模式识别与机器学习大作业实验报告
细胞图像识别与分割

自 74 江自远 2017011524

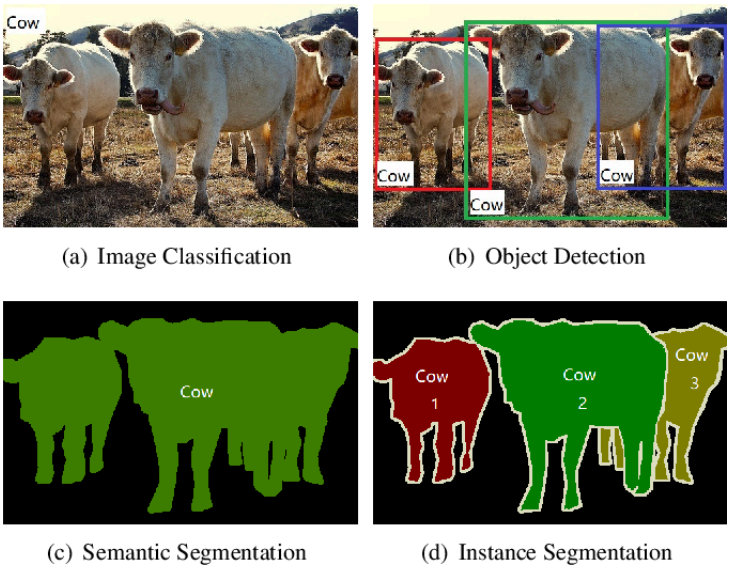
2020 年 6 月 17 日

摘要

细胞图像识别与分割是医学图像处理中重要的一部分。本次大作业的任务是对两组细胞图像数据集进行实例分割。实例分割是图像识别任务中较难的一类问题，需要逐像素点将目标从背景中识别出来，并区分同一类型目标的不同实例。本次任务相对简单，目标只有一种类别，且细胞形状较为规整。鉴于数据集的特性，采用 U-Net 以及 U-Net++ 进行语义分割，并采用 Meanshift 与分水岭 (Watershed) 算法进行后处理以得到不同实例。同时，也针对该数据集的特性对算法进行了优化与改进。最终结果显示，采用 U-Net 与 Meanshift 在数据集 1(dataset1) 中取得了最优解，而 U-Net 与分水岭算法在数据集 2 中达到了最优。

1 文献调研

计算机视觉 (Computer Vision) 目前被认为有四大经典任务，分别是图像分类 (Image Classification)、目标检测 (Object Dectection)、语义分割 (Semantic Segmentation) 与实例分割 (Instance Segmentation)。本次大作业需要完成的是实例分割任务，它兼具语义分割与目标检测的特点，前者需要逐像素点地标记目标，而后者需要区分出同类目标的不同实例。下图很好地阐述了这四种任务的不同特点。



(a) Image Classification (b) Object Detection

(c) Semantic Segmentation (d) Instance Segmentation

图 1: 视觉四大任务 [1]

截止目前 (2020 年 6 月)，实例分割的方法大致可以分为两大类：单阶段 (Single Stage) 实例分割与两阶段 (Two Stages) 实例分割。而两阶段方法又可以再进一步细分为自上而下 (Top-Down) 方法与自下而上 (Bottom-Up) 方法。

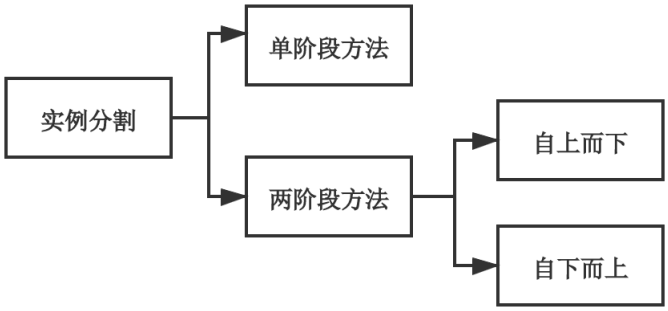


图 2: 实例分割方法

具体来说，两阶段方法指的是语义分割与区分实例的方法结合的方法。其中，首先通过目标检测的方法划定实例所在的限定区域 (Bounding Box)，再在框内进行语义分割的方法被称为自上而下的实例分割方法；相对应地，自下而上的实例分割方法则是先进行语义分割，再通过后处理的方法以区分不同实例。虽然在许多数据集上，许多单阶段方法在目前都获得了比两阶段方法更优的表现结果，但是两种方法都有其优势与局限点，因此进行学习与调研是必要的。接下来简要介绍这些方法的一些经典模型，进行比较后给出本次大作业选择的方法。

