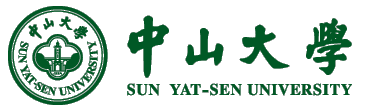
****

**主 题：基于深度学习的脑肿瘤**

**MRI 图像分类研究**

**科 目： 机器学习**

**摘 要**

本研究基于Kaggle公开脑肿瘤MRI数据集（253张图像），系统比较了SimpleCNN、ResNet18与EfficientNet-B0三种深度学习模型在肿瘤二分类任务中的表现。通过颅骨剥离、数据增强与迁移学习等标准化流程，实验结果显示EfficientNet-B0在测试集上取得最优性能：准确率达94.9%，召回率高达100%（零漏诊），且Grad-CAM可视化表明其决策聚焦于真实病灶区域。然而，训练过程分析揭示了小样本带来的根本挑战——模型性能对数据划分高度敏感，ResNet18出现明显震荡，SimpleCNN则快速过拟合。本研究不仅验证了高效网络在医学影像中的潜力，也警示了深度学习在数据受限场景下的脆弱性，为未来研究指明了扩大数据规模、引入3D与多模态信息等方向。

**关键词：**脑肿瘤检测；MRI；深度学习；EfficientNet；可解释人工智能

**ABSTRACT**

This study systematically compares SimpleCNN, ResNet18, and EfficientNet-B0 on a publicly available brain tumor MRI dataset (253 images) for binary classification. Using standardized preprocessing—including skull-stripping, data augmentation, and transfer learning—EfficientNet-B0 achieved the best test performance: 94.9% accuracy and perfect 100% recall (zero false negatives), with Grad-CAM visualizations confirming its focus on actual tumor regions. However, training dynamics expose a fundamental challenge: performance is highly sensitive to data splits, with ResNet18 showing instability and SimpleCNN overfitting rapidly. While validating the potential of efficient architectures in medical imaging, this work also highlights the fragility of deep learning under data scarcity and points toward future directions such as larger datasets, 3D modeling, and multi-modal fusion.

**KEYWORDS：**Brain tumor detection; MRI; Deep learning; EfficientNet; Explainable AI

**目 录**

**一、引言 1**

（一）研究背景 1

（二）研究目标 1

（三）研究方法 2

**二、数据描述性统计 2**

**三、数据预处理 4**

（一）颅骨剥离 4

（二）数据增强 5

（三）尺寸归一化 6

（四）数据划分 6

**四、模型架构 6**

（一）卷积网络（SimpleCNN） 6

（二）残差网络（ResNet18） 7

（三）高效网络（EfficientNet-B0） 8

（四）模型参数对比 8

**五、模型训练与性能评估 8**

（一）训练策略 8

（二）评估指标 9

**六、实验结果 10**

（一）模型训练过程分析 10

（二）分类性能对比 12

（三）ROC曲线分析 13

（四）混淆矩阵解读 15

（五）可解释性分析（Grad-CAM） 16

**七、结论、不足与展望 18**

（一）模型性能差异的简要解释 18

（二）小数据集下的根本性挑战 19

（三）临床应用潜力与现实局限 19

**附录 21**

**一、引言**

**（一）研究背景**

脑肿瘤是中枢神经系统中最常见且最具危害性的疾病之一，其发病率虽低于其他实体瘤，但致死率和致残率极高，严重威胁人类生命健康与生活质量。根据世界卫生组织（WHO）的统计数据，原发性脑肿瘤患者的五年生存率普遍较低，尤其在高级别胶质瘤（如胶质母细胞瘤）患者中，中位生存期往往不足15个月。早期、准确的诊断是改善预后、制定个性化治疗方案的关键前提。然而，脑肿瘤在临床表现上具有高度异质性，症状常与其他神经系统疾病重叠，导致误诊或延迟诊断的风险显著增加。

医学影像技术的发展为脑肿瘤的无创检测提供了强有力工具。其中，磁共振成像（Magnetic Resonance Imaging, MRI）凭借其卓越的软组织对比度、多参数成像能力以及无电离辐射等优势，已成为临床脑肿瘤筛查、定位、定性及疗效评估的“金标准”。典型的脑肿瘤 MRI 图像通常表现为局部信号异常区域，如T1加权像低信号、T2加权像高信号，并可能伴随占位效应、水肿或强化征象。然而，MRI图像的解读高度依赖放射科医师的经验与主观判断，不仅耗时费力，而且在面对微小病灶、非典型表现或图像伪影时，极易出现漏诊或误诊。此外，全球范围内放射科医师资源分布不均，基层医疗机构往往面临专业人才短缺的困境，进一步加剧了脑肿瘤早期诊断的挑战。

在此背景下，人工智能（Artificial Intelligence, AI），特别是深度学习（Deep Learning）技术的迅猛发展，为医学影像分析带来了革命性机遇。卷积神经网络（Convolutional Neural Networks, CNNs）因其强大的局部特征提取与空间层次建模能力，已在多种医学图像分析任务中展现出媲美甚至超越人类专家的性能。通过端到端的学习方式，深度学习模型能够自动从原始像素中挖掘与疾病相关的深层语义特征，实现高精度、高效率的自动化诊断，有效缓解医疗资源压力，提升诊疗一致性。

**（二）研究目标**

本研究聚焦于利用深度学习技术对脑部MRI图像进行二分类任务——即判断图像中是否存在肿瘤。这一任务虽看似简单，却是构建更复杂肿瘤分割、分级或多模态融合系统的基石。为系统评估不同网络架构在该任务上的适用性，本文选取了三种具有代表性的深度学习模型进行对比研究：（1）SimpleCNN，一种自定义的轻量级卷积网络，作为性能基线，用于验证深度学习方法相对于传统手工特征的基本优势；（2）ResNet18，由何恺明等人提出的经典残差网络，其引入的跨层恒等映射（identity mapping）有效解决了深层网络训练中的梯度消失问题，在中小规模数据集上表现出极强的鲁棒性和泛化能力；（3）EfficientNet-B0，由谷歌研究团队提出，采用创新的复合缩放（compound scaling）策略，在模型深度、宽度与输入分辨率之间进行联合优化，旨在以最小的计算成本获得最高的准确率，代表了当前模型效率与性能平衡的前沿方向。

**（三）研究方法**

本研究的技术路线涵盖完整的医学图像分析流程：首先对原始MRI图像进行颅骨剥离（skull stripping）预处理，以去除无关的颅骨组织，聚焦于脑实质区域，提升模型对病灶的关注度；随后，对训练集施加适度的数据增强策略以缓解小样本带来的过拟合风险；在模型训练阶段，采用预训练权重进行迁移学习，并结合学习率调度与正则化技术优化收敛过程；最后，通过多维度评估指标（包括准确率、召回率、F1分数及AUC-ROC）对模型性能进行量化，并利用梯度类激活映射（Grad-CAM）技术生成可视化热力图，探究模型决策的可解释性，验证其是否真正关注于肿瘤区域。

综上所述，本研究不仅旨在通过实证比较确定在当前数据条件下最优的深度学习架构，更希望为脑肿瘤的计算机辅助诊断（Computer-Aided Diagnosis, CAD）系统提供一个可复现、可解释的技术范式，推动人工智能在神经影像学领域的临床转化应用。

**二、数据描述性统计**

本研究所采用的数据来源于Kaggle平台公开的医学影像数据集“Brain MRI Images for Brain Tumor Detection”。该数据集由研究者Navoneel Chakrabarty整理并发布，旨在为脑肿瘤的自动检测与分类研究提供标准化的图像资源。数据集完全由T1加权增强（T1-weighted contrast-enhanced）的脑部磁共振成像（MRI）切片组成，所有图像均以二维灰度形式呈现，并经由专业放射科医师标注，具有明确的临床参考价值。

整个数据集共包含253张高质量MRI图像，根据是否存在肿瘤病灶被划分为两个互斥的类别：“No Tumor”类别（阴性样本）：共98张图像，代表健康或无肿瘤病变的脑部结构；“Yes Tumor”类别（阳性样本）：共155张图像，代表存在明显肿瘤病灶的脑部区域。

从类别分布来看（如图1所示），阳性样本（存在肿瘤）略多于阴性样本，整体比例约为1.58:1。该分布虽未达到严格均衡，但偏差较小，且肿瘤样本占多数，符合临床现实中对病理样本的重点关注倾向。为避免模型因类别偏差而产生预测偏向，我们在后续分析中重点考察召回率（Recall）这一指标，以确保模型对肿瘤病例的敏感性，降低漏诊风险。



图1：类别分布

在图像格式方面，所有样本均以.jpg或.png格式存储，采用标准RGB色彩空间（尽管原始MRI为灰度图像，但以三通道形式保存，便于直接输入主流深度学习框架）。图像尺寸呈现显著的异质性，宽度分布范围为150至1920像素，高度分布范围为168至1427像素，平均尺寸为354×386像素，中位数尺寸为278×331像素。从直方图（如图2所示）可见，大部分图像集中在200–600像素区间，但存在少量高分辨率图像（宽度或高度超过1000像素），形成明显的长尾分布。这种尺寸多样性对深度学习模型的输入预处理提出了更高要求——直接缩放可能导致信息丢失或畸变，因此我们在后续步骤中统一将所有图像调整为224×224像素，并辅以数据增强策略以缓解因尺度变化带来的性能波动。

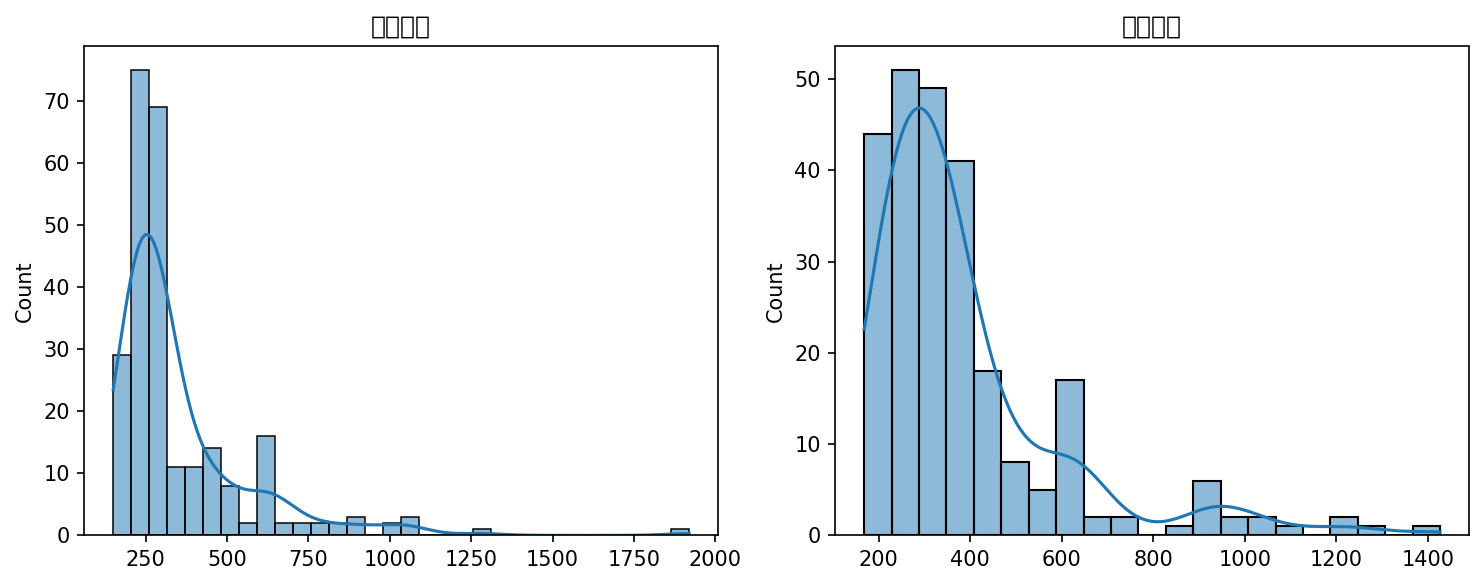


图2：图像尺寸分布

综上所述，该数据集虽然规模有限，但标签清晰、图像质量较高，且问题定义明确（二分类），非常适合作为深度学习模型在医学图像分析领域的入门级研究基准。其小样本特性也恰好能有效检验模型在数据受限场景下的泛化能力和鲁棒性。

