Selected Topics in Visual Recognition using Deep Learning
HW3 Report: Nuclei Segmentation

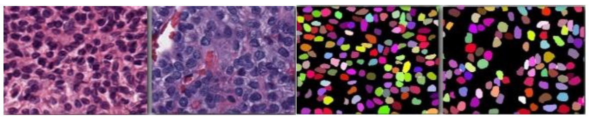
310706004 Hsin-Ying Lien

I. Introduction

In this assignment, I have tried to perform nuclei segmentation with two object segmentation libraries in frameworks of Keras and Pytorch. Since I had insufficient memory problem in Keras, I finally used Facebook AI Research's Python API—Detectron2 to finish the task. I used Mask R-CNN model based on ResNeXt-101 and fine-tune it to accomplish the object segmentation on the given nuclei dataset. It reached a final mean average precision about 0.24237 on the test dataset.

Code GitHub link: https://github.com/angelalien/VRDL-HW3

Dataset: Nuclei segmentation dataset
(training: 24 images with 14,598 nuclei, test: 6 images with 2,360 nuclei)



II. Methodology

I implement Detectron2 to perform nuclei segmentation by referring to similar task program on GitHub. I first downloaded the train dataset from Google Drive, and convert the mask information into a COCO format json file. Then I moved the image file into a new folder for training. The final file structure is shown below.

```
|— dataset
|   |— test   #contains all test images
|       |— TCGA-50-5931-01Z-00-DX1.png
|       |— …
|   |— train   #contains all train images and masks
|       |— TCGA-18-5592-01Z-00-DX1
|           |— images
|           |— masks
|       |— …
|   |— train_new   #contains all train images
|       |— TCGA-18-5592-01Z-00-DX1.png
|       |— …
|   |—test_img_ids.json
|— nucleus_cocoformat.json
```

1. Data Pre-process

1) Read image files, and save the information of width, height and name.
2) Read mask files, and use OpenCV's function—threshold and findContours to get the segmentation information.
3) Calculate the area and bounding box coordinates.
4) Save the annotation information into a json file—*nucleus_cocoformat.json*.

2. Model architecture

The final model I used is Mask R-CNN based on ResNeXt-101 + feature pyramid network (FPN) backbone with standard conv and FC heads for mask and box prediction, pretrained on ImageNet. Mask R-CNN, which is an improvement of Faster R-CNN, consists of Faster R-CNN and instance segmentation structure. Finally, the fully connected layer outputs segmentation results by performing box regression and classification.

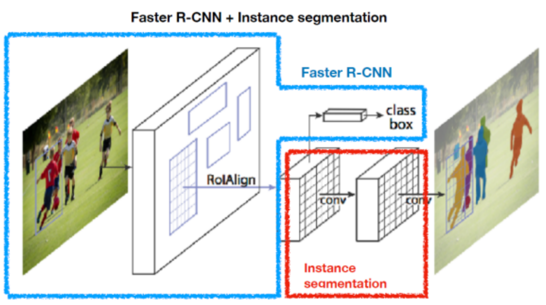
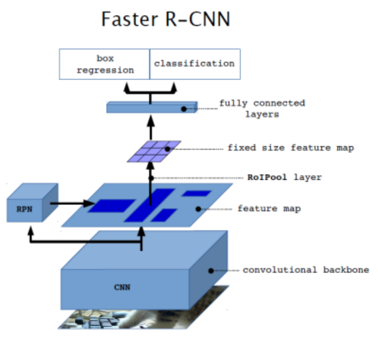
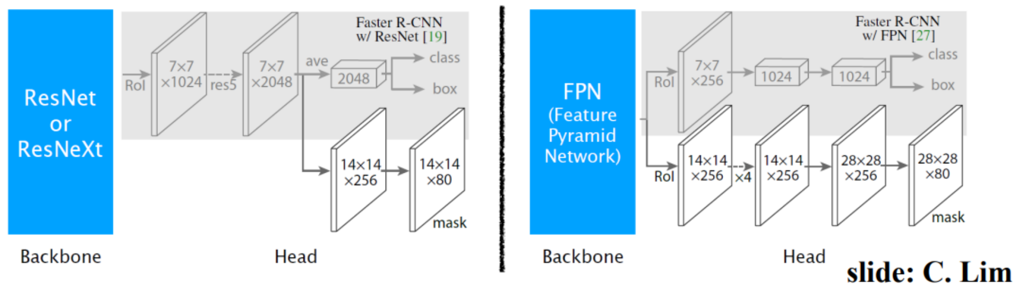
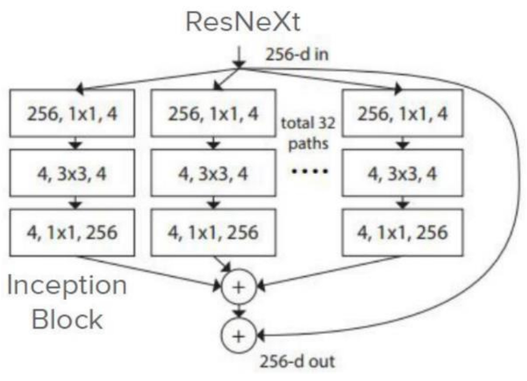