首先对于两阶段法，自下而上的方法是最早出现的，原因在于语义分割的发展也较早，因此只需进行后处理就可以进行实例分割了。2015 年，Long et al.[2] 提出的全卷积网络 (Fully Convolutional Networks, FCN) 是语义分割的经典之作，它创新地将以往卷积神经网络的全连接层换成了卷积层，并进行不同尺度信息的融合，达到了很好的效果。遵循相同的原理，Ronneberger et al.[3] 设计了 U-Net，U-Net 比 FCN 实现了更为丰富的信息融合，降采样与上采样对感受野 (Receptive Field, RF) 的控制使得特征提取更为精确完整，在下文会对该网络进行细致分析，这里先略过。除此之外，诸如 SegNet[4]、Deeplab[5] 都是很有特点的语义分割模型。其中，Deeplab 在后处理方法中采用的条件随机场 (Conditional Random Field,CRF) 很好地解决了 U-Net 等模型在进行上采样后特征损失的问题。在进行语义分割后，需要区分不同的实例。近年来，实例嵌入 (Instance Embedding) 的方法很有前景，其中许多方法的后处理都值得借鉴。Brabandere et al.[6] 在 CVPR2017 的文章中，在后处理中使用了判别式损失函数训练网络将每个像素投影到高维特征空间，使得相同实例的像素靠近而不同实例的像素相互远离，并使用聚类的方法 (Meanshift) 来获得不同实例。

在自上而下的方法中，最为经典的代表模型就是 He et al. 提出的 Mask R-CNN 模型 [7]，它的诞生经过了 R-CNN[8]，Fast R-CNN[9] 等模型的一系列迭代，在许多大数据集上的实例分割都有很好的表现。它采用 Box Head 进行目标检测，Mask Head 进行实例分割，适合较大的、复杂的数据集。

单阶段法则抛去了两阶段法的束缚，基于区域候选网络 (Region Proposal Network,RPN) 的锚框 (anchor)，开发出了一系列模型，如 SOLO[10] 与 YOLACT[11]；相对应地，也有一系列不需要锚框 (anchor-free) 的算法，如 PolarMask[12] 等。

在本次大作业中，数据集的特点是：(1) 医疗图像 (细胞)，(2) 数据集规模较小 (两个数据集均分别只有 170 张左右以供训练)。因此，首先需要排除的方法是单阶段方法，虽然其相比于双阶段方法，在端到端性能，模型特点上都有较大优势，但是大部分的模型仍然需要一定规模的训练集进行训练，因此暂时不予考虑。在双阶段方法里，自上而下的方法仍然面临着训练集规模不足的问题，因此也暂时排除。最后，考虑采用自下而上的方法。在语义分割阶段，采用适合医疗图像的 U-Net 模型进行训练，具体的后处理方法将会在下文中进一步讨论。

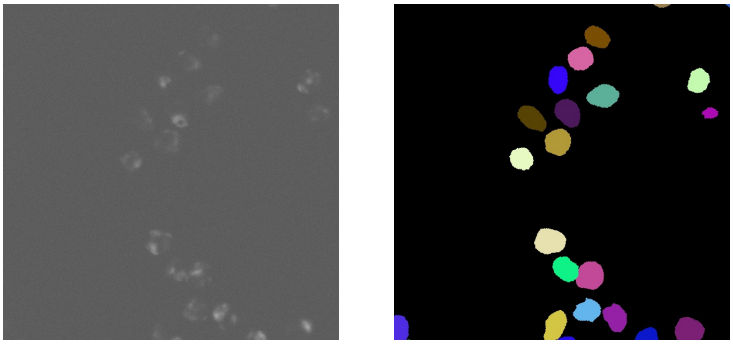
2 数据处理流程

本次数据集均由课程大作业所提供。训练集的图像以及 ground truth(GT) 为 tiff 格式，存储格式均为 uint16。GT 的像素点值由 0 到 n 的整数构成，0 为背景，不同的正整数代表不同的实例。对于数据集 2，

训练集的 GT 来自于个人标注 (5 张) 以及 DIC-C2DH-HeLa 数据集 (150 张左右，其中排除了测试集的数据，以防止信息泄露)[13]。具体来说，数据集 1 的图片大小为 628*628，共 175 张训练集数据，33 张测试集数据。数据集 2 的图片大小为 500*500，共 167 张训练集数据，6 张测试集数据。

2.1 数据可视化

数据可视化分为原始图像的可视化以及 GT 图像的可视化。原始图像的可视化只需读取图像并以 uint8 的格式输出即可。对于 GT 图像，需要对相同实例赋以相同的颜色，采用 np.random.randint(low=0, high=255, size=3) 实现，最后同样输出 uint8 格式。下图为具体示例。



