

**本科毕业设计(论文)**

**FINAL PROJECT/THESIS OF UNDERGRADUATE**

**(2017届)**

**基于深度学习的医学图像语义分割算法研究与开发**

**Research and Development of Medical Image Semantic Segmentation Algorithm Based on Deep Learning**

|  |  |
| --- | --- |
| **学　　院** | 医疗器械与食品学院 |
| **专　　业** | 医学信息工程 |
| **学生姓名** | 冯杰 |
| **学　　号** | 1319640722 |
| **指导教师** | 周雷 |
| **完成日期** | 2017年5月 |

# 承诺书

本人郑重承诺：所呈交的毕业论文是在导师的指导下，严格按照学校和学院的有关规定由本人独立完成。文中所引用的观点和参考资料均已标注并加以注释。论文研究过程中不存在抄袭他人研究成果和伪造相关数据等行为。如若出现任何侵犯他人知识产权等问题，本人愿意承担相关法律责任。

承诺人(签名)：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

日期： 年 月 日

# 摘 要

医学图像处理领域有着广阔的应用，而语义分割又是医学图像处理领域中一大核心。语义分割是指将图像分成若干特定的、具备共同性质的区域并提取出感兴趣的指标的技术和过程。语义分割在对病变区域进行定量分析的时候有着重要的意义，是人工智能在医疗临床上迈出的重要一步。传统的分割方法只能针对边缘，针对特定方法进行分割，不能很好地适应临床中多变的情况。本文针对语义分割中传统分割方法死板，针对特定领域不能很好适应的问题提出新的解决方法，尝试使用深度学习技术引入医学图像处理领域，以求能够让软件在达到基本分割精度要求的前提下，能够自我学习，不断提高自己分割精度的效果。本文对人工智能领域中的深度学习技术进行了简要介绍，给出了对其中的重要概念和重要公式，并在此基础上拓展，介绍三个在图像语义分割领域中较为典型的模型。本次课题使用SegNet、FCN、DeepLab三个网络模型进行试验，分割CT图像中的肝脏和肾脏部位，并给出实验所需的工具和核心代码。同时会对结果进行可视化和计算精度进行评估，完成对课题的研究，最后对实验结果进行总结，分析课题研究中尚存在的问题和对进一步对深度学习和医学图像语义分割的研究方向给出一点建议。

**关键词**：医学图像 语义分割 深度学习 人工智能

# ABSTRACT

Medical image processing has a broad application, and semantic segmentation is the medical image processing field in a major core. Semantic segmentation is the technique and process of dividing an image into a number of specific, unique areas and extracting the target of interest. Semantic segmentation plays an important role in the quantitative analysis of the lesion area, which is an important step in the clinical practice of artificial intelligence. The traditional segmentation method can only be aimed at the edge, for a specific method of segmentation, can not be well adapted to the clinical changes in the situation. In this paper, we propose a new solution to the problem that the traditional segmentation method is not suitable for the specific domain, and try to use the deep learning technology to introduce the field of medical image processing in order to make the software meet the requirement of basic segmentation precision, To self-learning, and constantly improve the accuracy of their own segmentation effect. This paper gives a brief introduction to the deep learning techniques in the field of artificial intelligence, gives some important concepts and important formulas, and on this basis, it introduces three typical models in the field of image semantic segmentation. This topic uses SegNet, FCN, DeepLab three network model to test, segment the CT images of the liver and kidney parts, and give the necessary tools and core code. At the same time, the results will be evaluated and the accuracy of the evaluation is carried out to complete the research on the subject. Finally, the experimental results are summarized, the existing problems in the research are analyzed, and the suggest and research direction of further study of semantic segmentation of deep learning and medical image is given..

**KEY WORDS**: Medical Image Semantic Segmentation Deep Learning Artificial Intelligence

# 目 录

[摘 要](#_Toc483402703)

[ABSTRACT](#_Toc483402704)

[第1章 绪论 1](#_Toc483402706)

[1.1 语义分割的背景研究 1](#_Toc483402707)

[1.2 医学图像分割现状 1](#_Toc483402708)

[1.3 深度学习技术的尝试 2](#_Toc483402709)

[1.4 本文的内容和安排 2](#_Toc483402710)

[第2章 深度学习技术 5](#_Toc483402711)

[2.1 神经网络基本介绍 5](#_Toc483402712)

[2.2 卷积神经网络介绍 7](#_Toc483402713)

[2.2.1 卷积 7](#_Toc483402714)

[2.2.2 池化 9](#_Toc483402715)

[2.2.3 全连接 10](#_Toc483402716)

[2.2.4 非线性变换 10](#_Toc483402717)

[2.3 技术工具介绍 11](#_Toc483402718)

[2.3.1 Ubuntu 11](#_Toc483402719)

[2.3.2 Caffe 11](#_Toc483402720)

[2.3.3 CUDA 12](#_Toc483402721)

[2.3.4 CuDNN 12](#_Toc483402722)

[2.3.5 Python 12](#_Toc483402723)

[第3章 基于深度学习的图像语义分割 13](#_Toc483402724)

[3.1 SegNet网络模型 13](#_Toc483402725)

[3.1.1 Convolution 13](#_Toc483402726)

[3.1.2 Batch Normalisation 13](#_Toc483402727)

[3.1.3 pooling&Upsamping 16](#_Toc483402728)

[3.1.4 Deconvolution 16](#_Toc483402729)

[3.1.5 softmax 16](#_Toc483402730)

[3.1.6 output 17](#_Toc483402731)

[3.2 实验方案与安排 17](#_Toc483402732)

[3.2.1 数据筛选 17](#_Toc483402733)

[3.2.2 数据预处理 18](#_Toc483402734)

[3.2.3 构建网络模型 19](#_Toc483402735)

[3.2.4 训练模型 24](#_Toc483402736)

[3.2.4 预测（分割） 24](#_Toc483402737)

[3.2.5 评估 27](#_Toc483402738)

[3.2.5 FCN网络模型 28](#_Toc483402739)

[3.2.5 DeepLab网络模型 30](#_Toc483402740)

[第4章 总结 33](#_Toc483402741)

[4.1 课题完成情况 33](#_Toc483402742)

[4.2 尚存在的问题 33](#_Toc483402743)

[4.3 进一步研究的方向 33](#_Toc483402744)

[参考文献 35](#_Toc483402745)

[附录1 37](#_Toc483402746)

[附录2 41](#_Toc483402747)

[致 谢 45](#_Toc483402748)

# 第1章 绪论

## 1.1 语义分割的背景研究

计算机视觉中有许多基本问题，其中重要的之一是图像的语义分割,另外两大核心研究问题是图像分类和物体检测，其中当属语义分割这个任务最具有挑战性。语义分割的目标是对图像的每个像素点进行分类,将图像分割为若干个视觉上有意义的或感兴趣的区域，最终获得一副具有像素语义标注的图像,以利于后续的图像分析和视觉理解[1]。目前，语义分割是计算机视觉和模式识别领域非常活跃的研究方向，并在很多领域具有广泛的应用价值。例如，在新兴领域，经过对人体的语义分割可以定位出人脸、躯干、着装等信息，从而协助网民在互联网购物过程中完成自动试衣等功能；在自动驾驶领域，通过对车体前方场景的语义分割可以准确定位路线、车体和行人等场景或物体信息，从而提升自动驾驶的安全性[2]

## 1.2 医学图像分割现状

伴随着科技的发展，各种影像设备的一直在普及，全世界的医疗机构每天都要产生海量的医学影像资料，使得医学影像成为医学技术当中发展最快的领域之一。成像技术已经成为一种无侵害观察器官组织解剖结构的主要手段，例如计算机断层扫描CT，磁共振成像MRI、正电子发射断层成像PET，单光子发射型计算机断层扫描ECT等。伴随着单位存储的像素提高，图像的细节越发复杂。传统分割算法无法跟上计算机处理能力的不断发展，反而所需要的时间不断增加，通用性也越来越差。如何高效而又精确的在图像中分割出感兴趣的器官成为国际上一个前沿的研究命题。因此，影像类的专家一直都非常重视医学图像处理技术，而图像处理技术中的一个经典难题就是医学图像分割，不仅是进行图像分析和识别首要解决的问题，同时也是制约医学图像处理中可视化、不同模态图像配准和融合、三维重建等相关技术发展与应用的瓶颈。

医学图像分割目的在于将病变区域从图像中分割出来，从而便于后续的定量分析。由于医学成像系统具有多样性，使得成像图像易出现模糊、噪声等不期望特征，因此分割的准确性对医生判断疾病的真实情况并做出正确的诊断至关重要。医学图像分割的应用非常广泛，目前国内外的研究主要集中在分割各种细胞、组织和器官。例如对核磁共振颅脑图像的分割，就可以清晰地描绘出颅脑各个解剖结构的边界如灰质、白质和脑脊液等。在临床诊断，病理分析以及人体器官三维重现等方面，医学图像分割在里面发挥了至关重要的作用[3]。

从图像分割提出至今，已经诞生了各种各样的实现方法。传统的分割方法大致有三种类型，一是基于特定理论的分割，二是基于边缘的分割，三是基于区域的分割。由于图像的类型繁多，图像数据常常又具有不确定性，所以说并不存在一种通用的分割算法可以按照人的理解准确地分割任意一幅图像, 常常发生分割错误的现象。有些分割算法可以直接用于一些图像，而有些算法只能分割特殊图像，也就是说在一类图像中行之有效的分割算法可能在另一类图像中分割的却一团糊涂。因此，对于图像分割技术的研究一直是学术领域的热点项目，成功的分割有助于进一步识别和理解图像。不同的图像有不同的特点，选择合适的算法从而使分割的效果令人满意是十分困难的。因此，人们在图像分割算法研究的道路上从未停止过脚步，期望能够探索找到较好的分割算法并得到较为满意的分割结果，以更好的服务于图像分析理解任务。

## 1.3 深度学习技术的尝试

[深度学习](http://baike.baidu.com/item/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0)的概念于2006年的3篇革命性的论文被引领人工智能领域。深度学习本身源于对[人工神经网络](http://baike.baidu.com/item/%E4%BA%BA%E5%B7%A5%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C)的研究。其中一种常见的深度学习结构是含多隐层的多层感知器。深度学习的基本工作方式是通过分析和组合低层特征，推导和映射形成更加抽象的高层特征，以此来表示属性类别，从而通过高层特征的研究发现具有分布式特征表示形式的数据的特点标识。深度学习成为了人工智能研究中的一个新的领域，其研究目的在于建立、模拟一种类似于人类大脑进行分析学习的计算机表示形式，即神经网络，它模仿人脑分析学习事物的方式和机制来解释数据，例如语言文字，音频，照片等。深度学习技术的引入，是医学图像分割领域的一次尝试，它通过大量数据事先训练模型以提高分割精度，这种新型的研究和实现方式具有广阔的前景，可以用来纠正某些分割中的错误。

## 1.4 本文的内容和安排

本课题主要通过CT产生的内脏成像作为样品，通过计算机技术提取图像中的肝脏和肾脏作为研究目标。SegNet网络模型[4]、DeepLab网络模型[5]以及全卷积神经网络（FCN）[6]模型做出实例分析，通过比较得出一个最好的分割算法，将其应用在医学图像分割中，从而实现对肝脏和肾脏提取的完全分割，满足临床诊断和治疗辅助需求。相信这项技术具有很大的医疗研究的意义。实际中，图像分割的结果主要通过Python 的PIL库实现，结合深度神经网络算法和Caffe框架完成对图像的提取与解析等功能，体现了计算机技术在医学图像发展中的重要作用。

