Visual Recognitionusing Deep Learning

2025 Spring, Homework 3

313551078 吳年茵

Github : `https://github.com/nianyinwu/CV_HW3`

1. Introduction

In this lab, we have to train an instance segmentation model to solve an instance segmentation task to segment the mask of medical cells with 4+1 classes (including background) and design other tricks to improve the model’s performance, such as modifying the architecture. We use a dataset of 209 colored medical images for training and validation and 101 for testing.

I adopted the Mask R-CNN [1] model as my base model from the Pytorch library [2] and ResNet [3] as the backbone of Mask R-CNN. I also use pre-trained weights from ResNet trained on ImageNet dataset [4] to converge better and learn current training data based on previously learned visual features. To improve the model's performance for this task, I experiment with some tricks, such as introducing the module or modifying model parameter, such as detection per image, to have better metrics in average precision 50.

2. Method

A. Data Preprocessing

In the data preprocessing, I split the training data into training and validation with the ratio of 8:2. Due to hardware limitations, if the image has a height or width exceeding 1024, the image and mask are resized such that the long edge is scaled down to 1024 while maintaining the original aspect ratio. Otherwise, the original size is retained. I also do some data preprocessing and augmentation techniques to the data using PyTorch's transforms module [5].

The preprocessing process of training data is as follows:

1. Convert the image into a PyTorch tensor.
2. Randomly jitter the image brightness by up to ±30% of the original value.
3. Convert the image to grayscale with a probability of 0.2.
4. Convert the tensor to float32 and scale pixel values from [0, 255] to [0.0, 1.0].

The preprocessing process of validation and testing data is as follows:

1. Convert the image to a PyTorch tensor.
2. Convert the tensor to float32 and scale pixel values from [0, 255] to [0.0, 1.0].

B. Model Architecture

B.1. Base Architecture

1. Backbone

   The backbone in the Mask R-CNN used in this work is ResNet-50 [3]. It extracts hierarchical and discriminative visual features that guide the model in identifying important regions within the input image.

2. Neck

   After the backbone extracts features, the model employs a Feature Pyramid Network (FPN) to fuse multi-scale features from different backbone layers. This allows the model to effectively handle objects of various sizes by leveraging features at multiple resolutions and, in this part, using an extra batch normalization layer after each convolutional layer.

3. Head

   Subsequently, the model employs a Region Proposal Network (RPN) to generate a set of object proposals by sliding a small convolutional network over the feature maps. For each anchor, it predicts an objectness score and refines the bounding box coordinates. After applying non-maximum suppression (NMS), the top-N proposals are forwarded to the RoI head for further processing.

Finally, the Region of Interest (RoI) head takes the top-N proposals from the RPN and extracts fixed-size feature maps for each proposal using RoI Align. Compared to the model in the previous lab, Faster R-CNN [6], Mask R-CNN incorporates RoI Align instead of RoI Pooling. RoI Align computes the value of each sampling point using bilinear interpolation from nearby grid points on the feature map, effectively avoiding quantization errors and enhancing the alignment accuracy between the input image and extracted features. Then, the extracted features are passed through two parallel branches: one consisting of fully connected layers to perform classification and bounding box regression, and another consisting of a small fully convolutional network to predict a pixel-wise binary mask for each RoI. Specifically, the classification and box regression branch predicts class labels and refines bounding box coordinates to enclose objects tightly. In contrast, the mask branch independently predicts an accurate segmentation mask for each detected instance.

B.2. Modification (including additional experiments modification)

In this section, I will present the modification in the base architecture and the modification of additional experiments. In all experiments, I modified the number of classes in the ROI head from the default 91 classes to 5, including one background class and four foreground object classes, to align with the target dataset. After observing the dataset, I found that most images contain many cell instances. Therefore, during inference, I increased the detections_per_img parameter in the ROI head from 100 to 1000, as the default setting was insufficient to capture all relevant detections in this task.

In additional experiments, I wanted to introduce a Convolutional Block Attention Module (CBAM) [7] into every layer of the backbone network except the first layer. Since CBAM sequentially applies channel and spatial attention, it may help the model learn more informative and fine-grained features along both dimensions, enhancing its ability to focus on meaningful regions. Furthermore, I also conducted experiments comparing models trained with and without data augmentation to avoid the overfitting problem.

