

**贵州大学人工智能课程实训**

系统设计报告

|  |  |
| --- | --- |
| 开发人员: | 陈明立 |
| 作品名称: | 基于改进的ResUNet的脑肿瘤图像分割系统 |
| 填写日期: | 2023/07/25 |

**目录**

[第1章 作品概述 4](#_Toc141273725)

[1.1 作品介绍 4](#_Toc141273726)

[1.2 产生背景 4](#_Toc141273727)

[1.3 用户群体 6](#_Toc141273728)

[1.4 主要功能 7](#_Toc141273729)

[1.5 应用价值及推广前景 7](#_Toc141273730)

[第2章 问题分析 8](#_Toc141273731)

[2.1 问题来源 8](#_Toc141273732)

[2.2 现有解决方案 10](#_Toc141273733)

[2.2.1 图像语义分割传统方法 10](#_Toc141273734)

[2.2.2 基于深度学习的语义分割技术 13](#_Toc141273735)

[2.3 本作品要解决的痛点问题 17](#_Toc141273736)

[2.4 解决问题的思路 18](#_Toc141273737)

[2.4.1 作品功能及性能需求 18](#_Toc141273738)

[2.4.2数据集 18](#_Toc141273739)

[第3章 功能介绍 19](#_Toc141273740)

[3.1 总体介绍 19](#_Toc141273741)

[3.2 系统工作流程 20](#_Toc141273742)

[3.3 图像分割模块 20](#_Toc141273743)

[3.3.1 FCN 20](#_Toc141273744)

[3.3.2 U-net 21](#_Toc141273745)

[3.3.3 ResUNet 22](#_Toc141273746)

[3.3.4 Pro-ResUNet 23](#_Toc141273747)

[3.4 界面功能模块 24](#_Toc141273748)

[第4章 技术方案 25](#_Toc141273749)

[4.1 系统总体设计方案 25](#_Toc141273750)

[4.2 UI界面设计 26](#_Toc141273751)

[4.2.1 主界面模块 26](#_Toc141273752)

[4.2.2 文件加载模块 27](#_Toc141273753)

[4.2.3 绘制图像模块 28](#_Toc141273754)

[4.2.4 绘制3D模型模块 30](#_Toc141273755)

[4.3 Pro-ResUNet图像分割模型训练 31](#_Toc141273756)

[4.3.1 数据预处理 31](#_Toc141273757)

[4.3.2 构建模型 33](#_Toc141273758)

[4.3.3 训练过程 34](#_Toc141273759)

[4.3.4 计算评价指标 39](#_Toc141273760)

[4.3.5 技术方法 40](#_Toc141273761)

[第5章 测试分析 43](#_Toc141273762)

[5.1 图像分割分割模型训练评价指标 43](#_Toc141273763)

[5.1.1 Dice系数 43](#_Toc141273764)

[5.1.2 PPV(Positive predictive value) 44](#_Toc141273765)

[5.1.3 Sensitivity 44](#_Toc141273766)

[5.1.4 Hausdorff\_95 44](#_Toc141273767)

[5.2 模型训练参数设置 45](#_Toc141273768)

[5.3 图像分割训练结果对比 46](#_Toc141273769)

[5.4 图像分割训练结果分析 48](#_Toc141273770)

[第6章 作品总结 49](#_Toc141273771)

[6.1 作品特色与创新点 49](#_Toc141273772)

[6.1.1 数据预处理 49](#_Toc141273773)

[6.1.2 构建模型 49](#_Toc141273774)

[6.2 作品展望 49](#_Toc141273775)

# 作品概述

## 作品介绍

近年来，基于深度学习的方法已经广泛地应用在计算机视觉的不同任务中，如图像处理、目标检测等。众多深度学习方法的成功应用得益于卷积神经网络(Convolution Neural Network, CNN),Lecun等首次将CNN成功应用于图像分类识别，这使CNN得到了极大的关注。同时也可将CNN应用于图像分割。与传统的分割方法相比，基于深度学习的分割方法具有自主学习与特定任务相适应的特征，可以高效地完成任务。深度学习方法在脑肿瘤分割中也得到了广泛应用，如Zhang等利用原始大脑的MR图像训练粗糙分割的FCN模型，检测出肿瘤的整体区域，再使用肿瘤区域作为训练样本训练精确分割的FCN得到脑肿瘤的内部结构。但是在两阶段的分割过程中缺失了脑肿瘤图像局部信息，分割精度不高。Vittikop等使用U-Net网络对脑肿瘤MR图像进行分割，通过跳跃连接(Skip Connection)融合了浅层和深层的图像特征信息，取得了不错的效果。然而简单的图像拼接会得到更多的冗余信息，影响分割效果。针对上述方法的不足之处，本文提出了基于改进ResUNet模型的脑肿瘤MR图像分割方法。

## 产生背景

脑肿瘤或称为颅内肿瘤（Brian Tumor）是一种恶性的人体疾病，由人脑内异常细胞聚集而成。脑肿瘤按照其致命程度可以分为恶性肿瘤和良性肿瘤，其中恶性肿瘤按照病变最开始发生的区域可以分为两类：继发性肿瘤与原发性肿瘤。继发性肿瘤或称为转移性肿瘤，是指从其他有癌变组织的器官转移而来的癌细胞形成的病变组织，通常由肺、肾脏等器官转移而来。原发性肿瘤是指起源于大脑自身组织（比如说：脑神经、脑膜、脑下垂体）的肿瘤，常见的原发性肿瘤包括胶质瘤（glioma）、胚芽肿瘤、脑膜瘤、神经鞘瘤、脑下垂体肿瘤等，其中胶质瘤是最常见的原发性颅内肿瘤，跟据文献统计结果，大约有27%的脑肿瘤为胶质瘤，而且在脑部恶性肿瘤当中占比约为81%。根据世界卫生组织（WHO）的报告脑胶质瘤按照其恶性程度可以分为四个级别，分别为WHO I~IV级，其中，WHO I级和WHO II级属于低级别胶质瘤（Low-grade glioma，LGG），WHO III级和WHO IV级属于高级别胶质瘤（High-grade glioma，HGG）。目前对于脑胶质瘤的治疗按现在的医学水平来说仍旧不够乐观，仅有部分LGG患者可以通过手术治疗成功，然而对于大部分胶质瘤患者来说，统计结果显示其5年生存率仅有9.8%。 脑肿瘤的治疗方法通常可以分为手术治疗、化学治疗以及放射线治疗，其中手术治疗依旧是最主要的治疗方法，其主要目的为通过外科手术的手段，尽可能地切除肿瘤组织，将肿瘤细胞移除干净，从而避免肿瘤细胞继续增生。此方法对于低级别胶质瘤比较有效，几乎可以达到完全切除肿瘤细胞的手术效果，通常再辅助一些化学治疗和放射线治疗，可以达到较好的治疗效果。但是这种手术方法只有在病变早期，肿瘤较小并且与周边组织易于区分的情况下才能达到理想效果。如果肿瘤生长在大脑的敏感区域，外科手术则难以进行，并且脑肿瘤没有特定的临床特征可以用来判断是否有病变发生，当出现可查觉症状时，病情很有可能已经加重。因此对于医生与患者来说，尽早检测出脑肿瘤是至关重要的事情。 脑肿瘤诊断最重要的方式之一是影像诊断，借助医学影像可以更准确并快速地进行脑肿瘤诊断，并且对手术后的追踪治疗也有很大帮助。常用的医学影像技术主要包括：核磁共振成像（Magnetic Resonance Imaging，MRI）、计算机断层扫描（Computed Tomography，CT）、脑波图等方式。在诊断脑胶质瘤这一应用场景下，核磁共振成像（MRI）表现最为出色，其对颅内软组织的敏感性更高，对于脑肿瘤的检测效果最佳。并且MRI成像又包括多种模态，不同模态呈现的图像信息不同，呈互补的关系。主要模态包括：T1加权成像（T1），此模态可更好地呈现脑部解剖结构；T2加权成像（T2），此模态对含水组织具有更高的敏感度，因此可以更好地呈现病变组织；FLAIR（流体衰减反转恢复图像）可以抑制脑脊液，使得该组织与其他组织更易于区分；增强T1加权成像（T1CE），此模态使用造影剂对图像进行对比度增强，增加了血管组织的强度。核磁共振成像（MRI）凭借其特质已经成为了最常用的脑肿瘤诊断影像。如图1.1所示，分别展示了MRI图像的四种不同模态，a)为FLAIR序列，b)为T1序列，c)为T1c序列，d)为T2序列。

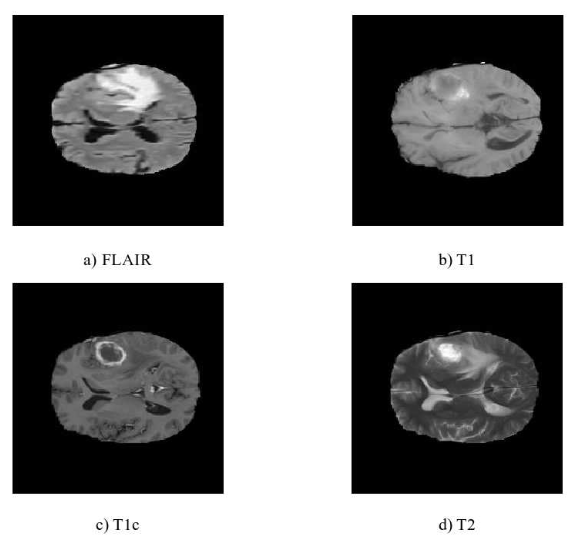


图 1.1 MRI成像的四种序列图

临床手术中，医生借助MRI影像手动分割脑肿瘤，这样的模式不但受制于MRI影像成像的质量，也依赖于医生的临床经验，该过程受制于医生的个人主观性思考，并且手术成本也比较高昂。因此，如果可以设计一种使用计算机技术全自动分割脑肿瘤的方法来支撑医生的诊断与治疗，那么在降低脑肿瘤的恶化几率、增加治疗成功几率以及改善术后恢复程度等方向上都有很大的意义，并且还可以减少医生的工作压力并提升工作效率。因此，研究怎样结合多个模态的MRI影像，设计出一种可以全自动分割脑肿瘤的方法，具有很重要的意义。目前，借助计算机技术的分割算法也逐渐受到医生和专家的认可，正成为一项热门的研究课题。

## 用户群体

1）医疗机构。医院、诊所等医疗机构可以作为潜在用户，使用本作品来帮助医生和放射科医师在临床实践中自动识别脑肿瘤。这可以提高诊断的准确性和效率，辅助医生制定合适的治疗计划。

2）医学研究机构。医学研究机构可能对本作品感兴趣，用于研究脑癌的临床表现、进展和治疗效果等方面的研究。本作品可以帮助研究人员从医学图像中自动提取脑肿瘤信息，为临床研究提供有力的工具和数据支持。

3）医学影像技术提供商。医学影像技术提供商可以将本作品作为他们的产品或服务的一部分，为医疗机构提供脑肿瘤识别的解决方案。这可以帮助提升他们的产品或服务在医学影像领域的竞争力。

4）医学影像研究人员和学者。对于从事医学影像研究的学者和研究人员，本作品可以作为研究工具，用于脑癌的自动识别和分析。这可以为他们的研究提供高效、精确的医学图像分析工具，从而加快研究进展。

5）医学影像技术开发者和研究人员。对于从事医学影像技术开发的研究人员和开发者，本作品可以作为研究和开发的基础，用于脑癌识别和自动提取的算法研究和验证。这可以帮助他们改进和优化医学影像处理算法，并推动医学影像技术的发展。

## 主要功能

本作品针对于医疗图像中的脑肿瘤图像，实现了脑癌分割，可以批量处理多张图像，通过深度学习方法自动识别脑肿瘤，并提取相应的特征信息。

## 应用价值及推广前景

脑肿瘤图像通常因为质量易受干扰、结构比较复杂等因素而难以正确分析，即使对于经验丰富的医生来说，通过图像进行分析也需要大量的时间和精力。而本作品通过深度学习技术，大幅提高图像的处理速度和精度。

基于上述优点，本作品可以为医生提供快速而准确的脑肿瘤标注，从而显著减轻医生在诊断脑疾病时的压力。此外，由于处理速度快且精度高，可以大大帮助医生减少人为错误，提高医生的工作效率，节省宝贵的诊断时间，提高诊断的准确性和一致性。

综上所述，本产品的应用价值在于能够提供快速、准确、自动化的脑肿瘤识别，有望在医学诊断领域带来显著的实际应用价值。本作品效果图如图1.2所示。



图 1.2 作品效果图

# 问题分析

## 问题来源

脑肿瘤是一种在脑组织中非自然分裂和不正常生长的异常细胞，发病率较高，死亡率超过3%,严重危害人体的健康。其中胶质瘤是源于颅内的常见脑肿瘤之一，可分为高级别胶质瘤(HGG)和低级别胶质瘤(LGG),HGG患者的生存率极低，但通过对LGG早期的诊断和治疗，能够大幅提高病人的寿命。因此在早期对脑肿瘤进行精确的诊断尤为重要。核磁共振(Magnetic Resonance, MR)成像作为一种非入侵式的脑肿瘤成像技术，可产生高分辨率、无损伤的脑影像，为医生的临床诊断提供准确的信息，是鉴别脑肿瘤的重要技术之一。根据不同的成像条件，脑部图像可分为Flair模态、T1加权模态、T1ce模态及T2加权模态，临床上医生通常结合四种不同模态图像来进行诊断。由于脑肿瘤形状多异、位置大小分布不均、边界复杂等因素，目前脑肿瘤图像主要还是由专家手动进行分割，费时费力，误判率较高，不同专家对脑肿瘤图像的分割结果存在较大的差异，影响患者后续的治疗。因此，研究全自动的脑肿瘤分割方法对临床诊断和治疗脑肿瘤具有非常重要的意义。

分割脑肿瘤的算法有很多，目前比较常用的是基于阈值、区域生长和基于深度学习的分割方法。基于阈值的方法通过设置不同的阈值，然后与图像的灰度值进行比较从而将图像的像素点分类，以达到分割的目的。一般可分为固定阈值法和自适应阈值法,如Saad**[1]**等将每张图像分成多个子区域，每个区域使用直方图阈值技术检测出图像的高强度和低强度像素数，通过对比正常区域与病变区域来确定最佳阈值。但由于脑肿瘤结构较复杂，基于阈值的分割方法通常用于确定脑肿瘤的位置，不能有效解决脑肿瘤分割问题。

区域生长的分割方法通过在图像的目标区域设置1个或多个种子点，然后搜索该种子点的相邻像素，把相邻区域中具有相似特点的像素合并至该区域，直到周围没有新的像素点能够添加到该区域为止，实现区域的生长。由于噪声对区域生长方法的影响较大，Weglinski**[2]**等使用中值滤波技术来处理输入的脑肿瘤图像，然后再使用区域生长法对图像进行分割。Reddy**[3]**等在基于区域生长法的基础上，通过将输入的图像划分为网格，然后在每个网格中选取1个种子点进行区域生长。然而区域生长分割方法对图像的阴影区域分割会出现分割不完整或过分割的问题。

基于深度学习的脑肿瘤图像分割，核心问题是在可接受的消耗内，通过深度学习技术对脑肿瘤图像的不同层次的特征信息进行提取、融合并做出分类判断，以提高算法模型最终的分割准确程度。在本文中，我们利用深度学习技术提出和实现了多种用于脑部肿瘤分割的图像分割模型，在缓和算力需求的情况下，使得计算机能够自动化地将脑部肿瘤图像进行准确的分割。这种方式能够在一定程度上减轻医生对于病例诊断的负担，从而承担起一部分医学图像辅助诊断和术前规划及预后的职责，并为之后的技术发展提供技术和理论支持。综上所述，本研究具有重要的学术意义和开阔的应用场景。

## 现有解决方案

### 图像语义分割传统方法

（1）传统分割方法的定义及分类

模糊地讲，语义分割任务就是讲图像中的不同像素点进行分类。多年来语义分割在不同人人群中有着不同的解释和表述，为了能够更加清晰地定义这一任务，Gonzalez R C**[4]**等在数字图像处理一书中通过集合对其给出了比较正式的定义：设集合R代表整个图像的区域，则对于R的分割可以视作将集合R分成N个满足以下条件的非空子集：

1. 对于所有的i和j，且i≠j，有∩=Ø
2. 对于i=1,2,3,...,N，有
3. 对于i≠j，有
4. 对于i=1,2,3,...,N，是连通区域

其中，是对所有集合R中元素的逻辑谓词，用于判断不同的像素点是否为同一个类别，Ø代表空集。以上五个条件分别意味着分割任务的不同要求。条件（a）代表语义分割需要对图中所有的像素点进行分类，且所有分类结果的并集应等于原图像；（b）代表了不同分割结果不能够相互重叠，也就是单个像素点有且只能有一个类别；（c）代表了具有同一个类别像素点应该具有某些相同的语义特征；（d）代表了被分类到不同类别的像素点应该具有不同的语义特征；（e）代表了不同语义类别区域应当是互相连通的。一个合格的语义分割算法应当能够在满足以上五个条件的情况下尽可能精准地从图像中提取出感兴趣的区域。

传统的图像分割方法大致可以以下几个类别：基于阈值的分割、基于边缘的分割、基于区域的分割、基于图论的分割、基于能量泛函的分割、基于数学形态的分割以及运动分割。这些方法的共同点在于他们对像素做出分类的主要依据是图像的一些低级特征，也就是图像的灰度、纹理、颜色以及形状等，并通过数学理论支撑建立模型，从而对像素做出分类。

（2）传统图像分割方法

本节选取了传统分割方法中最为常用的几种方法进行详细阐释，以下将分别对他们进行介绍。

1. 基于阈值的分割技术，基于阈值的分割技术在传统图像分割算法中是数量最多的一类，主要是因为在当时普遍基于图像灰度进行分割的时代，基于阈值的分割技术对于多维特征空间（例如彩色图像）也有较为良好的适配能力。 基于阈值的分割技术的主要思想是通过在图像灰度分布之中找到一个合适的灰度值，能够将图像中所有的像素点划分为两个类别：灰度值大于该值的和灰度值小于该值的，从而实现在图像中对像素进行分类的效果，这个特定的值就是选取的阈值。通常而言，基于阈值的分割技术通过设立阈值的数量来对应需要进行分割的目标的数量。只采用一个阈值进行分割的方法被成为单阈值方法，而采用多个阈值的方法则被称为多阈值方法。其中单阈值分割法最为出名的一个算法就是直方图双峰法。

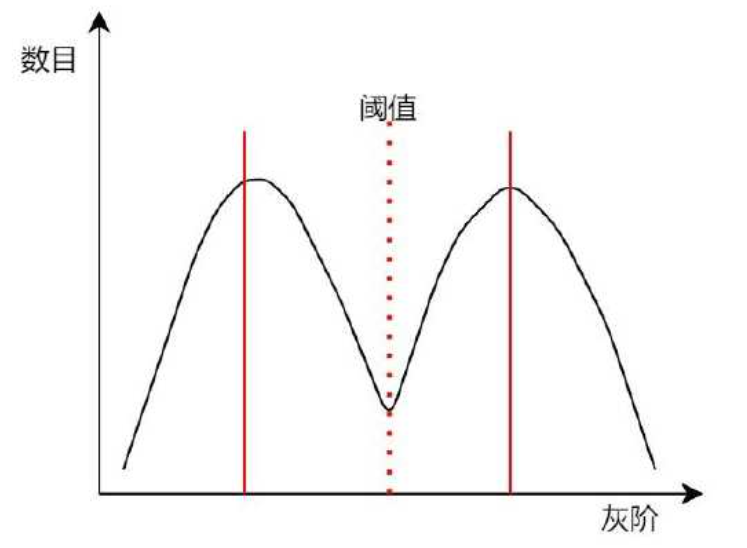


图 2.1 双峰形态分布直方图

对于一个图像f(x,y)，通过将f(x,y)进行灰度直方图化，如果得到如图2.1所示具有明显的双峰形态分布的直方图，就选取中间对应的谷值作为阈值T，之后将f(x,y)所有的像素点按照阈值从而可以得到每个点的像素类别g(x,y)，得到最终的二值分割图像。直方图双峰法的分割方法简单明了，对于结构简单的图像具有不错的分割效果，但对于具有复杂的分割任务而言，因为其选取阈值的特性往往不能取得很好的效果。 这一类分割方法在自然图像、文字提取等不同领域都取得了不错的成效，但脑肿瘤图像这种更为复杂的图像而言，阈值分割精度有限，往往不能取得很好的效果。

1. 基于区域生长的分割技术，区域生长算法主要是根据同一区域内不同像素间相似程度来聚类像素点，从而达到分割目的算法。区域生长算法又可以进一步分成两种生长算法，种子法和区域重叠法。种子法在算法开始时，需要在每个分类的区块中确定一个初始像素点，也就是种子像素点，之后通过合并该像素点周围区域相似的像素点和区域，由此促使不同语义区域不断生长，直至满足条件。使用种子区域生长算法主要有三个关键点：首先需要能够正确选定每个语义区域内代表性的种子像素点；其次需要确定能够正确促使相同语义的像素点被划入到对应的语义区域内的生长规则；最后需要确定使生长停止的触发条件。 种子的选取可以通过人工进行，也可以通过灰度图判定等方式进行，还能够在首次测试之前随机选取种子完整地进行一次区域生长之后，选择距离聚类重心最接近的像素点作为种子重新进行训练。总之种子的选取没有固定的规则，通常都是随着分割任务的特性改变而发生变化。 分裂合并法是出了种子像素点作为初始点之外的另外一种区域分割方法。分裂合并法通过将图像进行不断的等大小拆分，得到不同的区域，直至单个像素点，再将满足生长条件的区域连接起来的方式。下面以图像四叉树算法为例，介绍其具体算法步骤：对任意一个区域，当时，将其这个区域进行面积相等的四个子区域；对相邻的两个区域和。当时，则将他们合并起来；如果无法完成第一或第二步，结束算法。

区域生长算法能够通过生长的方式将同种区域连接起来，从而得到良好的边界划分的结果，但是这种算法容易受到噪声的影响，对于实际生活中会受到各种干扰的脑肿瘤图像并不算适用。

1. 基于边缘检测的分割技术，基于边缘检测分割技术的基本思想是定位图像中不同目标的边缘像素，并将其进行连接以构成不同目标的边界。基于边缘检测的分割技术主要利用了不同语义信息的区域之间边界会发生灰度的强烈变化这一特点来进行。 基于边缘检测的分割技术对于图像中的边缘特征有着较好的定位，但是通常不能很好地对有相同语义区域的边缘进行闭合，会在算法过程中产生大量细碎的边缘点，所以得到相应的信息之后还需要借助其他方式，例如边缘闭合技术，进一步完成分割任务。

### 基于深度学习的语义分割技术

（1）深度学习技术的发展

深度学习技术主要是基于传统神经网络技术的衍生和发展。上世纪中叶，Warren Mcculloch**[5]**等提出了人工神经网络的数学模型，奠定了后世神经网络结构的理论基础。Frank**[6]**模拟人类感知能力提出了一种机器，并命名为感知机。感知机结构较为简单，它是由阈值单元和线性单元构成的单层网络。单层感知机虽然只能够对线性的任务进行分类，但它的出现代表着现代CNN结构的雏形。1969年，Marvin Minsky**[7]**等通过改进了单层感知机在表达能力上的缺陷，提出了多层感知机模型，多层感知机通过加入隐藏层，增强了神经网络模型的表达能力，使得多层感知机能够处理异或问题，其结构如图2.2所示。之后，众多学者的研究发现MLP也存在一定的问题，不能对隐藏层的权值进行训练。

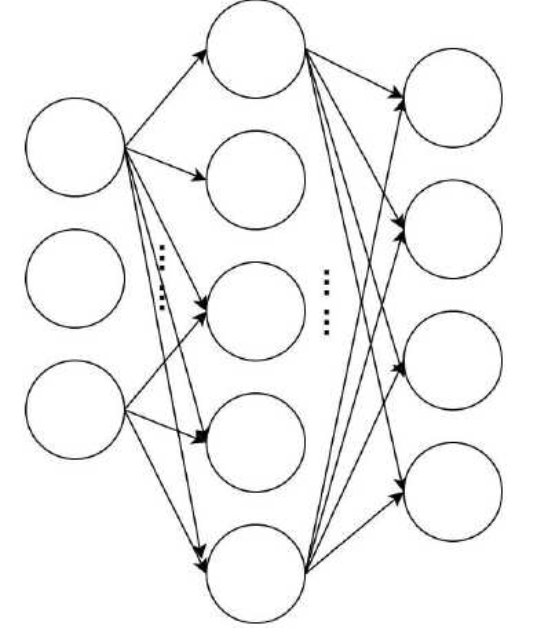


图 2.2 多层感知机

Hinton**[8]**于1986年提出了用于改良感知机模型的反向传播算法解决了隐藏层难以训练的遗留问题，并采用Sigmoid非线性函数对模型的非线性映射，解决了之前感知机模型只能表达线性结果的困境。反向传播的过程如图2.3所示，通过网络前向传播时通过网络最终结果与正式标签之间的差异衡量出各个神经元需要承担的误差，并通过计算不同神经元的梯度将误差进行分配，从而使得隐藏层中的神经元学习参数可以得到更新。尽管反向传播算法对于过于深层次的网络会出现梯度消失的现象，但不可否认的是反向传播算法的出现对于深度学习而言是一次重大革新。

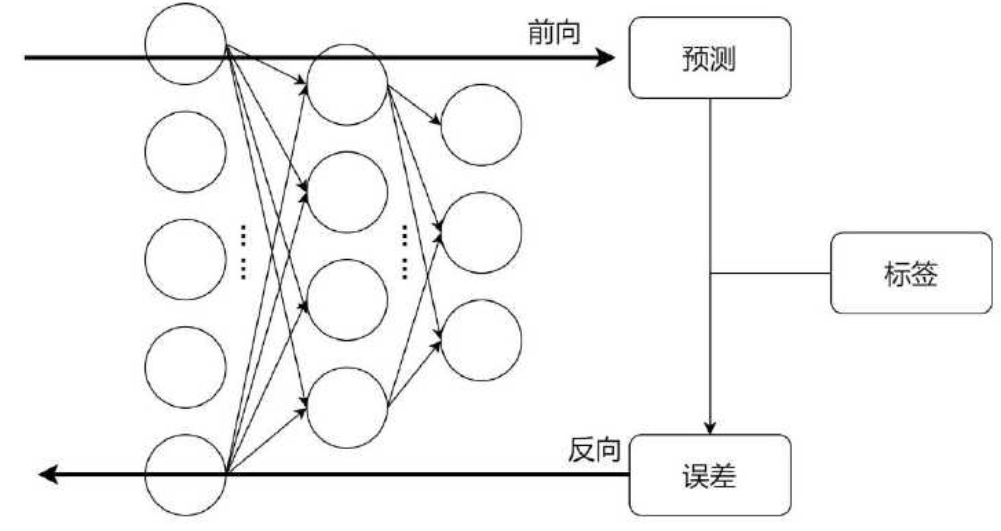


图 2.3 反向传播

反向传播在1989年被LeCun**[9]**引入到了卷积神经网络模型之中，用于监督网络训练。LeCun在该篇论文中还提出了另一个非常重要的理论，即卷积神经网络中的权值共享。权值共享压缩了模型的参数消耗，提供了对过拟合风险的保障，同时也加速了训练过程。

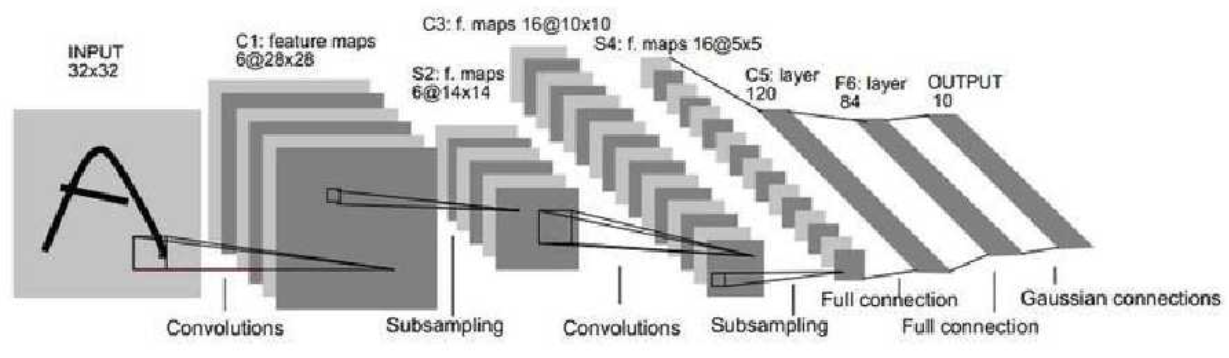


图 2.4 LeNet-5

1998年，LeCun基于之前的成果，提出了LeNet-5网络。LeNet-5结构如图2.4所示，尽管LeNet-5放到如今来看显得有些古老，但是不可否认地是LeNet具备了大部分与如今各类层出不穷的深度学习模型相近的结构，单单从结构上而言几乎与如今的CNN别无二致。LeNet主要由三种模块构成，分别是卷积层、池化层以及全连接层。

尽管卷积层得名于卷积运算，但通常在实际使用中会使用更加简便的互相关运算进行替代。在卷积神经网络中，卷积层是最为核心的组件之一，通常在网络中起到对输入图像提取特征的作用。如图2-5所示，假设左侧的矩阵为输入矩阵，那么我们定义一个大小为2×2，步幅为1，填充为0的卷积核。通过在输入矩阵上不断滑动，将输入矩阵对应位置与卷积核上的值做点乘运算，我们就可以得到最后右侧的输出值。

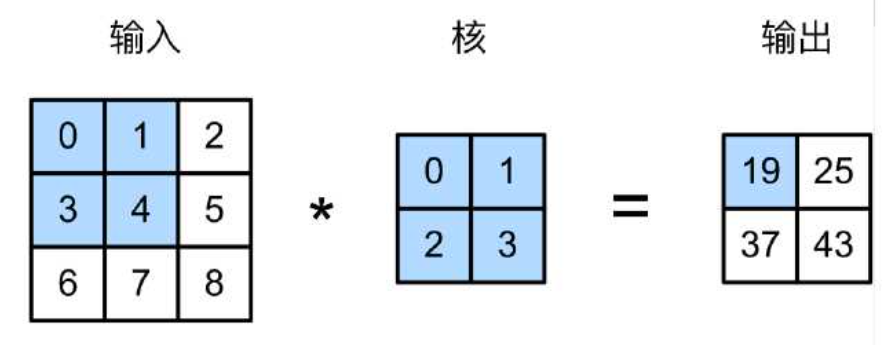


图 2.5 卷积运算

由于卷积层的线性表达特性，在实际网络中通常会在卷积层之后连接一个激活函数，来处理非线性的问题，增强模型的表达能力。卷积层对于特征提取工作至关重要，卷积层的设计事关模型最终的效果。 通过卷积运算，我们可以对输入的图像进行特征抽取和压缩的效果，但由于实际上卷积操作会引入额外的参数，并且会提高过拟合的风险，于是池化操作被引入了卷积神经网络作为主要组件之一。池化主要通过最大化策略和平均策略对图像进行压缩。池化层相对于卷积层有着更小的参数量的同时，还为模型带来了图像平移旋转的宽容度，这使得模型会具有更好的泛化能力。

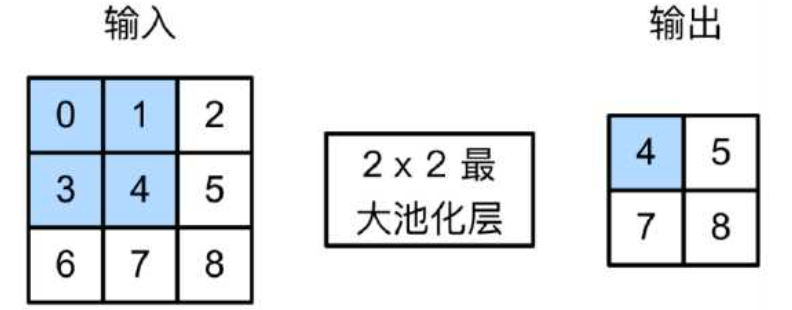


图 2.6 池化层

如图2.6中，我们定义了一个大小为2×2的最大池化层对输入图像进行降维，通过对于输入图像中的对应区域中最大值进行提取，从而在尽量不损失图像信息的情况下对输入图像进行降维。

2012年，Alexnet**[10]**和训练方法被提出标志着深度学习的热潮再次出现在人们的视野中。Alexnet采用了GPU加速模型的训练，并且完全转向了有监督训练的模式。同时在激活函数的选择上，Alexnet选择了ReLU，并实现了DropOut用于避免过拟合。

何凯明[4]等于2015年提出了残差网络，并通过一个轻型的结构块解决了这个问题。通过使用恒等映射函数绕过sigmoid类似的非线性变换层，也就是快捷连接（Shortcut Connection）的方式优化卷积块结构。加入了这种快捷连接的卷积层会形成新的映射方式，从而使得上层的更多信息可以更加便捷地传入到深层的卷积层之中。通过这种方式就可以建立深层网络的同时，避免网络退化。

（2）基于深度学习技术的语义分割

Long**[11]**等提出了FCN，一种通过去除全连接而通过卷积核大小为 1 的卷积层完成对像素分类的网络结构，如图 2.7之后，FCN成为了许多基于编码-解码结构分割网络的基础结构。针对FCN方法存在的不能有效处理场景间的关系以及全区信息的问题，Zhao**[12]**等提出了 PSPnet，通过引入金字塔池化模块进行多尺度的信息提取，针对 FCN的全局信息提取进行了改善。而 LC Chen**[13]** 等提出 DeepLab系列框架，v1通过引入 CRF解决了卷积过程中平移不变性导致的位置信息丢失;v2引入了ASPP空洞金字塔池化和空洞卷积，空洞卷积能够在提供额外参数消耗的情况下扩大卷积核的感受野，从而捕获到更大尺度上的特征信息，而带空洞的金字塔池化能在多尺度上稳定分割目标；v3提出了一个针对物体便捷的检测模块，并使用深度分离卷积核Xception模块。

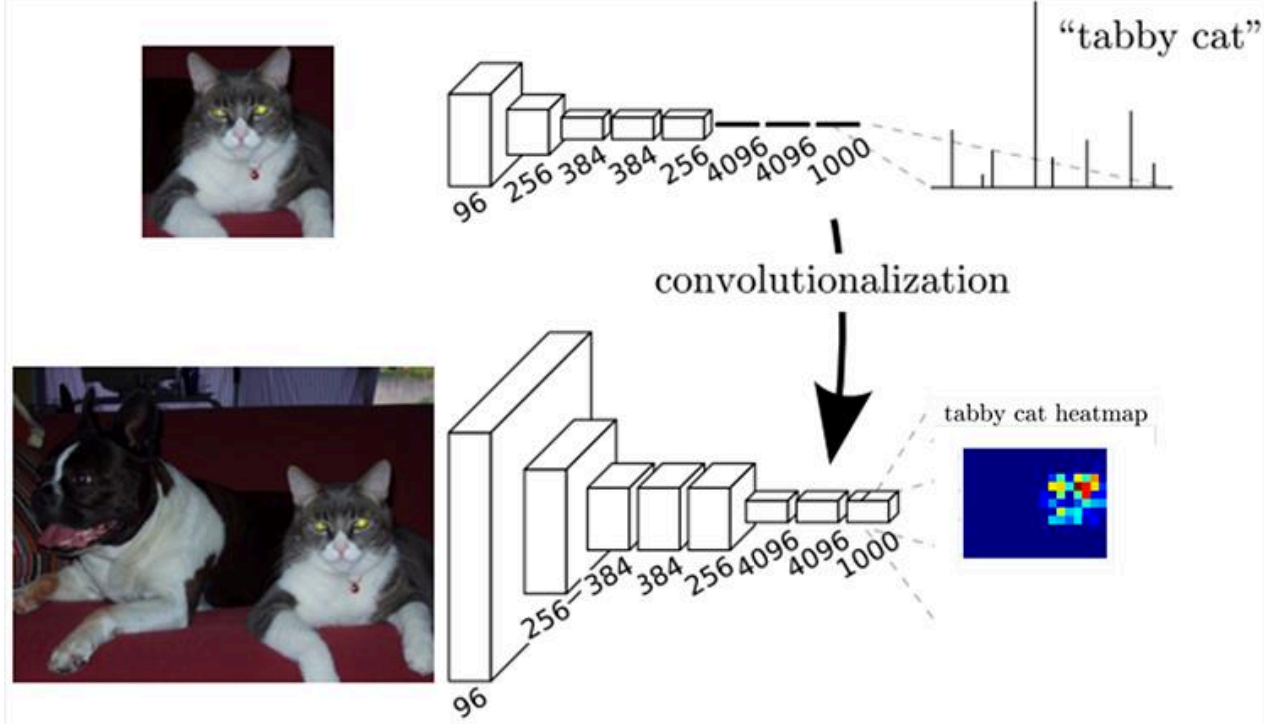


图 2.7 FCN示意图

Ronneberger**[14]** 等上出了 U-net 框架[52] 。U-net 使用了编码解码器结构对原始图像进行压缩和特征提取，通过一个拓扑结构为 U 字形的网络分两路进行上采样及下采样 U-Net 的结构，其结构如图 2.8 所示。U-net 有两个最大的特点，U 型结构和跳层连接,保证了最后恢复出来的特征图融合了更多低级特征，从而能够对任务区域的细节有更好的判断。许多对于医学影像的图像分割都是基于 U-net 的思想和框架上进行的。

除了 FCN 框架的方法，He K等在R-CNN的基础上提出了可用于分割工作的Mask R-CNN。Mask R-CNN 在 Faster R-CNN 的基础上进行了改进，在每个 RoI加入一个用于预测分割掩码的支路，并添加了并列的 FCN 层，使得该层与已有的bounding box层和分类层平行运作，最终实现了像素级别的实例分割。同时 Mask R-CNN还改进了Faster R-CNN中 ROI 池化层，加入了双线性插值，降低了量化取整操作产生的误差。

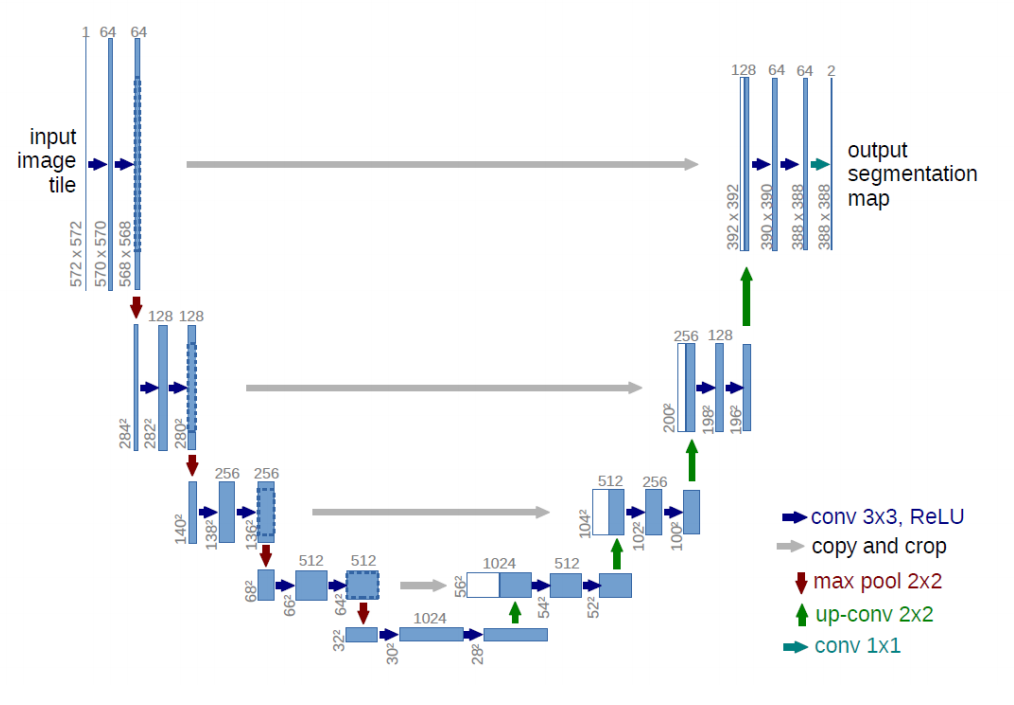


图 2.8 U-net

## 本作品要解决的痛点问题

1）由于脑组织结构复杂，其特征具有多尺度、多方向、多层次的特点，同时受到形态、纹理等多种因素的影响，如何从图像中提取有效的特征是脑肿瘤图像分析的难点之一。

2）脑组织深度较浅，受到血流、组织衰减等多种因素的影响，图像质量容易受到干扰，如噪声、伪影、散斑等，这些因素会影响图像的清晰度和对细节的表现，因此如何减弱对图像质量的影响也尤为重要。

3）脑肿瘤图像分析的最终目标是准确诊断肿瘤，但良性和恶性肿瘤在图像上表现出的特征往往相似，因此如何对于良性和恶性肿瘤进行精确分割，成为了本作品旨在解决的一个痛点问题。

4）脑肿瘤图像的数据量庞大，需要大量的人工标注和分类，因此如何解决数据的高效标注也是本作品需要解决的核心问题。

5）基于深度学习的脑癌监测方法目前虽然有着较好的进展，但是其相关模型的鲁棒性差，训练成本高，有时候还会存在训练过拟合问题，因此选择什么样的深度学习网络及参数，使得模型训练速度快、鲁棒性强、不会有训练过拟合等问题，是本作品丞待解决的问题。

## 解决问题的思路

### 作品功能及性能需求

1）图像处理功能。需要对脑肿瘤图像进行预处理，包括去除伪影、降噪、图像增强等处理，使得图像更加清晰，提高图像的质量以便提升脑肿瘤监测的正确性。

2）模型训练和测试功能。需要基于深度学习算法构建脑肿瘤监测模型，并进行训练和测试，得出模型的性能指标，如Dice、Sensitivity、PPV、Hausdorff等。

3）实时检测功能。需要对输入的脑肿瘤图像进行实时检测，输出患病概率或诊断结果，为医生提供参考。

4）用户界面。需要制作良好的用户界面可以使得医生操作更加简单直观，提高工作效率。需要提供图像上传、图像分割、图像分类等功能。

5）算法优化。对于基于深度学习的脑肿瘤监测算法，需要不断优化算法，提高模型的精度和鲁棒性，以提高其实际应用价值。

### 2.4.2数据集

为验证上述方法的有效性，本文选用目前公知的医学图像数据集进行实验，如图所示。实验使用BRATS2019数据集.其使用“.nii”格式存储，数据集包括训练集与验证集，其中训练集图像是由专业医师手动标注的脑肿瘤分割结果，包含背景（标签0）、坏死区域（标签1）、水肿区域（标签2）和增强肿瘤（标签4）四类。验证集的分割标签并不公开，为提高方法评估的公平性，研究者需要将验证集的分割图像结果上传至在线平台进行评估，进而由在线评估平台返回方法评测的指标数值结果。

在MRI脑肿瘤数据集规模上，BraTS2019训练集包括259例高级别胶质瘤（HighGradeGliomas，HGG）数据和76例低级别胶质瘤（LowGradeGliomas，LGG）数据。验证集则包含125个未知肿瘤样本。每个脑肿瘤数据样本分别包含大小为240×240×155的T1、T1ce、T2和Flair四个模态，如图2.9所示。此外，用于评估MRI脑肿瘤分割的三个区域分别是全部肿瘤（WholeTumor，WT）、肿瘤核心（TumorCore，TC）和增强肿瘤（EnhancingTumor，ET），其中（a）T1模态；（b）T1ce模态；（b）T2模态；（d）FLAIR模态；（e）专家标注的分割结果GroundTure，其中红色部分为坏死及非增强肿瘤（NCR/NET）区域，绿色部分为水肿（ED）区域，黄色部分为增强肿瘤（ET）区域。

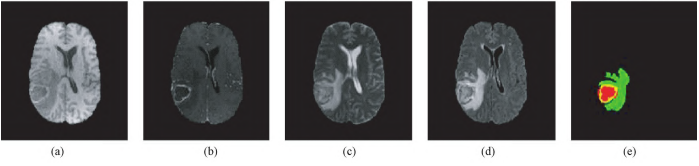


图 2.9 数据集

# 功能介绍

## 总体介绍

针对现有医学图像分割技术和传统医学图像风格存在的问题技术，本系统采用了改进的ResU-Net对脑肿瘤图像进行分割，使用了更深层的残差网络，提高模型的准确率。同时，本系统提供丰富的界面功能，包括文件加载、绘制图像、绘制3D模型。

## 系统工作流程

整个流程会涉及到算法模块、界面模块三个模块

1）录入病人信息：一张脑肿瘤MRI图像对应一位病人，一位病人对应多张图象。用户在主界面选择一张图像并输入，完成预测分割后数据库将信息保存在信息表中（Message Table）

2）图像处理：用户进入图像分析界面进行图像分析，图像分析界面会显示用户已录入的图片，用户可在该界面对图像进行分割或者诊断分析

3）信息保存：用户在图像分析界面点击保存按钮，选择保存图像的文件夹，之后，系统会更新对应表项的信息。

## 图像分割模块

### FCN

全卷积网络（Fully Convolutional Networks，FCN）是Jonathan Long**[15]**等人于2015年在Fully Convolutional Networks for Semantic Segmentation一文中提出的用于图像语义分割的一种框架，是深度学习用于语义分割领域的开山之作，如图3.1所示。FCN将传统CNN后面的全连接层换成了卷积层，这样网络的输出将是热力图而非类别；同时，为解决卷积和池化导致图像尺寸的变小，使用上采样方式对图像尺寸进行恢复，其核心思想是：

1）不含全连接层的全卷积网络，可适应任意尺寸输入；

2）反卷积层增大图像尺寸，输出精细结果；

3）结合不同深度层结果的跳级结构，确保鲁棒性和精确性。

对于FCN分割图像，需要进行以下步骤：

1）对图像进行预处理，包括缩放、裁剪和归一化等操作，以便输入到FCN模型中进行分割。

2）使用已经训练好的FCN模型对图像进行分割。FCN网络的输出是一个与输入图像相同大小的矩阵，其中每个像素都被标记为属于哪个类别。

3）对FCN模型输出的矩阵进行后处理，如去除边框、填充空洞等操作，以得到更精确的分割结果。

4）可以使用可视化工具将分割结果可视化，以便进行检查和评估。

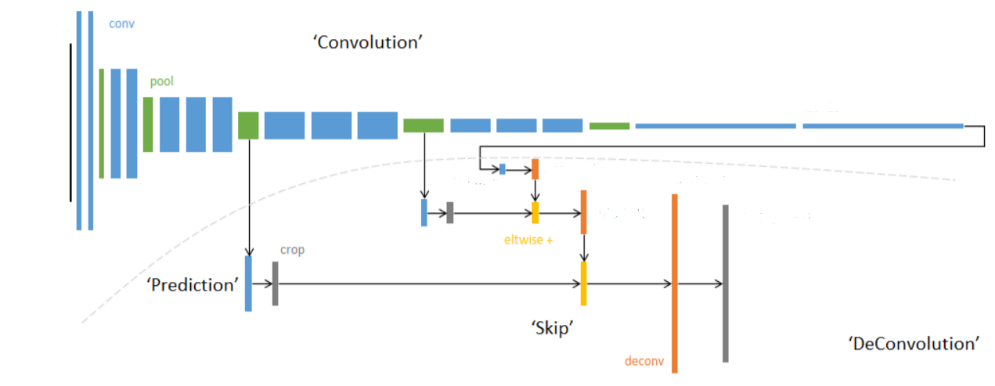


图 3.1 FCN网络结构

### U-net

UNet是一种用于图像分割的卷积神经网络，它由Ronneberger**[14]**等人在2015年提出，被广泛应用于医学图像的分割任务中。UNet的主要特点是它的U形结构，它包含一个编码器和一个解码器，其中编码器由一系列卷积和池化层组成，用于提取图像的特征，而解码器由一系列卷积和上采样层组成，用于将特征映射恢复到原始图像的尺寸和分辨率。UNet还使用了跳跃连接（skip connection）技术，将编码器中的某些特征图与解码器中对应的特征图进行连接，以保留更多的空间信息，提高分割的准确性。

UNet的训练过程通常使用交叉熵损失函数和随机梯度下降等优化算法进行优化。在训练过程中，可以使用数据增强技术来扩充训练集，以提高模型的泛化能力和鲁棒性。

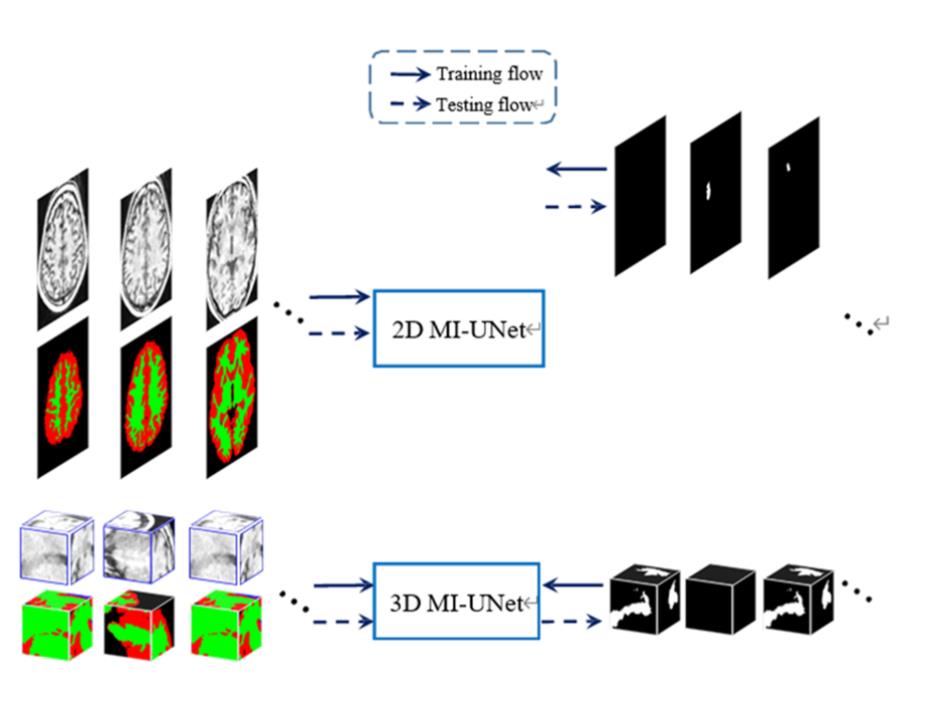


图 3.2 Unet分割

使用UNet进行图像分割时，需要将输入图像喂入网络，然后得到一个与输入图像相同大小的分割结果。分割结果通常使用标签或颜色进行可视化，以便进行检查和评估，如图3.2所示。

### ResUNet

ResUNet是一种基于ResNet和UNet的深度学习模型，用于图像分割任务。它的设计思想是将ResNet的深度残差结构和UNet的上采样和跳跃连接结构相结合，以实现更准确和更稳定的分割结果。图3.3显示了残差块结构，图3.4显示了ResUNet网络的整体结构。

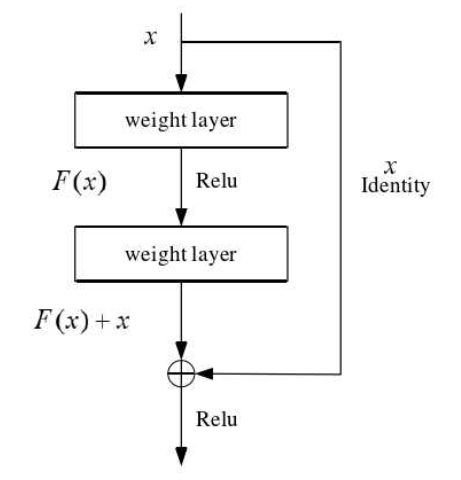


图 3.3 残差结构示意图

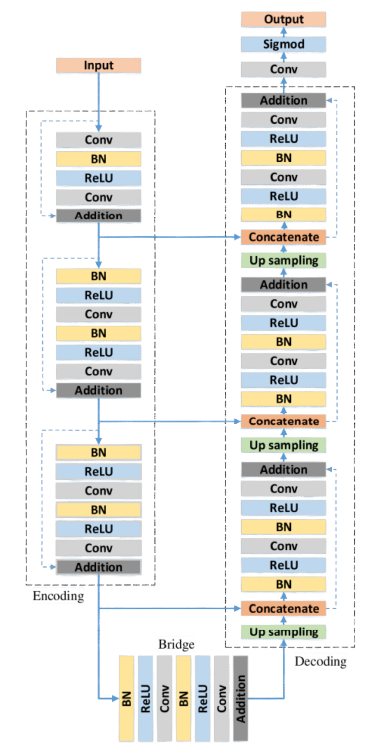


图 3.4 ResUnet

ResUNet的训练过程通常使用交叉熵损失函数和随机梯度下降等优化算法进行优化。在训练过程中，可以使用数据增强技术来扩充训练集，以提高模型的泛化能力和鲁棒性。

### Pro-ResUNet

Pro-ResUnet是改进版的ResUNet，采用了更多的卷积层，相对于ResUNet更深，更多的卷积层和残差块能提高网络的表达能力和空间信息保留能力，这使得网络在处理大规模的多维的图像分割时具有更好的性能。图3.5显示了Pro-ResUNet的网络结构。

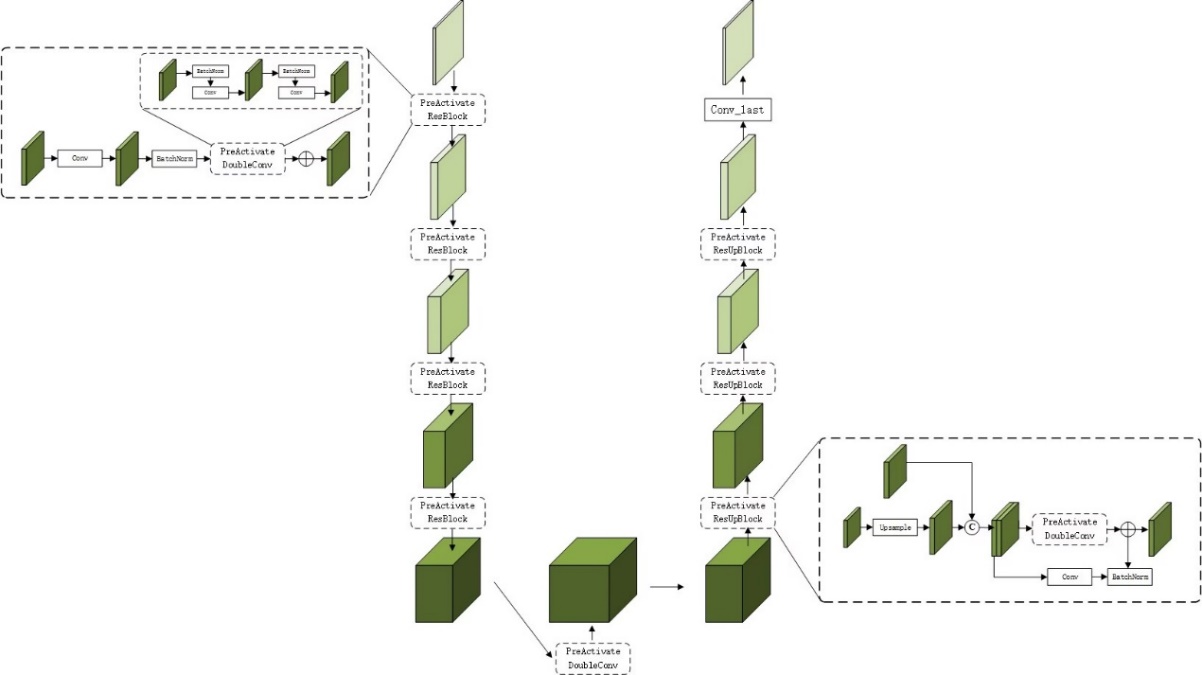


图 3.5 Pro-ResUNet

我们使用PyTorch框架进行Pro-ResUNet的实现，网络的核心组成部分是由预激活的残差块组成的编码器和解码器，以及连接它们的跳跃连接。编码器中包含四个预激活的残差块，预激活的卷积层是一种对传统卷积层的改进，它将激活函数移动到卷积操作之前，从而避免了在卷积操作后引入的信息丢失和梯度消失问题，并能加快收敛速度。每个残差块在传统的残差结构之前再引入一卷积层和BN层，用于逐步降采样特征图。解码器中包含四个预激活的残差块和四个上采样层，每个残差块也包含两个卷积层和一个跳跃连接，用于逐步恢复分辨率，在每个上采样层中，跳跃连接将来自相应编码器层的特征图与上采样后的特征图合并。

Pro-ResUNet在性能方面优于U-Net的原因主要包括以下几个方面：跳跃链接、残差块、下采样多加入卷积层、上采样基于双线性插值。

1）跳跃链接：Pro-ResUNet中的跳跃链接起到了连接编码器和解码器的作用，将编码器中层级较高的特征图与解码器中层级较低的特征图进行融合，从而使得解码器中的每一层都能够利用编码器中的所有层级信息，提高了网络的表达能力和分割性能。在U-Net中，虽然也有跳跃链接，但是只是简单地将编码器中的特征图与解码器中的特征图进行拼接，无法进一步利用编码器中的层级信息，因此在处理一些复杂的图像分割任务时可能表现不佳。

2）残差块：Pro-ResUNet中使用的是预激活的残差块，并且相比于传统ResUNet采用更多的残差块，有更好的特征表达能力和泛化能力。在残差块中，通过引入跳跃连接来避免梯度消失和梯度爆炸问题。此外，由于残差块中的跳跃连接可以直接将信息从输入传递到输出，因此可以更加有效地利用模型的容量，提高网络的分割性能。

3）多加入卷积层：Pro-ResUNet中在每个下采样残差块中均额外加入了一个卷积层，这一设计可以进一步增加网络的非线性表达能力，从而更好地适应复杂的分割任务。另外，多加入卷积层还有助于增加网络的深度和宽度，提高网络的表达能力和分割性能。

4）上采样基于双线性插值：在ResUNet中，上采样后的图像特征与对应的编码器层级特征进行拼接，而在Pro-ResUNet中则是将它们进行融合。这是因为在双线性插值的上采样方法中，拼接操作可能会引入一些不必要的噪声和信息冗余，所以Pro-ResUNet中采用了融合方法来避免这种情况。

## 界面功能模块

我们采用PyQt5框架构建了一个功能强大且易于使用的可视化用户界面。它为Python程序员提供了一组用于创建GUI应用程序的Python模块，并且可以跨平台运行于Unix、Windows和Mac OS等多个操作系统中。PyQt5包含了超过620个类和6000个函数及方法，可以实现诸如窗口、对话框、按钮、菜单、工具栏、表格等常见的GUI界面元素，并支持多种事件处理和信号槽机制，方便开发者进行应用程序的逻辑控制和交互操作

在本系统中，我们设计了两个主要部分，即主界面和图像分析。主界面用于引导用户进行功能操作。图像分析界面用于对脑肿瘤图像进行处理，用户可以对单张图像或所有图像进行处理，操作灵活方便。

使用我们设计的用户界面，医生可以更加快速地进行图像处理和诊断，提高了医疗工作的效率和准确性。同时，我们的系统还具有友好的用户界面和易于操作的功能，使得用户可以快速上手，并且更好地利用系统提供的功能。

# 技术方案

## 系统总体设计方案

本作品的设计分为2个部分——ui界面设计、算法设计。在ui界面设计部分，我们基于PyQt5制作了脑肿瘤分割系统的可视化界面；在图像处理算法设计部分，我们基于当前主流的PyTorch框架，分别使用U-Net、FCN以及自主改良的ResUNet网络进行图像分割处理。本作品的总体设计图如图4.1所示。

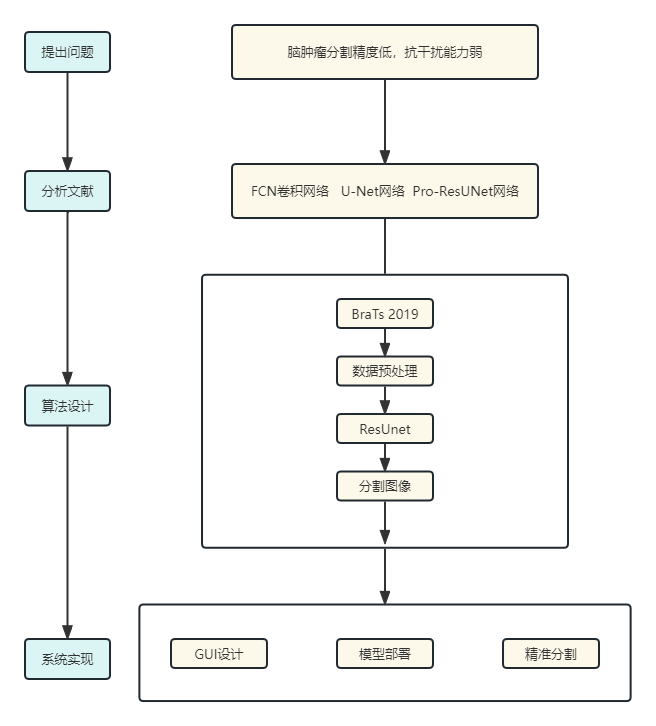


图4.1 系统总体设计图

## UI界面设计

我们采取PyQt5工具进行了界面设计，作品主要有四个部分：主界面模块、文件加载模块、绘制图像模块、绘制3D模型模块。

### 主界面模块

本模块我们主要设计的主要目的是进行功能引导，主界面集成了4个摁键，分别用以加载图像，加载模型预测的分割信息，绘制2D切片展示，绘制3D建模。



图4.2 主界面

### 文件加载模块

该板块包括加载图像和加载分割，主要功能是接受传入的图像根据模型自动预测生成分割信息文件，并且待传入生成的分割文件后对二者进行初步预处理。这部分有两个主要函数，并且这两个函数都使用了多进程的方式进行处理，以提高处理效率：

（1）merge\_nii\_gz(nii\_gz\_path, mask\_path, output\_nii\_path)用于合并两个nii.gz文件，并将合并后的结果保存为nii文件。具体实现中，使用了nib.load()函数加载了nii.gz文件，然后使用itk.image\_from\_array()函数将numpy数组转换成itk图像，最后使用itk.imwrite()函数将itk图像保存为nii文件。

（2）slices\_image(nii\_gz\_path, output\_folder)用于将nii.gz文件的每个切片保存为png图像。具体实现中，使用了nib.load()函数加载了nii.gz文件，并使用numpy.clip()函数将图像像素值剪辑到指定范围内，然后使用numpy.ndarray的数组操作将图像像素值归一化到[0,1]范围内，最后使用PIL.Image库将归一化后的图像保存为png格式的图像文件。这一步也是为了后续对3D图像进行2D层面上的切片展示做准备。

### 绘制图像模块

绘制图像板块包含了主界面上面的两个按键：颜色切换与绘制图像。绘制图像的功能是在对图像和分割信息两个3D文件进行切片后，从三个视角：横断面视角（Axial View），冠状面视角（Coronal View），矢状面视角（Sagittal View）。横断面视角是指沿着人体横向切割，得到的二维图像。在CT或MRI扫描中，横断面视角通常是最为常见的视角，因为它能够展示出人体内部结构的水平分布情况；冠状面视角是指沿着人体前后方向切割，得到的二维图像。可以展示出人体内部结构的前后分布情况，例如心脏或肺部结构；矢状面视角是指沿着人体左右方向切割，得到的二维图像。

在图像或者分割的每个子视图上面，通过滚轮滑动可以切换展示不同切片，通过鼠标对某个视图某个切分上的重点部位进行点击，还可以同步展示在另外两个视角中该部位的切片和位置，方便医生可以更好的观察肿瘤在脑部的状态。

点击颜色切换摁键后，系统展示出经过颜色特殊渲染的肿瘤分割图像，黄色为浮肿区域(ED,peritumoral edema) 、绿色为增强肿瘤区域(ET,enhancing tumor)、红色为坏疽(NET,non-enhancing tumor)。医生可以自主选择需要对肿瘤区域进行颜色加强对比显示，或者是期望更好观察其原始状态。

该部分有多个主要函数：

1）on\_bt\_plot\_image\_clicked函数首先使用nibabel库加载nii.gz文件，并创建两个OrthoSlicer3D对象，分别用于显示原始图像和分割图像。然后，使用slice\_image\_plus函数将nii.gz文件的每个切片保存为png图像。最后，使用MedicalImageVisualizer类将保存的图像在GUI中显示出来。

2）slice\_image\_plus函数用于将nii.gz文件的每个切片保存为png图像，并将这些图像保存到指定的文件夹中。具体实现中，该函数使用nibabel库加载nii.gz文件，并将其转换为numpy数组。然后，对数组进行剪切和归一化处理，将像素值转换为[0, 1]的范围。接下来，使用PIL库将每个切片保存为png图像，同时保存相应的标签图像。

3）on\_bt\_load\_img\_clicked和on\_bt\_load\_seg\_clicked函数用于在GUI中选择nii.gz文件和分割文件，分别将文件路径存储到self.slice\_nii\_path和self.seg\_nii\_path变量中。

4）on\_bt\_plot\_2d\_clicked函数使用OrthoSlicer3D类将nii.gz文件在2D平面上显示。该函数首先检查是否已经加载了nii.gz文件，然后调用OrthoSlicer3D类的plot\_2d方法，在GUI中显示2D图像。

5）on\_bt\_change\_color\_clicked函数首先检查当前使用的颜色映射（cmap）是否为'viridis'或'gray'，如果是'viridis'则改为'gray'，如果是'gray'则改为'viridis'。然后，如果已经存在分割图像（self.fig2），则关闭该图像，重新加载分割图像，并使用新的颜色映射（self.seg\_cmap）创建一个新的OrthoSlicer3D对象，并显示该对象。

图形用户界面, 图示, 应用程序

描述已自动生成

图4.3 分割无颜色2D切片展示

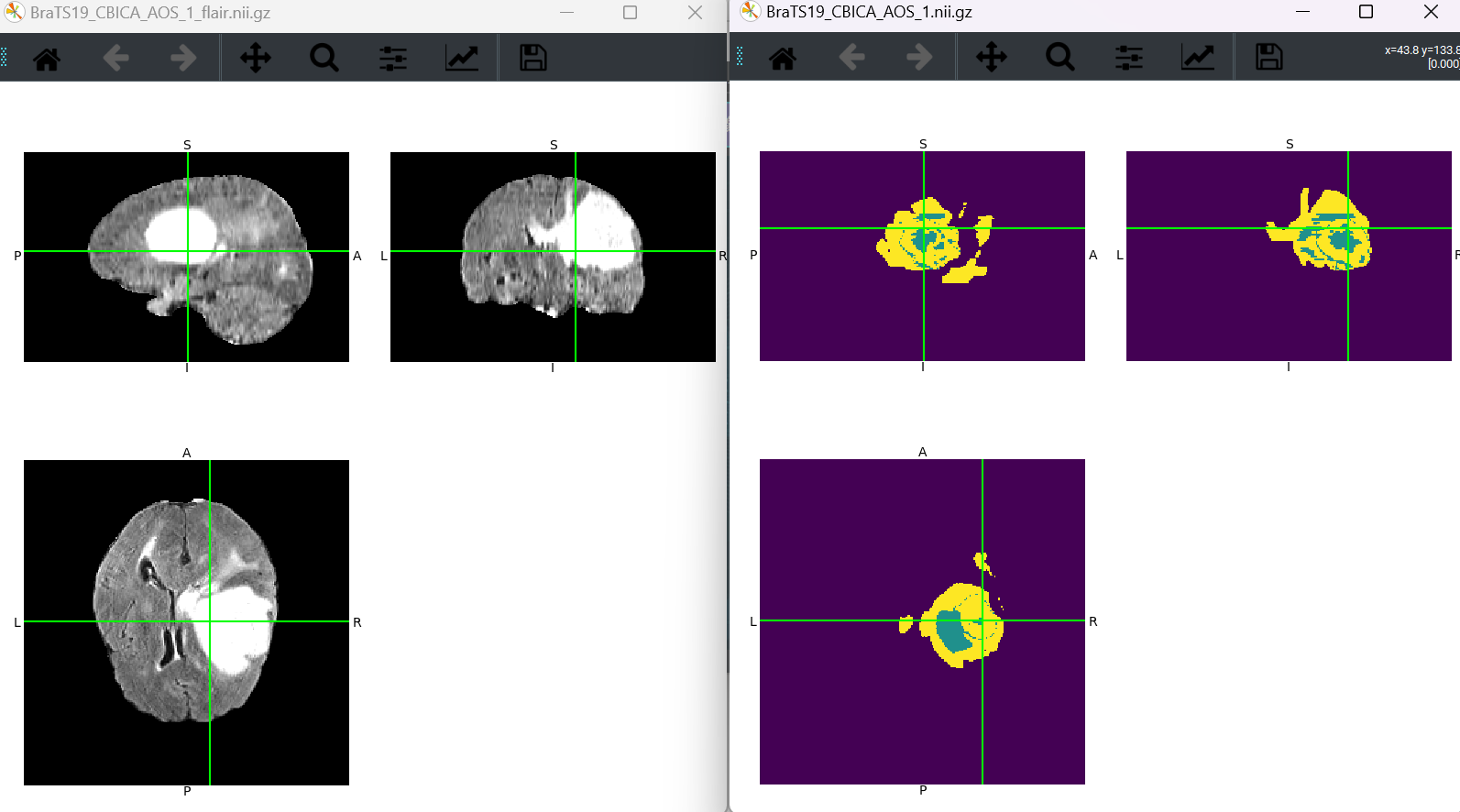


图4.4 分割有颜色2D切片展示

### 绘制3D模型模块

3D模块的主要功能是对大脑图像和分割出来的肿瘤进行3D建模并显示，并且根据鼠标可以对建模的3D图像进行旋转、放大缩小等操作，通过该功能医生可以更直观、更精准的了解到大脑内肿瘤的状态。

图形用户界面, 应用程序

描述已自动生成

图4.5 3D建模

## Pro-ResUNet图像分割模型训练

### 数据预处理

（1）步骤主要以下几步:

a）对各个模态进行标准化

b）对各模态及其GT数据进行裁剪

c）对各模态及其GT数据进行切片,并抛无病灶切片,最后合并各模态的切片,然后保存为Numpy

（2）标准化多模态

四个序列是不同模态的图像,因此图像对比度也不一样,所以采用z-score方式来对每个模态图像进行标准化,即将每个模态的数据标准化为零均值和单位标准差 ,但是GT文件是不需要进行标准化的.函数实现代码如下：

1. def normalize(slice, bottom=99, down=1):
2. """
3. normalize image with mean and std for regionnonzero,and clip the value into range
4. :param slice:
5. :param bottom:
6. :param down:
7. :return:
8. """
9. #有点像“去掉最低分去掉最高分”的意思,使得数据集更加“公平”
10. b = np.percentile(slice, bottom)
11. t = np.percentile(slice, down)
12. slice = np.clip(slice, t, b)#限定范围numpy.clip(a, a\_min, a\_max, out=None)
13. #除了黑色背景外的区域要进行标准化
14. image\_nonzero = slice[np.nonzero(slice)]
15. if np.std(slice) == 0 or np.std(image\_nonzero) == 0:
16. return slice
17. else:
18. tmp = (slice - np.mean(image\_nonzero)) / np.std(image\_nonzero)
19. # since the range of intensities is between 0 and 5000 ,
20. # the min in the normalized slice corresponds to 0 intensity in unnormalized slice
21. # the min is replaced with -9 just to keep track of 0 intensities
22. # so that we can discard those intensities afterwards when sampling random patches
23. tmp[tmp == tmp.min()] = -9 #黑色背景区域
24. return tmp

（3）裁剪

裁剪这个步骤根据自己的数据集进行选择,对于BraTs数据集是有必要的,下图是一个序列MR图像,其中灰色部分为脑部区域，黑色则为背景，背景信息在整幅图像中的比例较大，而且背景对于分割没有任何帮助。但是要对每一个像素进行分类，图像中肿瘤区域占比非常小，因此会出现严重的数据不平衡。为了提升模型分割的性能，首先对数据进行预处理操作。从医生角度来看这个ＭＲ图像，会自动过滤掉这个背景信息，把所有目光集中在脑部区域，因此去除脑部区域周围的背景信息是必要的。

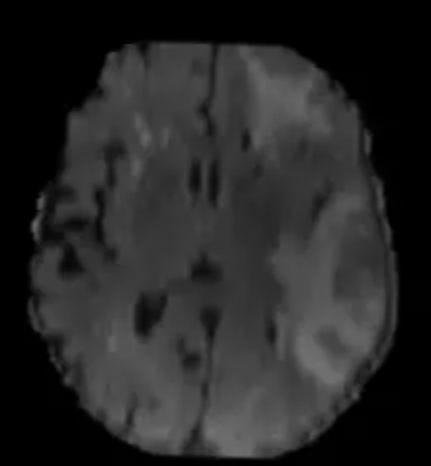


图 4.6 MR图像

1. def crop\_ceter(img,croph,cropw):
2. #for n\_slice in range(img.shape[0]):
3. height,width = img[0].shape
4. starth = height//2-(croph//2)
5. startw = width//2-(cropw//2)
6. return img[:,starth:starth+croph,startw:startw+cropw]

（4）切片、抛无病灶切片、合并各模态的切片、保存为Numpy

由于大多数医学图像都是三维数据,所以只有切成2D数据,才能适应2D网络,此外切片中不含有病灶部分的可以舍弃,同样也是为了缓解类别不均衡问题,又由于是多模态,因此要将各模态的切片组合成多通道,最后保存为Numpy, 而对于其相应的GT切片直接保存为Numpy。

1. #切片处理,并去掉没有病灶的切片,合并多模态组合多通道
2. for n\_slice in range(flair\_crop.shape[0]):
3. if np.max(mask\_crop[n\_slice,:,:]) != 0:
4. maskImg = mask\_crop[n\_slice,:,:]
6. FourModelImageArray = np.zeros((flair\_crop.shape[1],flair\_crop.shape[2],4),np.float)
7. flairImg = flair\_crop[n\_slice,:,:]
8. flairImg = flairImg.astype(np.float)
9. FourModelImageArray[:,:,0] = flairImg
10. t1Img = t1\_crop[n\_slice,:,:]
11. t1Img = t1Img.astype(np.float)
12. FourModelImageArray[:,:,1] = t1Img
13. t1ceImg = t1ce\_crop[n\_slice,:,:]
14. t1ceImg = t1ceImg.astype(np.float)
15. FourModelImageArray[:,:,2] = t1ceImg
16. t2Img = t2\_crop[n\_slice,:,:]
17. t2Img = t2Img.astype(np.float)
18. FourModelImageArray[:,:,3] = t2Img
19. imagepath = outputImg\_path + "\\" + str(pathlgg\_list[subsetindex]) + "\_" + str(n\_slice) + ".npy"
20. maskpath = outputMask\_path + "\\" + str(pathlgg\_list[subsetindex]) + "\_" + str(n\_slice) + ".npy"
21. np.save(imagepath,FourModelImageArray)#(160,160,4) np.float dtype('float64')
22. np.save(maskpath,maskImg) # (160, 160) dtype('uint8') 值为0 1 2 4

imagepath保存的是多模态切片合并后的数据,因为裁剪的大小是 160×160 ,而且是 4 个模态,因此保存后的 Numpy 的 shape 是(160,160,4) ,dtype(′float64′)

maskpath 保存的是其对应的GT切片的数据,并没有做任何处理,它的 shape 是 (160,160) , dtype(′uint8′) ,由于是多标签,其上面元素的值有0、1、2、4标签值。

### 构建模型

实现Pro-ResUNet网络，通过下采样和上采样操作实现了对图像的降采样和恢复，利用跳跃连接提高了特征图的稳定性和精度，最终实现了图像分割任务。

class Pro-ResUNet(nn.Module):  
 def \_\_init\_\_(self, args):  
 super(Pro-ResUNet, self).\_\_init\_\_()  
 self.down\_conv1 = PreActivateResBlock(4, 64)  
 self.down\_conv2 = PreActivateResBlock(64, 128)  
 self.down\_conv3 = PreActivateResBlock(128, 256)  
 self.down\_conv4 = PreActivateResBlock(256, 512)  
 self.double\_conv = PreActivateDoubleConv(512, 1024)  
 self.up\_conv4 = PreActivateResUpBlock(512 + 1024, 512)  
 self.up\_conv3 = PreActivateResUpBlock(256 + 512, 256)  
 self.up\_conv2 = PreActivateResUpBlock(128 + 256, 128)  
 self.up\_conv1 = PreActivateResUpBlock(128 + 64, 64)  
 self.conv\_last = nn.Conv2d(64, 3, kernel\_size=1)  
 def forward(self, x):  
 x, skip1\_out = self.down\_conv1(x)  
 x, skip2\_out = self.down\_conv2(x)  
 x, skip3\_out = self.down\_conv3(x)  
 x, skip4\_out = self.down\_conv4(x)  
 x = self.double\_conv(x)  
 x = self.up\_conv4(x, skip4\_out)  
 x = self.up\_conv3(x, skip3\_out)  
 x = self.up\_conv2(x, skip2\_out)  
 x = self.up\_conv1(x, skip1\_out)  
 x = self.conv\_last(x)  
 return x

### 训练过程

首先使用 parse\_args 函数解析命令行参数。根据命令行参数和数据集名称生成模型的名称，并在 models 文件夹中创建一个相应的文件夹来存储模型和日志文件。根据命令行参数的设置来定义损失函数 criterion，用于在训练过程中计算损失。使用 PyTorch 提供的 DataLoader 类加载训练数据集和验证数据集，使用 Dataset 类来定义数据集访问方式。mymodel.\_\_dict\_\_[args.arch](args) 创建一个名为 model 的模型实例，其中 args.arch 参数指定了模型的类型。最后，使用指定的优化器来优化模型，并在训练过程中对模型进行训练和验证，以更新模型参数，并记录训练和验证的损失和 IoU 值。在训练过程中，还实现了一个简单的早停策略，如果验证集的 IoU 值连续若干轮没有提高，则停止训练。

1. 使用 parse\_args 函数解析命令行参数
2. def parse\_args():
3. parser = argparse.ArgumentParser()
4. parser.add\_argument('--name', default=None,
5. help='model name: (default: arch+timestamp)')
6. parser.add\_argument('--arch', '-a', metavar='ARCH', default='Pro-ResUNet',
7. choices=arch\_names,
8. help='model architecture: ' +
9. ' | '.join(arch\_names) +
10. ' (default: Pro-ResUNet)')
11. parser.add\_argument('--deepsupervision', default=False, type=str2bool)
12. parser.add\_argument('--dataset', default="jiu0Monkey",
13. help='dataset name')
14. parser.add\_argument('--input-channels', default=4, type=int,
15. help='input channels')
16. parser.add\_argument('--image-ext', default='png',
17. help='image file extension')
18. parser.add\_argument('--mask-ext', default='png',
19. help='mask file extension')
20. parser.add\_argument('--aug', default=False, type=str2bool)
21. parser.add\_argument('--loss', default='BCEDiceLoss',
22. choices=loss\_names,
23. help='loss: ' +
24. ' | '.join(loss\_names) +
25. ' (default: BCEDiceLoss)')
26. parser.add\_argument('--epochs', default=10000, type=int, metavar='N',
27. help='number of total epochs to run')
28. parser.add\_argument('--early-stop', default=20, type=int,
29. metavar='N', help='early stopping (default: 20)')
30. parser.add\_argument('-b', '--batch-size', default=18, type=int,
31. metavar='N', help='mini-batch size (default: 16)')
32. parser.add\_argument('--optimizer', default='Adam',
33. choices=['Adam', 'SGD'],
34. help='loss: ' +
35. ' | '.join(['Adam', 'SGD']) +
36. ' (default: Adam)')
37. parser.add\_argument('--lr', '--learning-rate', default=3e-4, type=float,
38. metavar='LR', help='initial learning rate')
39. parser.add\_argument('--momentum', default=0.9, type=float,
40. help='momentum')
41. parser.add\_argument('--weight-decay', default=1e-4, type=float,
42. help='weight decay')
43. parser.add\_argument('--nesterov', default=False, type=str2bool,
44. help='nesterov')
45. args = parser.parse\_args()
46. return args
47. 构建数据集

构建数据集的实现，我们使用了Pytorch工具包中的Dataloader函数，它是一个用于加载数据的工具，用于在训练深度学习模型时对数据进行批量处理和并行加载。它可以自动进行数据的批量划分、多线程加载和数据顺序打乱，从而在训练过程中提高数据加载的效率。

1. train\_loader = torch.utils.data.DataLoader(
2. train\_dataset,
3. batch\_size=args.batch\_size,
4. shuffle=True,
5. pin\_memory=True,
6. drop\_last=True)
7. val\_loader = torch.utils.data.DataLoader(
8. val\_dataset,
9. batch\_size=args.batch\_size,
10. shuffle=False,
11. pin\_memory=True,
12. drop\_last=False)
13. 使用 Dataset 类来定义数据集的访问方式
14. class Dataset(torch.utils.data.Dataset):
15. def \_\_init\_\_(self, args, img\_paths, mask\_paths, aug=False):
16. self.args = args
17. self.img\_paths = img\_paths
18. self.mask\_paths = mask\_paths
19. self.aug = aug
20. def \_\_len\_\_(self):
21. return len(self.img\_paths)
22. def \_\_getitem\_\_(self, idx):
23. img\_path = self.img\_paths[idx]
24. mask\_path = self.mask\_paths[idx]
25. #读numpy数据(npy)的代码
26. npimage = np.load(img\_path)
27. npmask = np.load(mask\_path)
28. npimage = npimage.transpose((2, 0, 1))
29. WT\_Label = npmask.copy()
30. WT\_Label[npmask == 1] = 1.
31. WT\_Label[npmask == 2] = 1.
32. WT\_Label[npmask == 4] = 1.
33. TC\_Label = npmask.copy()
34. TC\_Label[npmask == 1] = 1.
35. TC\_Label[npmask == 2] = 0.
36. TC\_Label[npmask == 4] = 1.
37. ET\_Label = npmask.copy()
38. ET\_Label[npmask == 1] = 0.
39. ET\_Label[npmask == 2] = 0.
40. ET\_Label[npmask == 4] = 1.
41. nplabel = np.empty((160, 160, 3))
42. nplabel[:, :, 0] = WT\_Label
43. nplabel[:, :, 1] = TC\_Label
44. nplabel[:, :, 2] = ET\_Label
45. nplabel = nplabel.transpose((2, 0, 1))
46. nplabel = nplabel.astype("float32")
47. npimage = npimage.astype("float32")
48. return npimage,nplabel
49. 损失函数
50. class BCEDiceLoss(nn.Module):
51. def \_\_init\_\_(self):
52. super(BCEDiceLoss, self).\_\_init\_\_()
53. def forward(self, input, target):
54. bce = F.binary\_cross\_entropy\_with\_logits(input, target)
55. smooth = 1e-5
56. input = torch.sigmoid(input)
57. num = target.size(0)
58. input = input.view(num, -1)
59. target = target.view(num, -1)
60. intersection = (input \* target)
61. dice = (2. \* intersection.sum(1) + smooth) / (input.sum(1) + target.sum(1) + smooth)
62. dice = 1 - dice.sum() / num
63. return 0.5 \* bce + dice
64. 训练模型
65. def train(args, train\_loader, model, criterion, optimizer, epoch, scheduler=None):
66. # 创建 AverageMeter 对象记录损失和 IoU
67. losses = AverageMeter() # 平均损失
68. ious = AverageMeter() # 平均交并比
69. # 切换模型至训练模式
70. model.train()
71. # 遍历训练集数据加载器
72. for i, (input, target) in tqdm(enumerate(train\_loader), total=len(train\_loader)):
73. # 将输入和目标移至 GPU
74. input = input.cuda()
75. target = target.cuda()
76. # 计算输出和损失
77. if args.deepsupervision:
78. # 如果使用深度监督，为每个输出计算损失并取平均
79. outputs = model(input)
80. loss = 0
81. for output in outputs:
82. loss += criterion(output, target)
83. loss /= len(outputs)
84. iou = iou\_score(outputs[-1], target)
85. else:
86. # 如果未使用深度监督，计算输出的损失和交并比
87. output = model(input)
88. loss = criterion(output, target)
89. iou = iou\_score(output, target)
90. # 更新损失和交并比记录器
91. losses.update(loss.item(), input.size(0))
92. ious.update(iou, input.size(0))
93. # 计算梯度并执行优化步骤
94. optimizer.zero\_grad()
95. loss.backward()
96. optimizer.step()
97. # 构建一个 OrderedDict，记录损失和交并比的平均值
98. log = OrderedDict([
99. ('loss', losses.avg),
100. ('iou', ious.avg),
101. ])
102. return log

每一次epoch训练结束,都会对验证集进行测试Iou指标,如果比之前最好的还要好就保存本次训练模型,如果超过args.early\_stop这个参数还没有训练更好的话,便结束训练,这个原理就是early\_stop,主要还是防止网络训练过度,造成过拟合现象,这也就是验证集虽然没有直接参与训练,但是却在其中扮演了一个非常重要的角色。

### 计算评价指标

1）Dice Similarity Coefficient（戴斯相似性系数）

Dice系数是一种集合相似度度量指标,通常用于计算两个样本的相似度,值的范围 0−1 ,分割结果最好时值为 1 ,最差时值为 0。

1. def dice\_coef(output, target):#output为预测结果 target为真实结果
2. smooth = 1e-5 #防止0除
3. if torch.is\_tensor(output):
4. output = torch.sigmoid(output).data.cpu().numpy()
5. if torch.is\_tensor(target):
6. target = target.data.cpu().numpy()
7. intersection = (output \* target).sum()
8. return (2. \* intersection + smooth) / \
9. (output.sum() + target.sum() + smooth)

2）IOU

1. def iou\_score(output, target):
2. smooth = 1e-5
3. if torch.is\_tensor(output):
4. output = torch.sigmoid(output).data.cpu().numpy()
5. if torch.is\_tensor(target):
6. target = target.data.cpu().numpy()
7. output\_ = output > 0.5
8. target\_ = target > 0.5
9. intersection = (output\_ & target\_).sum()
10. union = (output\_ | target\_).sum()
11. return (intersection + smooth) / (union + smooth)

3）Sensitivity

1. def sensitivity(output, target):
2. smooth = 1e-5
3. if torch.is\_tensor(output):
4. output = torch.sigmoid(output).data.cpu().numpy()
5. if torch.is\_tensor(target):
6. target = target.data.cpu().numpy()
7. intersection = (output \* target).sum()
8. return (intersection + smooth) / \
9. (target.sum() + smooth)

4）PPV

1. def ppv(output, target):
2. smooth = 1e-5
3. if torch.is\_tensor(output):
4. output = torch.sigmoid(output).data.cpu().numpy()
5. if torch.is\_tensor(target):
6. target = target.data.cpu().numpy()
7. intersection = (output \* target).sum()
8. return (intersection + smooth) / \
9. (output.sum() + smooth)
10. Hausdorff\_95 (95% HD)

Dice对mask的内部填充比较敏感，而hausdorff distance 对分割出的边界比较敏感。Hausdorff\_95就是是最后的值乘以95%,目的是为了消除离群值的一个非常小的子集的影响。

1. import numpy as np
2. from hausdorff import hausdorff\_distance
3. # two random 2D arrays (second dimension must match)
4. np.random.seed(0)
5. X = np.random.random((1000,100))
6. Y = np.random.random((5000,100))
7. # Test computation of Hausdorff distance with different base distances
8. print("Hausdorff distance test: {0}".format( hausdorff\_distance(X, Y, distance="manhattan") ))
9. print("Hausdorff distance test: {0}".format( hausdorff\_distance(X, Y, distance="euclidean") ))
10. print("Hausdorff distance test: {0}".format( hausdorff\_distance(X, Y, distance="chebyshev") ))
11. print("Hausdorff distance test: {0}".format( hausdorff\_distance(X, Y, distance="cosine") ))
12. # For haversine, use 2D lat, lng coordinates
13. def rand\_lat\_lng(N):
14. lats = np.random.uniform(-90, 90, N)
15. lngs = np.random.uniform(-180, 180, N)
16. return np.stack([lats, lngs], axis=-1)
17. X = rand\_lat\_lng(100)
18. Y = rand\_lat\_lng(250)
19. print("Hausdorff haversine test: {0}".format( hausdorff\_distance(X, Y, distance="haversine") ))

### 技术方法

1）Weight decay

Weight decay是一种常用的正则化方法，用于减少模型的过拟合现象。在神经网络中，模型的参数通常通过反向传播算法进行学习，而weight decay是通过在损失函数中添加正则项来实现的。具体而言，weight decay会对模型的权重施加一个惩罚，使得较大的权重会被抑制，从而降低模型的复杂度。

在实际应用中，weight decay通常被添加到损失函数的目标项之后，形式为：

其中，表示数据项的损失函数， 表示模型的第i个权重，表示L2范数，表示weight decay的超参数，用于控制正则化的程度。通过调整的值，可以控制weight decay的影响程度，从而达到更好的正则化效果。

需要注意的是，weight decay对于不同的模型和任务可能会产生不同的效果，因此需要根据实际情况进行调整。此外，也可以结合其他正则化方法一起使用，以进一步提高模型的泛化能力。

weight decay的超参数是，它可以被调整以控制weight decay的影响程度。通常，较小的会导致较弱的正则化效果，而较大的则会导致更强的正则化效果。因此，在实际应用中，需要根据模型的复杂度和数据集的大小等因素来选择合适的值，以达到最佳的正则化效果。当设置过大时，weight decay可能会抑制模型的学习能力，导致模型的性能下降。另外，weight decay的效果也可能会受到其他超参数的影响，例如学习率等。因此，在进行超参数调整时，需要综合考虑多个超参数之间的相互影响，以找到最佳的超参数组合。

2）Momentum

Momentum是一种优化算法，用于在训练神经网络时更新模型的参数。与标准的梯度下降算法不同，Momentum算法引入了一个动量项来加速参数的更新。具体而言，Momentum算法在每次参数更新时，会使用上一次参数更新的方向和大小来调整本次更新的方向和大小。这样做的效果是，如果在连续的更新中梯度方向保持不变，那么Momentum算法将加速参数的更新，从而更快地收敛到最优解。

Momentum算法的更新公式如下：

其中，表示在时间步t时的动量向量，γ是动量的超参数，η是学习率，表示损失函数J对参数θ的梯度。Momentum算法在更新参数时，不仅使用当前的梯度信息，还使用了上一次的动量信息，从而使得参数更新方向更加平滑，避免了梯度变化较大时的震荡问题。Momentum算法可以加快参数更新的速度，从而更快地收敛到最优解。但是，在实际应用中，Momentum的超参数γ需要根据具体的任务和模型进行调整，以达到最佳的性能。一般来说，较小的γ会导致更平缓的更新轨迹，而较大的γ则会导致参数更新的速度变快。通常γ的取值范围为0到1之间，常见的取值为0.9。

3） BCEDiceLoss

BCEDiceLoss是一种常用的用于图像分割任务的损失函数，它是二分类交叉熵损失函数和Dice系数损失函数的线性组合。BCEDiceLoss既能够惩罚模型对类别的错误分类，也能够惩罚模型分割结果与真实标签之间的不匹配。因此，它在应对类别不平衡和边界模糊等问题时具有一定的优势。

BCEDiceLoss的计算方式如下：

α是二分类交叉熵损失函数的权重系数，表示二分类交叉熵损失函数，表示Dice系数损失函数。Dice系数是一种用于评估分割模型性能的指标，它可以测量模型分割结果和真实标签之间的相似度。Dice系数的计算方式为：

TP表示分类模型正确预测的正类样本数，FP表示分类模型错误预测的负类样本数，FN表示分类模型错误预测的正类样本数。Dice系数的取值范围在0到1之间，越接近1表示模型分割结果和真实标签越相似，性能越好。

BCEDiceLoss的优点在于能够综合利用二分类交叉熵损失函数和Dice系数损失函数的优点，从而得到更全面和准确的模型评估结果。需要注意的是，BCEDiceLoss的超参数α通常需要根据实际情况进行调整，以达到最佳的性能。

# 测试分析

## 图像分割分割模型训练评价指标

Dice系数、PPV、Sensitivity和Hausdorff距离都是常用的用于评估分类或分割模型性能的指标，它们在实际应用中都有不同的作用和侧重方向。Dice系数侧重于分割结果和真实标签之间的重叠程度，它对分割结果和真实标签之间的重叠部分敏感。PPV也称为准确率，可以测量分类模型对正类的预测准确率。Sensitivity用以测量分类模型对正类的识别能力。Hausdorff95计算模型分割结果和真实标签之间的最大距离，可以衡量模型进行分割任务时对边缘的敏感度。因此，多个指标的综合使用可以更全面地评估模型的性能。

### Dice系数

Dice系数是一种常用的评估分割模型性能的指标，它衡量模型分割结果和真实标签之间的重叠程度。具体而言，Dice系数定义为分割结果和真实标签之间重叠像素数的两倍除以它们的总像素数之和，即：

其中X和 Y分别表示模型分割结果和真实标签的像素点集，|X|和|Y|分别表示它们的像素数。分子表示分割结果和真实标签之间的重叠像素数，分母|X|+|Y|表示分割结果和真实标签之间的总像素数。其中X = FN + TP，Y = TP + FP，上面公式可以转换为：

Dice系数的取值范围在0到1之间，越接近1表示模型分割结果和真实标签的重叠部分越多，性能越好。当分割结果和真实标签完全重合时，Dice系数为1；当它们完全不重叠时，Dice系数为0。Dice系数的优点在于它对分割结果和真实标签之间的重叠部分敏感，能够更好地衡量模型的性能。但是，它也有一些缺点，比如它不考虑分割结果和真实标签之间的形状和位置差异，可能导致评估结果不够准确。

### PPV(Positive predictive value)

PPV（Positive predictive value）也称为准确率（precision），它可以测量模型对正类的预测准确率。具体而言，PPV定义为真阳性（TP）的比例，即：

其中，TP表示模型正确预测的正类样本数，FP表示模型错误预测的负类样本数。PPV的取值范围在0到1之间，越接近1表示模型的正类预测准确率越高，性能越好。当模型对所有样本都预测为正类时，PPV为1；当模型对所有样本都预测为负类时，PPV为0。PPV对于那些需要确保结果准确性的任务非常重要，例如医学诊断、金融风险评估等。在这些任务中，错误的预测可能会带来严重的后果。

### Sensitivity

Sensitivity（灵敏度）也称为召回率（recall），它可以测量模型对正类的识别能力。具体而言，Sensitivity定义为真阳性（TP）的比例，即：

其中，TP表示模型正确预测的正类样本数，FN表示模型错误预测的负类样本数。Sensitivity的取值范围在0到1之间，越接近1表示模型对正类的识别能力越强，性能越好。当模型对所有正类样本都预测正确时，Sensitivity为1；当模型对所有正类样本都预测错误时，Sensitivity为0。Sensitivity可以评估出模型分割结果的覆盖率。

### Hausdorff\_95

Hausdorff距离是一种常用的用于评估分割模型性能的指标，它可以测量模型分割结果和真实标签之间的最大距离。Hausdorff距离的计算方式是对于每个像素点，分别计算它到另一个像素点集合中所有点的距离，然后取最大值，即：

dxy和dyx分别称为从X集合到Y集合和从Y集合到X集合的单向Hausdorff距离。即实际上首先对点集X中的每个点ai到距离此点ai最近的Y集合中点bj之间的距离‖ai-bj‖进行排序，然后取该距离中的最大值作为dxy的值；dyx同理可得。Hausdorff距离是两者中的较大者，它度量了两个点集间的最大不匹配程度。hausdorff distance对分割出的边界比较敏感。Hausdorff\_95就是是最后的值乘以95%,目的是为了消除一个非常小的离群值子集(即异常值子集)的影响。

## 模型训练参数设置

表5-1 模型训练参数设置

|  |  |  |
| --- | --- | --- |
| 参数 | 修改值 | 说明 |
| Epochs | 10000 | 模型迭代次数 |
| Batchsize | 18 | 批量大小 |
| Learning rate | 0.0003 | 学习率 |
| Weight\_decay | 0.0001 | 正则化用于减少模型的过拟合 |
| momentum | 0.9 | 优化算法引入动量项加速参数更新 |
| Early\_stop | 20 | 防止模型过拟合 |

## 图像分割训练结果对比

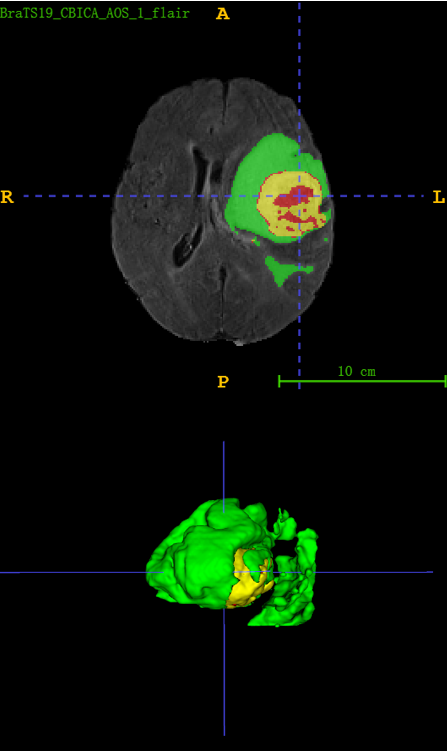
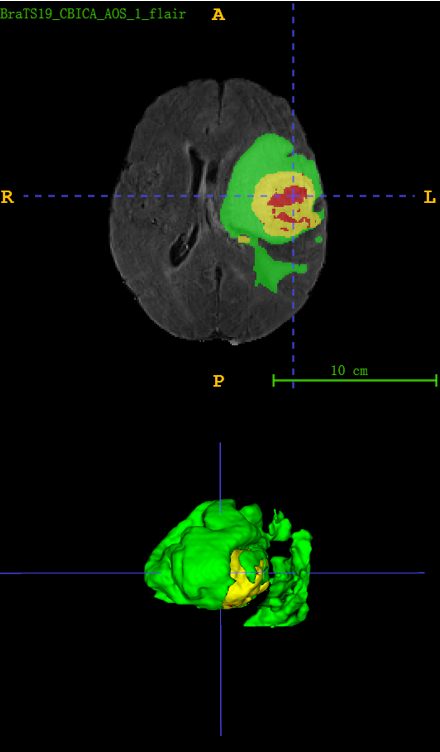


图5-1 预测分割结果对比（左原始数据集，右pro-resunet）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Dice | | | | PPV | | | |
| WT | TC | ET | avg | WT | TC | ET | avg |
| UNet | 0.8868 | 0.8203 | 0.8628 | 0.8566 | 0.8666 | 0.7782 | 0.8633 | 0.8360 |
| FCN8s | 0.8447 | 0.7155 | 0.5870 | 0.7157 | 0.8180 | 0.7264 | 0.7018 | 0.7487 |
| Pro-ResUNet | 0.9017 | 0.8719 | 0.8589 | 0.8775 | 0.9120 | 0.8744 | 0.9085 | 0.8983 |
|  | Sensitivity | | | | Hausdorff\_95 | | | |
| WT | TC | ET | avg | WT | TC | ET | avg |
| UNet | 0.8985 | 0.8491 | 0.8372 | 0.8616 | 0.5076 | 0.6843 | 0.3191 | 0.5037 |
| FCN8s | 0.8934 | 0.7890 | 0.5905 | 0.7576 | 0.5103 | 0.7739 | 1.0777 | 0.7873 |
| Pro-ResUNet | 0.9202 | 0.9141 | 0.8848 | 0.9063 | 0.2286 | 0.3845 | 0.2719 | 0.2950 |

表5-2 训练结果对比（总体）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Dice | PPV | Sensitivity | Hausdorff\_95 |
| UNet | 0.8566 | 0.8360 | 0.8616 | 0.5037 |
| FCN8s | 0.7157 | 0.7487 | 0.7576 | 0.7873 |
| Pro-ResUNet | 0.8775 | 0.8983 | 0.9063 | 0.2950 |

表5-3-1 训练结果对比（avg）

表5-3-2 训练结果对比（avg）

## 图像分割训练结果分析

经过改进的Pro-ResUNet性能方面有了很大的提升，在分割正确率、分割真正样本覆盖程度、对正类的识别能力以及对分割图像边缘敏感度方面均优于UNet与FCN8s。Pro-ResUNet中的跳跃链接起到了连接编码器和解码器的作用，将编码器中层级较高的特征图与解码器中层级较低的特征图进行融合，从而使得解码器中的每一层都能够利用编码器中的所有层级信息; 残差块使模型具有更快的收敛速度和更好的表达能力；Pro-ResUNet中在每个残差块中均加入了两个卷积层，这一设计可以进一步增加网络的非线性表达能力，更好地适应复杂的分割任务。提高了网络的表达能力和分割性能。Pro-ResUNet通过这些设计，使得网络具有更好的表达能力、更强的非线性拟合能力和更好的泛化能力，从而在图像分割任务中表现更好。

# 作品总结

## 作品特色与创新点

### 数据预处理

1）数据增强：在数据预处理中使用数据增强的方法对图像进行旋转、缩放、镜像等变换，扩充数据集，可以提高模型的泛化能力，有效避免模型对于局部特征的过拟合。

2）数据清洗：在数据预处理中使用数据清洗的方法对图像进行去噪、去除异常值等操作，提高数据的质量和可信度，减少模型学习到的噪声和误差。

3）多模态融合：将多个 MRI 影像拼接成多通道的数据，并去除无病灶切片以缓解类别不均衡问题,类别不平衡(class-imbalance)就是指分类任务中不同类别的训练样例数目差别很大的情况,但若差别很大,则会对学习过程造成困扰.我们的任务是分割,分割是一种对像素级别的分类,一个切片假如病灶很少甚至没有,那么就会出现严重的类别不均衡,学习的时候网络就会偏向于多的那一类。将有病灶的切片保存成 numpy 格式的文件，充分利用不同 MRI 模态提供的不同信息，提高模型的分类性能。

4）中心裁剪：对 CT 影像进行中心裁剪，保留了 CT 影像中心区域的信息，从而提高了模型对重要区域的识别能力，同时也减小了噪声和冗余信息对模型的干扰。

### 构建模型

1）基于 ResNet 和 U-Net 的结构：Pro-ResUNet 模型基于 ResNet 和 U-Net 的结构，采用预激活残差块和预激活双卷积层来构建网络，从而在保证模型深度的同时，提高了模型的准确性和稳定性。

2）多尺度特征融合：Pro-ResUNet 模型采用了多层级特征融合的方法，将不同层次的特征信息进行融合，从而提高了模型对不同尺度特征的提取能力。

3）逐级上采样：Pro-ResUNet 模型采用逐级上采样的方法，通过反卷积操作将特征图的分辨率逐步恢复到原始大小，从而提高了模型对细节信息的提取能力。

4）输出层：Pro-ResUNet 模型的输出层采用了 1x1 的卷积核，将特征图压缩成 3 个通道，分别对应肿瘤区域、背景和非肿瘤区域，从而实现了对肿瘤区域的分割。

## 作品展望

1）继续提高分割分类模型的性能，使其能应用到更多的场景。可以进一步探索更加先进的深度学习模型，以提高分割和分类准确率。

2）扩充建议的内容。并能考虑更多的因素，为病人提供更综合的建议。可以考虑采用更加智能化的算法来扩充建议的内容。

3）优化美化UI界面。在提高模型性能的同时，也要注重界面设计的美观性和用户体验，以吸引更多的用户使用。此外，需要考虑数据库的安全性和可扩展性，以确保数据的安全和高效管理。

4）发展网页版本。针对无法安装应用的用户，可以开发网页版的医学图像分析工具。通过网页版，用户可以在任何设备上方便地使用该工具，为更多的用户提供服务。

5）加强数据采集和预处理的质量控制。医学图像的质量对模型训练和诊断结果都有很大的影响。因此，可以采用更加规范化的数据采集流程和标注标准，并使用高质量的图像增强技术来增强数据的多样性和可靠性。

6）加强模型的可解释性和可视化能力。深度学习模型的黑盒特性常常限制了其在临床应用中的可信度和可接受性。因此，可以使用一些可解释性算法，来帮助解释模型的判断过程和结果，并为临床医生提供更加直观的可视化信息。

# 参考文献

[1]Yong C,Xuesong L,Qinlan X. Collaborative networks of transformers and convolutional neural networks are powerful and versatile learners for accurate 3D medical image segmentation[J]. Computers in Biology and Medicine,2023,164.

[2]Shangqi G,Hangqi Z,Yibo G, et al. BayeSeg: Bayesian modeling for medical image segmentation with interpretable generalizability[J]. Medical Image Analysis,2023,89.

[3]Zhenxi Z,Chunna T,Xinbo G, et al. Model-driven self-aware self-training framework for label noise-tolerant medical image segmentation[J]. Signal Processing,2023,212.

[4]Haiyang Z,Guanglei W,Yanlin W, et al. VCMix-Net: A hybrid network for medical image segmentation[J]. Biomedical Signal Processing and Control,2023,86(PB).

[5]Yanwen C,Ningdi X,Xin L, et al. P-TransUNet: an improved parallel network for medical image segmentation.[J]. BMC bioinformatics,2023,24(1).

[6]Zhaojin F,Jinjiang L,Zhen H, et al. Deep supervision feature refinement attention network for medical image segmentation[J]. Engineering Applications of Artificial Intelligence,2023,125.

[7]Shahed A,Kamrul M H. COMA-Net: Towards generalized medical image segmentation using complementary attention guided bipolar refinement modules[J]. Biomedical Signal Processing and Control,2023,86(PB).

[8]HongYu Z,Jiansen G,Yinghao Z, et al. nnFormer: Volumetric Medical Image Segmentation via a 3D Transformer.[J]. IEEE transactions on image processing : a publication of the IEEE Signal Processing Society,2023,PP.

[9]Himashi P,Munawar H,Zhaolin C, et al. Uncertainty-guided dual-views for semi-supervised volumetric medical image segmentation[J]. Nature Machine Intelligence,2023,5(7).

[10]Jun L,Bin S,Chengyan W, et al. Adaptive feature aggregation based multi-task learning for uncertainty-guided semi-supervised medical image segmentation[J]. Expert Systems With Applications,2023,232.

[11]Jun L,Nan C,Han Z, et al. MCRformer: Morphological constraint reticular transformer for 3D medical image segmentation[J]. Expert Systems With Applications,2023,232.

[12]Weiping D,Haipeng W,Jiashuang H, et al. FTransCNN: Fusing Transformer and a CNN based on fuzzy logic for uncertain medical image segmentation[J]. Information Fusion,2023,99.

[13]Suigu T,Fong C C,Xiaoyuan Y, et al. TransCS-Net: A hybrid transformer-based privacy-protecting network using compressed sensing for medical image segmentation[J]. Biomedical Signal Processing and Control,2023,86(PA).

[14]Xiuxian Z,Lianghui X,Chaoqun L, et al. RFE-UNet: Remote Feature Exploration with Local Learning for Medical Image Segmentation.[J]. Sensors (Basel, Switzerland),2023,23(13).

[15]Aidan B,Zezhong Y,Sanjay P, et al. Expert-level pediatric brain tumor segmentation in a limited data scenario with stepwise transfer learning.[J]. medRxiv : the preprint server for health sciences,2023.