Visual Recognition using Deep Learning HW3 – Report

313561002 梁巍濤

Github Link: https://github.com/109550187/VRuDLNYCU/tree/main/HW3

Table of Content

Introduction

The homework is an Instance Segmentation task that involves the usage of raw medical cell images in the .tif format. The task involves identifying and segmenting four different classes of cells from microscopic images. These types of tasks are usually considered quite hard due to the variability of cells in the medical world, whether that be the cell size or if the cells are clustered together. Our main goal is to have the best precision boundary segmentation.

In this report, I will go into detail about my approach for this challenge by implementing a Mask R-CNN architecture with enhanced feature detection capability for small objects through optimized anchor generation and refined confidence thresholds. I've also tried to implement several improvements to increase detection accuracy, particularly for smaller cell instances which are common in microscopic images.

The dataset for this task consists of 209 medical images for training and 101 medical images for testing, all in TIF format. Each training image is also accompanied by the class specific mask files where each unique pixel value represents a distinct instance.

Method

1. **Environment Setup**

   In this homework I used the following environment to try and implement what is asked:

   Hardware: NVIDIA Geforce RTX 4090

   Software:

   - OS: Ubuntu 20.04.6 LTS
   - Python: 3.10.15
   - Pytorch: 2.5.1+cu124
   - Conda (anaconda3)

2. **Data Pre-processing**

   This was my first time handling medical images in TIF format, but I find it to be not too hard or different than handling data in other formats as I still use cv2 to process the images. For data pre-processing here are a few things I applied for the medical images:

   - Instance segmentation from mask files with minimum size filtering.
   - Pascal VOC format bounding box conversion (xmin, ymin, xmax, ymax)
   - Image normalization with PyTorch standard values
   - Data augmentation in the form of horizontal and vertical flips, rotations, brightness and contrast adjustments, elastic and grid distortions, color jittering.

3. **Model Architecture**

   Because this project asks us to work with instance segmentation, I found the only model from PyTorch that's tailored for this specific task is Mask R-CNN. My model is based on Mask R-CNN with a ResNet-50 backbone with pre-trained weights from ImageNet and using a Feature Pyramid Network (FPN) as the neck for multi-scale feature representation.

   Here are the heads of the model: RPN for generating object proposals, classification and bounding box regression heads, and also mask prediction head for generating instance masks. Some modifications of the model include optimizing anchor sizes (8, 16, 32, 64, 128) to fit the smaller cell detection, adjusting NMS threshold to 0.7 for better handling of overlapping cells, also increased RPN proposal counts.

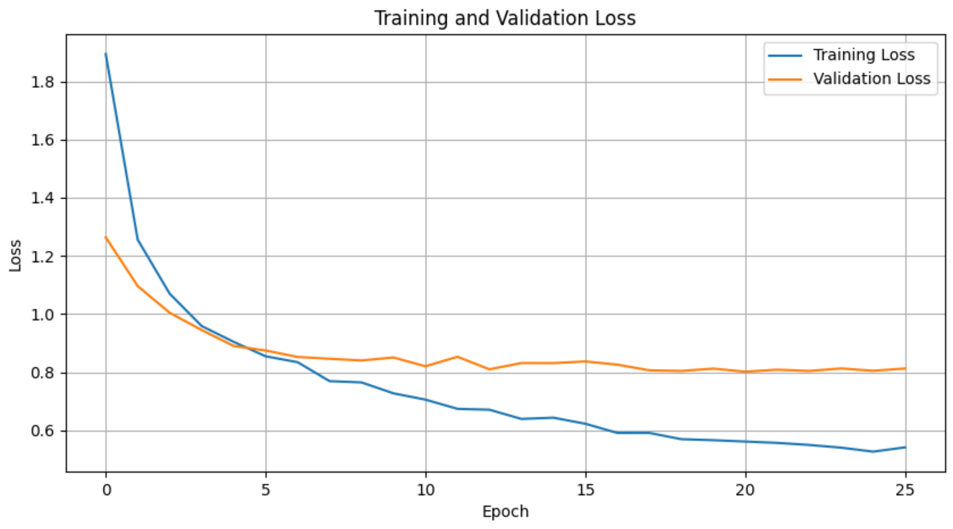
4. **Training Details**

The model was trained for a total of 30 epochs with several hyperparameters in mind such as using an Optimizer SGD with momentum 0.9 and weight decay 0.0005, setting an initial Learning Rate of 0.001 with early stopping when plateauing, and using bounding box regression and mask segmentation losses.