(a) 原始图像可视化　　(b) GT 图像可视化

2.2 数据预处理

由于本次大作业是基于 Pytorch 搭建的模型，因此首先需要重写 torch.utils.data.Dataset 类，并将新类记作 CellDataset。一共需要重定义两个接口，___init___ 与 ___getitem___。

其中在 ___init___ 中，需要完成对数据的读取，数据包括训练集原始图像 (training data) 以及 GT 图像 (labels)。因此在接口中添加两个图像的 path 并将其存入 CellDataset 的自定义变量 imgs 中，以便之后数据的抓取。此外，定义两类图像的转换 (transform)，对于原始图像，需要将其转为 tensor 并进行归一化处理；而对于 GT 图像，只需转为 tensor 即可。

在 ___getitem___ 中，需要将数据读取并裁剪 (resize) 到 640*640 的大小 (对于数据集 2，则是 512*512 的大小)，原因与模型有关，在后文中会解释具体原因。

处理完 CellDataset 后，就可以直接调用 torch 的 DataLoader 来对数据进行划分，随机抽取等操作了。对于测试集，也需要进行相同的裁剪与转换。

3 模型及算法原理

以下简要介绍本次大作业涉及到的模型及算法。

3.1 U-Net

U-Net 诞生之初便是为了小样本的医疗影像分割而设计的，为了达到这一目的，除了模型的设计以外，作者还进行了数据增强的操作 [3]，由于时间所限，本次没有采用该措施，故略过。

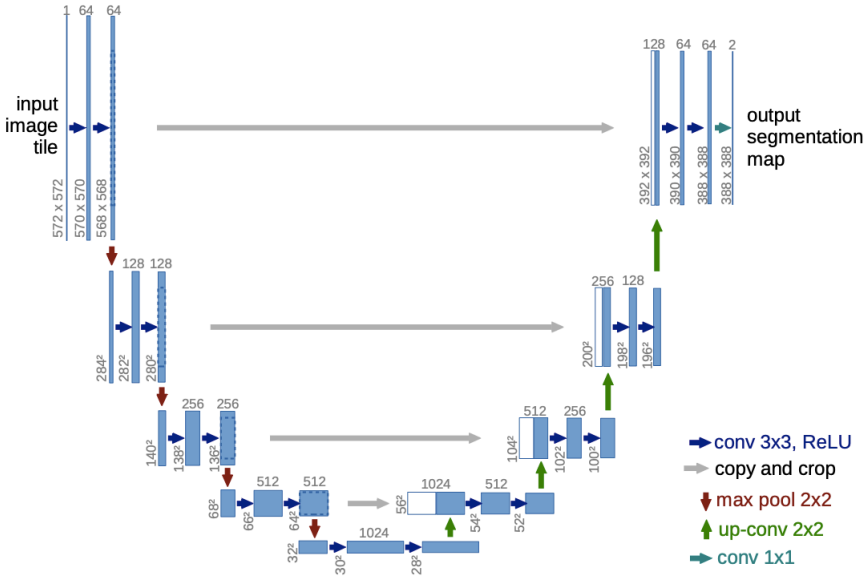


图 3: U-Net 模型架构

经典的 U-Net 架构如上图所示，由左侧的压缩路线和右侧的扩展路线构成。压缩路线遵照经典的卷积网络架构。其包含两个 3*3 卷积核的重复应用，每个卷积核后连接 ReLU 激活函数以及 2*2 的池化层 (max pooling)，步长 (stride) 为 2 以进行降采样 (downsampling)。每次降采样都将特征通道数翻倍。相对应地，在扩展路线中，每一步都经过一次上采样 (upsampling)，一个 2*2 的反卷积以减小通道数为原来的一半，然后再经过两个 3*3 的卷积核，每个卷积核后连接 ReLU 激活函数。最为关键的是，在扩展路线中，每个特征图谱 (feature map) 都是由来自上一层的反卷积与来自压缩路线的对应的裁剪 (cropped) 后的特征图谱拼接而成。其中，裁剪的原始是边界像素点在每次卷积后都有一定损失。在最后一层，通过一个 1*1 卷积将每个特征向量对应至期望的类别。总共该网络共有 23 个卷积层。需要注意的是，由于网络中的池化操作，要求输入图片的尺寸要匹配网络结构，这也是数据预处理中进行裁剪的原理。

Algorithm 1 U-Net training algorithm

Input: model, dataset, num_epochs, optimizer, criterion, scheduler, batch_size, acc_criterion, save_step

Output: trained_model

```
acc_list = []
for epoch in num_epochs do
    epoch_loss = 0
    train_acc = 0
    for step, (x, y) in dataset do
        optimizer.zero_grad()
        y_pred = model(x)
        loss = criterion(y_pred, y)
        loss.backward()
        optimizer.steop()
        epoch_loss += loss.item()
        train_acc += acc_criterion(y_pred, y)
    scheduler.step() acc_list.append(train_acc)
    if epoch % save_step == 0 then
        save model and acc_list
return model as trained_model
```