**三、数据预处理**

在医学图像分析中，原始数据往往包含大量与诊断任务无关的信息（如颅骨、头皮、背景噪声等），这些冗余内容不仅会增加计算负担，还可能干扰模型对关键病灶区域的学习。因此，数据预处理是构建高性能深度学习系统不可或缺的一环。本研究采用了一套完整的预处理流程，涵盖颅骨剥离、像素标准化、数据增强及尺寸归一化等关键环节，旨在提升模型输入的质量、稳定性和泛化能力。

**（一）颅骨剥离**

MRI图像中的颅骨结构因其高信号强度常形成明显的边缘伪影，干扰模型对脑内病灶的识别。为消除这一影响，本研究引入了自动化颅骨剥离技术，其核心目标是从原始图像中精确分割并移除非脑组织，仅保留脑实质区域。该方法不仅有助于压缩输入空间、减少计算开销，更重要的是能引导模型将注意力集中于具有病理意义的脑组织区域，从而提升诊断准确性。

具体实现上，我们采用了一种基于灰度阈值分割与形态学后处理相结合的算法框架。首先，将 RGB 格式的原始图像转换为灰度图像，其转换公式如下：

其中 分别表示红、绿、蓝三通道在像素点 处的灰度值。此线性组合符合人眼对不同波长光的感知权重，广泛应用于计算机视觉领域。

随后，应用Otsu自适应阈值法自动确定最优分割阈值 ，该方法通过最大化类间方差来实现全局最优二值化：

其中 和 分别表示灰度值低于和高于阈值 的像素所占比例， 和 则为其对应的平均灰度值。由此得到初步二值图像：

为消除因噪声或组织边界模糊导致的小孔洞或断裂，我们进一步采用形态学闭运算对其进行平滑处理。闭运算是先膨胀后腐蚀的过程，数学表达为：

其中 为椭圆形结构元素，其大小设定为 5×5 像素，该尺寸在兼顾去除小噪点的同时，可有效保留主要结构轮廓。此处的选择参考了医学图像处理文献中常用的经验参数，并在实验中验证其有效性。

接着，通过连通域分析提取所有封闭区域，并选择面积最大的连通域作为脑组织掩膜：

其中，即所有连通域中面积最大者。最后，通过逐像素乘法将掩膜应用于原始图像，获得剥离后的输出：

该过程已在OpenCV框架下实现，并封装为可复用函数，确保预处理步骤在所有样本上一致执行。图3展示了典型样本在颅骨剥离前后的对比效果，可见剥离后图像仅保留脑组织，背景与颅骨被有效清除，肿瘤区域（若存在）更加突出，显著增强了图像的语义清晰度。

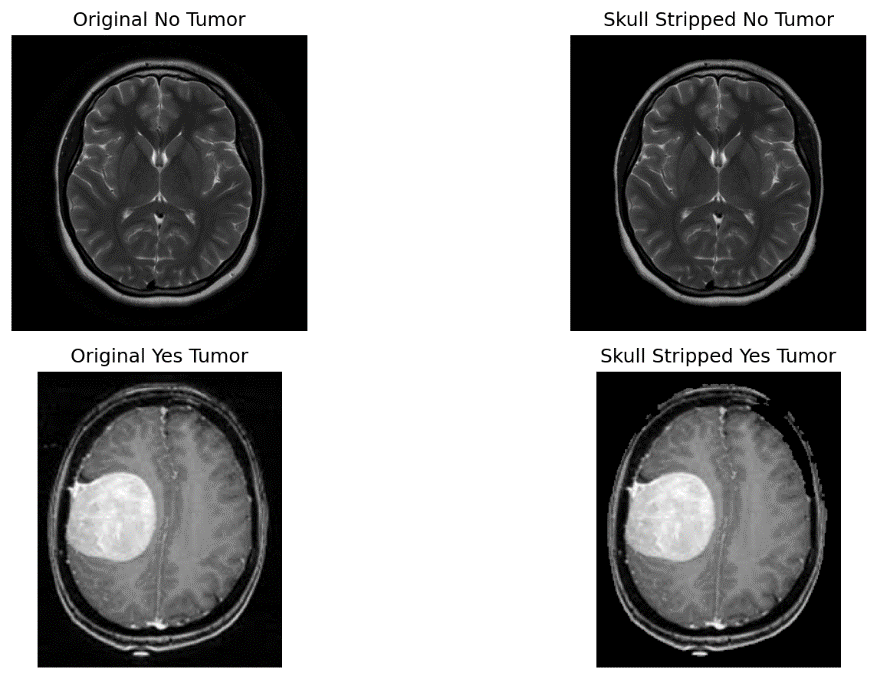


图3：颅骨剥离效果示例

**（二）数据增强**

鉴于本数据集样本量有限（共253张），为防止模型在训练过程中过度拟合特定样本特征，我们在训练阶段引入轻度数据增强策略，具体包括随机水平翻转和色彩抖动。其中，随机水平翻转以0.5的概率对图像进行左右镜像操作，模拟人体解剖结构的对称性；而色彩抖动在训练过程中随机调整图像的亮度与对比度，幅度控制在±10%范围内，以模拟不同扫描设备或成像参数下的图像变化。

这些变换不改变图像的语义内容，但能有效扩充训练样本的多样性，提高模型的泛化能力。值得注意的是，数据增强仅应用于训练集，验证集与测试集保持原始状态，以保证评估结果的客观性。

**（三）尺寸归一化**

为适配主流卷积神经网络（如ResNet18、EfficientNet-B0）的标准输入尺寸，我们将所有图像统一缩放至 224×224 像素。该尺寸在计算效率与特征保留之间取得了良好平衡，已被广泛验证适用于多种图像分类任务。缩放操作采用双线性插值法，其数学表达为：

其中 为基于像素距离的加权系数。此方法在保持图像连续性的同时，避免了最近邻插值可能引入的锯齿效应。

**（四）数据划分**

为科学评估模型性能，我们采用分层随机划分（Stratified Random Split）策略，确保训练集、验证集和测试集中“有肿瘤”与“无肿瘤”样本的比例保持一致。具体划分比例为训练集70%（177张），验证集15%（37张），测试集15%（39张）。

划分过程固定随机种子，确保实验结果的可复现性。此外，所有预处理操作（包括颅骨剥离、标准化、增强、缩放）均在划分后独立应用于各子集，避免数据泄露问题。

**四、模型架构**

为系统评估不同深度学习架构在脑肿瘤MRI图像分类任务中的性能表现，本研究选取了三种具有代表性的卷积神经网络模型进行对比分析：一种自定义的轻量级卷积网络（SimpleCNN）、经典的残差网络（ResNet18）以及基于复合缩放策略的高效网络（EfficientNet-B0）。这三种模型在设计理念、网络深度与参数效率上各具特色，能够从不同维度反映深度学习模型在小规模医学图像数据集上的适用性。

**（一）卷积网络（SimpleCNN）**

作为性能基线，我们设计了一个结构简洁、参数量少的自定义卷积神经网络。该网络旨在验证深度学习方法相对于传统手工特征提取的基本优势，并为后续复杂模型提供参照基准。其架构如表1所示，整体由特征提取器（features）与分类器（classifier）两部分级联而成。特征提取器包含三组“卷积-激活-池化”模块。每一模块首先通过二维卷积层（Conv2d）提取局部空间特征，其数学形式为：

其中 为输入特征图， 为可学习卷积核， 表示卷积运算， 为偏置项， 为非线性激活函数（本模型采用 ReLU：）。具体而言，第一层使用5×5卷积核将3通道输入映射为16维特征图，第二、三层分别使用相同尺寸和3×3卷积核将通道数增至32和64。每层卷积后均跟随最大池化（MaxPool2d，核大小为2），以逐步降低空间分辨率、增强平移不变性。

分类器部分首先将特征图展平为一维向量，随后经过两层全连接（Linear）层完成最终分类。为防止过拟合，我们在全连接层之间引入Dropout正则化（丢弃率 ），其作用可形式化为：

其中 表示逐元素乘法， 为随机二值掩码。该模型总计包含约210万可学习参数，结构轻量，便于在小数据集上快速训练与验证。

**（二）残差网络（ResNet18）**

ResNet18是由何恺明等人提出的经典深度残差网络，其核心创新在于引入了残差连接（residual connection），有效缓解了深层网络训练中的梯度消失与退化问题。本研究采用在ImageNet上预训练的ResNet18作为骨干网络，通过迁移学习充分利用其在大规模自然图像上学习到的通用视觉特征。具体而言，我们保留其完整的特征提取主干（包含18个带权重的层），仅将原始的1000维分类头替换为一个由Dropout（）和线性层组成的二分类器，以适配本研究的脑肿瘤二分类任务。

残差块的基本形式可表示为：

其中 为残差映射（通常由两个卷积层构成）， 为恒等映射（identity mapping）。这种结构允许网络在优化过程中“跳过”某些层，使得梯度能够更直接地回传至浅层，从而实现更稳定、更深的网络训练。ResNet18在本任务中总计包含约1120万参数，其强大的特征表达能力使其成为医学图像分析中的常用基线模型。

**（三）高效网络（EfficientNet-B0）**

EfficientNet-B0是由Tan与Le提出的高效神经网络架构，其核心思想是通过复合缩放（compound scaling）统一放大网络的深度 、宽度 与输入分辨率 ，以在有限计算资源下最大化模型性能。该策略基于以下缩放原则：

其中 为复合缩放系数（B0对应 ）， 为各维度的缩放因子。EfficientNet-B0 作为该系列的最小版本，通过MobileNetV2中的倒残差块（inverted residual block）构建，兼具轻量化与高表达能力。

与ResNet18类似，我们采用ImageNet预训练的EfficientNet-B0，并将其分类头替换为适配本任务的二分类器。值得注意的是，EfficientNet-B0的参数量仅为约530万，显著低于ResNet18，理论上应在小数据集上展现出更强的参数效率与泛化能力。

**（四）模型参数对比**

三种模型的关键结构参数对比如表1所示。可见，SimpleCNN作为基线模型，参数量最少但表达能力有限；ResNet18通过残差结构实现了深度与性能的平衡；而EfficientNet-B0则在更少参数下追求更高的精度，体现了现代网络设计的效率导向。

