小组项目报告

2022 至 2023 学年第 二 学期

课 程 名 深度学习与大数据智能

学生学号 20204166、20204171、20204170、20204137、20204156

学生姓名 杨奎、张一锋、付文韬、徐小龙、何锐

任课教师 文静

报告得分

基于深度卷积网络的肝脏分割

摘要：

肝脏分割在医学图像处理中起着重要作用，然而其挑战在于肝脏与周围组织的复杂结构和多样性。本研究基于深度卷积网络，选取了 U-Net 作为基础网络架构，以提高肝脏分割的准确性和效率。U-Net 采用编码-解码结构，通过逐步降采样和上采样操作捕捉不同尺度的特征信息，并通过跳跃连接实现低级和高级特征的融合。为了进一步增强网络的性能，我们引入了 ResNet 的残差结构替代 U-Net 的编码模块，以提高特征表示能力和信息传递效率。实验结果显示，我们的方法在公开肝脏分割数据集上达到了优秀的分割性能，有效地解决了复杂肝脏结构的分割问题。该研究对于临床肝脏疾病的诊断和治疗具有重要的应用价值。

关键词： 深度卷积网络，肝脏分割，UNet，医学图像处理

小组分工：

成员	工作	工作量
杨奎	编码，记录并整理项目报告，整理 PPT，汇报	50%
张一锋	制作网络结构部分 PPT	10%
付文韬	制作训练组件部分 PPT	10%
徐小龙	制作数据集处理部分 PPT	20%
何锐	制作剩余部分 PPT	10%

1 项目简介

1.1 基本框架介绍

1.1.1 pyTorch

PyTorch 是一个开源的机器学习框架，它基于 Python 语言，并提供了丰富的工具和库来支持构建、训练和部署深度学习模型

1.1.2 monai

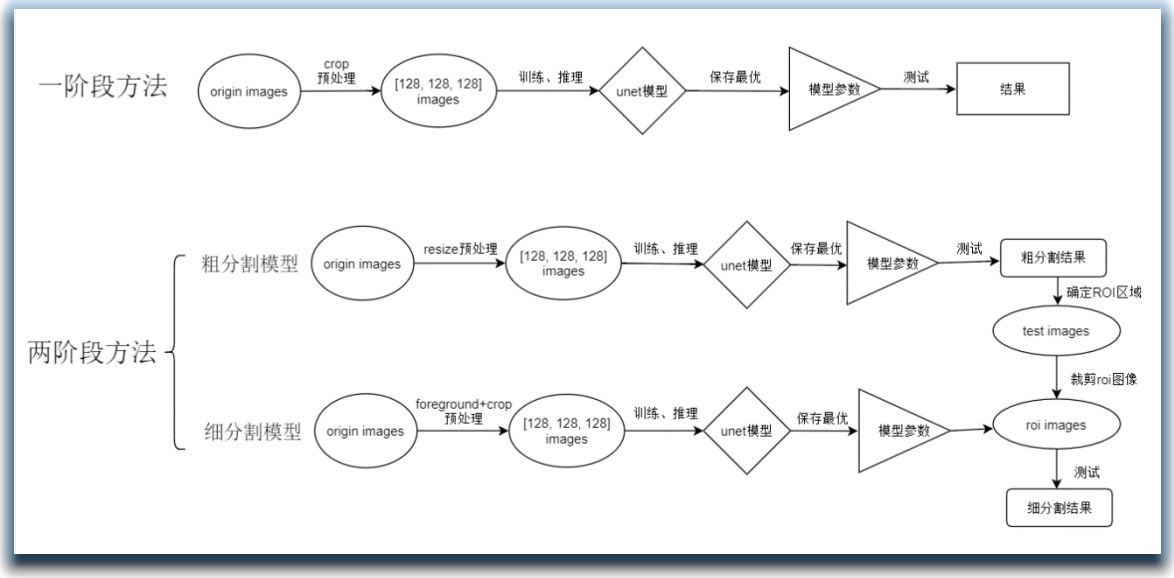
MONAI（Medical Open Network for AI）是一个用于医学图像分析的开源深度学习框架，旨在加速医学图像处理和人工智能（AI）模型的开发和部署，提供了丰富的工具和功能，用于处理医学图像数据、构建深度学习模型，并进行模型的训练、验证和推理。

MONAI 框架包含以下组件和功能：

1. **数据预处理**：MONAI 提供了丰富的医学图像数据预处理工具，包括图像的加载、裁剪、缩放、旋转、翻转、标准化、直方图均衡化等操作，用于准备用于深度学习模型的训练数据。
2. **数据加载和处理**：MONAI 支持常见的医学图像数据格式，如 NIfTI、DICOM 等，并提供了用于数据加载和处理的工具，例如数据集类（Dataset）、数据加载器（DataLoader），以便在训练和推理过程中高效地处理大规模医学图像数据。

3. **深度学习模型：**MONAI 支持多种常用的深度学习模型，如卷积神经网络（Convolutional Neural Networks, CNNs）、生成对抗网络（Generative Adversarial Networks, GANs）、循环神经网络（Recurrent Neural Networks, RNNs）等，并提供了用于构建和配置这些模型的工具，如模型定义（Module）、层（Layers）、损失函数（Losses）等。
4. **模型训练和验证：**MONAI 提供了用于模型训练和验证的高级工具，包括模型训练器（Trainer）、验证器（Validator）、学习率策略（Learning Rate Schedule）等，用于实现自定义的训练和验证逻辑，例如批量处理、学习率调整和模型性能评估。

1.2 项目思路



本次项目采用了一阶段和两阶段两种方法：

第一种方法是一阶段的，是普通的训练+推理过程，但是效果并不好。

因此转而采用两阶段的方法去做，主要思路是分别训练出粗细两个模型，由粗模型确定原图像的 ROI 区域，然后使用细模型对 ROI 区域进行分割得到最终结果。

2 相关文献介绍

2.1 参考文献

[1] U-Net: Convolutional Networks for Biomedical Image Segmentation

[2] Deep Residual Learning for Image Recognition

[3] UNet++: A Nested U-Net Architecture for Medical Image Segmentation

[4] nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation

[5] 3D UNet Learning Dense Volumetric Segmentation

[6] V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation

[7] Alom, Md Zahangir, Mahmudul Hasan, Chris Yakopcic, Tarek M. Taha, and Vijayan K. Asari. "Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation." arXiv preprint arXiv:1802.06955 (2018).

