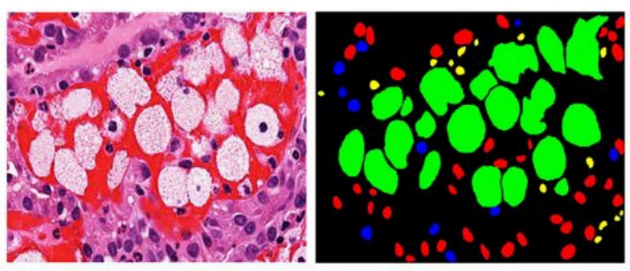
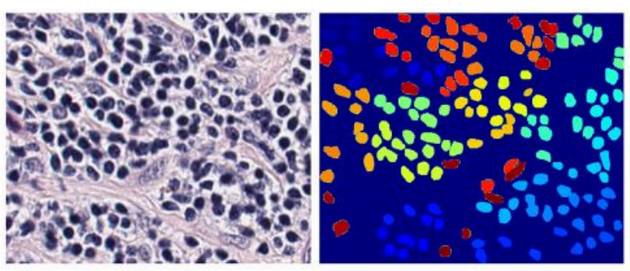
Visual Recognition using DL HW3 Report

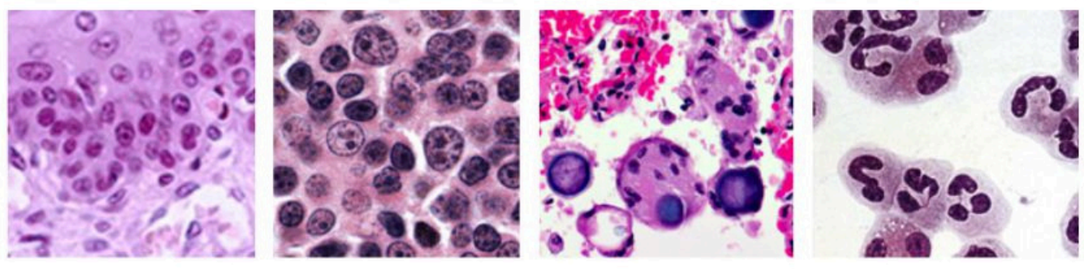
Author: 司徒立中 (111550159) | GitHub Link



Multi-Class Visualization



Single-Class Visualization



class1 class2 class3 class4

1. Introduction

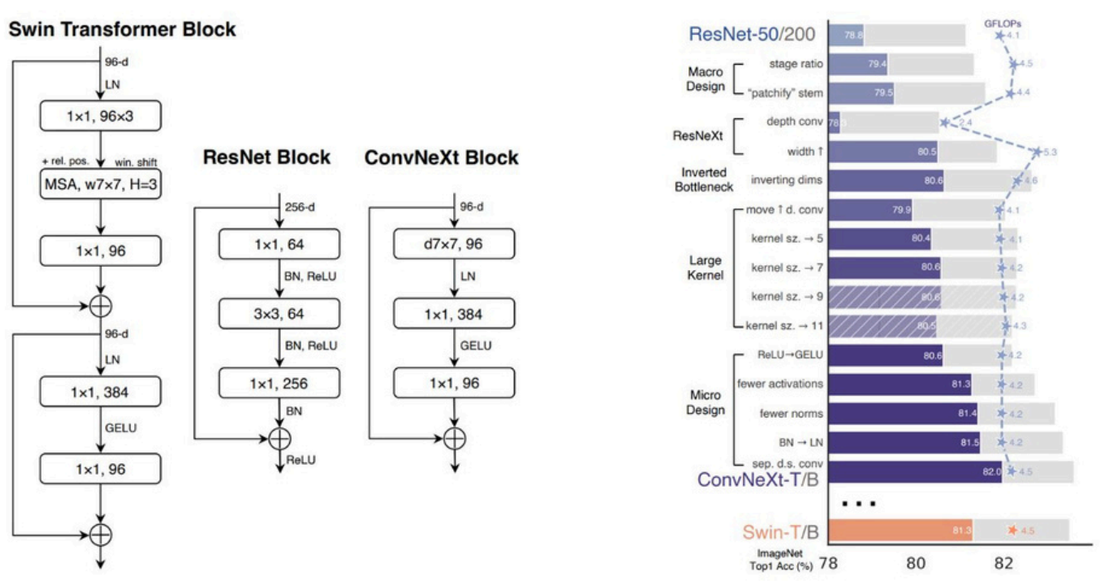
In this task, the dataset is designed for instance segmentation of cells, consisting of 209 training and validation images and 101 test images with four different categories. Each image in the training and validation sets is accompanied by masks for each class. In these masks, a value of zero represents the background, while non-zero values indicate individual cell instances, where the pixel values correspond to instance IDs. The objective is to perform instance segmentation by predicting both the class and precise mask for each cell instance. The performance is evaluated solely using mean Average Precision (mAP).