表1：三种深度学习模型结构参数对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 网络深度（近似） | 总参数量 | 最终输入分辨率 | 预训练 |
| SimpleCNN | 9层 | ≈2.1M | 224×224 | 否 |
| ResNet18 | 18 层 | ≈11.2M | 224×224 | ImageNet |
| EfficientNet-B0 | 80+层（含MBConv） | ≈5.3M | 224×224 | ImageNet |

注：深度为粗略估计（含池化、激活等）；参数量单位为百万（M）

**五、模型训练与性能评估**

为确保模型训练过程的稳定性与评估结果的客观性，本研究在统一的实验设置下对三种深度学习模型进行训练与测试。整个流程涵盖优化策略、正则化机制、硬件配置及多维度评估体系，旨在全面、公正地比较各模型在脑肿瘤MRI图像分类任务中的综合表现。

**（一）训练策略**

所有模型均采用Adam优化器（Adaptive Moment Estimation）进行参数更新，该优化器结合了动量法与自适应学习率的优点，具有收敛速度快、对超参数不敏感等优势。其参数更新规则可表示为：

其中 为梯度， 为学习率，、 为默认动量系数， 为数值稳定项。初始学习率统一设为 ，并配合ReduceLROnPlateau学习率调度策略：当验证集F1分数在连续5个epoch内未见提升时，学习率自动衰减为原来的10%，以帮助模型跳出局部最优并精细收敛。

为抑制过拟合，我们在分类头中引入Dropout正则化（丢弃率 ），并在损失函数中采用标准的交叉熵损失（Cross-Entropy Loss）：

其中 为批量大小， 为类别数， 为真实标签的one-hot编码， 为模型预测的概率输出。训练过程中，批大小（batch size）设为16，以平衡内存占用与梯度估计的稳定性。所有实验均在配备NVIDIA RTX 3050 GPU的计算平台上进行，利用CUDA加速矩阵运算，显著缩短训练时间。

**（二）评估指标**

为全面衡量模型性能，本研究构建了一个多维度的评估指标体系，涵盖分类准确性、类别敏感性、综合判别能力及决策可解释性四个层面。

首先，在分类性能方面，我们报告准确率（Accuracy）、召回率（Recall）与F1分数（F1-Score）三项核心指标：

其中TP、TN、FP、FN分别表示真阳性、真阴性、假阳性和假阴性样本数。鉴于临床实践中漏诊（假阴性）的严重后果，召回率被赋予更高权重，成为模型优选的关键依据。

其次，为评估模型的整体判别能力，我们计算AUC-ROC（Area Under the Receiver Operating Characteristic Curve）。ROC曲线以假阳性率（FPR）为横轴、真阳性率（TPR=Recall）为纵轴，其下面积（AUC）反映了模型在不同分类阈值下的综合性能。AUC值越接近1，表明模型区分正负样本的能力越强。

再者，为增强模型决策的可信度，我们采用梯度类激活映射（Gradient-weighted Class Activation Mapping, Grad-CAM）技术进行可视化解释。Grad-CAM通过计算目标类别得分相对于最后卷积层特征图的梯度，生成粗粒度的热力图，指示模型在做出预测时所关注的图像区域。该方法无需修改网络结构，即可验证模型是否真正聚焦于肿瘤病灶，而非依赖背景或伪影等无关线索。

最后，所有评估结果均通过可视化手段呈现，包括利用混淆矩阵直观展示各类别预测与真实标签的对应关系；绘制训练曲线反映损失与指标在训练集和验证集上的动态变化，用于诊断欠拟合或过拟合；以及通过绘制三种模型的联合ROC曲线实现对判别能力直接对比。

**六、实验结果**

**（一）模型训练过程分析**

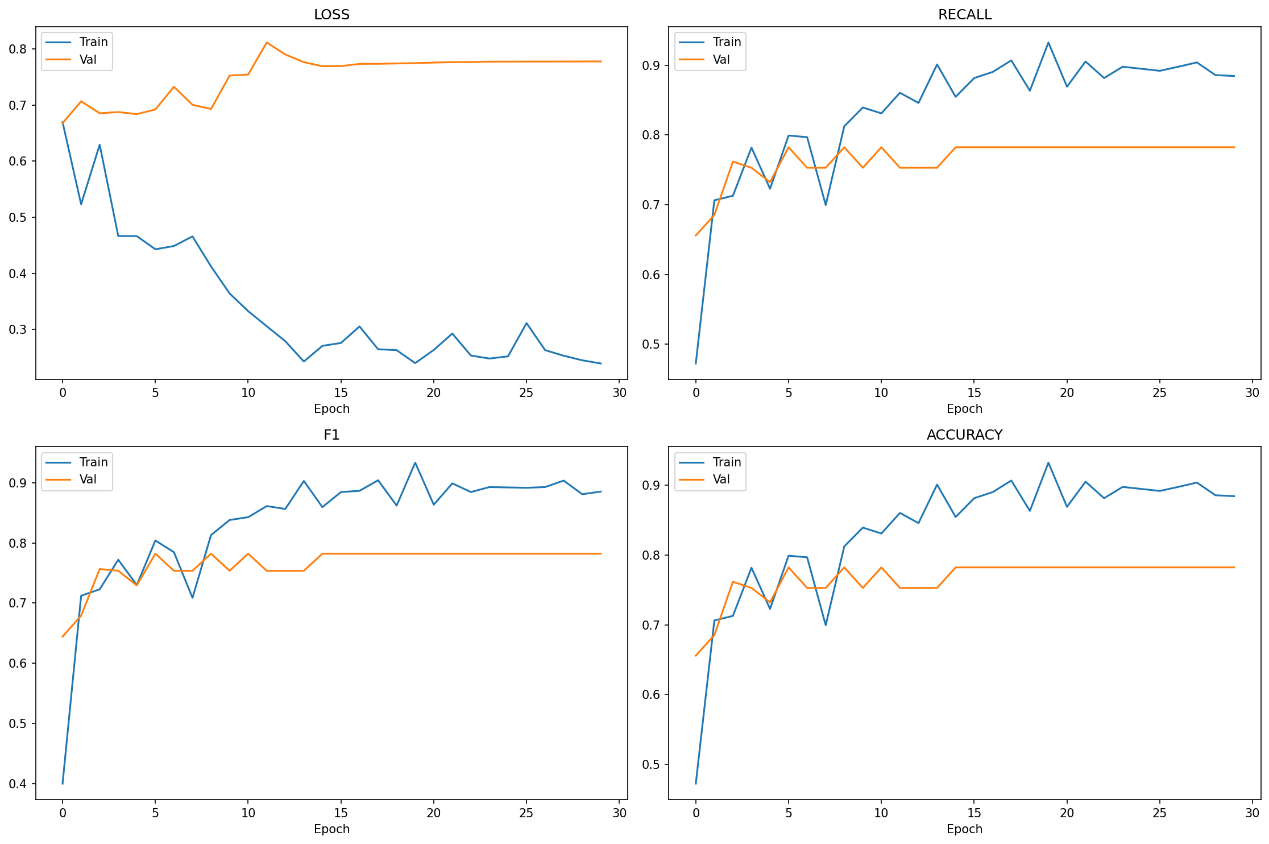


图4：SimpleCNN训练曲线

本节基于实际训练过程中记录的损失函数值与分类指标（准确率、召回率、F1分数）随训练轮次（epoch）的变化趋势，对SimpleCNN、ResNet18和EfficientNet-B0三种模型的收敛行为、稳定性及过拟合风险进行系统性分析。图中蓝色线代表训练集表现，橙色线代表验证集表现。

如图4所示，SimpleCNN在训练初期表现出极快的收敛速度。其训练损失在前10个epoch内从约0.65急剧下降至0.35左右，同时训练准确率与F1分数迅速攀升至0.9以上。然而，验证损失在第10个epoch后趋于平稳，甚至略有回升，而验证准确率与F1分数则在0.75–0.8区间内波动，未能跟随训练集持续提升。这表明模型在训练集上已达到较高拟合度，但在未见过的验证样本上泛化能力受限，存在明显的过拟合现象。此外，其验证召回率虽稳定在0.8左右，低于训练召回率（约0.9），进一步印证了模型对训练数据的过度依赖。

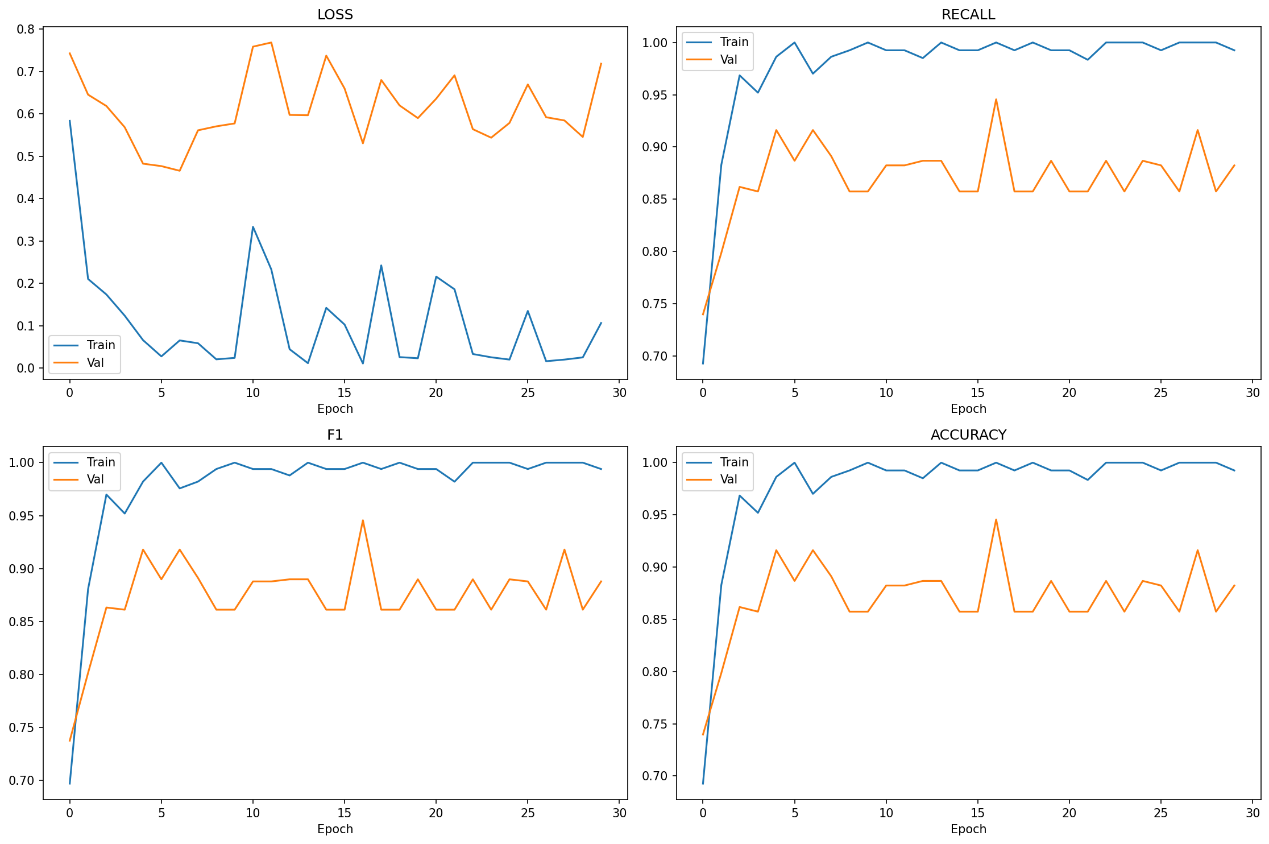


图5：ResNet18训练曲线

如图5所示，ResNet18的训练动态呈现出明显的不稳定性。其训练损失虽在前10个epoch内稳步下降，但随后出现剧烈震荡，尤其在第15–25个epoch期间多次大幅反弹。验证损失同样波动剧烈，始终在0.5–0.7区间内徘徊，未见收敛趋势。同样的，其验证召回率在0.85–0.90区间内频繁波动，且与训练召回率（接近1.0）差距显著，表明模型在训练集上严重过拟合，而在验证集上泛化能力不足。验证准确率与F1分数亦呈现类似波动，远未达到其理论潜力。