[8] Oktay, Ozan, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori et al. "Attention U-Net: learning where to look for the pancreas." arXiv preprint arXiv:1804.03999 (2018).

2.2 介绍

涵盖了经典 UNet 结构，以及在 3D 图像上的应用 3D Unet，还有引入残差模块的改进，深度融合改进的 Unet++,优化处理方式的改进 nnUNet。还有 VNet，虽然本次项目并未使用 VNet，后续处理相关问题时也值得尝试。

3 实验数据介绍

3.1 数据集简介

- 数据集来自“Medical Segmentation Decathlon Challenge”的肝脏分割部分，从老师处拷贝划分好的训练集、验证集和测试集。
- 数据集中包含肝脏和肝脏肿瘤，本次项目仅需对肝脏进行分割，根据“dataset.json”文件对标签进行相应修改，其中(0:背景、1:肝脏、2:肝

脏肿瘤)，将肝脏肿瘤部分全部归类为肝脏。

- 数据格式为“.nii”格式，使用 LoadImaged 调用 nibabel 库对数据进行读取。

3.2 预处理

3.2.1 transform

moai 提供了丰富的数据增强的 transform（变换）来处理医学图像数据

本次项目 transform 的使用说明：

1. ScaleIntensityRanged（强度缩放）：

医学图像数据不同于自然图像，其像素值取值范围等同于有符号十六位整数，根据数据集的情况来看，用固定值截断的方法将其限制在[-300, 300]的范围内

2. SpatialPadd（空间填充）

每个图像的形状一般都不相同，设置填充形状为[128, 128, 128]，在不足的维度上进行填充

3. RandCropByPosNegLabeld（随机裁剪）

图像太大，整个用于训练速度很慢，进行裁剪，裁剪形状为[128, 128, 128]，裁剪出 4 个子样本

4. Resized（调整大小）

将图像以插值的方式调整到指定的大小[128, 128, 128]

5. CropForegroundd（裁剪前景）

裁剪出图像的前景，并且设置在裁出的前景周围填充 30，每个图像裁剪结果不一致

3.2.2 使用情况

一阶段方法：

- 训练集：ScaleIntensityRanged、SpatialPadd、RandCropByPosNegLabeld
- 验证集：ScaleIntensityRanged
- 测试集：ScaleIntensityRanged

两阶段方法：

- 粗分割：提前将 origin 目录的原图像 Resized，存入 resized 目录，训练预处理只有 ScaleIntensityRanged
- 细分割：提前将 origin 目录的原图像 foreground，存入 foreground 目录，训练预处理 ScaleIntensityRanged + Crop

3.3 划分

一阶段：

- 训练集和验证集共 131 个样本，训练集：验证集=3：1=78：26，剩下 27 不处理（内存空间不足的问题）
- 测试集共 70 个样本

两阶段：

- 训练集和验证集共 131 个样本，训练集：验证集=4：1=104：27，全部使用
- 测试集共 70 个样本

3.4 加载

1. 使用 monai 中 Dataset 和 DataLoader 两个组件进行数据的预处理和加载

1）Dataset 是基本类，训练集采用 CacheDataset（提供数据缓存机制的扩展类）。由于训练集数据每轮训练都要用，它会一次性将数据全部加载到内存中，并且在 CPU 第一次加载数据时将变换后的数据缓存起来，避免重复的数据加载，访问效率更高。

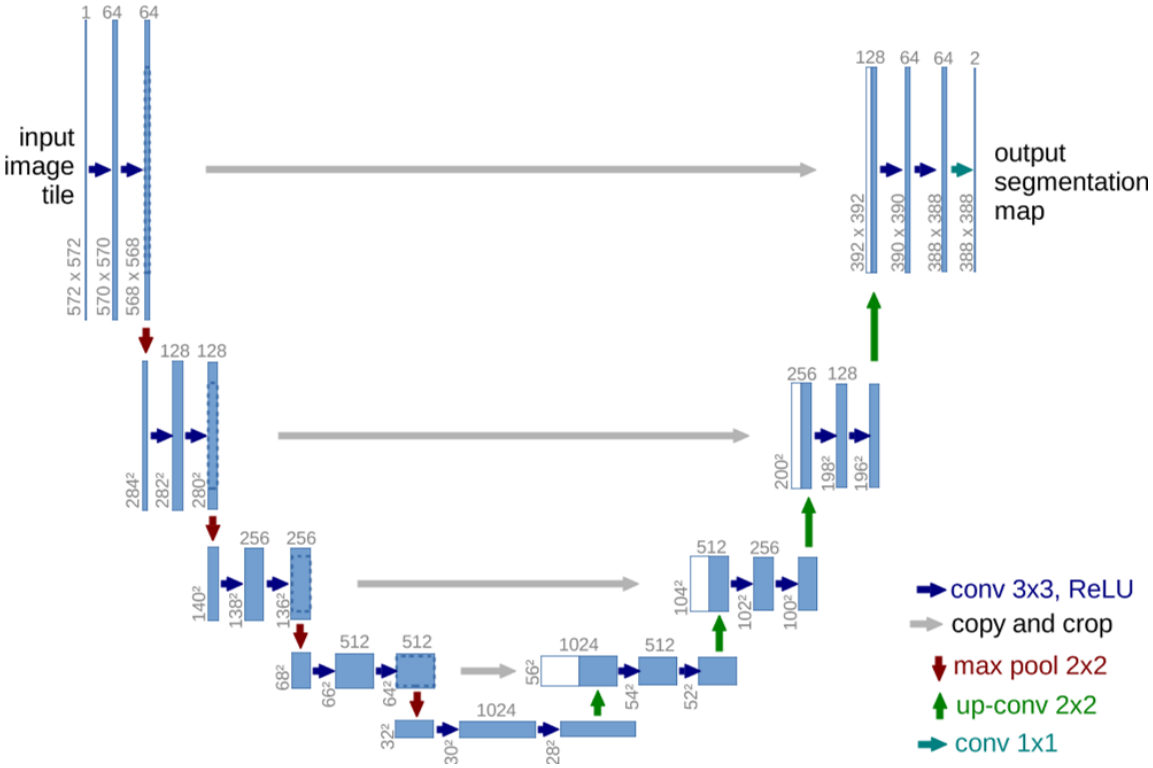
2）而对于验证集，就是用普通的 Dataset，验证一轮加载一次（访问频率低），其次是缓存空间有限而图像大（nii.gz 文件占用空间在 100M 到几个 G）