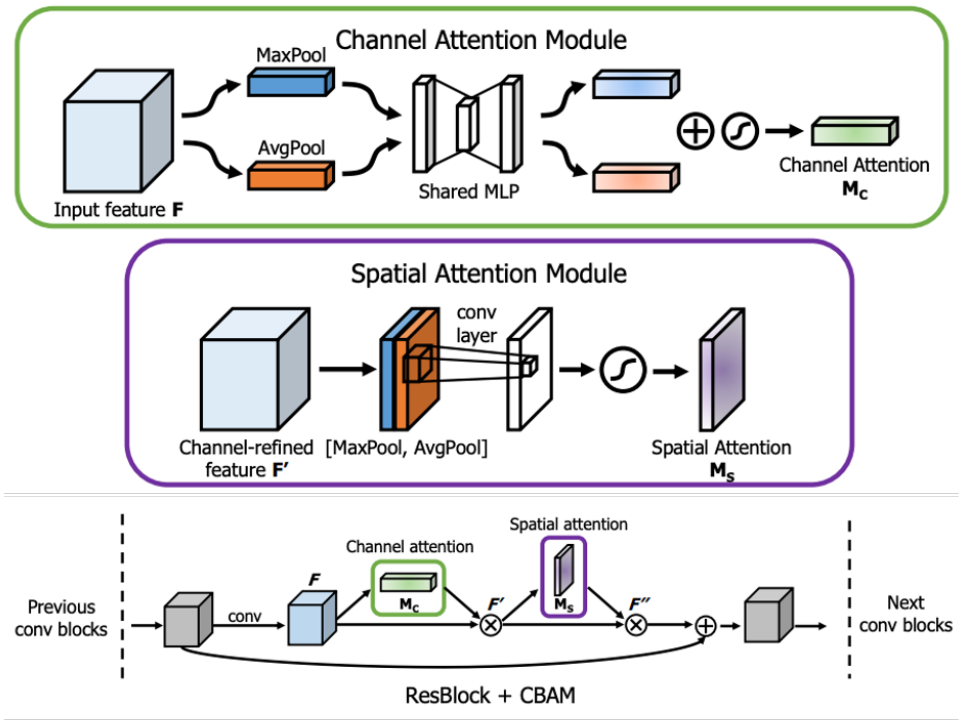


Fig. 1. CBAM module architecture from [7]

C. Hyperparameters Setting

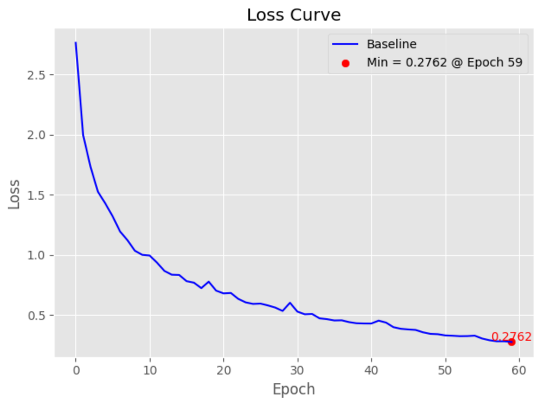
In the experiments, I used the following hyperparameter settings to train the model:

- Learning rate: 1e-4
- Batch size: 1
- Number of epochs: 60
- Optimizer: AdamW with a $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a weight decay of 5e-4.
- Learning rate scheduler: Cosine annealing with linear warmup during the first 25% of the total training steps, implemented using Hugging Face's library [8] .

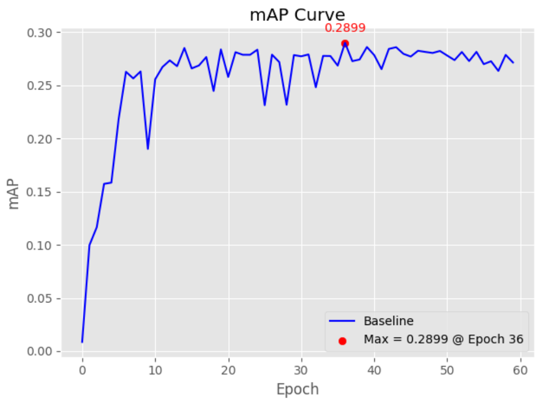
3. Experiment Results

I conducted four experiments to explore better performance: (1) Mask R-CNN only modified the number of classes in the ROI head. (baseline) (2) Mask R-CNN (baseline) with data augmentation. (3) Mask R-CNN added CBAM module in all backbone stages (except layer 1).

