模式识别大作业：CT 图像的探索

江仕蒙
陈旭鹏

摘要

利用机器学习尤其是深度学习进行图像识别和分析是近两年非常火热的方向，本题的数据来自 LUNA16，kaggle 的 Data Science Bowl 和国内的天池医疗大赛均使用该数据举办过相关的比赛，比赛要求通过分析带有结节位置标注的病人的 CT 图像，利用模式识别和机器学习的知识，建立模型预测结节的位置。我们主要探索了图像的处理、常用的一些机器学习模型，U-net 以及 3D-CNN 等模型。在探索过程中我们学习和总结到了很多图像处理和模式识别中遇到的一些问题和获得的经验。

关键词：CT 图像，结节检测，机器学习，图像处理，U-net, 3D-CNN

1 工作介绍

医疗图像的自动化处理和分析一直都是医疗领域的热点问题，尤其是近几年深度学习的火热使得其分析与检测变得相对更加容易一些。利用深度学习，人们已经对胸部、脑部和骨骼的 X 光、CT 以及 MRI 数据进行了很多的分析和诊断。本问题要求我们利用病人的 CT 图像进行诊断，CT 图像信息含量大，相对的也更难处理。由于计算资源以及时间的限制，我们主要做了图像处理，2D 与 3D 模型的探索工作，通过本次作业学习到了很多图像与机器学习和深度学习的知识。

2 2D 图像的简单尝试

我们利用助教已经提取好的 2D 图像，使用了几种常用的机器学习方法和 CNN 网络进行了简单的分类尝试。

2.1 数据来源

这部分的数据来源一是来自助教提供的原始正负样本，我们一共使用了 1950 个正样本和 1950 个负样本进行模型的训练和预测。

2.2 方法与指标

2.2.1 机器学习模型

我们使用了一些常用的机器学习模型进行二分类，这里我们利用 sklearn package 使用多种机器学习模型进行分类，包括：Logistic Regression（L2 regularization），Random Forest，Gaussian Naive Bayesian，Gradient Boosting(use xgboost package), KNN, Gaussian Process Classification and SVM(linear kernel).

type	accuracy
logistic l2	0.30153846
random forest	0.05512821
gaussian naive bayesian	0.43564103
xgboost	0.05461538
knn	0.12871795
gpr	0.27
svm(linear)	0.19769231

在分类之前我们先对图像进行了简单的预处理，将其标准化。接下来我们把正负样本加上 label，按照十折划分进行交叉验证。

2.2.2 CNN 与 VGG 模型

我们还使用了一个非常简单的 CNN 模型以及一个稍微复杂的 VGG16 网络进行分类的尝试。这里不再赘述 CNN 的相关基础知识，我们只使用了现成的 CNN 和 VGG16 框架，加入了 batch normalization 与 dropout 和 regularization 等方法帮助模型更好地学习。

2.2.3 指标

我们用 accuracy 作为评价指标，来查看几个分类模型的好坏，accuracy 的定义非常简单：如果将样本归为正确的类别，即称为预测正确（True），否则即为预测错误（False），因此：

$$\mathbf{Accuracy} = \frac{\mathbf{True}}{\mathbf{True} + \mathbf{False}} \tag{1}$$

2.3 结果

2.3.1 机器学习模型结果

可以发现几个机器学习模型的结果相当不好，甚至不如完全预测为正样本或负样本。

这是可以理解的，这几个机器学习模型都不能处理二维数据，因此输入的数据均进行了压平操作，从而丧失了局部的相关信息，而 CNN 这样的模型才可以学到图像的局部相关性。我们把模型学习的权重存储，可视化后可以发现其学习的相当杂乱随机，这也可以理解，毕竟简单的机器学习模型的权重很少，比如逻辑斯谛回归，只有图像尺寸大小的权重（如 1600），因此难以获得强大的学习能力，其接近于对所有样本的图片像素点进行平均学习，分类效果自然很差 (图 1)。

这里我们还需要解释一下模型的结果，老师当时提出疑问，既然预测结果这么差，那反过来预测不就行了，不应该出现小于 0.5 的结果。经过我们的反复核查，我们认为出现小于 0.5 的结果是可能的。首先经过模型内部的优化，我们可以看到模型在训练集上的预测结果都是大于 0.5 的，说明模型本身并没有存在问题。而反过来预测这个说法是不可行的，因为模型本身并不知道什么时候

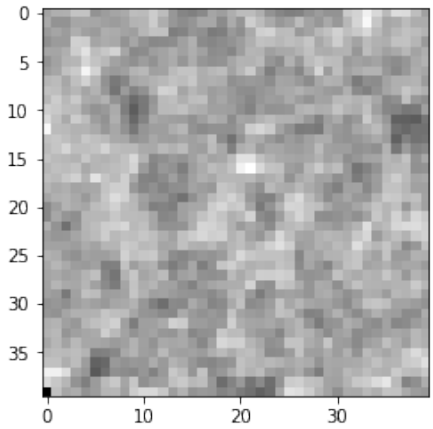


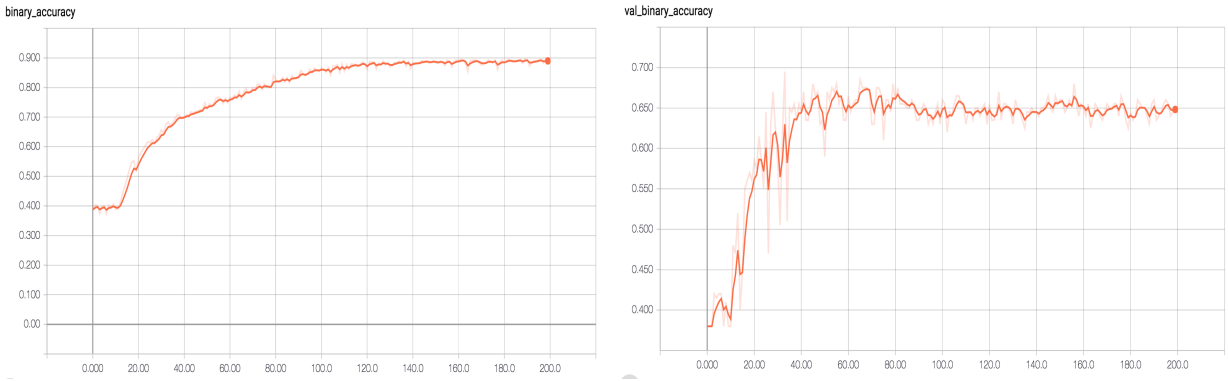
图 1: 逻辑斯谛回归模型的权重趋于对压平的图像平均学习

可以反过来预测，也就是说反过来预测这个决定本身就需要模型自己学习，我们不能在整个学习的结束后人为地将结果反过来，这样的模型泛化能力也会非常差，更换数据后依然无法预测好。事实上包括深度学习模型都可能在测试集上获得不足 0.5 的准确率，因此小于 0.5 的结果确实也是可以接受的。

2.3.2 CNN 与 VGG16 模型结果

我们尝试过一个基本的三层 CNN 模型，该模型学习图像特征的能力很差，其 loss 一直无法下降，accuracy 一直停留在 0.467，因此我们又使用了 VGG16 模型预测。

VGG16 网络的预测结果存在过拟合现象，所谓过拟合指的是模型在训练集上取得很好的训练结果（如较高的 accuracy），但是在测试集上表现差距较大。经过我们对图像的简单预处理，以及对模型进行正则化等方法，提升了模型的表现，最终在测试集和训练集上可以分别达到 0.8912 和 0.6500 的准确率，其曲线随训练的变化我们使用 tensorboard 可视化如图 2：



(a) training set accracy　　(b) test set accuracy

图 2: training set 和 test set 上模型预测准确率

接下来我们进行了一系列的图像预处理工作，继续从三维和二维上对本问题进行了一些探索。

3 图像处理与 3D 图像分析

3.1 图像预处理

此部分的内容我们首先参考了助教提供的教程，使用几个 python 的 package 进行预处理和数据的存储。

首先，进行图像处理的工具都只能进行二维图像的处理，因此首先要做的是将三维图像提出并存储，便于后续逐层读取与处理。这里我们使用了一个可以减小存储占用以减小内存消耗的 package：h5py，在大多数情况下我们会将三维 tensor 存储为 hdf5 格式，方便快速读取，有时我们也存储为 npy 格式，使用 numpy 存取。

在预处理之前，我们也提取了样本的信息，包括病人的结节三维坐标，直径，这里同一个病人可能产生多个结节。同时我们还需要从图像中获得该图像的 origin 和 spacing，这两个值也相当关键，origin 表示了图像在真实世界中的坐标，但是这里并没有解释清楚 origin 的概念，我们后续花了一些时间才搞清楚 origin 的概念。而 spacing 表示了两层图像之间的真实距离，不仅对于我们获得结节中心在矩阵中的定位有用，而且还对我们进行的额外的一步处理有作用：即把所有样本的层间距离统一处理成 1mm*1mm*1mm。

