NYCU Visual Recognition using Deep Learning
Homework 3

[312551185], [郭晏涵]

GitHub Link:

https://github.com/slovengel/312551185_HW3

Introduction:

This project aims to tackle an instance segmentation problem in medical images using Mask R-CNN. The dataset consists of RGB images, with 209 samples for training/validation, and 101 samples for testing. The objective is to detect and segment individual cell instances and correctly classify each into one of the predefined cell types.

To enhance segmentation performance, I employed several key strategies including transfer learning, adaptive learning rate scheduling, and K-fold cross-validation. The core of my method is built upon a Mask R-CNN framework with a ResNet-50-FPN backbone, fine-tuned for cell instance segmentation task.

The proposed model achieved 0.40 mAP in the CodaBench competition (on public leaderboard), demonstrating its effectiveness in handling the instance segmentation task.

Method:

1. **Data pre-processing**

   The entire dataset images are converted to tensors. Unlike traditional classification pipelines, the images are not resized to a fixed size, as instance segmentation requires maintaining the original spatial structure and object scale.

   Initially, I experimented with common augmentation techniques like RandomHorizontalFlip, RandomRotation and ColorJitter to improve generalization and reduce overfitting. However, these transformations led to a drop in both training and validation mAP. This is likely because such transformations distorted the morphology and spatial coherence of individual cell instances, making it harder for the model to learn accurate masks and boundaries. As a result, these augmentations were excluded from the final training pipeline.

   Additionally, to enable COCO-style mAP evaluation, I generated a custom ground truth JSON file following the COCO format. This file includes image metadata, object annotations, and segmentation masks for each cell instance, and is used during training and evaluation to compute segmentation performance metrics.

2. **Model architecture**

   The model is built upon the Mask R-CNN architecture, which extends Faster R-CNN by adding a third branch for pixel-level segmentation. It consists of three main components: a backbone, a neck (RPN), and a multi-task prediction head.

- Backbone: We use a ResNet-50 with Feature Pyramid Network (FPN) as the feature extractor. The ResNet-50 captures hierarchical spatial features, while the FPN improves performance on small and variably sized objects—critical for segmenting densely packed or small cellular structures.
- Neck (RPN): The Region Proposal Network operates on the multi-scale FPN feature maps to generate ROIs. It predicts anchor boxes likely to contain individual cells, scoring them based on objectness and refining their coordinates.
- Head: The detection head consists of two branches:
  - A classification branch that predicts the cell type.
  - A regression branch that refines the bounding box coordinates.
  - A mask prediction branch that generates a binary segmentation mask for each detected instance.

The default prediction head is replaced with a new one customized for class classification by setting the number of output classes to 5 (4 cell classes + background).

3. Hyperparameters and Training Details

The models were trained using a batch size of 4 for 20 epochs. The **AdamW** optimizer was used with a **learning rate** of **1e-4** and a weight decay of 1e-4 to prevent overfitting. Only parameters requiring gradients were updated during training.

CosineAnnealingLR learning rate scheduler was applied to dynamically adjust the learning rate over time. It creates a smooth and continuous decay of the learning rate without interruptions. The learning rate starts at **1e-4** in the first epoch and gradually decreases to **1e-6** by the final epoch.

To support mixed-precision training and improve computational efficiency, automatic mixed precision (AMP) was enabled using torch.amp.GradScaler().

Additional experiments:

1. Backbone Comparison

This experiment investigates the impact of different backbones on the performance of the Mask R-CNN framework for instance segmentation. Specifically, we compare ResNet-50-FPN and ResNeXt-50-32x4d-FPN as feature extractors within the same Mask R-CNN framework.

The detailed implementation is as follows:

1. Train two Mask R-CNN models independently using the same training pipeline, differing only in the backbone architecture.

2. The first model uses a ResNet-50 with FPN derived from the Benchmarking Detection Transfer Learning with Vision Transformers [1] paper. This is a widely adopted backbone known for strong multi-scale feature representation.
3. The second model replaces the standard ResNet with a ResNeXt-50-32x4d derived from the Aggregated Residual Transformation for Deep Neural Networks [2] paper. This is a more modern architecture that incorporates grouped convolutions for improved representational power at similar computational cost.

The following are the experiment results:

Model Name	**Best mAP**
ResNet-50-FPN	0.40
ResNeXt-50-32x4d-FPN	0.37

Analysis:

The ResNet-50-FPN outperformed the ResNeXt-50-32x4d-FPN in this instance segmentation task. The ResNet-50's consistent performance can be attributed to its well-established synergy with the Feature Pyramid Network, enabling robust detection and segmentation of small and overlapping cell instances. Its residual connections help preserve gradient flow during training, which is particularly beneficial for deeper networks.

The ResNeXt-50-32x4d, though more expressive in theory due to its grouped convolution design, slightly underperformed in this context. One possible explanation is that the grouped structure may not align as effectively with the FPN in capturing fine-grained spatial details required for precise instance masks in dense medical images. Additionally, tuning grouped architectures may require more task-specific adaptation to reach full potential.

2. Validation Strategy Comparison

This experiment compares two validation strategies for training a Mask R-CNN model on a medical instance segmentation task: K-Fold Cross-Validation and training with no held-out validation set. The objective is to evaluate the trade-off between robust model selection and full data utilization.

Implementation differences:

- K-Fold Cross-Validation partitions the training data into 5 folds. In each iteration, 4 folds are used for training and 1 for validation, enabling performance estimation across multiple data splits. The final model is selected based on the best average validation mAP.