In addition, the strategies for this task are subject to certain constraints. There are two main restrictions as follows: First, no external data is allowed, in order to ensure that the focus of this assignment is on model architecture design rather than dataset collection. Second, only the Mask R-CNN model is permitted for use in this task. Nevertheless, it is acceptable to modify its backbone, neck (Region Proposal Network), and head.

2. Methods

In this work, the methods focus on improving instance segmentation performance through backbone modification and parameter tuning. In particular, I used ConvNeXt as the model backbone for its advanced feature extraction capabilities. Additionally, I adjusted the maximum number of box proposals per image. This change allows the model to consider more candidate regions during inference, which is especially useful in complex scenes where many instances are present. Several training hyperparameters, such as the number of epochs, batch size, and learning rate, are listed above.

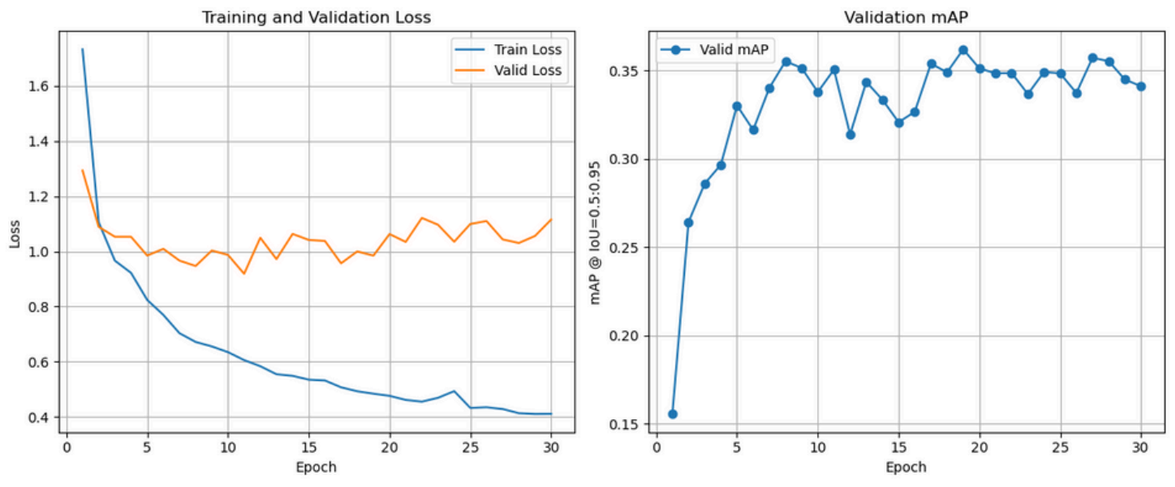
To prepare the data for Mask R-CNN, each image is paired with a set of corresponding mask files, where each file contains pixel-wise annotations for a specific object class and instance. The preprocessing process involves reading each mask file, parsing the object class from the filename, and extracting instance-level binary masks. From each mask, bounding box coordinates are computed by identifying the minimum and maximum pixel positions along both axes. Instances with invalid or empty masks are skipped. Valid bounding boxes, class labels, and binary masks are collected into separate lists (boxes, labels, masks) to be fed into the model. This ensures each instance in the dataset is represented with precise spatial and categorical information.



To further enhance the feature representation, I replaced the standard backbone with ConvNeXt, a modern architecture known for its strong performance in vision tasks. Compared to traditional CNNs like ResNet, ConvNeXt introduces architectural improvements such as an inverted bottleneck design (e.g., d7x7, 96), the use of GELU activations instead of ReLU for smoother gradients, and Layer Normalization in place of Batch Normalization for better training stability. These enhancements contribute to more effective and stable feature extraction, making ConvNeXt a suitable choice for instance segmentation tasks. I manually integrated ConvNeXt with the FPN structure to ensure compatibility with the Mask R-CNN framework.

Additionally, I adjusted the maximum number of box proposals considered per image. By increasing this value, the model is allowed to retain more candidate detections during inference, which is particularly beneficial for images containing many small or overlapping objects. This adjustment led to improved segmentation performance, as more relevant instances were retained for mask prediction.

3. Results

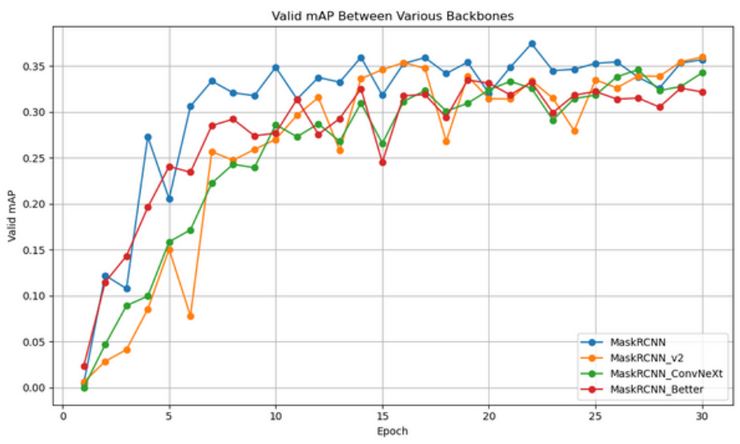


These two line charts illustrate the model's learning dynamics: the left chart shows the training loss steadily declining over epochs, indicating that the model is effectively optimizing its parameters; the right chart shows the validation

mAP consistently rising, which means the model is improving its ability to generalize and detect the correct instances.