基本的图像预处理包括这样几个步骤：

- 将图像二值化
- 去除边界，效果是将临近边界的一些离散点去除，减少冗余信息
- 实现对二值图像连通区域的标注，二值连通区域还具有传递性，如果像素点 A 与像素点 B 同值并邻接，我们称 A 与 B 连通，易得，如果 A 与 B 连通，B 与 C 连通，则 A 亦与 C 连通。一个连通区域可能包含很多个像素点，我们可以将同一个连通区域的所有像素点都用同一个符号来进行标记。
- 选择留下两个面积最大的连通区域
- 腐蚀操作，使肺部与连接的血管分离
- closing 闭运算，类似于先膨胀后腐蚀的操作，可用来填充空洞，这里也被用来保持结节与 lung wall 的贴合。
- 填充空洞，这里先用 roberts 边缘算子进行边缘检测，再填充空洞
- 最后，对经过上述一系列预处理的二值化图像，找到其值为零的点的坐标，将输入的图像对应坐标设置为 0，间接实现对预处理中找到的重点区域的突出

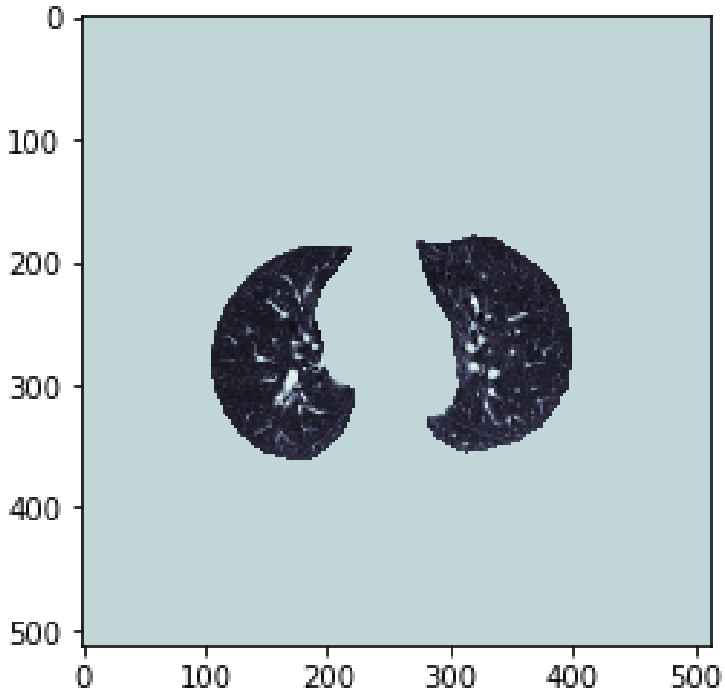


图 3: 处理后的单层图像示意图

接下来，通过不断调取上述流程的函数，就可以实现对所有样本的图像的逐层处理了，一个样本某层的示意图如图 2

接下来，我们还额外进行了一步操作，前面说到不同样本之间具有不同的 spacing（每个样本三个维度均可能有不同的 spacing 值），为了让模型学习起来更加统一，我们又进行了一步 resample 的操作，通过计算三个方向需要放缩的因子，我们使用了 scipy.ndimage 的 interpolation.zoom 函数对样本进行了 spacing 的调整，统一调整为 1mm*1mm*1mm。

我们顺便用 matplotlib 可视化了每一个样本的 3D 图像，可以方便后续查看（图 3）。虽然可以通过调整角度实现对不同图像的观测，但是还是不如软件查看方便。

3.2 提取 3D 结节区域

这个步骤是为了后续尝试使用 3D CNN 模型进行分类准备，因为前述的 origin 理解问题，以及 CT 图像特殊的坐标排列，使得这个步骤出了很多问题，我们在这个步骤花了很久的时间，一方面我们想办法克服提取结节的问题，一方面学习和研究了 3D CNN 的知识。

我们认为利用现有的程序如 3D RCNN 等对于我们来说主要就变成了重复他人的工作，而且计算资源的要求无法承受，因此我们决定从简单的地方做起探究三维图像的 CNN 模型。我们希望能够根据标注信息，提取出带有结节的正方块，再提取出对应体积的负样本，用 3D CNN 模型实现分类。

首先我们要利用从图片中获得的 origin 和 spacing，以及 csv 文件中的结节坐标来找到结节在三

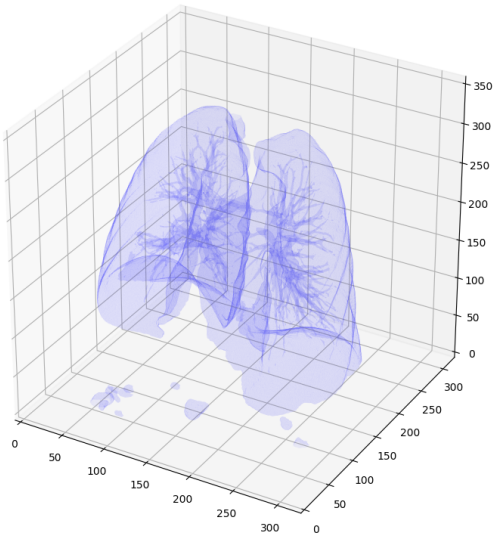


图 4: 样本的 3D 图像

维矩阵中的坐标。注意样本只有六百个，而每个样本的结节数量可能大于一，一共有 975 个结节标注。我们的思路很简单：用结节的中心坐标减去 origin 坐标，即可获得结节中心相对于 origin 的值。但是我们需要理解，origin 代表哪个点。通过查看结果，我们发现 origin 大多数为比较小的数，结节的中心坐标与之相减后都是正数，因此我们认为 origin 就应该是样本 tensor 的 [0,0,0] 位置。正常来说，用结节中心减去 origin 坐标的数值，应该还要处以 spacing 才可以得到矩阵坐标，但是因为我们额外进行了 resample，将 spacing 都调整为 1，因此不必再处以 spacing。

因为我们设计的 3D CNN 对输入图像的尺寸有要求，还没有做到可以灵活适应输入尺寸，因为我们对截取的含结节正方体的大小进行了限制。我们观察了一下结节的直径分布 (图 4)，决定选择三种尺寸：6mm,16mm,40mm。通过判断结节直径落入的区间，将其向上靠拢至尺寸。

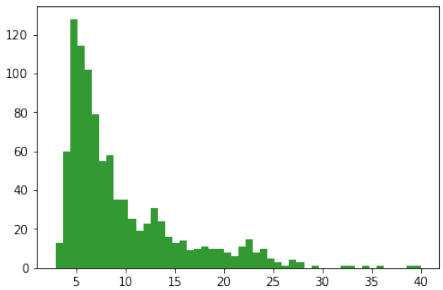


图 5: 结节的直径分布

负样本的提取，我们使用了一种比较简单的策略，即在正样本附近相邻的区域提取同样大小的立方体，先判断其相邻区域是否有其他结节，再分别判断其三个方向的延伸是否会超出样本范围，如果有不超出的就截取出来。

我们发现按照这个思路进行的三维正负结节样本提取总是无法成功，有很多提取出来的样本并不是正方体，在这个地方我们卡住了很久，检查了很多遍都没有找到问题所在，直到我们发现一个相关的数据说明，指出样本矩阵的坐标轴不是 (x,y,z) 而是 (z,y,x) 我们才发现问题所在，通过 numpy 的函数 einsum，我们把样本矩阵调整成了 (x,y,z)，与标注的坐标保持一致，得以顺利提取出了样本。

在提取样本的步骤还有一些后续的改动，因为我们发现分类效果不佳，网络的 loss 很难下降，我们重新分析了一下我们提取的样本的情况，发现部分正负样本存在全为 0 的情况，可能是之前预处理的步骤出现了一些问题。于是我们重新提取了较为正常的矩阵，又对矩阵做了逐层的标准化处理，更加方便模型学习。

3.3 3D-CNN 用于结节正负样本的分类

3.3.1 实验方法

3.3.1.1 实验设计

- 训练阶段：使用 3D CNN 对于所提取的 3D 肺癌结节样本进行分类训练。
- 检测阶段：将 512*512*512 的 CT 三维图像送入已经训练好的 3D CNN 网络进行卷积及池化操作，在最后一个卷积层后进行滑窗操作，提取样本送入已经训练好的全连接层进行分类，其中滑窗大小即为样本在最后一个卷积层后的大小。由此可以得到各个滑窗内含有结节的概率，由滑窗序号数也可以得到该滑窗在三维空间上的位置
- 训练阶段：sample 3D CNN fc classification result
- 检测阶段：CT 3D CNN sliding windows fc classification results。

3.3.1.2 评价标准

样本分类标准:

- True Positive(TP): 样本为正，判断为正
- True Negative(TN): 样本为负，判断为负
- False Positive(FP): 样本为负，判断为正
- False Negative(FN): 样本为正，判断为负

指标定义：

- True Positive Rate（真正率, TPR）或灵敏度（sensitivity），为正样本预测结果正确数 / 正样本实际数。
- True Negative Rate（真负率, TNR）或特指度（specificity），为负样本预测结果正确数 / 负样本实际数。
- 精确度（Precision），反映了被分类器判定的正例中真正的正例样本的比重。
- 准确率（Accuracy），反映了分类器对于正负样本整体的正确判断能力。