Algorithm 1 为对 U-Net 模型进行训练的伪代码，其中输入的 model 表示 U-Net 模型，num_epochs 表示训练进行的轮次，optimizer 是优化器，criterion 是损失函数，scheduler 是学习率的衰减函数，batch_size 是批训练数，acc_criterion 是正确率的评价指标，save_step 是保存模型参数及相关记录的间隔。它们的具体选择与实现将在第 4 部分详细阐述。

3.2 U-Net++

在 U-Net 诞生后，Zhou et al. 在 2018 年针对 U-Net 存在的缺陷提出了 U-Net++[14]。U-Net++ 的出现来源于对 U-Net 的一些质疑，例如为什么上采样只从最后一层开始，网络的层数该如何确定等，最终形成了一个可剪枝的巢式 U-Net 结构，并采用深监督 (Deep Supervision) 的方法解决梯度反向传播的问题。虽然 U-Net++ 不论从理论上还是实际表现上都要优于 U-Net，但由于其参数的增加，在本次作业小样本集的条件下的表现还需要进一步的实验确认。而 U-Net++ 模型的训练算法与 3.1 相似，因此不再赘述。

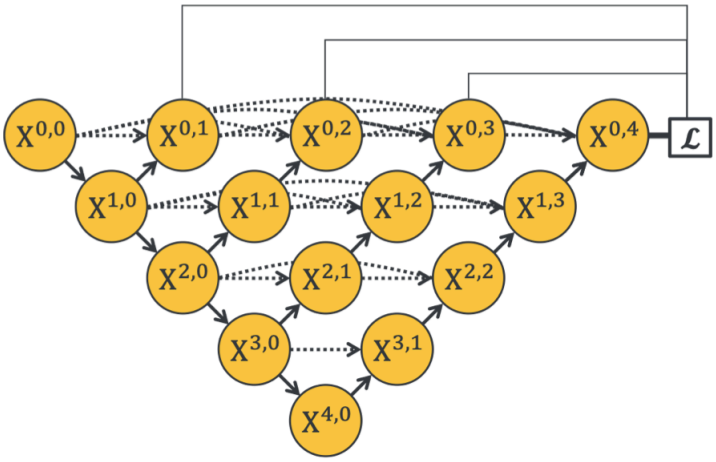


图 4: U-Net++ 模型架构

3.3 Meanshift 聚类方法

在进行语义分割后，要获得细胞的不同实例还需要通过后处理的方法。最直接的想法就是采用聚类进行区分。然而许多聚类方法，例如 K-Means 聚类、系统聚类等方法都需要指定聚类中心数，这显然不适合解决本次任务。因此，考虑采用不指定聚类数的均值漂移 (Meanshift) 算法 [15]。该算法的具体实现可直接通过调用 sklearn.cluster 库 [16] 中的 Meanshift 函数，这里简要讲解其原理。

Meanshift 的核心思想是寻找核密度极值点并作为簇的质心，然后根据最近邻原则将样本点赋予质心。首先要进行核密度估计，即使用核函数对样本空间某点进行估计，该点邻近区域的的样本会影响核密度的估计，而该区域大小由带宽 (bandwidth)h 所决定，一般来所采用高斯核 (Gaussian Kernel) 函数：

$$K(x) = \frac{1}{\sqrt{2\pi}h} e^{\frac{x^2}{2h^2}} \tag{1}$$

此外，该算法的另一个核心是均值漂移的过程。每一步算法的迭代过程都会朝着当前质心区域内密度极值方向漂移，方向由梯度确定，可求得下一个质心的位置：

$$x_k^{i+1} = \frac{\sum_{x_j \in N} K\left(x_j - x_k^i\right) x_j}{\sum_{x_j \in N} K\left(x_j - x_k^i\right)} \tag{2}$$

其中 N 为当前质心区域内样本的集合。以下是具体的算法流程。

Algorithm 2 Gaussian Meanshift Algorithm

Input: bandwidth h, bin_seeding, epsilon, max_iter

Output: labels

```
iter = 1
initialize central points x
while (shift distance < epsilon) or (iter > max_iter) do
    for x_k in x do
        get shift distance for x_k
        update x_k
        if dist(x_k,existed central points)<h then
            merge two centers
    iter += 1
label each sample point as the nearest central point
return labels
```

