VRDL HW3 Report

Author: YUN_JU, HO

Task: Instance Segmentation - Nuclei segmentation

GitHub link

2021_VRDL_HW3: https://github.com/LordHo/2021_VRDL_HW3.git

Introduction

Train an instance segmentation model to detect and segment all the nuclei in the image. In this work, the nuclei in each image are small and enormous. In the Kaggle competition, the main directions are using U-Net and Mask R-CNN. I finally choose Mask R-CNN as my model.

Dataset

Nuclear segmentation dataset contains 24 training images with 14,598 nuclear and 6 test images with 2,360 nuclear.

Dataset download
https://drive.google.com/file/d/1nEJ7NTtHcCHNQqUXaoPk55VH3Uwh4QGG/view

Kaggle competition
https://www.kaggle.com/c/data-science-bowl-2018

Environment

OS: Window 10

GPU: NVIDIA GeForce RTX 2080 Ti

Methodology

- **Data Pre-process**

  Due to the limit of my GPU, I divide image into 4 parts and do the same transform on mask. Dividing also decrease the redundant background in mask. After read image and mask, I normalize image and mask to range [0, 1] with dividing by 255. The finally package mask and other information into target. This part can see in `dataset.py`.

- **Model Architecture**

  I use Mask R-CNN (ResNet-50 + FPN) and Mask R-CNN (ResNet-101 + FPN),

and can find the model I use in `model.py`.

- **Hyperparameters**

  I use Adam with learning rate 1e-4 and 1e-5 as my optimizer. The first 60 epochs use learning rate 1e-4 and remaining epochs use learning rate 1e-5. This is done by lr_scheduler provided by torchvision and can see in `model.py`.

Data Post-process

- **Reconstruct the divided parts**

  In training part, I divide image and mask into 4 parts due to the limit of my GPU memory. Identically, I divide the input image into 4 parts, then predict each part. When all parts in image are done, I concatenate each parts into correct position and will find some nuclei has been divided. However, I didn't figure out good solution to concatenate, so I only concatenate each parts into correct position and didn't deal with some divided nuclei.

- **Transform to answer format**

  Format is same as COCO Dataset and set the all category_id to 1. The following is the structure of answer format.


```
[
  {"image_id": 1,
   "bbox": [342.7453918457031,
            123.58514404296875,
            19.428497314453125,
            19.685256958007812],
   "score": 0.9901022911071777,
   "category_id": 1,
   "segmentation":
     {"size": [1000, 1000],
      "counts": "mR_:7nn05M2N1O1O1O100O10000000O101N1O2N2N3Kbm^c
     },

  {"image_id": 1,
   "bbox": [245.0045928955078,
            326.71768188476562,
            24.647659301757812,
            18.889068603515625],
   "score": 0.9896910190582275,
   "category_id": 1,
   "segmentation":
     {"size": [1000, 1000],
      "counts": "fb_76on05M2N101O0O101O000O10000000000010001O0020
```


The counts in segmentation is the predicted mask encode by RLE. The following is the encode process.

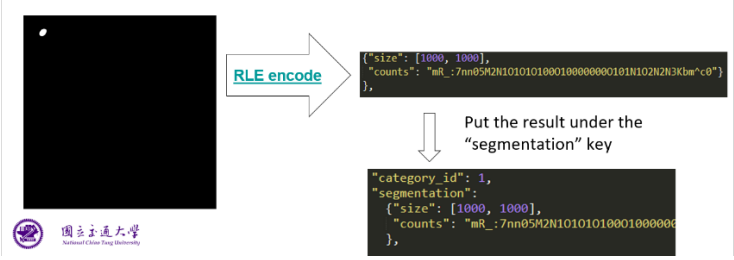

Table of Experiment Results

Using AP@0.5, IoU=.50 means set .50 as threshold of IoU.

Model	Backbone	AP
Mask R-CNN	ResNet-50 + FPN	0.20628
Mask R-CNN	ResNet-101 + FPN	0.14182

Best Model Weights

Mask R-CNN (ResNet-50 + FPN)

https://drive.google.com/file/d/17wpBJ5M0FB3xFWNmLOTPRsfdWl6Y4Ebu/view?usp=sharing

Summary

The Mask R-CNN with ResNet-101 + FPN should be better or at least perform as great as Mask R-CNN with ResNet-50 + FPN, but the result shows the opposite result. This must be something wrong, but I still not find the problem so far. I try divide image into 4 parts and 16 parts to train and to predict, divide into 16 parts have an advantage with limited GPU memory, but result almost as same as divide into 4 parts. In addition, divide into 16 parts will have more severe on the problem that some nuclei have been divided.

Reference

1. Kaggle 5th place solution (based only on Mask-RCNN)
   https://www.kaggle.com/c/data-science-bowl-2018/discussion/56326
2. mask-rcnn and instance segmentation network
   https://www.kaggle.com/c/data-science-bowl-2018/discussion/47690
3. Top 5, also with the weights
   https://www.kaggle.com/c/data-science-bowl-2018/discussion/56316
4. 10th place Code+Datasets (LB: 0.591) Mask R-CNN single model
   https://www.kaggle.com/c/data-science-bowl-2018/discussion/56238
5. 1st place solution
   https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741
   https://github.com/selimsef/dsb2018_topcoders/
6. Torchvision
   https://pytorch.org/vision/stable/index.html