VRDL HW3 Report

310551055 鄭伯俞

Github

- Repo Link: https://github.com/Nemo1999/NYCU_VRDL_HW3.git

Reference

- Mask RCNN arXiv:1703.06870 [cs.CV]
- Feature Pyramid Networks for Object Detection arXiv:1612.03144 [cs.CV]
- Detectron2 Github Repo

Introduction

In HW3, I use Mask RCNN to perform instance segmentation on the nucleus dataset. I experiment on different augmentation parameters, adjusting on the size of the crop region. Suitable cropping size helps the model focus on the small nucleuses and improve the performance of the mask-rcnn model. My final model achieved 0.2424 on the testing data and ranked 30 on the leaderboard.

Methodology

Data Preprocess

In the preprocessing step, I define all the pipelines in `makefile`. First, the script for target `getdataset` download and unzip the dataset from google drive. Second, the script for target `preprocess_mask` triggers a python process (defined in `preprocess_mask.py`)that reads and converts every bitmask annotation of an image into coco's rle format and saves the result in a pickle file in the image directory.

Finally, the module `nucleus_dataset.py` reads the preprocessed annotation file and packs all the information into detectron dataset format, and registers the dataset into detectron2's `DatasetCatalog` and `MetadataCatalog`. The training script can simply import `nucleus_dataset.py` and specify our training dataset by the registered name.

Data Augmentation

In the data augmentation step I apply `horizontal random flip` and `random crop` with relative width and height of `(0.5, 0.5)` in my final setting. **I found out that the testing**

result is sensitive to the relative size of the randomly crop region, as shown in the experiment section.

Model Architecture

The architecture I used is Mask-RCNN originally proposed by Kaiming He. Model parameter is defined by the pretrained configuration of detectron2 baselines, with **pretrained Resnet 101 backbone** and using FPN to generate multiscaled feature maps. FPN is an important improvement from the original Mask-RCNN implementation, because it combines the strong semantic information in the low-resolution layers and the weak semantic information in the high-resolution layers via the top-down pathway.

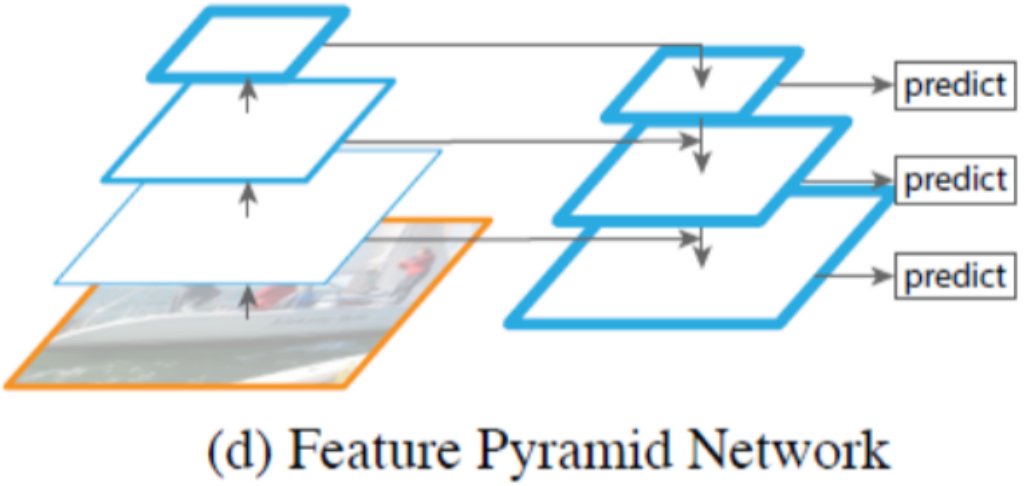

Image source:
https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610

Hyperparameters

Hyperparameters used in my final setting is as follows:

Name	Value	Explaination
SOLVER.MAX_ITER	3000	max training iteration (The best model is the checkpoint from 2249 iterations due to overfitting)
SOLVER.BASE_LR	0.00125	Learning rate is scale linearly from the default value according to IMS_PER_BATCH
SOLVER.STEPS	(2000, 2750)	Steps to downscale LR
SOLVER.IMS_PER_BATCH	1	due to GPU memory constraint
BACKBONE	build_resnet_fpn_backbone	

MODEL_ROI_MASK_HEAD. POOLER_RESOLUTION	28	This is doubled from the original value for better mask prediction.

Experiment Results

At the augmentation step, I tried different random cropping region sizes using the same hyperparameters and found out the relative size `(0.5, 0.5)` gives significant performance boost.

	crop size=0.7	crop size=0.5	crop size=0.3	crop size=0.2
Best Test mAP	0.231	**0.242**	0.233	0.233

Result Analysis

At first glance, this result is quite surprising because the cropping size should not affect the model's performance on small objects such as nuclei (they are too small to be clipped by cropping).

