VRDL Homework 3: Nuclei Segmentation

Zhi-Yi Chin
joycenerd.cs09@nycu.edu.tw

December 15, 2021

1 Introduction

In this homework, we participate in nuclei segmentation challenge on CodaLab. In this challenge, we perform instance segmentation on TCGA nuclei dataset from the 2018 Kaggle Data Science Bowl. This dataset contains 24 training images with 14,598 nuclei and 6 test images with 2,360 nuclei. For training, pre-trained models are allowed, but no external data should be used. We apply four existing methods to solve this challenge. Code is available at https://github.com/joycenerd/mmdet-nucleus-instance-segmentation.

2 Methodology

2.1 Data pre-processing

2.1.1 Train validation split

There are 24 images in the original training data, and no validation data provides. In order to evaluate our segmentation model's performance, we use 4 images in the training data as our validation data. We have 11,109 nuclei as our training data and 3,489 nuclei as our validation data.

2.1.2 COCO annotation

Many segmentation models load COCO style annotation files as input labels, so we transform binary mask images into COCO style annotation via jsbroks/imantics [1]. We only need to define the Mask, Image, Category, and Dataset object, imantics will handle all the other detailed things in COCO for us. The advantages of imantics are that it is easy to use, and the conversion from binary mask to COCO segmentation annotation is fast. The downsides are, first, there is a bug in this package. We will need to modify the source code so that no error occurs while using this package. Second, there are some invalid segmentation results, so we must delete them manually. We use this package to create two JSON files, one for training and another for validation.

2.2 Model Architecture

We have tried 4 existing methods in openmmlab/mmdetection [3]. We choose mmdetection for two reasons. First, it supports many robust existing methods. Second, it is easy to use. We only need to provide the data with the specific format and add our custom configuration file, and then we can train the model easily. The 4 existing methods we have tried are: Mask R-CNN [5], Cascade Mask R-CNN [2], Mask Scoring R-CNN [6], and PointRend [7].

2.2.1 Mask R-CNN

Mask R-CNN is a CNN-based image segmentation method. Mask R-CNN was developed on top of Faster R-CNN [10], a Region-Based Convolutional Neural Network.

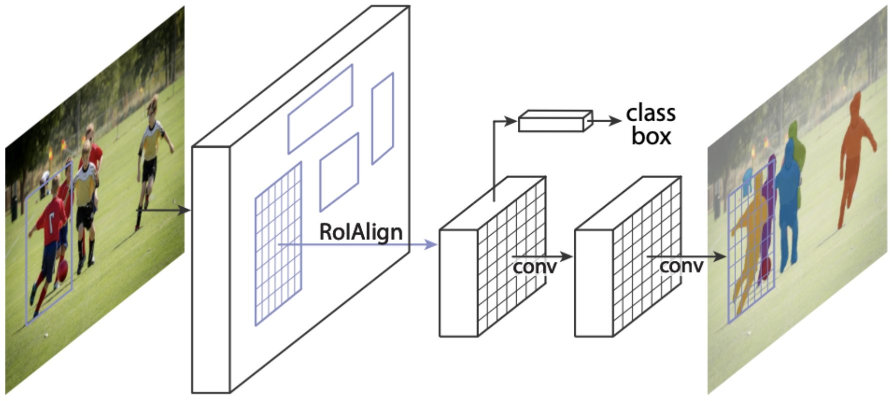

Figure 1: Mask R-CNN framework for instance segmentation.

There are four reasons we use Mask R-CNN:

1. It is simple to train.
2. It has good performance as a single-model method.
3. It is very efficient and adds only a small overhead to Faster R-CNN.
4. It is easy to generalize to other tasks.

Mask R-CNN is an extension of Faster R-CNN and works by adding a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition. The critical element of Mask R-CNN is the pixel-to-pixel alignment, which is the main missing piece of Fast/Faster R-CNN. Mask R-CNN adopts the same two-stage procedure with an identical first stage (RPN). In parallel to predicting the class and box offset in the second stage, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to most current systems, where classification depends on mask predictions.

