Visual Recognition using Deep Learning
Nuclei Segmentation

Department : IMM, Yi-Cheng Hung, ID : 310653005

September 18, 2022

Github

Github : https://github.com/YCHung1998/Instance-Segmentaion.git

Introduction
This assignment is part of a series of projects for the course Selected Topics in Visual Recognition using Deep Learning. This time it is instance segmentation problem. The goal is to grasp the nucleus in the cell and separate each nucleus independently. The collected data has been preliminarily collated, the size is 1000×1000. After the cells are stained, the position and number of the cell nuclei can be seen with the naked eye (dark color). The total number of data is 24+6 images, 24 records are the training set, and 6 records are the test set. This project can be a tool to rescue and help scientists. Instead of consuming a lot of manpower to count, it is better to use machines to assist us and make life more convenient. Below I will introduce in detail the methods I used and implemented.

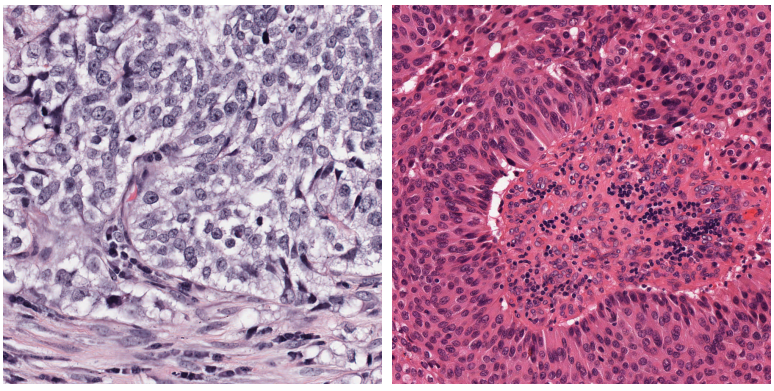


Figure 1: Raw training image data

0. Data preview

The training data will be given an original image picture and a mask folder, corresponding to each independent cell nucleus. In the beginning, observe how many targets and sizes need to be detected for each photo. According to statistics, the average is about 608 sheets. In the distribution, the number of nuclei in a small number of photos far exceeds the average. According to the line chart, we still use the number of nuclei in most photos to determine the maximum predictive number of a future image (take 500) to include the nuclei in most images. (You can also see the test data)

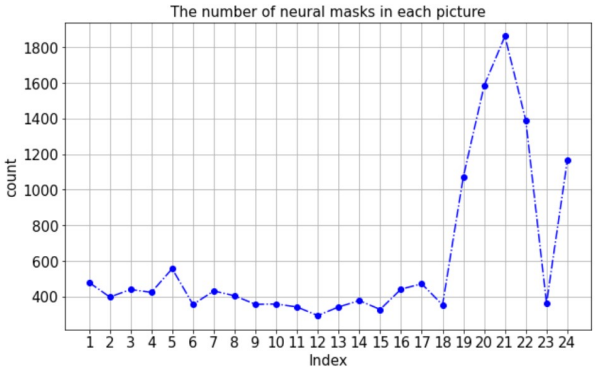


Figure 2: Line chart

In my opinion, the difference in target size is due to the observation under different magnifications of the microscope

1. Data preprocessing

(a) Transform The following list is transform on image in order:

- RandomCrop (0.5, 0.5) : Crop the image to quarters
- ResizeShortestEdge : resize the shortest edge to 608, the other is 800.
  (Notice : we want to obeserve more detail so enlarge the size . On the other hand, in order to meet the requirement of the back bone down sample for 5, the size width and height must divisible by 2^5)
- RandomSaturation : Randomly transforms saturation of an RGB image. Input images are assumed to have 'RGB' channel order. (something contradiction here but still want to try)
- Randomfilp : in prob. p=0.5 at horizontal direction. (default)

2. Model select

- Complain
  In the beginning, I tried hard to install the package which mention in slides hw3 (Mask_RCNN) and (Pytorch-mmdetection), but face something difficulty in the versions problem or the root can't be find in the new version.(Especially mask_RCNN)) This project implement the major model is Mack-RCNN. Mask-RCNN are succeed on the Faster-RCNN. In order to segment the object and clearly split each single objects, the function of the predict mask is attached to the end of PoI-Align layer in Mask-RCNN. All we used the model is based on the detectron2, and it has version problem. windows version and linux version. I have tried both versions and final I select the linux environment (because of my computer can't afford it).