Figure 1. The **Mask R-CNN** framework for instance segmentation.







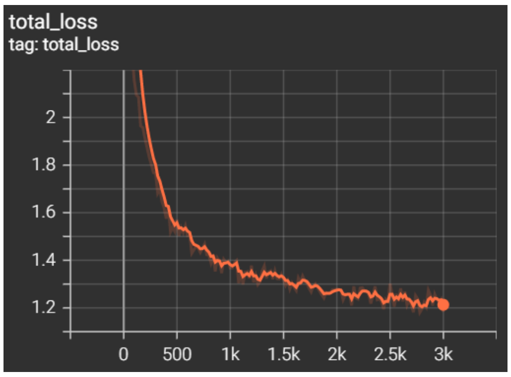
3. Hyperparameters

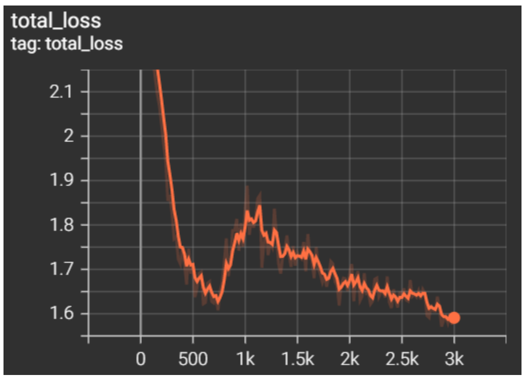
DATALOADER.NUM_WORKERS = 2
SOLVER.IMS_PER_BATCH = 2
SOLVER.BASE_LR = 0.00025
SOLVER.MAX_ITER = 3000
MODEL.ROI_HEADS.NUM_CLASSES = 1
MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE=128
MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5

III. Experimental Results and Observations

I have tried to adjust the hyperparameters setting to fine tune my model and reach a higher performance. During the process, I got some observation results.

1) Reducing anchor size leads to fluctuation of loss curve
   Since the size of nuclei is small, I try to reduce the anchor size (ANCHOR_GENERATOR.SIZES), hoping to actually catch the cell better. The original value is [[32], [64], [128], [256], [512]], and I reduce it to [[8], [16], [32], [64], [128]]. I find that the loss curve will rise for a short time at about 1000 iteration and then descend after reducing the anchor size (right image). Maybe it is because the small anchor size just fit the cell size to catch feature well, the loss decreasing severely at the beginning, with a little rising after that during training.





