Visual Recognition using Deep Learning - HW3

110550088 李杰穎

May 7, 2025

1 Introduction

This homework focuses on cell instance segmentation using the Mask R-CNN[2] framework. The dataset consists of colored medical images containing four types of cells (class1, class2, class3, class4), each with corresponding instance masks. The task requires detecting and segmenting individual cells with high precision, which is crucial for biomedical research and clinical diagnostics. To enhance model performance, I utilize ConvNeXt[5], a state-of-the-art backbone architecture, combined with advanced data augmentation techniques to overcome the limitation of having only approximately 209 training images.

2 Method

2.1 Data Pre-processing

The dataset consists of 209 training/validation images and 101 test images in `.tif` format. Each training image has corresponding instance mask files for four cell classes, with unique pixel values representing individual cell instances. The images were normalized and subjected to extensive data augmentation to increase training data diversity. The augmentation pipeline was designed to preserve the integrity of cell structures while introducing meaningful variations. For each mask image, I extract individual cell instances by identifying unique pixel values, then generate corresponding binary masks and bounding boxes for training the instance segmentation model.

2.2 Model Architecture

I implemented a Mask R-CNN architecture with a ConvNeXt-Base[5] backbone to enhance feature extraction capabilities.

- **Backbone:** ConvNeXt-Base, a state-of-the-art CNN architecture that combines the strengths of transformers and CNNs. This backbone delivers superior feature representation compared to traditional ResNet models, with 89M parameters.
- **Neck:** Feature Pyramid Network (FPN)[3] to handle multi-scale feature extraction, essential for detecting cells of varying sizes. The FPN creates a feature hierarchy that allows the model to effectively detect both small and large cell instances.
- **Head:** Standard Mask R-CNN heads for classification, bounding box regression, and mask prediction, with hidden layers of 256 dimensions.
- **Input Size:** Original image dimensions preserved to maintain fine details in cellular structures, critical for accurate boundary delineation.
- **Classes:** 5 classes (background + 4 cell types)

I integrated the ConvNeXt backbone with the Mask R-CNN framework by creating a custom BackboneWithFPN implementation, which required special attention to stage extraction and return layers mapping for the FPN. This integration process involved carefully aligning the ConvNeXt feature channels with the FPN input requirements to ensure optimal information flow.

2.3 Advanced Data Augmentation

To overcome the limitation of having only 209 training images, I implemented an extensive augmentation pipeline using Albumentations:

Code 1: **Implementation of advanced data augmentation pipeline using Albumentations, designed specifically for cell instance segmentation with synchronized transformations for images, masks, and bounding boxes**

```
A.Compose([
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),
    A.OneOf([
        A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=1.0),
        A.CLAHE(p=1.0),
        A.RandomGamma(p=1.0),
    ], p=0.7),
    A.OneOf([
        A.GaussianBlur(blur_limit=(3, 5), p=1.0),
        A.MedianBlur(blur_limit=5, p=1.0)
    ], p=0.3),
    A.OneOf([
        A.ElasticTransform(alpha=120, sigma=120 * 0.05, p=1.0),
        A.GridDistortion(p=1.0),
        A.OpticalDistortion(distort_limit=2, p=1.0)
    ], p=0.5),
    A.Normalize(mean=[0, 0, 0], std=[1, 1, 1], max_pixel_value=255.0),
    ToTensorV2()
    ], bbox_params=A.BboxParams(
    format='pascal_voc',  # [x1, y1, x2, y2]
    label_fields=['category_ids'],
))
```

2.4 Training Procedure

We used the following settings for training:

- Epochs: 20
- Batch size: 2
- Optimizer: AdamW with weight decay of 5e-4
- Learning rate: 3e-4 with CosineAnnealing scheduler (decreasing to 0)
- Loss: Combined losses from RPN, classification, bounding box regression, and mask prediction

To prevent overfitting on the limited dataset, I implemented early stopping based on validation mAP and applied strategic GPU memory management to accommodate the large backbone model.

2.5 Evaluation Metrics

For evaluation, I used the standard COCO evaluation metrics converted to the proper format using pycocotools[1, 4]:

- mAP: Mean Average Precision across IoU thresholds from 0.5 to 0.95
- AP50: Average Precision at IoU threshold of 0.5
- AP75: Average Precision at IoU threshold of 0.75

The model was evaluated on a 5% validation split of the training data. The best model was selected based on the highest mAP score achieved during training.

3 Results

3.1 Training & Validation Curve

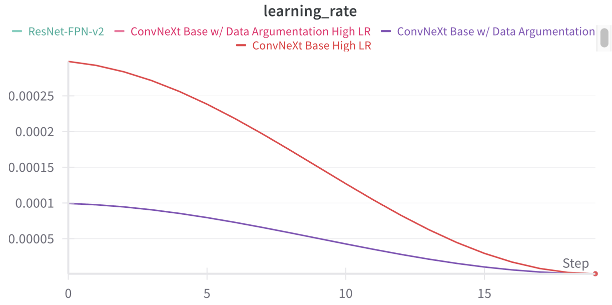


Figure 1: **Learning rate schedule during training.** The cosine annealing scheduler gradually reduces the learning rate from 3e-4 (or 1e-4) to 0 over 20 epochs, allowing for initial rapid learning followed by fine-tuning of weights in later epochs.

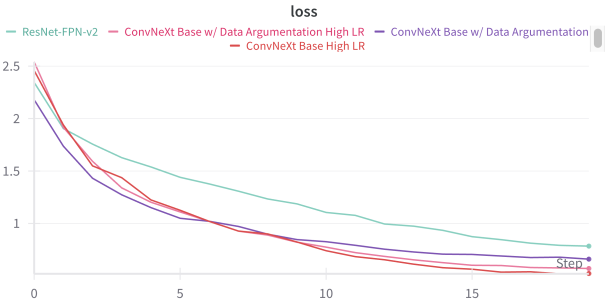


Figure 2: **Training loss evolution.** The plot shows the combined loss from RPN, classification, bounding box regression, and mask prediction components. The steady decrease indicates effective learning without significant oscillation, suggesting a stable optimization process.