This model is mainly for Mask-RCNN. What is the Mask RCNN? It inherits the bounding box Regression published by 2014 RCNN, and improves the ROI pooling proposed by 2015 Fast RCNN to RoI Align, which improves the accuracy of image segmentation. And in order to maintain the effect of the original problem classification, in the application of RPN, the Faster RCNN released in 2016 was also obtained. Throughout the article, mask rcnn integrates the concepts of previous generations of rcnn family, and has better development in instance segmentation. These effective content must be thankful to the author, so that we have so many resources to use.

There are some model structures in COCO-InstanceSegmentation for reference. Before selected the model, let's introduce meaning of the model name.

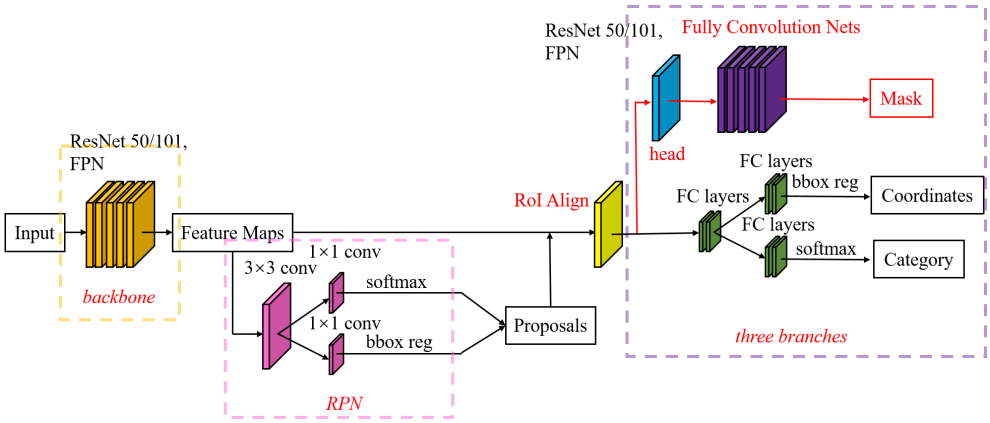


Figure 3: Mask-RCNN Structure

For Faster/Mask R-CNN, It provide baselines based on 3 different backbone combinations:

- **FPN**: Use a ResNet+FPN backbone with standard conv and FC heads for mask and box prediction, respectively. It obtains the best speed/accuracy tradeoff, but the other two are still useful for research.
- **C4**: Use a ResNet conv4 backbone with conv5 head. The original baseline in the Faster R-CNN paper.
- **DC5** (Dilated-C5): Use a ResNet conv5 backbone with dilations in conv5, and standard conv and FC heads for mask and box prediction, respectively. This is used by the Deformable ConvNet paper. .

Finally, we take COCO-InstanceSegmentation/mask_rcnn_R_101_C4_3x.yaml'

COCO Instance Segmentation Baselines with Mask R-CNN

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	mask AP	model id	download
R50-C4	1x	0.584	0.110	5.2	36.8	32.2	137259246	model | metrics
R50-DC5	1x	0.471	0.076	6.5	38.3	34.2	137260150	model | metrics
R50-FPN	1x	0.261	0.043	3.4	38.6	35.2	137260431	model | metrics
R50-C4	3x	0.575	0.111	5.2	39.8	34.4	137849525	model | metrics
R50-DC5	3x	0.470	0.076	6.5	40.0	35.9	137849551	model | metrics
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2	137849600	model | metrics
R101-C4	3x	0.652	0.145	6.3	42.6	36.7	138363239	model | metrics
R101-DC5	3x	0.545	0.092	7.6	41.9	37.3	138363294	model | metrics
R101-FPN	3x	0.340	0.056	4.6	42.9	38.6	138205316	model | metrics
X101-FPN	3x	0.690	0.103	7.2	44.3	39.5	139653917	model | metrics

Figure 4: Model load

One of the reasons why fpn is not used here is that there are no complicated interference details in the image, the test effect is not as good as C4, and the depth of the backbone finally chooses resnet50 instead of resnet101. The reason is that under the same settings, the performance is not much difference between the two models. If the data is small, I am afraid that the parameter training is insufficient.

