VRDL Homework 3: Nuclei Segmentation

Jia-Wei Liao
jw.sc09@nycu.edu.tw
Department of Applied Mathematics, NYCU

March 9, 2022

Abstract

In this homework, we implement the deep learning based techniques to do nuclear segmentation. We use the Mask R-CNN model on Detectron2 [6] which is the famous network for instance segmentation. After tuning the hyper-parameter, we beat the baseline and have the test mAP of 0.244385. Our code is available at https://github.com/Jia-Wei-Liao/Nuclear_Dataset_Segmentation.

1 Introduction

The nuclear segmentation dataset contains 24 training images with 14,598 nuclear and 6 test images with 2,360 nuclear. Since we don't have an annotation of data, we should generate annotation at the first, and then use Detectron2 [6] to register the custom datasets before training step. At the inference step, we should generate the submission file and upload to the CodaLab [7]. The submission of format should follow by COCO results which include image_id, category_id, segmentation, and score. In addition, the segmentation result should be the RLE encoded format.

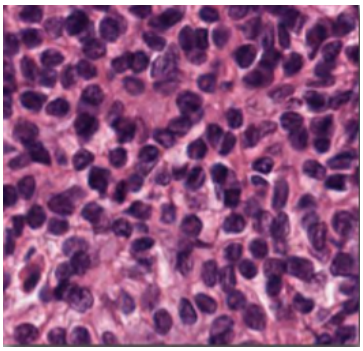

Figure 1. Image

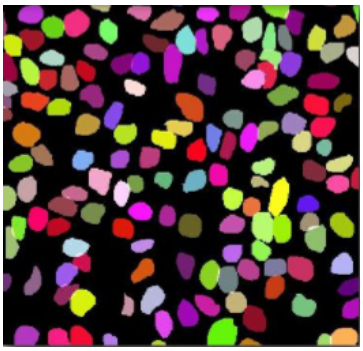

Figure 2. Ground truth

2 Data Pre-processing

To avoid over-fitting, we do the data augmentation as the folling:

- Random flip with left and right
- Random crop the image to 500×500
- Random adjust the image brightness

3 Mask R-CNN

Instance Segmentation: In 2017, Kaiming He proposed the Mask R-CNN [2] on the instance segmentation task. The instance segmentation combines detection with segmentation. I think that semantic segmentation like a pixel-wise classifier. It can not detect individual objects when the multiple objects overlap on the image. However, instance segmentation can generate the bounding box of the object, it can help the model distinguish the individual objects. We will introduce the Mask R-CNN as follow.

3.1 Network Architecture

The backbone of Mask R-CNN is ResNet50 or ResNet101. It can apply FPN to generate the multiscale feature map and input it to the RPN to produce the RoI. To unity the size of RoI, we can do the RoIAlign or RoIPolling and then split into box head and mask head.

