110612117 張仲瑜 HW3 report

I. Introduction

1. Goal

In this task, we aim to achieve as high mAP as possible in cell instance segmentation competition with Mask-RCNN [1] based model, and there are totally four kinds of cells given in .tiff format from the dataset.

2. Environment

- ✧ Laptop: Windows / RTX4060 * 1
- ✧ Kaggle account * 3: Linux / Tesla T4 * 2

3. Core Idea

I revised the Faster-RCNN [2] pipeline in hw2 into a Mask-RCNN pipeline by adding the mask branch and set its ROI pooler [2] to complete the task, and conducted experiments on the sampling rate, RPN [2] proposal threshold and different losses to optimize the performance.

II. Method

1. Data augmentation set

- ✧ Flip and Rotation
  Horizontal, Vertical flip and Rotation [3] are applied to simulate different angles of cells, aiming to increase the robustness of the model.
- ✧ ToGreyScale
  Some of the cells are dyed in the dataset for the sake of observation, so grayscales are applied to avoid the effect of dye.

2. Model Architecture

✧ Model name: SEresneXt101d_32x8d [4] [5]

Output size	SE-ResNeXt-101 (32×8d)		
144 × 144	conv, 7 × 7, 64, stride 2		
72 * 72	max pool, 3 × 3, stride 2		
72 × 72	conv, 1 × 1, 256 conv, 3 × 3, 256 conv, 1 × 1, 256 SE module	C=32	x3
36 x 36	conv, 1 × 1, 512 conv, 3 × 3, 512 conv, 1 × 1, 512 SE module	C=32	x4
18x18	conv, 1 × 1, 1024 conv, 3 × 3, 1024 conv, 1 × 1, 1024 SE module	C=32	x23
9x9	conv, 1 × 1, 2048 conv, 3 × 3, 2048 conv, 1 × 1, 2048 SE module	C=32	x3
1x1	dropout (p = 0.5)		
1x1	global average pooling, 100-d fc, softmax		

Total parameters: 93.6 M

✧ FPN: I use all four 4 layers of the model to extract feature maps.

```
def forward(self, x):
    x = self.stem(x)
    c2 = self.layer1(x)
    c3 = self.layer2(c2)
    c4 = self.layer3(c3)
    c5 = self.layer4(c4)
    return {
        '0': c2,
        '1': c3,
        '2': c4,
        '3': c5
    }
```

- ✧ Anchor size [2] and aspect ratio:
  Ascending pixels size as layers go deeper, with the common digit aspect ratio.

```
anchor_generator = torchvision.models.detection.rpn.AnchorGenerator(
    sizes = ((4, 8), (16, 32), (32, 64), (64, 128), (128, 256)),
    aspect_ratios = ((0.5, 1.0, 2.0),) * 5,
)
```

- ✧ Box ROI Align module [2]:
  4 feature maps with various sampling ratios.

```
roi_pooler = torchvision.ops.MultiScaleRoIAlign(
    featmap_names=['0', '1', '2', '3'],
    output_size=7,
    sampling_ratio=2
)
```

- ✧ Mask ROI Align module [1]:
  4 feature maps with sampling ratios.

