Visual Recognition using Deep Learning HW3 Report

110550093 蔡師睿

1 Introduction

This task focuses on instance segmentation, a problem that requires both object detection and pixel-level delineation of individual object instances within an image. Unlike semantic segmentation, instance segmentation distinguishes between multiple objects of the same class by assigning unique labels and precise boundaries to each instance. The dataset consists of colored medical images, including 209 images for training and 101 images for testing. The challenge imposes several constraints, such as the use of only the Mask R-CNN[1] model with a size limit of under 200 million parameters, and no reliance on external data. The implementation can be found at *https://github.com/Ray-1026/Visual-Recognition/tree/main/HW3*.

2 Method

2.1 Mask R-CNN

Mask R-CNN is an extension of Faster R-CNN[2] that adds a parallel branch for pixel-level segmentation. While Faster R-CNN performs object detection by predicting bounding boxes and class labels, Mask R-CNN enhances this framework by additionally predicting a binary segmentation mask for each detected object, enabling instance-level segmentation.

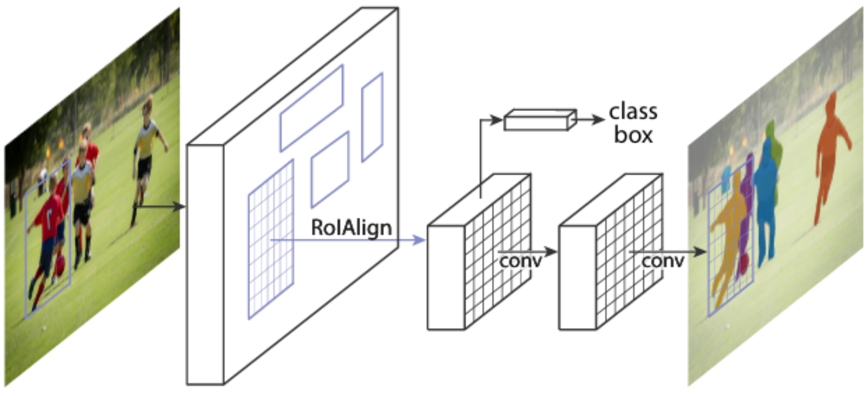


Figure 1: Mask R-CNN framework for instance segmentation.

A key improvement in Mask R-CNN is the introduction of RoI Align. RoI Pooling[3][4] operations may introduce spatial misalignments due to quantization, which can degrade mask quality. In contrast, RoI Align addresses this issue by eliminating quantization and using bilinear interpolation, thereby preserving precise spatial locations for more accurate feature extraction.

2.2 Modification

To efficiently build the Mask R-CNN model, I leveraged the torchvision library[5], which provides both the architecture implementation and pretrained weights.

Backbone. To improve feature representation, the default backbone in Mask R-CNN is replaced with a ConvNeXt-based[6] Feature Pyramid Network[7] architecture. ConvNeXt integrates design principles from both ResNet[8] and Vision Transformers[9], and offers enhanced performance while maintaining the efficiency of convolutional models. This modification aims to improve the model's ability to capture fine-grained details and generalize better on challenging visual patterns.

Data augmentation. Due to the limited size of the training dataset which contains only 209 images, the model is prone to overfit. To address this, data augmentation techniques, as shown in Algorithm 1, are employed to improve generalization.

Algorithm 1 Data Augmentation

```
$train\_tfm = T.Compose([$
    $RandomHorizontalFlip(p = 0.5),$
    $RandomVerticalFlip(p = 0.5),$
    $T.ToTensor(),$
$])$
```

Since the training inputs include both bounding boxes and masks, it is essential to ensure that all transformations applied to the images are consistently applied to the corresponding annotations. As illustrated in Code 1, the horizontal and vertical flip functions are customized to ensure that the bounding boxes and masks correctly follow the transformations applied to the images.