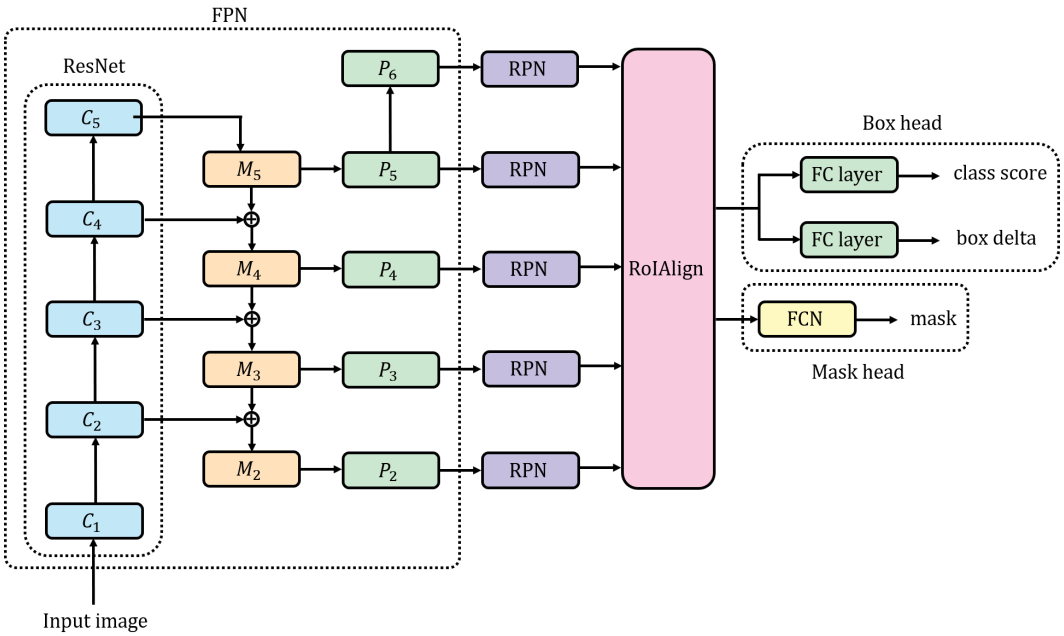

Figure 3. Mask R-CNN (ResNet50-FPN)

3.2 Feature Pyramid Networks (FPN)

The FPN present in [4]. It's a feature extractor and it can detect an object at different scales since the smaller feature map has a larger receptive field and the bigger feature map merged high resolution and low resolution with upsampling by addition. This can improve the accuracy of detecting the small objects.

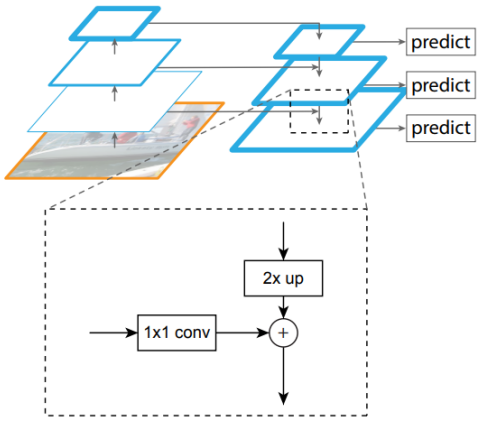

Figure 4. FPN

3.3 Region Proposal Network (RPN)

3.3.1 Sliding Window

The RPN is constructed by the fully convolutional network which can share computation to speed up object detection. The inputs of RPN is the convolution feature map of the last output layer and the outputs of RPN is a set of a rectangular object. We call them Region of Interest (RoI). To generate the region proposals, we set the sliding window on the convolutional feature map and use the CNN base model to map them into the low dimension feature.

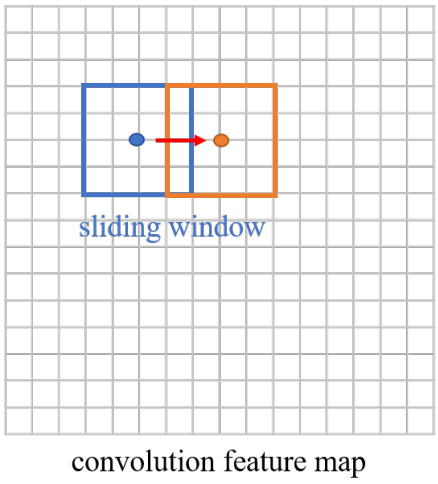

Figure 5. Sliding window

3.3.2 Anchors

An anchor is centered at the sliding window. It's associated with a scale and aspect ratio and it has the following purposes:

1. Determine whether the anchor has covered the target.
2. Modify the coordinate of the anchors which cover the target by regression.