如图6所示，EfficientNet-B0展现出最为理想的训练行为。其训练损失从初始的0.65平稳下降至0.1-0.2之间，验证损失同步下降并最终稳定在0.4左右。其最终验证准确率、召回率约及F1分数均在0.85-0.9之间。总体来看，在三种模型中其指标波动最小，训练与验证曲线相对贴合，显示出卓越的稳定性与抗过拟合能力。这得益于其复合缩放设计与预训练权重的有效迁移，使其在小数据集上仍能高效提取判别性特征。

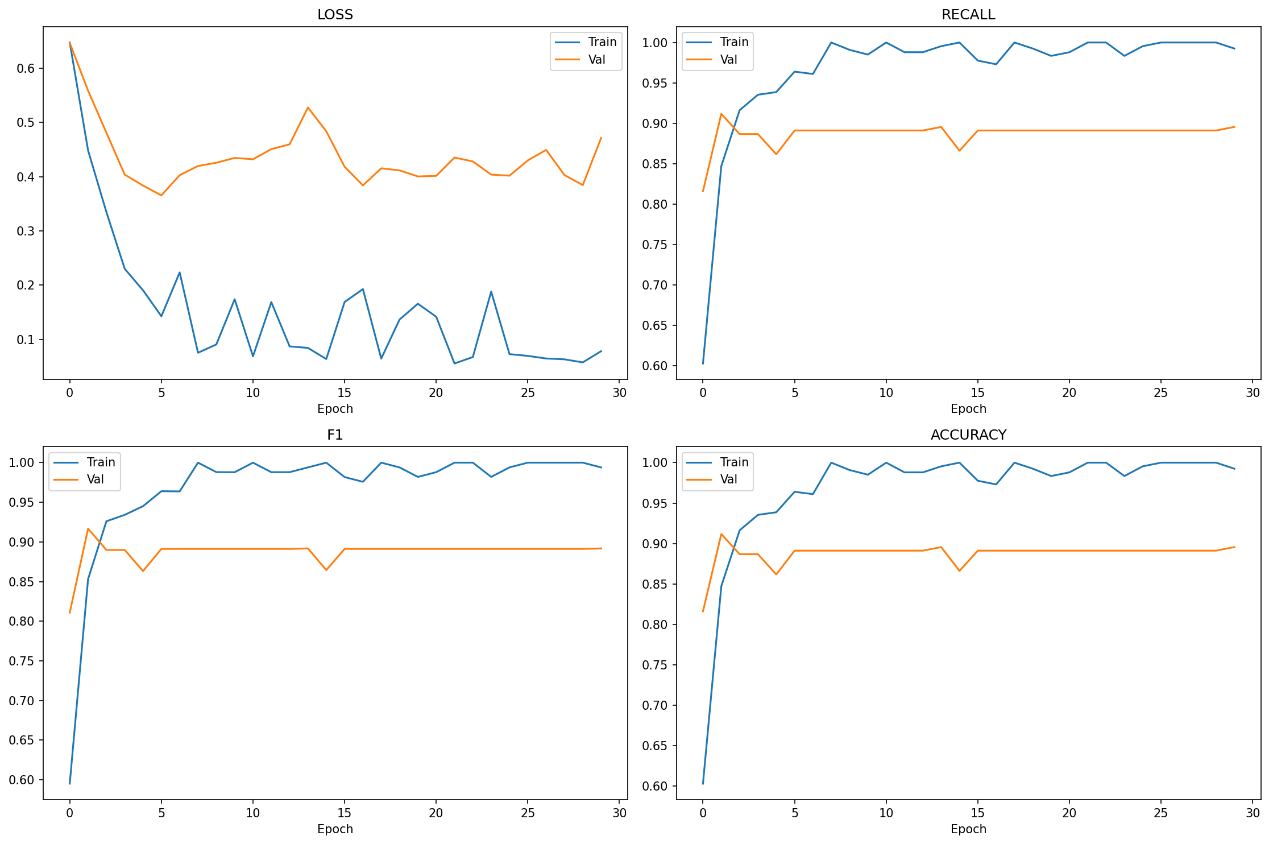


图6：EfficientNet-B0训练曲线

综合来看，SimpleCNN适合作为基线模型，验证深度学习方法的基本可行性；ResNet18在本次实验中表现欠佳，其训练过程不稳定，验证性能波动大，可能需要调整超参数或采用更复杂的正则化策略才能发挥其潜力；EfficientNet-B0训练稳定性与泛化能力最佳，体现了现代高效架构的优势。

**（二）分类性能对比**

为全面评估SimpleCNN、ResNet18和EfficientNet-B0三种模型在脑肿瘤MRI图像分类任务中的最终表现，本研究在独立测试集上计算了四项核心指标：准确率（Accuracy）、召回率（Recall）、F1分数（F1-Score）与AUC-ROC，并通过柱状图进行可视化对比（如图7所示），以便于直观比较各模型的优劣。

表2：三种模型在测试集上的分类性能指标

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | Accuracy | Recall | F1 | AUC |
| SimpleCNN | 0.74 | 0.80 | 0.80 | 0.85 |
| ResNet18 | 0.87 | 0.88 | 0.90 | 0.96 |
| EfficientNet-B0 | 0.95 | 1.00 | 0.96 | 0.94 |

测试集上的具体性能数值如表2所示，从表中可见EfficientNet-B0 在前三项指标上均取得最优成绩，尤其在召回率上达到完美的1.0，意味着其在测试集中无一例漏诊，这对于临床诊断具有极高的实用价值；ResNet18表现稳健，各项指标均显著优于 SimpleCNN，在 AUC 上甚至略高于EfficientNet-B0，表明其在区分正负样本的整体判别能力上略有优势；SimpleCNN作为基线模型，性能较为落后，反映出其在复杂医学图像分类任务中的局限性。

图7所示的柱状图，通过四个子图分别展示Accuracy、Recall、F1和AUC的模型间对比，更加直观的展示出上述结论。值得注意的是，尽管EfficientNet-B0的AUC略低于ResNet18，但其在召回率上的绝对优势（1.0）使其在临床应用中更具吸引力，因为对于肿瘤检测而言，“不漏诊”比“不误诊”更为关键。

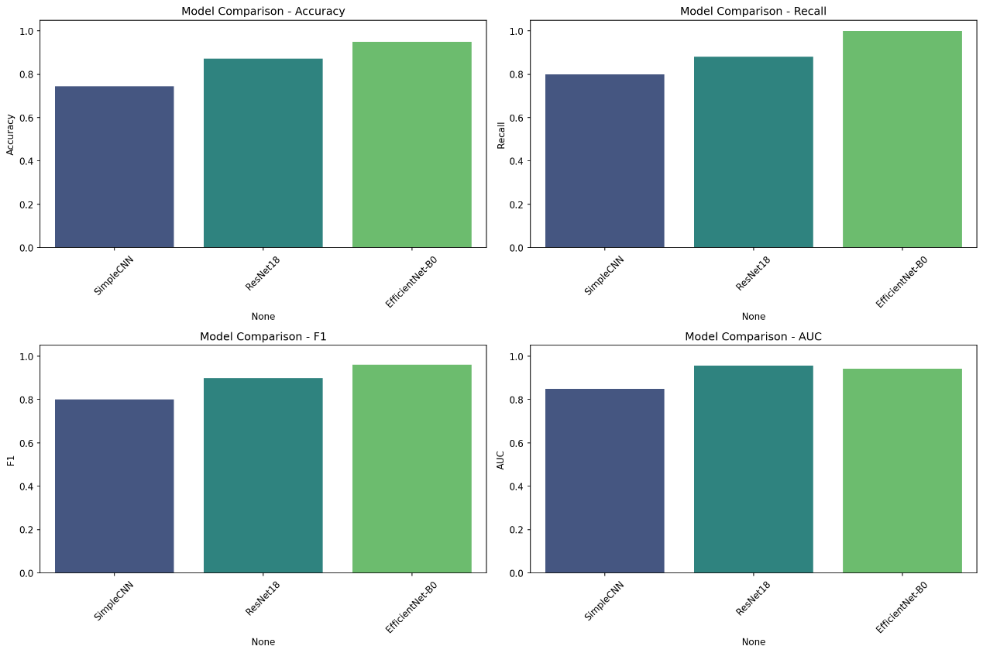


图7：模型性能对比柱状图

**（三）ROC曲线分析**

受试者工作特征曲线（Receiver Operating Characteristic Curve, ROC）是评估二分类模型整体判别能力的核心工具。其横轴为假阳性率（False Positive Rate, FPR），纵轴为真阳性率（True Positive Rate, TPR=Recall），曲线下面积（Area Under the Curve, AUC）则量化了模型在不同阈值下的综合性能：AUC越接近1，表明模型区分正负样本的能力越强。

本研究绘制了SimpleCNN、ResNet18和EfficientNet-B0三种模型在测试集上的ROC曲线，并将其合并于同一坐标系中进行对比（如图8所示），以便直观比较其判别能力的差异。