2. batch_size 设置为 1，避免对 GPU 显存的过高占用

4 实验方法描述

4.1 模型结构

4.1.1 Unet



图例：

- 深蓝色箭头：3*3 卷积和 ReLU 激活

- 红色箭头：2*2 最大池化，即下采样
- 绿色箭头：2*2 反卷积，即上采样
- 灰色箭头：复制与裁剪
- 青色箭头：1*1 卷积

4.1.2 解析

- 左侧是收缩路径，每层两次（卷积+ReLU）和一次 max 池化操作（下采样），每次下采样特征通道数量翻一倍。
- 右侧是对称扩展路径，每层一次上采样和两次（卷积+ReLU），将特征通道数量减半，并且拼接融合收缩路径同层的特征图。

4.1.3 特点

1. Unet 是 Encoder-Decoder 的结构，Encoder 部分提取特征，Decoder 部分恢复原始分辨率
2. U-net 通过通道的拼接保留了更多的维度/位置 信息，结合浅层特征和深层特征
3. 基本网络为 4 层或者 5 层，模型整体是对称的，适于模块化

4.2 构建

- 卷积层参数初始化使用 kaiming 初始化方法：

$$nn.init.kaiming\_(normal\_(m.weight, nonlinearity = \text{^}'relu\text{^}'))$$

以均值为 0、标准差为计算得到的标准差值来初始化输入参数的权重，指定 ReLU 激活函数以更准确地计算标准差，使模型在前向传播过程中保持方差不变，以减少梯度消失或梯度爆炸的问题。

- 基本模块：

1）_ConvINReLU3D，进行 conv+norm+drop+relu 的处理

2）_ConvIN3D，进行 conv+norm 的处理

- 编码器模块和解码器模块：

1）UnetTwoLayerBlock：两次_ConvINReLU3D

2）ResTwoLayerConvBlock ： residual_unit 进 行 _ConvINReLU3D + _ConvIN3D 的处理，shortcut_unit 进行_ConvIN3D 的处理，然后进行相加

- 模型有五层，每层的通道数设置为[16, 32, 64, 128, 256]

- 编码阶段--进行四次编码+池化的操作：

$$x = self.conv0\_0(x)$$

$$x1\_0 = self.conv1\_0(self.pool(x))$$

$$x2\_0 = self.conv2\_0(self.pool(x1\_0))$$

$$x3\_0 = self.conv3\_0(self.pool(x2\_0))$$

$$x4\_0 = self.conv4\_0(self.pool(x3\_0))$$

- 解码阶段--再进行四次解码+（上采样+拼接）的操作：

$$x3\_0 = self.conv3\_1(torch.cat([x3\_0, self.up(x4\_0)], 1))$$

$$x2\_0 = self.conv2\_2(torch.cat([x2\_0, self.up(x3\_0)], 1))$$

$$x1\_0 = self.conv1\_3(torch.cat([x1\_0, self.up(x2\_0)], 1))$$

$$x = self.conv0\_4(torch.cat([x, self.up(x1\_0)], 1))$$

- 最后进行主点卷积，进行不同通道特征图的融合

$$x = self.final(x)$$

4.3 训练组件

4.3.1 优化器

使用 monai 提供的 Novograd 优化器

$$optimizer = Novograd(model.parameters(), lr = args.lr)$$

两个特点：自适应学习率和梯度修剪

1) Novograd 通过根据梯度的方向和大小来自适应地调整学习率，以更好地适应不同参数的变化情况，这使得 Novograd 在训练深度学习模型时更容易找到全局最优解或更接近最优解的局部最优解。

2) 引入了一种新的正则化项，称为"orthogonal regularization"（正交正则化）。这个正则化项通过鼓励模型参数之间的正交性来提高模型的泛化能力。正交正则化可以使模型的参数更加均衡，减少过拟合的风险。