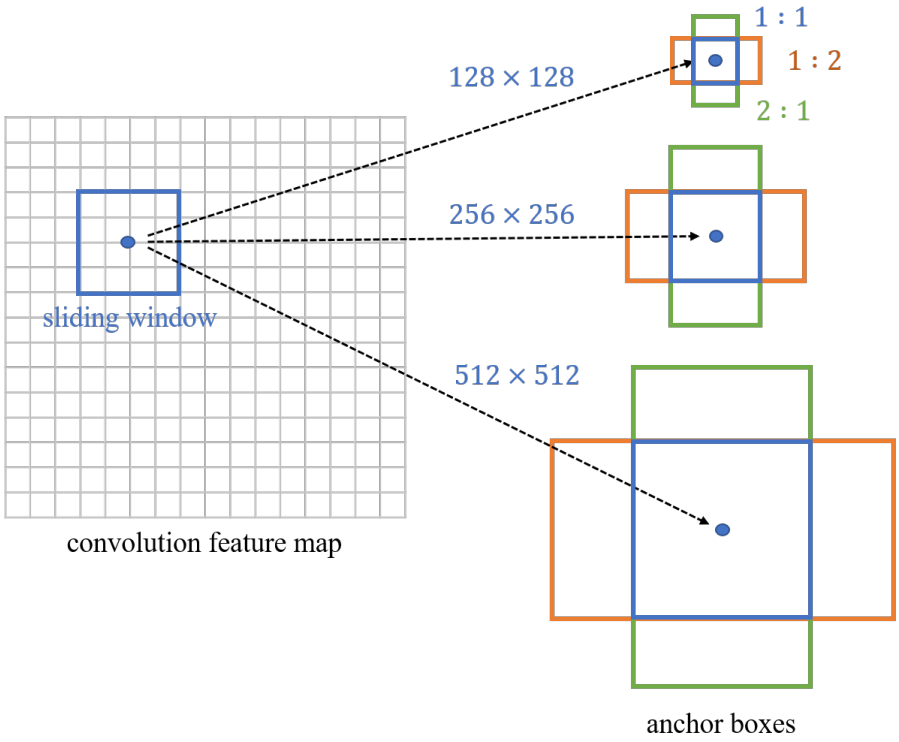

Figure 6. Anchor

3.4 RoIPooling and RoIAlign

Before enter to the box head and mask head, we need to unity the RoI size.

3.4.1 RoIPooling

The first method is RoIPooling. It has the following two steps:

Step 1. Get the RoI from the feature map which rounds the coordinate to the integer.

Step 2. To unity the RoI size, we divide the RoI into 3 parts and do the maximum pooling like the Figure 7.

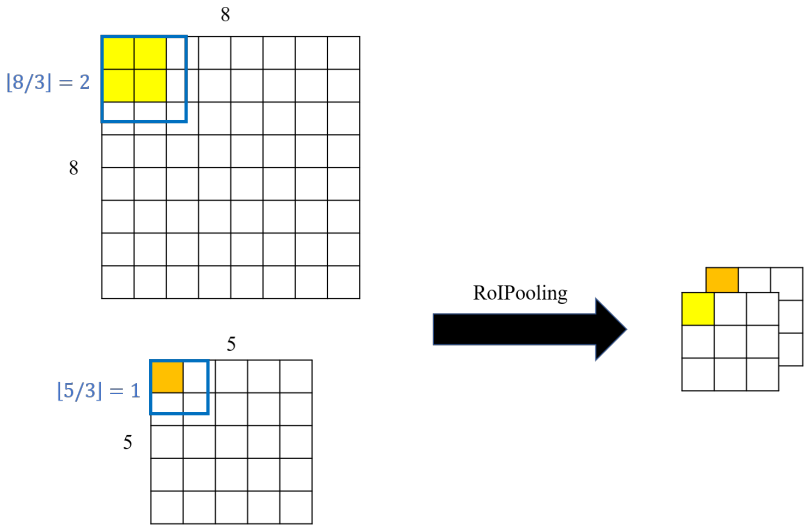

Figure 7. RoIPooling

3.4.2 RoIAlign

Since RoIPooling round the float number to integer two times, it makes errors. So the author of Mask R-CNN [2] uses RoIAlign to replace RoIPooling. They use bilinear interpolation to represent $f(x, y)$ from the nearby grid points

$$f(x,y) \approx \sum_{i=1}^{2}\sum_{j=1}^{2} f(x_{3-i}, y_{3-j}) \frac{(x-x_i)(y-y_j)}{(x_{3-i}-x_i)(y_{3-j}-y_j)}$$