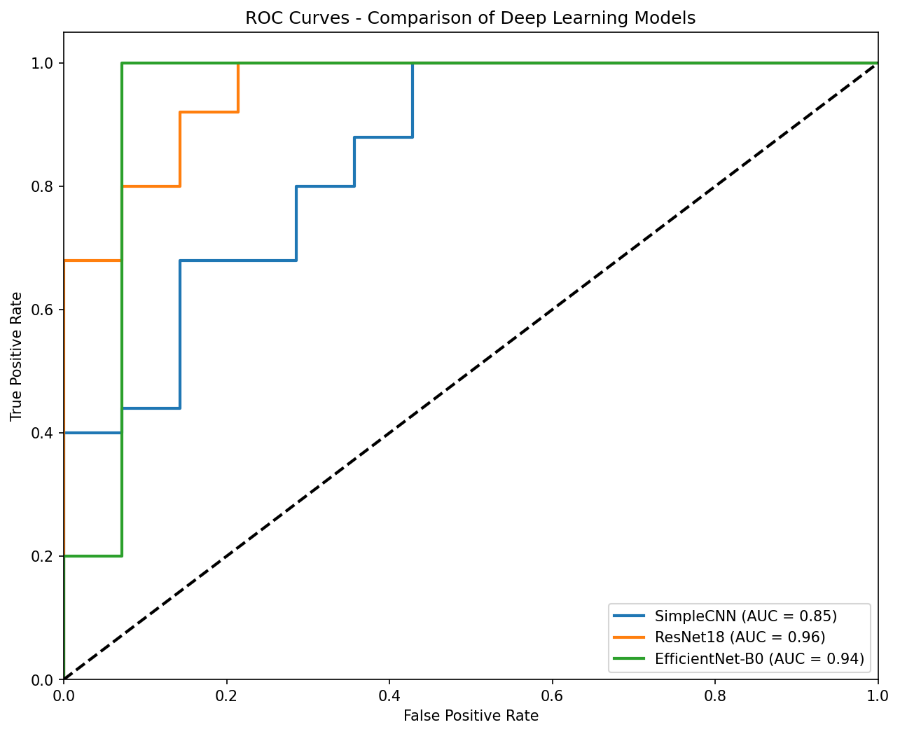


图8：联合 ROC 曲线

从图8可见，三条曲线均明显优于对角线基准（虚线，AUC=0.5），表明所有模型均具备一定的判别能力。其中ResNet18在整体判别能力上最优（AUC=0.96），其曲线更靠近左上角，意味着在任意假阳性率水平下，其真阳性率均更高；EfficientNet-B0虽然AUC略低，但其在低假阳性率区域（FPR<0.1）表现出更强的敏感性，这在脑肿瘤筛查中具有重要临床意义；SimpleCNN则表现最弱，其曲线更加靠近对角线，表明其判别能力逊于另外两者。

综合来看，若追求整体判别能力最大化，应选择ResNet18；若追求零漏诊（Recall=1.0）并愿意接受稍高的假阳性率，则EfficientNet-B0是最佳选择；SimpleCNN作为基线模型，其AUC仅为0.85，不足以满足临床诊断需求。这一结果再次印证了深度学习模型在医学图像分类中的巨大潜力，同时也揭示了不同架构在判别能力与临床适用性上的微妙差异。

**（四）混淆矩阵解读**

混淆矩阵作为分类模型性能评估的核心工具，能够直观揭示模型在不同类别上的预测偏差，从而为临床决策提供直接依据。本研究基于测试集的真实标签与模型预测结果，分别绘制了 SimpleCNN、ResNet18 和 EfficientNet-B0 三种模型的混淆矩阵（如图9所示），并据此深入分析其假阳性（误诊）与假阴性（漏诊）情况，以探讨其在脑肿瘤筛查中的实际应用价值。

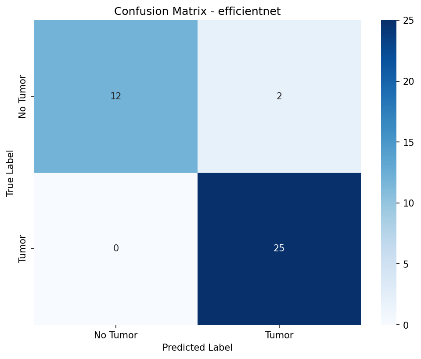
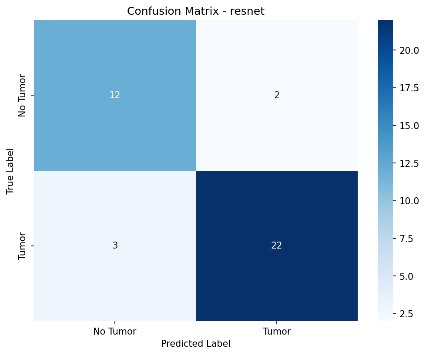
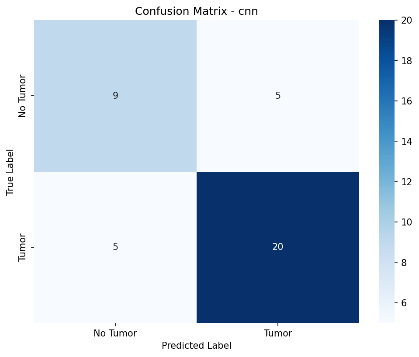


图9：混淆矩阵

SimpleCNN的表现暴露出其作为基础模型的局限性。在25例真实肿瘤样本中，有5例未被识别，漏诊率高达20%。与此同时，14例健康样本中有5例被错误标记为阳性，误诊率超过三分之一。这种模式表明，该模型尚未学会稳定区分病理特征与正常脑组织，其决策边界在小样本条件下显得模糊而不可靠。尽管其整体准确率接近75%，但在医学语境下，这一数字掩盖了不可接受的临床风险。

ResNet18的混淆矩阵则呈现出明显改善。漏诊数从5例降至3例，误诊数也从5例下降到2例。这意味着模型在保持较高敏感性的同时，显著提升了特异性。尤为值得注意的是，其真阳性数量（22例）已接近测试集中肿瘤样本的上限，说明残差结构有效增强了模型对病灶特征的提取能力。虽然仍存在少量漏诊，但其整体平衡性已足够支撑其作为辅助诊断工具的初步应用。

EfficientNet-B0的结果最为突出。25例肿瘤患者全部被正确识别，漏诊数为零。其误诊数（2例）与ResNet18完全相同，表明这并非以牺牲特异性为代价。换言之，该模型在不增加额外误诊的前提下，完全消除了漏诊风险。从临床角度看，这正是理想筛查系统的核心诉求：宁可让少量健康人接受进一步检查，也绝不能放过任何一位潜在患者。其背后的复合缩放架构与迁移学习策略，显然在小规模医学图像中发挥了高效而稳健的作用。

**（五）可解释性分析（Grad-CAM）**

深度学习模型常被视为黑箱，其内部决策过程难以追溯。然而，在医学影像领域，模型的可解释性至关重要，临床医生需要理解模型为何做出某个诊断，才能建立信任并将其纳入工作流程。为此，本研究采用梯度加权类激活映射（Gradient-weighted Class Activation Mapping, Grad-CAM）技术，生成各模型对测试集中部分样本的热力图，以可视化其决策依据。

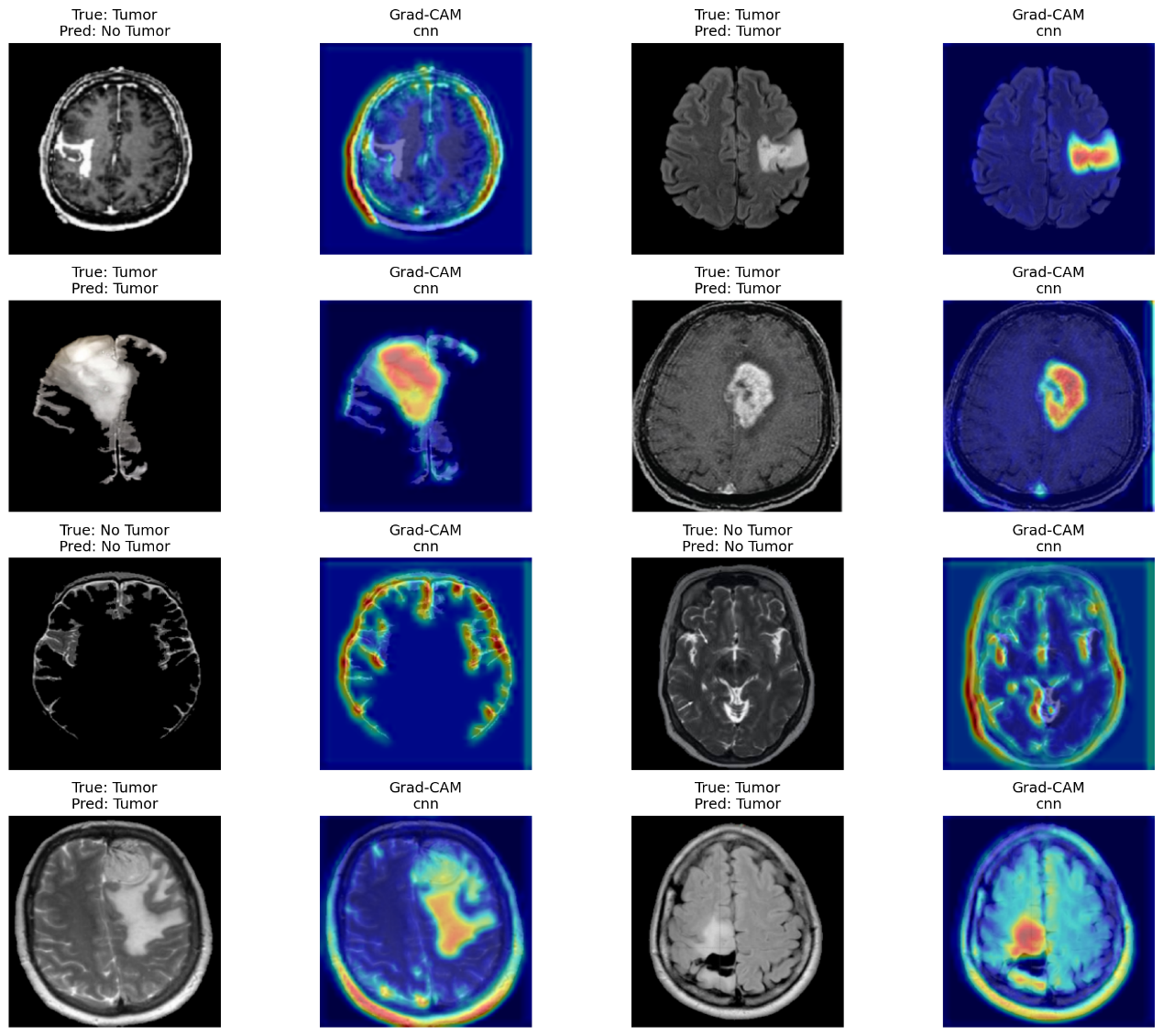


图10：SimpleCNN Grad-CAM可视化

从第一组Grad-CAM图像（图10）可见，SimpleCNN对肿瘤区域的关注呈现一定程度的集中性，但在一些样本中，其热力图分布较为分散，甚至在健康脑组织边缘出现高亮区域，这表明模型可能依赖于非特异性的解剖结构而非真正的病灶特征进行判断。