3. Main step

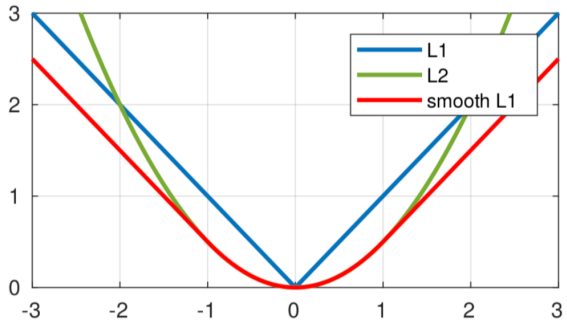
RPN step

1. Generate 9 anchor boxes, generated at each specified point.
2. Calculate the IOU value between the anchor and ground truth within the boundary.
3. Exclude anchors from 0.3 to 0.7 (<0.3 background, >0.7 foreground, other settings are ignored).
4. Calculate the coordinate transformation corresponding to the anchor.
5. Carry out anchor frame correction.

smoothL1 loss : a part of RPN loss (regression term)

$$L_1(x) = x^2, \qquad L_2(x) = |x|$$

$$smooth_{L_1}(x) \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases}, \quad \frac{d}{dx} smooth_{L_1}(x) = \begin{cases} x, & \text{if } |x| < 1 \\ \pm 1, & \text{otherwise} \end{cases}$$



$smoothL_1$ pros :

1. When the difference between the prediction frame and the ground truth is too large, the gradient value will not be too large.
2. The gradient of $smoothL_1$ exists at $x = 0$.

RPN

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

- λ : cls vs bbox loss balance param.
- N_{cls} : mini-batch size.
- N_{reg} : number of anchor location (~ 2500).
- p_i^* : ground truth label ($p^* L_{reg}$ loss only for **positive** anchors).
- L_{reg} : smoothL1 loss function.
- t_i : Parameterized preiction of bounding box.

  $t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a$
  $t_w = \log(w/w_a), \quad t_h = \log(h/h_a)$
  x, y, w, h is box center, width and height.
- t_i^* : Parameterized ground truth of closest bbox.

RoI Pooling vs Align

RoIAlign is proposed to solve the problem of regional mismatch in RoI Pooling in Faster R-CNN. Let's take an example to illustrate what is regional mismatch. The following figure shows : 1. After adjust the location of the anchor box in the feature map The area mismatch problem of RoI Pooling is caused by the rounding operation in the RoI Pooling process.

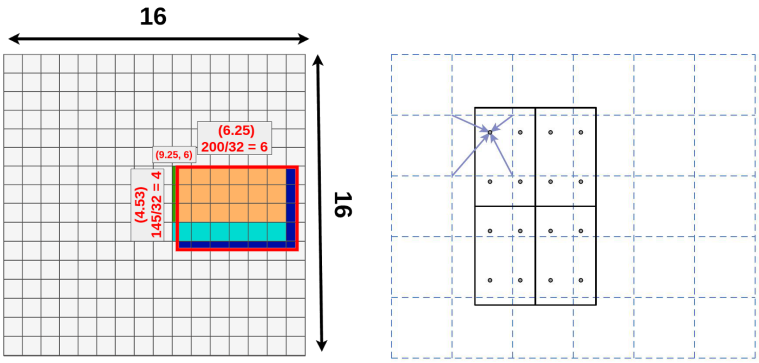


Figure 5: On the lfet hand side is RoIPooling method, the other is RoIAlign method. In RoIPooling, floating point issues caused a slight deviation in the selected area. After correcting to the complete grid, redundant areas appeared, ignoring potentially important information. And RoIAlign uses bilinear interpolation to retain the information of adjacent points, and more reasonably presents the value of the selected area to solve this problem. It also ensures that classification problems can be maintained and increase the mask accuracy in the results(RoIAlign.gif calculation process)