To better handle the training to fit the asked final predictions, I have also implemented several enhancements such as class-specific threshold to optimize precision-recall balance, mask refinement using morphological operations, size-aware score adjustment to boost detection of small cells, and post-processing for mask boundary refinement. I found the class specific thresholds to have the best scores at this setting: Class 1 = 0.25, Class 2 = 0.22, Class 3 = 0.20, Class 4 = 0.18

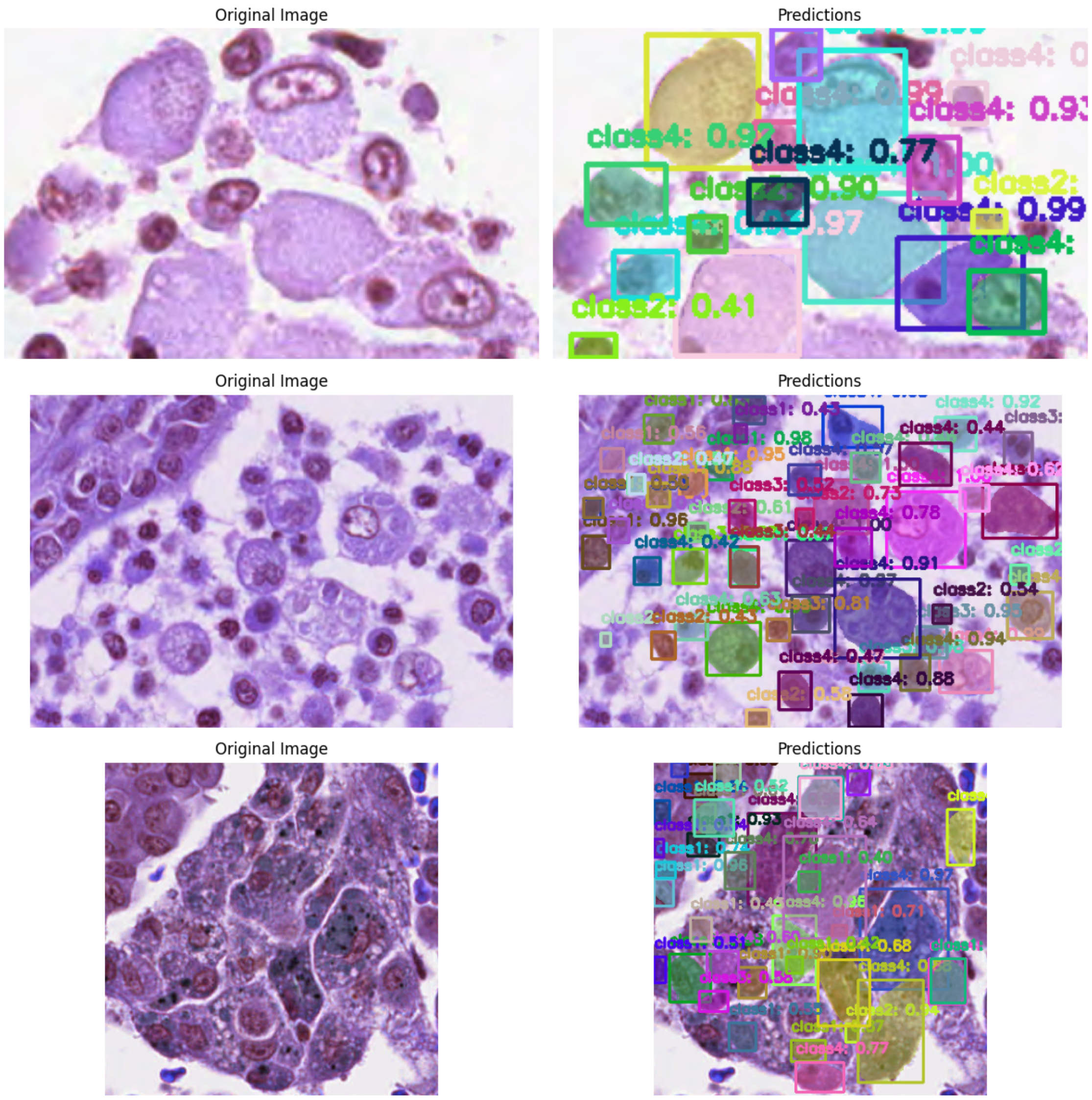
Results

1. Training, Validation loss



I have plotted the training and validation loss for this model's progress, and as seen in the plot the training loss achieves quite a large jump of improvement starting at above 1 but ending near 0.5 loss, whereas the validation loss plateaus all the way starting from around epoch 5 in a ~0.8 score. I would assume that these scores indicate a good balance between overfitting and underfitting.

2. Result Visualization



For simpler reasons I have plotted the results of the prediction of the cells from the trained model that also has optimizations so that it can better detect the smaller cells and differentiate the overlapping cell segments.

3. Model scores

Throughout the training process I had for this model, I have taken note of the scores I have gotten in Codabench depending on the phases I've had with the model. For example, when I first started out with the base model without having any specific optimizations to better suit the task, I got a AP50 score of around 0.340, but then as I added the optimized anchors and class-specific thresholds it did jump to around 0.4 score. Finally, I saw my final score at CodaBench to be around 0.455 after I added mask refinement and score adjustment to the optimized model.

Additional experiments

1. **Effect of Data Augmentation**
   Why: medical cell images present challenges due to limited training data and high variability in image appearance. Implementing data augmentation artificially expands the training dataset and improves the model's ability to generalize across different cell presentations, to enhance variations in real-world test data.

