人工智能导论：深度学习大作业

自63 鲍文轩 2016011423

自64 王卓远 2016013283

摘要

该份报告为《人工智能导论》课程深度学习大作业的项目报告。本项目针对可控环境的植物叶脉提取问题，采用含膨胀卷积的多层全卷积神经网络，在判定是否是叶脉的二分类问题上达到了0.98585的AUC分数和0.71338的F1分数，取得了较好的效果。

本份报告第一部分介绍了题目要求，即我们项目完成的功能。第二部分介绍我们使用的数据集。第三部分介绍了我们的算法。第四部分介绍了我们进行的实验和实验结果。第五部分展示了最终模型的结果。第六部分简述我们代码的组成和运行、测试方式。第七部分介绍本组两位同学的分工。最后一部分为总结。

一、题目简述

本次我组选择第二个题目，即植物叶脉提取。任务目标是对于干净（没有嘈杂背景、光线适中、叶子占据图片主要部分）的叶子图片，勾勒出叶子的轮廓和叶子上的叶脉。

考虑到叶脉本身往往颜色与周围叶肉不同，而颜色的差异表示了叶脉，该问题可以被视为一个边缘检测问题；但是，叶脉是具有一定的空间结构，有含义的物体，因此，该问题也可以被看作一个图像分割问题。

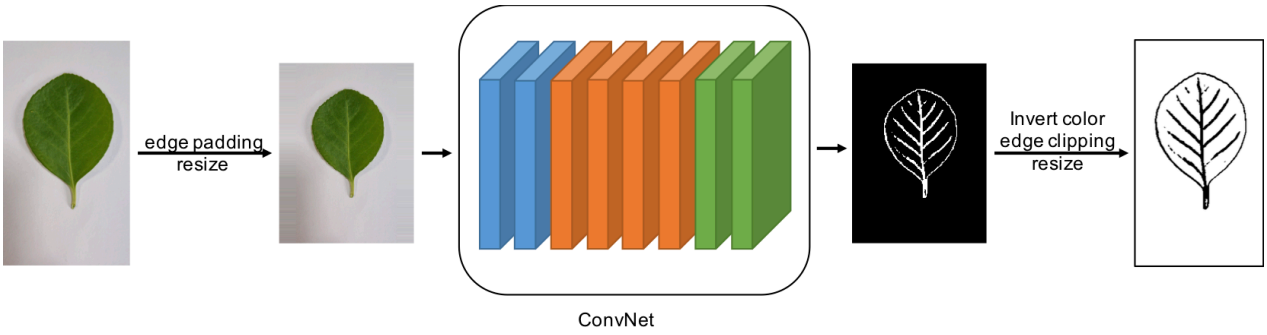
二、数据集

本项目采用的数据集是由《人工智能导论》课程的同学们收集并标注的。包含冬青、紫丁香、五叶爬山虎三类植物，共418张叶片图片及对应标注的叶脉图片。数据集的特点有：

- 叶片图片拍摄于可控环境，而非自然环境：叶片图片大多有浅色、干净背景，光线适中，叶子居中且占据主要部分。但是，也有部分图片背景嘈杂，或叶片质量较差。图片大小、比例不一，拍照质量也因人而异。
- 标注由人工完成：同学们使用Photoshop软件画出叶片的轮廓和叶脉。大多数标注仅画出主叶脉和一级叶脉，也有部分标注画出二级及以上的叶脉。大部分标注的叶脉图片与叶片图片对齐。

排除掉明显没有对齐的数据（叶片图片与叶脉图片大小不同），数据集有368张有效图片。

三、我们的方案



模型结构示意

3.1 数据预处理

由于数据集是同学们自己收集的，没有经过处理，无法直接为模型所用，我们对数据进行必要的预处理。排除掉明显没有对齐的数据后，我们将剩下的数据进行edge-padding到宽高比为3:4（过高则左右对称padding，过宽则上下对称padding），然后将所有的叶片和叶脉图片resize到300×400。

这里没有选择对图片进行剪切，是为了避免图片的非背景部分被切去。不直接resize的原因主要是避免叶片图片被拉伸，导致其自然空间结构在尺度上被破坏。进行edge padding能够使得图片的padding的部分比较平滑，而不会由于padding产生明显的边界，有利于训练的进行。选择宽高比为3:4，是因为数据集中大多数图片原先的宽高比就是3:4，且该比例也是大部分手机拍照的默认比例。