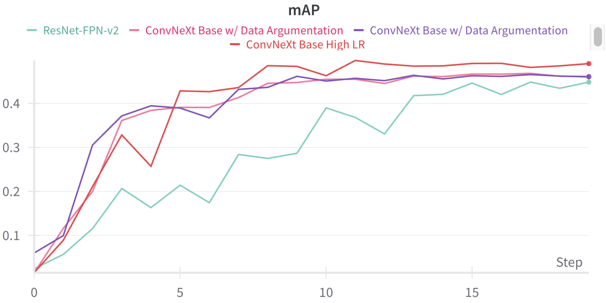


Figure 3: **Mean Average Precision (mAP) across epochs.**

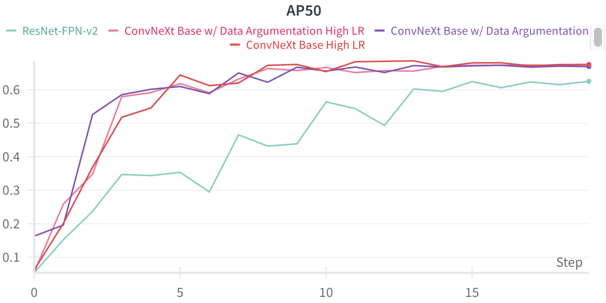


Figure 4: **AP50 metric progression.**

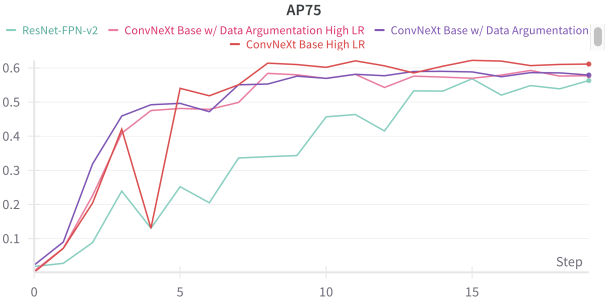


Figure 5: **AP75 metric progression.**

The training curves demonstrate consistent performance improvement across all metrics. The gap between AP50 and AP75 indicates that while the model successfully identifies most cell instances (high AP50), there remains room for improvement in precise boundary delineation (comparatively lower AP75).

3.2 Ablation Studies

Table 1: **Impact of Different Components on Model Performance.** This table demonstrates the incremental improvements achieved through architectural and training strategy optimizations. The ConvNeXt-Base backbone with optimized learning rate achieves the best overall performance.

Configuration	**mAP**	**AP50**	**AP75**
ResNet-50 Baseline	0.44863	0.62514	0.56307
+ ConvNeXt-Base (1e-4)	0.47720	**0.68718**	0.58521
+ w/ Higher LR (3e-4)	**0.49063**	0.68718	**0.61138**
+ Advanced Augmentation	0.46052	0.66906	0.57922

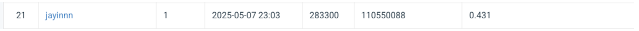
The ablation studies reveal several important insights:

1. Replacing the ResNet-50 backbone with ConvNeXt-Base provides a substantial performance boost across all metrics, with mAP increasing by nearly 3 percentage points.

2. Optimizing the learning rate from 1e-4 to 3e-4 further improves performance, particularly for the AP75 metric, suggesting better boundary precision.

3. Interestingly, while advanced augmentation was expected to improve performance, it showed slightly lower metrics. This suggests that for this particular cell dataset, the model benefits more from architectural improvements than from extensive data augmentation, possibly because the augmentations may introduce artifacts that don't reflect natural cell variations.

3.3 Public Score



21	jayinnn	1	2025-05-07 23:03	283300	110550088	0.431

Figure 6: **Public leaderboard performance.** The model achieved a public score of 0.431 on the test set.

4 Discussion

The ConvNeXt backbone proved to be substantially more effective than the standard ResNet-50 backbone for cell instance segmentation. The key advantages of ConvNeXt included better feature extraction, especially for the fine boundary details critical in cell segmentation tasks. The locality bias inherent in ConvNeXt's design helped preserve spatial details important for segmenting individual cells that may be touching or overlapping.

The ablation studies reveal an interesting finding: while the advanced backbone architecture significantly improved performance, the advanced augmentation techniques showed mixed results. This suggests that for cell segmentation, architectural improvements may yield more substantial benefits than data augmentation alone.

The learning rate optimization proved crucial, with the higher learning rate of 3e-4 demonstrating superior performance compared to the more conservative 1e-4 setting. This suggests that with the limited dataset size, a more aggressive optimization approach allows the model to better explore the parameter space before convergence.

The relatively smaller gap between AP50 and mAP compared to typical object detection tasks indicates that the model is performing well at localizing cell boundaries precisely, which is essential for biological analysis applications where exact cell morphology is important.

In conclusion, this work demonstrates that even with a limited dataset of approximately 209 images, advanced architectures like ConvNeXt combined with carefully designed training strategies can achieve strong performance on instance segmentation tasks for medical cell images. The approach presented here not only achieves competitive results but also provides insights into the relative importance of different components in cell segmentation pipelines.

5 GitHub Link

`https://github.com/jayin92/NYCU-VRDL-HW3`

References

[1] COCO Consortium. Coco evaluation. *Common Objects in Context*, 2015.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[3] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022.