2) Some post process of masks may not help for the performance
   At the beginning, I use a function to clean overlaps between bounding boxes, fill small holes, and smooth boundaries of the segmentation results. However, after I try to skip the preprocessing, I find that the mean average precision even increases by 0.0001. I supposed that the method may filter out some proper mask results so that lower the output score.
3) Batch size seems to make no difference to the training performance
   I have tried batch size with 64, 128 and 512. The mean average precision of the output is closed. Therefore, the batch size doesn’t matter so much.

4) Data augmentation may not help

   I try to use Detectron2's transforms function to perform data augmentation to the dataset. By referring to the discussion of 2018 Data Science Bowl Kaggle competition, I resize the image to 512*512 pixels, randomly flip, rotate and crop the image. However, the result doesn't improve. I think that the effect of data augmentation may be more obvious to a smaller dataset or in a longer train iteration period.

5) Reducing test threshold below 0.5 doesn't help

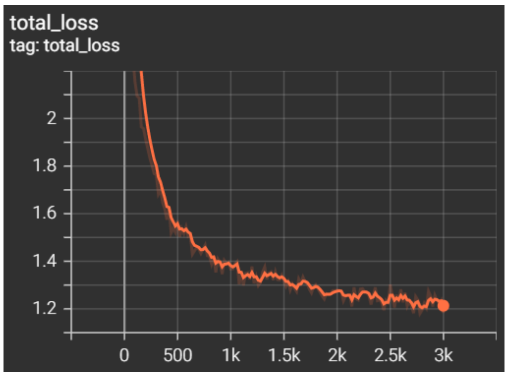
   SCORE_THRESH_TEST value is the threshold used to filter out low-scored bounding boxes predicted by the model. I try to set the value to 0, 0.1, and 0.5, but it makes no difference to the mean average precision of the output. I think that it is because the detection results with lower score are wrong, which will not be helpful for the precision score.

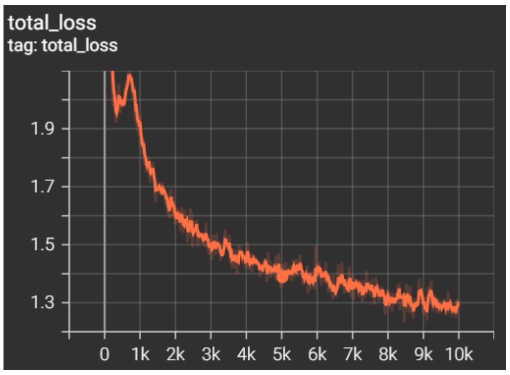
6) Increasing the detection object number per image doesn't help

   The original detection object number per image (TEST.DETECTIONS_PER_IMAGE) is 100. I try to set the value to 2000, but the result remains the same. I supposed that it is because the model doesn't detect so much objects.

7) ResNeXt-101 converge fast in early iteration than ResNet-50

   The loss of ResNeXt-101(left image) gradually converge until 3000 iteration, and the loss of ResNet-50(right image) gradually converge until 10000 iteration. It takes 3 hours and 1 hour to run 3000 iteration with ResNeXt-101 and ResNet-50. We can find that ResNeXt-101 converge in earlier iteration, but takes more time to run each iteration.





IV. Summary

Nowadays, it’s easy for us to implement pretrained models and fine-tune it to deal with object segmentation tasks. However, we have to consider our available GPU devices and memory while training with training data of large memory. At this moment, proper choose of object detection tools and some hyperparameter setting play an important role in avoiding this situation if we are not equipped with advanced hardware environment. Besides, I find that it doesn’t always improve the performance to use other’s suggested hyperparameter setting of the similar nuclei segmentation task on the Internet. We can refer to those advices, but have to make some experiments to find the most proper setting for our dataset. As a result, proper choosing and experiments of hyperparameter setting will lead to better performance.

V. Reference

Detectron2 Beginner's Tutorial
https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5

NVRDL2019FALL Data Science Bowl 2018
https://github.com/vbnmzxc9513/Nuclei-detection_detectron2

2018 Data Science Bowl | Kaggle
https://www.kaggle.com/c/data-science-bowl-2018/discussion/56326