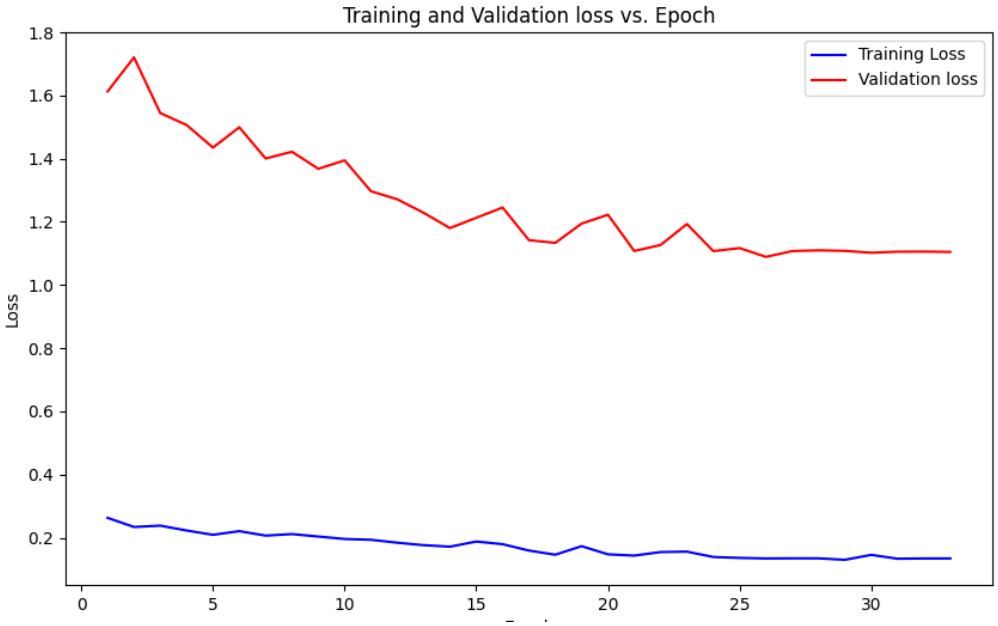
```
mask_roi_pooler = torchvision.ops.MultiScaleRoIAlign(
    featmap_names=['0', '1', '2', '3'],
    output_size=14,
    sampling_ratio=4
)
```

3. Hyperparameters (best one)
   - ✧ Pretrained on imageNet1k
   - ✧ Training/ testing box NMS [2] threshold: 0.5
   - ✧ Training/ testing box score threshold: 0.05
   - ✧ Anchor map: As image above
   - ✧ Aspect ratios: As image above
   - ✧ Learning rate: 5e-5 and * 0.5 decay factor whenever both validation accuracy and loss didn't decrease in 2 consecutive epochs.

- ✧ Optimizer: AdamW
- ✧ Batch size: 1
- ✧ Epoch: Max is 50 but with early-stopping patience = 7 epochs, stopping at epoch 26.
- ✧ Weight decay = 1e-3

III. Result:

1. Best performance
   - ✧ mAP: 0.4027 on private testing
   - ✧ Learning curve (epoch vs Training loss & loss)



IV. Additional experiment:

1. Hypothesis: Adding the CIOU loss [6] as one of the losses in faster-RCNN loss, and adding the Dice loss [7], Boundary loss [8] as one of the losses in mask-RCNN loss can improve the performance.

   ✧ In the original design, losses are composed of three parts, which are Classification loss [1] (Cross Entropy), Box_loss [1] (Smooth L1) and Mask loss [1] (Binary Cross Entropy). Since Cross Entropy suffers from sample imbalance and doesn't consider the boundary accuracy, Dice Loss and Boundary loss are applied to address the corresponding problem. Furthermore, since we use mAP to evaluate the model, Complete Ciou loss takes IoU, Central distance and width height ratio into consideration, which can intuitively fit mAP.

   ✧ Result

Loss component	mAP
CE + Smooth L1 + BCE	0.385
CIOU + Dice + Boundary	0.401

   ✧ Implication

   As we discussed above, the performance did largely increase, showing the importance of evaluating intersection and boundary areas.

2. Hypothesis: Adjusting the classification loss into Focal loss can boost performance.
   - ✧ Since the ratio of background to foreground in the instance segmentation task could be lower than 1/1000, the background part would dominate the whole loss when training. As a result, Focal loss is applied to increase the penalty of hard sample.

   - ✧ Result

Mask Sampling Rate	mAP
CIOU + Dice + Boundary	0.401
Focal Loss + CIOU + Dice + Boundary	0.4148

   - ✧ Implication
     As I expected, with alpha = 0.25 and gamma = 2.0, the model are more sensitive to hard foreground sample which results in the boost of performance.