对于标注的叶脉图片，除了与叶片图片进行了同样的padding、resize操作，还对其进行了二值化。叶脉所在的像素为1，即白色；非叶脉的像素为0，即黑色。

在进行测试时，图片预处理的部分参数需要逐图进行保存，在后处理时需要用这些数据将叶脉图片还原至原图大小。

3.2 全卷积神经网络

对于该问题，我们采用了一种仅有卷积层构成的桶形网络，图片从输入到输出的尺寸一直不变（但是通道数有变化）。由于resize到300×400的图片中叶脉的宽度一般仅有几个像素，我们需要避免对图片再在空间上降维。因此，我们不使用任何池化层，卷积层的步长也被设置为1。

作为低级特征，叶脉本身很容易通过卷积层识别出来。但是，直接采用浅层卷积神经网络，网络仅能依靠极其局部的信息对每一像素是否属于叶脉进行预测，而没有使用全局信息，会出现欠拟合问题。因此网络的预测结果不佳。实际实验中，也发现在叶脉之外有大量的区域被预测成了叶脉，预测结果出现「斑点」。

为了利用叶片的全局信息，我们需要让神经网络在空间上进行更大范围的「交流」。池化层和全连接层可以达到这一目的。但是，池化层会降低空间精度，而全连接网络必然会极大的增加参数个数，使得训练速度减慢，并难以避免地出现过拟合问题。因此，我们选择采用膨胀卷积，更高效地扩张感受野。严格来说，此时最后一层特征图中的每个像素点的预测过程仍然仅依靠了比较大的「局部」信息，并未真正扩展到整张图。但是，由于叶脉本身具有树状结构：从叶脉上的某一点开始，向茎的反方向看去，这一「枝」的空间结构与整个叶脉是相似的。因此，仅依靠局部卷积的网络，只要感受野增大到一定范围，则已经可以对叶脉有比较好的提取效果。

对于此全卷积神经网络，实验中采用了多种结构，包括了不同的膨胀率、膨胀率组合、网络宽度与深度等。部分结果可以参考第四节的实验部分。最终我们选择了如下模型作为最终提交的模型。

Layer	Number of Kernel	Kernel Size	Activation	Dilation Rate
conv 1	32	5×5	ReLU	/
conv 2	32	5×5	ReLU	/
conv 3	32	5×5	ReLU	$(2, 2)$
conv 4	32	5×5	ReLU	$(4, 4)$
conv 5	32	5×5	ReLU	$(6, 6)$
conv 6	32	5×5	ReLU	$(8, 8)$
conv 7	32	1×1	ReLU	/
conv 8	1	1×1	Sigmoid	/

3.3 损失函数

3.3.1 Cross-Entropy和Hinge损失函数

二分类问题常常最后一层采用Sigmoid函数，即：

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

为解决Sigmoid函数在x值较大时出现的梯度过小导致训练速度慢的问题，使用Sigmoid函数作为输出层激活函数时，往往采用交叉熵损失函数，即：

$$L_{\text{cross-entropy}} = y \log \frac{1}{\hat{y}} + (1 - y) \log \frac{1}{1 - \hat{y}}$$

其中y为真值，$\hat{y}$为预测值。

同时，受到SVM的软分类和最大间隔分类的启发，我们也尝试在网络最后一层不使用任何激活函数，而采用Hinge损失函数，即：

$$L_{\text{hinge}} = \begin{cases} \max\{0, 1 - \hat{y}\} & , y = 1 \\ \max\{0, \hat{y} + 1\} & , y = 0 \end{cases}$$

一般的写法会将正负两类样本记为± 1，此处为了一致起见，采用上述的表示。

3.3.2 损失函数微调

本任务虽然可以被视为一个二分类问题，但正（叶脉）、负（非叶脉）样本数量相差极大。一张300×400的图片中，叶脉所占的像素数往往只有几千。采用默认的损失函数容易偏向于将所有像素预测为非叶脉。因此，我们对如上的损失函数进行了修改：

$$
\begin{aligned}
L^*_{\text{cross-entropy}} &= \alpha \cdot y \log \frac{1}{\hat{y}} + (1-y) \log \frac{1}{1-\hat{y}} \\
L^*_{\text{hinge}} &= \begin{cases} \alpha \cdot \max\{0, 1-\hat{y}\} & , y = 1 \\ \max\{0, \hat{y}+1\} & , y = 0 \end{cases}
\end{aligned}
$$

通过选择$\alpha > 1$，可以让网络对于叶脉部分的预测给予更多的「注意」，让参数更积极地向预测出叶脉的方向优化，使得网络训练能够更快得到合理的预测模型，同时一定程度上提高模型的表现。在实验部分我们对α的选择进行了讨论，最终选择$\alpha = 4$是一个比较合理的选择。值得注意的是，这个取值并不是使得正负样本平均的取值。