2.2.2 Cascade Mask R-CNN

The performance of deep learning object detectors tends to degrade with increasing the IoU (Intersection over Union) thresholds. They usually suffer from two main factors: (1) Overfitting during training, due to exponentially vanishing positive samples, i.e., many positive samples are gone when the IoU threshold increases. (2) Inference-time mismatch between the IoUs for which the detector is optimal and those of the input hypotheses. e.g., training at higher(lower) IoU threshold but test at lower(higher) IoU threshold. Cascade Mask R-CNN extended Mask R-CNN is proposed to solve the above problems.

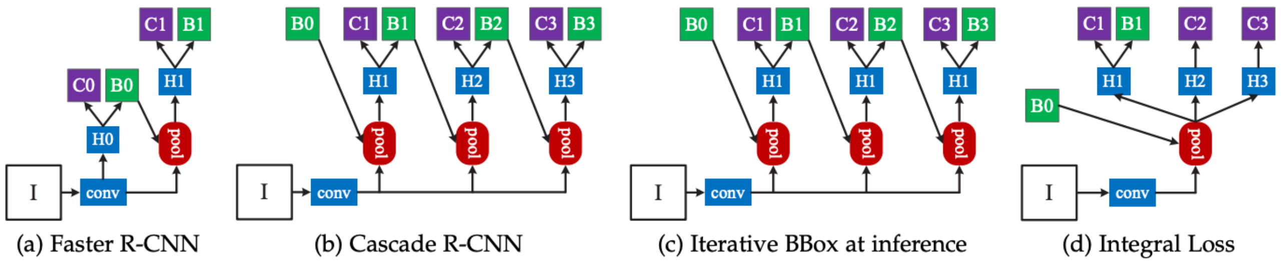

Figure 2: The architectures of Faster R-CNN, Cascade R-CNN, iterative bbox at inference and integral loss.

Different from iterative bbox that uses the same H_1, different heads are used at different stages, i.e., H_1, H_2, H_3 are used. Furthermore, each is designed for one specific IoU threshold from small to large, preventing overfitting. The cascaded regression (using different heads) is a resampling procedure, not a post-processing step, providing good positive samples to the next stage. There is no discrepancy between training and inference since the architecture and IoU thresholds are the same during training and inference, solving the inference-time mismatch problem.

We use Cascade Mask R-CNN mainly for one reason: the distribution of the proposals in the training and inference has no difference since the cascaded regression method is used in the training and inference.

2.2.3 Mask Scoring R-CNN

Mask Scoring R-CNN involves a deep network learning the quality of its predictions. This leads to more accurate mask quality scores than those of the model's predecessor, the state-of-the-art Mask R-CNN. The AP (average precision) improves by about 1.5%.

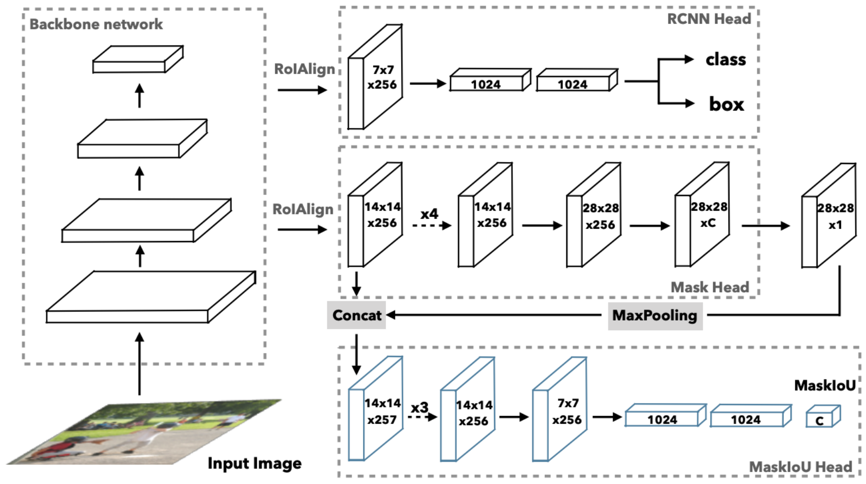

Figure 3: Network architecture of Mask Scoring R-CNN. The input image is fed into a backbone network to generate RoIs via RPN and RoI features via RoIAlign. The RCNN head and Mask head are standard components of Mask R-CNN. For predicting MaskIoU, the predicted mask and RoI feature are input. The MaskIoU head has 4 convolution layers (all have kernel=3 and the final one uses stride=2 for downsampling) and 3 fully connected layers (the final one outputs C classes MaskIoU.)