Code 1: Customized transform function

```python
from torchvision.transforms import v2 as T
from torchvision.transforms import functional as F

class RandomHorizontalFlip(T.RandomHorizontalFlip):
    def forward(self, image, target):
        if torch.rand(1) < self.p:
            image = F.hflip(image)
            if target is not None:
                _, _, width = F.get_dimensions(image)
                target["boxes"][:, [0, 2]] = width - target["boxes"][:, [2, 0]]
                target["masks"] = F.hflip(target["masks"])

        return image, target


class RandomVerticalFlip(T.RandomVerticalFlip):
    def forward(self, image, target):
        if torch.rand(1) < self.p:
            image = F.vflip(image)
            if target is not None:
                _, height, _ = F.get_dimensions(image)
                target["boxes"][:, [1, 3]] = height - target["boxes"][:, [3, 1]]
                target["masks"] = F.vflip(target["masks"])

        return image, target
```

Box score threshold. As in homework 2, the box score threshold is adjusted during inference based on the confidence scores of the predictions. Based on the public scores, I found that the threshold of 0.5 results in better mAP.

Train validation split. The dataset is only divided into training and testing sets, and thus I split 10% of the training dataset as a validation set to monitor the training process. After finalizing all hyperparameters, the validation set is merged back into the training set, and the model is retrained using the full training data.

2.3 Hyperparameters

- random seed: 42
- epochs: 20
- batch size: 4
- AdamW:
  - learning rate: $1e{-}4$
  - weight_decay: $1e{-}2$
- Cosine Annealing Schedule:

  - T_max: 20
  - eta_min: $1e-6$
- box score threshold: 0.5

3 Results

3.1 Training Curve

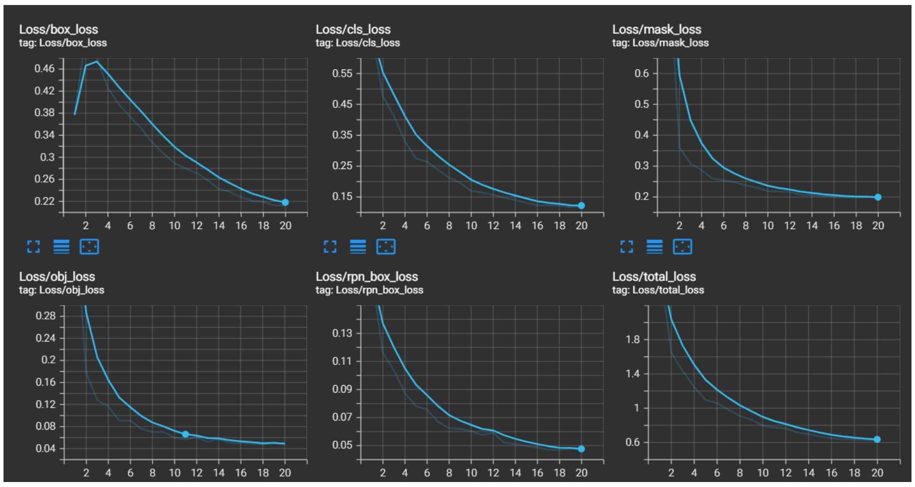
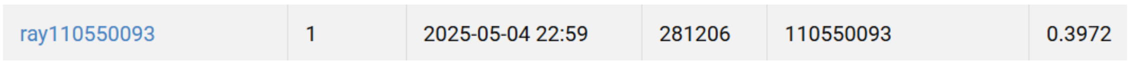


Figure 2: Training losses.

3.2 Public Score



ray110550093	1	2025-05-04 22:59	281206	110550093	0.3972

Figure 3: Public score on Codabench

4 Additional Experiments - Ablation Studies

4.1 Different backbones

The EfficientNetV2[10] and ConvNeXt models I used are from torchvision[5] and were trained on ImageNet[11]. The ResNet-50[8] model used is provided by torchvision as maskrcnn_resnet50_fpn, which is pretrained on the COCO dataset[12].