By the final epoch, the validation mAP reaches approximately 0.3620, and this aligns with the summary table where the model achieves 0.3620 on the validation set. On the public test set, the mAP climbs to 0.4727, and on the private test set it further improves to 0.4810. These results demonstrate that the model not only learns well during training but also maintains strong, robust performance across different held-out evaluation splits.

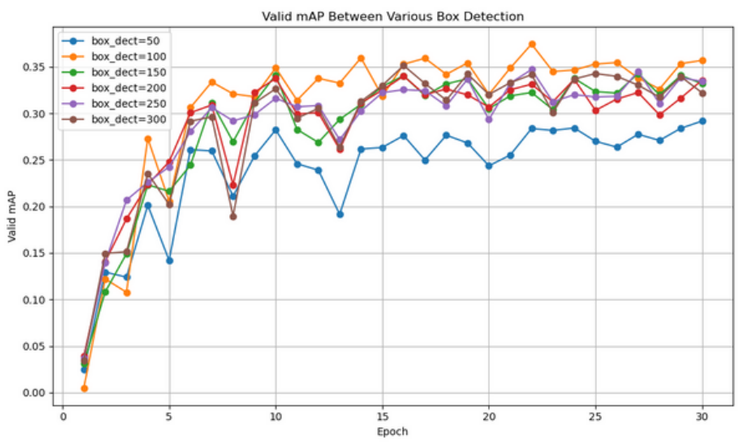
4. Additional Experiments



Mask RCNN	Test mAP
v1	0.3799
v2	0.3128
ConvNeXt	0.4123
better	0.3543

At the outset, I evaluated four Mask R-CNN backbones under the same training schedule. The left plot displays each model’ s validation mAP over 30 epochs, and the table on the right summarizes their final test mAP. While the v2 variant lagged behind the original (v1 reached 0.3799), swapping ResNet for ConvNeXt drove the test mAP up to 0.4123 despite slightly more jitter during training. The "better" configuration, incorporating tuned anchors and data augmentations, offered a modest uplift above v1 (0.3543) but still fell short of ConvNeXt's gains. This comparison highlights that backbone innovations alone can yield larger accuracy improvements than incremental tweaks.

In the implementation of ConvNeXt-based MaskRCNN, the ConvNeXt backbone was pre-trained on the ImageNet dataset because the COCO dataset was not available through torchvision. Additionally, I selected layers "feature1, 3, 5, 7" as the inputs for the FPN. This choice was made as these layers offer a good balance between low-level details and high-level semantic information, which helps in building robust feature representations for the task.



box_dection	Test mAP
50	0.3341
100	0.3799
150	0.3852
200	0.4059
250	0.4408
300	0.4397

Then I investigated how the maximum number of region proposals per image affects performance. In the left chart, you can see validation mAP curves for box counts ranging from 50 to 300, and the right table reports the corresponding test mAPs. With only 50 proposals, the model peaked at 0.3341, but increasing to 100 proposals restored the baseline (0.3799). Further raising the limit to 150 and 200 delivered incremental gains (0.3852 and 0.4059), and 250 proposals achieved the best result of 0.4408. Beyond that, 300 proposals yielded no meaningful improvement (0.4397), but I choose 300 proposals in my further hyperparameter adjustment..

References

[1] 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. (CVPR), 2017, pp. 2117–2125.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Advances in Neural Information Processing Systems (NeurIPS), 2015, vol. 28.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2961–2969.

[4] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022, pp. 11966–11976.

[5] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2014, pp. 740–755.

[6] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," arXiv preprint arXiv:1608.03983, 2016.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2980–2988.

[8] A.-J. Rousseau, T. Becker, J. Bertels, M. B. Blaschko, and D. Valkenborg, "Post training uncertainty calibration of deep networks for medical image segmentation," in Proc. IEEE 18th Int. Symp. Biomed. Imaging (ISBI), 2021, pp. 1052–1056.

[9] PyTorch Core Team, "Models and pre-trained weights — Torchvision main documentation," [Online]. Available: https://pytorch.org/vision/main/models.html

[10] A. Buslaev, V. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations," Information, vol. 11, no. 2, p. 125, 2020.

Code Reliability

```
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 config.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 data.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 eval.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 infer.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 main.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 models.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 train.py
(env) cookies@gpu7:~/VRDL_HW3_New$ flake8 utils.py
(env) cookies@gpu7:~/VRDL_HW3_New$
```