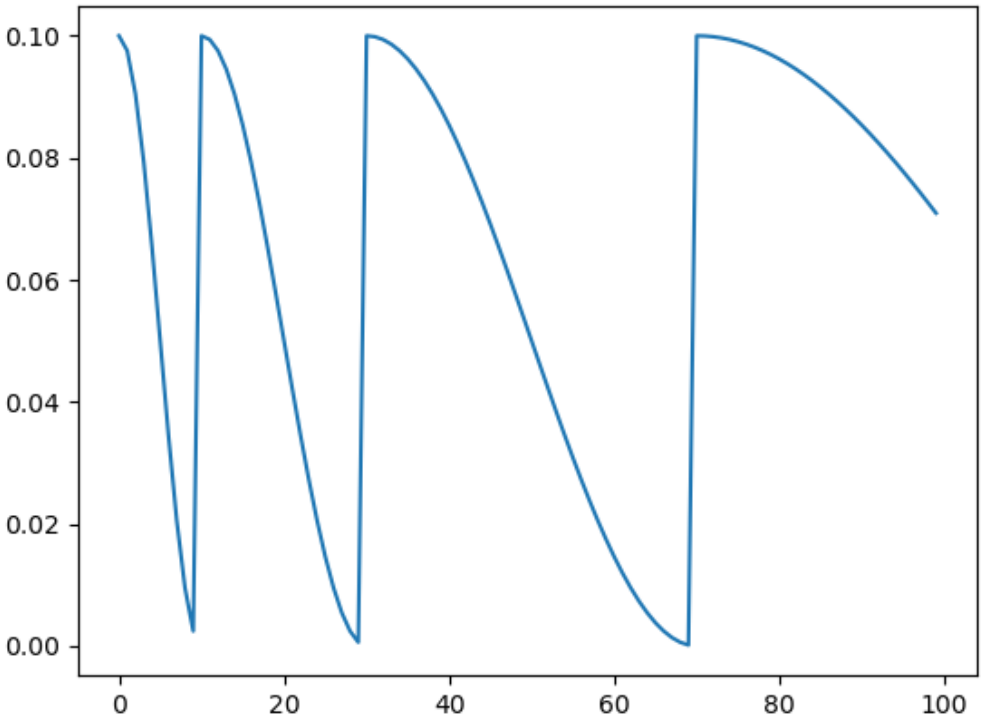
4.3.2 学习率调度器

使用 torch 提供的 CosineAnnealingWarmRestarts（余弦退火策略）学习率调度器，

$$scheduler = CosineAnnealingWarmRestarts(optimizer = optimizer, T\_0 = 5, T\_mult = 3)$$

其中，T_0 是学习率第一次达到最大值的 epoch，T_mult 控制学习率变化周期，设置为 3，分别在 epoch 为 5,20,50....时达到最大值

学习率变化图示（T_mult=2 时）：



Novograd 优化器则可以自适应地调整每个参数的学习率，以更好地适应不同参数的变化情况

CosineAnnealingWarmRestarts 帮助模型在训练初期更快地收敛，并在后续阶段进行更细致的调整。它可以有效地帮助模型跳出局部最优解并找到更优的全局最优解

通过 Novograd+CosineAnnealingWarmRestarts 的组合，可以提高模型的训练效果，并更快地达到更好的性能水平。

4.3.3 评价指标

使用 monai 提供的 DiceMetric

$dice\_metric = DiceMetric(include\_background = False, reduction = "mean", get\_not\_nans = False)$

每个批次有多个滑动窗口做验证，取 dice 系数的平均值

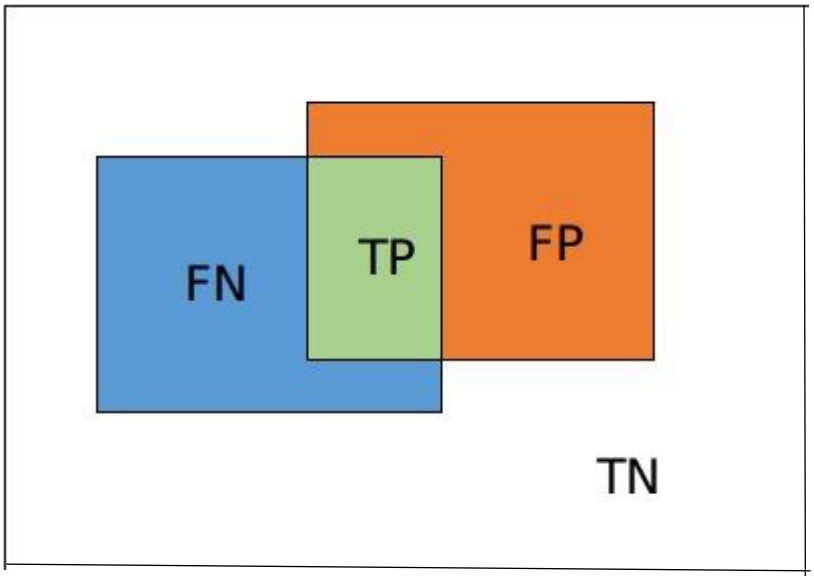
对于二分类问题，一般预测值分为以下几种:

1) TP: true positive，真阳性，预测是阳性，实际也是正例。

2) TN: true negative，真阴性，预测是阴性，实际也是负例。

3) FP: false positive，假阳性，预测是阳性，实际是负例。

4) FN: false negative，假阴性，预测是阴性，实际是正例。



dice 系数：

$dice = 2TP/(2TP + FP + FN)$

4.3.4 损失函数

使用 monai 提供的 DiceLoss

$loss\_function = DiceLoss(to\_onehot\_y = False, sigmoid = True, include\_background = False)$