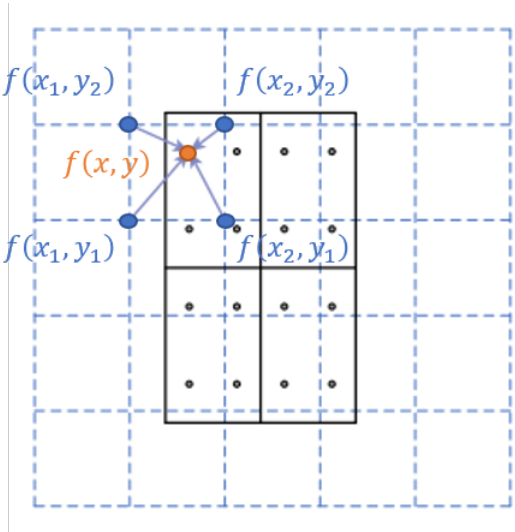

Figure 8. RoIAlign

Compare with the RoIPooling, the RoIAlign can get the high quality than RoIPooling

4 Losses for Mask R-CNN

4.1 Classification Loss

The classification of the the RPN and box head is the binary cross entropy.

4.2 Box regression

4.2.1 Part of RPN

Let $(x_a, y_a, w_a, h_a), (x_g, y_g, w_g, h_g), (x_p, y_p, w_p, h_p)$ be the center, width, height of anchor box, ground truth box, and predicted box. First, we set

$$t_x = \frac{x_g - x_a}{w_a}, \quad t_y = \frac{y_g - y_a}{h_a}, \quad t_w = \ln\left(\frac{w_g}{w_a}\right) \text{ and } t_h = \ln\left(\frac{h_g}{h_a}\right)$$

and denote $t = (t_x, t_y, t_w, t_h)$ which do the normalize by the anchor. Suppose the outputs of model is $\hat{t} = (\hat{t_x}, \hat{t_y}, \hat{t_w}, \hat{t_h})$. The box regression loss is defined as the smooth L_1 function

$$\mathcal{L}_{\text{box}}(\hat{t}, t) = \sum_{i \in \{x,y,w,h\}} \mathcal{R}(\hat{t}_i - t_i)$$

in which

$$\mathcal{R}(x) = \begin{cases} 0.5x^2, \text{ if } |x| \leq 1 \\ |x| - 0.5, \text{ otherwise} \end{cases}$$

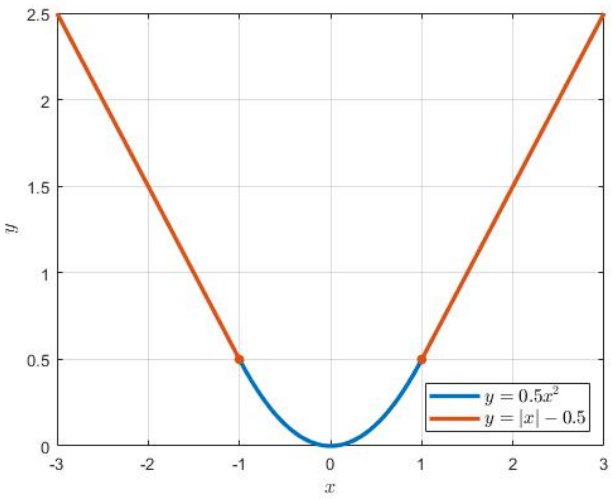

Figure 9. Smooth L_1 function

After we optimize the regression loss, we can get the proper $\hat{t}$ which is close to t. Then we can get the predicted box as the following:

$$\begin{cases} x_p = x_a + w_a \hat{t}_x \\ y_p = y_a + h_a \hat{t}_y \\ w_p = w_a \cdot e^{\hat{t}_w} \\ h_p = h_a \cdot e^{\hat{t}_h} \end{cases}$$

The smooth L_1 function is a robust loss that is less sensitive to outliers than L_2 loss used in our model. It grow slowly outside of $[-1, 1]$ that can prevent exploding gradients and be smooth on the $[-1, 1]$.

