人脑核磁共振影像海马体结构检测与分割
（华为）
项目详细方案

2018/4

目　　录

1 引言

1.1 项目的研究背景

海马体(Hippocampus)，又名海马区、大脑海马，位于双侧大脑内部颞叶内，由于形状与海洋生物海马相似，故称为海马体。海马体是哺乳类动物的中枢神经系统大脑皮质的一个部位，它主管人类的近期记忆，类似于计算机的内存，将近几周或几个月内的记忆暂时存储。海马发达的人记忆能力也会好一些。，然而，海马中的信息如果在一段时间没有被使用，便会自行“删除”，即忘记了。因此，海马体是大脑神经系统的重要组成部分，与人类认知、记忆有着重要的联系[1]。

脑组织可大致分为灰质(Grey Matter，GM)和白质(White Matter，WM)，海马体对称分布于位于大脑双侧颞叶内侧深部的内褶灰质区，主要负责学习、记忆，以及空间定位。如图1所示，沿纵轴方向，海马体从前至后依次可分为头、体、尾三部分，全长约4-5cm，分别占全长的35%、45%、20%[2]，海马体体积很小，平均为2-3毫升。

图1 海马体结构示意图

海马体在侧脑室底部绕脉络膜裂形成弓形隆起，其边缘被侧脑室下角和环池内的脑脊液所包围，下侧与海马旁回相邻。海马头膨大，以杏仁核和侧脑室颞角的侧隐窝和为界；上界是脉络膜裂、内侧界为四叠体池、下界为海马白质、外侧界为侧脑室何海马体灰质；海马尾的上界为穹隆[1]。

1.2 项目的研究意义

阿兹海默症(Alzheimer’s Disease，AD，老年痴呆症)是一种发病进程缓慢、并随时间不断恶化的神经退行性疾病，常见于老年群体，也偶见年轻患者[4]。

阿兹海默症的真正成因至今仍然不明，且其引发的神经退化过程是不可逆的，目前并无有效停止或逆转病程的治疗，仅少数可能暂缓症状的方法。主要早期症状为健忘，部分患者衍生出行为或性格的改变，和“轻微行为能力受损”，对日常作息焦虑、无法控制冲动、多侵略性等。严重者往往因此 脱离家庭和社会关系，并逐渐丧失身体机能，最终导致死亡。随着世界人口加速 步入老龄化，阿兹海默症发病人数近年来急速上升，且患者需要他人长期照看，相应产生的看护成本巨大，因此已逐渐演变为全球范围广泛关注的焦点问题之一。因此，对疾病的早期诊断极为重要。

对于阿兹海默症的计算机诊断通常使用脑部成像[2]，例如核磁共振成像（Magnetic Resonance Imaging，MRI）、结构性磁共振成像(structural Magnetic Resonance Imaging， sMRI)、功能性磁共振成像（functional Magnetic Resonance Imaging，fMRI）等图像数据[5-6]。如图2所示，在磁共振图像中，海马体结构具有以下特性:其主要部分属于脑灰质，且海马的形状不规则、体积较小，左右海马对称分布于双侧颞叶内侧。沿长轴从前至后依次为头、体、尾三部分，且周围包绕着白质、皮质和脑室。如图1-3所示，是双侧海马在一幅三维脑部磁共振图像中位置的相对关系图，其中包括横截面、冠状面和矢状面的断层图，其中用实线标出的为专家分割好的左右海马标签，绿色表示左海马，黄色表示右海马。

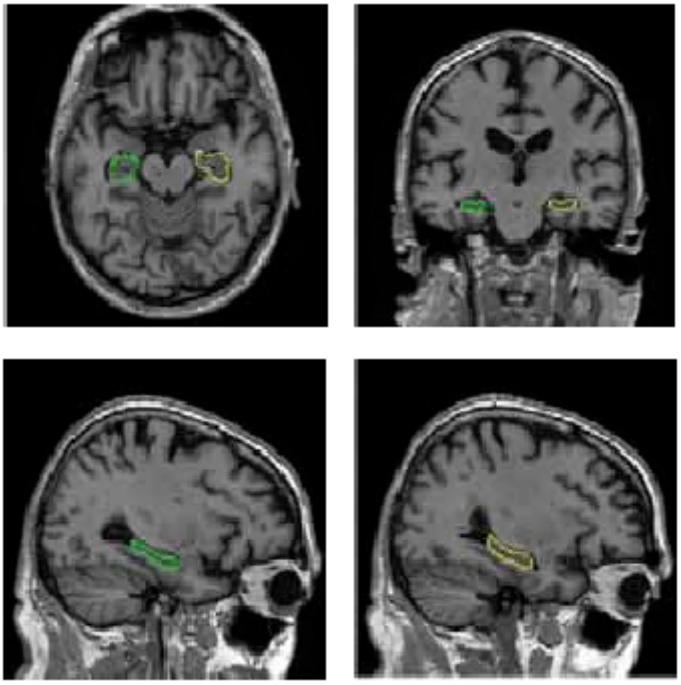


图2 MRI图像中左右海马体位置

研究表明海马体的体积和功能的异常与很多精神疾病有紧密关系。例如：颞叶癫痫、阿尔兹海默综合症、重度抑郁症、精神分裂等[7]。

海马体的衰弱及萎缩引起阿尔兹海默综合症，世界范围内约有两千万阿尔兹海默症患者，绝大多数发生在65岁以上的老年人。随着年龄的增大，患上AD的几率也越来越大，先是影响记忆，进而影响语言、情感表达和行为能力。阿兹海默症的早期临床表现可通过核磁共振技术对患者脑部进行三维造影，继而基于影像分析进行诊断以及相关治疗方案的设计。 在判断海马体是否萎缩时，医生通常需要对海马体结构进行分割，并进行形状和体积分析。然而海马体尺寸小、形状不规则并因人而异，且在常规核磁共振影像下与周边组织结构对比度低，边界不清晰甚至不连续。非具备多年临床经验的影像科医生难以进行精准分割。

海马体在人脑中发挥的重要作用使其成为神经影像学研究的一个特别重要的目标。海马体因其功能参与记忆机制而众所周知，并且还通过其结构直接与记忆相关的疾病如阿尔茨海默氏病相关联。

1.3 国内外研究现状分析

在国外，医疗影像智能分析公司发展较早，目前部分公司已经较为成熟。相比之下，国内的医疗影像分析公司刚刚兴起，十多家较为明确涉足将人工智能用于分析医疗影像，从而提高诊断效率和精确性的公司都是近十年成立的。据统

计，在美国医疗影像数据的年增长率为 63%，而放射科医生数量年增长率仅为 2%；根据动脉网的数据，国内医疗影像数据和放射科医师的增长数据分别为 30%和 4.1%。如果能借助人工智能的方式解读影像，以辅助诊断，可以有效其中的弥补缺口。而国内医护人员短缺的情况，只会比美国更甚，而且影像科医师在医院的收入与地位不高。

1. 海马体的分割

为了深入研究海马体的体积形态变化与疾病的关系，从 MRI 中分割出海马体成为主要研究手段，方法主要有手动分割、半自动分割和智能分割三种。

(1)手动分割方法。

目前，海马体的手动分割结果仍然被认为是海马体积形态分析的金标准。手动分割指有经验的专家在 MRI 的切片上逐层勾画。对于一个具有丰富的解剖知识经验的医生，手动分割一对左右海马体至少需要两到三个小时[8]。该过程枯燥费时，主观性强且不具有可重复性[9]。因此，实现海马体的自动分割，对研究海马体的结构和形态具有现实意义，手动分割需要专家全程参与，是一个不可重复的过程。

(2) 半自动分割方法。

1987 年，Kass 等提出 Snake 模型[10]，该算法首先选择一个初始轮廓，然后利用初始轮廓进行迭代，使其轮廓沿着减小的方向移动，能最终得到一个优化的估计边界。但该方法对初始轮廓较为敏感，通常需要理想的先验轮廓。1990 年，C.R.Jack 为了找到一种精确的可复用的脑组织分割方法，分别将阈值法、边界跟踪法和随机标记法应用于海马体的分割实验中，最终结合阈值法和边界跟踪法进行海马体分割[11]。Starck 等人提出的海马体分割方法是基于特征的边缘提取方法，利用脑部冠状面横断面的信号差，对图像中灰质、白质、脑脊液进行分类，再结合匹配滤波器以及水平集方法获得优化的海马边界实现分割[12]。

这些半自动的分割方法在很大程度上提高了分割效率，节约了时间成本。但是，仍然需要人工设置阈值、选择种子点或设置初始轮廓等先验信息。为了加快病理学和临床实验的研究进程，国内外学者对海马体的自动分割方法进行了深入的研究和探索。

(3) 智能分割方法。

目前，已有权威机构开发了关于脑组织自动分割的软件工具 FSL 和 Freesurfer。 FSL 是由牛津大学 FMRIB 中心研发的用于 FMRI， MRI 和 DTI 脑成像数据分析综合工具库，其中 FIRST 工具包基于贝叶斯形状模型，可用于分割海马体，杏仁核等皮下组织。Freesurfer 由哈佛大学和麻省理工大学医学成像中心研发，用于处理大脑 3D 结构图像数据，利用专家先验概率信息估计的方法进行自动皮层和皮下核团分割的工具。这两个软件包需在 Linux 操作系统中使用，且需熟悉各种指令，对使用者要求较高。

基于图谱配准的分割方法成为很多学者研究的重点方法。Bajcsy 等人首先提出将图谱应用于 CT 图像自动定位与测量中。随后，Haller[13]等人首次利用单图谱的方法实现海马体分割。但是，由于不同个体间解剖结构差异大，单图谱难以适应复杂的图像差异性， Heckmann 提出了基于多图谱配准的自动分割算法。Han[14]等人采用多图谱以及变形距离模型得到 CT 分割结果；Yang 等人基于

Demons 算法进行多图谱配准，配准后用精确与性能评估算法提高结果的准确度[15]。COLLATE 算法此基础上增加了一致性概率的估计，从而提高融合精度[16]。

Bruce Fischl 提出了一种基于概率先验的海马体自动分割方法，利用训练集中的手工标签进行概率估计，获得估计概率信息指导分割[17]。如图 3-6 所示：

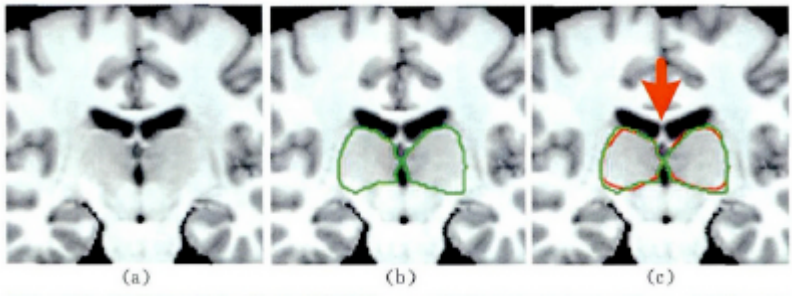


图3数据丘脑的分割结果

(a)待分割图像 (b)专家手动分割的金标准 (c)自动分割图像

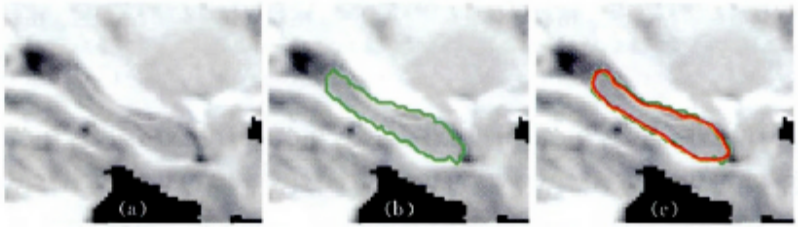


图4 海马体的分割结果

(a)待分割图像 (b)专家手动分割的金标准 (c)自动分割图像

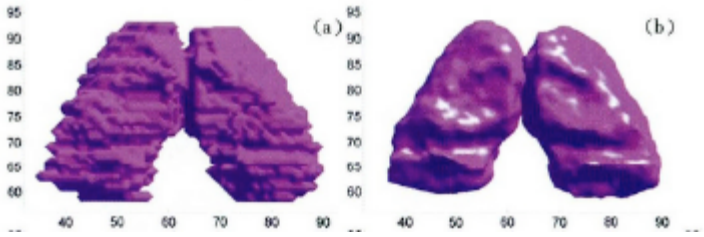


图5 丘脑分割后三维重建结果

(a)专家手动分割的金标准 (b)自动分割

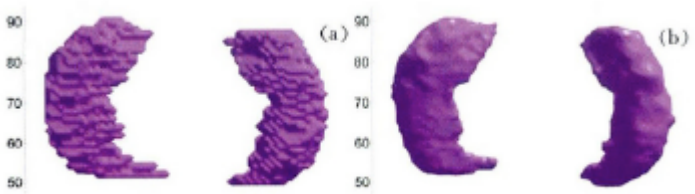


图6 海马体分割后三维重建结果

(a)专家手动分割的金标准 (b)自动分割

2. 海马体的检测

深度神经网络常被应用于机器自动识别，在阿尔兹海默症的自动识别上也具有良好的效果[18-19]。卷积神经网络作为深度神经网络的一种网络结构[20]，文献[18-19]都采用了深度神经网络的方法对阿尔兹海默症进行分类诊断，而且两者对正常人NC和患者AD的分类准确率都达到91.4%。

2 研究方案

2.1 研究目标、研究内容和海马体特点

2.1.1 研究目标

本项目综合考虑人脑核磁共振影像 MRI 海马体结构的分割与检测诊断相结合，研究人脑核磁共振影像 MRI 海马体结构的三维建模成像，采用虚拟现实技术进行展示；重点对经典的海马结构分割算法进行优化和改进，对分割出的海马体进行数据比对，检测出是否患有阿尔兹海默综合症，提高诊断率。

具体研究目的：

(1)研究人脑核磁共振影像 MRI 海马体结构的图像构成，对 MRI 图像进行预处理并进行三维建模成像，将二维 MRI 采用虚拟现实技术进行呈现；

(2)综合考虑现有海马结构分割算法，找出现有算法的缺陷及应用不足，并进行改进、优化与升华，研究出消耗量小、快速、准确的分割算法；

(3)研究已分割出的海马体，进行精确的数据比对，改进算法对图像进行训练和识别。将同一个对象的图像识别结果用于对该对象的联合诊断，检测出是否患有阿尔兹海默综合症，提高诊断率。

2.1.2 研究内容

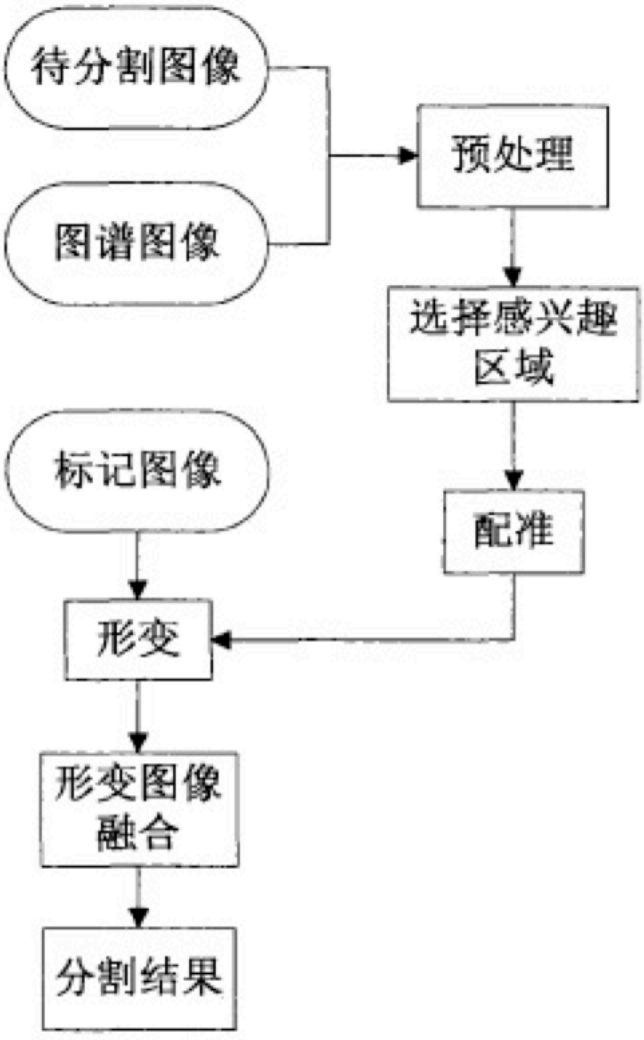
本项目研究的核心内容是采用深度学习方法对海马结构分割算法进行改进，提高分割效率和海马体检测结果的诊断率。

(1)研究图像预处理过程，对图像进行三维成像；

(2)研究分割图像的配准问题，采用卷积神经网络进行海马体算法的实现；

(3)采用深度学习的方法对 3D 脑图像进行提取、识别，获得联合的诊断的结果。

首先对 18 个脑部磁共振图像进行预处理,并根据海马体的解剖位置选择感兴趣区域后;再对待分割的目标图像与图谱进行配准,得到相应变形场的几何变换参数;然后,利用这些变换参数对图谱对应的标记图像进行形变,形变后的图像采用图像融合技术,便可得到综合后的海马体分割结果。



实验流程图

2.1.3 脑部 MRI 海马体特点

(1)海马体模型的建模，如何利用脑组织形状分析工具将二值海马图像转化为由顶点和三角面构成的表面网格，为后续的分割做基础，并将 3D 建模转化为虚拟现实效果呈现；

(2)如何将深度学习应该在分割算法和诊断算法，在有的算法上进行改进；

(3)如何使用将 Matlab 的使用转换为 Python 语言实现。

磁共振成像技术以磁共振现象为基础，将接收到的磁共振信号经计算机处理重建出不同层面的图像。相对于其他成像方式，例如：CT(Computed Tomography)，超声成像(Ultrasound)；

磁共振成像因其无任何电离辐射伤害的优点而被广泛应用于人体脑部疾病检查。

不同成像方式对应的脑部图像

(a)超声图像；(b)CT 图像；(c)磁共振图像

MRI 设备相对于其他影像设备具有明显优势。例如：

1、成像分辨率高，尤其是适用于软组织、神经系统、脑功能的成像；

2、由于 MRI 成像原理与其他设备不同，因此不会产生电离辐射；

3、可以实现任意断层的成像，也可获得人体横截面、冠状面、矢状面及任何方向层面的图像；

4、多参数成像，可调节适用于不同成像对象的参数；

5、对比度成像，可获取多种不同加权特性的图像，如 T1 加权、T2 加权、质子密度加权等，提高诊断质量。

针对海马体的特点以及磁共振图像的成像优势，目前的研究海马体的分割主要依赖于磁共振图像。通过磁共振图像对海马体进行体积测量和形态分析，可对阿尔兹海默症等疾病进行诊断。在磁共振图像中，海马体结构具有以下特性：其主要部分属于脑灰质，且海马的形状不规则、体积较小，左右海马对称分布于双侧颞叶内侧。沿长轴从前至后依次为头、体、尾三部分，且周围包绕着白质、皮质和脑室。如图所示，是双侧海马在一幅三维脑部磁共振图像中位置的相对关系图，其中包括横截面、冠状面和矢状面的断层图，其中用实线标出的为专家分割好的左右海马标签，绿色表示左海马，黄色表示右海马。

2.2 基于 MRI 的海马体分割算法分析

2.2.1 拟采取的研究方法

深度学习算法源于机器学习中人工神经网络的研究，通过组合低层特征形成抽象的高层表示，从而发现数据的分布式特征。深度学习常用的模型或算法包括 DBN, AE （AutoEncoder，自动编码器）、SC（Sparse Coding，稀疏编码)CNN（Convolutional Neural Networks，卷积神经网络)等。

（1）BM(波尔兹曼机，Boltzmann Machine)

这种网络中的神经元是随机的神经元，神经元的输出包含两种状态(未激活和激活)，用二进制的 0 和 1 表示，状态的取值由概率统计法则决定。从功能上将，BM 是由随机神经元全连接组成的反馈神经网络，且对称连接，无自反馈，包含一个可见层和一个隐含层的 BM 模型如图 7 所示。

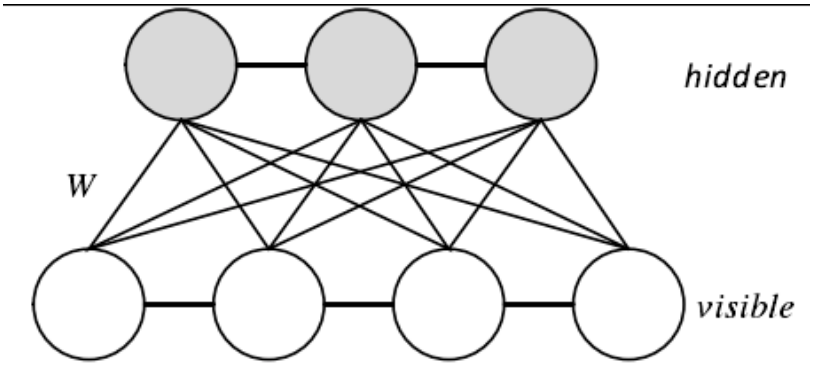


图 7 BM 模型

（2）卷积神经网络模型

卷积神经网络中的特征提取、特征映射以及子抽样的具体过程如下:

- 特征提取，每一层的局部接收域为下一层的神经元提供输入，使其可以对局部特征进行提取。
- 特征映射，多个特征映射组成了网络的每一个计算层，其中特征映射是以二维平面的形式存在的，在约束下，平面中的神经元有着相同的权值集。
- 子抽样，该层在卷积层之后，其功能是实现局部平均与子抽样，从而对于平移等变换，特征映射的输出的敏感度降低。

如图 8 所示，卷积神经网络对网络的结构进行了限制是利用接收域的局部连接实现的。权值共享是卷积神经网络的另一个特点，同一隐含层的神经元共享一个权值集，可以极大地减少自由参数的数量。卷积神经网络能够实现一种输入到输出的映射关系，并能够学习到大量的这种映射关系，且无需输入和输出之间的精确的数学公式，卷积神经网络输入和输出之间的映射能力主要通过已知模式对其本身进行训练得出。卷积神经网络的所有权值会在训练前用不同的小随机数进行初始化，并执行有监督训练。

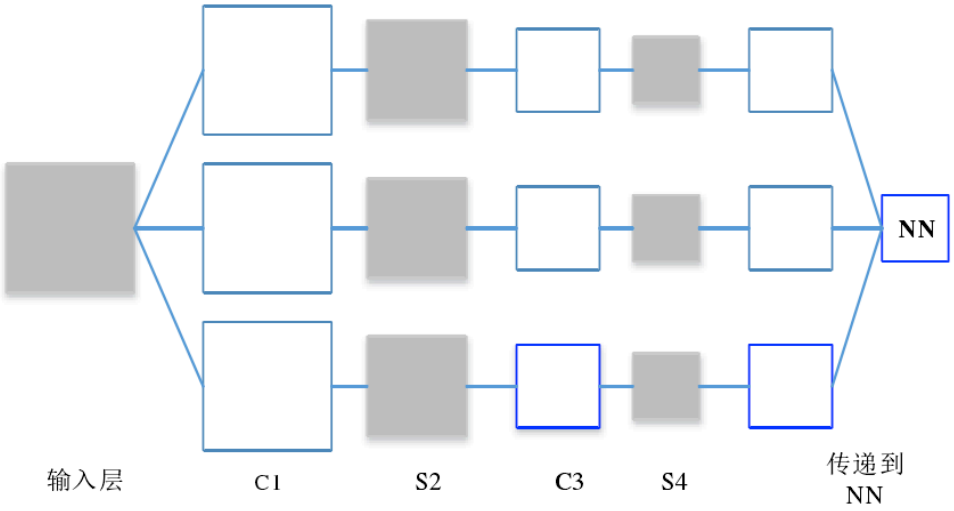
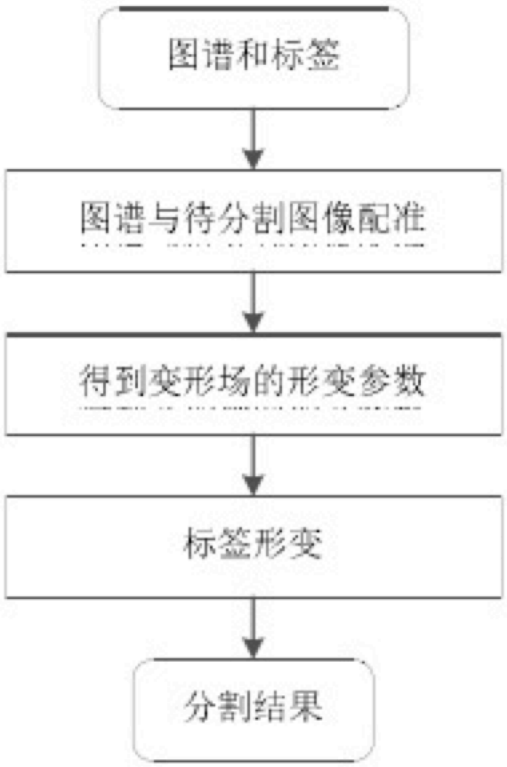


图 8 卷积神经网络

2.2.2 基于图谱配准的海马体分割方法

基于图谱配准的分割方法的实质是将对目标的分割转化为图像之间的配准问题，通过配准过程计算图谱与待分割图像之间的形变参数，然后将形变参数应用到分割对象的标签上，将形变后的标签作为预分割结果。根据图谱数量的不同，可将其分为：基于单图谱的海马体分割方法和基于多图谱的海马体分割方法。



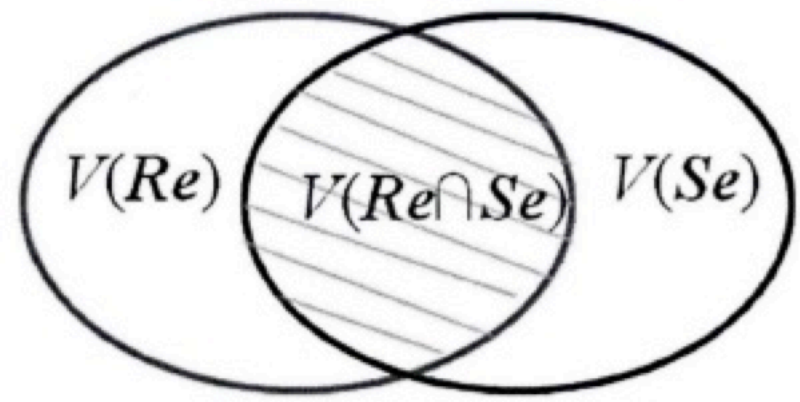
基于图谱配准的分割方法流程

2.2.3 Dice 相似性测度

Dice 相似性测度是对分割结果与人工分割的参考图像的重叠率进行评价的,计算公式如下所示:

$$Dice(Se,Re)=\frac{2V(Se\cap Re)}{V(Se)+V(Re)}$$

其中 Se 表示分割结果,表示人工分割的参考图像,r 表示分割区域的体积大小。相似性测度 Dice 值越大,说明分割结果与人工分割的参考图像的重叠率越高,结果越精确。图示如下:



相似性测度Dice值的计算方法

3 技术路线与实现方案

我们训练了海马体外观的卷积神经网络模型。首先，我们描述模型架构及其训练过程。

3.1 分割模型

我们模型的一般架构是预处理和后处理调用外部工具。ConvNet 在中间部分使用标记为：特征图数量（高度），数据大小（宽度），图层类型的标准框描述来描述。

```
from tensorlayer.layers import *
def u_net(x, is_train=False, reuse=False, n_out=1):
    _, nx, ny, nz = x.get_shape().as_list()
    with tf.variable_scope("u_net", reuse=reuse):
        tl.layers.set_name_reuse(reuse)
        inputs = InputLayer(x, name='inputs')
        conv1 = Conv2d(inputs, 64, (3, 3), act=tf.nn.relu, name='conv1_1')
        conv1 = Conv2d(conv1, 64, (3, 3), act=tf.nn.relu, name='conv1_2')
        pool1 = MaxPool2d(conv1, (2, 2), name='pool1')
        conv2 = Conv2d(pool1, 128, (3, 3), act=tf.nn.relu, name='conv2_1')
        conv2 = Conv2d(conv2, 128, (3, 3), act=tf.nn.relu, name='conv2_2')
        pool2 = MaxPool2d(conv2, (2, 2), name='pool2')
        conv3 = Conv2d(pool2, 256, (3, 3), act=tf.nn.relu, name='conv3_1')
        conv3 = Conv2d(conv3, 256, (3, 3), act=tf.nn.relu, name='conv3_2')
        pool3 = MaxPool2d(conv3, (2, 2), name='pool3')
        conv4 = Conv2d(pool3, 512, (3, 3), act=tf.nn.relu, name='conv4_1')
        conv4 = Conv2d(conv4, 512, (3, 3), act=tf.nn.relu, name='conv4_2')
        pool4 = MaxPool2d(conv4, (2, 2), name='pool4')
        conv5 = Conv2d(pool4, 1024, (3, 3), act=tf.nn.relu, name='conv5_1')
        conv5 = Conv2d(conv5, 1024, (3, 3), act=tf.nn.relu, name='conv5_2')

        up4 = DeConv2d(conv5, 512, (3, 3), (nx/8, ny/8), (2, 2), name='deconv4'
                                # out_size=(nx/8, ny/8), strides=(2, 2)
        up4 = ConcatLayer([up4, conv4], 3, name='concat4')
        conv4 = Conv2d(up4, 512, (3, 3), act=tf.nn.relu, name='uconv4_1')
        conv4 = Conv2d(conv4, 512, (3, 3), act=tf.nn.relu, name='uconv4_2')
        up3 = DeConv2d(conv4, 256, (3, 3), (nx/4, ny/4), (2, 2), name='deconv3'
        up3 = ConcatLayer([up3, conv3], 3, name='concat3')
        conv3 = Conv2d(up3, 256, (3, 3), act=tf.nn.relu, name='uconv3_1')
        conv3 = Conv2d(conv3, 256, (3, 3), act=tf.nn.relu, name='uconv3_2')
        up2 = DeConv2d(conv3, 128, (3, 3), (nx/2, ny/2), (2, 2), name='deconv2'
        up2 = ConcatLayer([up2, conv2], 3, name='concat2')
        conv2 = Conv2d(up2, 128, (3, 3), act=tf.nn.relu,  name='uconv2_1')
        conv2 = Conv2d(conv2, 128, (3, 3), act=tf.nn.relu, name='uconv2_2')
        up1 = DeConv2d(conv2, 64, (3, 3), (nx/1, ny/1), (2, 2), name='deconv1')
        up1 = ConcatLayer([up1, conv1] , 3, name='concat1')
        conv1 = Conv2d(up1, 64, (3, 3), act=tf.nn.relu, name='uconv1_1')
        conv1 = Conv2d(conv1, 64, (3, 3), act=tf.nn.relu, name='uconv1_2')
        conv1 = Conv2d(conv1, n_out, (1, 1), act=tf.nn.sigmoid, name='uconv1')
    return conv1
```

```
from tensorlayer.layers import *
def u_net(x, is_train=False, reuse=False, n_out=1):
    _, nx, ny, nz = x.get_shape().as_list()
    with tf.variable_scope("u_net", reuse=reuse):
        tl.layers.set_name_reuse(reuse)
        inputs = InputLayer(x, name='inputs')
        conv1 = Conv2d(inputs, 64, (3, 3), act=tf.nn.relu, name='conv1_1')
        conv1 = Conv2d(conv1, 64, (3, 3), act=tf.nn.relu, name='conv1_2')
        pool1 = MaxPool2d(conv1, (2, 2), name='pool1')
        conv2 = Conv2d(pool1, 128, (3, 3), act=tf.nn.relu, name='conv2_1')
        conv2 = Conv2d(conv2, 128, (3, 3), act=tf.nn.relu, name='conv2_2')
        pool2 = MaxPool2d(conv2, (2, 2), name='pool2')
        conv3 = Conv2d(pool2, 256, (3, 3), act=tf.nn.relu, name='conv3_1')
        conv3 = Conv2d(conv3, 256, (3, 3), act=tf.nn.relu, name='conv3_2')
        pool3 = MaxPool2d(conv3, (2, 2), name='pool3')
        conv4 = Conv2d(pool3, 512, (3, 3), act=tf.nn.relu, name='conv4_1')
        conv4 = Conv2d(conv4, 512, (3, 3), act=tf.nn.relu, name='conv4_2')
        pool4 = MaxPool2d(conv4, (2, 2), name='pool4')
        conv5 = Conv2d(pool4, 1024, (3, 3), act=tf.nn.relu, name='conv5_1')
        conv5 = Conv2d(conv5, 1024, (3, 3), act=tf.nn.relu, name='conv5_2')

        up4 = DeConv2d(conv5, 512, (3, 3), (nx/8, ny/8), (2, 2), name='deconv4'
                               # out_size=(nx/8, ny/8), strides=(2, 2)
        up4 = ConcatLayer([up4, conv4], 3, name='concat4')
        conv4 = Conv2d(up4, 512, (3, 3), act=tf.nn.relu, name='uconv4_1')
        conv4 = Conv2d(conv4, 512, (3, 3), act=tf.nn.relu, name='uconv4_2')
        up3 = DeConv2d(conv4, 256, (3, 3), (nx/4, ny/4), (2, 2), name='deconv3'
        up3 = ConcatLayer([up3, conv3], 3, name='concat3')
        conv3 = Conv2d(up3, 256, (3, 3), act=tf.nn.relu, name='uconv3_1')
        conv3 = Conv2d(conv3, 256, (3, 3), act=tf.nn.relu, name='uconv3_2')
        up2 = DeConv2d(conv3, 128, (3, 3), (nx/2, ny/2), (2, 2), name='deconv2'
        up2 = ConcatLayer([up2, conv2], 3, name='concat2')
        conv2 = Conv2d(up2, 128, (3, 3), act=tf.nn.relu,  name='uconv2_1')
        conv2 = Conv2d(conv2, 128, (3, 3), act=tf.nn.relu, name='uconv2_2')
        up1 = DeConv2d(conv2, 64, (3, 3), (nx/1, ny/1), (2, 2), name='deconv1')
        up1 = ConcatLayer([up1, conv1] , 3, name='concat1')
        conv1 = Conv2d(up1, 64, (3, 3), act=tf.nn.relu, name='uconv1_1')
        conv1 = Conv2d(conv1, 64, (3, 3), act=tf.nn.relu, name='uconv1_2')
        conv1 = Conv2d(conv1, n_out, (1, 1), act=tf.nn.sigmoid, name='uconv1')
    return conv1
```

分割模型建立

3.1.1 预处理

首先，通过沿着每个维度的三个连续的一维卷积滤波器来对输入进行滤波。这往往会突出显示包含感兴趣特征的边缘，并且由于已经表明，1D 滤波器的层数可以以较低的存储成本近似于二维等值，因此将该想法应用于三个维度。一个完整的 3D 卷积层随后使用 48 个特征图，这被选为表示容量和内存消耗之间可接受的折衷。下一部分首先使用减少空间尺寸的 maxpool 层（从而对更多空间不变特征敏感），接着是两层 3D 卷积滤波器。我们使用残差连接，已经证明它具有适当的拓

扑结构，可以提高训练收敛速度。我们使用的所有卷积核的大小为 3。再次重复 maxpool-ResNet 模块，这将生成 48 个特征映射，其空间维度比原始数据小四倍。

在这一点上，相关信息被编码，使得具有 3D 卷积的两个放大层可以学习生成海马概率图。由于分割是通过升级来自低空间分辨率的数据创建的，因此我们添加了一个细化网络，该网络经过训练可重用初始图层中的特征以改进描绘。卷积滤波器在空间上不变，但我们希望只关注海马周围的区域；因此，期望特征的初始映射在直接海马区域之外被设置为零（其通过平滑然后阈值化当前概率掩模来定义，概念上相当于执行分割掩模的体素扩张）。最终的结果是概率分割，当我们的验证实验需要二进制掩码时，可以选择进一步将阈值设置为 50%。

```
data_temp_list = []#定义了临时的链表
for it, im in tqdm(enumerate(metadata[totaldata].Path.values),#枚举metadata里的进度条
                   total=totaldata.sum(), desc='Reading MRI to memory'):

    img = nib.load(im).get_data()#nib.load(im)载入并获取数据，img是python中的API，也是类
    data_temp_list.append(img)


data_temp_list = np.asarray(data_temp_list)
m = np.mean(data_temp_list)
s = np.std(data_temp_list)

print(m)
print(s)
print(data_temp_list.shape)
```

预处理核心代码

```
Reading MRI to memory: 100%|██████████| 2/2
[00:00<00:00, 29.30it/s]
0.0
0.0
(2, 192, 192, 160, 1)
Validation
Reading MRI to memory:   0%|          | 0/2
[00:00<?, ?it/s]C:/Users/lenovo/Desktop/form
data.py:54: RuntimeWarning: invalid value
encountered in true_divide
  img = (img - m) / s
Reading MRI to memory: 100%|██████████| 2/2
[00:00<00:00,  4.79it/s]
Reading LABEL to memory: 100%|██████████| 2/2
[00:00<00:00,  6.19it/s]
(320, 192, 192, 1)
(320, 192, 192, 1)
Train
Reading MRI to memory:   0%|          | 0/2
[00:00<?, ?it/s]C:/Users/lenovo/Desktop/form
data.py:98: RuntimeWarning: invalid value
encountered in true_divide
  img = (img - m) / s
Reading MRI to memory: 100%|██████████| 2/2
[00:00<00:00,  5.76it/s]
Reading LABEL to memory: 100%|██████████| 2/2
[00:00<00:00,  6.12it/s]
(320, 192, 192, 1)
(320, 192, 192, 1)
```

预处理运行结果

```
save_dir='C:/Users/lenovo/Desktop/CC/'
with open(save_dir + 'train_input.pickle', 'wb') as f:
     pickle.dump(X_train_input, f, protocol=4)
with open(save_dir + 'train_target.pickle', 'wb') as f:
     pickle.dump(X_train_target, f, protocol=4)
#with open(save_dir + 'train_input.pickle', 'rb') as f:
     # pickle.dump(X_train_input, f, protocol=4)
with open('C:/Users/lenovo/Desktop/CC/train_input.pickle', 'rb') as f:
    model = pickle.load(f)
```

train_input.pickle	2018/4/17 19:34	PICKLE 文件	46,081 KB
train_target.pickle	2018/4/17 19:34	PICKLE 文件	46,081 KB

导出二进制格式的文件

3.1.2 损失函数

使用了损失函数：

$$L = (1-\alpha) \times MSE[t - -y_1] + \alpha \times MSE[t - -y_2]$$

其中 t 是目标标签，y1 和 y2 是两个网络输出，MSE 是体素方均方算子，并且 α 是常数标量。也就是说，损失函数使用细化和非细化分割输出错误的加权和。阿尔法的价值在整个训练过程中演变；在预热阶段，α 被设置为 0，然后在大多数训练中 α 被设置为 0.75，然后在最后阶段 α 被设置为 1。

3.1.3 训练模型

作为一个监督学习系统，训练数据由一对图像（一个海马区域图像，一个相应的蒙版图像）组成，并且该模型通过分析数千个提供为训练的这样的对来学习。我们的训练集包含两种类型的输入：首先，使用 FreeSurfer 分割的四个队列中的对象数据作为学习目标。另外，数据集大小通过下面描述的增强过程增加三倍。增强是指对真实数据（及其目标）应用特定变换以人为创建更多的训练数据，有助于机器学习。

```
###======================== LOAD DATA ==================================###
## by importing this, you can load a training set and a validation set.
# you will get X_train_input, X_train_target, X_dev_input and X_dev_target
import dataa as dataset
X_train = dataset.X_train_input
y_train = dataset.X_train_target[:,:,:,np.newaxis]
X_test = dataset.X_dev_input
y_test = dataset.X_dev_target[:,:,:,np.newaxis]

if task == 'all':
    y_train = (y_train > 0).astype(int)
    y_test = (y_test > 0).astype(int)
else:
    exit("Unknow task %s" % task)
```

导入数据

```
    ###======================== DEFIINE MODEL =======================###


        ## nz is 4 as we input all Flair, T1, T1c and T2.
t_image = tf.placeholder('float32', [batch_size, nw, nh, 1], name='input_image')
        ## labels are either 0 or 1
t_seg = tf.placeholder('float32', [batch_size, nw, nh, 1], name='target_segment')
        ## train inference
net = model.u_net(t_image, is_train=True, reuse=False, n_out=1)
        ## test inference
net_test = model.u_net(t_image, is_train=False, reuse=True, n_out=1)

    ###======================== DEFINE LOSS =========================###
    ## train losses
out_seg = net.outputs
dice_loss = 1 - tl.cost.dice_coe(out_seg, t_seg, axis=[0,1,2,3])#, 'jaccard', epsilon=
iou_loss = tl.cost.iou_coe(out_seg, t_seg, axis=[0,1,2,3])
dice_hard = tl.cost.dice_hard_coe(out_seg, t_seg, axis=[0,1,2,3])
loss = dice_loss

        ## test losses
test_out_seg = net_test.outputs
test_dice_loss = 1 - tl.cost.dice_coe(test_out_seg, t_seg, axis=[0,1,2,3])#, 'jaccard',
test_iou_loss = tl.cost.iou_coe(test_out_seg, t_seg, axis=[0,1,2,3])
test_dice_hard = tl.cost.dice_hard_coe(test_out_seg, t_seg, axis=[0,1,2,3])

###======================== DEFINE TRAIN OPTS =======================###
t_vars = tl.layers.get_variables_with_name('u_net', True, True)
with tf.device('/cpu:0'):
    with tf.variable_scope('learning_rate'):
        lr_v = tf.Variable(lr, trainable=False)
    train_op = tf.train.AdamOptimizer(lr_v, beta1=beta1).minimize(loss, var_list=t_vars)
```

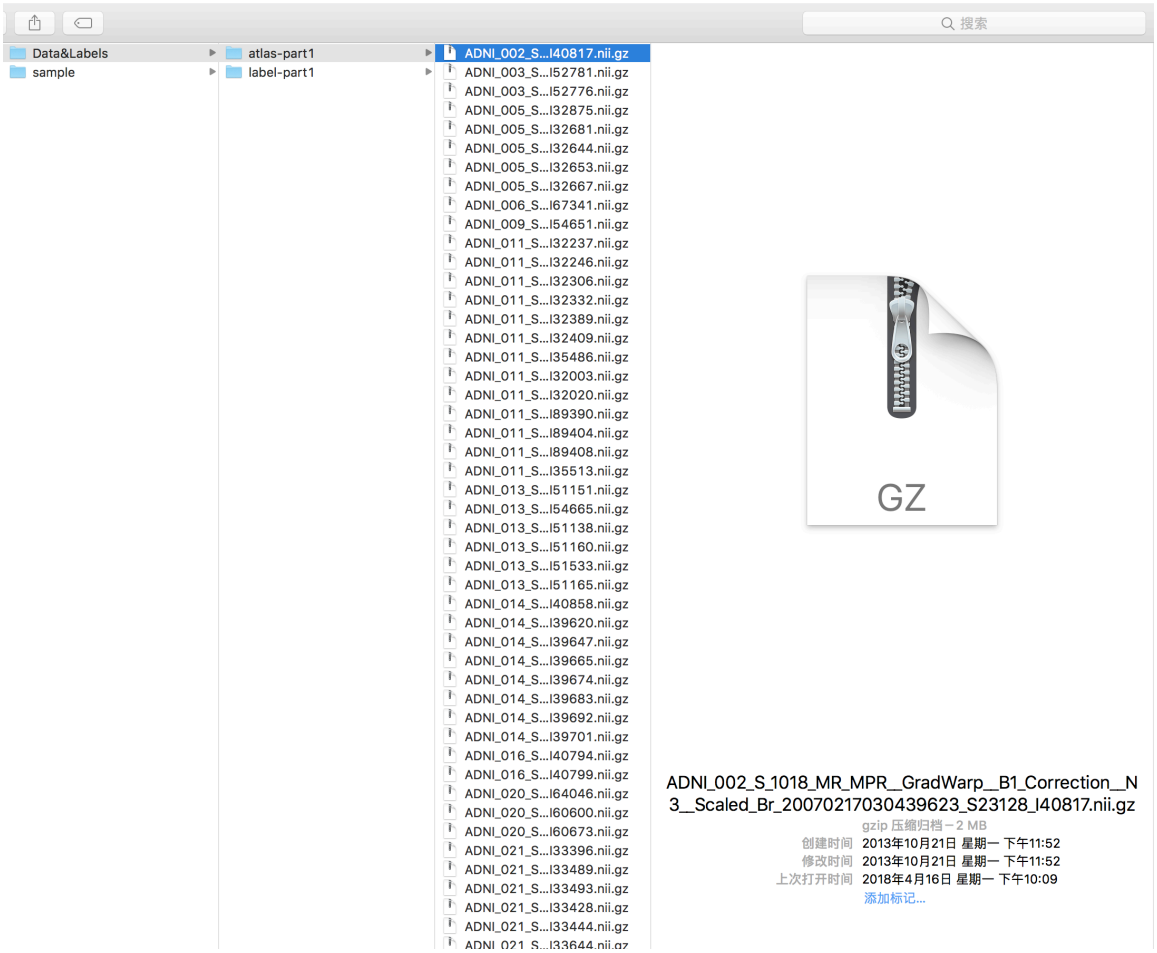
建模

```
        ###======================== SAVE MODEL ========================###
        tl.files.save_npz(net.all_params, name=save_dir+'/u_net_{}.npz'.format(task), sess=sess)


if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser()

    parser.add_argument('--task', type=str, default='all', help='all, necrotic, edema, enhance')

    args = parser.parse_args()

    main(args.task)
```

保存模型

3.1.4 队列图像

来自华为提供的图像集

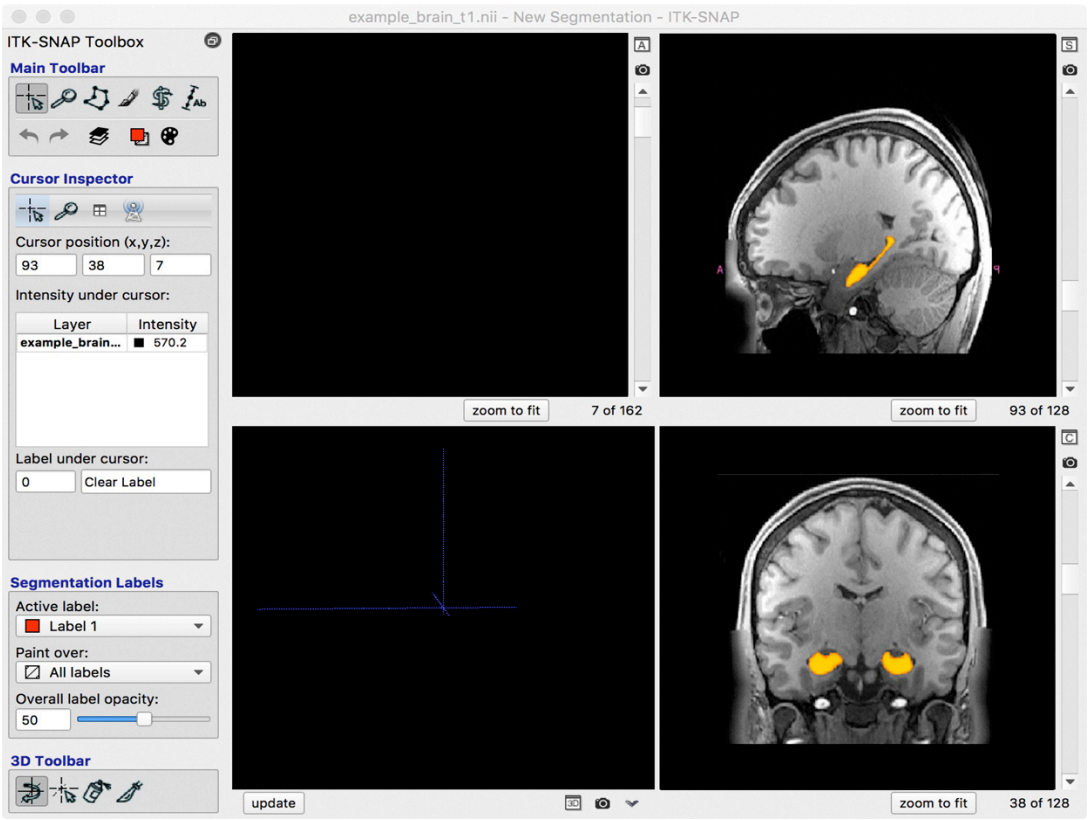


3.1.5 高精度单张图像

为了给我们的模型提供准确的分割知识，除了从 FreeSurfer 处理的实际数据中获得的训练实例外，我们通过增强单个图像创建了更高分辨率的单幅图像和相应的更高精度的手动执行的海马分割目标。

我们通过平均我们自己的“MPRAGE 重复测量”数据集创建了高 SNR，更高分辨率（0.6 毫米）的大脑图像。该数据集包括单个受试者，在各种 T1-MPRAGE 对比下，使用 3T MRI 扫描 123 次，改变参数如 TI，TR，体素大小或并行成像量。虽然这些采集的预期目标是量化 T1 序列变化下的形态测量算法的行为，并与扫描持续时间进行权衡，但在此我们使用数据集作为通过平均提供高 SNR 图像的方式。具体而言，对 35 个最高分辨率采集进行重采样和平均，以产生 0.6mm 各向同性分辨率下的高 SNR 输入图像。尽可能准确地跟踪可见的海马边界地标，然后手动分割该图像。我们推断 ConvNet 可以认识到这些边界的有用性，并将这些知识用于其细分模型。由于 FreeSurfer 派生的示例集的分辨率有限，部分音量效果，来源的质量可变以及自动分割的不准确性，该知识可能无法在 FreeSurfer 派生的示例集中清晰地包含。

海马分割可能具有挑战性。我们已经描述了指南和地标以分割单个病例。然而，由于我们的综合多平均图像具有更好的信噪比和对比度，因此客观分割它们相对容易。实际上，海马旁回的海马边界在图像中是清晰的，并且与心室角的边界也具有良好的对比，如同 Pulvinar 的后边界一样。在海马 - 杏仁核交界处，我们遵循视觉可观察的高信号边界。其他边界使用了 alveus，fimbria 和其他相对高强度。我们在 ITKsnap 上进行了分割，使用三轴视图来更好地理解每个体素标记。

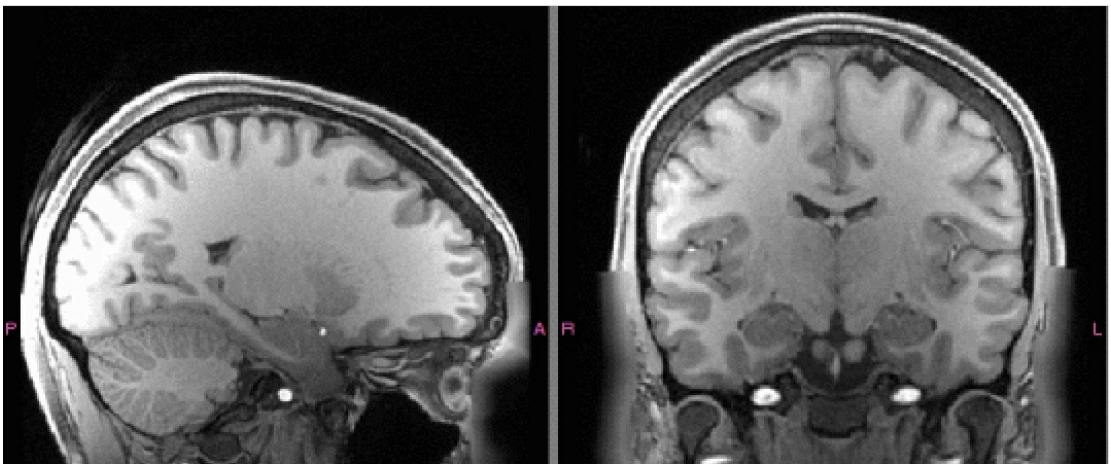


3.1.6 运行平台

核心 ConvNet 通过使用 CuDNN 5.1 的 Lasagne 库在 Tensorflow 上实施。所有增强都是在 CPU 上离线进行的。图像处理和统计分析使用 Python 软件包 Scipy，Nibabel，Pandas 和 Statsmodels。

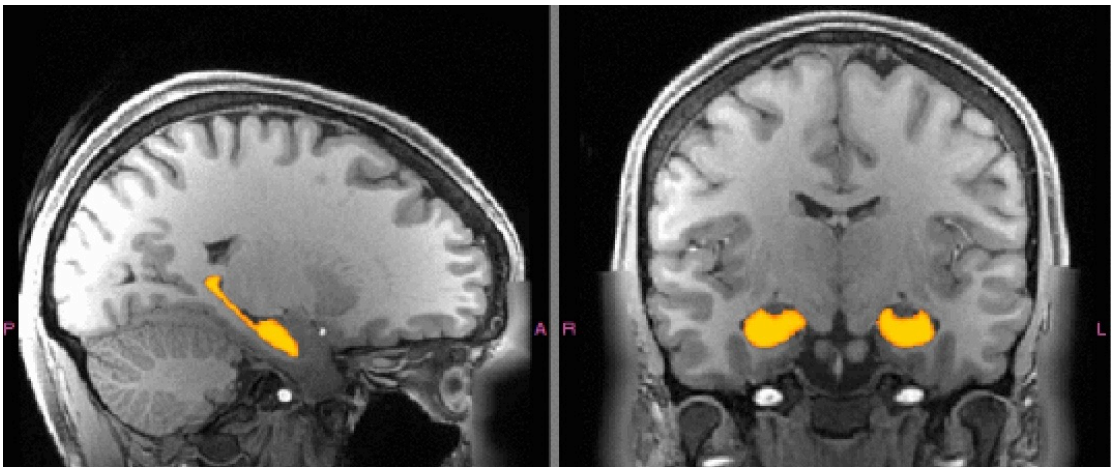
4 训练结果

如图所示，通过我们的数据增强方案显示由同一来源产生的四种合成海马变体。该图的其余部分说明了从我们目视检查的数十个随机测试对象中挑选的各种测试对象分割。结果似乎是合理的，因为分割趋向于沿着界标，并且即使利用低分辨率信息也可以准确定位杏仁核边界。



图一

对于本节剩余部分，我们进行了验证实验，并将结果与 FreeSurfer 进行了比较。我们在大量的脑部 MRI 图像上运行了我们的程序。以下实验中呈现的所有结果均使用相同的最终模型进行计算，仅在未包含在训练集中的任何形式的 MR 图像上评估。



5 主要参考文献

[1]潘红.脑部 MRI 海马体三维分割算法研究[D].成都：电子科技大学，2015.
[2] B. E. Hirsch. Gray's Anatomy: The Anatomical Basis of Clinical Practice [J]. Journal of the American Medical Association, 2009, 17:1829-1830.
[3] 海马体基础知识[OL]. http://www.baike.com/wild/海马体.
[4]林伟铭，高钦泉，杜民. 卷积神经网络诊断阿尔兹海默症的方法[J]. 计算机

应用, 2017 , 37 (12) : 3504-3508.
[5] H MATSUDA. MRI morpheme try in Alzheimer’s disease[J].Ageing Reviews,2016,30:17-24.
[6]JIE B, WEE C Y , SHEN D G ,et al. Hyper-connectivity of functional networks for brain disease diagnosis[J].Medical Image Analysis ,2016,32:84-100.
[7] E A Maguire, H J Spiers, C D Good, et al. Navigation expertise and the human hippocampus: a structural brain imaging analysis [J]. Hippocampus, 2003, 13(2):250-259.
[8]R E Hogan. Magnetic resonance imaging deformation-based segmentation of the hippocampus in patients with mesial temporal sclerosis and temporal lobe epilepsy[J]. Journal of Digital Imaging,2000, 13(1 Supplement):217-218.
[9]N R Pal, S K Pal. A review on image segmentation techniques[J]. Pattern Recognition, 1993,26(93):1277-1294.
[10]M Kass, A Witkin, D Terzopoulos. Snakes: Active contour models[J]. INTERNATIONAL JOURNAL OF COMPUTER VISION, 1988, 1:321-331.
[11]C R. Jr Jack, M D Bentley,C K Twomey,et al. MR imaging-based volume measurements of the hippocampal formation and anterior temporal lobe: validation studies[J]. Radiology,1990,176(1):205-209.
[12]W J Niessen, J P W Pluim, M A Viergever ,et al. Medical Image Computing and ComputerAssisted Intervention — MICCAI 2000[M]. Springer Berlin Heidelberg, 2000.
[13] JWHaller, G . Christrnsen, S C Joshi, et al. Digital Atlas-based Segmentation of the Hippocampus [J[. Computer Assisted Radiology, 1995, 152-157
[14] X Han, L S. Hibbard, N P Oconnell. Automatic Segmentation of Parotids in Head and Neck CT Images using Multi-atlas Fusion[C]. Medical Image Computing and Computer Assisted Intervention, Beijing, 2010.
[15] J Yang, Y Zhang, L Zhang, et al. Automatic Segmentation of Parotids from CT Scans Using Multiple Atlases[C]. Medical Image Computing and Computer Assisted Intervention, Beijing, 2010.
[16] A J Asman, B A Landman. Robust Statistical Label Fusion Through Consensus Level,Labeler Accuracy, and Truth Estimation (COLLATE)[J]. IEEE Transactions on Medical Imaging ,2011, 30(10):1779-1794.
[17]B Fischl, D H Salat, E Busa, et al. Whole Brain Segmentation-Automated Labeling of Neuroanatomical Structures in the Human Brain[J]. Neuron, 2002, 33(3):341-355.
[18]F LI, L TRAN,K H THUNG, et al.A robust deep model for improved classification of AD/MCI patients [J].IEEE Journal of Biomedical and

Health Informatics,2015,19(5):1610-1616.
[19]S Q LIU, S D LIU, W D CAI, et al. Multimodal neuroimaging feature learning for multiclass diagnosis of Alzheimer’s disease [J].IEEE Transactions on Biomedical Engineering,2015,62(4):1132-1140.
[20]A KRIZHEVSKY, I SUTSKEVER, G E HINTON. ImageNet classification with deep convolutional neural networks[C]// / / Proceedings of the 2012 25th International Conference on Neural Information Processing Systems.New York: Curran Associates Inc., 2012: 1097-1105.