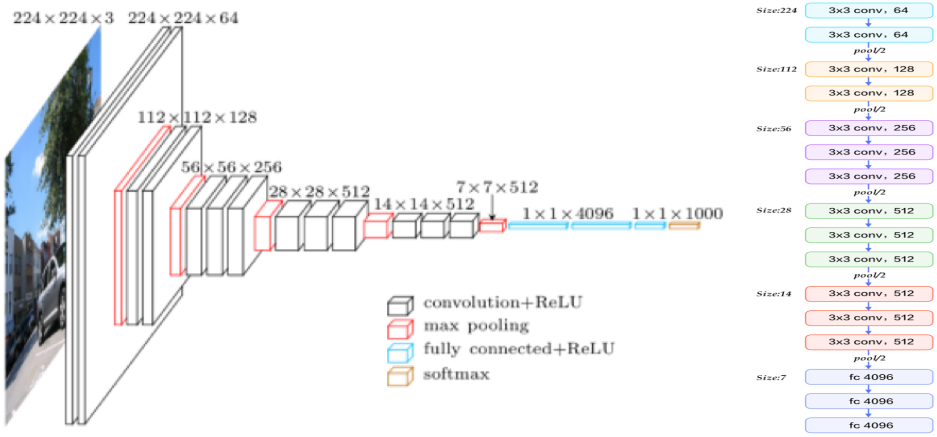
其定义如下：

$$\mathbf{TPR} = \frac{TP}{TP + FN}$$
$$\mathbf{TNR} = \frac{TN}{TN + FP}$$
$$\mathbf{P} = \frac{TP}{TP + FP}$$
$$\mathbf{ACC} = \frac{TP + TN}{TP + TN + FT + FN}$$

3.3.1.3 方法与程序来源：

深度学习框架：Tensorflow 基于 Tensorflow 的 API：Tensorlayer

卷积神经网络结构： 如图 6 所示，以 VGG-16 为基础。



(a) (b)

图 6: VGG16 结构

在 VGG-16 的基础上，进行了一些调整：

- 减少池化层数量: VGG16 原本是对于边长为 224 的图片进行操作, 所以采用了 5 个 maxpooling 池化层，而对于边长为 40 的样本，减少池化层数量至三个，使得卷积网络部分输出 feature map 的边长为 5
- 减少全连接层数量：为了减少过拟合以及减少参数数量以满足内存要求，减少了全连接层的数量以及每层神经元数量，实际使用的情况为 4096-1000-2 的全连接层结构
- 减少通道数量：由于 VGG16 是对于输入为 RGB 三通道彩色图像进行操作，而我们的样本为单通道二值图像，所以适当减少了卷积层的通道数，为原本通道数量的一半，即 32-64-128-256-256

样本大小： 在大作业展示后，我们从交流中了解到结节的判断不应当仅限检测于结节生长的小区域，有可能与更大范围内的肺结构相关。所以我们将最初的边长为 6mm,16mm,40mm 三种样本大小改为最终的统一使用边长为 40mm 的样本。在进行数据增强之后，总共有正样本 2640 个，负样本 2640 个。

训练参数设置： 损失函数：交叉熵损失，x 为预测，z 为样本标签

$$\mathbf{loss}(\mathbf{x}, \mathbf{z}) = -\sum_i (x_i * \log z_i + (1 - x_i) * \log(1 - z_i)) \tag{2}$$

优化子：Adam，$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$，见图 7。

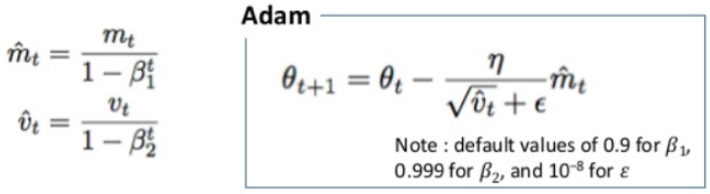


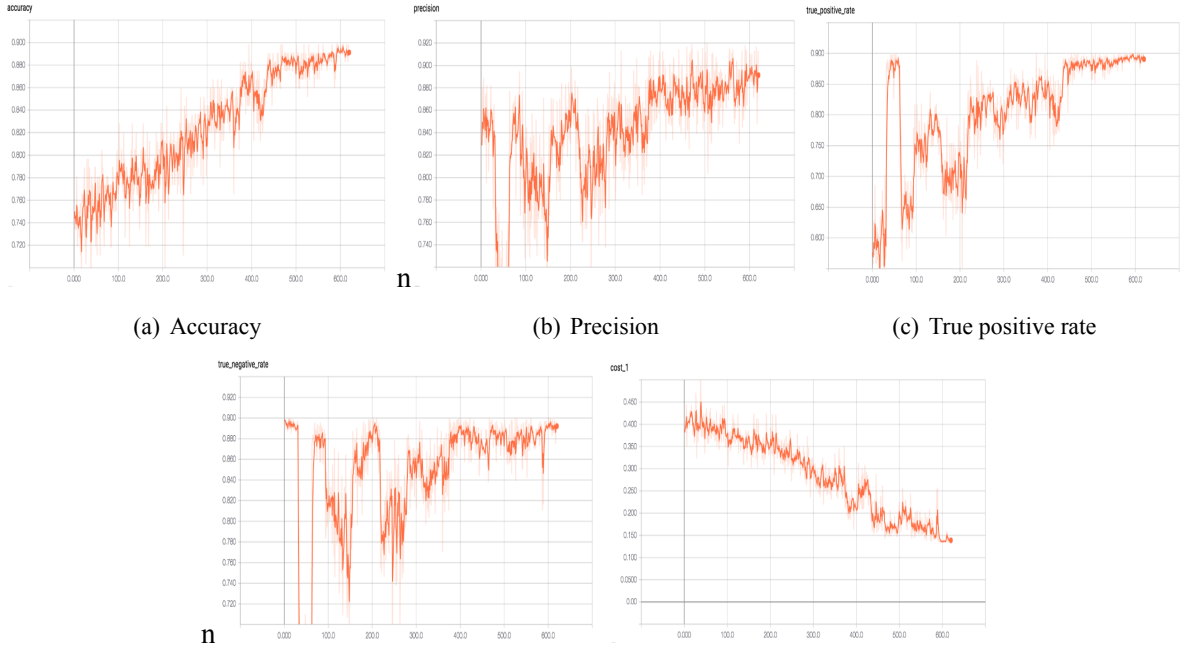
图 7: Adam optimizer

学习速率：0.0001
训练集与验证集样本数：4000 和 1280，训练集与验证集中正负样本数相同。
Minibatch：128

3.3.1.4 实验结果

训练集： 此处图像记录的是每个 step 的各项指标的变化，即训练每个 minibatch 后各项指标的变化情况，故毛刺较为明显。我们是在此前已经训练好的网络上对数据增强后的样本进一步训练，可以看到整体准确率在完成第一个 epoch 后已经超过了 70%，且整体趋势一直保持增长，到训练结束时已经接近 90%，cost 的趋势也是一直下降。但是可以看到 TPR,TNR 和 precision 有一个较大的跳变，具体来说 TPR 大幅增长，TNR 和 precision 大幅度下降，根据定义分析原因应当为在这

几轮训练当中 TP 和 FP 都大幅度增加，且 FP 的增长幅度大于 TP。由于在取 minibatch 前我们对整个 batch 进行了 shuffle，可能 shuffle 的结果正好是这几次训练当中正样本占大多数，所以导致网络分类结果更加倾向于将样本分类为正样本。这样的结果具有较大的随机性，由于训练耗时过长，所以没有时间重新训练来验证这种推测。总体来说，在停止训练的时候各项指标均接近 90%。结果由图 8 所示



(a) Accuracy (b) Precision (c) True positive rate

(d) True negative rate (e) Loss function

图 8: training set 指标展示

验证集： 从验证集的结果来看，结果与训练集相比有一定的差距，各项指标在结束训练时在 70% 到 80% 之间，并且出现了 cost 上升和 NPR 下降的情况，这说明模型已经出现了过拟合，训练结果的泛化推广能力不够理想。总体来说，TNR 很快就达到了一个约 90% 高值，而 TPR 则是缓步上升，说明负样本的特征相较于正样本的特征较容易学习。结果由图 9 所示