Total loss

The multi-task loss function of Mask R-CNN combines the loss of classification, localization and segmentation mask: $L = L_{cls} + L_{box} + L_{mask}$, where L_{cls} and L_{box} are same as in Faster R-CNN. The mask branch generates a mask of dimension $m \times m$ for each RoI and each class; K classes in total. Thus, the total output is of size Km^2. In this project k=1.

Because the model is trying to learn a mask for each class, there is no competition among classes for generating masks.

L_{mask} : is defined as the average binary cross-entropy loss, only including $k-th$ mask if the region is associated with the ground truth class k.

$$-\frac{1}{m^2}\sum_{1\le i,j\le m}[y_{ij}\ln\hat{y}_{ij}^k + (1-y_{ij})\ln(1-\hat{y}_{ij}^k)]$$

where y_{ij} is the label of a cell (i,j) in the true mask for the region of size $m \times m$; $\hat{y}_{ij}^k$ is the predicted value of the same cell in the mask learned for the ground-truth class k.

Training Setting

- the achor generator size : (8, 16, 32, 64, 128)
  the goal is not really big so check as smaller as the anchor can.
- set the epoch (24 iter/per epoch)=100.

- set the learning rate with warming up and the lr scheduler.
  base lr = 0.1 and in the first 3 epoch (72 iter), given specific steps to renew the learning rate by multiply $\gamma = 0.1$.

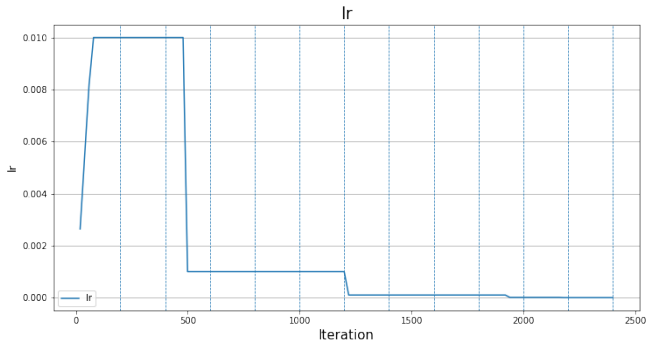


Figure 6: learning rate warm up and decay.

- save the model for every period 200 iter.
  (On the last figure the blue vertical line is the point that I had saved.)

Inference Setting

- DETECTIONS_PER_IMAGE = 500.
  The number of nuclei in the test set data I observed is not as large as the gap between the number of nuclei in training and training. A quick comparison with the naked eye should be within 500 800 and you should be able to grasp it.(And I'd try 500, 700, 800 not change too much.)

- Implement the augmentation

4. Results

The following table show some result that I had tried, the first use the backbone ResNet101 performance not bad, but in the ResNet50 performance more well, and use less parameter. Another trial is add the random saturation from the given range in the data-preprocessing. I originally thought that the target would appear deeper in the picture after dyeing. By adjusting the saturation, the color difference of the data can be more distinguished, and the model identification can be assisted. But the result is not as simple as I thought.(Maybe need to give more range trials.)

model	acc	Remark
mask_rcnn_R_101_C4_3x	0.244082	1599-iter
mask_rcnn_R_50_C4_1x	**0.244597**	1599-iter
mask_rcnn_R_50_C4_1x	0.244517	2199-iter
mask_rcnn_R_50_C4_1x	0.242970	saturation $2 \sim 10$
mask_rcnn_R_50_C4_1x	0.243030	saturation $0.5 \sim 2$
mask_rcnn_R_50_FPN_1x	0.231493	model_final