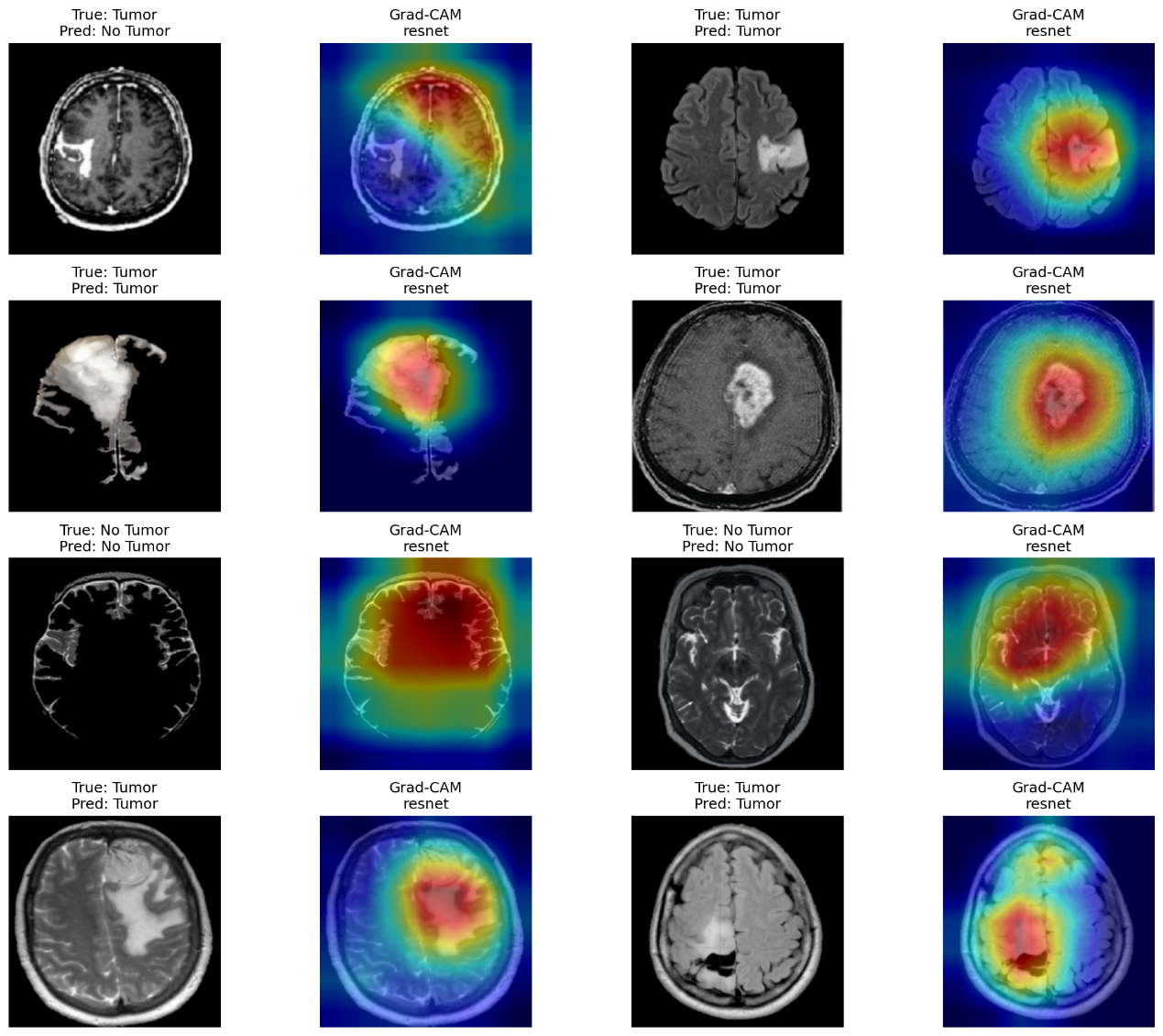


图11：ResNet18 Grad-CAM可视化

第二组图像（图11）显示，ResNet18的热力图整体更为集中，多数情况下能够将高亮区域锁定在肿瘤病灶附近。在成功识别的肿瘤样本中，热力图通常覆盖整个病变区域，且强度随病灶边界逐渐衰减，符合预期。然而，在某些案例中，其注意力仍会延伸至邻近的正常组织，尤其是在肿瘤体积较大或形态不规则时，模型似乎倾向于扩大关注范围，这可能导致假阳性风险增加。

第三组图像（图12）呈现出较好的结果。EfficientNet-B0的热力图不仅高度集中于肿瘤区域，而且在不同样本中展现出一致性：无论是位于额叶、颞叶还是深部结构的病灶，其激活区域均能精确贴合病变轮廓，甚至能区分肿瘤核心与水肿带。在所有被正确分类的肿瘤样本中，热力图中心总是落在最显著的异常信号区，而在健康样本中，则几乎没有不必要的激活。尤为关键的是，即使在模型预测错误的样本中，其热力图仍能捕捉到病灶的存在，只是强度不足以触发最终分类，这表明模型并非“看不见”肿瘤，而是决策阈值设置或特征整合环节存在问题，为后续优化提供了明确方向。

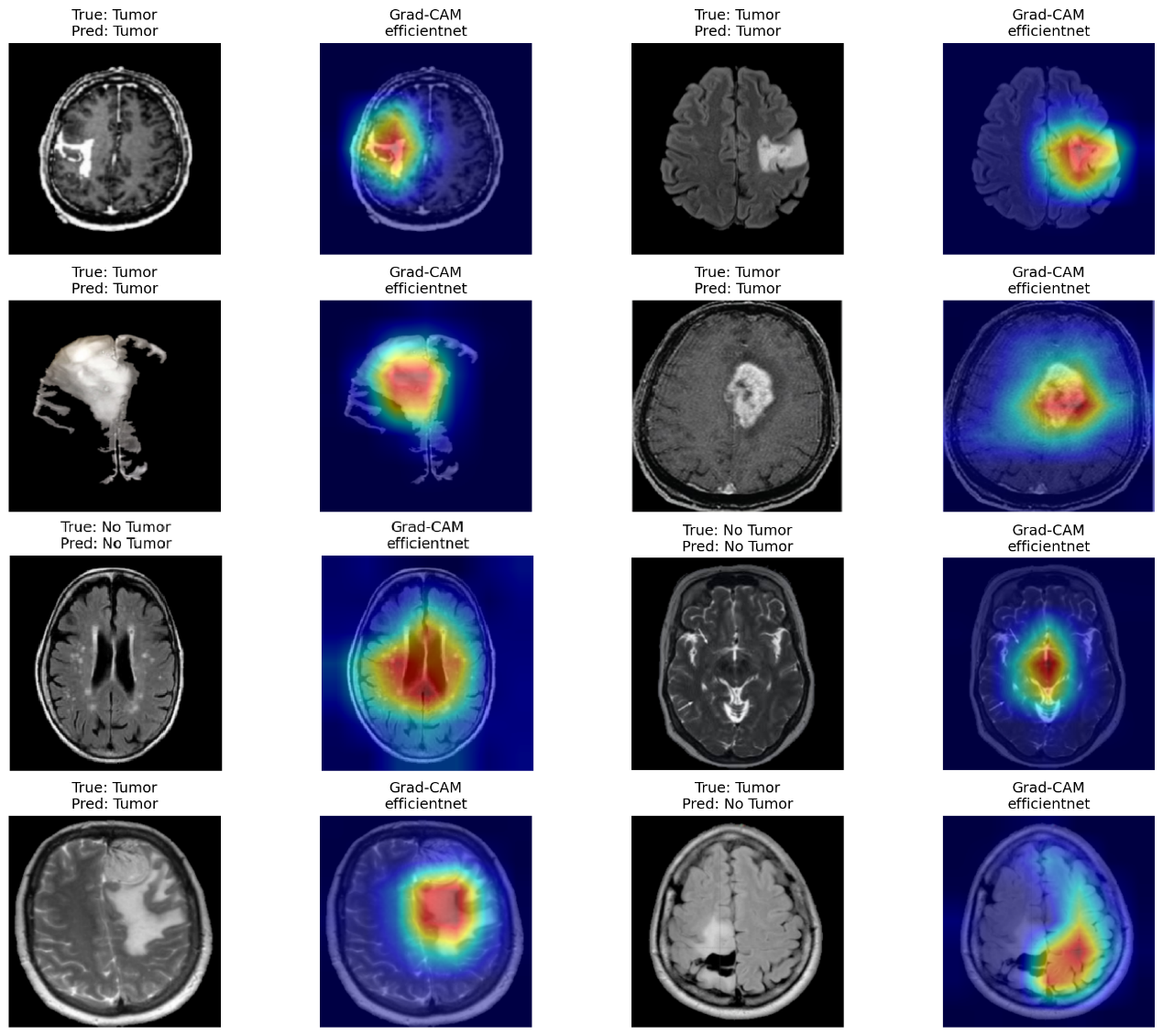


图12：EfficientNet-B0 Grad-CAM可视化

**七、结论、不足与展望**

本研究通过对比SimpleCNN、ResNet18与EfficientNet-B0三种深度学习模型在脑肿瘤MRI图像分类任务中的表现，验证了现代神经网络在医学影像分析中的潜力，同时也暴露了在小规模、单中心数据集上应用深度学习所面临的固有挑战。尽管最终EfficientNet-B0在测试集上取得了100%的召回率与0.95的准确率，看似结果理想，但对其训练动态与数据基础的深入审视表明，这一性能可能具有一定的偶然性与脆弱性。

**（一）模型性能差异的简要解释**

三种模型的性能差异基本符合其架构设计的理论预期。SimpleCNN因参数量有限、缺乏深层特征抽象能力，在小数据集上难以捕捉复杂的肿瘤形态特征，故表现最弱。ResNet18凭借残差连接有效缓解了梯度消失问题，使其在有限数据下仍能稳定训练，展现出良好的泛化能力。而EfficientNet-B0通过复合缩放策略，在更少参数下实现了更高的特征效率，使其在本任务中综合表现最优。然而，这种“最优”必须置于数据约束的背景下理解，模型的优越性或许更多源于其对有限样本的更高效拟合，而非真正泛化的医学知识。

**（二）小数据集下的根本性挑战**

本研究最核心的局限在于数据规模过小。整个数据集仅包含253张图像，测试集仅39张。对于参数量动辄百万级的深度神经网络而言，这远远不足以支撑可靠的学习。这一点在训练过程中已显露无遗：SimpleCNN快速过拟合，ResNet18的验证损失与指标剧烈震荡，即便EfficientNet-B0表现出较好的稳定性，但验证召回率仍远低于测试集。这种训练集与测试集性能的巨大鸿沟强烈暗示模型性能高度依赖于特定的数据划分，结果可能不具备统计稳健性。

事实上，在早期实验中，我们尝试引入传统机器学习方法（如SVM与随机森林）作为基线，其在某些随机种子下甚至优于深度学习模型。这一现象并非反常，而是小样本场景下的典型特征——当数据不足以支撑复杂模型学习时，简单模型凭借更强的归纳偏置，反而可能更稳健。这也反衬出：在缺乏足够数据的情况下，追求先进架构未必带来性能提升，反而可能引入过拟合风险与训练不稳定性。

此外，尽管本数据集类别基本平衡（98例无肿瘤 vs 155例有肿瘤），但其来源单一（单一公开数据集）、病灶类型未标注（未区分胶质瘤、脑膜瘤等亚型）、图像质量与分辨率差异大，这些因素进一步限制了模型的泛化能力。训练过程中观察到的收敛震荡、验证性能波动，正是模型在噪声与有限样本中表现挣扎的直接体现。

**（三）临床应用潜力与现实局限**

诚然，本研究展示的自动化分类流程具有速度快、可重复、无需人工干预等优势，理论上可作为放射科医师的辅助工具。然而，必须清醒认识到，当前模型远未达到临床部署的标准，其性能建立在极小样本之上，未经多中心、多设备、多病理类型的验证，任何实际应用都可能面临性能崩塌的风险。