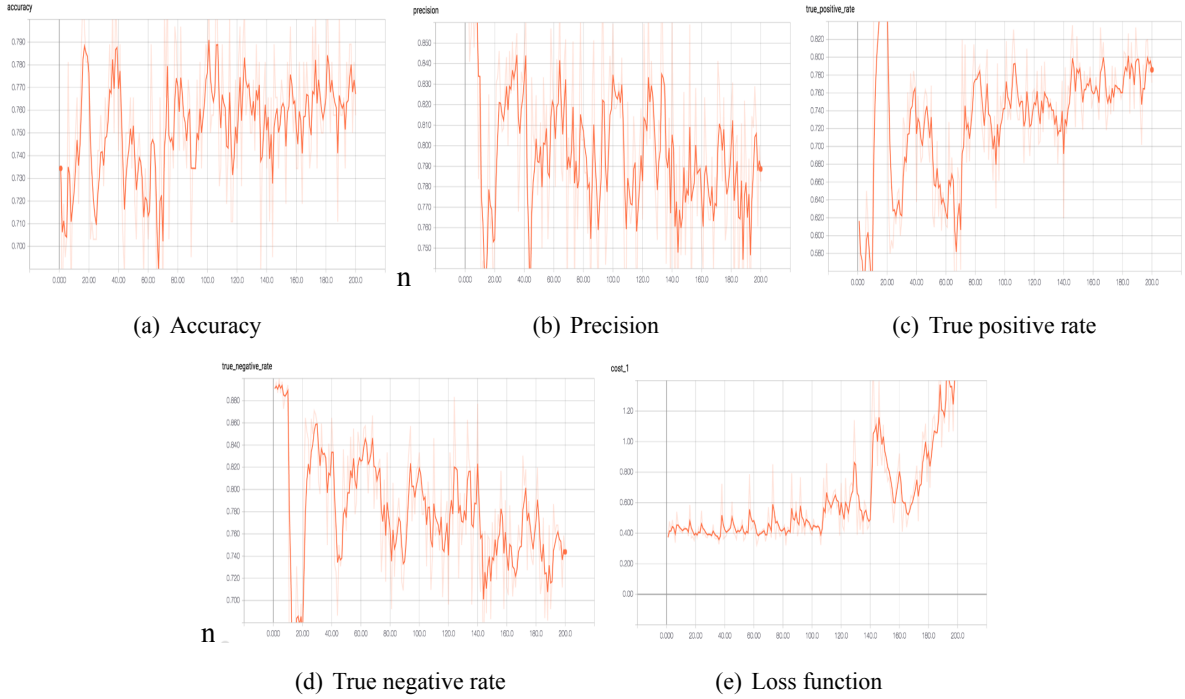
3.3.1.5 实验讨论 在实验过程当中，我们遇到了不少的问题。最初我们是选择了边长为 6mm，16mm,40mm 三种变长的样本分别进行训练。如图 10 所示是我们得到的结节直径分布，为了不使结节直径相对于样本边长过小，影响训练效果，我们才采取这种策略，最初只有直径 6mm 的样本 448 个，16mm 的样本 684 个，40mm 的样本 186 个，导致了训练结果非常不理想。

以下是不同大小样本集的训练结果：

6mm 样本验证集结果如图 11

16mm 样本结果如图 12

40mm 样本结果如图 13



(a) Accuracy (b) Precision (c) True positive rate

(d) True negative rate (e) Loss function

图 9: training set 指标展示

从验证集的正确率来看，6mm 的样本正确率在 50% 以下，16mm 和 40mm 的样本正确率也只略微高于 50%，但是从训练集正确率来看，已经较为稳定地达到了 80% 的正确率，所以实际上已经是处于过拟合的状态了。验证集的正确率在整个训练的过程中并未出现高值，说明网络并未学习到能够推广到其他样本的特征。

所以接下来我们围绕着调整网络结构来试图获得更好的效果，比如增减卷积层数量，改变池化层位置，改变损失函数为均方误差，减少池化层，给卷积层权重增加 L2 正则化，maxnorm 正则化等等，然而训练结果依然没有太大的变化，正确率在 50% 左右浮动。

由于 50% 这个数字实在是太过敏感，我们猜测可能是网络什么都没有学习到，只是在进行均匀预测。在检查具体的预测结果后，发现了问题所在——正样本大部分预测错误，负样本大部分预测正确。这可能是由于在划分训练集和验证集之前，将样本随机打乱所造成的。虽然整体正负样本数量相同，但是可能在划分时，在训练集中加入过多负样本，导致正样本的识别能力较差。于是后来调整为在训练集和验证集中均加入数量相同的正负样本。然而，出乎意料的是，训练结果依然很差，我们不得不怀疑是样本本身的问题。

在仔细检查样本之后，我们发现，提取的样本存在大量空样本，即样本数据均为 0。我们不得不重新提取样本，最后分别得到了直径为 6mm，16mm，40mm 的样本 162，364，150 个，且将数值全部二值化，即数据只有 1 和 0，分别代表灰度在阈值以上和阈值以下。

对于新的样本，训练结果如下：

直径 6mm 样本训练集与验证集如图 14

直径 16mm 样本训练集与验证集如图 15

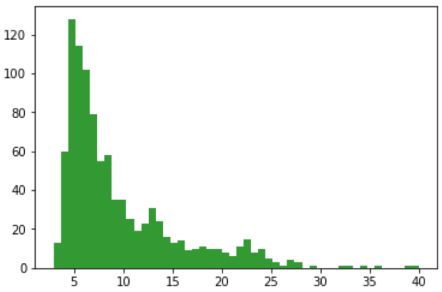


图 10: 样本分布

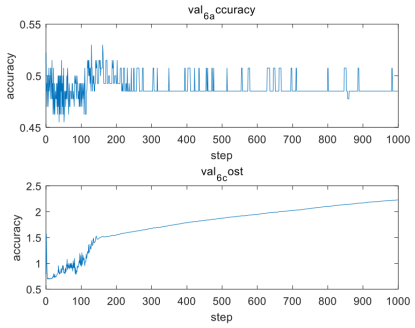


图 11: 6mm 样本验证集结果

直径 40mm 样本训练集与验证集如图 16

这里就是在我们进行展示前得到的结果，从曲线来看，效果不理想。在和其他小组交流之后，发现他们所使用的样本数比我们大一个数量级，如一个小组使用了 8000 个样本，且只使用一种大小的样本，考虑更大空间内的结构特征。在进行这些改进后，如实验结果中所述，训练效果得到了大幅度提升。

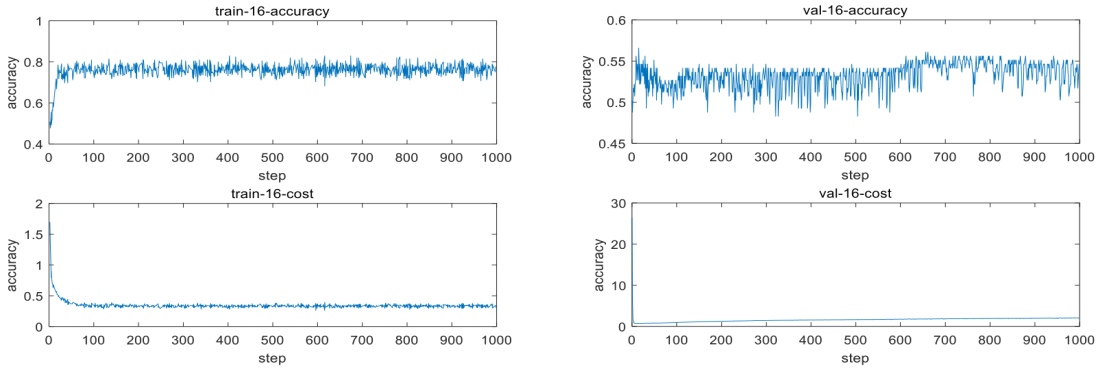
4 图像处理与 2D 图像分析

我们希望在 2D 图像分析上做一些改进，尝试能否对 2D 图像进行比较好的分类。因此我们尝试了一些稍有改进的预处理以及尝试使用 U-net 网络寻找结节区域。

4.1 图像预处理

这部分我们尝试了与第一部分略有不同的图像预处理方法。

首先还是获得结节的中心坐标，因为上面的尝试，我们可以很容易地找到在样本矩阵中结节的位置。我们接下来想使用一个网络来寻找到结节所在的区域，即后文提到的 U-net，所以为了给 U-net 提供 label，我们需要标注出结节的区域，也就是 Mask。寻找 Mask 的工作可以通过结节三维坐标和直径完成。我们采取两种方式寻找 Mask，一是一结节中心所在的 z 层向上向下一共取三层，这些 mask 离结节中心最近，保留的信息应该最多，第二是根据不同结节的直径把有结节的层



(a) 训练集　　(b) 验证集

图 12: 16mm 样本结果

的 Mask 都提取出来。因此我们可以使用定义的提取 mask 函数把样本层以及其对应的 mask 层方便地提取出来。

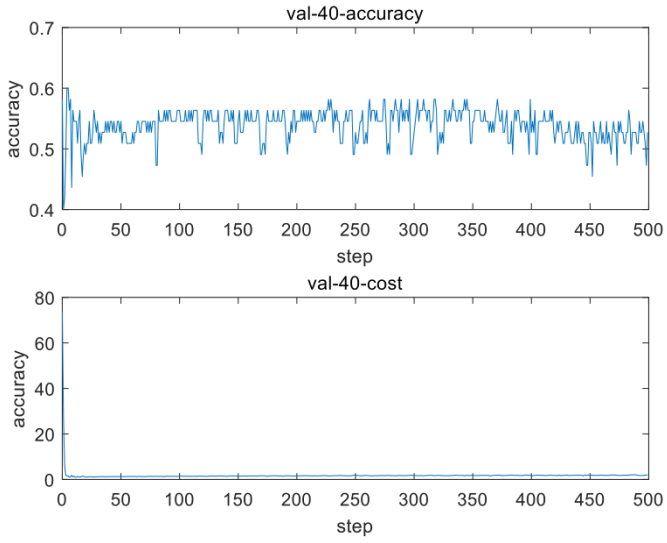
同样道理我们也提取了负样本出来，即不含结节的层，思路与 3D 的样本提取时类似。我们依旧采取一个结节提取三层以及根据直径提取两种策略。

我们可以查看其中一个样本的提取结果（图 17）:

我们希望主要分析肺部的特征，因此希望找到和肺部相关的区域，这样可以为后续减少 U-net 的结节搜索区域提供可能。可以看到图中除了中间亮度较低的肺部，还有亮度较高的脊柱、肋骨，以及肌肉、脂肪等组织。我们希望能够留下暗的区域，去掉亮的区域。这是一个二分类的问题，我们也可以通过设置寻找阈值来筛选需要的区域。因此一个合理的想法是使用聚类算法将像素点聚为两类，保留较暗的区域。因此我们对一张图中所有像素点的亮度做概率密度分布，然后用 K-means 算法，找出这个明暗分解的阈值, 聚出两类，找到两类中心，将两个聚类中心的均值作为阈值。

在聚类之前我们先进行了标准化，并且再对边缘进行处理，通过寻找中间范围内 (100-400 in 512) 的像素点的均值，将图像的最大最小值替换为均值，减少像素点极端值对聚类的影响。接下来我们进行聚类，对像素点的亮度的分布聚类，然后进行一系列的预处理工作，这里我们把感兴趣的肺部区域称为 ROI，即 region of interest。

- 将亮、暗两类聚类中心的均值作为阈值，大于阈值的像素设置为 0，小于阈值的设置为 1，实现二值化，且肺部区域被标记为 1，白色。
- 腐蚀操作，增大非 ROI 区域的黑色部分，使之尽可能连通。
- 膨胀，增大 ROI 的白色区域，尽可能消除小的黑色区域。
- 接下来，图片会有三个主要的连通区域，包括肺部，体外以及两者之间的身体轮廓区。分别进行标注，然后提取出 region 的信息，利用三个区域的 region 信息，选出高和宽符合一定要求的区域，即为肺部区域。



(a) 训练集

图 13: 40mm 样本结果

图 18 是寻找 ROI 的一系列操作的示意图。

这样的话我们就可以通过产生的 ROI，来找到原图中对应的区域，如图 19 所示：

在这个过程中我们也发现部分图片的提取 ROI 的区域并不正常，但是由于时间所限我们不能一张一张地人为修正，如果想要更好地提取 ROI，应该从几何特征角度设置更加合理的提取条件。

这里我们已经提取出了肺部区域以及每张对应的 mask 区域，我们再进行一步操作，将肺部区域的背景不要置为 0，而是改成一个与 mask 区域像素均值有关的数，我们找到肺部结节区域的像素均值和标准差，将肺部背景区域的像素值设置为 $\mu - \sigma$。

接下来我们将提取出的图像和 mask 用五折和十折交叉验证进行划分。我们也使用了图像变换的方法，使用各种常用的仿射变换增加了样本量。以一个病人的侧面胸片为例，我们通过旋转、缩放、对称、平移等变换人为增加了一些样本。以一个样本为例（图 20）：

这样我们可以得到至多两万多个样本，当然其实这个数量有点过多了，因此我们主要还是使用原图像进行训练。

另外我们还对图像分辨率进行调整，因为内存的问题，我们将 512*512 的图像通过插值法调整为 224*224。

4.2 寻找二维图像的结节区域并判断是否为结节

4.2.1 U-net 网络

对于二维图像，我们有一个很朴素的思想，即先用某种网络找到疑似结节，再训练一个分类器对是否是结节进行判断。这种思路不但直观，而且显得很可靠。深度学习模型虽然好处很多，但是有一个很大的弊端，就是可靠性不足，因为图像的特征众多，很难说清楚模型究竟学习的区域是否

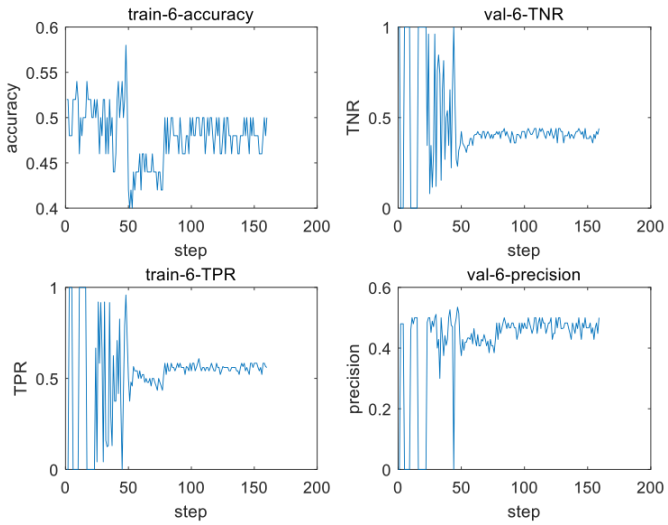


图 14: 重做的 6mm 样本验证集结果

是我们希望其学习的。比如对于判断是否为结节的模型，也许存在一个模型，学习的区域并不是结节，而是其他图像特征，也有可能将结果预测的很好。对于普通的图像分类问题，我们致力于提升其综合准确率即可，而对于医疗图像来说，我们觉得一个很重要的因素是可靠性。而能够实现区域定位的网络中就有专门为医疗图像领域设计的 U-net。

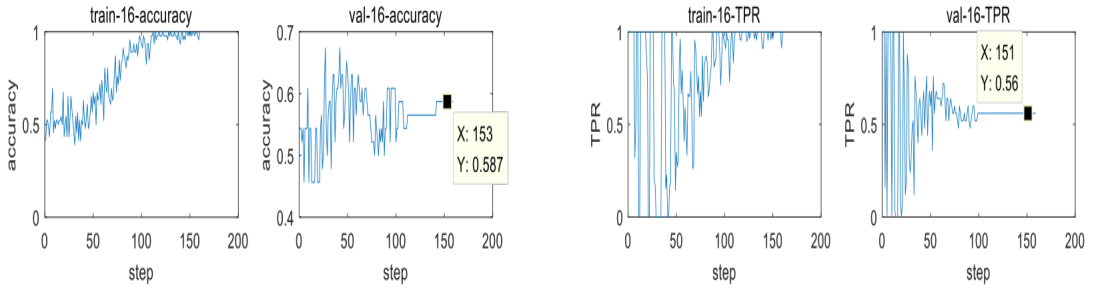
（另外，我们在阅读学习医疗图像相关研究，比如用深度学习利用胸片数据诊断心脏病和肺部疾病的研究时就发现，很多模型只是努力提升准确率，并没有考虑可靠性。我们顺便产生了这样的想法：针对这类问题，对心脏和肺部等需要关注的特殊区域先进行标注（如圈画），然后训练一个 U-net 来找到希望重点关注的区域，再训练一个分类器，加大重点区域的权重进行训练和预测。这样的重点学习关键区域的模型应该能更加可靠。）

U-net 是医学图像领域很常用的用于目标检测的深度学习网络，其能够通过对低层次特征映射的组合，构建高层次复杂特征，来实现精确定位。

2014 年加州大学伯克利分校的 Long 等人提出全卷积网络（FCN），这使得卷积神经网络无需全连接层即可进行密集的像素预测，使用这种方法可生成任意大小的图像分割图，且该方法比图像块分类法要快上许多。之后，语义分割领域几乎所有先进方法都采用了该模型。

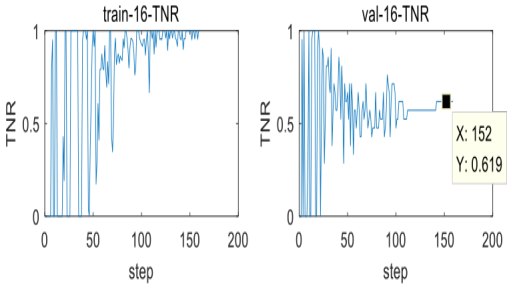
除了全连接层，使用卷积神经网络进行语义分割存在的另一个大问题是池化层。池化层不仅扩大感受野、聚合语境从而造成了位置信息的丢失。但是，语义分割要求类别图完全贴合，因此需要保留位置信息。U-Net 是一种编码器-解码器结构。编码器逐渐减少池化层的空间维度，解码器逐步修复物体的细节和空间维度。编码器和解码器之间通常存在快捷连接，因此能帮助解码器更好地修复目标的细节。

正如上面所说，U-Net 是一种基于 FCN 的框架，是一个全卷积神经网络，输入和输出都是图像，没有全连接层。较浅的高分辨率层用来解决像素定位的问题，较深的层用来解决像素分类的问题，但是 U-Net 并不像 FCN 将特征相加，而是 concatenate 生成双倍通道的特征图，再卷积。

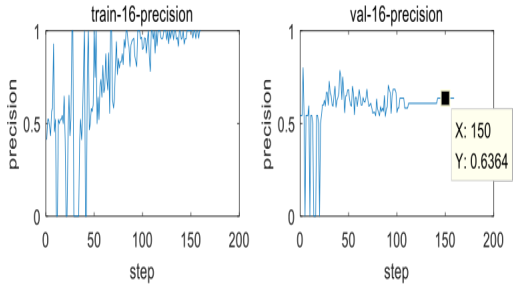


(a) Accuracy

(b) TPR



(c) TNR



(d) Precision

图 15: 重做的 16mm 样本结果

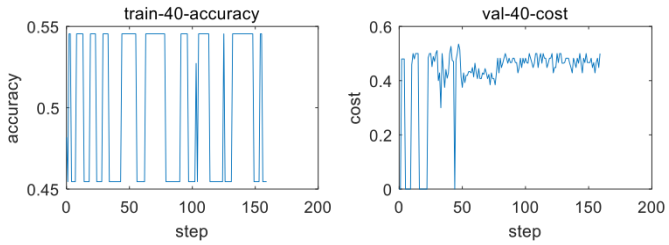


图 16: 重做的 40mm 样本验证集结果

我们使用了如图 21 所示的 U-net 网络

4.2.1.1 并行加速尝试 为了加快训练速度，我们尝试了几种并行加速的方法，因为神经网络参数很多，训练起来比较慢，因此我们想实现并行的计算，这其中的难点是虽然计算梯度等步骤可以分开在不同的 GPU 上做，但是每轮训练都应该把结果合并起来计算损失函数等，否则就相当于在多个 GPU 上分别训练模型，因此我们实现了多 GPU 上结果的合并，这样就可以使用多个 GPU 加快运算的速度了。