其中，输入的 bin_seeding 是初始质心的种子设置，epsilon 和 max_iter 是两种停止条件的判断。

3.4 分水岭算法

除了采用传统的聚类方法，在数字图像处理上的一种方法也可以对实例进行划分，该算法由于其具体的执行过程被称为分水岭 (Watershed) 算法 [17]。分水岭算法有很多不同的具体做法 [18]，其中 OpenCV 库中所采用是基于 markers 的分水岭算法，算法的具体实现在此略过，而展示如何从语义分割的结果 (二值化的 masks) 获得最后的实例分割的算法。

Algorithm 3 Watershed based on markers

Input: binary masks(img)

Output: labels

```
1: opening = opening_operation(img)
2: sure_bg = dilate(opening)
3: dist_transform = distanceTransform(opening)
4: sure_fg = threshold(dist_transform, THRESHOLD=dist_transform.max() * k)
5: unknown = sure_bg - sure_fg
6: markers = connectedComponents(sure_fg)
7: markers = markers + 1
8: markers[unknown==255] = 0
9: labels = watershed(img, markers)
10: return labels
```

整个算法的第 1 到 6 行都是为了求取 markers。markers 可以理解为每个实例的带有一定宽度的外轮廓，markers 的求取是分水岭算法有效的关键。首先需要对二值化的图像进行开操作，开操作的目的是去除图片中的小噪点 (这样的操作有可能会导致某些预测准确的小细胞被抹去)。然后进行膨胀操作，获得确定是背景的部分；进行距离变换然后取一定的阈值 (代码中的 k 控制阈值)，获得确定是前景的部分。最后在前景部分获得联通域，作为 markers。markers+1 的原因是 watershed 算法以 1 作为背景，然后将所有未知区域置零 (背景-前景即为未知部分)，最后运行 watershed 算法，即可得到每个像素点对应的标签，完成实例分割。

4 实现过程

本次大作业于 Google Colab Pro 上完成，搭载 GPU 为 Tesla P100，并采用 Pytorch 进行训练。在经过参数与模型的选择后 (具体分析将在第 5 部分进行)，选择采用 U-Net+Meanshift 的方法对两个数据集进行实例分割。接下来结合整个项目的流程架构对每个部分函数的功能进行分析，并给出具体参数的选择，具体分析将在第 5 部分中阐述。

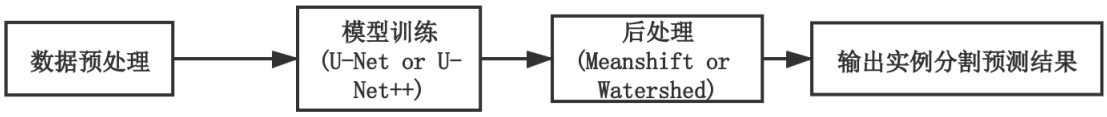


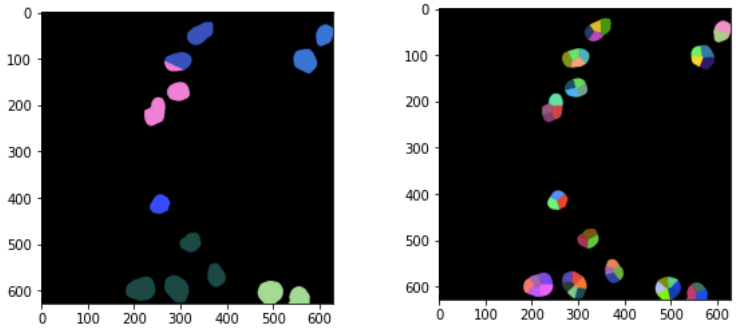
图 5: 算法实现流程图

在数据预处理部分，在第 2 部分已经提及具体的操作函数，需要注意的是，为了保证复现能力，需要设计随机数种子。为了确保训练时不出现过拟合的情况，采用 random_split 的方法按照 8:2 的比例划分训练集和验证集。

在训练过程中，关键函数为 train_model，该函数的功能为对模型进行训练并保存，具体的算法在 3.1 和 3.2 中均有详细的介绍。在通过了一系列对照试验后，最终对于两个数据集，选取 batch_size=2；初始学习率为 1e-3；损失函数为 BCELoss，即交叉熵损失函数；优化器选择 Adam；学习率衰减函数选择 StepLR，每 10 个 epoch 衰减为原来的 0.8 倍；总共训练 150 个 epoch，每 10 个 epoch 保存一次模型参数，并在训练时记录 loss 值，训练时间，以及训练集与验证集的准确率。在判断模型准确率时，需要用到 IoU(Intersection over Union) 函数。记预测结果为 x，GT 为 y，IoU 定义为

$$IoU = \frac{x \cap y}{x \cup y} \tag{3}$$

在后处理部分，函数 meanshift 与 watershed 分别对应 3.3 和 3.4 中的两种后处理算法，它们分别在数据集 1 和数据集 2 上获得了最佳表现。其中，meanshift 采用的函数需要给定带宽，而带宽的大小直接决定了分类的好坏，因此需要预先使用一定的方法来选取合适的带宽 (R)。