第一章绪论，简介了语义分割的研究背景、发展现状以及深度学习技术的简介。

第二章主要说明了卷积神经网络有关的概念和实验方法，分别成为引出卷积（Convolution）、池化（Pooling）、全连接（Full Connection）和非线性变换（Non-linear Transformation）的基础理论，这四个基本操作构成了深度卷积神经网络的基石，并是之后构建网络模型和对模型进行优化的基础。另外介绍了介绍本文所使用的工具和技术背景，本次实验一个开源的深度学习框架Caffe，结合Nvidia的CUDA通用GPU计算平台进行图像的语义分割，利用Python可视化技术对数据进行预处理和最后实验结果的可视化。具体内容将会一一介绍。

第三章为本文的重点部分，主要介绍了基于深度学习的医学图像语义分割的相关理论以及演化过程，并且通过实验对SegNet、DeepLab、FCN网络模型的实现和应用进行了分析，对选取的参数进行了实验仿真。最后通过评估三个网络模型的实验结果对三个模型进行精确率(Precision)和召回率(Recall)的计算，并比较优劣。

第四章为总结，对本次实验所获得的三个模型的分割结果进行比较和评估，同时简要分析课题中仍然存在的问题，提出进一步开展研究的见解与建议。

# 第2章 深度学习技术

## 2.1 神经网络基本介绍

神经网络是一种模仿生物神经网络(动物的中枢神经系统，特别是大脑)的结构和功能的数学模型或计算模型，用于对函数进行估计或近似[7]。神经网络由大量的人工神经元联结进行计算。图2-1是神经元的基本示意图：

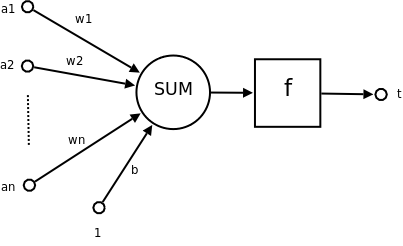


图2-1 神经元基本示意图

其中：

* a1~an为输入向量的各个分量。
* w1~wn为神经元各个突触的权值。
* b为偏置。
* f为传递函数，通常为非线性函数。
* t为神经元输出。

数学表示

* 为权向量，为的转置。
* 为输入向量。
* b为偏置项
* f为传递函数。

大多数情况下人工神经网络能在部信息的输入或者刺激的基础上改变内部结构，是一种能够做到自适应的系统结构[8]。现代神经网络是基于非线性统计的一种数据建模工具，图2-2为一个全连接神经元网络示意图。典型的神经网络一般具有以下三个部分：

* **结构（Architecture）：**结构指定了网络中的变量和它们的拓扑关系。例如，神经网络中神经元连接的[权重](https://zh.wikipedia.org/wiki/%E6%9D%83%E9%87%8D)（weights）和神经元的激励值（activities of the neurons）可以作为神经网络的变量。
* **激励函数（Activity Rule）：**大部分神经网络模型具有一个短时间尺度的动力学规则，叫做激励函数，也叫做激活函数，它是用来将神经元的输入进行一些特定值的转换，如将连续的数值型转换成[0,1]输出，也就是说定义神经元如何根据其他神经元的活动来改变自己的激励值。一般激励函数依赖于网络中的权重（即该网络的参数）。
* **学习规则（Learning Rule）：**学习规则指定了网络中的权重如何随着时间推进而调整。这一般被看做是一种长时间尺度的动力学规则。一般情况下，学习规则依赖于神经元的激励值。它也可能依赖于监督者提供的目标值和当前权重的值。

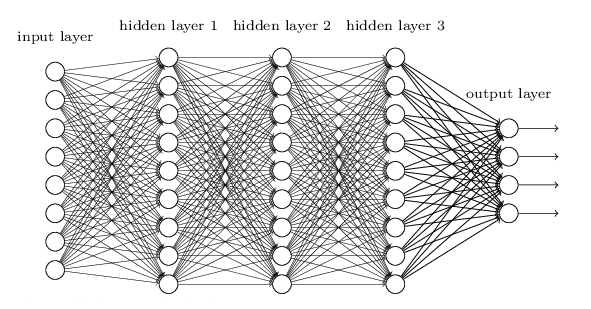


图2-2 神经元网络

这个神经网络的特点就是，上层每一个神经元都和下一层的每个神经元有着输入输出的连接。

神经网络的构筑理念是受到生物（人或其他动物）大脑神经网络功能的运作启发而产生的。人工神经网络通常是通过一个基于数学统计学类型的学习方法（Learning Method）得以优化，所以人工神经网络也是数学统计学方法的一种实际应用,但是传统神经网络具有权值过多，收敛慢，模型难以训练的缺点。

## 2.2 卷积神经网络介绍

卷积神经网络是最近发展起来，并引起广泛重视的一种高效识别方法。20世纪60年代，Hubel和Wiesel在研究过程中发现，猫脑皮层中用于局部敏感和方向选择的神经元具有独特的生理结构，他们发现这种特殊的网络结构可以有效地降低反馈神经网络的复杂性，由此提出了卷积神经网络（Convolutional Neural Networks简称CNN）。

一般来说，CNN的基本结构包括两层：

1. 特征提取层：每个神经元的输入与前一层的局部接受域相连，并提取该局部的特征。一旦该局部特征被提取后，它与其它特征间的位置关系也随之确定下来。
2. 特征映射层：网络的每个计算层由多个特征映射组成，每个特征映射是一个平面，平面上所有神经元的权值相等。特征映射结构的激活函数需要具有影响函数核小的特性，一般来说采用sigmoid函数作为卷积网络的激活函数，使得特征映射具有位移不变性。此外，由于一个映射面上的神经元共享权值，能够大大减少网络自由参数的个数。卷积神经网络中的每一个卷积层都紧跟着一个用来求局部平均与二次提取的计算层，这种特有的两次特征提取结构减小了特征分辨率。

CNN主要用来识别二维图形，而二维图形具有位移、缩放及其他形式扭曲不变性。因为CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显示的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，即在2.2.1中论述的权值共享，所以网络具有可以并行学习的优点，这也是全连接神经网络所不具备的，而卷积网络独有的优势。卷积神经网络的这种优越性，使其在语音识别和图像处理方面有着极高的效率和精确度，其结构上更接近于实际的生物大脑神经网络，模拟生物的思维方式，权值共享降低了网络的复杂性，特别是图像一般都是以多维输入向量的形式保存，而这种多维输入向量可以直接输入网络这一特点降低了特征提取和分类过程中数据重建的复杂度[9]。

深度学习在图像识别领域取得了巨大成功，其原因得归功于计算机硬件的发展，硬件支撑起整个复杂模型的运算，还有大规模标记图像的出现，Stanford大学Li等发起的Image Net大规模视觉识别挑战（Image Net Large Scale Visual Recognition Challenge，ILSVRC）提供了上百万张带有标签的图像，很大程度上促进了深度学习在这个领域的前进。相对于传统方法，深度学习在图像语义分割领域取得的巨大进步的一个关键因素，在于深度卷积神经网络可以通过训练数据自动学习特征。卷积神经网络学习时主要包含了4种基本操作：卷积（Convolution），池化（Pooling）、全连接和非线性变换。

### 2.2.1 卷积

在泛函分析中，卷积是通过两个函数和生成第三个函数的一种数学算子，表征函数与经过翻转和平移的的乘积函数的曲边梯形的面积。如果将参加卷积的一个函数看作区间的指示函数，卷积还可以被看作是“移动平均”的推广。设(x),g(x)是R上的两个可积函数，作积分：

可以得知，在定义域，几乎所有的x总会符合这个积分。同时我们定义一个新函数h(x)，称为函数f与g的卷积，记为h(x)=()(x)。我们可以轻易验证：()(x) = ()(x)，并且()(x)仍为可积函数。这就是说，把卷积代替乘法，空间是一个代数，甚至是[巴拿赫代数](https://zh.wikipedia.org/w/index.php?title=%E5%B7%B4%E6%8B%BF%E8%B5%AB%E4%BB%A3%E6%95%B0&action=edit&redlink=1)。虽然在上述过程中为了方便理解和假设条件，只是将卷积作为一种运算符号，理论上并不需要对函数有特别的限制条件。但是一般来说，会要求这个函数至少是可测函数（measurable function）。

函数和的卷积记作，是定义在上的可测函数，它是其中一个函数翻转并平移后与另一个函数的乘积积分，是一个对平移量的函数，也就是

如果函数不是定义在上，可以把函数定义域以外的值都规定成零，这样就变成了一个定义在上的函数。

对于定义在上的函数和，卷积定义为：

这里一样把函数定义域以外的值当成零，如果本来不是所有整数域的话，也可以扩展函数到所有整数上。当的支撑集（support）为有限长度M，上式会变成有限和：

通过这种卷积操作，能大大减少神经网络需要学习的参数。如果我们准备一张10001000的像素的图像，假设模型结构中有一百万个隐层神经元，模型结构为全连接的网络（每个隐层神经元都与图像的每一个像素点相连），这样就有100010001000000=10^12个连接，也就是10^12个权值参数，这是非常庞大的，几乎是无法进行学习的。然而就像人的大脑，并不是将整张图片所有像素印刻在脑中，而是通过一个局部和范围的特点，来反映整张图像，图像的空间联系是局部的，那么网络结构中每一个神经元都不需要对全局图像做感受，在某一层中的神经元只感受局部的图像区域，然后将特征输入到更高层，由更高层将这些抽象的，感受不同局部的神经元综合起来就可以反映全局图像的特点了。这样，我们就达到了减少连接目的了，也就是降低了神经网络需要训练的权值参数的数目。示意图见图2-3。

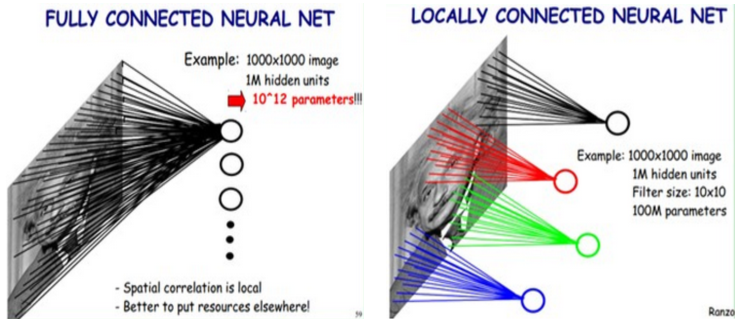


图2-3 图像卷积

假如设定一个神经元能够局部感受的大小是1010，则隐层中各个神经元只需要与这1010的局部图像相连接，那么按照上述假设的网络有一百万个隐层神经元，只有10101000000个即一亿个连接，即10^8个参数，对比原来的10^12个参数减少了四个数量级。但是即使如此，网络中仍然存在大量参数，对于学习来说仍然是不现实的，而进一步减少参数的方法就是局部权值共享。

由以上设定可知，隐含层的每一个神经元都连接1010的图像区域，也就是说每一个神经元存在1010=100个连接权值参数。权值共享就是设定每个神经元感受局部时的方式或者说对应的这100个参数是相同的，也就是说着100个参数同时应用于所有的神经元（每个神经元用的是同一个卷积核去卷积图像）。使用权值共享的设定后，只需要训练100个参数即可，大大减少了参数量，解决了当网络层数过深时由于参数过多无法训练的问题。