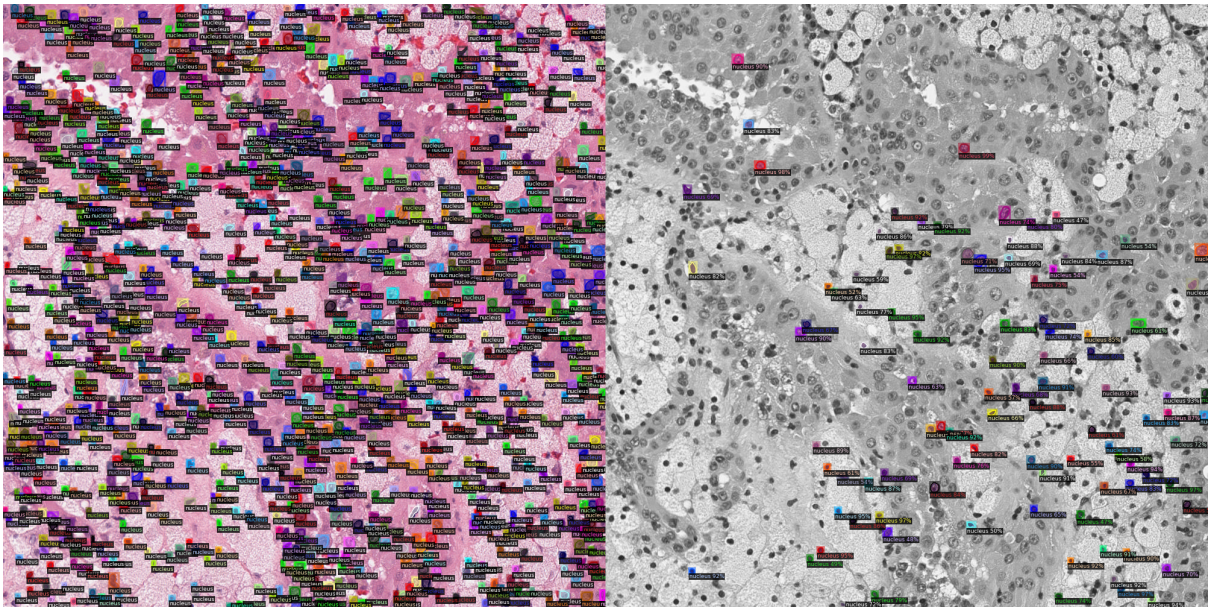

Fig 1. The model struggles to detect smaller nuclei. (left: Ground Truth , right: Prediction)

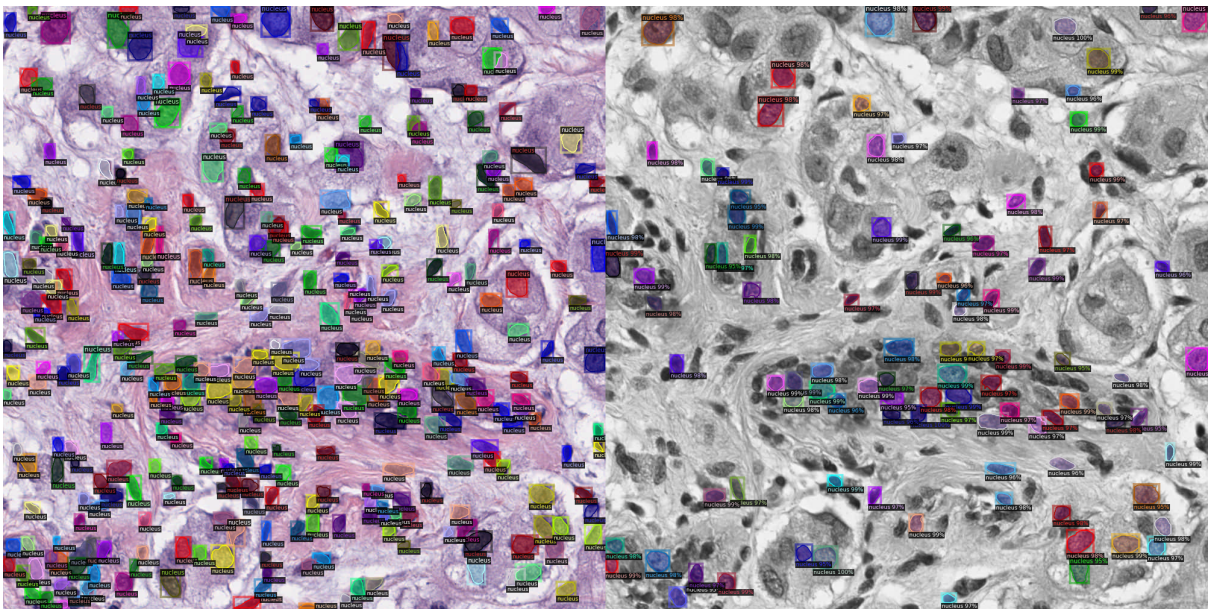

Fig2. The model does better on medium-size nuclei at the center , and struggles to detect larger nucleuses at the top portion.
(left: Ground Truth, right: Prediction)

After I visualized the prediction result of the models (Fig1 and Fig2), I found out that the model fails to detect a very large portion of nuclei even in the training images, this may be due to large variation of the size of nucleus in different images. I think that when relative=0.5, random cropping changes the scale of the training image and accidentally better matches the distribution of scale of the nucleuses in the testing set. So I get much higher testing results at a particular cropping size.

In theory, the scale matching problem should be mitigated by the mechanism of FPN, but since we have a very limited amount of training images, the model fails to detect nucleuses with sizes that are too far away from the distribution mode.

A possible test for this explanation is to apply random cropping with random size (eg. randomly select from (0.2,0.2) to (0.7, 0.7)) , which further extends the distribution of nucleus dimension in the training distribution. However, due to time constraints, I didn't finish the test.

Conclusion

In conclusion, during hw3, I learned the theory and implementation behind the mask r-cnn model, and did a little experiment on the hyper-parameters on relative size of random cropping. And find out that model performance depends on the size of the cropping area. I propose a possible explanation to this phenomenon and a testing method for my hypothesis. Overall, my model achieves acceptable results but still has space for improvement.