模式识别与机器学习大作业
细胞分割

2020 年 7 月 2 日

摘要

细胞分割是医学影像处理的重要方向，其在辅助诊断，疾病风险预测等方面起到了重要的作用。但细胞分割的标注信息往往需要耗费大量的人工，因此难以保证在大量的训练集下能够有足够的与其匹配的标注信息。所以在实例分割中需要解决标注较少的问题，此外，还需解决细胞重叠和粘连的问题。在本次作业中，我在两个数据集中均使用了数据增广, 语义分割接连通域后处理的算法。对于数据集二，我进行了人工标注以及预处理和数据增广操作扩充数据集，以达到较好的语义分割结果。在语义分割的算法中，我先后使用了 $U-Net$ 和 U^2-Net[**?**]，并且使用细胞间的 Jaccard 相似度作为评估指标。结果说明后者能够在语义分割上达到更好的效果，经后处理后达到了 dataset1:0.835515、dataset2:0.854498 的得分。

关键词: 实例分割，细胞语义分割，连通域处理

1 文献调研

在本次作业的完成过程中，我首先在网上查阅相关资料调研了细胞实例分割的处理方法，并找到了一些相关的前沿论文进行阅读，包含的主流的处理方法基本可以分为两类，分别是:

1) 语义分割网络并后处理

2) 图像实例分割网络

因此接下来我也分别在这两个方面介绍我所查找到的相关方法。

1.1 任务描述

在本次大作业中，共给出了两个数据集，这两个数据集均为包含细胞的医学影像。前者为具有充足标注的数据，而后者仅具有少量标注并且存在大量的细胞粘连。

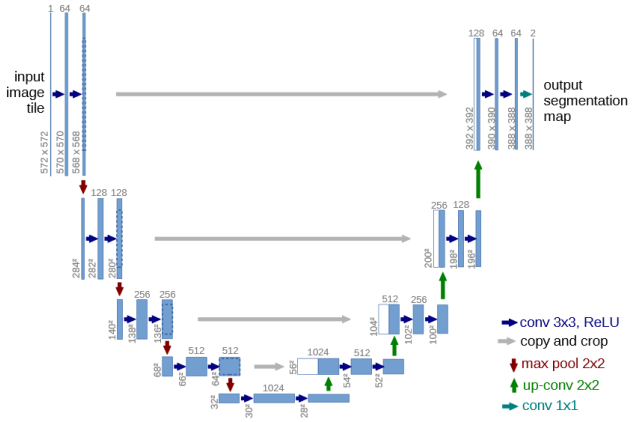
而任务则是通过传统算法或机器学习算法对两个数据集的测试集进行像素级的细胞实例分割，将每个细胞分为一类，并以 jaccard 相似度作为最终的评价指标。

1.2 语义分割并后处理

实例分割问题可以拆解成语义分割和后处理的串行处理，也即首先通过语义分割网络，将图像分为前景和背景，前景部分即为所有细胞，而背景部分为除细胞外的部分。

1.2.1 U-net,U2-net

其中的语义分割方法我查阅到的最主流的网络即为 U-net[?]，其被广泛地应用于医学图像分割当中，网络结构如下图所示:



其主要通过一个 Decoder 和 Encoder 的 U 型结构来进行分割，其中的 Encoder 部分通过不断地卷积和下采样进行**特征提取**，而 Decoder 部分则是通过上采样，将特征映射至分割的 ground truth。

此外，我查阅到了今年的 CVPR 论文,U2-net[1]，其同样是用于像素级语义分割的网络结构，在这篇文章中，作者提出了网络结构可以使得网络结构更深并且不占用大量的显存，并且提出了一种 U 性的残差 block，用于改善卷积过程中的特征学习，整体网络结构如下图 [??]:

在上图的网络中，其编码部分均使用了其中提出的 U 型残差 block 结构，其经过六个阶段的编码和译码，对每一个阶段的译码结果进行融合，其在每一个 stage 输出的为前景的概率，则将六张结果进行融合后作为输出。其基本结构类似于 Unet 而比 Unet 更深并且经过了融合的输出，在本次作业中也证实了其分割效果要优于 Unet。

1.2.2 后处理方案

在后处理部分主要针对已经分割后的二值图进行每个细胞的划分，在这时仅需要将不同的细胞划分为不同的类别，据此我经过思考与查阅，可尝试的方案共有如下三种:

聚类法

[1]U2-Net:Going deeper with nested U-structure for salient object detection.CVPR 2020:Accepted 28 April 2020

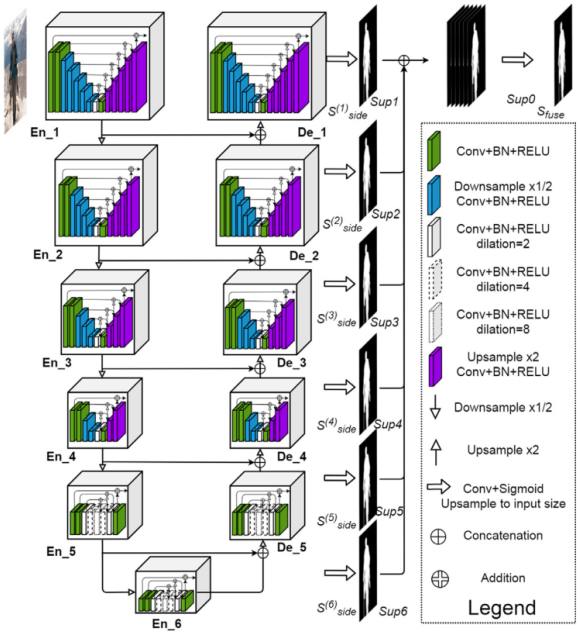


图 1: U2-net

对于二值图，由于细胞本身属于球形，因此可以使用聚类的方法对于细胞前景的坐标进行聚类，但需要解决的问题为确定聚类数，在此可通过 kmeans 的肘部法或基于密度峰值[2][?] 的方法进行确定。但经过检验，肘部法效果极差。

连通域提取

假设语义分割能够将大部分不粘连的细胞分割开，那么几乎每一个连通域便是一类细胞，因此可以通过连通域提取将每一个连通域划分为一个类别，这种后处理方法对于语义分割的结果要求较高。

分水岭

分水岭法同样用于分割，其通过模拟泛洪自上而下对灰度图漫延的过程对原图像进行分割，但其对于噪声敏感，极易造成过分割，实际分割过程中需要对过分割问题进行处理。

1.3 实例分割网络

在考虑数据集 2 这种细胞粘连的情况时，我同样去调研了实例分割网络，例如直接进行实例分割的网络，其典型的代表为 Mask-RCNN[?]，是何恺明团队在 ICCV2017 获得的最佳论文，其主要的分割框架如下图所示:

如上图所示，mask rcnn 的处理思路分为两个阶段，在第一个阶段内通过扫描图像生成候选区域，第二个阶段将候选区域分类并生成边界框和二值的掩码 mask。通过两个阶段对于每一个目标进行分割并分类，从而实现实例分割。其同样可以迁移至细胞实例分割任务中对每个细胞进

[2]聚类部分课件中第 31 页

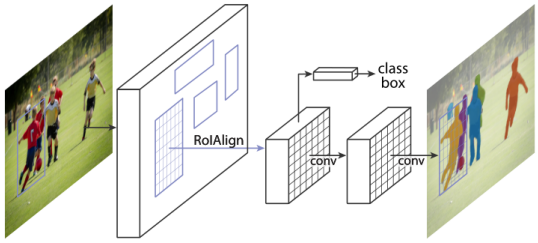


图 2: Mask-Rcnn

行探测并进行分类。

1.4 方法比较与选用

在上述的两大类方法中实例分割相当于综合了语义分割与后处理，我的理解是在例如 mask rcnn 的实例分割网络中，其同时包含了语义分割和后处理，其在候选区域的探测阶段实际上以及学到了包含语义分割的语义特征，而后处理同样通过深度的网络结果进行参数拟合，利用网络结构对学到的候选区域进行分类。而语义分割加后处理方法，在后处理时往往会选用一些数字图像处理的传统算法来进行分类。我认为基于此数据集，其单个细胞的可分开性较强，由于其几何特征较为简单，因此在后处理阶段应当可以使用传统算法来完成，这也是之后我在两个数据集均选用了语义分割和后处理。

而在语义分割我所调研的两个结构之中，U2-net 基于了残差结构并且多阶段的分割结果进行了融合，网络结构更深，因此我也认为在之后的语义分割中应该 u2net 要更为出色，在经过试验后也证实了这一点。

在后处理方法中，三者各有优缺点。使用聚类算法对于圆形的细胞是易于聚类的，但是其聚类数确定起来相对较为困难，若聚类数确定的结果和实际结果出现偏差，则会导致后处理效果极差。连通域处理对于语义分割的结果要求较高，需要细胞不粘连，而使用分水岭则对于噪声较为敏感，极其容易出现过分割。而在本次大作业的完成过程中，我对于上述的三种方法均有尝试。

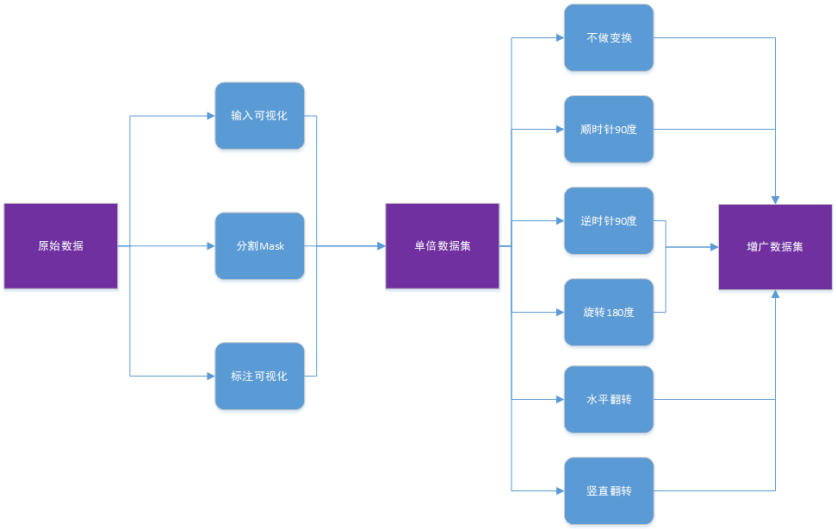
2 数据处理流程

在本小节我将分别对我在两个数据集上的数据处理流程进行简述，并分析其处理的原因。在两个数据集上，由于我均是使用的语义分割和后处理的结构，因此我将在此分别说明两个数据集的预处理和后处理流程。

2.1 数据集一

2.1.1 预处理

数据集一给了充足的标注数据，但数据量较少, 此外，由于 u2net 的输入数据和标签的数据类型是和 unet 相同的，因此我需要对标注图像将其转换为二值的 mask 进行输入，整个数据预处理流程如下图所示:



也即首先将原始数据集进行可视化，并且将标注数据可视化，此外将标注数据二值化，这一步也较为简单，只需将所有的标注细胞设置为 255，而背景部分保存为 0 即可实现。由此可以获取这三组图像的单倍的数据集，为了进行扩增，我将图像依次进行了: 顺时针旋转 90°，逆时针旋转 90°，水平翻转，竖直翻转，以及水平竖直翻转 (旋转 180°)。由此可以得到 1050 张训练数据和训练标签，输入网络中进行训练。

在后处理的部分，我首先说明我的网络输出为前景概率，我没有直接在输出时进行分割，而是将概率作为灰度图存下[3], 因此后处理包含了以下的两项内容: 阈值分割与连通域提取，处理流程如下所示:

首先根据得到的灰度图选定合适的阈值，由于我使用的训练代数较多，因此也选择了比 0.5 更高的阈值，目的是去除一些细胞间不必要的粘连，能够更好地将不同的细胞分至不同的连通区

[3]在实现过程中会详细说明

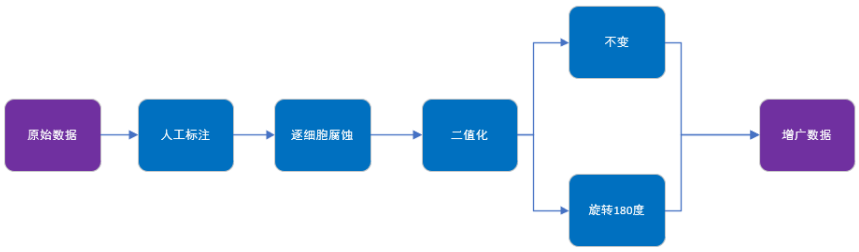
域。在二值化之后进行连通域提取，将每一个连通域作为一个细胞的类别，将结果保存为 uint16 的格式便完成了后处理。

2.2 数据集二

数据集二则具有更少的标注，并且细胞之间有大量的粘连，因此我对于数据集的增广方法首先是进行标注[4], 在完成标注后，我便进行后面的预处理。

在后面的预处理中，我的基本思路是想要类似于数据集一将不同的细胞分开，而我们的标注数据大部分细胞是有所粘连的，但是标注数据中已经标出了每一个不同的细胞便于提取，因此我的基本想法是对于每一张图片，单独拿出其每一个细胞进行一个较大的腐蚀，将所有腐蚀后的细胞重新汇集在同一张图上，并二值化为前景图，依次将标注信息分开，标注信息被腐蚀为“内核”，具体如下图所示:

左图是对标注信息直接二值化的结果，而右图是逐个细胞进行腐蚀后得到的内核部分的结果，可以看到右侧图已经将粘连细胞完全分开，因此如果能学习到完全分开的内核，在后处理阶段划分标签后进行逐个膨胀即可。我的预处理流程如下图所示:



在这里由于上述的各个同学的标注工作得到的风格有所不同，因此我挑选了部分标注结果作为数据集进行训练，这部分数据共有 151 张，对于挑选的部分数据进行增广，最终我所使用的增广形式为原数据以及原数据顺时针旋转 90° 之后所组成的共 302 张数据放入之后的 u2net 进行训练。

[4]标注数据的工作和其他同学共同完成，因此我们使用了相似的标签进行训练，他们分别是蔡烨怡 (2017010922) 程晔安 (2017011627) 吕康晨 (2017011532) 吴文绪 (2017010910)

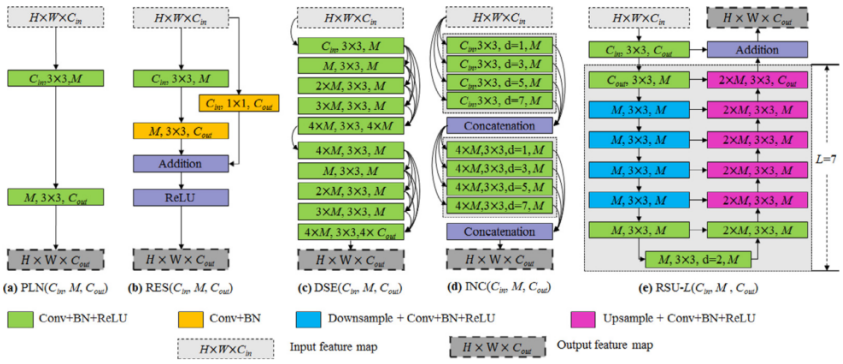
后处理 在后处理阶段同样与数据集一相似，我选择连通域方法对学习到的分开的“内核”进行分类，在获取到每一类的内核后，那么单独对每一个内核进行多次小范围膨胀，弥补孔洞并且将原内核恢复至细胞的大小，也即通过膨胀为每一个细胞增加其自己的轮廓，这样对每一类分别膨胀能够保证在膨胀后不会因为发生粘连而无法将不同的细胞分成不同的类别。经检验，这种后处理结合预处理的方法是最有效的。

3 算法原理

在本次的作业中，我对于两个数据集均采用了语义分割网络 u2-net 来完成，两个数据集在算法流程的区别之处为预处理和后处理阶段，也即在监督信息上有少许的区别。在本小节中，我将仔细说明 u-2net 的网络结构和算法的原理。并给出相应的在后处理和预处理阶段用到的处理算法的流程。

3.1 u-2-net 结构

在文献调研中已经提到，该网路整体上为 U 型结构，并在不同层次提取到的特征进行了整合，此外，也在不同的核大小下进行了特征的提取，其中如图 [??] 中的结果，其中的每一个 block 在都是文章中提到的一种 U 型结构，具体结构如下图所示:



在上图中的最右侧为作者所提出的 U 型网络结构，这个结构是十分类似于 Unet 的，其中的绿色框表示卷积层，而蓝色框表示下采样并卷积的部分，紫色框表示上采样并卷积的部分，在此图中 L 表示该模块的深度，其整体上同样为一个编码译码的结构来提取特征，在论文中使用的输入为 (288×288)，C_{in} 为输入通道数，C_{out} 为输出通道数，M 为中间通道数，因此我将该 L 为 7 的 block 的特征提取流程列表 (仅 encoder 部分) 如下, 其同样也是在网络输入的第一个 block，因此输入通道数为 3:

Part	Layer	Output
RSU	Input	$(288 \times 288 \times 3)$
	Conv+BN+ReLU	$(288 \times 288 \times C_{out})$
	Conv+BN+ReLU	$(144 \times 144 \times M)$
	(DownSample+Conv+BN+ReLU)	$(72 \times 72 \times M)$
	(DownSample+Conv+BN+ReLU)	$(36 \times 36 \times M)$
	(DownSample+Conv+BN+ReLU)	$(18 \times 18 \times M)$
	(DownSample+Conv+BN+ReLU)	$(9 \times 9 \times M)$
	Conv+BN+ReLU	$(9 \times 9 \times M)$

将其译码部分列出，如下表所示:

Part	Layer	Output
RSU	Input	$(9 \times 9 \times 3)$
	Conv+BN+ReLU	$(9 \times 9 \times C_{out})$
	Conv+BN+ReLU	$(9 \times 9 \times C_{out})$
	(UpSample+Conv+BN+ReLU)	$(18 \times 18 \times M)$
	(UpSample+Conv+BN+ReLU)	$(36 \times 36 \times M)$
	(UpSample+Conv+BN+ReLU)	$(72 \times 72 \times M)$
	(UpSample+Conv+BN+ReLU)	$(144 \times 144 \times M)$
	(UpSample+Conv+BN+ReLU)	$(288 \times 288 \times C_{out})$
	Conv+BN+ReLU	$(288 \times 288 \times C_{out})$

上表列出了在其中经过每一层之后的特征图的大小的变化，文章中称该模块为 RSU-L，而对应图 [**??**] 的每一个 level 均使用了此模块，对应上图将网络整体中的各个模块参数列出:

stage	En_1	En_2	En_3	En_4	En_5
block	RSU-7	RSU-6	RSU-5	RSU-4	RSU-4F
C_{in}	3	64	128	256	512
M	32	32	64	128	256
C_{out}	64	128	256	512	512
En_6	De_5	De_4	De_3	De_2	De_1
RSU-4F	RSU-4F	RSU-4	RSU-5	RSU-6	RSU-7
512	1024	1024	512	256	128
256	256	128	64	32	16
512	512	256	128	64	64

其中的 RSU-4F 表示空洞卷积，扩大了感受视野，上述网络结构在每一个译码网络后对应得到前景的概率。而 u2net 的结构便是通过这六个概率图进行预测，在实际使用时，我在作者给出

的 github 链接上[5]的代码基础上进行修改，在我们的数据集上成功运行。而输出时，我使用了最顶层预测得到的概率作为最终的结果，训练时将这六个结果计算至损失函数中反向传播。以上便是语义分割部分的网络结构原理，由于在作业的最终提交我均采用的此网络，因此不再仔细地分析 unet 的流程。

此外在此算法的细节部分，我列举如下:

- 数据增强: 放缩至 $(320 \times 320 \times 3)$, 随机裁剪为 $(288 \times 288 \times 3)$.(论文原意)
- 超参数: 包括 batch 大小，训练代数，优化器学习率等等。
- 损失函数选择了 BCE 损失，优化器使用 Adam。

而这部分具体的执行步骤便是通过前向传播和反向传播不断更像网络参数，直至达到很好的拟合效果。

3.2 后处理和预处理的部分算法流程

在之前已经提到了我的预处理和后处理的步骤，由于在数据集二的预处理和后处理都是包含了对数据集一的处理，因此我仅简要描述数据集二上的算法流程。

预处理时前面也已经提到，我将标注信息进行了修改，提取到其内核部分进行学习, 实际上是对标注的特征进行了增强，这里的算法也较为简单，我使用简单的伪代码描述如下:

Algorithm 1: 标注图像增强

Initialization: the input uint8 man seg mask,$\forall mask \in Dataset$

// 每幅图像

forall $k = 1, 2, \ldots, len(Dataset)$ **do**

  $mask \leftarrow masks(k)$

  $newmask \leftarrow zeros(shape(mask))$

  // 各个细胞单独腐蚀

  forall *cell = 1,2,…,numofcells in mask* **do**

    $temp\_image \leftarrow mask(mask == cell)$

    $kernel \leftarrow square(20)$

    $temp\_image \leftarrow erosion(temp\_image, kernel)$

    $newmask \leftarrow newmask + temp\_image$

  end

end

上述算法原理便是对于每一个图像，将其每一个细胞单独提取出来，作为一个新的图像的前景，将其腐蚀后并粘贴在新的标注的相应位置，这里腐蚀使用的形态结构为正方形且边长为 20，

[5] https://github.com/NathanUA/U-2-Net

这是为了保证将所有细胞分开，由于腐蚀和膨胀对于细胞边界而言近似可逆的，因此在后处理阶段膨胀即可。

后处理阶段进行膨胀的算法几乎与上述的过程相同，并且在第 (??) 节中已经叙述，因此在此不再赘述。上述预处理算法中所包含的变量为: 输入实例标注，单细胞区域，输出二值标注。

此外，需要说明的是，在后处理阶段我使用的连通域提取算法是基于 cv2 中封装的函数，我同样也尝试了许多其他的后处理算法，但其效果不太理想也没有作为我在网站提交的结果，这些算法我将一并放入之后的第 (??) 节中进行详细说明。

4 实现过程

整体上的实现过程主要分为三部分: 预处理，语义分割，后处理。最终得到测试集的实例分割结果，在本节，我首先说明如何运行文件以能够实现结果复现，其次给出在两个数据集整体的实现流程图。

4.1 实现方式

首先，我使用的代码参考于论文原文中作者给出的代码，在其代码上进行修改使其能够在我的数据集上运行，并且提交的最高结果也是使用其 loader 和 train 部分的代码运行加上我的后处理得到的结果。而后我按照其原意将 dataloader 部分进行了重构，以及 train 部分，并增加了一些评价指标的实现，例如在 train 部分加入了 iou 和 dice coff 的计算，依次记录运行的结果。因此，我所上交的代码包含我在作者代码修改后的代码，以及我对于其进行重构后的代码，由于我对于重构的代码仅运行了少量的迭代次数进行测试，则完整的复现流程仍需运行在作者代码修改后的代码。

从预处理至后处理，实现的基本过程如下所示:

- 运行 generate_data1/2.py: 该模型的功能是对原始的.tif 数据转换为 uint8 进行保存，并且对原始的标注图片便为二值图片，并且进行一定的数据增广后扩充数据集，数据集一而言，原始数据集共有 175 张图像，扩充后共有 1050 张图像。而在运行 generate_data2.py 运行时，则将原有的 151 张图像进行二值分割，特征增强和数据增广，最终得到 302 张训练数据。

- 运行 traindata1/traindata2.py: 该部分代码是对数据集 1 和数据集 2 的训练部分代码，其中的超参数如果需要修改，则在代码部分中可修改超参数的默认值。该部分是对于两个数据集的训练过程，输出为训练过程的日志文件，以及训练过程的 loss 和 iou 以及 dice 的过程值，并每隔一定迭代次数保存的模型。

- 根据训练结果挑选模型。找到合适的模型，并将模型的名称在预测部分代码进行修改。而如果需要复现结果，则可使用我所提交的两个模型，这两个模型在经过后处理后达到了我

在排行榜的最高得分。

- 运行 preddata1/preddata2.py: 该部分我也是将数据集一和数据集二的代码分开，这部分的功能是对测试集进行预测，并且将预测的前景概率转化为 3 通道的彩色图，其可视化是类似于一张灰度图，这是因为在之后的二值化过程中，方便修改分割阈值而不是严格按照概率大于 0.5 划分前景和背景。但两组代码几乎相同，仅是使用的保存模型以及测试集路径和保存路径有所不同。其输出为预测结果的灰度图像，保存至本地即可。

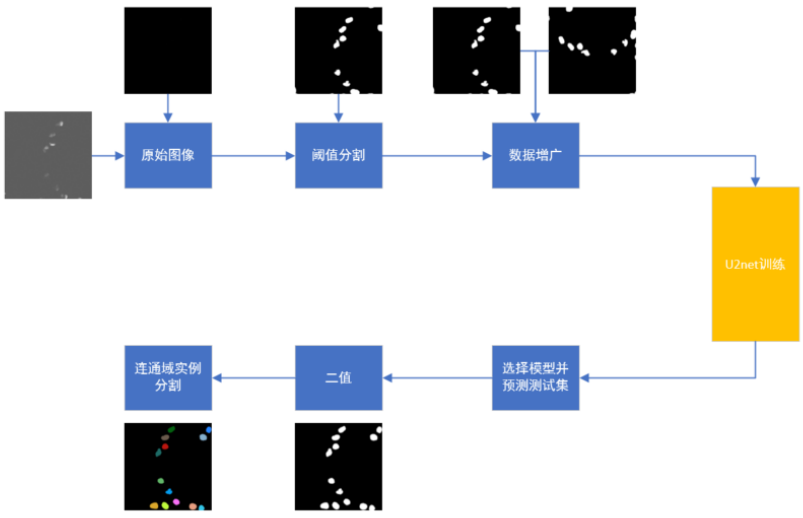
- 运行 handle1/handle2.py: 该部分代码为数据后处理。原理在之前已经说明，该部分的输出为两个文件夹，其中一个文件夹包含了实例分割后的可视化图像，另一个文件夹则是将实例分割后的保存为 uint16 的文件。

经过上述的运行过程，便可以完整地对训练集进行一个合理的分割预测，上述为具体的代码运行方式，接下来我会对两组数据集的实际运行流程进行可视化的说明。

此外，需要注意的是，由于代码中所需数据的路径，我在大作业完成期间是将其放在服务器上运行，因此使用了服务器的绝对路径，在本次提交的作业中，我将路径改为了在压缩包中数据的相对路径。如需复现，请注意路径的修改问题。有关各个文件的存放目录，我会在 README 文件中详细说明。

4.2 数据集一的整体实现流程

对数据集一，我绘制了整体的处理流程如下所示:

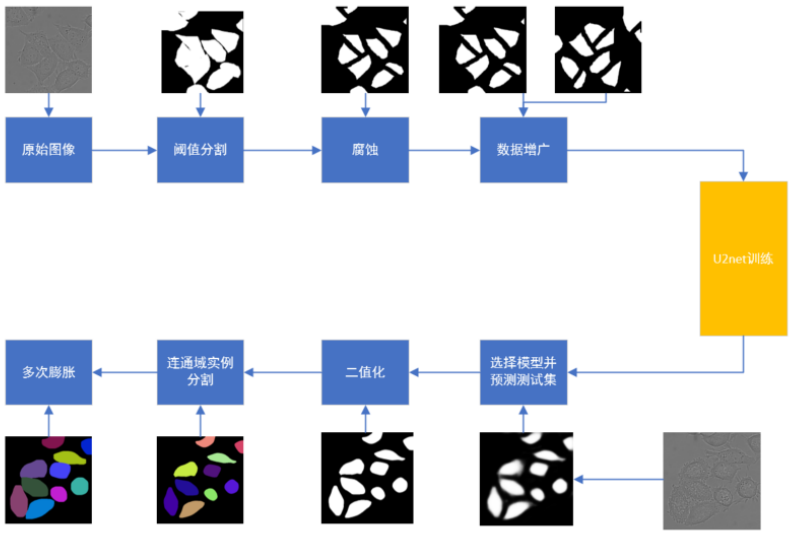


其中上半部分为预处理部分，右侧黄色框代表深度神经网络学习的模块，而下半部分则为后处理部分，其中模块附近的图像为其处理阶段的代表，上半部分我选择了训练集第一张贴出，下

半部分后处理我选择了测试集第一张贴出。整体的实现流程如上图，而每一个模块所需要运行的代码部分也在上一小节给出，因此不再赘述。

4.3 数据集二的整体实现流程

数据集二的预处理部分增加了腐蚀的部分，在后处理也相应的增加了膨胀的部分，整体处理流程绘制如下图所示:



上图清晰地展示了从训练集至测试集的分割过程，以及特征增强和后处理的过程，同样在上半部分表示了预处理部分，下半部分为后处理部分，也能够看出，预处理和后处理多出的环节为腐蚀和膨胀，但这一简要的处理能够大幅提升分割的结果。

5 实验结果与模型性能分析

在本小节中我将首先说明在两个数据集上的我的实验结果，其次对于模型不同的参数以及不同的处理方法进行横向的对比分析。

5.1 模型结果

在本次的预测结果中，语义分割的结果对于最终的实例分割的结果尤为重要，因此我也将语义分割结果和后处理的结果分开介绍，但着重介绍语义分割的结果。

在本模型中使用的损失函数为 BCE loss，其函数表达式为:

$$loss(x_i, y_i) = -w_i[y_i log x_i + (1 - y_i) log(1 - x_i)] \tag{1}$$

在此训练过程中，根据之前提到的网络结构，u2net 的六层 level 会分别输出六个概率输出，以及一个单独的将其汇总后的概率预测输出，对于这七个输出分别计算其 bce loss 将其汇总相加得到最终的 loss，最终在 dataset1 上经过 106000 次个 batch 的迭代，得到了 bce loss 为 0.253333，其中对于第二层的输出其 loss 为 0.008061，对于数据集二，我最终提交的结果使用了 1800 次迭代的输出结果，其合成 bce loss 为 1.586861，同样，第二层单独的 bce loss 为 0.206399。

在我提交的最高结果中，其中使用到的语义分割结果，我在训练集上评估其最终的二分类 iou 和 dice 系数，得到的结果为:

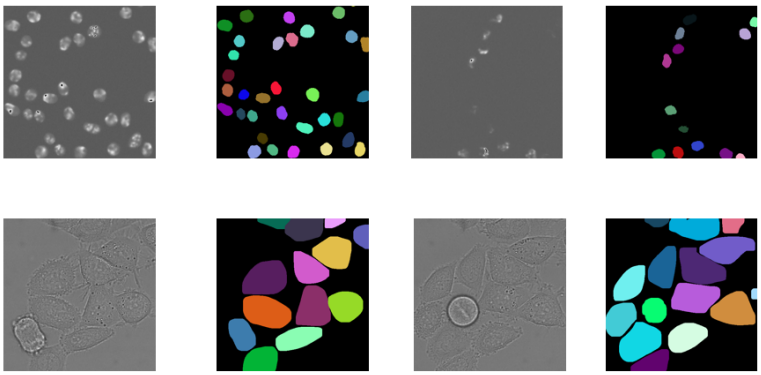
数据集	bce loss	Iou	Dice
dataset1	0.253333	0.87563	0.98107
dataset2	1.586861	0.77498	0.87200

其中的 dice 系数和 Iou 的计算公式分别为:

$$dice = \frac{2|X \cap Y|}{|X| + |Y|}, Iou = \frac{|X \cap Y|}{|X \cup Y|} \tag{2}$$

上述结果均为我所使用的的提交的模型，在整体训练集得到的平均结果，可以看到 dataset2 的二分类的 dice 和 iou 并不是极高，这是由于我所迭代的次数较少的原因，倘若迭代次数足够多，那么在训练集上的效果会更好。但是考虑到人工标注的差异性以及该监督信号为腐蚀后的内核，因此在此部分仅需将各个细胞分割开即可，无需使得训练集上有着极高的拟合效果。

在后处理后，经过在网页中提交的测试，这两组数据集分别取得了 0.835515、0.854498 的 jacaard 相似度得分。列举一组可视化的分割结果:



从上图可以看出，分割结果较为良好，在数据集一中大部分的细胞均被分开，在数据集二中通过分割后的膨胀在测试集也取得了较好的效果，并且可以发现对每个细胞几乎都能过通过语义分割找出，不存在漏细胞的情况。

5.2 不同模型参数对比分析

在本小节内，我主要分析语义分割网络的性能，对比几组不同的参数下的 loss，该 loss 记录的为第二层输出的单独的 loss2，以此层的 bce loss 当做观察指标，iou 以及 dice 系数三个评估

指标。在这里我首先说明在两组数据集上使用的一些参数:

参数	batch size	优化器	学习率	输入大小
值	12	Adam	0.001	(288×288)

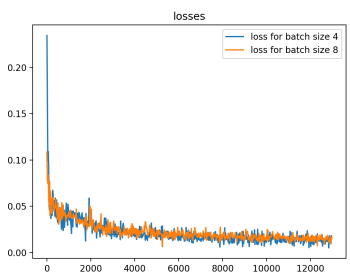
以下的对比参数实验均是单独参数在基于以上的参数进行的修改。

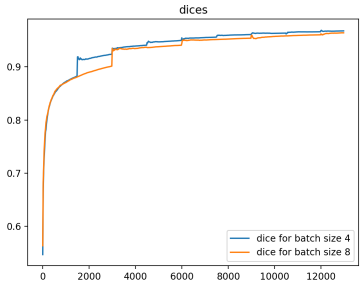
5.3 改变 batch

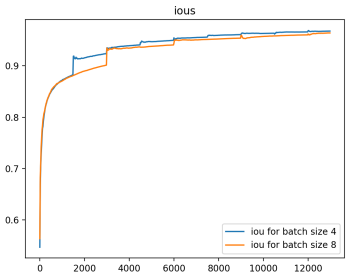
对于训练 batch，当 batch 较大时，能够更快速的训练，但同样会占用更多的显存，在这里我为了对比实验，在两个数据集上分别跑了 100 代，将其每隔 20 个 batch 记录一次评估指标的结果。

5.3.1 数据集一

在数据集一上的结果如下所示:



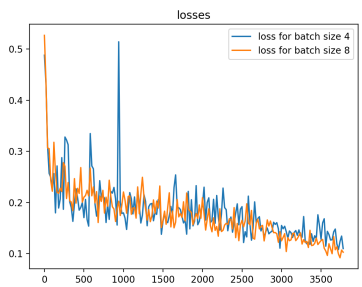


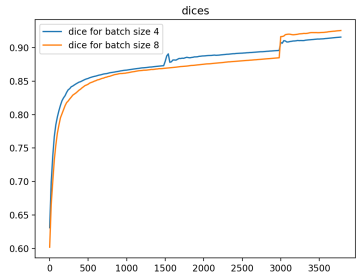


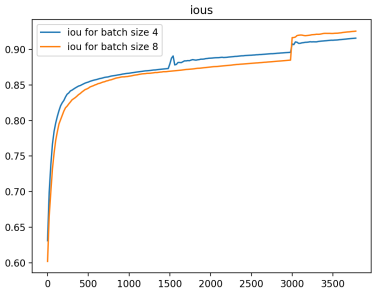
在数据集一上通过改变 batch size 分别为 8 和 4 的结果可以看到，在三种评价指标上两者几乎没有差别，其中在后两个指标上的出现突变是因为在每 3000 次迭代的过程中，我动态更新其平均值作为记录，因此在 3000 次的整数倍会出现向上的突变。可以看到在此参数下改变 batch 大小，当 batch 为 4 时会略强于 batch 为 8 的结果。

5.3.2 数据集二

在数据集二上同样原理绘制图像，得到的结果如下图所示:







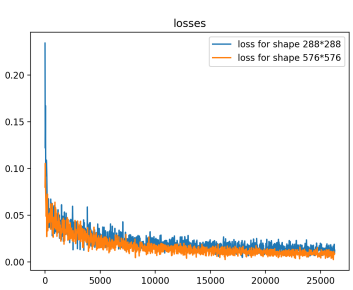
可以看到在数据集二上，增大 batch 能够有少许的提升，后两幅图应当着重看突变后的值，因此这是从 3000 次迭代后开始计算的，更加能说明随着迭代次数的增加，在 iou 和 dice 系数上都是大一些的 batch 能够取得更好的效果，并且可以看到 loss 也是在 batch 为 8 时会更小，因此认为可能在数据集二大 batch 有助于语义特征的学习。

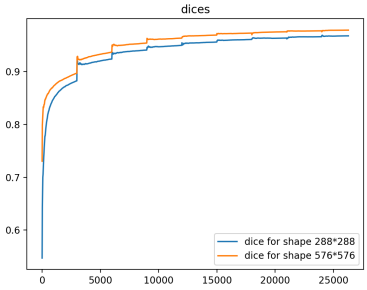
5.4 改变数据输入尺寸

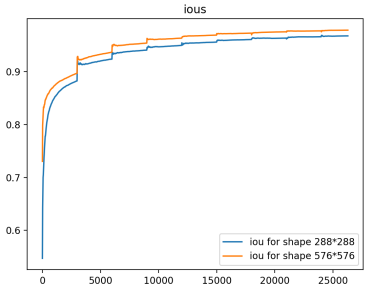
在本小节通过改变数据增强方法进行对比分析，原有数据增强方法均为放缩至 320 后随机裁剪至 288, 我在本小节内仅对此部分进行修改，对比分析实验结果。

5.4.1 数据集一

在数据集一中，我想若放缩至更大的图像大小，可能会包含更多的语义信息，从而得到更好的语义分割结果。但同时更大的图像输入尺寸，需要消耗更多的计算资源，也会占用更大的显存。因此我将其放大一倍，放缩至 640×640 后随机裁剪至 576×576, 得到的运行结果对比如下所示:



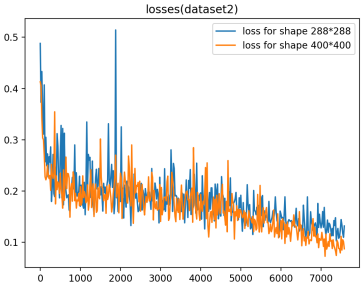


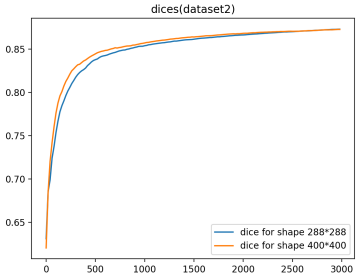


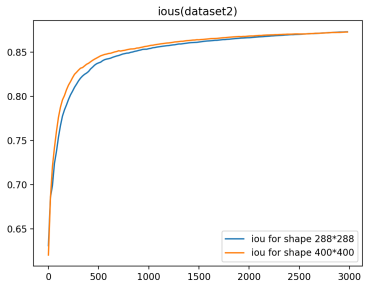
由上图可知，在输入图像尺寸更大时，训练集 loss 明显会更低，并且在 dice 系数和 iou 上能够看出两者具有一个较为明显的差别，因此验证了我在上面的猜想，当输入较大的尺寸对数据集一会带来分割性能的提升。

5.4.2 数据集二

数据集二由于原始图像的大小为 500×500, 因此对照组仍选用 288×288, 而实验组我将其放缩至 450×450 后，随机裁剪到 400×400 的大小进行输入，得到的结果如下所示:







在上图的右侧两图中，我仅比较了 3000 次迭代以内的结果，从总体上看，loss 上大尺寸输入的图像具有稍明显的优势，而在 dice 系数和 iou 上，没有明显的提升，并且在最终两者几乎接近一致。因此对于语义分割的性能从后两个评估指标而言在迭代次数较少时有一些提升，当迭代次数较多时这种提升则较少。

通过以上的对比，我认为如果计算资源足够时，可适当增加图像的输入尺寸以保留更多的语义信息，可以对最终的分割效果带来少量的提升。若计算资源不足的情况，则无需在此方面进行改进，这是因为效果的提升有限，但耗费的计算资源以及耗时都会明显增加。

6 其余算法尝试

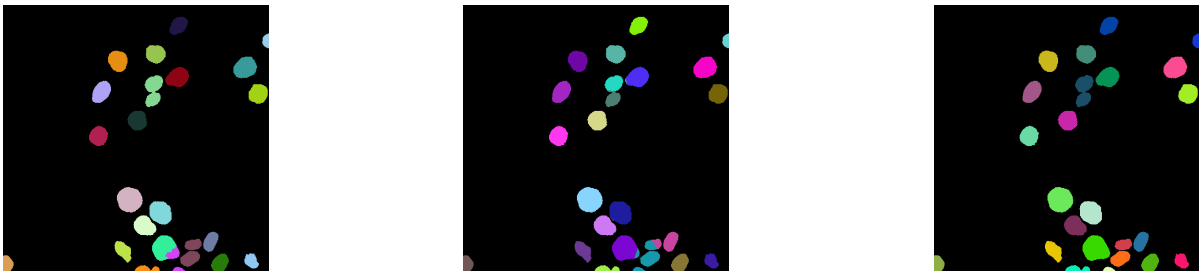
在本小节我将简要说明一些我在本次大作业的一些其他算法尝试，以及其为什么没有被采用的原因。

6.1 不同后处理算法尝试

在本小节内我主要对比一些不同的后处理算法，由于在测试集无法自己计算出 jaccard 系数，并且并未将所有的后处理结果进行提交，因此仅一些肉眼可见的结果作为对比

6.1.1 数据集一

在数据集一上除了最终采用的连通域划分外，我还尝试了使用聚类方法将语义分割结果进行实例分割，具体采用了 kmeans 和 Birch 聚类。但我使用肘部法往往确定出的聚类数和实际细胞数相差较大，经常在 k=5 以内出现肘部，因此并不适用。为了对比分析，我在一次语义分割结果手动数出了一些细胞数量，在另一次语义分割的结果应用这些数量作为聚类数，则对比一张图像的三种后处理如下所示:



从左至右依次是 Birch，Kmeans 和连通域提取的方法，从左侧两图可以看出，其出现了过分割的现象，这是因为当给出的聚类数多于实际数目时，便会出现过分割，即使在左侧图中粘连细胞仍没有被分开。为了分割的鲁棒性，应当选择类似于右图的连通域提取，若结果粘连过多，同样可以采取应用在数据集二的预处理方法，但从结果来看，仅有少量图像具有粘连，因此无需调整。

6.1.2 数据集二

在数据集二上我同样尝试了不同的算法，分别是分水岭和连通域提取，以测试集第五章图像作为示例，我分别贴出未在预处理加入腐蚀，在预处理加入腐蚀后分水岭和在预处理加入腐蚀后连通域提取的算法，结果如下:



上图所示的分水岭算法均是使用了在阈值分割得到的二值图上基于距离的分水岭算法，可以看到, 最左侧不仅细胞大量粘连，而且使用分水岭方法出现过分割，也不能有效地将各个细胞区分开。中间图像由于对造成敏感，因此出现了少量的过分割现象。但通过中间的图像也不难发现，可直接使用连通域提取获得更好的结果，因此最终采用连通域提取也是基于各个细胞都区分开的基础。

6.2 网络尝试

起初，我尝试的网络结构为经典的图像分割网络 unet，但是经过训练后发现其语义分割的效果并不是十分理想，我在 dataset1 的验证集上得到的最高的 dice 系数也为 0.8 左右，这和我在 u2net 中得到的效果相差很多。包括我在测试阶段和最初的提交都是基于 unet 进行训练的，在调试参数过后的提升也不明显。其在网页上提交的结果，jaccard 最高在 0.67 左右，由于其并未将连通域分开，因此连通域提取的方法也并不适合。因此我也进一步去调研文献，查找其他的语义分割网络，从而得到我现在所使用的网络结构。

7 总结

通过完成本次细胞分割的课程大作业，我对细胞分割领域有了最初的尝试，也通过文献调研使得我对这一领域的一些基本处理方法有了整体的了解。在完成的过程中，我进行了各种尝试，在起初也遇到了瓶颈，包括在数据集二上没有想出有效地办法时也在考虑是否应换用实例分割网络。但通过助教和同学们在课程群里的讨论，对我有很大的帮助，同时我也和一些同学交流探讨自己的思路，推动我在认知实例分割并想出有效的办法的道路上前行。原先我只接触过图像分类任务，以及通过无监督学习对细胞完成前景和背景的分离，这是我第一次接触实例分割任务，总而言之，能在一段较短的时间内通过大量查阅资料，使我对全新的领域具有了一定的了解，并且尝试后还取得了不错的效果。

总体上，这次大作业不仅使我的调试网络的能力和数字图像形态学处理的能力进一步巩固，也使我查找文献，阅读并理解文献的能力有所提高。作为这门课程的一个总结性的课程作业，从最初的线性分类器的原理到期末时的课程学习大作业，总体上让我对机器学习和模式识别领域有了足够的认识和浓厚的兴趣，想必会对我日后的科研道理有所帮助。最后，感谢助教和老师们一学期以来的付出!

A 附录

A.1 代码来源说明

本次语义分割网络部分的代码来源为参考文献中的论文作者给出的开源代码，代码地址为:https://github.com/NathanUA/U-2-Net

我基于上述代码进行了修改，加入了一些评估指标的计算。此外，我也将部分代码进行了重构，放入 my_implement_test 文件夹中，但此重构代码仅经过少量的训练进行测试，若需复现，仍需运行我修改后的作者代码。

A.2 数据集二数据标注说明

数据集二我选择了人工标注训练集，由于标注任务量较大，因此我和几位同学合作对**部分训练集**进行了人工标注，并共同得到一份标注结果，除了我自己标注的数据外，我还使用到了这些同学的标注信息: 蔡烨怡 (2017010922) 程晔安 (2017011627) 吕康晨 (2017011532) 吴文绪 (2017010910)