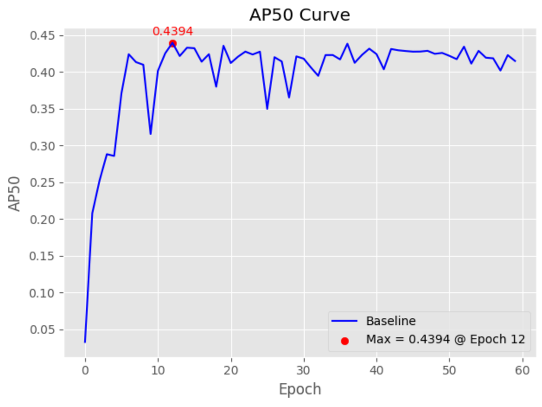
1. Mask R-CNN (baseline)



(a) Loss Curve



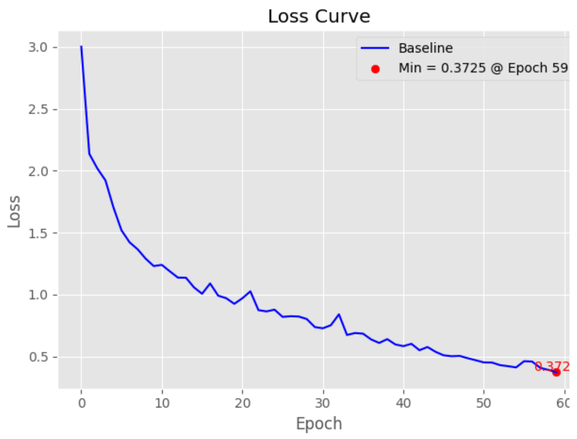
(b) mAP Curve



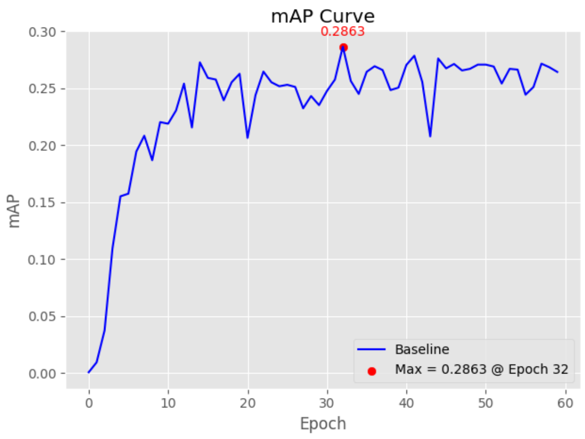
(c) AP50 Curve

In the first experiment, Mask R-CNN (baseline) only modified the number of classes in the ROI head from 91 to 5, and trained the model with ResNet-50 backbone pretrained weights. The loss curve shows that the model converges slowly, the lowest loss is **0.2762**, but in the mAP and AP50 validation curve, we can see that the metrics don't increase after 31 epochs, which has the highest mAP in **0.2899**, and the AP50 in **0.4394**. Thus, I thought this model's generalization ability might still have room to improve. In addition, I submit the result using the baseline model to Codabench, which can achieve a **0.3945** public prediction score.

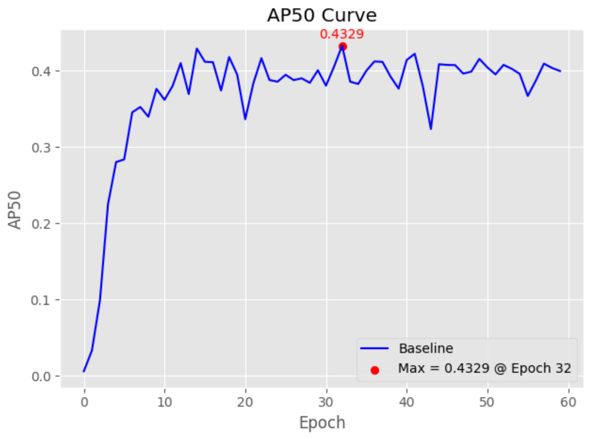
2. Mask R-CNN (baseline) with data augmentation



