Visual Recognition using Deep Learning HW3

110550085 房天越

GitHub Repo Link

https://github.com/TianYueh/NYCU-Visual-Recognition-2025-HW3

Introduction

In this homework, we aim to do instance segmentation for microscopic cell images using pure vision model with Mask R-CNN [1]. The core idea is to leverage a pre-trained ResNet50 backbone with a Feature Pyramid Network (FPN) for multi-scale feature extraction [2], then apply Mask R-CNN heads for region proposal, classification, bounding-box regression, and pixel-level mask prediction.

Method

1. Data Pre-processing

    I. Image Loading and Conversion

    Read original TIFF images with OpenCV, convert BGR to RGB, and transform to PyTorch tensors by using torchvision.transforms.functional.to_tensor.

    II. Fixed Resizing

    Resize both images and their corresponding masks to a fixed resolution of 512*512 pixels for training.

    III. Instance Annotation Generation

    For each class mask tiff files, apply cv2.connectedComponents to separate individual instances. Filter out empty or tiny masks. Compute the minimal bounding box (xmin, ymin, xmax, ymax) for each instance to assemble the target dictionary.

    IV. Batch Collation

    Use a function collate_fn to package each image and its multiple targets (boxes, labels, masks) into a tuple format compatible with Mask R-CNN [1].

2. Model Architecture
   I. Backbone

      ResNet50 with FPN [2], initialized from ImageNet pre-trained weights. The FPN merges feature maps from ResNet layers C2–C5 into multi-scale feature maps P2–P6 via lateral connections and upsampling.

   II. Neck (FPN)

      Implements the standard top-down FPN to fuse features across scales.

   III. Region Proposal Network (RPN)

      Slides over FPN feature maps to generate anchor proposals, predicting objectness scores and bounding-box offsets as in [1].

   IV. ROI Heads:

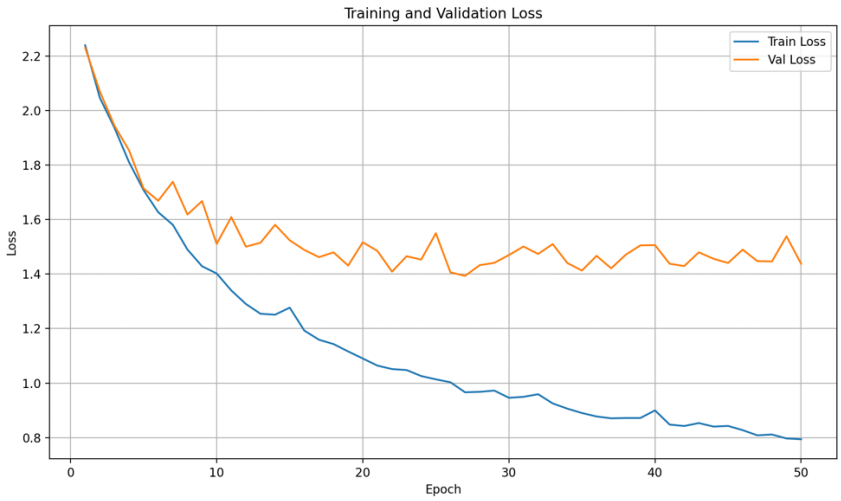
      Box Head: Applies RoI Align on proposals, followed by fully connected layers to output class scores and refined bounding-box coordinates [1].

      Mask Head: For each positive region, applies convolutional layers to predict a 28×28 binary mask for the assigned class [1].

   V. Hyperparameters
      A. Learning Rate: 0.001, optimized with SGD (momentum=0.9, weight decay=5×10^{-4}).
      B. Mixed Precision: Enabled via torch.cuda.amp and GradScaler for faster training and reduced memory footprint.
      C. Batch Size: 1.
      D. Epochs: 50.
      E. Input Size: 512*512 pixels.
      F. DataLoader Settings: num_workers=4, pin_memory=True.

Results

1. Training Curves



From the two curves, we can see the training loss keeps to decline. However, the validation loss stops to converge at about epoch 20, and became the lowest at epoch = 27.

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 2980–2988.

[2] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 2017, pp. 936–944.

[3] H. Hu, J. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 7132–7141.

[4] Z. Zheng, S. Ye, B. Chen, and S. Wang, "Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression," in *Proc. AAAI Conf. Artif. Intell.*, New York, NY, USA, Feb. 2020, pp. 12993–13000.

[5] Torchvision Contributors, "Mask R-CNN in torchvision," GitHub repository, 2025. [Online]. Available: https://github.com/pytorch/vision/blob/main/torchvision/models/detection/mask_rcnn.py

Additional Experiments

1. Different Score Threshold

   For this homework, I also tried to use different score threshold, I tried to use.

Finally, I found that the best result is at score about 0.55. This might because that not too much important details would be deleted, and also not too much redundant instances would be considered.
For the score thresholds of 0.5, 0.55, and 0.6. The number of instances that pass the threshold are 3669, 3315, and 2931.

2. Different Resize Size
   For the resize size, I tried to use 256, 400, 512, and 1024.
   In my assumption, since the cell dataset is very detailed, so I think larger resizing size might have better results.
   However, the best results are at resize = 512.
   This might be because that the larger the resized sizes are, the more possible that the model would tend to focus on some details that are actually redundant, so the best results actually come at a balance of detail finding and redundant details ignoring.
   Also, the time for training at larger size is significant longer than at the smaller sizes.