$$L_{dice} = 1 - \frac{2|X \bigcap Y|}{|X|+|Y|}$$

其中 |X∩Y| 是 X 和 Y 之间的交集， |X| + |Y| 表示 X 和 Y 之间的并集，分子乘

2 为了保证分母重复计算后取值范围在［0,1］之间

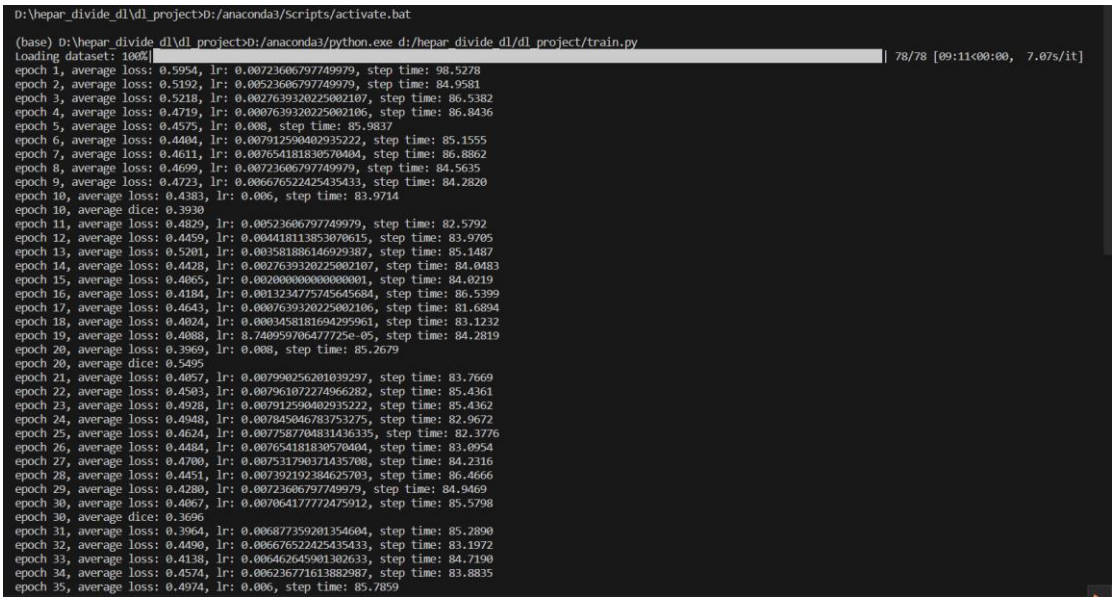
5 实验结果及分析

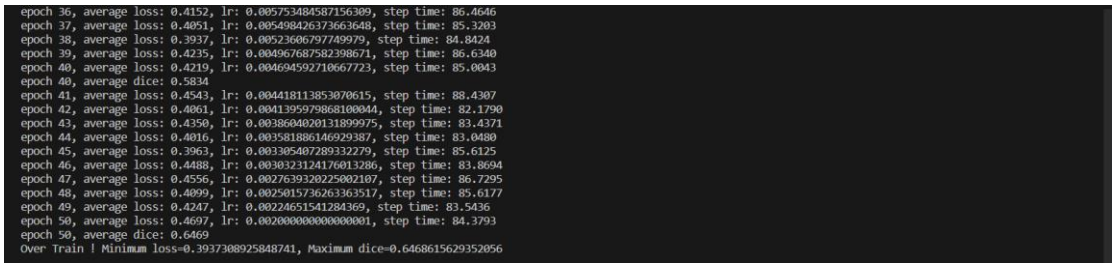
5.1 实验结果

5.1.1 一阶段

1. 第一次结果

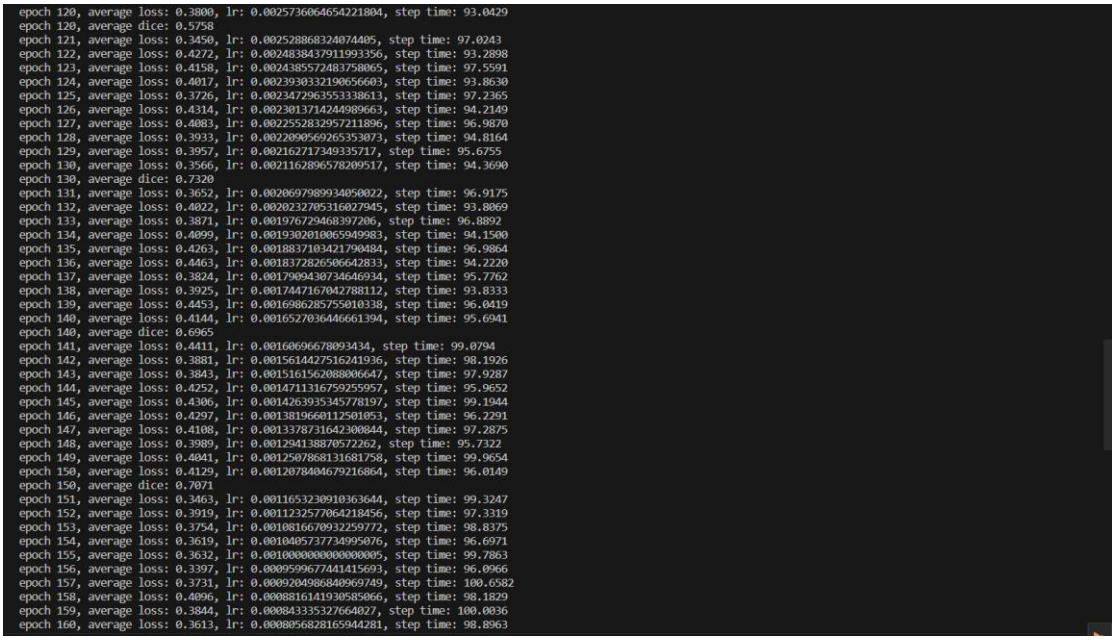
- 总样本=131，训练集：验证集=3：1，分别为 78：26，空闲 27 个样本
- 使用 UnetTwoLayerConvBlock 模块作为编码和解码模块
- 学习率初始设置为 0.008，学习率调度器为余弦退火 CosineAnnealingWarmRestarts，T_0=5，T_mult=3
- 训练轮数为 50，验证间隔轮数为 10