Backbone	public mAP
ResNet-50	0.33
EfficientNetV2_m	0.36
ConvNeXt_base	**0.40**

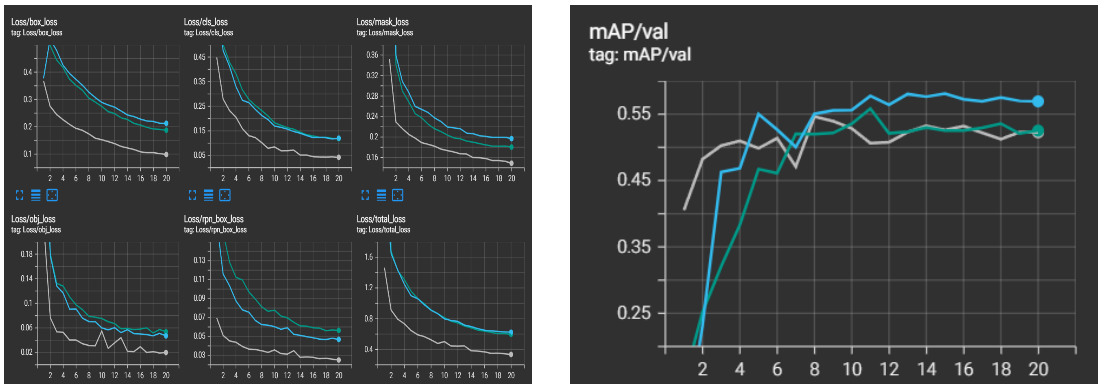


Figure 4: Training losses (left) and validation mAP (right). **Blue cureve**: convnext; **green curve**: efficientnetv2; **gray curve**: resnet50.

4.2 Trainable layers

I experiment with different numbers of trainable layers in the backbone. Freezing early layers helps retain pretrained low-level features and reduces overfitting, especially when training on a small dataset. However, I found that fine-tuning more layers results in better performance and lower losses, as shown in Fig. 5.

Trainable layers	public mAP
3	0.39
5	**0.40**

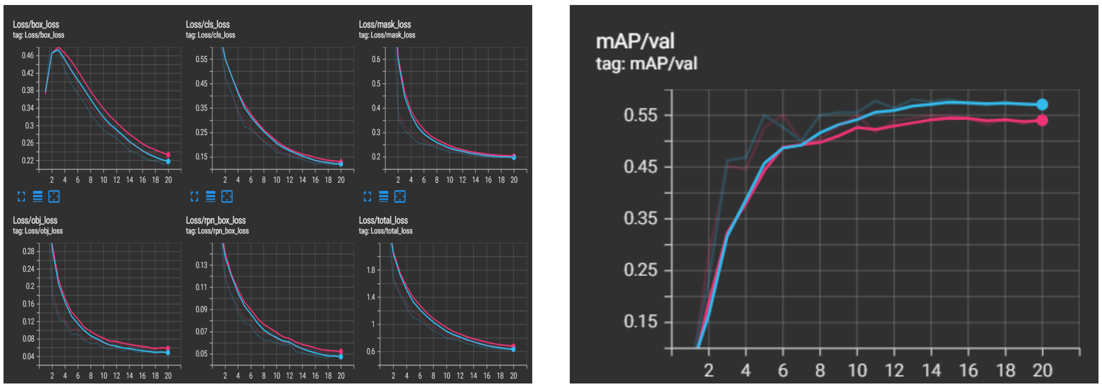


Figure 5: Training losses (left) and validation mAP (right). **Blue cureve**: 5 layers; **red curve**: 3 layers.

4.3 Adapt min-max size

This experiment adapts min size to 400 and max size to 1000 which affects how input images are resized before being passed to the network.

Adapt minmax size	public mAP
V	0.36
X	**0.40**

4.4 Different box score threshold

Box score threshold	public mAP
0.7	0.37
0.5	**0.40**

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[4] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[5] T. maintainers and contributors, *Torchvision: Pytorch's computer vision library*, `https://github.com/pytorch/vision`, 2016.

[6] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 976–11 986.

[7] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[10] M. Tan and Q. Le, "Efficientnetv2: Smaller models and faster training," in *International conference on machine learning*, PMLR, 2021, pp. 10 096–10 106.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[12] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, Springer, 2014, pp. 740–755.