4.2.2 Part of box head

The box regression for the box head want to modify the predicted coordinates of the bounding box which is the same as the box regression for RPN. The difference is the box regression for the box head does not have the anchor box. So the way of it updates the parameters is

$$\begin{cases} x_p \leftarrow x_p + w_p t_x \\ y_p \leftarrow y_p + h_p t_y \\ w_p \leftarrow w_p \cdot e^{t_w} \\ h_p \leftarrow h_p \cdot e^{t_h} \end{cases}$$

where (x_p, y_p, w_p, h_p) is the prediction of RPN and (t_x, t_y, t_w, t_h) is the output of box head.

4.3 Mask Loss

Since our task is instance segmentation, the mask may be overlapping. The author in [2] apply the sigmoid to per-pixel instead of the softmax, and use the binary cross-entropy loss to compute mask loss.

Pixel-wise Binary Cross-Entropy Loss

Let $Y, \hat{Y} \in \mathbb{R}^{m \times m \times 2}$ be the ground truth with one-hot label and prediction. The pixel-wise binary cross-entropy loss is

$$\mathcal{L}_{BCE}(\hat{Y}, Y) = -\frac{1}{2} \sum_{c=1}^{2} \sum_{i,j} Y_{i,j}^{(c)} \log \hat{Y}_{i,j}^{(c)}$$

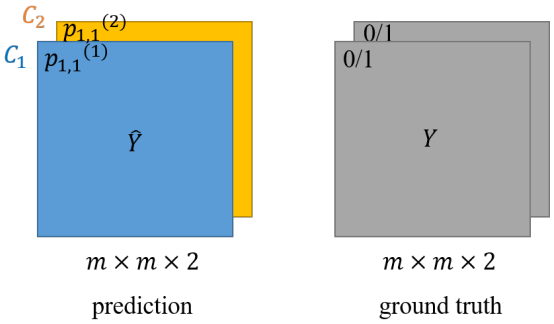

Figure 10. Mask

4.4 Loss Curve

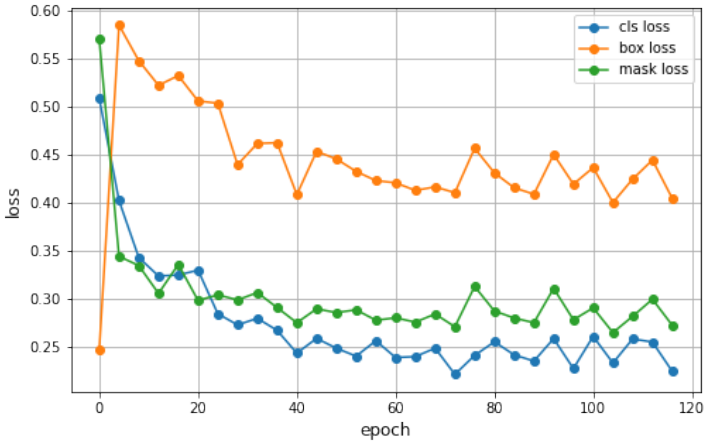

Figure 11. Loss for box head and mask head

5 Optimization

5.1 Optimizer

Most people choose the Adam to train. But according to our experiment in the homework 2, the Adam got the worst performance and the SGD got better than AdamW. In this homework, our task is instance segmentation. It combines detection and segmentation. I think that SGD should be better at this time, so I choose SGD directly and train with the learning rate schedule.

5.2 Learning Rate Schedule

5.2.1 Warm Up

At the beginning of training, the warm-up method set a smaller learning rate to train some epochs. We know that the deep learning model is very much affected by initial weights. So, if we choose a large learning rate, it may lead to instability of the model. The warm-up method can lead the model to gradually become stable at the beginning. On the other hand, it promotes the convergence speed of the model.

Our experience: If we close the warm-up learning rate, it produces exploding gradients. So, I am even more convinced that the warm-up method is a good trick for training.