We choose Mask Scoring R-CNN for three reasons:

1. The improvements it brings are consistent and noticeable (compared with Mask R-CNN).
2. It appears robust to different backbone networks and frameworks, proving the method's effectiveness.
3. This approach is innovative and easy to implement.

Mask Scoring R-CNN consists of Mask R-CNN with a MaskIoU prediction network named MaskIoU head. This network compares the predicted mask with its ground truth mask using pixel-level IoU. The MaskIoU module is trained using regression. The model prioritizes more accurate mask predictions and penalizes the score of the instance mask if it has a high classification score while the mask is not accurate. This leads to a better interpretation of actual mask quality than current instance segmentation methods.

2.2.4 PointRend

PointRend treats segmentation as a rendering problem, using a segmentation strategy to select a set of non-uniform points and calculate labels adaptively. PointRend can be incorporated into common instance segmentation meta-architectures such as Mask R-CNN, and semantic segmentation meta-architectures such as FCN [8].

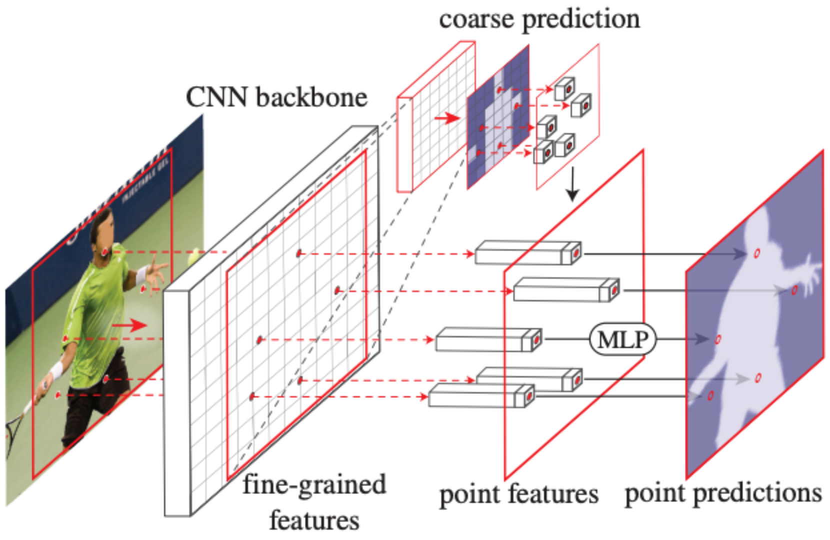

Figure 4: PointRend model architecture. For instance segmentation, first output a rough semantic segmentation result (7×7), and then iterate the up-sampling process, each up-sampling process includes: (1) Bilinear interpolation upsamplng. (2) Execute the PointRend module, select the point with high uncertainty, extract its feature vector, and use a shared MLP to get its output result.

The method proposed by PointRend is simple and effective. The segmentation maps from the proposed approach have much better boundary label maps than methods without non-uniform grids. It is nice to see that combining simple ideas from computer graphics can help with computer vision tasks.

PointRend employs an iterative subdivision algorithm inspired by image rendering in computer graphics. In their paper, researchers argue that an analogy between image segmentation and occupancy grids in rendering can be leveraged to build a neural network model that will perform better and output higher-resolution segmentation masks. The proposed module takes one or more CNN output feature maps, defined over a regular square grid several times coarser than the original image resolution. Then using a point selection strategy, the algorithm first chooses a small number of points on the grid. A point-wise feature representation is extracted for each of these points using bilinear interpolation of the coarse grid. Finally, a small neural network is trained to predict a label from the point-wise feature representation.

2.3 Hyperparameters

We add our custom configuration files mmdetection for our own needs. We change the number of classes to 1 since we only have 1 class. Also, we change the coco instances in the source code to our class name. Lastly, we make the maximum epoch larger (200). The rest of the other settings use the default setting in mmdetection.