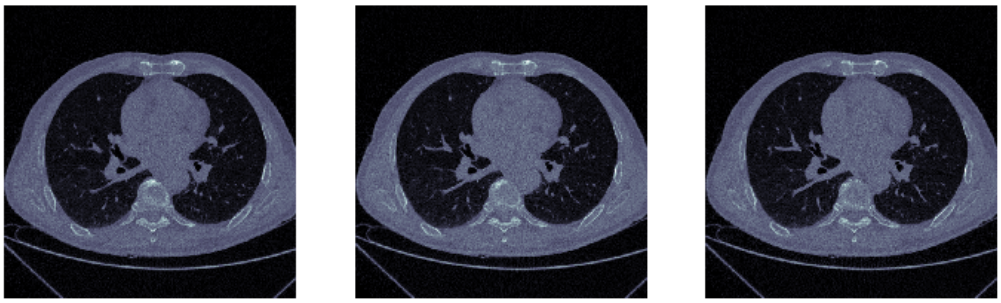


图 17: 与结节中心最近的三层

4.2.2 Loss function

4.2.2.1 DICE 系数 损失函数是我们本次实验重点学习，调试的一个部分。对于 U-net 来说，最经典的损失函数来自 DICE 系数，其常被用来衡量两个集合的相似度，定义如下：

$$\mathbf{DICE} = \frac{2|X \cap Y|}{|X| + |Y|} \tag{3}$$

利用 DICE 系数，我们可以衡量预测的 mask 区域和真实区域的重叠程度，显然 DICE 越接近 1，说明模型的效果越好。因此我们可以将 U-net 网络的损失函数定义为 DICE 系数的相反数，利用梯度下降法，使用不同的优化器（比如 SGD+momentum，Adam 等）进行优化。

4.2.2.2 DICE 系数作为损失函数的问题 我们利用 DICE 类别的损失函数进行了很多轮的尝试，通过模型每轮训练返回的 validation set 上的 DICE 系数，我们可以发现在二维图像上训练模型来预测 Mask 是一件非常困难的事情。当然一方面可能是只通过二维预测结节本来就比较困难，另一方面可能是 DICE 损失函数的问题。我们通过查阅资料发现有一些人经历了同样的问题，即用 DICE loss function，预测出的结节与真实结节不重合，或者预测出的区域面积很大时模型就提前停止了训练（我们设置了在 validation set 上如果超过五轮 dice coefficients 不上升就提前中止训练）。我们认为是模型很难学到 Mask 的区域，因此预测面积无法继续缩小 (图 22)，而一旦缩小后的面积没有包括 Mask 区域，DICE 系数就会突然阶跃至 0。

4.2.2.3 限制区域大小的尝试 我们希望能够想办法限制预测区域的大小，因为结节占整幅图的比例相当小，我们希望将预测的结果限制到比较小的区域内。一般想实现特殊的要求，可以通过改进 loss function 实现，因此我们想实现一个能够提高 DICE 同时还可以缩小预测区域面积的损失函数。

这里我们首先想到，也许可以通过限制预测区域的方差实现对预测区域的大小的限制。这里我们进行了较多的思考和探索，为了叙述流畅，我们主要叙述我们认为比较合理的推理过程。

我们假设预测区域和真实区域都符合某种分布，我们希望拿一个参数分布拟合区域的经验分布。为了方便起见我们选择了多元正态分布拟合经验分布。接下来我们需要寻找二维正态分布的均值与方差的表示形式。我们用 KL 散度衡量正态分布 Q(x) 与经验分布 P(x) 的相似度。通过最小化

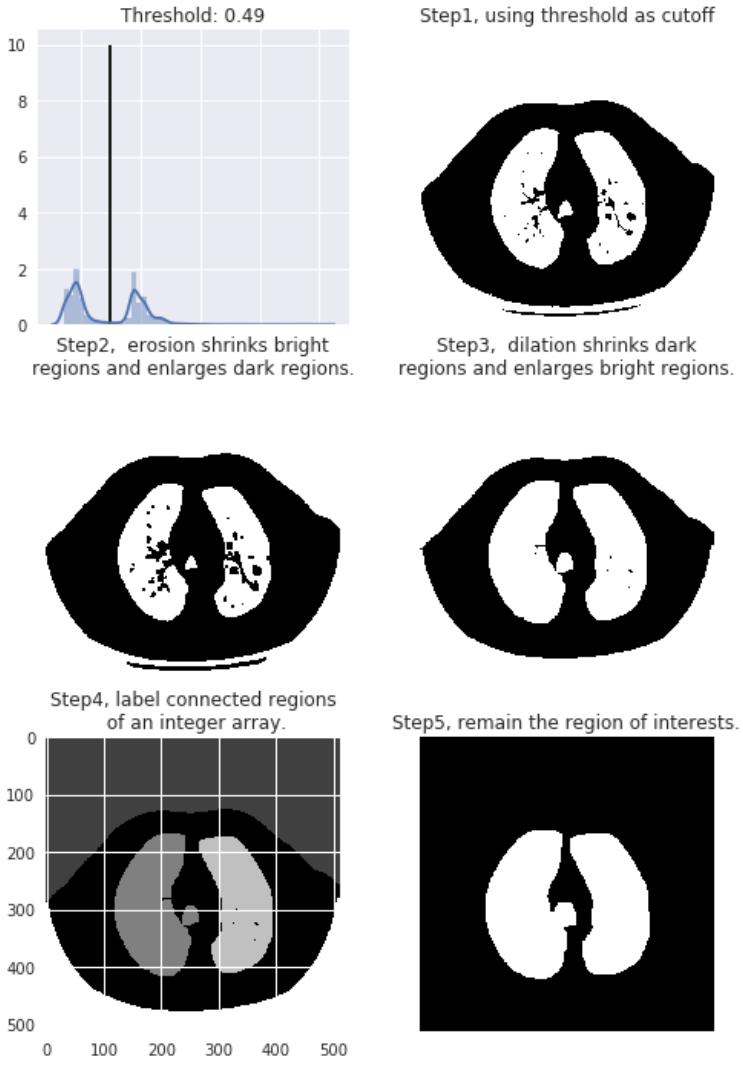


图 18: 寻找 ROI (region of interest)

KL 散度来寻找均值和方差。我们先考虑一维的情况。具体做法如下:

$$Q(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$D_{KL}(P||Q) = \sum_{i=1}^{N} p(x_i) log \frac{Q(x_i)}{P(x_i)}$$

$$= \sum_{i=1}^{N} p(x_i)[-\frac{1}{2} ln(2\pi) - ln\sigma - \frac{1}{2\sigma^2}(x_i - \mu) - lnp(x_i)$$

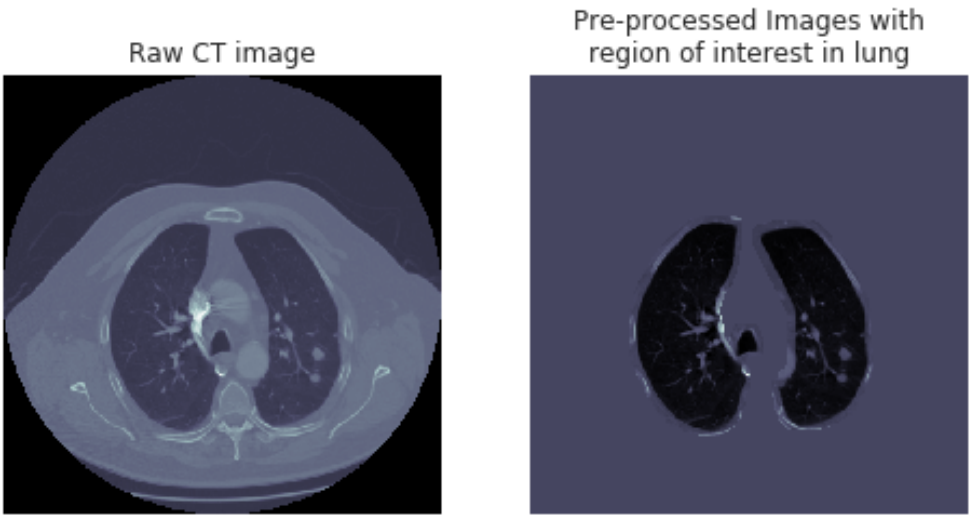


图 19: 利用 ROI 找到原图中对应区域

通过求取极值获得 μ, σ^2 的分布：

$$\frac{\partial D_{KL}}{\partial \mu} = \sum_{i=1}^{N} p(x_i)[-\frac{1}{\sigma^2}(\mu - x_i)] = 0$$

$$\frac{\partial D_{KL}}{\partial (\sigma^2)} = \sum_{i=1}^{N} p(x_i)[-\frac{1}{\sigma} + \frac{1}{\sigma^3}(\mu - x_i)^2] = 0$$