5.2.2 Step Decay

To further improve results and make the model converge to the global minimum, we can adjust the learning rate by exponent decay at the specific epoch.

$$\text{Learning Rate} \leftarrow \text{Learning Rate} \times \text{Decay Factor}$$

5.2.3 Hyperparameter

- Batch size: 1
- Epoch: 120
- Base learning rate: 0.01
- Warm up epoch: 3
- Epoch of step decay: 20, 50, 80, 90

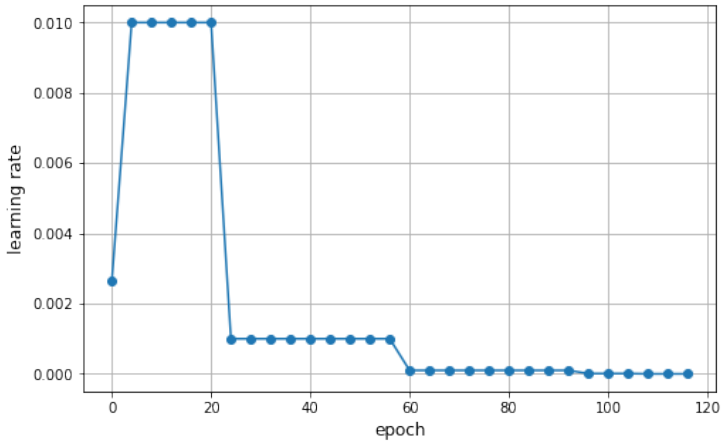

Figure 12. Learning rate schedule

6 Experiments

We use the data augmentation, learning schedule for training and try the different backbones. Our experiments result display as Table 1.

model	backbone	mAP_{50} (%)
Mask R-CNN	**ResNet-50-C4**	**24.4385**
Mask R-CNN	**ResNet-50-FPN**	24.0068
Mask R-CNN	**ResNet-101-C4**	24.2977
Mask R-CNN	**ResNet-101-FPN**	24.1530

Table 1. mAP

Figure 13. prediction of TCGA-G9-6348-01Z-00-DX1 on test set

7 Conclusion

In this homework, we implement Mask R-CNN and do the data augmentation to avoid over-fitting. First, we used the ResNet101 as the backbone and did not use warm-up learning rate, the box loss occurred the exploding gradients. After we use the warm-up learning rate, it leads the loss to be stable at the first few epochs. We uploaded the results of the first training. We beat the baseline and get the test mAP of 0.243613.

Next, we tried the different backbone ResNet50, the test mAP increase 0.244385. I think the parameters of ResNet101 are more than ResNet50, so it should have more training data. But in this task, we only have a few training data so that the result is not as expected. On the other hand, we train the 120 epoch at the first. We try to increase the epoch to 200, but it leads to over-fitting. It seems that an epoch between 100 and 200 is the best choice.

Results

#	User	Entries	Date of Last Entry	Team Name	mAP ▲
1	CL_Kuo	36	12/15/21	310552027	0.24525 (1)
2	darker	20	12/14/21	310554017	0.24518 (2)
3	niny1212	12	12/15/21	0716228	0.24482 (3)
4	ktng	20	12/07/21	309652030	0.24470 (4)
5	EJ_Hung	7	12/15/21	309657002	0.24464 (5)
6	YCHung	21	12/15/21	310653005	0.24460 (6)
7	Hsuan	12	12/05/21	310554014	0.24439 (7)
8	JWSAM	29	12/16/21	309652008	0.24438 (8)
9	c132	18	12/14/21		0.24436 (9)
10	zycn	32	12/14/21	309551178	0.24436 (10)

Figure 14. CodaLab

References

[1] Ross Girshick, Fast R-CNN, *CVPR*, 2015.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, Mask R-CNN, *CVPR*, 2017.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *CVPR*, 2016.

[4] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, Feature Pyramid Networks for Object Detection, *CVPR*, 2017.

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, Focal Loss for Dense Object Detection, *CVPR*, 2017.

[6] Detectron2 github:
https://github.com/facebookresearch/detectron2

[7] CodaLab website:
https://codalab.lisn.upsaclay.fr/competitions/333?secret_key=3b31d945-289d-4da6-939d-39435b506ee5