3.4 最佳F1分数决策

对于Cross-Entropy损失函数，一个自然的阈值选择是0.5，对于Hinge损失函数，自然的选择是0。但实际上，我们还可以在模型训练结束之后，通过对阈值进行微调来提升分类效果。

此处我们使用F1分数对我们的决策阈值进行调整，以0.01的精度选择在验证集上最大化F1分数的阈值。对于Cross-Entropy损失函数，我们在0.1到0.9的范围进行选择；对于Hinge损失函数，在-2到2的范围选择。值得注意的是，由于使用了验证集选择阈值，此时的验证集不再是模型表现的一个无偏估计。我们对于模型表现的评估就可能略高于其实际水平。

3.5 数据后处理

一般来讲，我们希望预测得到的叶脉图片与远叶片图片一样大，且是对齐的。因此，预测出叶脉图片后，还需要对预测的叶脉图片进行后处理，使得其尺寸、位置相同。此外，考虑到标注的叶脉图片是黑色叶脉，白色背景，我们还需要对预测出的图片进行反色。

后处理的操作与预处理完全相反。利用预处理时保存的信息，这里我们先对反色的图片进行resize，成为原图padding之后的大小，然后切去其上下或左右padding的区域，就可以得到与原图大小一致、对齐的图片。同时，由于这一步一般是将低分辨率图片转化为高分辨率，在进行resize时我们选择`PIL`库中的抗锯齿选项。

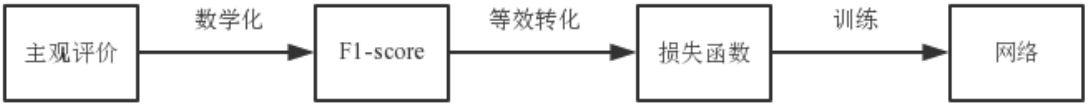
四、对比实验

由于数据集较小，我们随机将80%的数据作为训练集，20%的数据作为验证集，不额外分出测试集。所有实验均采用相同的数据划分。结果本应采用k折交叉验证，多次实验取平均，但是由于作业时间比较紧迫，且需要与其他组共享计算资源（特别是最后两天时），还是放弃了进行重复性的实验，为其他组留出一些计算资源。

同时这里我们需要明确一些衡量指标的关系，方便更好的对这些指标进行理解和修正。首先，我们明确本次任务的最终目的是可以获得人类主观感受上清晰、完整、准确的叶脉。为了能够量化这一指标，我们采用F1-score对网络输出结果进行评价。但在训练过程中，由于F1-score本身不连续也不可导，故不可以作为网络的损失函数进行梯度的反向传播，于是我们间接采用交叉熵Cross Entropy以及Hinge函数作为网络的损失函数对网络进行训练。

为了获得更好的效果，我们需要提升三个指标之间的一致性。由于F1-score是对实际问题数学化的过程，且定义明确使用广泛，故无需进行参数整定。而不同的损失函数能否帮助训练后的网络获得更好的F1-score则需要进行讨论。于是我们对不同损失函数在不同参数下的结果进行了对比。

整个问题中不同指标的作用以及相互之间的关系如下图所示：



4.1 损失函数超参数α

由上一节我们知道，交叉熵函数和Hinge函数都实现了对两类错误的惩罚，但实际上两类错误的出现次数差异很大，而且由于我们的目标是获得更大的F1-score，故可以考虑调整对于两类错误惩罚的权重，获得更好的效果。

经过简单的测试，我们发现将第一类错误对应的惩罚权重α增加有利于F1-score的提升，于是我们选择$\alpha = 2, 4, 6, 8$进行对比实验，来选择最好的超参数。实验中，我们固定网络架构、batchsize为16、optimizer为RMSProp等参数不变，Loss Function均选用Hinge函数，Padding均选用same padding，统一训练1000代，得到实验结果如下：