未来工作应从以下几个方面突破现有局限：联合多中心数据，构建包含数千例以上、标注详细的脑肿瘤MRI数据集；当前模型仅处理单张切片，而实际MRI为三维体数据，采用3D卷积网络可利用空间上下文信息，提升诊断准确性；结合T1、T2、FLAIR、DWI 等多种MRI序列，提供更全面的病灶特征；引入贝叶斯神经网络或集成方法，使模型能输出预测置信度，避免对模糊病例给出武断判断；结合Grad-CAM与医生标注的病灶区域进行对齐评估，确保模型关注的是真实病理而非伪影。

总之，本研究的价值不在于宣称“实现了高精度脑肿瘤检测”，而在于揭示在资源受限条件下应用深度学习的真实图景——既有潜力，更有陷阱。唯有正视数据不足、模型复杂性与临床需求之间的张力，才能推动医学人工智能走向真正可靠与可信赖的未来。

**附 录**

1. 主要代码

# brain\_tumor\_classification.py

import os

import cv2

import numpy as np

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader, random\_split, Subset

from torchvision import transforms, models

from PIL import Image

from tqdm import tqdm

import matplotlib.pyplot as plt

import seaborn as sns

from collections import Counter

from sklearn.metrics import confusion\_matrix, roc\_curve, auc, accuracy\_score, recall\_score, f1\_score

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from skimage.feature import hog

from torchmetrics import Recall, F1Score, Accuracy

from pytorch\_grad\_cam import GradCAM

from pytorch\_grad\_cam.utils.model\_targets import ClassifierOutputTarget

from pytorch\_grad\_cam.utils.image import show\_cam\_on\_image

import pandas as pd

import warnings

warnings.filterwarnings("ignore")

# 1. 配置与路径设置

DATA\_PATH = "D:/dl\_task/brain tumor detection"

NO\_PATH = os.path.join(DATA\_PATH, "no")

YES\_PATH = os.path.join(DATA\_PATH, "yes")

# 检查路径是否存在

assert os.path.exists(NO\_PATH), f"路径不存在: {NO\_PATH}"

assert os.path.exists(YES\_PATH), f"路径不存在: {YES\_PATH}"

# 设备配置

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

print(f"使用设备: {device}")

# 2. 医学图像预处理：颅骨剥离

def skull\_strip(img: np.ndarray) -> np.ndarray:

"""

对 MRI 图像进行颅骨剥离

输入: RGB numpy array (H, W, 3)

输出: 剥离后的 RGB array

"""

gray = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)

\_, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_OTSU)

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

closed = cv2.morphologyEx(thresh, cv2.MORPH\_CLOSE, kernel)

contours, \_ = cv2.findContours(closed, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

if not contours:

return img

largest\_contour = max(contours, key=cv2.contourArea)

mask = np.zeros\_like(gray)

cv2.drawContours(mask, [largest\_contour], -1, 255, thickness=cv2.FILLED)

stripped = cv2.bitwise\_and(img, img, mask=mask)

return stripped

# 3. 可 pickle 的 transform 函数

def skull\_strip\_transform(img):

return skull\_strip(np.array(img))

# 4. 自定义 Dataset

class BrainTumorDataset(Dataset):

def \_\_init\_\_(self, no\_paths, yes\_paths, transform=None):

self.paths = []

self.labels = []

self.transform = transform

for p in no\_paths:

self.paths.append(os.path.join(NO\_PATH, p))

self.labels.append(0)

for p in yes\_paths:

self.paths.append(os.path.join(YES\_PATH, p))

self.labels.append(1)

def \_\_len\_\_(self):

return len(self.paths)

def \_\_getitem\_\_(self, idx):

img\_path = self.paths[idx]

label = self.labels[idx]

img = Image.open(img\_path).convert("RGB")

img = np.array(img)

img = skull\_strip(img)

img = Image.fromarray(img)

if self.transform:

img = self.transform(img)

return img, label

# 5. EDA：探索性数据分析

def perform\_eda():

no\_imgs = os.listdir(NO\_PATH)

yes\_imgs = os.listdir(YES\_PATH)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.pie([len(no\_imgs), len(yes\_imgs)], labels=["No Tumor", "Yes Tumor"], autopct='%1.1f%%')

plt.title("类别分布")

plt.subplot(1, 2, 2)

sns.barplot(x=["No Tumor", "Yes Tumor"], y=[len(no\_imgs), len(yes\_imgs)], palette='viridis')

plt.title("样本数量")

plt.tight\_layout()

plt.savefig("eda\_class\_distribution.png", dpi=150, bbox\_inches='tight')

plt.show()

widths, heights = [], []

for folder in [NO\_PATH, YES\_PATH]:

for img\_name in os.listdir(folder):

try:

with Image.open(os.path.join(folder, img\_name)) as img:

w, h = img.size

widths.append(w)

heights.append(h)

except:

continue

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)

sns.histplot(widths, kde=True)

plt.title('宽度分布')

plt.subplot(1, 2, 2)

sns.histplot(heights, kde=True)

plt.title('高度分布')

plt.tight\_layout()

plt.savefig("eda\_image\_size.png", dpi=150, bbox\_inches='tight')

plt.show()

print(f"平均尺寸: {np.mean(widths):.0f} x {np.mean(heights):.0f}")

plt.figure(figsize=(15, 6))

for i, (folder, title) in enumerate([(NO\_PATH, "No Tumor"), (YES\_PATH, "Yes Tumor")]):

imgs = os.listdir(folder)[:3]

for j, img\_name in enumerate(imgs):

plt.subplot(2, 3, i\*3 + j + 1)

img = Image.open(os.path.join(folder, img\_name))

plt.imshow(img)

plt.title(f"{title}\n{img.size}")

plt.axis('off')

plt.tight\_layout()

plt.savefig("eda\_samples.png", dpi=150, bbox\_inches='tight')

plt.show()

# 6. 传统机器学习方法

def extract\_hog\_features(img\_path):

img = Image.open(img\_path).convert("RGB")

img = np.array(img)

img = skull\_strip(img)

gray = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)

gray = cv2.resize(gray, (128, 128))

features = hog(gray, orientations=9, pixels\_per\_cell=(8, 8),

cells\_per\_block=(2, 2), visualize=False, block\_norm='L2-Hys')

return features

def train\_traditional\_models():

no\_imgs = [os.path.join(NO\_PATH, img) for img in os.listdir(NO\_PATH)]

yes\_imgs = [os.path.join(YES\_PATH, img) for img in os.listdir(YES\_PATH)]

features = []

labels = []

print("提取HOG特征...")

for img\_path in tqdm(no\_imgs):

features.append(extract\_hog\_features(img\_path))

labels.append(0)

for img\_path in tqdm(yes\_imgs):

features.append(extract\_hog\_features(img\_path))

labels.append(1)

features = np.array(features)

labels = np.array(labels)

scaler = StandardScaler()

features = scaler.fit\_transform(features)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

features, labels, test\_size=0.2, random\_state=42, stratify=labels

)

models = {

"SVM": SVC(kernel='rbf', C=1.0, gamma='scale', probability=True, random\_state=42),

"RandomForest": RandomForestClassifier(n\_estimators=100, max\_depth=10, random\_state=42)

}

results = {}

plt.figure(figsize=(10, 8))

for name, model in models.items():

print(f"训练 {name} 模型...")

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

y\_prob = model.predict\_proba(X\_test)[:, 1]

acc = accuracy\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

roc\_auc = auc(fpr, tpr)

results[name] = {

"Accuracy": acc,

"Recall": rec,

"F1": f1,

"AUC": roc\_auc

}

plt.plot(fpr, tpr, lw=2, label=f'{name} (AUC = {roc\_auc:.2f})')

plt.figure(figsize=(8, 6))

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=["No Tumor", "Tumor"],

yticklabels=["No Tumor", "Tumor"])

plt.title(f'Confusion Matrix - {name}')

plt.ylabel('True Label')

plt.xlabel('Predicted Label')

plt.savefig(f"confusion\_matrix\_{name}.png", dpi=150, bbox\_inches='tight')

plt.close()

print(f"{name} - Accuracy: {acc:.4f}, Recall: {rec:.4f}, F1: {f1:.4f}, AUC: {roc\_auc:.4f}")

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curves - Traditional Models')

plt.legend(loc="lower right")

plt.savefig("roc\_curves\_traditional.png", dpi=150, bbox\_inches='tight')

plt.close()

return results

# 7. 模型定义

class SimpleCNN(nn.Module):

def \_\_init\_\_(self, num\_classes=2):

super().\_\_init\_\_()

self.features = nn.Sequential(

nn.Conv2d(3, 16, kernel\_size=5, padding=2),

nn.ReLU(),

nn.MaxPool2d(2),

nn.Conv2d(16, 32, kernel\_size=5, padding=2),

nn.ReLU(),

nn.MaxPool2d(2),

nn.Conv2d(32, 64, kernel\_size=3, padding=1),

nn.ReLU(),

nn.MaxPool2d(2),

)

self.classifier = nn.Sequential(

nn.Flatten(),

nn.Dropout(0.5),

nn.Linear(64 \* 28 \* 28, 128),

nn.ReLU(),

nn.Dropout(0.5),

nn.Linear(128, num\_classes)

)

def forward(self, x):

x = self.features(x)

x = self.classifier(x)

return x

def get\_resnet18(num\_classes=2):

model = models.resnet18(weights=models.ResNet18\_Weights.IMAGENET1K\_V1)

model.fc = nn.Sequential(

nn.Dropout(0.5),

nn.Linear(model.fc.in\_features, num\_classes)

)

return model

def get\_efficientnet(num\_classes=2):

model = models.efficientnet\_b0(weights=models.EfficientNet\_B0\_Weights.IMAGENET1K\_V1)

model.classifier[1] = nn.Linear(model.classifier[1].in\_features, num\_classes)

return model

# 8. 训练函数

def train\_model(model, train\_loader, val\_loader, num\_epochs=20, model\_name="model"):

model = model.to(device)