因此我们有：

$$\mu = \sum_{i=1}^{N} x_i p(x_i)$$

$$\sigma^2 = \sum_{i=1}^{N} (x_i - \mu)^2 p(x_i)$$

同理的，我们可以考虑二维的情况，这里我们假设正态分布在两个维度是对称的（isotropic normal distribution），这样会有一些好处，比如协方差矩阵是一个对角阵，联合分布的方差为其行列式的值，

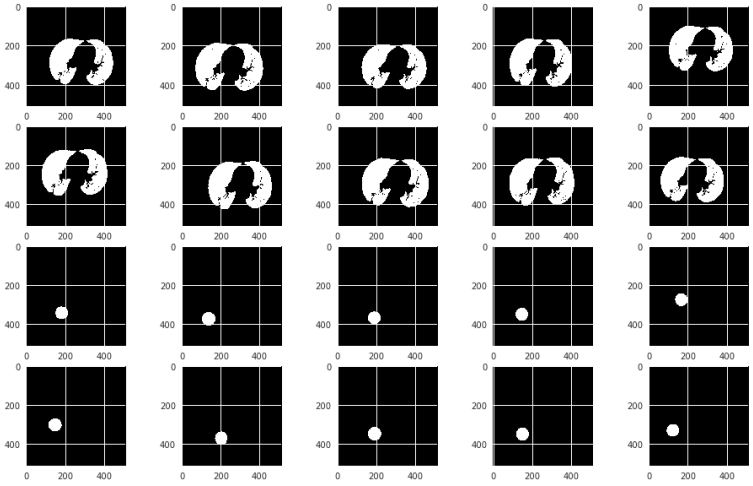


图 20: image augmentation

也就是 σ_x^2, σ_y^2 的乘积。下面直接给出结果：

$$Q(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp[-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(x-\mu_y)^2}{2\sigma_y^2}]$$

$$\mu_x = \sum_{i=1}^{N}\sum_{j=1}^{N} x_i p(x_i, y_j)$$

$$\mu_y = \sum_{i=1}^{N}\sum_{j=1}^{N} y_i p(x_i, y_j)$$

$$\sigma_x^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} (x_i - \mu_x)^2 p(x_i, y_j)$$

$$\sigma_y^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} (y_i - \mu_y)^2 p(x_i, y_j)$$

这里我们找到了一个区域如果遵从正态分布假设时边缘分布方差的表达式，如我们上面提到的，其联合分布的方差正是 $\sigma_x^2 \times \sigma_y^2$。

因此我们的思路已经很明晰了：如果可以计算出预测出的 Mask 区域的方差，就可以作为约束项添加进损失函数中，在最理想的情况下，通过限制方差大小，同时提高 DICE，我们也许能找到理想的区域。具体的实现过程稍微有些曲折，在图像上显然没有直接的公式计算出方差，因此我们借助 numpy，通过模拟离散的经验分布函数，分别出 x 和 y 方向的均值，计算出图片每个像素点的联合概率，进一步依据公式计算出方差。这样我们得到了新的损失函数：$lossfunction = -dice + \sigma_x^2 \times \sigma_y^2$，之前我们是用 numpy 实现的损失函数，但是为了使得损失函数是可训练和优化的，我们使用 keras 的基本操作重写了损失函数的代码。

在更改损失函数后我们又进行了模型的训练的尝试，我们对 dice，方差等指标进行监测，结果发现问题很明显：方差是一个很大的项，DICE 项在初始阶段不会被考虑，而优化方差的过程中很

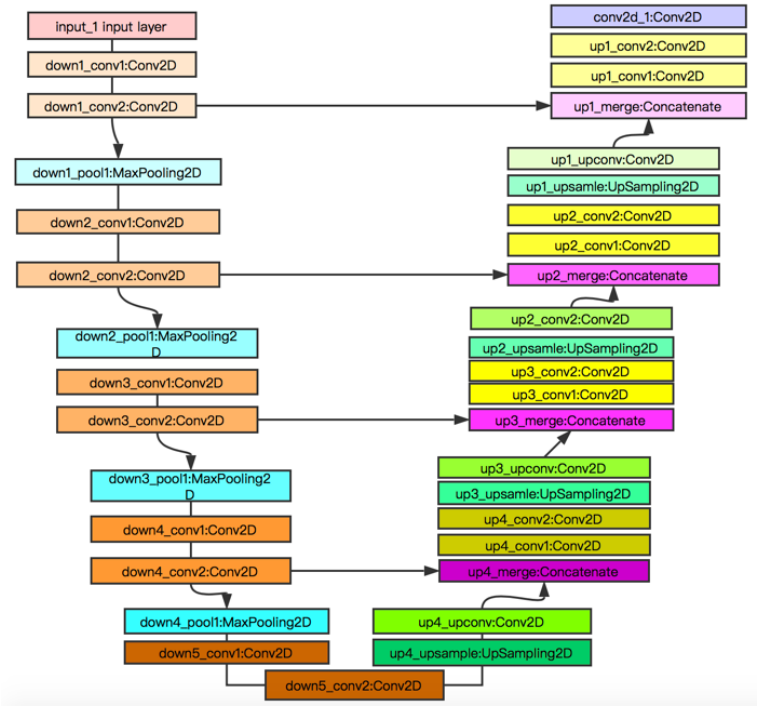


图 21: U-net 结构示意图

大概率 DICE 项直接变为零了。因此我们做了几点改动：一是在方差项前增加系数 α，用于调节方差项，二是将方差改为标准差，即公式改为：

$$\sigma_x^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}|(x_i - \mu_x)|p(x_i, y_j)$$

$$\sigma_y^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}|(y_i - \mu_y)|p(x_i, y_j)$$

损失函数为：$lossfunction = -dice + \alpha \times \sigma_x \times \sigma_y$ 在训练的过程中我们依然发现 DICE 系数无法上升，经过多轮训练最终还是会降为 0，这可以理解：两项很难协同作用，很可能在优先优化方差的时候，区域很快缩小到和结节区域无法重合，因为本问题的特点，结节区域相当小，因此这种困难非常常见。

我们试图进行过比较大范围的 α 的搜索，依然难以搜索到比较好的结果：

alpha	0.0001	0.0003	0.0006	0.0008	0.001	0.002	0.003	0.004	0.005
dice	0.3015	0.0068	0.0059	0.0096	0.0325	0.0104	0.0270	0.0097	0.0000
sigma	54.906	55.612	51.856	32.402	51.278	32.566	51.368	0.029	55.999

alpha	0.006	0.008	0.009	0.01	0.02	0.03	0.05	0.08	0.1
dice	0.0055	0.0349	0.0055	0.0354	0.0055	0.0000	0.0000	0.0000	0.005545
sigma	31.043	55.999	31.247	55.999	0.000	0.001	0.000	55.999	0.0006

图 22: 预测出的区域面积过大

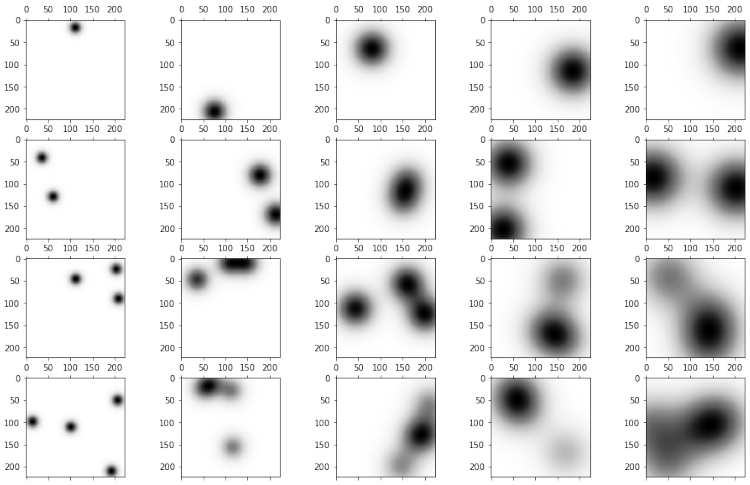


图 23: 正态假设下随机设置均值、方差和个数生成的点族示例

接下来我们进一步想改进损失函数，我们想不筛选 α，而是将预测的区域的标准差处以真实区域的标准差，作为第二项，这样其与 dice 基本在相似的数值尺度，可能会方便优化。但是我们发现模型的优化器会尽量将方差优化的很小，体现在图 23 上就是优化预测区域完全消失。这显然是不好的，因此我们进一步更改了损失函数，从目的上考虑，我们希望预测区域和真实区域的形状和大小都尽量相似，因此我们将第二项进一步修改为预测标准差处以真实标准差的商与 1 的差值的绝对值。即 $loss = -dice + |1 - \frac{\sigma_x(pred)\times\sigma_y(pred)}{\sigma_x(true)\times\sigma_y(true)}|$，希望将预测区域与真实区域的面积预测得尽量相似，同时试图提高 DICE。我们还另外尝试了在 DICE 的分子分母分别增加一个常数项，使其更加“smooth”，我们监控了真实与预测区域的方差及其比值，和训练集与验证机的 DICE，我们发现这次方差可以保证得比较好，但是 DICE 依然是接近零的数值。说明该方法依然行不通。