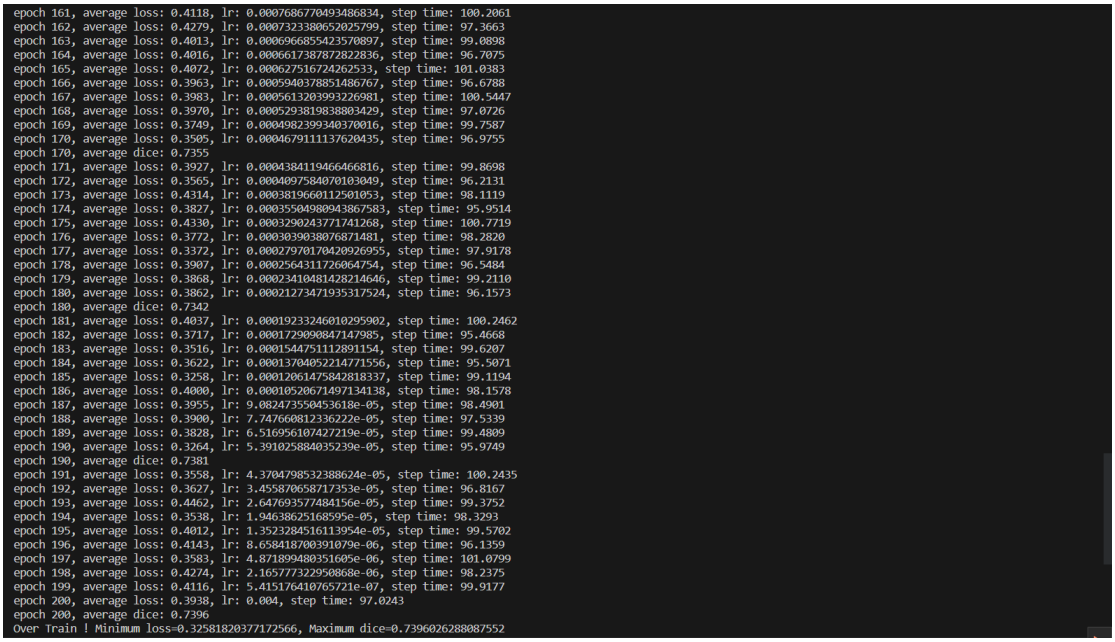




2. 第二次结果

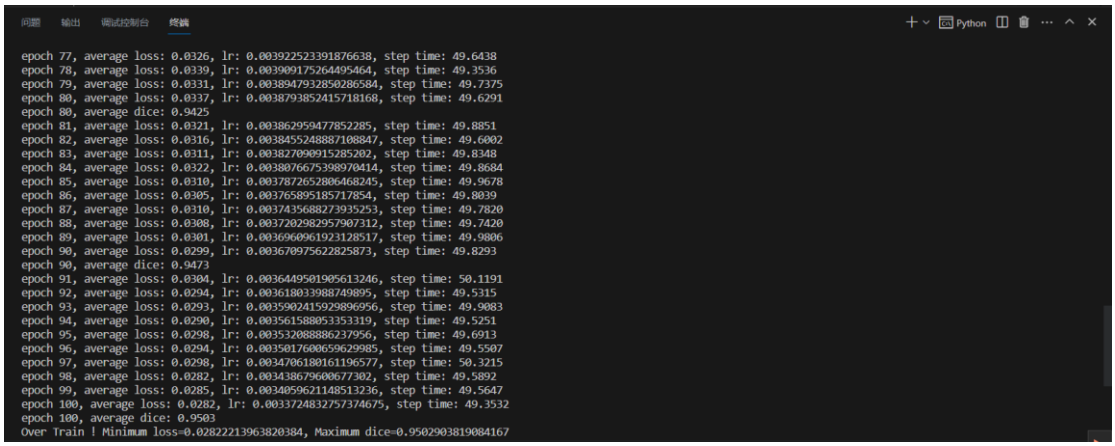
- 使用 ResTwoLayerConvBlock 模块作为编码和解码模块
- 总样本=131，训练集：验证集=3：1，分别为 78：26，空闲 27 个样本
- 学习率初始设置为 0.004，学习率调度器为余弦退火 CosineAnnealingWarmRestarts，T_0=5，T_mult=3
- 训练轮数为 200，验证间隔轮数为 10