(a) $R = 80$ (b) $R = 10$

图 6: 带宽对聚类效果的影响

选取的方法可以采用 Jaccard 相似度求和 (记为 C) 来对聚类效果进行度量。其定义如下

$$Criteria = \frac{\sum_{i=1}^{n} Jaccard\,(Cell_i)}{n}$$

对每个 ground truth 细胞的 Jaccard 相似度求和并除以 ground truth 细胞数量。其中 n 是 ground truth 细胞的数量，$Cell_i$ 表示第 i 个 ground truth 细胞。

由于在最优带宽两侧的 C 都会下降，因此考虑从高带宽往下迭代判断，直至找到最大的 C 即为最优的带宽。可以算得数据集 1 的最优带宽为 $R = 22$。然而该方法也存在一定局限性，meanshift 能在该数据集

上适用的原因是，细胞大小差别不大，且粘连较少（meanshift 在细胞粘连处表现较差）。因此，数据集 2 在分水岭算法上获得了更好的结果。

对于数据集 2，由于细胞粘连较多，如果按照正常的做法，仍然会出现无法区分实例的结果。这时可以考虑在预处理时，对每个细胞的 GT 进行腐蚀操作，然后以此进行网络训练，此时的网络参数就会更好地学习到不同细胞的边界，在后处理后再将获得的 labels 进行等比例膨胀即可。

5 实验结果与模型性能分析

由于时间所限，许多参数的设置和模型的选择都是采用控制变量法进行，且仅对数据集 1 进行了较多实验 (以下数据均为在数据集 1 中获得的)。这样的方法存在一定局限性，因为不同的参数之间可能存在一定的相关性，较为科学的做法是遍历所有可能的组合，也就是采用 grid search 的方法寻找最优超参数。故这也是本次实验的一个遗憾之处。

(1)batch size 的选择：batch size 较大理论上可以加快运行速度，但是对 GPU 的显存需求也更大。在此条件下，选择 batch size=2 是较为平衡的做法。

(2) 初始学习率与衰减方式：初始学习率决定模型的收敛速度。学习率过小不仅影响收敛速度，还可能加剧模型的过拟合程度。

表 1: 学习率的影响

学习率	收敛时 epoch	IoU
1e-2	50	0.83
1e-3	120	0.89
1e-4	200	0.86

表 1 针对数据集 1 进行实验，收敛和 IoU 都是针对验证集的指标。可以看出，初始学习率选择 1e-3 应是较优选择。而衰减方式选择了较为鲁棒和实用的定步长衰减法，具体设置在上文中有详细解释。

(3) 损失函数选择：在语义分割模型中，较为常用的两种损失函数分别是交叉熵损失函数以及 Dice 损失函数 [19]。交叉熵损失函数可以表示为

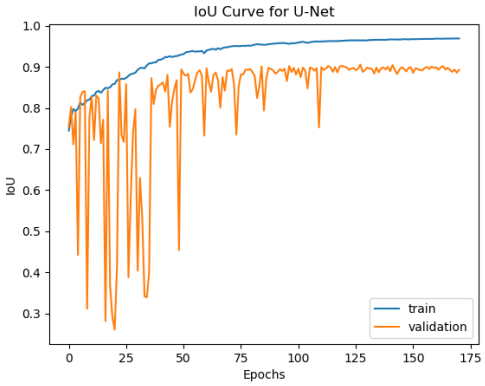
$$CrossEntropy = -\sum_{classes} y_{true} \log(y_{pred})$$

Dice 损失函数则与 IoU 类似，本质上衡量两个样本的重叠部分，即

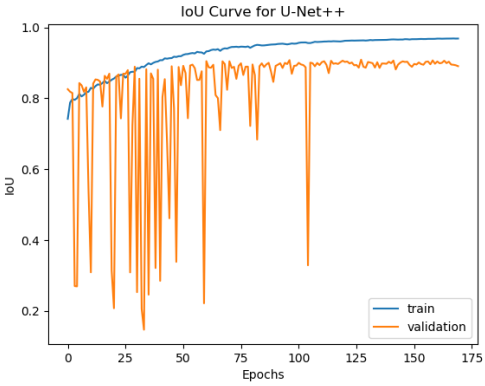
$$Dice = \frac{2|A \cap B|}{|A| + |B|}$$