3 Experimental Results

model	backbone	multiscale	lr schd	epoch	mAP
mask	r50	✓	3x	196	0.2310
mask	x101	✓	3x	61	0.2304
cascade	r50	✓	3x	10	0.231
cascade	x101	✓	3x	22	0.2298
pointrend	r50	✓	3x	48	**0.2326**
ms rcnn	x101	✗	1x	180	0.2317

Table 1: This is the test result of our first setting. In this setting, we use 20 images for training and 4 images for validation. We apply multi-scale in training when we use Mask R-CNN, Cascade Mask R-CNN and Mask Scoring R-CNN. We also use warmup learning rate scheduler.

model	backbone	multiscale	lr schd	epoch	mAP
mask	r50	✓	3x	199	0.2323
mask	x101	✓	3x	72	0.2316
cascade	r50	✓	3x	198	0.2428
cascade	x101	✓	3x	48	**0.2444**
pointrend	r50	✓	3x	198	0.2439
ms rcnn	x101	✗	1x	147	0.2420

Table 2: This is the test result of our second setting. In this setting, we use all 24 images for training. The rest is same as the first setting.

3.1 Interesting findings

3.1.1 Using all the data for training

We have two settings. Table 1 shows the results that have independent validation data; that is, we split 24 training images into 20 images for training and 4 for validation. Table 2 shows the results that use all 24 training images for training. We found out that all the models perform better when using all the images for training. The only downside of this setting is that training and validation use the same data, so choosing which checkpoint for testing is not easy.

3.1.2 Does validation score matters?

For setting 1, most models get better results when the validation score is higher. For setting 2, the one with a higher validation score does not always yield better testing results. We think the reason is that we use all 24 images for both training and validation, and the model overfits the data.

3.1.3 Backbone comparison

We use two different backbones for Mask R-CNN and Cascade Mask R-CNN: ResNet50 and ResNeXt101. For setting 1, using ResNet50 yields better testing results than using ResNeXt101. However, for setting 2, ResNeXt101 yields better testing results, which is the opposite of setting 1. We think this is because ResNeXt101 is a larger model compared to ResNet50, so in order to stimulate its full potential, more training data is needed.

4 Summary

4.1 Discussion

4.1.1 ResNet101

In addition to the two backbones ResNet50 and ResNeXt101, in the experimental results, we have also tried ReNet101, which is what the baseline use. However, the results are terrible. We have tuned the learning rate and learning rate scheduler, but the results are worse than ResNet50 and ResNeXt101. We do not know the reason, but at least we know ResNet101 may not suit our task.

4.1.2 QueryInst

We have also tried out a recent method QueryInst [4] which is proposed in ICCV 2021. According to the original paper, this is a simple and effective query-based instance segmentation method driven by parallel supervision on dynamic mask heads. However, we find this model hard to train. In the training phase, the validation segmentation results are mostly mAP 0. We find out that query-based approaches are much less robust than Faster / Mask R-CNN in terms of training and optimization. Furthermore, we do not have much time to tune this method due to the time limit, so we give up on this. Nevertheless, if this method works, we may get a better result on the test set.

4.1.3 COCO stuff

We observe that there are some recent methods [9, 11] that rely on COCO-stuff annotation instead of the original COCO methods. Furthermore, our data may not be suitable for these methods since it is impossible to transform our binary mask into a COCO-stuff annotation format.

4.2 Conclusion

In this homework, we have tried four different instance segmentation methods: Mask R-CNN, Cascade Mask R-CNN, PointRend, and Mask Scoring R-CNN. We get the best testing results when using 24 images as training data and apply Cascade Mask R-CNN with ResNeXt101 as the backbone.

References

[1] J. Brooks. Image semantics, 2019. URL https://github.com/jsbroks/imantics.

[2] Z. Cai and N. Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, page 1–1, 2019. ISSN 1939-3539. doi: 10.1109/tpami.2019.2956516. URL http://dx.doi.org/10.1109/tpami.2019.2956516.

[3] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[4] Y. Fang, S. Yang, X. Wang, Y. Li, C. Fang, Y. Shan, B. Feng, and W. Liu. Instances as queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6910–6919, October 2021.

[5] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[6] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang. Mask scoring r-cnn. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] A. Kirillov, Y. Wu, K. He, and R. Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 9799–9808, 2020.

[8] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3431–3440, 2015.

[9] S. Qiao, L.-C. Chen, and A. Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10213–10224, 2021.

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[11] T. Vu, K. Haeyong, and C. D. Yoo. Scnet: Training inference sample consistency for instance segmentation. In *AAAI*, 2021.