图 24: 预测出的区域无法重叠

4.2.2.4　继续改进损失函数　我们搜索了相关的资料，想在我们的损失函数上继续改进，我们发现使用 DICE 加上 binary crossentropy 的方案可以在一些数据上取得不错的效果。交叉熵的定义如下：

$$C = -\frac{1}{n}\sum_{x}[ylna + (1-y)ln(1-a)] \tag{4}$$

其中 y 为期望的输出，a 为实际输出。加上交叉熵后我们的损失函数变成：

$$\textbf{Loss} = -\textbf{DICE} + |1 - \frac{\sigma_x(pred) \times \sigma_y(pred)}{\sigma_x(true) \times \sigma_y(true)}| + \textbf{Binary Cross Entropy} \tag{5}$$

4.2.3　使用数据与结果

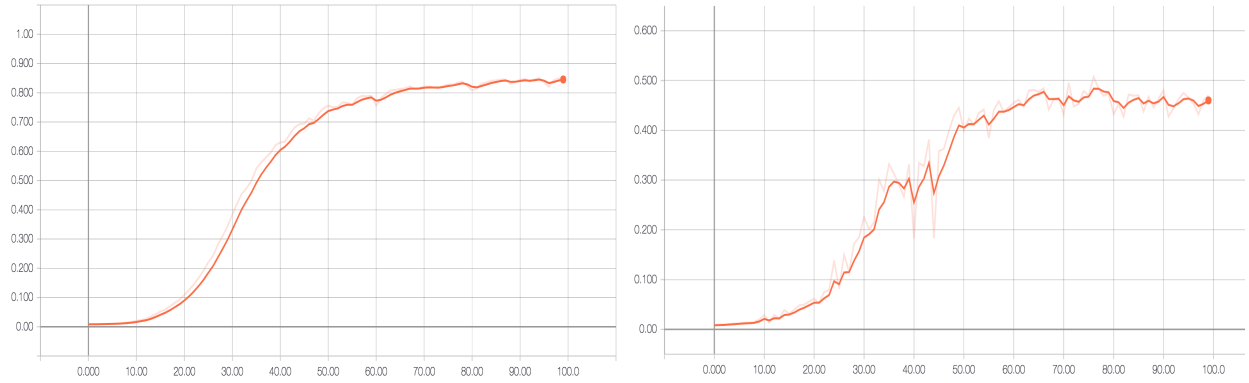
使用 U-net 进行分割的部分我们进行了很多探索，我们尝试了多种损失函数，更换了我们处理的多种数据，包括三层的样本，依据结节直径而定的样本，进行图像仿射变换增加的样本，不进行随机打乱的样本等，进行了多种搭配。这里我们只展示最终取得较好结果的数据和结果。

4.2.3.1　数据集　每个结节中心点所在层以及相邻的两层组成的三层图像为一个样本，经过旋转，平移，拉伸，对称等仿射变换，每个图像生成十张图像，并且经过了 4.1 部分叙述的图像预处理，经过筛选我们获得了包含 14400 张图片的训练集以及 3600 张图片的测试集。

4.2.3.2　结果　使用前面定义的 U-net，以及 4.2.2.4 的损失函数，加上 4.2.3.1 的数据集，我们取得了有所改进的结果。因为训练非常的缓慢，因此我们设置经过大约一百轮的训练即停止，虽然在测试集上结果并不非常理想，但是已经远远好于之前的方法。我们在训练集和测试集上的 DICE 指标可以分别达到 0.84 和 0.41, 如图 24 所示

4.2.4　在寻找到的结节区域进行分类

按照我们的设想，在 U-net 能够寻找到正确结节区域的情况下，我们可以继续做一个分类问题：我们利用训练好的模型在其他负样本层也同样试图寻找一个结节区域，这样我们可以在每层都预测



(a) training set DICE　　(b) valid set DICE

图 25: training set 和 test set 上模型预测 DICE 指标

出一个疑似结节区域出来（某些层可能有多个疑似结节区域），利用之前的标注，我们可以判断某层是否含有结节。如果判断含有结节，我们还可以根据 U-net 找到的区域获得结节中心点的坐标。

分类模型有两种思路，一直是基于传统的特征提取，比如对于 U-net 预测出的结节区域提取其图像特征，与上述的图像处理方法类似，我们可以找到图中的一个或几个疑似结节区域，计算其面积，重心，偏心率，等面积圆的直径等几何特征，以及该层的结节数量等特征。接下来与第一部分类似，我们可以训练一个机器学习模型与随机森林或者 XGBoost 等，依据特征来判断该层是否含有结节。

4.2.4.1　数据与指标　我们使用的数据为 4.2.3 部分的模型在测试集上预测出的三层样本对应的 Mask（3600 张图像），以及我们提取的 3600 张负样本，为了方便起见，我们定义的取法为：在距离结节中心最近的，且不在结节所在的任何一层之内的图像层作为负样本。我们查看模型对于有结节层和无结节层预测是否有结节的相关指标，使用如下几个指标：Accuracy,Precision，Recall,F1-score。其含义如下：

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

$$\textbf{F}_1 = \frac{2 * \textbf{precision} * \textbf{recall}}{\textbf{precision} + \textbf{recall}}$$

其中 TP，TN，FP，FN 分别指真阳性，真阴性，假阳性，假阴性，用于衡量模型对于特定层是否具有结节的预测效果。

type	Accuracy	Precision	Recall	F-1
Random Forest	0.61125	0.6140	0.5991	0.6065
XGBoost	0.6738	0.6776	0.7456	0.7010

4.2.4.2 结果展示 这里我们使用了两种种常见的机器学习模型作为分类器：Random Forest，XGBoost。查看模型对于样本层是否有结节的预测效果如下：

可以看到 XGBoost 的效果略好于 Random Forest，但是由于上一个步骤本身效果并不非常理想，以及我们的特征提取方法比较固定，因此几个指标也并不是很高。

4.2.4.3 讨论 进一步的，我们也可以利用 CNN 来进行特征提取，或者直接对每层进行分类即可，可以进一步提高模型的指标。CNN 对于图像信息的提取能力很强，训练得当的话应该会有更好的分类效果。但是需要注意的是，上面的机器学习方法是通过我们提取的固定特征学习的，虽然结果不是很好，但是也许更加“可靠”，因为我们很难说清楚 CNN 类网络究竟学到了什么。我们也可以训练一个 RCNN 网络，因为疑似结节区域的矩形框可以比较容易标注出来，因此标注步骤不是问题，我们可以用 RCNN 网络来告诉我们每个疑似结节是否是真正的结节。这是接下来可能的工作方向。

我们通过这部分的尝试，学习到了很经典的处理医疗图像的 U-net 网络，并且在损失函数部分进行了很多尝试和思考。我们认为接下来的思路也很明确，有一些公开的方法会使用 3D U-net 进行结节区域的寻找。在网络结构上，需要把 2D 的卷积改为 3D 的卷积，但是考虑到重复出该过程需要很多繁琐的步骤，同时计算资源的消耗会非常大，时间并不允许，而且我们可能没有能力在现有的方法上短期内做出改进，因此我们就把重点放到了之前的工作上，没有来得及进行 3D U-net，3D RCNN 等更多三维层面的网络的探索。

5 总结

通过这次的模式识别大作业，我们学习和巩固了很多模式识别与机器学习的知识，对非常有趣且未来有巨大实用价值的医疗图像领域有了更深入的了解，通过学习各种资料，我们掌握了一系列的图像处理工具，学习了一些机器学习和深度学习的模型，对网络结构和参数调整都有了更多的认识，还利用所学知识对损失函数进行了探讨，虽然时间所限，还有很多工作未能进行，但是我们还是感觉通过这次大作业锻炼了我们诸多的能力，获益匪浅。

分工：陈旭鹏：2D,3D 数据预处理,2D 模型等其他部分内容，部分 3D-CNN 工作，排版。江仕蒙：3D-CNN 部分。

6 References

[1]J. M. Carrillo-de Gea, G. Garc´ıa-Mateos, J. L. Fern´andez-Alem´an, and J. L. Hern´andez-Hern´andez, “A computer-aided detection system for digital chest radiographs,”Journal of Healthcare Engineering, vol.

[2]S. Candemir, S. Jaeger,W. Lin, Z. Xue, S. Antani, and G. Thoma, "Automatic heart localization and radiographic index computation in chest x-rays,"in SPIE Medical Imaging. International Society for Optics and Photonics, 2016, pp. 978 517–978 517. 2016, 2016.

[3]A. Kumar, Y.-Y. Wang, K.-C. Liu, I.-C. Tsai, C.- C. Huang, and N. Hung, "Distinguishing normal and pulmonary edema chest x-ray using gabor filter and svm,"in Bioelectronics and Bioinformatics (ISBB), 2014 IEEE International Symposium on. IEEE, 2014, pp. 1–4.

[4]G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. S´anchez, "A survey on deep learning in medical image analysis,"arXiv preprint arXiv:1702.05747, 2017.

[5]Yuxi Dong, Yuchao Pan, Jun Zhang and Wei Xu, "Learning to Read Chest X-Ray Images from 16000+ Examples Using CNN", Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2017 IEEE/ACM International Conference on.

[6]Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation"