**模式识别与机器学习大作业 细胞图像识别与分割**

门恺文 自83 2018011501

**摘 要**：本次大作业面临的问题是对未标注的细胞图像运用机器学习的方法进行分割标注。其中dataset1对应着第一个任务，训练集标注比较充分，且细胞相对较小、较稀疏，可以使用传统方法或深度学习方法；dataset2对应着第二个任务，训练集的标注较少，且细胞大而紧密，不容易分开，使用深度学习的方法会得到较好的结果。

代码在tensorflow.keras框架下编写和运行，采用语义分割加后处理的方法。对于第一个训练集，搭建Unet卷积神经网络，对原训练图像扭曲、平移得到更多的训练样本，将标签二值化，之后用训练样本对搭建好的Unet网络进行训练，得到训练模型。采用此训练模型对测试集进行预测，得到预测结果。之后对预测结果二值化，并采用DFS给每个细胞不同的标签，即可得到多分类结果。最终结果jaccard相似度为0.640389。对于第二个训练集，采取相似的方法，并进一步调整模型参数，但结果不尽如人意，尝试了Unet论文用到的训练集，结果仍然不好，所以最终采取了dataset1\_demo中的方法进行传统的分割。jaccard相似度为0.06左右。

**关键词：**细胞分割； 深度学习； 卷积神经网络

**1. 文献调研**

对于细胞分割问题，既可以使用传统的方法，也可以使用深度学习的方法。传统的方法主要包括基于阈值的分割、基于边缘的分割、基于区域的分割、基于图论的分割、基于能量泛函的分割[1]、基于聚类的分割[2]。深度学习方法主要包括滑窗法、Unet等[3]。

传统方法中，基于阈值的分割包括直方图双峰法、固定阈值分割、半阈值分割、迭代阈值图像分割和自适应阈值图像分割[4]。直方图双峰法是典型的单域值分割方法，该方法依托于图像中有明显的背景前景，寻找灰度图双峰之间的谷值作为阈值，而本次作业中细胞图像前景与背景不是很分明，所以这种方法不适合。固定阈值分割是选定一个阈值，大于该阈值的设为一个标签，小于该阈值的设为另一种标签。半阈值分割是一种增强边缘的算法，把图象分割为N等份，计算每一份中前景点的数量，当前景点达到一定数量时，进行阈值分割，否则不做处理，通过此方法可以使图像边沿分明[5]。迭代阈值图像分割是采用阈值逼近的方法进行阈值选取。自适应阈值图像分割应用于图像背景亮度不同时的自适应调整[4]。

基于边缘的分割方法是通过图像不同区域边界线上的像素值突变实现的。一般的，像素值的突变包括阶跃型和屋顶型，边缘角点和兴趣点的检测器包括Canny边缘检测器、Harris角点检测器、SIFT检测器、SURF检测器。这个在此次大作业中可以用于将预测出图像的边缘分开。

基于区域的分割是按照图像的相似性准则划分为不同的区域，包括种子区域生长法、区域分裂合并法、分水岭法。种子区域生长法根据像素的相似性聚集像素点来达到区域生长。区域分裂合并法是将图像分为若干互不相交的区域，之后按照一定的规则分裂合并。分水岭法对微弱的边缘有着良好的响应，但图像中的噪声会使分水岭算法过分割。[1]

基于图论的分割方法采用了图割的方法，首先根据一个图的前景和背景训练贝叶斯分类器，之后根据节点建立起一个图，并以预测出的两点间的权重添加边，之后对图分割。[2]

基于能量泛函的分割主要是活动轮廓模型及发展起来的算法，而活动模型分为参数活动轮廓模型和几何活动轮廓模型。这种方法通过求解能量泛函的最小值找到目标的轮廓，实现图像分割。

基于聚类的分割是采用KMeans等聚类算法，将图像中像素一致的相近的点聚为一类，之后通过二值化可以达到分割的目的。[2]

传统算法在细胞比较稀疏的情况下可以得到较好的分割算法，但在细胞密集的情况下较难分割出细胞。

深度学习领域，滑窗法预测窗重心有着很好的分割效果，可以在训练样本不多的情况下取得精确的、像素级别的分类。然而，这种方法有两个缺点：一是十分慢且有冗余，二是输入尺度和性能之间的矛盾，即感受野和分割精度呈负相关。而基于FCN网络的Unet则可以通过很少的训练样本取得很好的分割效果，采用U形网络，在对称层之间也有连接，充分利用已有的图像，达到很好的分割效果。[6]所以本次作业最终选择了Unet。