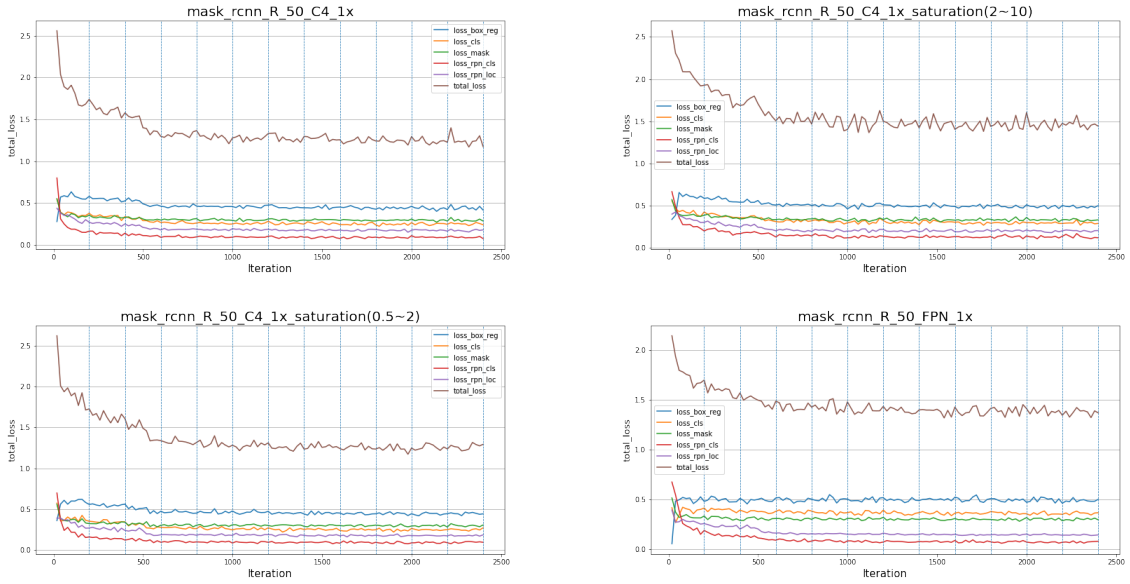


Figure 7: The training process diagrams drawn according to different settings and models are all observed with the value of loss. Each blue line of the adjustment line represents the model parameter record accessed every 200 steps, through the graph and the straight line relationship Select the parameters I want to do inference