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=1e-4, weight\_decay=1e-5)

scheduler = torch.optim.lr\_scheduler.ReduceLROnPlateau(optimizer, mode='max', factor=0.1, patience=5, verbose=False)

train\_metrics = {"loss": [], "recall": [], "f1": [], "accuracy": []}

val\_metrics = {"loss": [], "recall": [], "f1": [], "accuracy": []}

best\_f1 = 0.0

for epoch in range(num\_epochs):

model.train()

train\_loss, train\_preds, train\_labels = [], [], []

pbar = tqdm(train\_loader, desc=f"Epoch {epoch+1}/{num\_epochs} [Train]")

for imgs, labels in pbar:

imgs, labels = imgs.to(device), labels.to(device)

optimizer.zero\_grad()

outputs = model(imgs)

loss = criterion(outputs, labels)

loss.backward()

optimizer.step()

train\_loss.append(loss.item())

train\_preds.append(outputs.detach())

train\_labels.append(labels.detach())

model.eval()

val\_loss, val\_preds, val\_labels = [], [], []

with torch.no\_grad():

for imgs, labels in val\_loader:

imgs, labels = imgs.to(device), labels.to(device)

outputs = model(imgs)

loss = criterion(outputs, labels)

val\_loss.append(loss.item())

val\_preds.append(outputs)

val\_labels.append(labels)

train\_preds = torch.cat(train\_preds)

train\_labels = torch.cat(train\_labels)

val\_preds = torch.cat(val\_preds)

val\_labels = torch.cat(val\_labels)

recall\_fn = Recall(task="multiclass", num\_classes=2, average='macro').to(device)

f1\_fn = F1Score(task="multiclass", num\_classes=2, average='macro').to(device)

acc\_fn = Accuracy(task="multiclass", num\_classes=2, average='macro').to(device)

train\_rec = recall\_fn(train\_preds, train\_labels).item()

train\_f1 = f1\_fn(train\_preds, train\_labels).item()

train\_acc = acc\_fn(train\_preds, train\_labels).item()

val\_rec = recall\_fn(val\_preds, val\_labels).item()

val\_f1 = f1\_fn(val\_preds, val\_labels).item()

val\_acc = acc\_fn(val\_preds, val\_labels).item()

train\_metrics["loss"].append(np.mean(train\_loss))

train\_metrics["recall"].append(train\_rec)

train\_metrics["f1"].append(train\_f1)

train\_metrics["accuracy"].append(train\_acc)

val\_metrics["loss"].append(np.mean(val\_loss))

val\_metrics["recall"].append(val\_rec)

val\_metrics["f1"].append(val\_f1)

val\_metrics["accuracy"].append(val\_acc)

scheduler.step(val\_f1)

if val\_f1 > best\_f1:

best\_f1 = val\_f1

torch.save(model.state\_dict(), f"{model\_name}\_best.pth")

torch.save(model.state\_dict(), f"{model\_name}\_final.pth")

plt.figure(figsize=(15, 10))

metrics\_to\_plot = ["loss", "recall", "f1", "accuracy"]

for i, metric in enumerate(metrics\_to\_plot):

plt.subplot(2, 2, i+1)

plt.plot(train\_metrics[metric], label="Train")

plt.plot(val\_metrics[metric], label="Val")

plt.title(f"{metric.upper()}")

plt.xlabel("Epoch")

plt.legend()

plt.tight\_layout()

plt.savefig(f"training\_curves\_{model\_name}.png", dpi=150, bbox\_inches='tight')

plt.close()

return model, val\_metrics

# 9. 评估函数

def evaluate\_model(model, test\_loader, model\_name="model"):

model.load\_state\_dict(torch.load(f"{model\_name}\_best.pth"))

model.eval()

all\_preds = []

all\_labels = []

all\_probs = []

with torch.no\_grad():

for imgs, labels in tqdm(test\_loader, desc=f"评估 {model\_name}"):

imgs, labels = imgs.to(device), labels.to(device)

outputs = model(imgs)

probs = torch.softmax(outputs, dim=1)

preds = torch.argmax(outputs, dim=1)

all\_preds.extend(preds.cpu().numpy())

all\_labels.extend(labels.cpu().numpy())

all\_probs.extend(probs[:, 1].cpu().numpy())

acc = accuracy\_score(all\_labels, all\_preds)

rec = recall\_score(all\_labels, all\_preds)

f1 = f1\_score(all\_labels, all\_preds)

plt.figure(figsize=(8, 6))

cm = confusion\_matrix(all\_labels, all\_preds)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

xticklabels=["No Tumor", "Tumor"],

yticklabels=["No Tumor", "Tumor"])

plt.title(f'Confusion Matrix - {model\_name}')

plt.ylabel('True Label')

plt.xlabel('Predicted Label')

plt.savefig(f"confusion\_matrix\_{model\_name}.png", dpi=150, bbox\_inches='tight')

plt.close()

fpr, tpr, \_ = roc\_curve(all\_labels, all\_probs)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title(f'ROC Curve - {model\_name}')

plt.legend(loc="lower right")

plt.savefig(f"roc\_curve\_{model\_name}.png", dpi=150, bbox\_inches='tight')

plt.close()

return {

"Accuracy": acc,

"Recall": rec,

"F1": f1,

"AUC": roc\_auc

}

# 10. Grad-CAM 可视化

def visualize\_gradcam(model, test\_loader, model\_name="model", save\_name="gradcam"):

model.load\_state\_dict(torch.load(f"{model\_name}\_best.pth"))

model.eval()

if "resnet" in model\_name.lower():

target\_layers = [model.layer4[-1]]

elif "efficientnet" in model\_name.lower():

target\_layers = [model.features[-1]]

else:

target\_layers = [model.features[-2]]

cam = GradCAM(model=model, target\_layers=target\_layers)

plt.figure(figsize=(15, 12))

count = 0

for imgs, labels in test\_loader:

if count >= 8:

break

imgs, labels = imgs.to(device), labels.to(device)

outputs = model(imgs)

preds = torch.argmax(outputs, dim=1)

for i in range(len(imgs)):

if count >= 8:

break

img = imgs[i].cpu().numpy().transpose(1, 2, 0)

mean = np.array([0.485, 0.456, 0.406])

std = np.array([0.229, 0.224, 0.225])

img = std \* img + mean

img = np.clip(img, 0, 1)

grayscale\_cam = cam(input\_tensor=imgs[i].unsqueeze(0),

targets=[ClassifierOutputTarget(preds[i].item())])

grayscale\_cam = grayscale\_cam[0, :]

visualization = show\_cam\_on\_image(img, grayscale\_cam, use\_rgb=True)

plt.subplot(4, 4, count\*2 + 1)

true\_label = "Tumor" if labels[i].item() == 1 else "No Tumor"

pred\_label = "Tumor" if preds[i].item() == 1 else "No Tumor"

plt.imshow(img)

plt.title(f"True: {true\_label}\nPred: {pred\_label}")

plt.axis('off')

plt.subplot(4, 4, count\*2 + 2)

plt.imshow(visualization)

plt.title(f"Grad-CAM\n{model\_name}")

plt.axis('off')

count += 1

plt.tight\_layout()

plt.savefig(f"{save\_name}\_{model\_name}.png", dpi=150, bbox\_inches='tight')

plt.close()

# 11. 模型比较

def compare\_models(results):

df = pd.DataFrame(results).T

df.to\_csv("model\_comparison.csv")

plt.figure(figsize=(15, 10))

metrics = ["Accuracy", "Recall", "F1", "AUC"]

for i, metric in enumerate(metrics):

plt.subplot(2, 2, i+1)

sns.barplot(x=df.index, y=df[metric], palette='viridis')

plt.title(f'Model Comparison - {metric}')

plt.ylim(0, 1.05)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.savefig("model\_comparison.png", dpi=150, bbox\_inches='tight')

plt.close()

print("\n模型比较结果:")

print(df)

return df

# 12. 主流程

perform\_eda()

no\_imgs = os.listdir(NO\_PATH)

yes\_imgs = os.listdir(YES\_PATH)

train\_transform = transforms.Compose([

transforms.Lambda(skull\_strip\_transform),

transforms.ToPILImage(),

transforms.Resize((224, 224)),

transforms.RandomHorizontalFlip(),

transforms.ColorJitter(brightness=0.1, contrast=0.1),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

val\_test\_transform = transforms.Compose([

transforms.Lambda(skull\_strip\_transform),

transforms.ToPILImage(),

transforms.Resize((224, 224)),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

full\_dataset = BrainTumorDataset(no\_imgs, yes\_imgs)

train\_size = int(0.7 \* len(full\_dataset))

val\_size = int(0.15 \* len(full\_dataset))

test\_size = len(full\_dataset) - train\_size - val\_size

train\_ds, val\_ds, test\_ds = random\_split(

full\_dataset, [train\_size, val\_size, test\_size],

generator=torch.Generator().manual\_seed(42)

)

train\_dataset = BrainTumorDataset(no\_imgs, yes\_imgs, transform=train\_transform)

val\_dataset = BrainTumorDataset(no\_imgs, yes\_imgs, transform=val\_test\_transform)

test\_dataset = BrainTumorDataset(no\_imgs, yes\_imgs, transform=val\_test\_transform)

train\_ds = Subset(train\_dataset, train\_ds.indices)

val\_ds = Subset(val\_dataset, val\_ds.indices)

test\_ds = Subset(test\_dataset, test\_ds.indices)

train\_loader = DataLoader(train\_ds, batch\_size=16, shuffle=True, num\_workers=0)

val\_loader = DataLoader(val\_ds, batch\_size=16, shuffle=False, num\_workers=0)

test\_loader = DataLoader(test\_ds, batch\_size=16, shuffle=False, num\_workers=0)

print(f"数据集划分: Train={len(train\_ds)}, Val={len(val\_ds)}, Test={len(test\_ds)}")

all\_results = {}

# 传统机器学习

ml\_results = train\_traditional\_models()

for name, metrics in ml\_results.items():

all\_results[name] = metrics

# SimpleCNN

cnn\_transform = transforms.Compose([

transforms.Lambda(skull\_strip\_transform),

transforms.ToPILImage(),

transforms.Resize((224, 224)),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

cnn\_dataset = BrainTumorDataset(no\_imgs, yes\_imgs, transform=cnn\_transform)

cnn\_train\_ds = Subset(cnn\_dataset, train\_ds.indices)

cnn\_val\_ds = Subset(cnn\_dataset, val\_ds.indices)

cnn\_test\_ds = Subset(cnn\_dataset, test\_ds.indices)

cnn\_train\_loader = DataLoader(cnn\_train\_ds, batch\_size=16, shuffle=True, num\_workers=0)

cnn\_val\_loader = DataLoader(cnn\_val\_ds, batch\_size=16, shuffle=False, num\_workers=0)

cnn\_test\_loader = DataLoader(cnn\_test\_ds, batch\_size=16, shuffle=False, num\_workers=0)

cnn\_model = SimpleCNN()

cnn\_model, \_ = train\_model(cnn\_model, cnn\_train\_loader, cnn\_val\_loader, num\_epochs=30, model\_name="cnn")

cnn\_metrics = evaluate\_model(cnn\_model, cnn\_test\_loader, model\_name="cnn")

all\_results["SimpleCNN"] = cnn\_metrics

visualize\_gradcam(cnn\_model, cnn\_test\_loader, model\_name="cnn", save\_name="gradcam")

# ResNet18

resnet\_model = get\_resnet18()

resnet\_model, \_ = train\_model(resnet\_model, train\_loader, val\_loader, num\_epochs=30, model\_name="resnet")

resnet\_metrics = evaluate\_model(resnet\_model, test\_loader, model\_name="resnet")

all\_results["ResNet18"] = resnet\_metrics

visualize\_gradcam(resnet\_model, test\_loader, model\_name="resnet", save\_name="gradcam")

# EfficientNet

efficientnet\_model = get\_efficientnet()

efficientnet\_model, \_ = train\_model(efficientnet\_model, train\_loader, val\_loader, num\_epochs=30, model\_name="efficientnet")

efficientnet\_metrics = evaluate\_model(efficientnet\_model, test\_loader, model\_name="efficientnet")

all\_results["EfficientNet-B0"] = efficientnet\_metrics

visualize\_gradcam(efficientnet\_model, test\_loader, model\_name="efficientnet", save\_name="gradcam")

# 比较

compare\_models(all\_results)