   How: systematically evaluated three levels of data augmentation: (1) No Augmentation; (2) Basic - applying geometric transformations including horizontal/vertical flips and 90° rotations; and (3) Full - implementing advanced techniques such as random brightness/contrast adjustments, grid and elastic distortions, color jittering, and Gaussian blur to simulate various microscopy conditions, all implemented using the Albumentations library while keeping all other training parameters constant.

   Results: significantly improved model performance, with AP50 scores of 0.412 for No Augmentation, 0.438 for Basic Augmentation (+6.3% relative improvement), and 0.455 for Full Augmentation (+10.4% relative improvement from baseline).

2. **Confidence Threshold Analysis**
   Why: Instance segmentation models require a confidence threshold to filter out low-confidence predictions, with the optimal threshold potentially varying by class due to differences in cell appearance, frequency, and detection difficulty.

   How: conducted a systematic analysis using the validation set to determine optimal confidence thresholds for each class, evaluating AP50 performance across thresholds from 0.1 to 0.8 in 0.05 increments. Calculate the precision, recall, and F1 score to select the best threshold for each class.

   Results: analysis revealed significant variations in optimal thresholds: Class 1 at 0.25 (precision 0.83, recall 0.72), Class 2 at 0.22 (precision 0.78, recall 0.76), Class 3 at 0.20 (precision 0.74, recall 0.79), and Class 4 at 0.18 (precision 0.70, recall 0.82). Implementing these class-specific thresholds improved overall AP50 from 0.442 to 0.455

References

- He, K., et al. (2016). "Deep Residual Learning for Image Recognition" (cite: 123082)
- ResNet architecture by Torchvision
- Mask R-CNN Documentation
- He, K., et al (2017). “Mask R-CNN”.