- In contrast, the no validation strategy uses the entire training dataset without a separate validation set, maximizing data exposure but at the risk of overfitting and lacking generalization feedback.

The following are the experiment results:

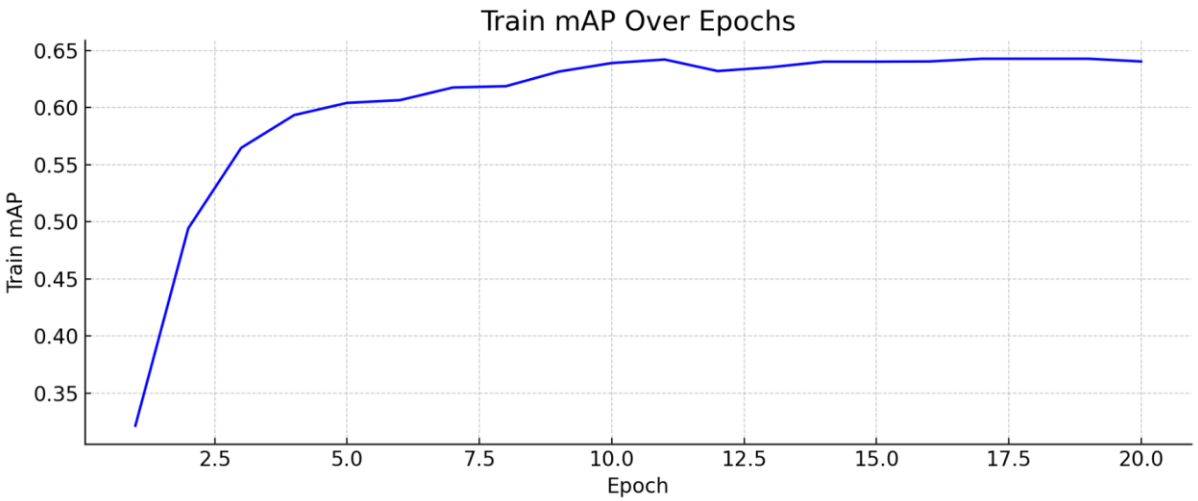
Model Name	**Best mAP**
No Validation Set	0.40
K-Fold Cross-Validation	0.38

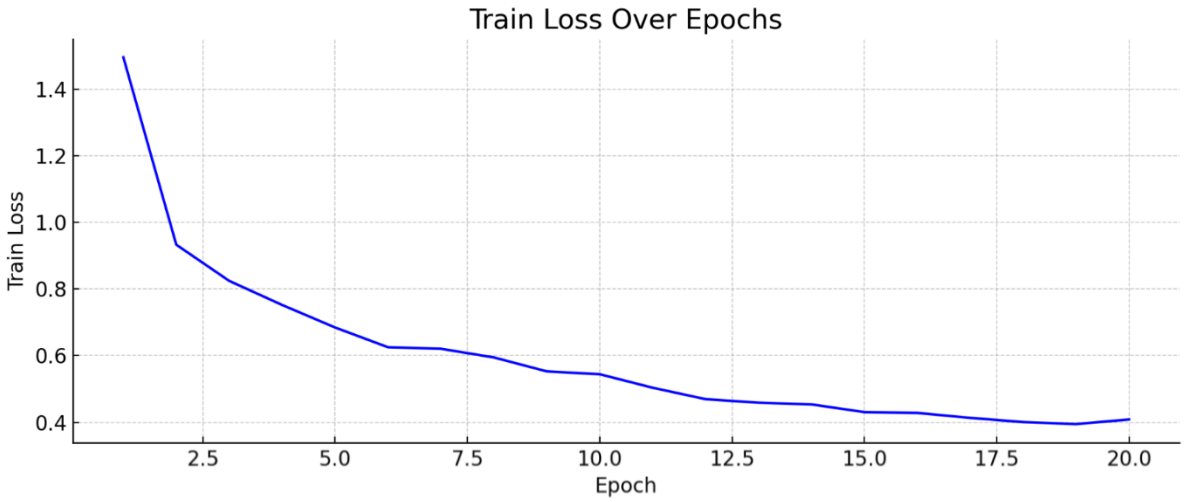
Analysis:

While K-Fold Cross-Validation offers a more robust evaluation by averaging performance across data splits, it can slightly underperform in terms of final mAP because each fold is trained on less data. In contrast, training on the full dataset without a validation set allows the model to learn from all available samples, potentially improving segmentation accuracy. However, this comes at the cost of not being able to monitor generalization or perform model selection. In this experiment, the full-data approach achieved slightly higher mAP, but its reliability depends heavily on the quality of the test set for true performance evaluation.

Results:

Based on the experimental results, neither ResNeXt-50 nor K-Fold Cross-Validation provided significant improvements. Since the combination of the Mask R-CNN with ResNet-50-FPN backbone and training on the full dataset without a validation set achieved the best performance, it was selected as the final model for inference. The following are the learning curves for training the proposed model.





Early stopping was applied at epoch 10 to mitigate potential overfitting, as the training loss continued to decrease, but the training mAP showed signs of plateauing. This strategy helps preserve the model's generalization ability and avoids fitting noise in the training data.

Code Reliability:

I lint the code using Flake8 to follow the PEP8 instructions.

References:

[1] Benchmarking Detection Transfer Learning with Vision Transformers https://arxiv.org/abs/2111.11429

[1] Aggregated Residual Transformation for Deep Neural Networks https://arxiv.org/abs/1611.05431