交叉熵在类别不均衡的问题中表现较差，在数据集 1 中，两个损失函数的表现结果类似，选择交叉熵作为损失函数。

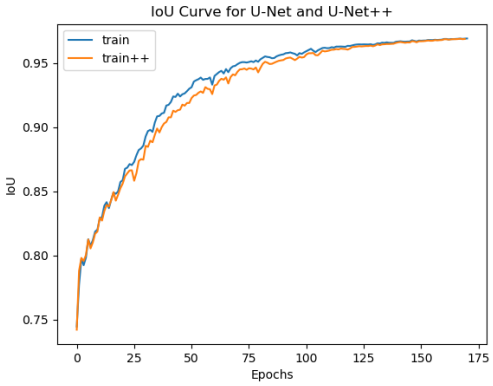
(4) 模型的选择：在相同参数设置下，比较 U-Net 和 U-Net++ 的表现情况。



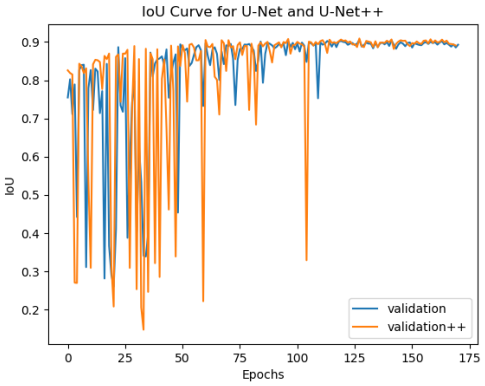
(a) U-Net IoU 曲线



(b) U-Net++ IoU 曲线



(c) U-Net 与 U-Net++ 训练曲线



(d) U-Net 与 U-Net++ 验证曲线

图 7: 数据集 1 相关曲线

从上图可以看出,U-Net 与 U-Net++ 的性能相差不大。其中在训练集的表现 U-Net 整体优于 U-Net++，在验证集上，两种模型均在 epoch 125 开始趋于收敛，且 IoU 稳定在 0.9 左右。两种模型在该数据集上的表现类似的原因可能是 U-Net++ 虽然提高了模型的总体性能，但是其参数更多，拟合也更加困难，在有限数据集上可能会限制其发挥。如果进行数据增强等操作，可能会使 U-Net++ 的表现得到提升。

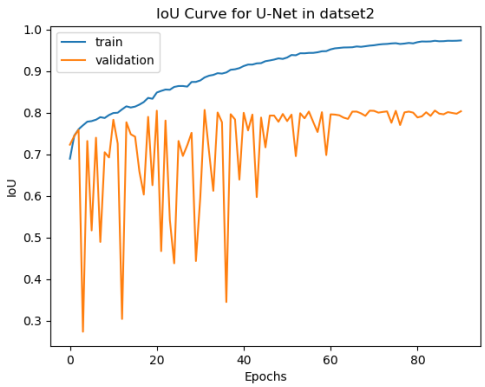


图 8: U-Net 在数据集 2 的表现

在数据集 2 中，训练准确率同样很快上升，然而验证的准确率却收敛于 0.8 附近。这是由于为了使后处理中分水岭算法可以区分不同实例，导致在数据预处理的过程中需要进行细胞的腐蚀操作。这样一来，实际上细胞的 GT 图像就不能很好地反映真实图像的特征，就会导致验证的准确率下降。可行的改进方法包括减小预处理时细胞的腐蚀程度，或是更改预处理的方式 (在附录 1 中有提及可行的解决方案) 或是在后处

理阶段采用其他的方法，从而能够区分不同的实例。

最终，数据集 1 使用 U-Net+Meanshift 算法在评测网站上获得了 0.778 的得分，数据集 2 使用 U-Net+Watershed 算法获得了 0.808 的得分。

6 总结

本次实验采用了语义分割 + 后处理的方法对细胞医疗图像进行了实例分割。实验对比了 U-Net 和 U-Net++ 在不同数据集上的表现，对比了 Meanshift 和分水岭算法在后处理上的特点，针对不同的数据集选择不同的算法，最后达到了相对出色的分割结果。然而，本次实验也有许多不足之处，有很大的改进空间。首先，在面对数据集过小的情况下，可以采用数据增强的方法。其次，还有很多的分割算法值得研究与实践，在附录 1 中会提到。最后，模型的架构可能存在调整的余地，例如调整卷积核大小，全卷积层数等，同时在数据预处理方面也可以有更多创新的余地，同样在附录 1 中会进一步介绍。

本次大作业过程中学习到了许多知识，也克服了不少困难。一个月前我对图像分割方面了解几乎为零，在一个月后已经能够自己写出一个完整的项目，十分有收获。在此感谢老师和助教们的辛勤付出，让我们在这段艰难的时期也能学有所获，祝你们一切顺利！