(a) Loss Curve



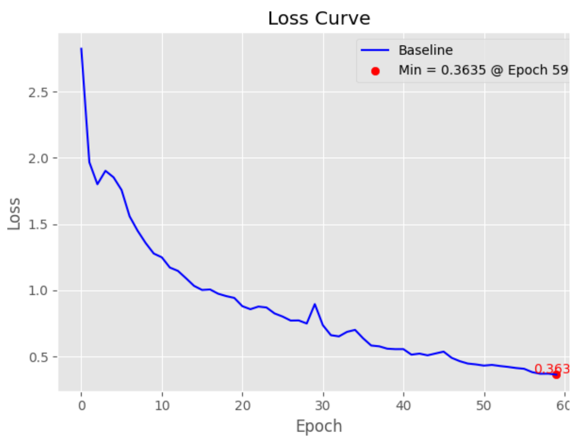
(b) mAP Curve



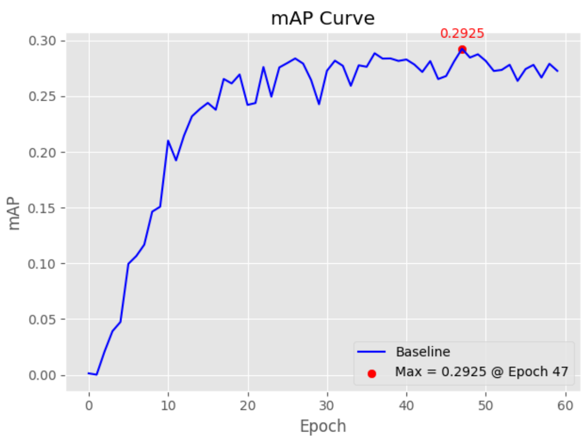
(c) AP50 Curve

As a result of the second experiment, a baseline model with data augmentation was trained. The loss curve shows that the model converges slowly; after 60 epochs, the final loss reached **0.3725**, the lowest observed loss in this setting, but it remains higher than that of the baseline model. In the validation results, the highest mAP achieved was **0.2863**, and the highest AP50 was **0.4329**, slightly lower than those obtained by the baseline model. Interestingly, despite achieving somewhat lower performance on the validation set than the baseline, I submit this result to Codabench, which achieves a **0.4499** public prediction score. The model trained with data augmentation performed better on the test dataset than the baseline model. Thus, I thought that using augmentation may have improved the model' s generalization ability to unseen data. It may have helped the model become more robust to variations in the test set, even though this effect was not reflected in the validation metrics.

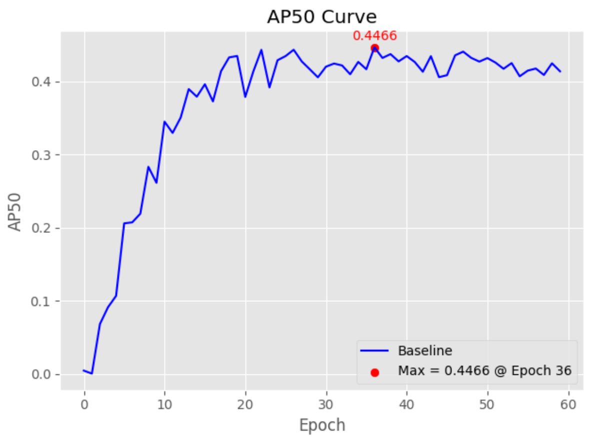
3. Mask R-CNN + CBAM in All Backbone Stages (except layer 1)



(a) Loss Curve



(b) mAP Curve



(c) AP50 Curve

In the third experiment, the CBAM module was inserted after every block of layers 2—4 in the ResNet50-FPN backbone. The loss curve shows that the model also converges slowly in the first two training settings, and after training 60 epochs, the final loss is **0.3635**, which is the lowest loss in this setting but still higher than the baseline model. However, in the validation curve, we can see that the highest mAP is **0.2925** and AP50 is **0.4466**, both of which are higher than the first two settings. However, when handing in the result of this model in the test dataset, its AP50 is **0.3892**, falling slightly below the baseline. I guess that the extra three attention layers improved training and validation performance, but did not generalize as well since our training dataset is so small, which may cause the model to overfit the training distribution.

Thus, in the codabench, I selected the result that uses Mask R-CNN (baseline) with data augmentation trained at the last checkpoint, which has the highest **0.4499** public prediction score.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[2] https://pytorch.org/vision/main/models/mask_rcnn.html.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[5] http://pytorch.org/vision/main/transforms.html.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[7] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[8] https://huggingface.co/docs/transformers/main_classes/optimizer_schedules.