因此可以看到，通过卷积和权值共享，基本解决了由于参数量庞大导致无法训练的问题。而最终我们的目的是进行像素的分类，如何将大量的像素进行缩放到有限的类别中，则需要使用到池化操作。

### 2.2.2 池化

一般来说，通过卷积层后，网络会得到一种代表图像特征的输入，称之为特征图（feature map），在得到了这些特征 (features) 之后，下一步就是通过这些特征去做进一步的分类。理论上讲，人们可以用所有卷积层的输出特征去训练我们的分类器，例如使用softmax分类器，但这样做又会面临大量参数不可控带来的问题。例如：对于一个 9696 像素的图像，假设我们的卷积层有400个卷积核,即输出为400个通道，每一个卷积核的大小为88，，那么每一个通道将会得到一个(96−8+1)(96−8+1) = 8989 维的卷积特征，由于有 400 个通道，所以每个样例 (example) 都会得到一个 8989400 = 3,168,400 维的卷积特征向量。要训练一个拥有超过3百万特征输入的分类器非常困难，而且由于参数过多，参数对于分类的影响敏感，容易出现过模型针对特殊数据工作良好，对于一般数据效果不佳，即过拟合 (over-fitting)的现象。

回到图像本身来说，我们认识一副图像总是通过局部去观看全局的，并且在空间上各个部分是有联系的，称之为 “静态性”的属性。也就是说，我们观看一副图像时在一个图像区域观察到的特点极有有可能在另一个区域也出现，并且根据空间上的联系，会很自然地推断过去。因此，为了表示大的图像特征，一个自然而然的想法就是使用聚合统计的方式，根据图像的不同位置的特征，来推理和分类。例如，人们可以计算一个特定大小的图像范围，求得该范围的某个指定的特征。一般使用的是计算该图像范围内所有像素点的平均值或最大值。这些概要统计出来的结果不仅不会丢失低纬度的图像特征，同时还可以防止过拟合现象，改善分类的结果。这种对一个区域进行基于聚合统计的操作就叫做池化 (pooling)。池化就是将输入的图像划分为若干个矩形区域，对每个子区域输出最大值或者平均值，一般较多使用最大值池化，见图2-4。

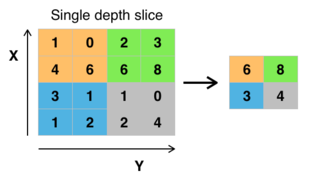


图2-4 最大值池化

### 2.2.3 全连接

同传统神经网络中的层相似，在全连接层中下一层神经元的输出同上一层所有神经元的输入都有关．通过全连接层可以使得网络的参数在训练样本上快速收敛。

### 2.2.4 非线性变换

人脑对客观世界的理解并不是线性的，而是一种复杂的非线性映射。即2-1提到的神经网络的三个基本部分之一的激励函数，用户变换神经元间的输入输出。因此，神经网络在设计中往往会通过非线性激活函数来模拟人脑的非线性认知行为．常用的非线性激活函数主要包括Sigmoid函数、Tanh函数和线性整流函数（Rectified Linear Units, ReLU），定义为：

Sigmoid:

Tanh:

ReLU:

见图2-5，左边为Sigmoid函数曲线，右边为ReLU函数曲线由于Sigmoid函数在0附近有坡度，会产生非跃迁态，因此本次实验采用ReLU函数。

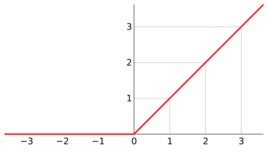
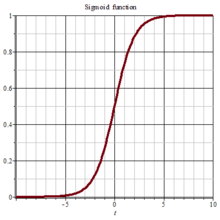


图2-5 Sigmoid曲线和ReLU曲线

## 2.3 技术工具介绍

本次毕业设计采用了Ubuntu14.04 + Caffe + CUDA8.0 + CuDNN + Python进行研究，下面对这些工具进行一一介绍。

### 2.3.1 Ubuntu

Ubuntu是面向普通用户为主的，桌面和易用性良好的Linux发行版，这里采用14.04版本，主要是因为此版本稳定并且与其他所需要使用到的工具兼容性良好，避免因为兼容性问题产生不必要的bug。

### 2.3.2 Caffe

Caffe是一个清晰，可读性高，快速的深度学习框架。Caffe包含了CNN使用最广泛的库。它诞生于伯克利视觉和学习中心（BVLC），通常用于学术研究和学习，拥有良好的结构设计和更快的速度是它相对于其它框架的优势。同时Caffe实现了多模式的工作方式，能够同提供[CPU](https://zh.wikipedia.org/wiki/CPU)和[GPU](https://zh.wikipedia.org/wiki/GPU)两种计算方式，为了高效率的运作，底层代码由贴近硬件层的[C++](https://zh.wikipedia.org/wiki/C%2B%2B)实现，并对用户使用做了良好的封装，用户可以使用它提供的Python和[MATLAB](https://zh.wikipedia.org/wiki/MATLAB)的接口进行快速上手。

Caffe非常适合作为深度学习的工具，它具有如下特点：

1. 速度快，这是非常重要的优势。一般深度网络模型的训练耗时都非常久，因此一个快速得出训练结果的框架便具备好的使用优势。MKL、OpenBLAS、cuBLAS、Nvidia CuDNN等计算库都已经被Caffe集成进来。
2. 适合做特征提取，实际上适合做二维图像数据的特征提取。
3. Caffe完全开源，遵循BSD-2协议。
4. Caffe提供了一整套方便用户使用的脚本和工具，包含常见的模型训练、预测脚本、微调参数工具、发布、数据预处理，以及良好的自动测试结构。
5. Caffe在国内外有比较活跃的社区。

### 2.3.3 CUDA

CUDA(Compute Unified Device Architecture)，是显卡厂商[NVIDIA](http://baike.baidu.com/item/NVIDIA)推出的运算平台。 CUDA™是一种由NVIDIA推出的通用[并行计算](http://baike.baidu.com/item/%E5%B9%B6%E8%A1%8C%E8%AE%A1%E7%AE%97)架构，该架构使[GPU](http://baike.baidu.com/item/GPU)能够解决复杂的计算问题。 它包含了CUDA[指令集架构](http://baike.baidu.com/item/%E6%8C%87%E4%BB%A4%E9%9B%86%E6%9E%B6%E6%9E%84)（[ISA](http://baike.baidu.com/item/ISA)）以及GPU内部的并行计算引擎。GPU对图像处理和计算较CPU有着天然的优势。

### 2.3.4 CuDNN

Nvidia厂商在CUDA的平台基础上，专门针对深度学习设计出一套GPU计算加速库，可以运行在Linux，Windows，Mac上，CuDNN目前支持的深度学习框架包括Caffe，TensorFlow，ConvNet等。

### 2.3.5 Python

Python是业界一种常用的解释型计算机程序设计语言，拥有丰富的科学计算函数库和图像处理库。Python相对于其它语言的最大的优势就是简洁的语法以及社区产生的众多扩展库，其整个语言体系的特点就是精炼和优雅，能够使用比较少的代码快速完成一些功能，十分适合工程技术、科研人员进行数据建模，梳理实验数据、制作图表，甚至开发科学计算应用程序。本次毕业设计使用Python对实验数据进行预处理和结果的可视化。由于Python2和Python3版本差异较大，许多代码不兼容，本次实验的版本定为采用Python2.7。

# 第3章 基于深度学习的图像语义分割

## 3.1 SegNet网络模型

SegNet是剑桥大学提出的一种图像语义分割神经网络结构，初衷是为了实现汽车的自动驾驶，[机器人](http://lib.csdn.net/base/robot)的智能导航。SegNet提出的同时还在github上公布了基于Caffe的结构模型文件和开放的源代码。剑桥大学通过研究FCN网络模型，并对VGG-16网络结构做出了修改，最终提出了成功的SegNet语义分割网络。

SegNet网络结构如图3-1所示，左边的RGB Image为待分割图片，右边的图片为输出分割图像。SegNet是一个对称的网络结构，有10组隐藏层组成，左边5组中蓝色的卷积层用来提取高维特征，绿色的池化层作为分割并使图片变小，SegNet作者将这个部分称之为Encoder，右边5组中红色的上采样层使之前缩小了的图像还原变大，蓝色的反卷积层使得图像被提取的特征得以还原，与Encoder对应作者将这个部分称之为Decoder，最后通过Softmax损失层计算特征的分类概率，输出不同分类的最大值。这就是大致的SegNet过程，下面对这个过程里面使用到的方法进行介绍。

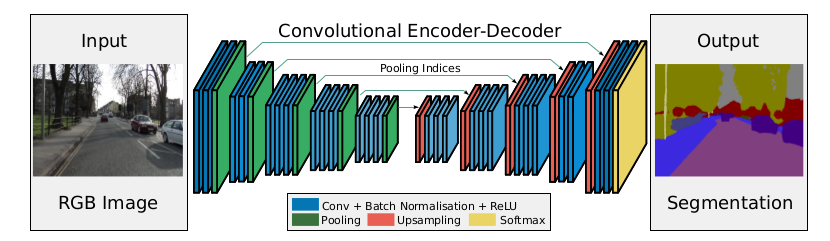


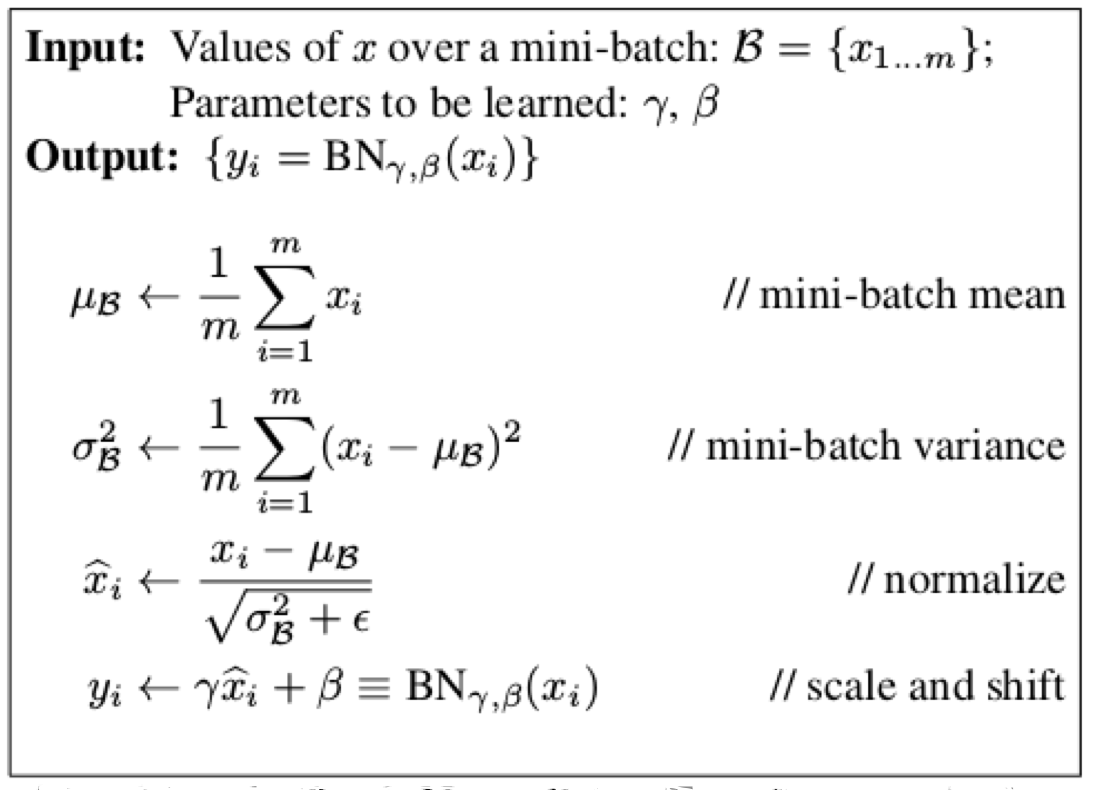
图3-1 SegNet网络模型

### 3.1.1 Convolution

在Encoder过程中，通过卷积来减少神经元全连接数量，也就是减少训练参数，提取特征值。SegNet为了使卷积操作之后，图像大小不会改变，使用的卷积为same卷积。SegNet的训练过程中图像不通过卷积来改变大小，而都是通过池化或者上采样来进行改变。在Decoder过程中，为了使在之前卷积过程丢失的信息还原，先使用上采样变大图像，然后用过反卷积作用于变大后的图像来丰富信息，反卷积也同样使用same卷积。SegNet中的卷积操作与传统CNN的卷积并没有区别。

