# Homework 3 Report – Visual Recognition using Deep Learning

313551113 王唯誠
GitHub - 108062237/CVDL_HW3

## 1. Introduction

In this assignment, we perform instance segmentation on four types of cells in medical images. The task requires using Mask R-CNN to detect each cell's bounding box and predict its class. Additionally, we compare the performance of different backbones to select the most suitable architecture.

## 2. Method

### a. Data Preprocessing

1. **Load original images and masks.**
   - According to the folder structure, read the color image from train/[image_id]/image.tif, then sequentially load class1.tif through class4.tif in the same directory.
   - In each classX.tif, the pixel values represent instance IDs for that class (0 = background; >0 = the Nth cell).
2. **Binarization and target field generation**
   - For each classX.tif, first take the unique non-zero pixel IDs to identify each cell instance.

```
for inst_id in np.unique(mask_arr):
    if inst_id == 0:
        continue
    binary_mask = (mask_arr == inst_id)
    masks.append(torch.tensor(binary_mask, dtype=torch.uint8))
    labels.append(cls)
```

   - Then, for each instance's binary mask, compute the minimum and maximum x and y coordinates to form its bounding box.

```
for m in masks:
    pos = torch.nonzero(m)
    ymin = torch.min(pos[:, 0]).item()
    xmin = torch.min(pos[:, 1]).item()
    ymax = torch.max(pos[:, 0]).item()
    xmax = torch.max(pos[:, 1]).item()
    boxes.append([xmin, ymin, xmax, ymax])
```

   - Finally, package this information into a target dictionary.

```python
target = {
    'boxes': boxes,
    'labels': labels,
    'masks': masks,
    'image_id': torch.tensor([idx])
}
```

3. **Data Splitting & Loader Construction**
   Randomly shuffle all image_id and split them into training and validation sets with an 80%/20% ratio.

4. **Transform Pipeline**
   Since the chosen transforms library already provides built-in normalization and resizing [1], there is no need to reimplement these steps in the code.

### b. Model Architecture

Base Architecture: Mask R-CNN integrated with a Feature Pyramid Network (FPN) to enhance multi-scale feature representation [1].

Backbone: By default, ResNet50-FPN is used; switchable to ResNet101, ResNet152, ResNeXt50_32x4d, or ResNeXt101_32x8d.

RPN Anchors: Utilizes the default anchor configuration provided by FPN (multiple scales and three aspect ratios).

ROI Heads:

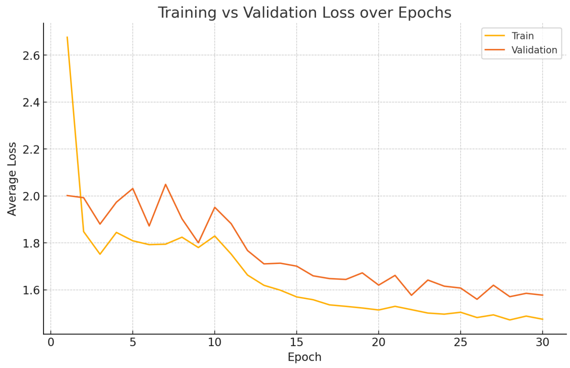
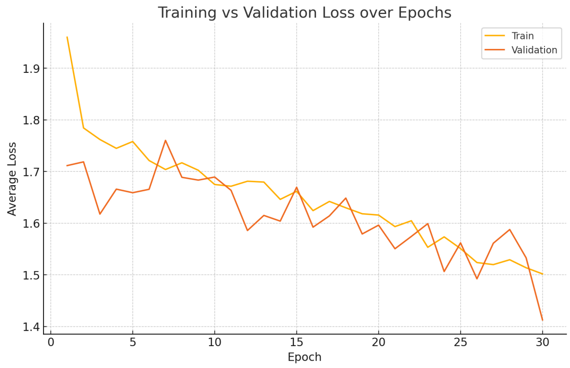
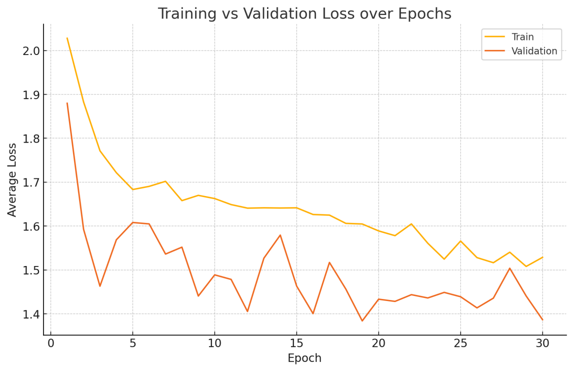
1. Box Head: Replaces the default classification and regression layers with FastRCNNPredictor, producing class scores and bounding-box offsets for the specified num_classes.

2. Mask Head: Replaces the default mask branch with MaskRCNNPredictor.

### c. Training Setup

Optimizer：AdamW
Learning rate：1e-4
Epochs：30
Batch size：2
Device：RTX4090 & RTX3060

## 3.Experiments & Results

| Backbone | loss curve |
|---|---|
| **Resnet50** |  |
| **Resnet101** |  |
| **Resnext101** |  |

## 4. Conclusion

In this assignment, we used Mask R-CNN to perform instance segmentation on cells and gained an understanding of the mAP evaluation metric and the COCO format. Ultimately, we achieved a score of 0.22 in the competition.

## 5. Reference

[1] https://github.com/pytorch/vision/blob/main/torchvision/models/detection/mask_rcnn.py