**2. 数据处理流程**

**2.1 dataset1**

**2.1.1 训练集**

由于需要扩充训练集，所以对训练集进行数据增强，即通过ImageDataGenerator 所带的 flow\_from\_directory 函数，调整函数的参数，对训练集图像进行平移、旋转、翻转等操作，并合并为一个生成器。为了统一训练样本，并为语义分割做准备，将图像归一化，并把标签二值化。由于本次训练的标签是背景为零，细胞标注有不同的数字，所以二值化的实现设计为原本为0的保持，原本大于零的设为1。同时，为了加快训练速度，预处理阶段将训练集图片大小调整为（256，256）。

**2.1.1 测试集**

测试集数据的处理主要是测试集的读取和调整。测试集本身就是灰度图，所以读取时采用cv2.imread读入原图。在预测过程中发现了预测样本标签和测试集标签对不齐的情况，所以最后采用了按标签顺序读入，而不是直接按文件夹中的文件顺序读入。训练集生成的增强数据由于多了一维batchsize，所以训练集数据大小是2\*256\*256\*1，为了与训练集统一，测试集数据也要相应调整到这个大小，即将测试集数据大小调整为（256，256），并归一化，之后在每个样本前后各增加一列。

**2.2 dataset2**

dataset2最终的实现是采用了传统的分割算法，只涉及测试集的数据预处理。需要对每个图像标准化，即转化成uint8格式，并统一为三通道彩色图像。

**3. 算法原理**

算法包括模型搭建、数据预处理、模型训练和预测、数据后处理四部分。

**3.1 dataset1**

**3.1.1 模型搭建**

采用unet结构，输入为预加载权重和输入数据的大小，输出为Unet模型。包含十层卷积神经网络，最后一层是输出层，是使用sigmoid激活函数的卷积层，实现二分类；第五层是中间的连接层，采用两次卷积，使用relu激活函数，边缘处设为不补零，并添加dropout层避免过拟合；前四层每层均有两次卷积和一次池化，卷积层采用relu激活函数，池化层池化大小均为(2,2)；除输出层外的后四层与前四层一一对应，对上一层进行上采样，并卷积，之后建立与对称层的联系，从而对应到对称的层，之后再经过两次卷积，卷积层的激活函数均设为relu，边缘处均设为不补零。

optimizer采用Adam，输入为学习率Ir，和动量参数beta\_1, beta\_2, epsilon，输出为该优化器。设置参照keras中文文档[7]和自己实际训练模型时得到的训练结果。

对于loss函数，由于dataset1样本背景较多，前景较小，所以需要调整训练时损失函数中前景所占比重，所以loss函数采用focal-loss[8]。输入为前景背景损失代价的指数gamma和平衡参数alpha,输出为focal\_loss\_fixed损失函数。

本次作业中一共使用了三个评价函数，一个是常用的accuracy，另外添加了与focal-loss相对应的dice\_coef。同时，由于jaccard相似度与IoU类似，所以加入了keras自带的MeanIoU。

**3.1.2 数据预处理**

数据预处理包含了三个函数，train\_Generator, adjustData, test\_Generator。前两个是对训练集数据的预处理，后一个是对测试集数据的预处理。

**3.1.2.1 train\_Generator**

用于生成训练集，基本原理是借助数据增强函数ImageDataGenerator.flow\_from\_directory读取训练集并进行数据增强。输入为：

batch\_size：生成增强数据后每次读入的大小；

train\_path：原训练集图像路径；

mask\_path：原训练集标签路径；

image\_folder：图像文件夹名称；

mask\_folder：标签文件夹名称；

aug\_dict：增强参数，包括旋转角度、平移距离等；

image\_color\_mode：生成的图像颜色模式，默认为灰度；

mask\_color\_mode：生成的标签颜色模式，默认为灰度；

image\_save\_prefix：增强后的图像保存前缀；

mask\_save\_prefix：增强后的标签保存前缀；

flag\_multi\_class：是否是多分类，默认为False；

num\_class：分类数，默认为2；

save\_to\_dir：增强后的图片保存路径，默认为None；

target\_size：增强后图片的尺寸，默认为(256, 256)；

seed：数据增强时设置的随机种子，默认为1。

函数输出为训练图像及标签的生成器。

算法变量image\_datagen和mask\_datagen分别为图像和标签的ImageDataGenerator，用于增强数据；image\_generator和mask\_generator分别用于承接图像和标签生成的数据；train\_generator用来将增强后的图片和标签打包。img, mask是两个循环变量，用于循环调整每组图片和标签。算法的具体执行步骤为先生成两个数据增强器，之后通过数据增强器生成增强数据，之后将增强的数据分别循环调整。

**3.1.2.2 adjustData**

训练集循环调整时用到的函数，在二分类时用于把图像归一化，标签二值化。

输入：

img：要调整的图像；

mask：要调整的标签；

flag\_multi\_class：是否进行多分类的标志，此处一致为False；

num\_class：分成几类，此处一致为2.

输出：调整后的图像和标签：img，mask。

该函数中没有其他变量，执行步骤为先对img，mask标准化为uint8格式，之后将img/255归一化，把mask/255后，mask中为0的像素点继续保持，大于0的像素点设为1.

**3.1.2.3 test\_Generator**

读取测试数据，并调整为适合模型的大小。

输入：

test\_path：测试集路径；

target\_size：读入后的调整大小，默认为(256,256)；

flag\_multi\_class：是否是多分类，默认为False；

n：测试数据数量，默认为33.

输出：打包好的测试集。

算法变量i为循环变量，用于循环处理文件夹中的图片；img为每次读入的图片。具体执行步骤为以i为循环，依次读入对应的文件名，将img/255归一化，调整大小为（256，256），在img前后各加一列。

**3.1.3 模型训练与预测**

用预处理后的训练数据训练模型，再用模型预测测试数据。输入为搭建好的模型，预处理后的训练集和测试集，输出为测试集预测结果。

算法变量data\_gen\_args是设置增强时的模型参数，myGene是训练集生成器，model­\_checkpoint用于在过拟合之前及时保存模型，testGene是测试集生成器，results是测试结果。算法具体执行步骤为先得到训练集生成器，然后进行训练，之后用训练好的模型测试，得到测试结果。

**3.1.4 数据后处理**

主要是调整图像大小、二值化、标准化并给每个细胞标上不同的标签。

整体输入为预测结果，输出为后处理后的图像。其中子函数DFS的输入为数组result，两个数组下标ij，颜色color，无输出。子函数IndexInRange输入为下标i，边界bond，默认为628。输出为i是否满足0<= i <bond的布尔变量。

算法中间变量为：循环变量k遍历每个预测结果，循环变量i，j遍历结果的每个像素；大小为256的布尔型数组color\_used用于表示256种颜色的使用情况，避免重复使用；临时变量result是results的第k个元素，方便处理；color是每次遇到没有更新标签的细胞时生成的0-255之间的随机数，该随机数跟着DFS扩展到整个细胞区域，使得每个细胞有不同的标签。

具体执行步骤为：通过一个循环，遍历每个预测结果，每次循环生成一个新的color\_used数组，除了0和255外其他位置的初始值为False。result用于承接循环到的results成员，将result大小调整回（628，628），并以0.5为阈值对result二值化，小于0.5的设为0，大于0.5的设为1。然后将result乘255，并标准化为uint8。之后遍历result的每个像素，发现像素值为255时，生成一个0-255的随机数color，用color\_used检测该color是否被用过，如果用过了，则生成新的color，直到是没用过的。之后把color\_uesd的该位置设为True，然后color进入递归函数DFS，DFS采用深度优先搜索给该像素点及周围的所有值为255的像素点上色。所有细胞上色完成后存入文件夹。实际运行过程中，出现过由于直接执行所有结果的循环程序崩溃的情况，但是分开来执行每个结果的处理和存储不会使程序崩溃。即每次手动改变k的值，该段程序附在循环处理的后面。

**3.2 dataset2**

由于dataset2没有探索出好的深度学习的办法，所以使用的是demo中给出的传统方法。

**3.2.1 unit16b2uint8**

输入为待调整图像img，输出为uint8格式的img。原理为将数据格式设为uint8.具体执行步骤为输入img判断是否是uint8格式，若是，则直接返回，若不是，则返回类型转换之后的数据。

**3.2.2 img\_standardization**

输入为原图像img，输出为调整后img。将所有img转化为uint8格式和三通道彩色图。具体执行步骤为先将img转换为uint8格式，然后判断img维度，如果是二维，则扩展为三维，第三维长度为3；如果是三维，则直接返回；否则输出维数报错。

**3.2.3 load\_images**

输入为文件夹的路径，输出为images数组，用于将数据读入。依次读入文件夹中的每个文件，进行标准化，之后合并成一个列表。

**3.2.4 bgr\_to\_gray**

输入为待转换图像，输出为转换后图像。把彩色图转化为灰度图。

**3.2.5 BinaryThresholding**

分割算法所在类，输入为阈值Threshold，图像img，输出为分割后的结果label\_img。具体执行步骤为先得到img的灰度图gray，然后对gray进行阈值分割，得到掩模，然后对掩模中值滤波除噪，之后进行轮廓提取，返回分割好的图像。

**3.2.6 主函数**

主函数用于调用分割的类并生成分割结果存储起来。具体执行步骤为以110为阈值，调用BinaryThresholding，建立result\_path，然后调用load\_images将测试集数据读入，之后遍历每个样本，进行分割和存储。

**4 实现过程**

**4.1 dataset1**

整体架构（主函数）如图1

train\_Generator

Unet

model

test\_Generator

results

DFS

遍历像素点

result

resize

二值化化

=255？

Y

color

N

out

图1

**4.1.1 Unet**

Unet的整体结构如图2.

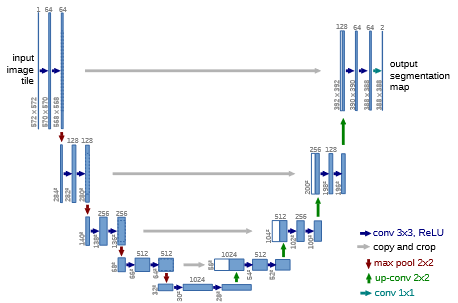


图2

有四层下连接层，四层上连接层，一层中间层，一层输出层，总体呈U形。每个下连接层都有两次卷积，一次池化。卷积层用Conv2D，输出节点从64开始，每次增加两倍，activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal'；池化层用MaxPooling2D，pool\_size=(2, 2)。上连接层有一次上采样，三次卷积，和与对称层的连接。第一次卷积的输入是上一层的上采样UpSampling2D(size = (2,2))，与对称层的连接采用concatenate，其中axis=3，卷积的参数与下连接层一致。中间层采用两次卷积，一个避免过拟合的Dropout，卷积层的设置与上连接层一致，Dropout层参数设为0.5。

**4.1.2 adjustData**

功能如图3.相当于图像和标签的统一化。

img

mask

/255

out

/255

mask[i.j]>0?

Y

mask[i,j]=1

N

mask[i,j]=0

uint8

uint8

out

图3

**4.1.3 train\_Generator**

功能如图4，使用ImageDataGenerator的flow\_from\_directory函数进行数据增强。

image\_datagen

mask\_datagen

image\_generator

mask\_generator

train\_generator

yield

adjustData

图4

**4.1.4 test\_Generator**

功能如图5，生成测试集。

img=cv2.imread

/255

target\_size

cv2.resize

yield

np.reshape

(1,)+img.shape+(1,)

图5

**4.1.5 IndexInRange & DFS**

IndexInRange仅在DFS中使用过，所以一起介绍。IndexInRange返回下标是否在范围内，DFS是一个递归函数，实现过程如图6.



图6

**5 实验结果与模型性能分析**

**5.1 dataset1**

对于Unet模型，尝试了不同的optimizer，loss，metrics。optimizer中的Adam设置有Ir分别为1e-4和1e-5，以及加入其他参数调试，得到的accuracy如表1.

|  |  |
| --- | --- |
| optimizer参数 | accuracy |
| Ir=1e-4 | 0.88 |
| Ir=1e-5 | 0.97(不稳定) |
| Ir=1e-5, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08 | 0.97(较稳定) |

表1

可见当Ir为1e-4时，accuracy很低，这说明学习率太高了，需要调低；当Ir为1e-5时，虽然accuracy显著提高，但是不稳定，仍然会有很低的情况，说明参数设置不完善；当同时设置了beta\_1, beta\_2, epsilon，accuracy变得较高且稳定。所以对于optimizer参数最后选择了第三种。

loss尝试过mse，focal\_loss, dice\_coef\_loss, 得到预测结果观测如表2.

|  |  |
| --- | --- |
| loss函数 | 预测结果 |
| mse | 有大量背景被预测成了前景 |
| focal\_loss | 比较适合背景较多前景较少的情况 |
| dice\_coef\_loss | 分割效果不够好 |

表2

可见，在本次大作业中，由于背景和前景的比例不均衡，所以使用mse时会出现预测结果不够好的问题，而focal\_loss对这类问题解决得比较好，所以loss函数最终选择了focal\_loss。

对于评价函数，影响如表3.

|  |  |
| --- | --- |
| 评价函数 | accuracy |
| 只有accuracy | 较低 |
| 加入MeanIoU | 显著提高 |
| 加入dice\_coef | 变化不明显 |

表3

可见，当三个评价函数均加入时，结果最好且最稳定，原因可能是对于细胞分割的问题，MeanIoU和dice\_coef可以较准确反映出模型的性能，使得模型相应做出调整，准确率较高。所以评价函数三者均加入。

**5.2 dataset2**

dataset2有过Unet网络的尝试，和传统算法的尝试。

Unet网络尝试如表4.

|  |  |
| --- | --- |
| 尝试 | 效果 |
| loss函数采用mse | 分割结果较差 |
| 采用网上的数据集 | 分割结果较差 |
| 采用原本的已标记的数据集并增强 | 分割效果较差 |
| 采用dataset1的数据集 | 分割效果较差 |

表4

由表4可知，已有的尝试得到的分割效果都比较差，主要原因是没有足够多的合适的数据集，网络参数设置不够合理，以及最后时间紧张但运行太慢，只好退而求其次，采用传统方法。

**6 总结**

Unet网络对于医学影像的分割有着优良的性能，本次大作业依托Unet完成了细胞分割，采用了数据增强等数据预处理方法和深度优先搜索等数据后处理方法。并在实验和资料查询过程中调整了参数，得到了较好的结果。

**7 感想**

本次大作业是我第一次接触到神经网络，有很多陌生的东西。在开始的时候会遇到不明白模型的输入输出有什么具体的要求，不知道数据增强是什么原理，也不知道应该在何处改进等问题，但是在第一次终于跑出来了一个不是全零的结果时那由衷的喜悦却让我记忆深刻。虽然这结果不太好，但是还是感到本次大作业对我有很大的帮助，让我对神经网络有了初步的认识。也期望从排名靠前的同学那里学习到更好的方法。

**8 参考文献**

[1] Nango明楠. 图像分割 传统方法 整理[EB/OL]. Medium. <https://zhuanlan.zhihu.com/p/30732385>

[2] venus. python计算机视觉学习———图像分割[EB/OL]. Medium. [2019-07-16]. <https://blog.csdn.net/qimingxia/article/details/95723482>

[3] MaggieQuan. 深度学习之图像分割[EB/OL]. Medium. [2019-05-06]. <https://blog.csdn.net/qq_36492210/article/details/89875708>

[4] wangduo. 基于阈值的图像分割方法[EB/OL]. Medium. [2016-10-31]. <https://www.cnblogs.com/wangduo/p/5556903.html>

[5] 潘振兴，谢涛. 基于半阈值的字符分割与识别研究[J]. 国防科学技术大学研究生院. P22

[6] lavendelion. 经典论文解析——Unet和Vnet——图像分割[EB/OL]. Medium. [2019-11-17]. <https://blog.csdn.net/weixin_41424926/article/details/103105801>

[7] keras中文文档. 优化器的用法[EB/OL]. Medium. <https://keras.io/zh/optimizers/>

[8] 独孤金泽. focal-loss的keras实现[EB/OL]. Medium. [2019-05-22]. <https://blog.csdn.net/qq_35747066/article/details/90437002>