### 3.1.2 Batch Normalisation

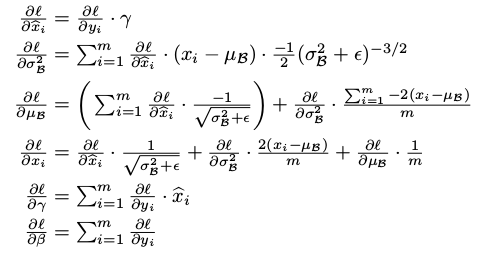
Batch Normalization即为批量标准化，是一种由google提出的训练优化方法，其目的是改善网络学习速率较慢的问题。Noralization的过程如下所示。



输入：输入数据x1..xm  
 计算过程中可以看到：  
 1.求数据均值；  
 2.求数据方差；  
 3.数据进行标准化  
 4.训练参数γ，β   
 5.输出y通过γ与β的线性变换得到原来的数值   
训练时：   
 1.向前传播，BN层不改变特征值，只保存着上一层输入特征值的均值和方差，即对卷积和的特征值进行了标准化操作，特征值输出到下一层卷基层时仍然是当初流入BN的特征值。   
 2.向后传播，根据保存在BN层中的均值与方差，从输出到输入通过链式求导法则反向进行计算，求得每个卷积层的梯度从而计算出当前的学习速率。   
测试时：

每个BN层都会将求总体均值于方差的操作应用于整个训练集的数据，假设有一张测试图像进入BN层，BN层会进行输入权值的统计并计算出均值与方差，然后根据训练集中整体的无偏估计计算BN层的输出。因此在测试阶段，卷积层的输出会受到BN层的修改，因此激活层ReLU的输入也受到影响。

在训练过程中，正向传播BN不会改变上一层的输出，只记录下γ与β，在反向传播的时候，根据已经求得保存的γ与β通过链式求导方式，求出学习速率以至改变权值。具体过程如下：



BN解决的问题是梯度消失和梯度爆炸。梯度消失可以见于sigmoid函数中，从图3-2可以明显看见，虽然sigmoid函数使得x输入后得到的结果在[0,1]之间，但是x过大或者过小后，输出相对于输入x的改变会变得很小，也就是输出受x改变的影响变得不明显，那么经过sigmoid函数的输入范围就很小了。

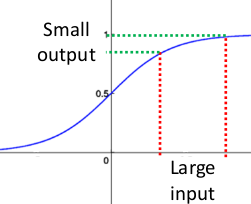


图3-2 sigmoid梯度消失

同理在各种神经网络中，如果上一层的激活输出很大，可以预见其梯度就会很小，而在反向传播过程中，其权值学习速率就是而其斜率（梯度），所以就会如图3-3所示，出现一种现象，就是浅层的权值变化小，处于基本慵懒（不学习）状态，后面几层的权值一直在变化，一直在学习的现象，这就是梯度消失。

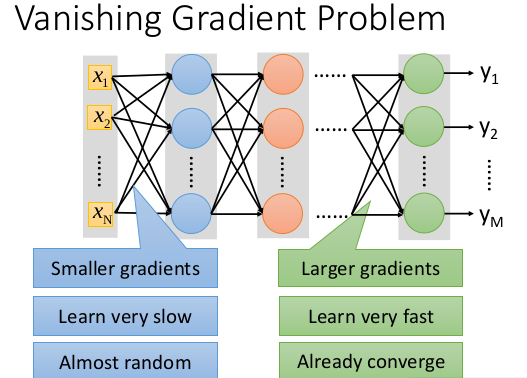


图3-3 深浅学习速率不一致

关于梯度爆炸，根据链式求导法， 第一层偏移量的梯度=每一层的激活层斜率乘以该层的权值 ，假如激活层斜率均为最大值0.25，所有层的权值为100，这样梯度就会指数增加。

### 3.1.3 pooling&Upsamping

池化(Pooling)在CNN中是使图片缩小一半的手段，如第2章所介绍。在SegNet中的Pooling与其他Pooling多了一个index的功能，也就是每次Pooling，都会保存通过max选出的权值在2x2 filter中的相对位置。Upsamping就是Pooling的逆过程，Upsamping使得图片变大2倍。我们清楚的知道Pooling之后，每个filter会丢失了3个权重，这些权重是无法复原的，但是在Upsamping层中可以得到在Pooling中相对Pooling filter的位置。所以Upsampling中先对输入的特征图放大两倍，然后把输入特征图的数据根据Pooling indices放入。

### 3.1.4 Deconvolution

上采样之后，图像已经大小已经被放大，但是除了在池化时被记住位置的权值数据，其他位置的权值均为0，因为数据在池化过程中已经丢失。因此，需要通过反卷积操作来填补丢失的内容。同时反卷积必须保证图像大小不变，只填充数据。

### 3.1.5 softmax

softmax层是用来计算分类概率的一层，其本质是一个损失层。在SegNet网络模型中我们使用SoftmaxWithLoss函数来对像素点进行最大似然指的分类。其中SoftmaxWithLoss是最常用的计算损失的函数，损失即为预测值和实际标注值的差。

softmax损失层计算方式是，其输入的每一项的softmax结果的Logistic损失。它在概念上与softmax层相同，后跟多项Logistic损失层，但提供了更数值稳定的梯度。具体来说，softmax函数的定义如下：

它在逻辑回归里起到的作用是将线性预测值转化为类别概率：假设是第个类别的线性预测结果，带入softmax的结果其实就是先对每一个取指数函数变成非负，然后除以所有项之和进行归一化，现在每个就可以解释成观察到的数据属于类别的概率，或者称为似然。

在逻辑回归的目标函数中，求出每个数据属于类别的概率，并取最大的概率为该数据的分类结果，这就是最大似然原则。假设数据所对应的类别为，则根据我们刚才的计算最大似然就是要最大化的值。通常来说处理上是使用最小化的值，这两者的结果在数学上是等价的。具体定义如下：

而SoftmaxWithLoss其实就是把两者结合到一起，只要把的定义展开即可：

### 3.1.6 output

在SegNet网络模型的输出中，最后一个层为损失层，它会根据每一个像素在所有类别最大的概率，输出其的分类类别，记为该像素的label，最终完成所有像素的分类，通过像素的类别不同，我们即可以将图像进行分割。

## 3.2 实验方案与安排

本次毕设实验有如下几个步骤：

1. 数据筛选
2. 数据预处理
3. 编写prototxt文件构建网络模型
4. 训练模型
5. 用训练好的模型进行预测
6. 评估。

### 3.2.1 数据筛选

本次实验数据为3.46GB，约9329张真实病人的CT检查图像，包含CT图像和对应的手工标记出待分类区域的groundtruth图像。

本次筛选的条件是，图像较为典型，能肉眼明显观测出肝脏和肾脏区域，图像亮度无明显差距。

最后筛选出较为典型的共650张CT图像+650张对应的groundtruth图，样本见图3-4，左边为原CT图，右边为groundtruth图。

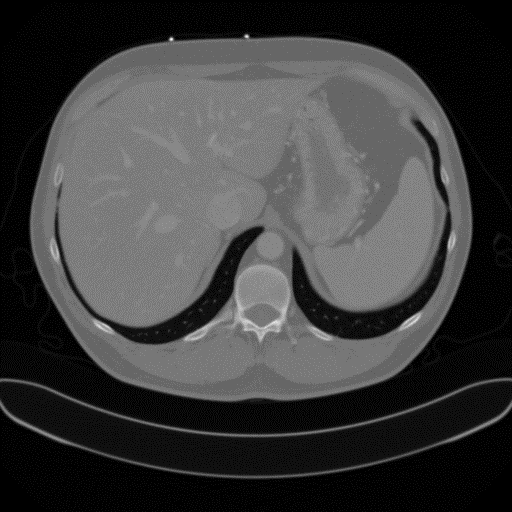
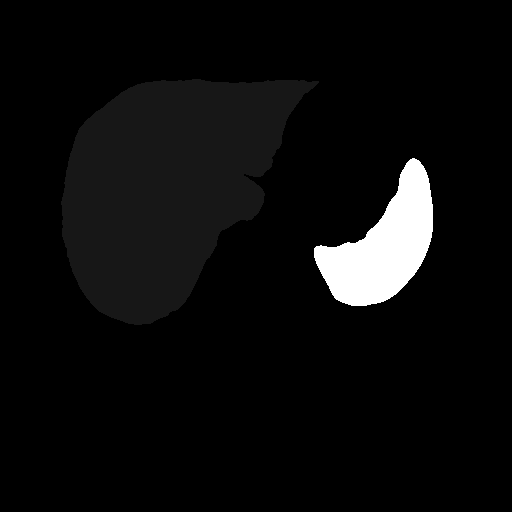
 

图3-4 数据样本

### 3.2.2 数据预处理

由于手工标记的ground truth图像不满足训练的要求，训练要求每个像素点的值为其分类标签，因此我们需要使用python进行数据预处理，做法是遍历所有像素点，并且对每一个像素点根据其灰度值的范围进行分类的标注。这里图像背景的像素值的范围为0-10，标注类别为1，肝脏区域为10-200，标注类别为2，肾脏区域为200-255，标注类别为3。这里需要使用到两个核心的Python类库，一个是PIL，用于读取图片，设置图片的模式等，另一个是numpy，这是一个成熟强大的数学运算库，可以实现数组和矩阵的便捷运算，以下是核心Python代码：

from PIL import Image

import numpy as np

import matplotlib.pyplot as plt

import os

'''

*获取某一路径下的所有文件*

'''

def getFiles(path):

ret = []

for root, dirs, files in os.walk(path):

for file in files:

ret.append(os.path.join(root, file))

return ret

'''

*标记groundtruth图像每个像素点的类别*

*并将原始512x512的图像裁剪到 480x360尺寸*

'''

def generateGT(imgpath, savename):

img = np.array(Image.open(imgpath))

rows, cols = img.shape

for i in range(rows):

for j in range(cols):

img[i, j] = getLabel(img[i, j])

p = '/'.join(savename.split('/')[:-1])

if not os.path.exists(p):

os.makedirs(p)

Image.fromarray(img).crop((16, 76, 496, 436)).save(savename)

'''

*标注某一像素点 背景为1 肝脏为2 肾脏为3*

'''

def getLabel(pixel):

# print pixel

if (pixel > 10 and pixel <= 200):

return 2

elif (pixel >= 200 and pixel <= 255):

return 3

else:

return 1

使用Python进行图像的预处理之后，还需要将所有图像的路径写入train.txt文件，供Caffe读取。Caffe通过读取配置在prototxt中的source项，一般是某个文件的路径，如train.txt文件寻找工作数据。详情见3.2.3。其中从数据中取出100张图像用作测试样本，同理需要将图像的保存路径写入test.txt文件中。因此我们需要将预处理好的文件的路径按照“训练图片路径 label图片路径”的格式存入文本，由于样本量大，此工作仍然由Python完成。filelist.py的核心代码如下：

path = '/media/zmbhou/0002AEEC00089D02/fj/proed-data-480-360'

trainTxt = path + '/train.txt'

files = getFiles(path)

f = open(trainTxt, 'a')

print('starting ...')

for file in files:

if(file.find('.GT') > 0):

# *保存路径*

f.write(file.replace('.GT', '').replace('.png', '.jpg') + ' ')

f.write(file + '\n')

f.close()

print('done ...')

### 3.2.3 构建网络模型

Caffe使用prototxt文件构建网络模型和定义训练过程中的优化参数（超参数），prototxt具有清晰易懂的特点。因此训练之前我们需要先编写好训练所需的超参数solver.prototxt和训练网络tran\_net.prototxt。

train\_net.prototxt定义核心代码如下：

name: "VGG\_ILSVRC\_16\_layer"

layer {

name: "data"

type: "DenseImageData"

top: "data"

top: "label"

dense\_image\_data\_param {

source: "/media/zmbhou/0002AEEC00089D02/fj/proed-data-480-360/train.txt" # *之前预处理时的图片路径文本*

batch\_size: 1 # *Change this number to a batch size that will fit on your GPU*

shuffle: true

}

}

layer {

bottom: "data"

top: "conv1\_1"

name: "conv1\_1"

type: "Convolution"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

}

num\_output: 64

pad: 1

kernel\_size: 3

}

}

layer {

bottom: "conv1\_1"

top: "conv1\_1"

name: "conv1\_1\_bn"

type: "BN"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 1

decay\_mult: 0

}

bn\_param {

scale\_filler {

type: "constant"

value: 1

}

shift\_filler {

type: "constant"

value: 0.001

}

}

}

layer {

bottom: "conv1\_1"

top: "conv1\_1"

name: "relu1\_1"

type: "ReLU"

}

layer {

bottom: "conv1\_1"

top: "conv1\_2"

name: "conv1\_2"

type: "Convolution"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 2

decay\_mult: 0

}

convolution\_param {

weight\_filler {

type: "msra"

}

bias\_filler {

type: "constant"

}

num\_output: 64

pad: 1

kernel\_size: 3

}

}

layer {

bottom: "conv1\_2"

top: "conv1\_2"

name: "conv1\_2\_bn"

type: "BN"

param {

lr\_mult: 1

decay\_mult: 1

}

param {

lr\_mult: 1

decay\_mult: 0

}

bn\_param {

scale\_filler {

type: "constant"

value: 1

}

shift\_filler {

type: "constant"

value: 0.001

}

}

}

layer {

bottom: "conv1\_2"

top: "conv1\_2"

name: "relu1\_2"

type: "ReLU"

}

layer {

bottom: "conv1\_2"

top: "pool1"

top: "pool1\_mask"

name: "pool1"

type: "Pooling"

pooling\_param {

pool: MAX

kernel\_size: 2

stride: 2

}

}

layer {

name: "loss"

type: "SoftmaxWithLoss"

bottom: "conv1\_1\_D"

bottom: "label"

top: "loss"

softmax\_param {engine: CAFFE}

loss\_param: {

weight\_by\_label\_freqs: true

ignore\_label: 11

class\_weighting: 1.1595

class\_weighting: 1.1826

class\_weighting: 1.1640

class\_weighting: 1.1817

class\_weighting: 1.1051

class\_weighting: 1.1826

class\_weighting: 0.6446

class\_weighting: 0.8418

class\_weighting: 0.6823

class\_weighting: 0.2478

class\_weighting: 0.3614

}

}

layer {

name: "accuracy"

type: "Accuracy"

bottom: "conv1\_1\_D"

bottom: "label"

top: "accuracy"

top: "per\_class\_accuracy"

}

在图4-1中可以看到，SegNet模型是高度对称的，并且有10个层组组成。上面给出的prototxt部分代码就是第一个层组，此层组由卷积层，BN层，非线性激活层，池化层组成。首先是数据输入层，name为data，数据格式类型为DenseImageData，top指定了层的输出，dense\_image\_data\_param中的source指定了数据的来源，batch\_size为批输入的图像数量，由于个人硬件水平有限，这里只取1，shuffle为true代表输入的图片将会进行洗牌打乱。其次是定义的conv\_1层，此layer的bottom为data，说明data的输入为它的输入，type为Convolution代表此层为卷积层，将会进行卷积操作，param中lr\_mult为1，在solver.prototxt中超参数定义过一个学习率属性为base\_lr，则实际学习率为此处的lr\_multbase\_lr，num\_output定义了输出的通道数，kernel\_size定义了卷积操作中的核大小为3。随后接入了BN层和ReLU层，此时完成了第一组网络层次的构造。以此类推prototxt文件定义个整个SegNet模型。对称之后的最后一层网络为损失层，使用SoftmaxWithLoss来对像素点求最大似然率进行分类。

solver.prototxt的代码如下：

net: "/media/zmbhou/0002AEEC00089D02/fj/SegNet/Models/segnet\_train.prototxt" # *Change this to the absolute path to your model file*

#test\_initialization: false

#test\_iter: 1

#test\_interval: 10000

base\_lr: 0.001

lr\_policy: "step"

gamma: 1.0

stepsize: 10000000

display: 20

momentum: 0.9

max\_iter: 30000

weight\_decay: 0.0005

snapshot: 2000 #1000

snapshot\_prefix: "/media/zmbhou/0002AEEC00089D02/fj/SegNet/Models/snap/" # *Change this to the absolute path to where you wish to output solver snapshots*

solver\_mode: GPU

其中：

net指定了之前定义好的train\_net.prototxt文件的路径，供Caffe读取并构造网络。

base\_lr为基本的学习率，实际学习率为lr\_multbase\_lr，lr\_mult在train\_net.prototxt网络模型的param中定义。

display为每迭代训练多少次进行一次终端输出信息，信息包括正在处于的迭代次数，学习率，各类的损失率等。

max\_iter为最大迭代次数，迭代到这个次数之后将会停止学习，一般来说，迭代到一定次数之后会拟合到一个层度，提高迭代次数继续学习对参数的调整效果不明显。

weight\_decay为权值衰减值，为了防止过拟合。

snapshot为快照，定义了迭代多少次保存一次网络权值的快照，快照可用来中断学习之后恢复训练。

snapshot\_prefix为快照文件的保存路径前缀。训练好的模型也会保存在snapshot里，可以用来进行后续的预测步骤。

solver\_mode为使用的训练模式，有CPU和GPU两个选项。由于此次是对图像进行的预测和分类，GPU对图像的运算有着天然的优势，能极大提高得出结果的时间，故使用GPU模式[10]。

### 3.2.4 训练模型

将Caffe平台搭建完成，将预处理好的数据和编写好的prototxt文件准备完毕后，在Ubuntu终端中键入命令开始训练。

$ cd /home/zmbhou/deeplearning/SegNet/caffe-segnet/build/tools

$ ./caffe.bin train -gpu 0 -solver /media/zmbhou/0002AEEC00089D02/fj/SegNet/Models/segnet\_solver.prototxt

caffe提供了多种使用的接口，此种方式是最简单和基础的，即使用终端通过命令启动caffe训练或者测试。上述命令的含义是，运行caffe.bin，设置使用机器编号为0的GPU，即第一个GPU，同时提供训练的solver超参数的文本路径。除这种使用方式之外，Caffe还提供了python接口和C++接口，让用户可以通过编写程序来使用caffe，但是原理上不变。

神经网络的训练是一个非常消耗时间的过程，训练时间根据数据集大小和迭代次数不同而不同。本次数据集为650张480360的图像，迭代次数为4万次，总耗时6小时17分钟。

### 3.2.4 预测（分割）

模型训练好之后，生成\_iter\_40000.caffemodel快照文件，此文件保存了迭代40000次训练后的模型权值参数，我们用此权值参数计算了BN后，即可对99张图像的测试集进行预测，即可在Ubuntu终端键入命令。

$ cd /home/zmbhou/deeplearning

$ python2.7 SegNet/Scripts/compute\_bn\_statistics.py $MODELPATH/segnet\_train.prototxt $MODELPATH/snap/\_iter\_30000.caffemodel $MODELPATH/inference/ # *compute BN statistics for SegNet*

上述命令会使用之前训练好的\_iter\_40000.caffemodel快照文件来计算出BN，然后我们会使用BN文件来进行最后的图像分割。

$ python2.7 SegNet/Scripts/test\_segmentation\_camvidMY.py --model $MODELPATH/segnet\_inference.prototxt --weights $MODELPATH /inference/test\_weights.caffemodel --iter 100 # *Test SegNet*

其中，test\_segmentation\_camvidMY.py是一个Python脚本，调用了Caffe的Python接口对图像进行分割之后，同时对结果进行了可视化处理，核心代码如下：

*# Import arguments*parser = argparse.ArgumentParser()  
parser.add\_argument(**'--model'**, type=str, required=**True**)  
parser.add\_argument(**'--weights'**, type=str, required=**True**)  
parser.add\_argument(**'--iter'**, type=int, required=**True**)  
args = parser.parse\_args()  
  
caffe.set\_mode\_gpu()  
  
net = caffe.Net(args.model,  
 args.weights,  
 caffe.TEST)  
sta=datetime.now()  
print args.iter  
**for** i **in** range(1, args.iter):  
  
 net.forward()  
  
 image = net.blobs[**'data'**].data  
 label = net.blobs[**'label'**].data  
 input\_shape = net.blobs[**'data'**].data.shape  
 predicted = net.blobs[**'prob'**].data  
 image = np.squeeze(image[0,:,:,:])  
 output = np.squeeze(predicted[0,:,:,:])  
 ind = np.argmax(output, axis=0)  
 ed=datetime.now()  
 print ed-sta  
 r = ind.copy()  
 g = ind.copy()  
 b = ind.copy()  
 r\_gt = label.copy()  
 g\_gt = label.copy()  
 b\_gt = label.copy()  
  
 Sky = [128,128,128]  
 Building = [128,0,0]  
 Pole = [192,192,128]  
 Road\_marking = [255,69,0]  
 Road = [128,64,128]  
 Pavement = [60,40,222]  
 Tree = [128,128,0]  
 SignSymbol = [192,128,128]  
 Fence = [64,64,128]  
 Car = [64,0,128]  
 Pedestrian = [64,64,0]  
 Bicyclist = [0,128,192]  
 Unlabelled = [0,0,0]  
  
 label\_colours = np.array([Sky, Building, Pole, Road, Pavement, Tree, SignSymbol, Fence, Car, Pedestrian, Bicyclist, Unlabelled])  
 **for** l **in** range(0,11):  
 r[ind==l] = label\_colours[l,0]  
 g[ind==l] = label\_colours[l,1]  
 b[ind==l] = label\_colours[l,2]  
 r\_gt[label==l] = label\_colours[l,0]  
 g\_gt[label==l] = label\_colours[l,1]  
 b\_gt[label==l] = label\_colours[l,2]  
  
 rgb = np.zeros((ind.shape[0], ind.shape[1], 3))  
 rgb[:,:,0] = r/255.0  
 rgb[:,:,1] = g/255.0  
 rgb[:,:,2] = b/255.0  
 RGB1 = np.zeros((ind.shape[0], ind.shape[1], 3))  
 RGB1[:,:,0] = r  
 RGB1[:,:,1] = g  
 RGB1[:,:,2] = b  
 strTN=**'SegNet/output/TEST/SINGLE'**+str(i)+**'.png'** cv2.imwrite(strTN,RGB1)  
  
 image = image/255.0  
  
 image = np.transpose(image, (1,2,0))  
 output = np.transpose(output, (1,2,0))  
 image = image[:,:,(2,1,0)]  
 strN=**'SegNet/output/TEST/res'**+str(i)+**'.png'** p1=plt.subplot(211)  
 p1.imshow(image,vmin=0, vmax=1)  
 p2=plt.subplot(212)  
 p2.imshow(rgb,vmin=0, vmax=1)  
 plt.savefig(strN)  
 sta=datetime.now()  
  
print **'Success!'**

在上面的代码中，我们通过参数构造了Caffe.net，并且把数据读取后放入网络进行预测分割，得到的结果图像经过像素RGB的转换，根据不同的分类结果，处理成不同的颜色以进行直观的可视化，最后将成功图保存在本地。本文给出4个SegNet模型分割图像结果较好的示意图，如图3-5所示，上图为原图，原图下为模型分割的结果。

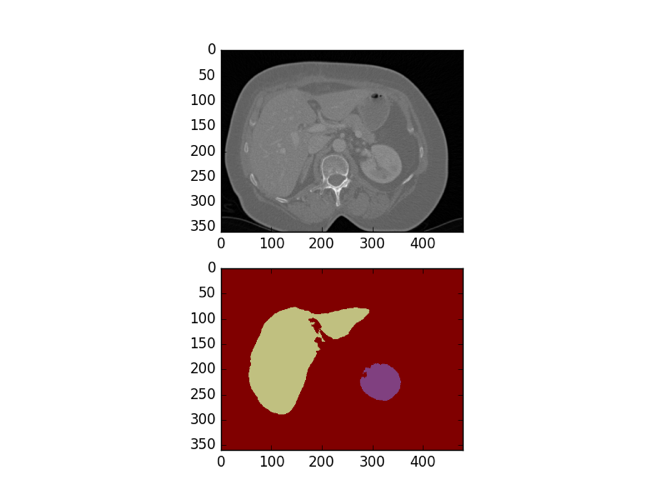
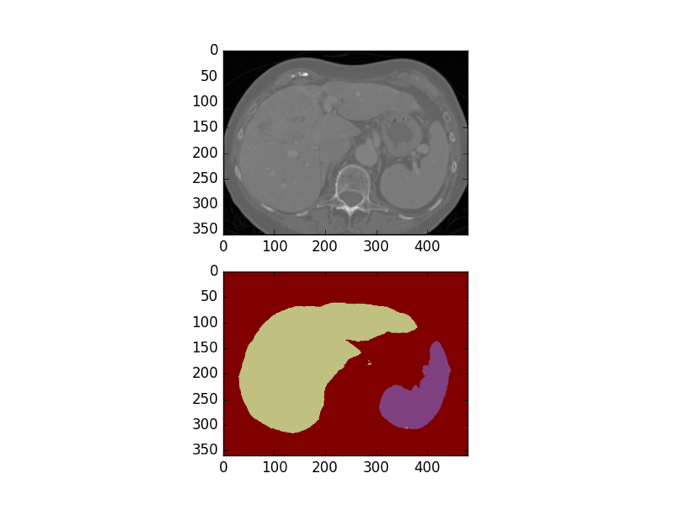
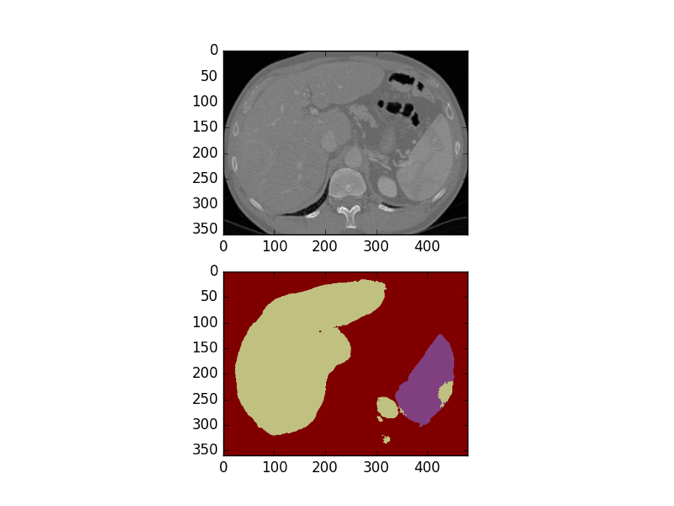
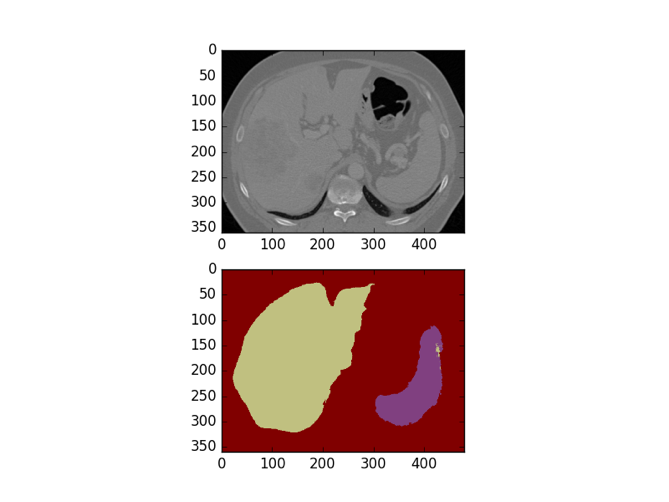
 

图3-5 SegNet分割结果

### 3.2.5 评估

可以看到，虽然精度仍然有些不足，但是已经能够很好地将肝脏和肾脏两个部分分割出来。为了能够更好地说明模型的分割效果，我们使用数理统计中的精确率(Precision)和召回率(Recall)来量化评估结果，下面将使用python脚本来对实验结果进行批量精确率和召回率的计算。

精确率(Precision)指的是被模型判为真的所有样本中有多少是真正的真样本，在这里即为被分类为肝脏或者肾脏的像素点，有多少是真正的分类正确像素点。假设模型的正样本集合为A，真正的正样本集合为B，计算公式如下：

求全率(Recall)指的是所有的真样本中有多少被模型判为真样本，在这里即为ground truth图中的肝脏和肾脏的像素点，有多少被结果图分辨出来，计算公式如下：

F-Measure是IR（信息检索）领域常用的评价标准，我们使用f1-meansure来综合评定模型的好坏，假设P为精确率，R是求全率，计算公式如下：

Python脚本核心代码为：

fr = open(test, 'r')  
fw = open(res, 'w')  
index = 1  
listLiver = []  
listKidney = []  
for line in fr.readlines():  
 RESImg = Image.open(RESImagePath + str(index) + '.png').convert('L')  
 RESCount = getRESCount(RESImg, 'liver')  
 print(str(index) + '.png count is ' + str(RESCount))  
  
 GTImg = Image.open(line.split(' ')[1].replace('\n', ''))  
 trueCount = calcuTrueCount(RESImg, GTImg, 2)  
 print(str(index) + '.GT.png count is ' + str(trueCount))  
 rate = 0.0  
 if(RESCount == 0 and trueCount == 0):  
 rate = 1.0  
 elif(RESCount == 0 and trueCount != 0):  
 rate = 0.0  
 else:  
 rate = trueCount / RESCount  
 listLiver.append(rate)  
 fw.write(str(index) + '.png ')  
 fw.write(str(rate))  
 fw.write('\n')  
 index += 1  
fr.close()  
fw.close()  
print('average is ' + str(sum(listLiver) / len(listLiver)))

最后，得出的平均精确率为85.07%，实验全部精确率数据见附录表1，求全率平均为 82.70%，实验全部召回率数据见附录表2。这里给出表3.1里面包含了精确率，求全率和f1-meansure的结果。

表3.2 FCN评估值

|  |  |  |
| --- | --- | --- |
| 精确率 | 求全率 | f1-meansuer |
| 85.07% | 82.70% | 83.87% |

可以看到，经过初步的训练，模型已经能够达到85%以上的分割精度,求全率也达到了80%以上，说明SegNet网络能够比较好地工作在我们的CT图像上。

下面实验还将简单介绍另外两个模型：FCN 和DeepLab。

### 3.2.5 FCN网络模型

FCN网络模型具有里程碑式的意义，它开拓了图像语义分割领域，在FCN提出之前，神经网络CNN只能对整张图像进行分类和回归，不能够做到针对图像中特定的部分进行识别。FCN实现了端对端，像素对像素的分类，关键就是建立完全卷积的网络，可以采用任意大小的输出，并通过有效的推理和学习产生相应的大小输出。相比较于传统的CNN网络如AlexNet、VGG等，FCN的解决办法是将网络中最后的全连接层变成卷积层，然后加上上采样层（反卷积）将特征还原成图像。典型的识别网络，表面上采取固定大小输入并产生非空间输出，这些网络的全连接层具有固定尺寸并丢弃空间坐标只收取特征值，但这些全连接层也可以被视为卷积层，卷积层具有覆盖其整个输入区域的内核。FCN使用卷积层来替代传统CNN中的全连接层，为了对应CNN网络，FCN把最后三层全连接层都转换成为三层卷积层。在传统的CNN结构中，前5层是卷积层，后面由一个长度为4096的一维向量组成第6层和第7层，第8层也是一个一维向量，长度为1000，代表着模型学习得到的1000个不同类别的概率。而FCN将这3层变换为卷积层，卷积核的大小 (通道数，宽，高) 分别为 (4096,1,1)。

图3-6给出了从CNN到FCN变换的网络模型图。

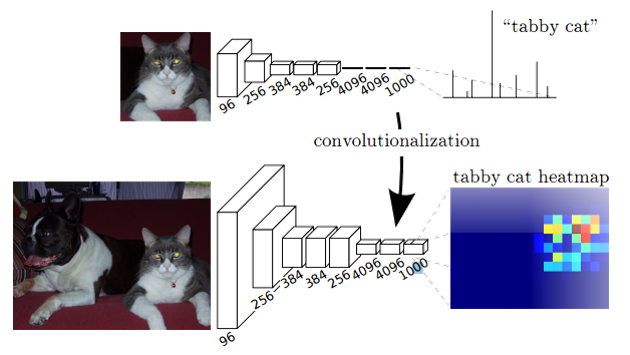


图3-6 CNN到FCN的变换

FCN中为了优化结果，会添加一个跳层连接，因为如果将全卷积之后的结果直接上采样，那么最后得到的结果图将会很不够精细，所以FCN创造性地将处于不同层级的池化层的结果进行连接，然后进行上采样来优化输出。具体结构如图3-7：

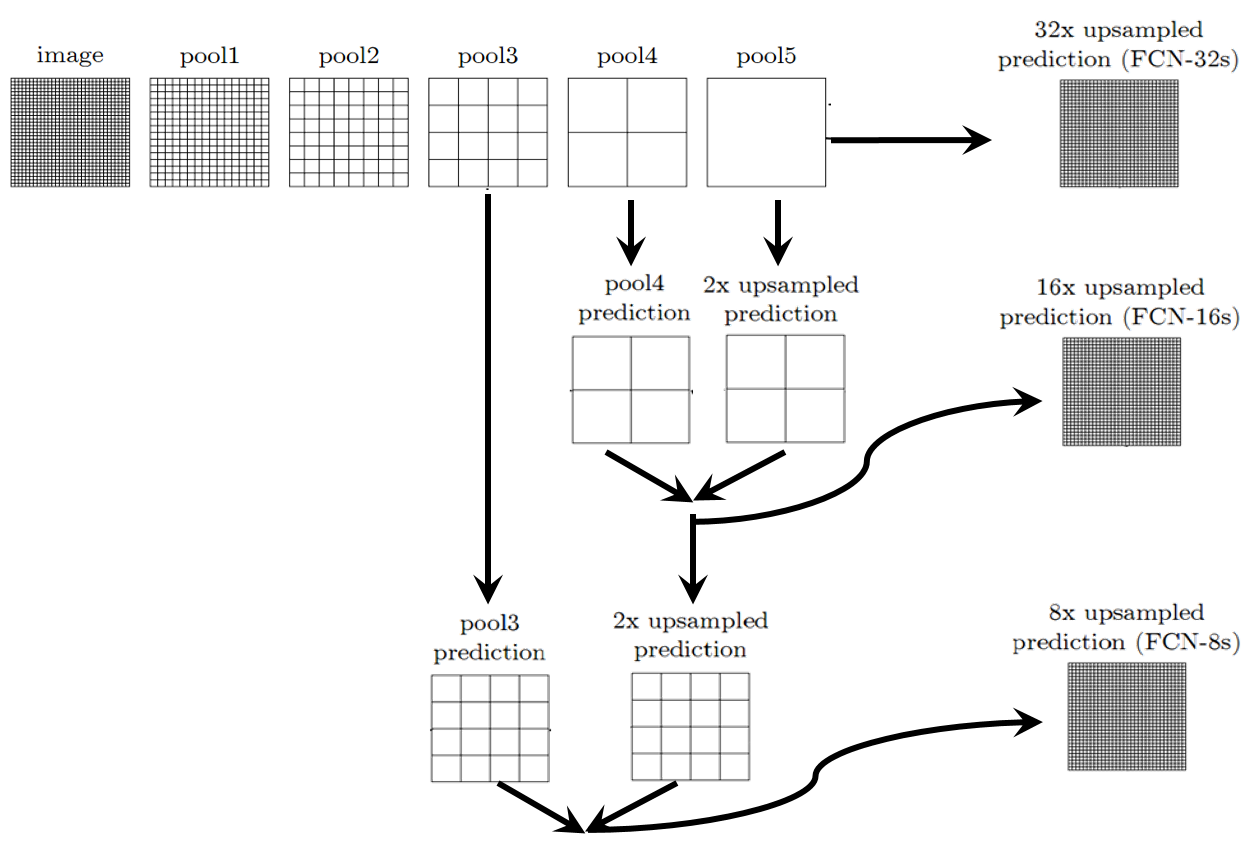
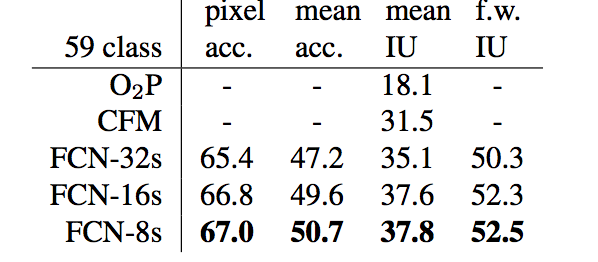


图3-7 跳层连接

可以看到pool3层的结果和pool4、pool5层的采样结果进行跳层连接，然后进行上采样输出。

FCN基于已经训练好的VGGNet结构的精度如下表所示，可以看出像素级分类的精度达到了60%以上，见表3.1[5]。

表3.2 FCN评估值



### 3.2.5 DeepLab网络模型

DeepLab模型由google提出，在VOC上的排名比较靠前，Deeplab网络结构和FCN有点相似，并且也是基于VGG网络上进行微调。但是DeepLab在处理分割图上，使用的方式要比原来FCN的方式具有创造性。

DeepLab网络模型如图3-8所示

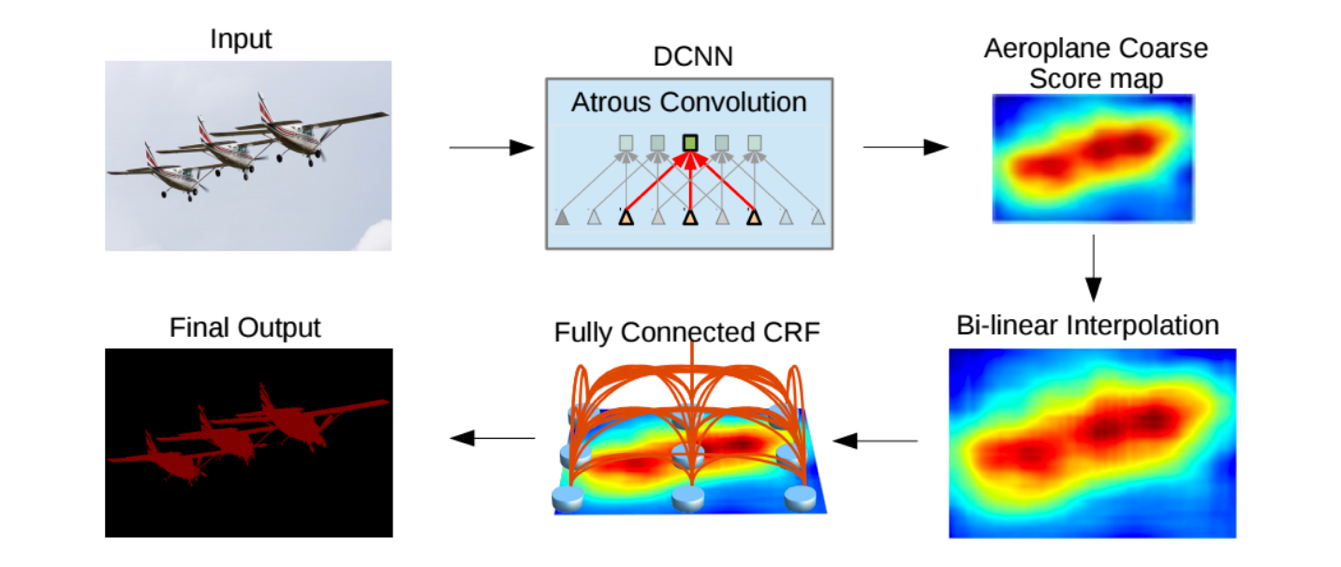


图3-8 DeepLab网络模型

可以看到，DeepLab的结构先使用深度全连接神经网络抽取特征图,去掉深度网络后面的全连接层，用atrous convolution层替代，最终得到了原图1/8大小的特征图，然后用最原始的双线性插值恢复到原图大小，最后使用全连接CRF平滑边界，获得图像语义分割的结果。

其中Atrous Convolution是DeepLab模型中提出了一种新的卷积，带孔的卷积。由于重头开始训练过于缓慢，因此许多模型都希望在现有训练好的模型上进行结构的修改，而这样会产生一个对立的问题，别人训练好的模型并不能很好地适配。DeepLab试图通过Atrous Convolution来解决这个问题，做到既能使用别人已经训练好的模型进行微调，又能够得到更加适配的分割图。

Atrous Convolution如图3-8所示，在传统的卷积或者池化过程中，3-9上面图中所示，每个过滤器在特征图里权重在空间上是连续的。而Atrous Convolution会将某一层的步长由1变成2，为了保证过滤器的感受野大小是相同的，后面的层需要采用hole算法，具体实现方式就是使用跳层连接，将连续的连接关系根据hole size大小进行转换[11]。

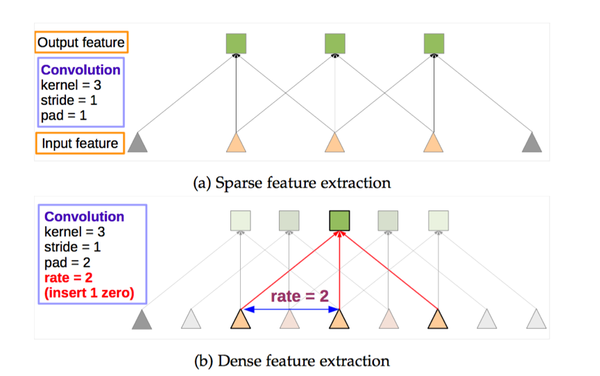


图3-9 Atrous Convolution

所以，通过Atrous Convolution之后能够实现池化后的感受野大小相同，并且保证结果在输出的时候不会丢失精细度，而且还具有可以微调的优点，如图3-10所示意。

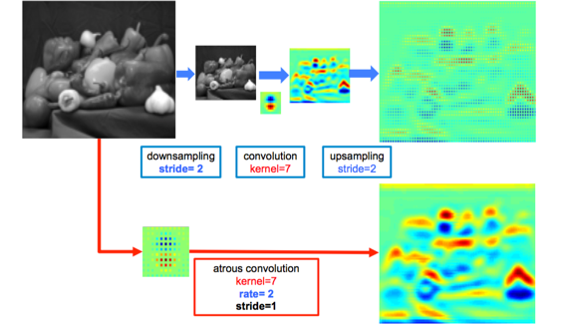


图3-10 Atrous Convolution和传统卷积对比

上图给出了传统卷积和Atrous Convolution卷积的对比，可以看到使用Atrous Convolution之后感受野不变，而且微调之后可以更好地输出特征值。

由于FCN和DeepLab对医学CT图像不敏感，实验结果不理想，故此处不再放出实验结果示意图。

# 第4章 总结

## 4.1 课题完成情况

本次课题通过深度学习技术，特别是通过SegNet网络对医学CT图像进行语义分割，完成了一次将新兴技术引入传统领域的尝试，通过对实验结果的可视化和对结果的精度计算，完成了对深度学习算法在医学图像处理领域的可用性研究，以及对进一步使用深度学习算法来辅助医生诊断，智能化医疗提出可能。

## 4.2 尚存在的问题

对比三个网络模型SegNet，FCN，DeepLab对图像的语义分割结果，可以看出即使是较为成功的SegNet模型，仍然存在着精度不足的问题，如果希望真正应用到临床，仍需要做大量的工作。而FCN和DeepLab在一般图像上工作良好，而到了医学CT这种特殊的图像上时，工作较差，由于这三个模型都是一般普适的模型，没有对特定问题进行深度的定制和优化，所以在精度上和真实场景仍然由改进和提高的空间。

在另一方面，受限于个人设备的硬件水平，即使是训练如此小的样本，仍然花费了大量的时间，因此如果希望将深度学习技术用于生产，那么对设备的硬件水平是一个很高的要求。并且由于深度学习本身网络层次比较深这个特性所致，导致总体来说对比其它方法，深度学习需要大量的资源，包括硬件资源和时间资源。

## 4.3 进一步研究的方向

深度学习技术提出至今不久，正处于快速发展阶段，各种理论和实际应用仍然有大量可以研究的领域，特别是在医疗这种传统的领域，有着方方面面的研究方向。针对于语义分割来说，如何进一步提高精度，识别病灶是这个问题的核心所在。由于医疗领域的特殊性，对所用仪器和软件要求无比严格，因此对精度和准确度的要求会非常高。而目前深度学习领域中所提出的模型，并不具备针对医疗图像进行优化提取特征值，因此研究和构造一个专门适用于医疗图像的网络模型是进一步的研究方向。

另外，由于深度学习技术具有参数不确定的特点，并没有一套完整的理论提供对参数调整的研究，因此需要专业领域人士的配合，如何将网络模型中的参数调整到较优值或者最优值，是另外一个需要大力研究的方向。

深度学习技术，或者说人工智能正在快速地发展，正慢慢应用于我们生活的方方面面，相信这个技术也会慢慢在医疗领域落地生根，造福于人类，前景无比广阔。

# 参考文献

1. 乔玲玲. 图像分割算法研究及实现[D]. 武汉理工大学, 2009.
2. 魏云超, 赵耀. 基于DCNN的图像语义分割综述[J]. 北京交通大学学报, 2016, 40(4):82-91.
3. 万生阳, 王小鹏, 何士和,等. 基于形态学多尺度修正的脑肿瘤分割[J]. 计算机应用, 2014, 34(2):593-596.
4. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.
5. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, PP(99):1-1.
6. Chen L C, Papandreou G, Kokkinos I, et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.[J]. 2017:1-1.
7. 岳祖洲, 陈继述. 人工神经网络的发展[J]. 应用光学, 1994(2):6-16.
8. 周惠芳, 王迎旭, 陈文明. 一种改进的BP神经网络[J]. 自动化技术与应用, 2013, 32(9):7-9.
9. 凡保磊. 卷积神经网络的并行化研究[D]. 郑州大学, 2013.
10. 赵永科. 深度学习:21天实战Caffe[M]. 电子工业出版社, 2016.
11. 王燕. 特定类的图像语义分割[D]. 上海师范大学, 2012.

# 附录1

附录表1 实验数据各precision的值

|  |  |
| --- | --- |
| **Precision** | |
| 对应项 | 精确率 |
| 1.png | 0.6646176447846593 |
| 2.png | 0.9695475955745426 |
| 3.png | 0.9876560971398975 |
| 4.png | 0.8997579101452539 |
| 5.png | 0.9046286876907427 |
| 6.png | 0.6946219573490342 |
| 7.png | 0.9489087836623162 |
| 8.png | 0.0 |
| 9.png | 0.9091743624580021 |
| 10.png | 0.9999237223493517 |
| 11.png | 0.992889704406663 |
| 12.png | 1.0 |
| 13.png | 1.0 |
| 14.png | 0.9331378299120234 |
| 15.png | 0.9614683512477215 |
| 16.png | 0.999597164034805 |
| 17.png | 0.49629325173920324 |
| 18.png | 0.4274679945438709 |
| 19.png | 0.5075853953310858 |
| 20.png | 0.9773679357133448 |
| 21.png | 0.9188714872820171 |
| 22.png | 0.9676406788311854 |
| 23.png | 0.9290304747425886 |
| 24.png | 0.9357118815213733 |
| 25.png | 0.8968153537909944 |
| 26.png | 0.6200042554700521 |
| 27.png | 0.5685560926258992 |
| 28.png | 0.6099882214369847 |
| 29.png | 0.6594389285000333 |
| 30.png | 0.8624688696427908 |
| 31.png | 0.6497144732523794 |
| 32.png | 0.6750421703682878 |
| 33.png | 0.7314223553345389 |
| 34.png | 0.5139166177334116 |
| 35.png | 0.7288750807468701 |
| 36.png | 1.0 |
| 37.png | 0.9750249086682166 |
| 38.png | 1.0 |
| 39.png | 1.0 |
| 40.png | 1.0 |
| 41.png | 0.9999559626563326 |
| 42.png | 0.9550675812801883 |
| 43.png | 1.0 |
| 44.png | 1.0 |
| 45.png | 0.9994341922005571 |
| 46.png | 0.9240981003996277 |
| 47.png | 0.6328282828282829 |
| 48.png | 0.9105668220886467 |
| 49.png | 0.9536397143515319 |
| 50.png | 0.978444263439609 |
| 51.png | 0.9082804501620046 |
| 52.png | 0.7278620352250489 |
| 53.png | 0.8991895781261835 |
| 54.png | 0.7372709388602321 |
| 55.png | 0.7840652877168905 |
| 56.png | 1.0 |
| 57.png | 0.9545120125834031 |
| 58.png | 0.9500234509354317 |
| 59.png | 0.9466332616455668 |
| 60.png | 0.9974582759148676 |
| 61.png | 1.0 |
| 62.png | 0.9878718923237066 |
| 63.png | 0.9976308320236448 |
| 64.png | 0.5957880628511136 |
| 65.png | 0.9523046379792858 |
| 66.png | 0.9562756511709005 |
| 67.png | 0.8877817260144621 |
| 68.png | 0.9584103759680574 |
| 69.png | 0.9868617944574131 |
| 70.png | 0.9909363500068148 |
| 71.png | 0.6453953030861921 |
| 72.png | 0.8637350403562483 |
| 73.png | 0.75788119236543 |
| 74.png | 0.9820599274763026 |
| 75.png | 0.9749347803401858 |
| 76.png | 0.9698325349380419 |
| 77.png | 0.7954342351125193 |
| 78.png | 0.9553022902420493 |
| 79.png | 1.0 |
| 80.png | 0.9991918423617219 |
| 81.png | 0.5390464348907273 |
| 82.png | 0.6823127800298698 |
| 83.png | 0.9664802704325024 |
| 84.png | 0.5741444270015699 |
| 85.png | 0.9616142873709845 |
| 86.png | 0.9678009513355287 |
| 87.png | 0.9679214710095578 |
| 88.png | 0.9425016734759827 |
| 89.png | 0.9624467454786558 |
| 90.png | 0.986816854401335 |
| 91.png | 0.5219246730874638 |
| 92.png | 0.5348190160250408 |
| 93.png | 0.4274127098568339 |
| 94.png | 0.6372842969463782 |
| 95.png | 0.5824515887032493 |
| 96.png | 0.6323471340481759 |
| 97.png | 0.6625288605188102 |
| 98.png | 0.814241810435147 |
| 99.png | 0.8230038888694251 |
| average | 0.8507194423377362 |

# 附录2

附录表2实验数据各Recall的值

|  |  |
| --- | --- |
| **Recall** | |
| 对应项 | 求全率 |
| 1.png | 0.9730571121299598 |
| 2.png | 0.9888280639034744 |
| 3.png | 0.9758905979017997 |
| 4.png | 0.9861166815783791 |
| 5.png | 0.9919129949804797 |
| 6.png | 0.38145183549544576 |
| 7.png | 0.9867643759814552 |
| 8.png | 0.0 |
| 9.png | 0.9966189019481565 |
| 10.png | 0.5475429693210534 |
| 11.png | 0.6993194806243815 |
| 12.png | 0.3838367479148109 |
| 13.png | 0.2558992413764198 |
| 14.png | 0.9960778536836872 |
| 15.png | 0.9956172705576047 |
| 16.png | 0.7230748309891989 |
| 17.png | 0.9976437467486765 |
| 18.png | 0.9933929117018027 |
| 19.png | 0.9952605157307225 |
| 20.png | 0.991474007750902 |
| 21.png | 0.9995408631772268 |
| 22.png | 0.9925656396429413 |
| 23.png | 0.9998016856717898 |
| 24.png | 0.9976673246007536 |
| 25.png | 0.9987375590851707 |
| 26.png | 0.9890815489491698 |
| 27.png | 0.999703520691785 |
| 28.png | 0.9994853452047039 |
| 29.png | 0.9910669577583272 |
| 30.png | 0.9988377098579423 |
| 31.png | 0.9994749384062361 |
| 32.png | 0.9935860963120053 |
| 33.png | 0.9994208829604463 |
| 34.png | 0.9944198964777174 |
| 35.png | 1.0 |
| 36.png | 0.2822274711903228 |
| 37.png | 0.5627156329065399 |
| 38.png | 0.2500139376707365 |
| 39.png | 0.15477575287579165 |
| 40.png | 0.08002613809627532 |
| 41.png | 0.8188013846819558 |
| 42.png | 0.9752155172413793 |
| 43.png | 0.5783141589593203 |
| 44.png | 0.001022731021741282 |
| 45.png | 0.7182671254300907 |
| 46.png | 0.9961053904936122 |
| 47.png | 0.04063168817692457 |
| 48.png | 0.5761151590545583 |
| 49.png | 0.9445553590782043 |
| 50.png | 0.9693362921121647 |
| 51.png | 0.9996412896691425 |
| 52.png | 0.9997060182268699 |
| 53.png | 0.9990112548648364 |
| 54.png | 0.9999579814277911 |
| 55.png | 0.9996562929633795 |
| 56.png | 0.0 |
| 57.png | 0.99497540593431 |
| 58.png | 0.9901958121724016 |
| 59.png | 0.998059795694149 |
| 60.png | 0.7564855888705669 |
| 61.png | 0.11920361740414341 |
| 62.png | 0.9880153983972109 |
| 63.png | 0.9802249469899511 |
| 64.png | 0.9647219079722024 |
| 65.png | 0.996446001751971 |
| 66.png | 0.9948373053524585 |
| 67.png | 0.989859529548666 |
| 68.png | 0.9885295743504698 |
| 69.png | 0.939994298745724 |
| 70.png | 0.9671111702304546 |
| 71.png | 0.49551590673086 |
| 72.png | 0.9828039395762739 |
| 73.png | 0.07398722914267769 |
| 74.png | 0.3087708770877088 |
| 75.png | 0.9938091692373152 |
| 76.png | 0.9925046592658617 |
| 77.png | 0.9981903912323181 |
| 78.png | 0.9948905769578585 |
| 79.png | 0.08709564705665998 |
| 80.png | 0.7977871403628635 |
| 81.png | 0.9980226072071541 |
| 82.png | 0.9963858424725822 |
| 83.png | 0.9216678717763317 |
| 84.png | 0.9993988249760897 |
| 85.png | 0.8984722071730415 |
| 86.png | 0.9577654881822372 |
| 87.png | 0.9817779048163499 |
| 88.png | 0.9998285952153578 |
| 89.png | 0.9978353712045867 |
| 90.png | 0.9957063478700118 |
| 91.png | 0.9981530935682732 |
| 92.png | 0.9991753092590279 |
| 93.png | 0.9981659099792537 |
| 94.png | 0.9958376063220125 |
| 95.png | 0.9898536481712662 |
| 96.png | 0.9988484514070485 |
| 97.png | 0.9993853970335164 |
| 98.png | 0.9981590889630961 |
| 99.png | 1.0 |
| average | 0.8270476983931168 |

# 致 谢

本论文是在导师周雷老师的悉心指导下完成的，在论文设计期间，本人得到了周雷老师的精心指导。他细心的想法，严谨的态度，苛求的理念，都让我倍感振奋。这样严肃的态度对我未来的学习和生活都将产生深远的影响。鄙人非常庆幸能够遇到这样的导师，导师的恩情将会永远铭刻在我心中！本次论文设计的每一个阶段都凝聚了周老师的心血，在此谨向周雷老师表示真挚的谢意和崇高的敬意。

值此论文完成之际，谨向在这五年来给予我关心、帮助、指导的所有亲人、老师、领导、朋友和同学们致以最衷心的感谢！感谢我所在的上海理工大学，在大学生涯中给予了我如此亲静祥和的生活环境和学习环境！

最后感谢所有关心和帮助过我的人们，谢谢你们！惟愿未来的人生道路能够一帆风顺，以不负你们的期待！