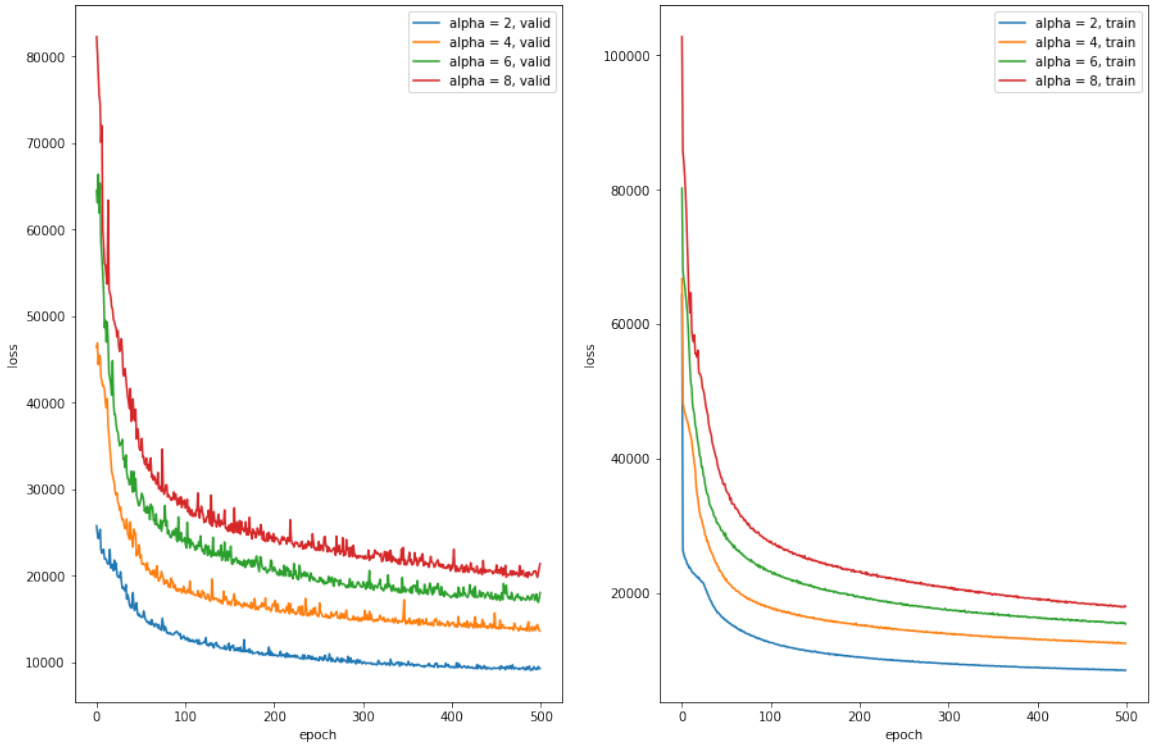
Hinge函数对应的baseline网络架构:

Layer	Number of Kernels	Kernel Size	Activation	Dilation Rate
conv 1	32	5×5	ReLU	/
conv 2	32	5×5	ReLU	/
conv 3	32	5×5	ReLU	$(2, 2)$
conv 4	32	5×5	ReLU	$(4, 4)$
conv 5	32	5×5	ReLU	$(6, 6)$
conv 6	32	1×1	ReLU	/
conv 7	1	1×1	None	/

最佳阈值与最佳F1分数：

α	Best F1 Score	Best Threshold
2	0.70400	-0.04
4	0.69761	0.63
6	0.69326	0.67
8	0.67990	0.87

训练损失函数变化曲线：



从结果可以看出，$\alpha = 2$时拥有最高的F1-score，而$\alpha = 4$时训练最为稳定。综合考虑训练的稳定性和F1-score，我们最终认定$\alpha = 4$为最优的超参数，并以此作为我们之后的参数标定。

4.2 损失函数形式

可以知道，交叉熵函数和Hinge函数两者在理论上均可以实现对网络的训练，但实际效果需要通过量化实验来进行考量。本节采用4.1中得到α=4作为标定超参数，分别以交叉熵函数和Hinge函数作为损失函数进行实验，得到结果如下：

Cross Entropy对应的baseline网络架构：

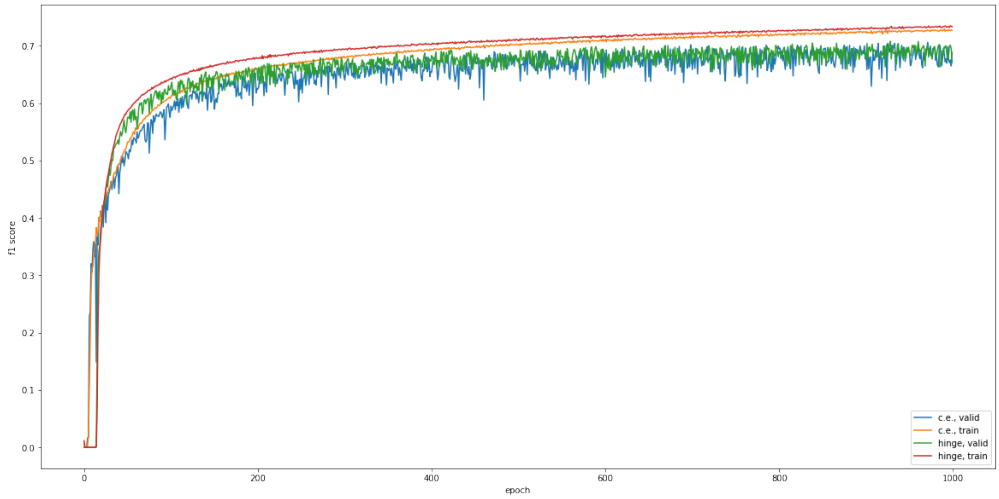
Layer	Number of Kernels	Kernel Size	Activation	Dilation Rate
conv 1	32	5×5	ReLU	/
conv 2	32	5×5	ReLU	/
conv 3	32	5×5	ReLU	$(2, 2)$
conv 4	32	5×5	ReLU	$(4, 4)$
conv 5	32	5×5	ReLU	$(6, 6)$
conv 6	32	1×1	ReLU	/
conv 7	1	1×1	Sigmoid	/

最佳阈值与最佳F1分数：

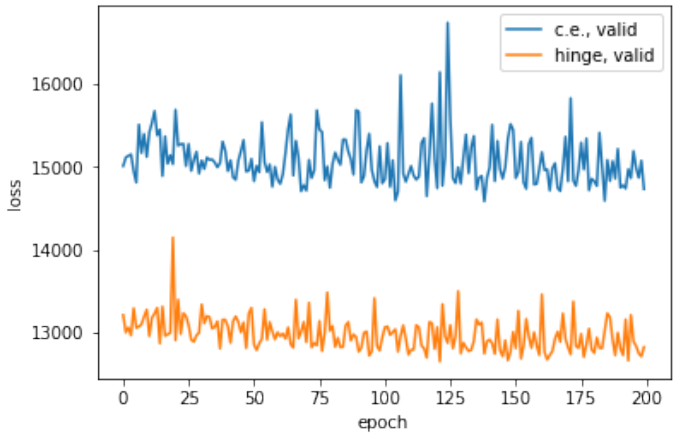
Loss Function	Best F1 Score	Best Threshold	AUC
Hinge	0.70961	0.48	0.98134

Cross Entropy	0.71081	0.57	0.98562

训练F1-score变化曲线：



最后200代损失函数变化曲线：



从结果中可以看到，用Hinge损失函数可以获得更稳定的训练过程，而使用交叉熵损失函数可以提高F1-score。对于本问题两者各有优劣，我们在后面的对比中统一选用Hinge函数作为损失函数。

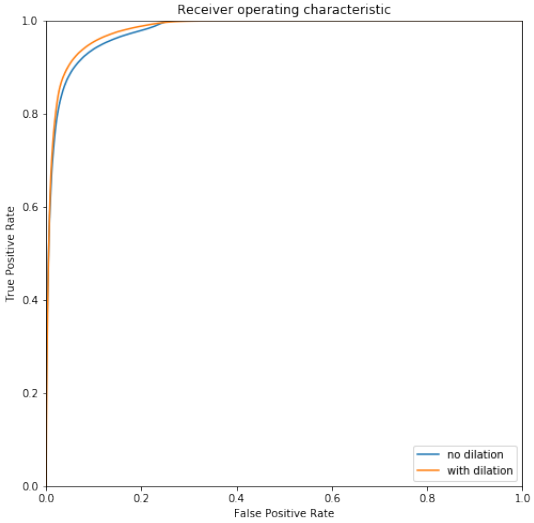
4.3 膨胀卷积的使用

从问题的特点出发可以知道，由于叶脉具有一定的空间特性，所以增加对于全局空间信息的感知在理论上可以获得获得更好的叶脉标注。为了验证这一假设，我们对比了加入膨胀卷积前后网络的效果，得到如下实验结果：

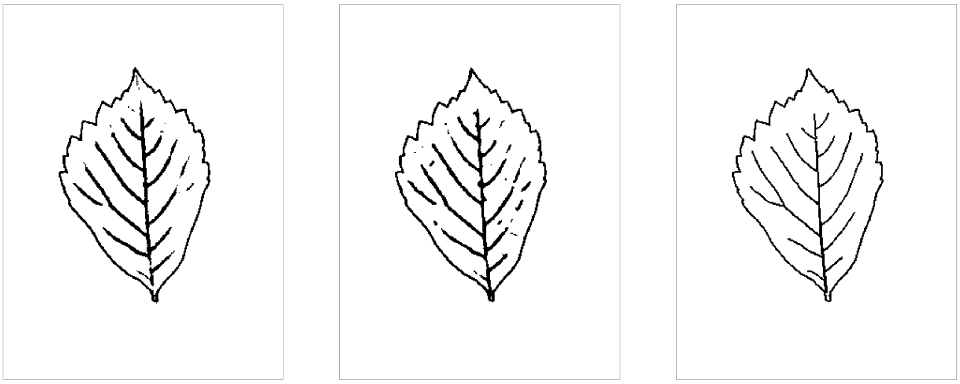
最佳阈值与最佳F1分数：

Dilation	Best F1 Score	Best Threshold	AUC
no	0.70961	0.48	0.97817
yes	0.69325	0.58	0.98135

ROC曲线：



实际效果对比：



注：从左至右依次为使用膨胀卷积，不使用膨胀卷积，实际叶脉标注。

从实验结果可以看出，膨胀卷积的使用确实可以帮助网络学习到更多有关叶脉结构的特征，从而获得更高的F1-score。另外，从图片结果中也可以看出，加入膨胀卷积后可以使得最终输出的叶脉图片噪点减少，获得更好的叶脉标注，这另一方面也证明了F1-score和主观评判的效果是一致的。

4.4 膨胀率

膨胀卷积本身可以帮助网络获得更多的全局信息，而膨胀卷积的参数设定是否会对结果产生影响这一点我们还不清楚，所以设计了实验对比不同膨胀率下网络的训练效果，并得到如下结果：

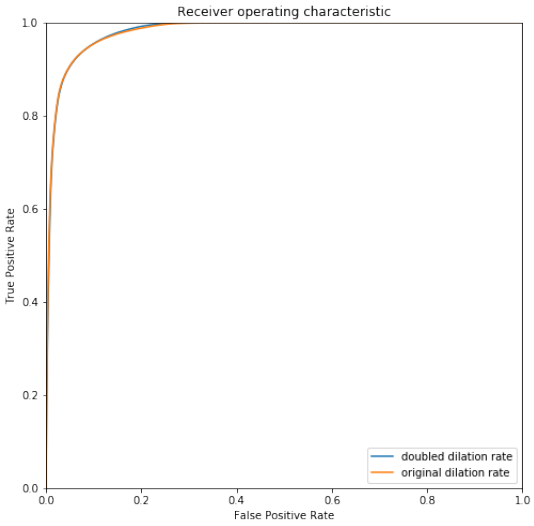
膨胀率参数设定：

Layer	Dilation Rate (New)	Dilation Rate (Original)
conv 1	/	/
conv 2	/	/
conv 3	$(4, 4)$	$(2, 2)$
conv 4	$(8, 8)$	$(4, 4)$
conv 5	$(12, 12)$	$(6, 6)$

最佳阈值与最佳F1分数：

Dilation Rate	Best F1 Score	Best Threshold	AUC
original	0.70961	0.48	0.98135
doubled	0.71570	0.59	0.98142

ROC曲线：



理论上，膨胀率的变化会影响卷积过程中每一个位置的信息来源及其感受野，但实际测试结果表明，在本问题中，适当改变膨胀率并不会对F1-score有太大影响，膨胀率这一参数的鲁棒性比较强。

4.5 膨胀卷积层的位置

同样是利用膨胀卷积，在网络的前端使用还是在网路后端使用是否存在差异？我们尝试将网络卷积部分倒置，膨胀系数的设定顺序也更随着倒置，得到新的网络并测试，得到如下结果：

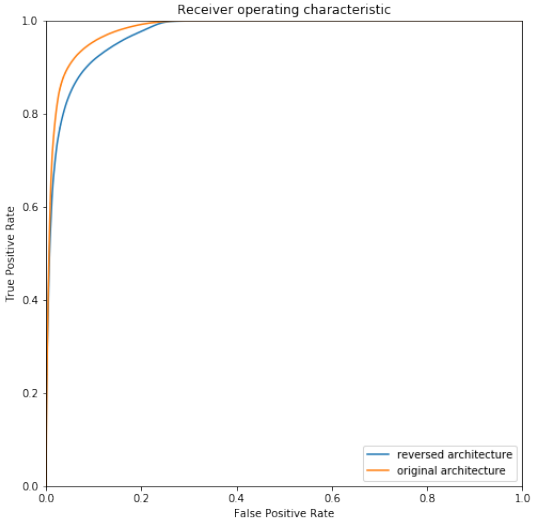
膨胀卷积层位置设定：

Layer	Dilation Rate (New)	Dilation Rate (Original)
conv 1	$(12, 12)$	/
conv 2	$(8, 8)$	/
conv 3	$(4, 4)$	$(4, 4)$
conv 4	/	$(8, 8)$
conv 5	/	$(12, 12)$

最佳阈值与最佳F1分数：

Architecture	Best F1 Score	Best Threshold	AUC
original	0.70859	0.58	0.98142
reversed	0.65519	0.91	0.97254

ROC曲线：



从结果可以知道，反向构架网络使得膨胀系数先大后小并不能更好地完成任务，原始构架中先进行特征的提取再进行膨胀卷积的做法与本问题更为契合。

4.6 深度

深度学习中的网络构架一般基于先验知识，而网络深度是一个很重要的维度，一般来说越深的网络可以进行的函数拟合性能越强，但对应也会有训练时间长，容易过拟合等问题。以本问题为例，适当增加深度，将原先Hinge函数的baseline架构增加一个膨胀卷积层，得到新的网络架构，保持其他参数一致并同样训练1000代，对比得到结果如下：

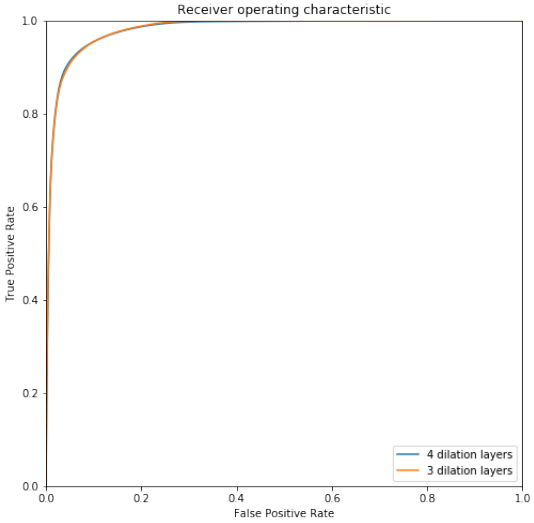
增加卷积层后的网络架构：

Layer	Number of Kernel	Kernel Size	Activation	Dilation Rate
conv 1	32	5×5	ReLU	/
conv 2	32	5×5	ReLU	/
conv 3	32	5×5	ReLU	$(2, 2)$
conv 4	32	5×5	ReLU	$(4, 4)$
conv 5	32	5×5	ReLU	$(6, 6)$
conv 6	32	5×5	ReLU	$(8, 8)$
conv 7	32	1×1	ReLU	/
conv 8	1	1×1	None	/

最佳阈值与最佳F1分数：

Number of Layers	Best F1 Score	Best Threshold	AUC
7	0.70961	0.48	0.98135
8	0.71571	0.59	0.98104

ROC曲线：



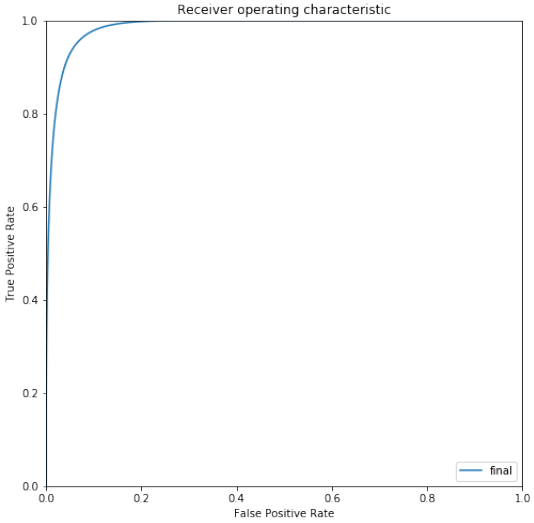
可以看出，深度的增加并没有明显提升网络的性能，说明对于本问题，原本的网络深度已经足够。当然，导致此结果也有可能是因为较深的网络没有训练完全的缘故，但是由于训练完全难以定义，故采取相同训练1000代的方式进行比较。

五、最终结果展示

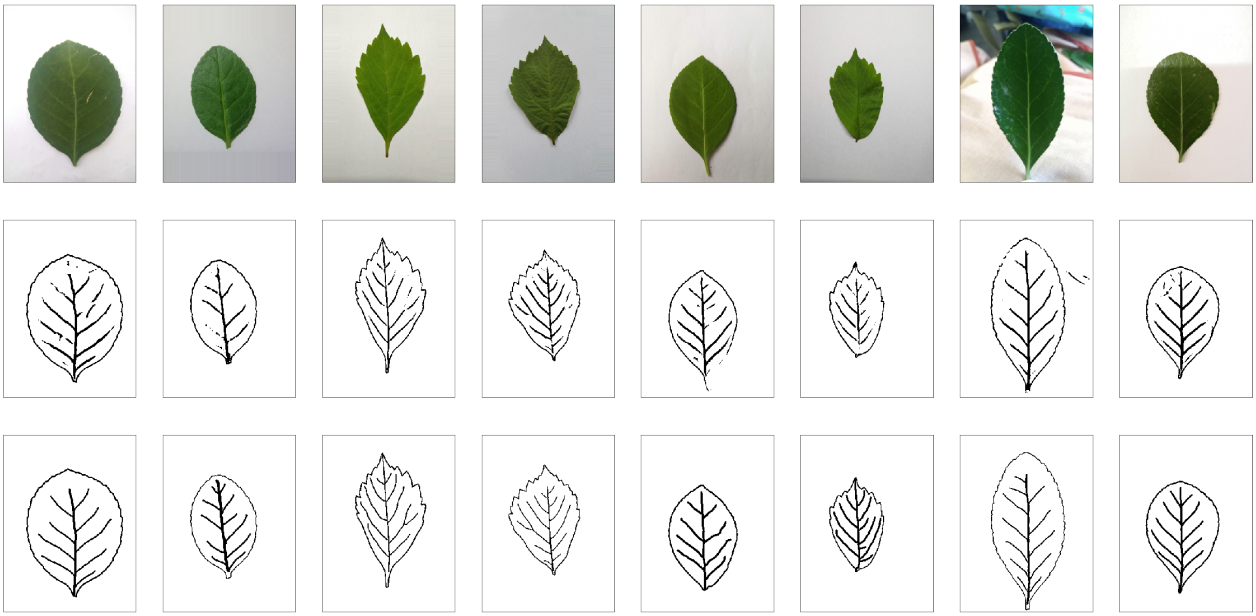
我们最终使用的模型的评价结果如下：

Best F1 Score	Precision	Recall	AUC	Best Threshold
0.71338	0.68210	0.74766	0.98585	0.50

ROC曲线如下：



对于验证集中的部分叶片预测结果如下。首行为叶片图片，第二行为我们模型最终给出的预测结果，第三行为标注的叶脉图片。



可以看出，我们的模型可以从原始叶片图片中提取出叶脉，并达到较为理想的效果。我们的模型对于不同的光照条件，叶片种类，拍照方式都展现出了一定的鲁棒性。特别地，在第七列的结果中，我们的模型甚至可以预测出背景中叶片的叶脉，是一个令人兴奋的结果。我们认为该模型经过调整也许可以用于自然环境中叶片图片叶脉的提取。

但是也可以看出，最终获得的叶脉图片中部分存在少量斑点，同时预测结果一定程度上会受叶片表面的瑕疵影响，也说明我们的模型还有可以改进的空间。当然出现这种情况也和标注的质量相关，不同标注者标注所依照的规则有所偏差，所以造成网络对于一些特质的学习不是很有把握，导致了一些地方斑点的形成。可以想象，如果有更为一致的数据集，我们的网络应该可以获得更加理想的结果。

六、测试接口说明

提交的压缩包中`test.py`文件为代码的测试文件，用于对待测试叶片图片的轮廓、叶脉勾勒。

代码运行需要一个txt文件，文件中有n行，每行是一张测试图片的路径。内容如：

```
/home/baowenxuan/datasets/class0/task2/leaf/2016013282 (1).jpg
/home/baowenxuan/datasets/class0/task2/leaf/2016013282 (2).jpg
...
/home/baowenxuan/datasets/class0/task2/leaf/2016013282 (10).jpg
```

在测试文件路径下，测试文件采用如下方式运行。

```
python3 test.py arg1 arg2
```

其中`arg1`为txt文件的路径；`arg2`为图片保存的路径，在该路径下会生成名为`vein_pred_pics`的文件夹，其中存放预测的叶脉图片，名字与叶片相同，格式为jpg。

注意：叶片图片中不应含有“.”字符，且应为三通道jpg图片。其他格式图片未经过测试，可能不能正常运行。

模型实测在Ubuntu下可以运行。若不能运行，可能是模型路径问题，可在`test.py`文件的187行中改变模型的路径：

```
model = load_model('test_model.h5', custom_objects={'tf': tf})
```

运行该份代码的环境：

tensorflow-gpu 1.13.1

Keras 2.2.4

scikit-learn 0.19.1

七、组内分工

我组实际并没有特别明确的分工，绝大多数内容都是两人共同完成。完成作业时交流也非常密切。

鲍文轩同学主要负责：图片预处理，模型结构调整，实验设计，测试代码编写，报告的二、三、六部分。

王卓远同学主要负责：论文调研，模型结构调整，实验设计，模型表现的评估与模型选择，报告的三、四部分。

参考：本次大作业参考了Keras官方文档和不同函数使用的一些示例算法。

八、总结

本次大作业让我们相对完整地完成了一个深度学习项目的开发，熟悉了深度学习模型的架构方法，并得到了相对不错的结果。同时利用对比实验的方法，定性探究各个模块的意义和不同参数的效果差异，使得我们对于网络内部的功能有了很为深刻的理解。同时，对于数据的采集、处理也让我们意识到高质量数据的珍贵程度以及其对于数据驱动方法的重要性。