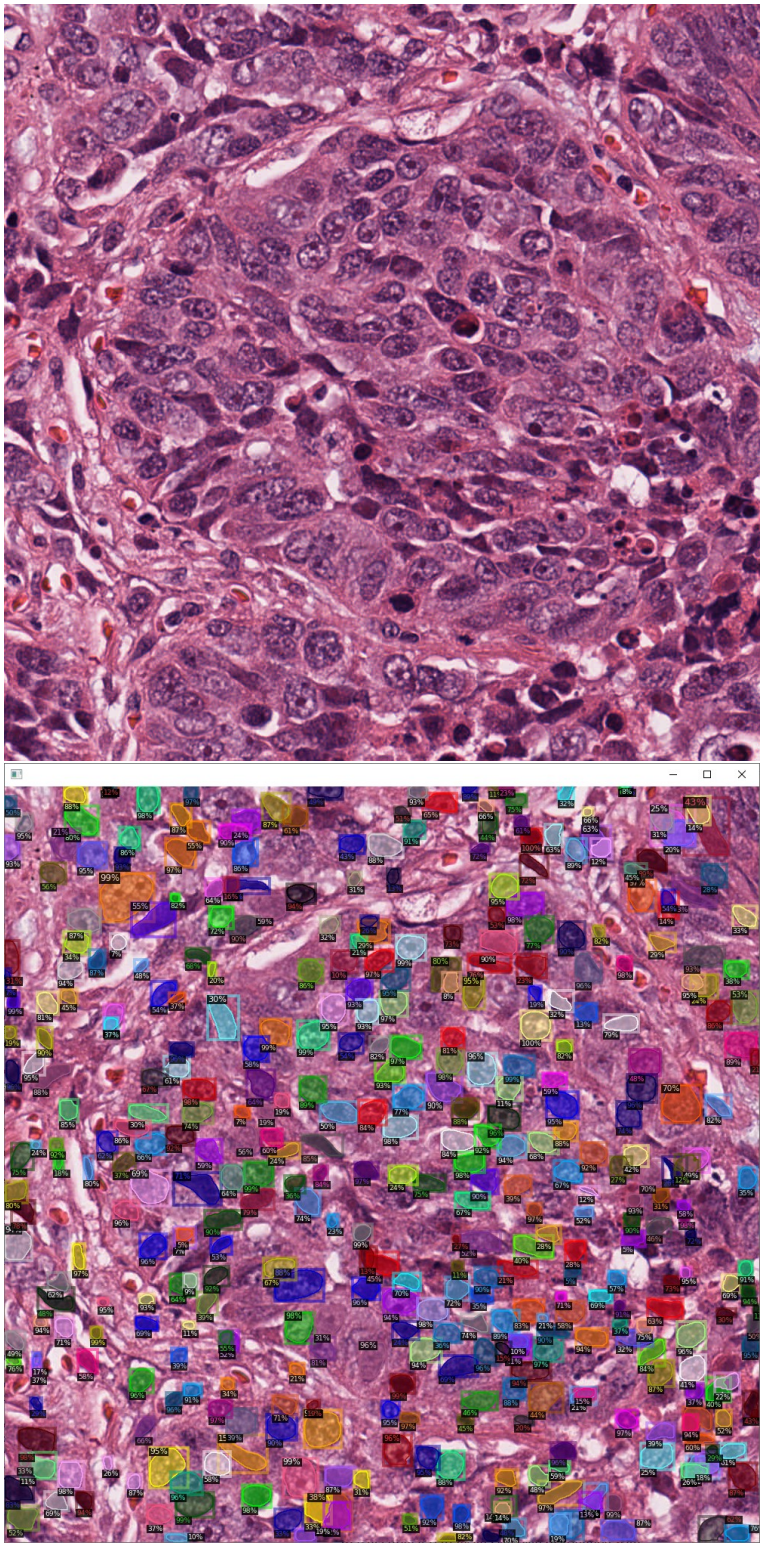


Figure 8: The upload is Test image (TCGA-50-5931-01Z-00-DX1.png) and the below is prediction results.

#	SCORE	FILENAME	SUBMISSION DATE	SIZE (BYTES)	STATUS	✔	
1	0.243093	ver1.zip	12/13/2021 17:50:51	87653	Finished		+
2	0.231493	ver2.zip	12/14/2021 06:01:19	93225	Finished		+
3	0.243922	ver2.zip	12/14/2021 06:03:26	93345	Finished		+
4	0.244082	ver2_0001599.zip	12/14/2021 06:07:42	93474	Finished		+
5	0.243948	ver2_0002199.zip	12/14/2021 06:12:35	93594	Finished		+
6	0.243973	ver2_0001199.zip	12/14/2021 08:31:05	95565	Finished		+
7	0.231374	ver3_00001399.zip	12/14/2021 12:05:58	99749	Finished		+
8	0.244035	ver2_0001399.zip	12/14/2021 12:16:42	99865	Finished		+
9	0.244088	test_notrecord.zip	12/14/2021 14:15:59	103317	Finished		+
10	0.244088	new_ver1944.zip	12/14/2021 14:55:10	104289	Finished		+
11	0.244088	new_ver1944_700.zip	12/14/2021 15:06:28	104893	Finished		+
12	0.244313	new_ver2_2243_0001799.zip	12/15/2021 08:06:42	123529	Finished		+
13	0.244517	new_ver2_2243_0002199.zip	12/15/2021 08:42:42	124313	Finished		+
14	0.234379	new_ver2_2243_0001199.zip	12/15/2021 08:53:54	125071	Finished		+
15	0.244597	new_ver2_2243_0001599.zip	12/15/2021 09:42:31	126757	Finished	✔	+

Figure 9: Submission result

Reference

[1.] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[2.] Ross Girshick; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448

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