3. Observation: The effect of mask threshold in the inference stage.
   - ✧ During the inference stage, we need to determine a threshold to turn the sigmoid output of mask into a binary mask.

   - ✧ Result

Mask threshold	mAP
0.5	0.4014
0.6	0.4027
0.7	0.4022

   - ✧ Implication:
     There's only a slight difference between them, showing that the other factors such as the accuracy of the bounding boxes and classification play a greater role than this factor.

V. Github
https://github.com/sharkccy/NYCU_CV_2025_Spring/tree/main/hw3

VI. Thoughts
Instead of directly changing the backbone into powerful backbone such as Vision Transformer based model, this time I choose to spend more time implementing different kinds of losses that are suitable for instance segmentation. Although it almost drove me crazy trying to coordinate custom losses into the original source code, it's still a valuable experience learning those losses.

model	box_sample	mask_sample	loss	box_score_thresh	mask_score_thresh	data aug	det per img	mAP
aa101d	2	2	ori	0.65 / 0.65	0.5	no	100 / 100	0.197
aa101d	2	2	ori	0.65 / 0.65	0.5	no	1000 / 1000	0.277
aa101d	2	2	ori	0.65 / 0.60	0.5	no	1000 / 1000	0.287
aa101d	2	2	ori	0.65 / 0.55	0.5	no	1000 / 1000	0.291
aa101d	2	2	ori	0.65 / 0.50	0.5	no	1000 / 1000	0.295
aa101d	2	2	ori	0.65 / 0.45	0.5	no	1000 / 1000	0.295
SEresneXt32x8d	2	4	ori	0.65 / 0.4	0.5	no	1000 / 1000	0.325
SEresneXt32x8d	2	4	ori	0.65 / 0.5	0.6	no	1000 / 1000	0.327
SEresneXt32x8d	2	4	ori	0.65 / 0.4	0.6	no	1000 / 1000	0.326
SEresneXt32x8d	2	4	ori	0.65 / 0.05	0.6	no	1000 / 1000	0.338
SEresneXt32x8d	2	4	ori	0.1 / 0.1	0.6	yes	1000 / 1000	0.351
SEresneXt32x8d	2	4	ori	0.05 / 0.05	0.6	yes	1000 / 1000	0.385
SEresneXt32x8d	2	4	*	0.05 / 0.05	0.5	yes	1000 / 1000	0.402
SEresneXt32x8d	2	4	*	0.05 / 0.05	0.6	yes	1000 / 1000	0.402
SEresneXt32x8d	2	4	*	0.05 / 0.05	0.7	yes	1000 / 1000	0.401

Fig. Some of my experiments

VII. Reference:

1. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). *Mask R-CNN*. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, 2961-2969. https://doi.org/10.1109/ICCV.2017.322
2. Girshick, R. (2015). Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169
3. Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J Big Data* **6**, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
4. Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1492–1500. https://doi.org/10.1109/CVPR.2017.634
5. Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7132–7141. https://doi.org/10.1109/CVPR.2018.00745
6. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU loss: Faster and better learning for bounding box regression. *Proceedings of the AAAI Conference on Artificial*

Intelligence, 34(07), 12993–13000. https://arxiv.org/abs/1911.08287

7. Milletari, F., Navab, N., & Ahmadi, S. A. (2016). V-Net: Fully convolutional neural networks for volumetric medical image segmentation. *Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV)*, 565–571. https://arxiv.org/abs/1606.04797
8. Kervadec, H., Dolz, J., Yuan, J., Desrosiers, C., Granger, E., & Ayed, I. B. (2019). Boundary loss for highly unbalanced segmentation. *Medical Imaging with Deep Learning (MIDL)*. https://arxiv.org/abs/1812.07032
9. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2980–2988. https://doi.org/10.1109/ICCV.2017.324