5.1.2 两阶段

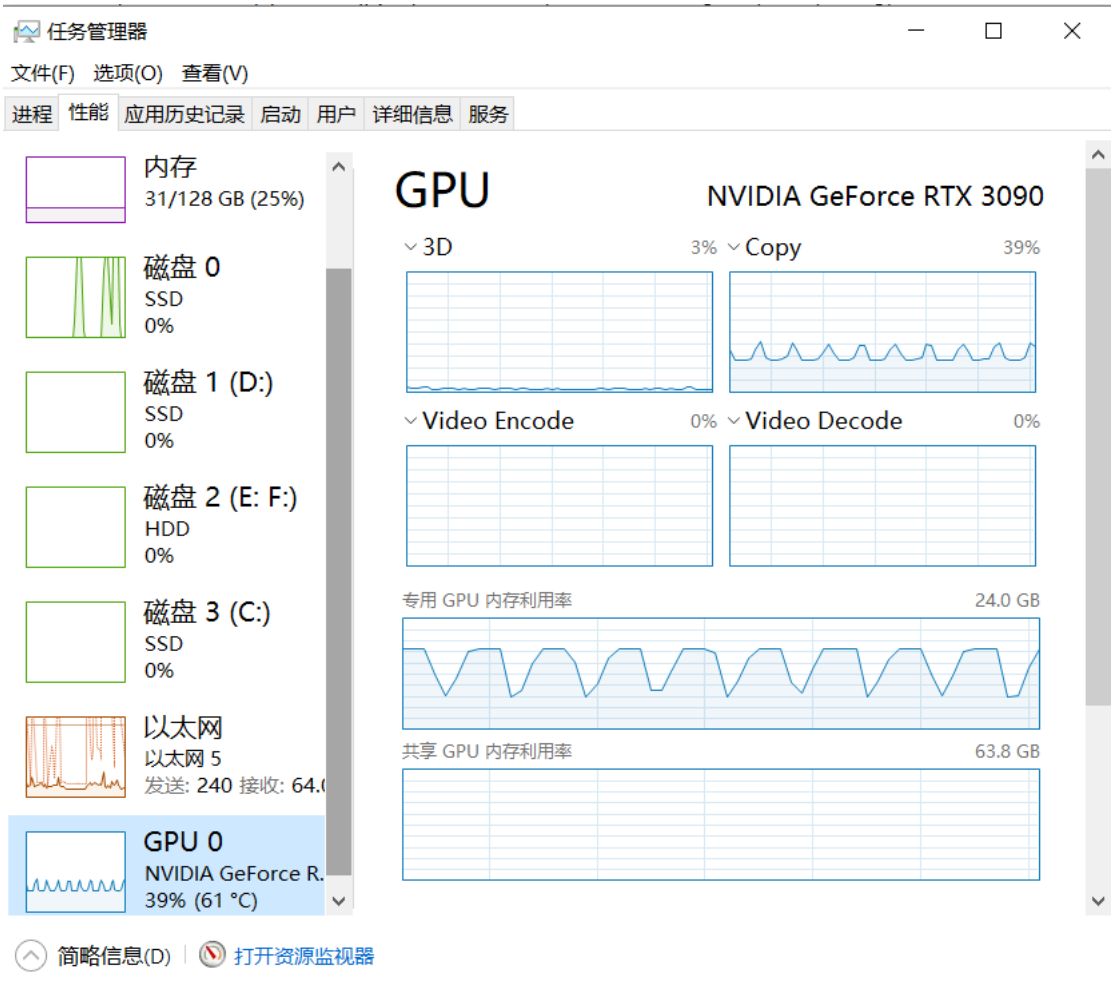
粗分割模型训练



训练 100 轮，训练时间大幅减少，准确率大幅提升，作为粗分割的模型比较合适

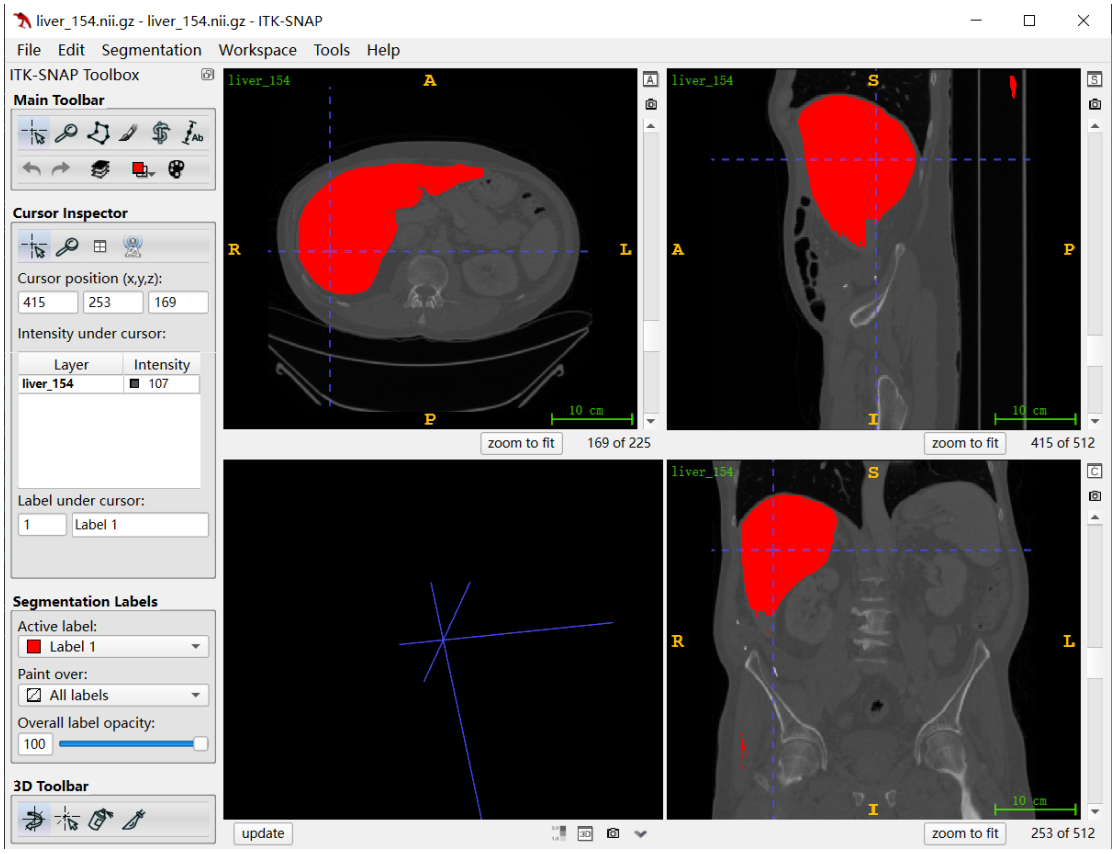
细分割模型训练

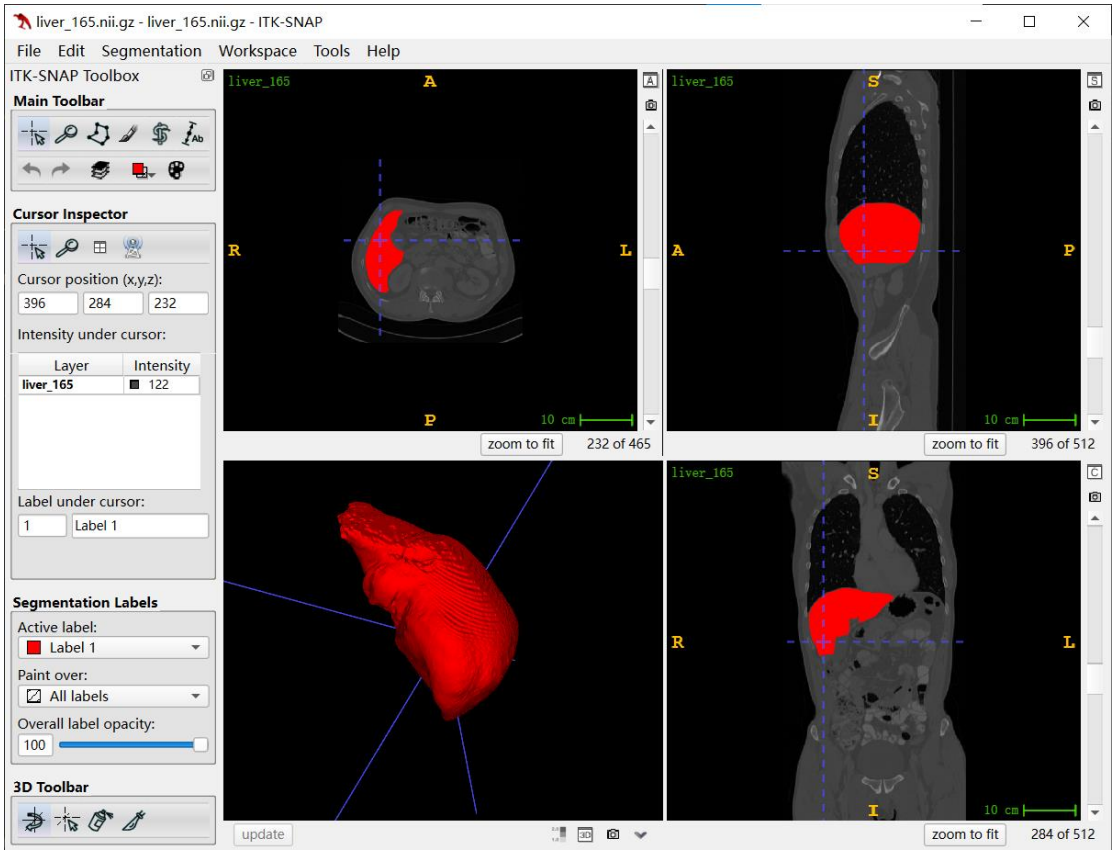
```
epoch 76, average loss: 0.1100, lr: 0.003934830438925728, step time: 118.1752
epoch 77, average loss: 0.1170, lr: 0.003922523391876638, step time: 118.1700
epoch 78, average loss: 0.1081, lr: 0.003909175264495464, step time: 118.3193
epoch 79, average loss: 0.1044, lr: 0.003894793285086584, step time: 118.1201
epoch 80, average loss: 0.1320, lr: 0.003879385241718168, step time: 118.2969
epoch 80, average dice: 0.9372
epoch 81, average loss: 0.1164, lr: 0.003862959477852285, step time: 117.6917
epoch 82, average loss: 0.0947, lr: 0.003845524888710884?, step time: 117.8253
epoch 83, average loss: 0.1092, lr: 0.003827090915285202, step time: 117.7431
epoch 84, average loss: 0.0978, lr: 0.003807667539897041?, step time: 117.7273
epoch 85, average loss: 0.0993, lr: 0.003787265280646824?, step time: 117.7707
epoch 86, average loss: 0.1139, lr: 0.003765895185717854, step time: 117.5967
epoch 87, average loss: 0.1053, lr: 0.003743568827393525?, step time: 117.7938
epoch 88, average loss: 0.0828, lr: 0.003720298295790731?, step time: 117.7787
epoch 89, average loss: 0.1197, lr: 0.003696096192312851?, step time: 117.4480
epoch 90, average loss: 0.0970, lr: 0.003670975622825873, step time: 118.5935
epoch 90, average dice: 0.9364
epoch 91, average loss: 0.0877, lr: 0.003644950190561324?, step time: 117.6838
epoch 92, average loss: 0.1027, lr: 0.003618033988749895, step time: 117.6807
epoch 93, average loss: 0.0984, lr: 0.003590241592989695?, step time: 117.8354
epoch 94, average loss: 0.1027, lr: 0.003561588053353319, step time: 117.8529
epoch 95, average loss: 0.1011, lr: 0.003532088886237956, step time: 117.7687
epoch 96, average loss: 0.0952, lr: 0.003501760065962998?, step time: 117.6828
epoch 97, average loss: 0.0924, lr: 0.003470618016119657?, step time: 117.7070
epoch 98, average loss: 0.1161, lr: 0.003438679600677302, step time: 117.8497
epoch 99, average loss: 0.0962, lr: 0.003405962114853236, step time: 117.7916
epoch 100, average loss: 0.1159, lr: 0.003372483275737467?, step time: 117.8775
epoch 100, average dice: 0.9358
Over Train ! Minimum loss=0.082752027047368?, Maximum dice=0.9398808236475344
```



