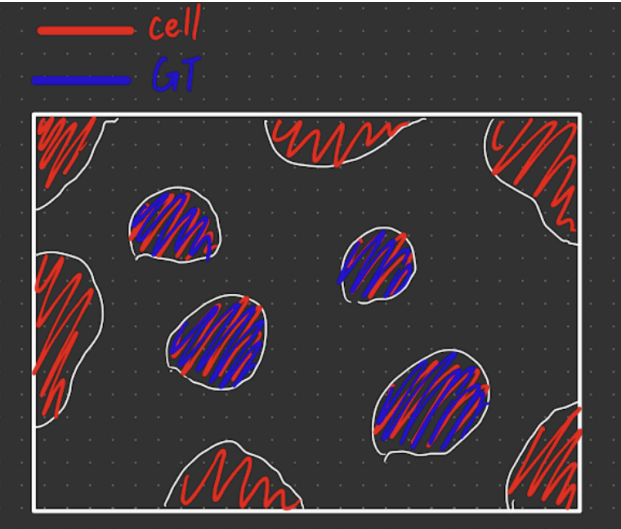
- [2] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in \*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)\*, Honolulu, HI, USA, 2017, pp. 1492–1500.
- [3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in \*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)\*, Honolulu, HI, USA, 2017, pp. 2117–2125.
- [4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in \*Proceedings of the IEEE International Conference on Computer Vision (ICCV)\*, Venice, Italy, 2017, pp. 2961–2969.



### **Additional experiment.**

While reviewing the dataset, I visualized all the images and ground truths by overlaying the masks onto the images for closer inspection. I discovered an interesting phenomenon: as illustrated in the diagram below, the red areas represent the actual occupied regions of the cells, but the ground truth only annotates the blue regions. In other words, the ground truth only includes cells that are fully contained within the image, while cells that are cut off by the image boundaries are not annotated.

Therefore, I attempted to filter the predictions during result generation by checking the bounding boxes — if a bounding box touches the image boundary, I discarded that prediction.

However, after applying this filtering rule, the final score did not change significantly.



281718	test-results.zip	2025-05-05 19:11	Finished	0.27	 
281305	test-results.zip	2025-05-05 02:56	Finished	0.27	 

**Github Link.**

<https://github.com/Seanlin0911/NYCU-DLCV-2025-Spring/tree/main>