附录 1

在大作业的撰写调研初期，我查阅了许多在细胞实例分割上有效且新颖的做法，然而由于时间所限，没有能够实践之，对比之，因此也没有写在文献调研部分。但它们仍十分有价值，因此摘录如下。

在 [20] 中，作者提出了端到端的实例分割方法，虽然仍是两阶段的分割方法，但是其将经典的分水岭算法与深度学习结合，生成能量图谱 (energy map)，从而提高了分水岭方法的分割性能。在之后的工作中也可以借鉴该方法来改进本次任务中的后处理部分。而 [21] 则为粘连细胞的实例分割提供了很好的解决方案，这与数据集 2 的情形十分类似，可以运用到预处理中。他的解决方案包括：(1) 增强 GT 细胞的粘连边缘，通过融化 + 腐蚀的方法。(2) 通过形状认知权重图谱 (Shape Aware Weight map, SAW) 的训练方式进行实例分割，可以很好地划分粘连细胞的边界。最终该方法在相关数据集上的表现优于 U-Net 系列算法。最后，[22] 针对神经细胞设计的逐像素点的端到端实例分割神经网络，虽然也是使用经典的目标检测 + 语义分割的模型，但是其在目标检测阶段设计的 Light-weight SSD Detector 则可以使得模型更好得到训练，并在实例分割问题上可以更灵活地运用。

附录 2

本次大作业全部的源代码、测试数据以及预训练模型均可在 https://github.com/ZiyuanJiang825/cell-instance-seg-pytorch 上下载。此外，该项目也已经完整上传至清华云盘，地址为:
https://cloud.tsinghua.edu.cn/d/ac14b9bde2554e97b097/

接下来介绍代码的具体内容：

· dataset: 存放数据集

· dataset1: 数据集 1 的训练数据以及测试数据，test_RES 为预测结果，即提交至评测网站的数据。

· dataset2: 数据集 1 的训练数据以及测试数据，test_RES 为预测结果，即提交至评测网站的数据。需要注意的是，自行标注数据集 (5 张) 已经放入 train_GT 文件夹中，并单独在 train_origin 文件夹中列出。

· celldataset.py：重构两个数据集类，分别命名为 CellDataset1 与 CellDataset2

· lib: 存放相关函数

· iou.py: 定义 IoU 函数。

· meanshift.py: 定义 meanshift 算法并输出预测结果。

· watershed.py: 定义分水岭算法并输出预测结果。

· model: 存放模型架构

· unet.py: 定义 U-Net 模型。

· unet_plus.py: 定义 U-Net++ 模型。

· pretrain: 存放预训练模型

· weights_80.pth: 数据集 1 训练 80 轮次后的 U-Net 模型参数。

· Nested_weights_130.pth: 数据集 1 训练 130 轮次后的 U-Net++ 模型参数。

· ds2_weights_90.pth: 数据集 2 训练 90 轮次后的 U-Net 模型参数。

· train.py: 运行该代码即可对模型进行训练，也可从中途开始训练。

· eval.py: 运行该代码即可使用训练后的模型进行预测并输出。

· README.md: 项目相关简介。

示例:(1) 从头开始对数据集 1 进行训练，使用 U-Net 模型：直接运行 train.py

(2) 使用 U-Net 的预训练模型并使用 meanshift 算法预测数据集 1 的结果：直接运行 eval.py

参考文献

[1] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 2020.

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[6] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. *arXiv preprint arXiv:1912.04488*, 2019.

[11] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: real-time instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9157–9166, 2019.

[12] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12193–12202, 2020.

[13] http://data.celltrackingchallenge.net/training-datasets/dic-c2dh-hela.zip.

[14] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 3–11. Springer, 2018.

[15] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[17] Jos BTM Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta informaticae*, 41(1, 2):187–228, 2000.

[18] Eduardo Tusa, Anthony Laybros, Jean-Matthieu Monnet, Mauro [Dalla Mura], Jean-Baptiste Barré, Grégoire Vincent, Michele Dalponte, Jean-Baptiste Féret, and Jocelyn Chanussot. Chapter 2.11 - fusion of hyperspectral imaging and lidar for forest monitoring. In José Manuel Amigo, editor, *Hyperspectral Imaging*, volume 32 of *Data Handling in Science and Technology*, pages 281 – 303. Elsevier, 2020.

[19] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[20] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[21] Fidel A Guerrero-Pena, Pedro D Marrero Fernandez, Tsang Ing Ren, Mary Yui, Ellen Rothenberg, and Alexandre Cunha. Multiclass weighted loss for instance segmentation of cluttered cells. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2451–2455. IEEE, 2018.

[22] Jingru Yi, Pengxiang Wu, Daniel J Hoeppner, and Dimitris Metaxas. Pixel-wise neural cell instance segmentation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 373–377. IEEE, 2018.