内存和 GPU 占用情况：内存没有负担，GPU 显存压力还是大

5.1.3 实验结果


liver_154.nii.gz - liver_154.nii.gz - ITK-SNAP
File Edit Segmentation Workspace Tools Help
ITK-SNAP Toolbox
Main Toolbar
Cursor Inspector
Cursor position (x,y,z):
415 253 169
Intensity under cursor:
Layer Intensity
liver_154 107
Label under cursor:
1 Label 1
Segmentation Labels
Active label:
Label 1
Paint over:
All labels
Overall label opacity:
100
3D Toolbar
liver_154
A
R L
P
10 cm
zoom to fit 169 of 225
S
A P
I
zoom to fit 415 of 512
update
R L
zoom to fit 253 of 512


liver_165.nii.gz - liver_165.nii.gz - ITK-SNAP
File Edit Segmentation Workspace Tools Help
ITK-SNAP Toolbox
Main Toolbar
Cursor Inspector
Cursor position (x,y,z):
396 284 232
Intensity under cursor:
Layer Intensity
liver_165 122
Label under cursor:
1 Label 1
Segmentation Labels
Active label:
Label 1
Paint over:
All labels
Overall label opacity:
100
3D Toolbar
liver_165
A
R L
P
10 cm
zoom to fit 232 of 465
S
A P
I
zoom to fit 396 of 512
update
R L
zoom to fit 284 of 512

5.2 结果分析

- 训练轮数的增加有助于模型尽可能找到全局最优解，从 50 轮到 200 轮的结果可以看出，dice 系数有了明显提升，但是训练轮数的次数提高使得训练所需时长大幅提升，在有限的时间内不是好方法
- 学习率的适当降低有助于模型找到最优解，从 0.008 到 0.004，高学习率会使得模型直接跳过最优解，导致没有记录到最优解，但是低的学习率也会导致模型寻找解的速度变慢，需要多次实验权衡
- 使用 ResNet 的残差模块去替换 Unet 的编码和解码模块，可以看到 dice 系数有提升，但与此同时模型的复杂度也提升了，对 GPU 的要求提高（24G 的显存勉强够用），训练时所需长也适当增加，这种方法有效
- 从一阶段方法转到两阶段方法，不仅准确率大幅提升，训练时间也大幅减少，对内存和 GPU 的压力也大幅降低。

6 总结与展望

6.1 项目总结

本次项目学习了深度卷积网络的基本概念，掌握了 unet 网络的构建方法，并且尝试用 ResNet 的残差结构去替换 unet 的编码和解码模块，查看不同的改进方式对训练效果的影响。通过本次项目，提高了处理数据、使用数据、分析数据的能力，提高了解决问题的能力，受益匪浅

6.2 思路改进

- 进行更有效的数据增强方法，如扭曲、随机旋转等，在本次项目中只是使用了空间填充和随机裁剪两个方法结合控制输入图像的形状相同
- 采用其他的学习率调度器，在本次项目中使用了余弦退火重启策略，可以尝试线性预热余弦退火等其他策略验证效果
- 增加 unet 模型的每层的通道数，在本次项目中是[16, 32, 64, 128, 256]，论文里面是[64, 128, 256, 512, 1024]，更多的通道意味着提取的特征更加全面和细致，但是模型的复杂度也会对应提高。