**长　沙　学　院**

基于深度学习的肺炎检测系统开发

学 院： 长沙学院

班 级： 人工智能2班

学 号： B20210307226

姓 名： 罗国权

2024年 xx 月 xx 日

# 选题与数据集选择

肺炎是由细菌、病毒或真菌感染引起的肺部炎症，是全球儿童和老年人群体的主要致死性疾病之一。尽管现代医疗技术不断进步，肺炎的早期检测和准确诊断仍然是临床上的一大挑战，尤其是在医疗资源匮乏的发展中国家。传统的诊断方法依赖于医生的专业经验和高质量的医疗设备，但这些条件在偏远地区难以满足。此外，人为因素可能导致诊断误差，从而影响治疗决策和患者预后。因此，如何利用人工智能技术开发一种准确、高效的肺炎自动诊断工具，具有重要的研究和实践意义。

本项目基于一个公开的胸部X光（Chest X-Ray）数据集，利用深度学习方法开发一种智能化肺炎诊断系统，以期能够辅助医疗工作者快速、准确地识别肺炎患者。这种方法能够显著提高诊断效率，减少人为误差，并降低医疗成本，最终提升患者的生存率和生活质量。

项目使用的Chest X-Ray Images (Pneumonia)数据集是一个公开的医疗图像数据集，包含了5863张JPEG格式的胸部X光影像，按照“训练（train）”、“验证（val）”和“测试（test）”三部分进行组织。这些数据进一步分为两个类别：“肺炎（Pneumonia）”和“正常（Normal）”。其中，“肺炎”类别的影像来自确诊患者，影像中显示出明显的病变特征，如肺部浸润或斑点状阴影。

数据集的采集来源于广州妇女儿童医疗中心的儿科患者，影像覆盖了1至5岁儿童的前后位胸部X光片。所有影像经过质量筛查，剔除了低质量或无法辨认的样本。为保证数据标注的准确性，两名经验丰富的临床医生对影像进行初步诊断，并由第三位专家复核评估集中的样本，极大地减少了标注偏差。

这一选题具有重要的理论价值和应用潜力：

**临床价值**：模型能够快速提供诊断建议，辅助医生决策，尤其是在医疗资源不足的地区发挥作用。

**技术创新**：探索深度学习技术在医疗领域的应用，为医学影像智能分析提供一个高效的解决方案。

**教育意义**：通过项目实践，团队成员可以深入理解深度学习技术在图像分类中的具体应用，以及如何应对真实数据集中常见的问题（如不平衡数据、过拟合等）。

本项目旨在利用Chest X-Ray Images (Pneumonia)数据集开发一个基于深度学习的肺炎智能诊断系统。通过模型的优化和评估，我们希望构建一个高效、精准的分类器，为临床诊断提供强有力的技术支持，同时推动医疗领域智能化的进程。

# 数据预处理

数据预处理代码：

*from torchvision import transforms*

*# Data transformations for training set*

*train\_transforms = transforms.Compose([*

*transforms.RandomResizedCrop(224), # Randomly crop the image to 224x224*

*transforms.RandomHorizontalFlip(), # Apply random horizontal flip*

*transforms.RandomRotation(15), # Rotate the image randomly within 15 degrees*

*transforms.RandomAffine(degrees=0, scale=(0.8, 1.2)), # Random scaling*

*transforms.ToTensor(), # Convert image to tensor format*

*transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) # Normalize image*

*])*

*# Data transformations for validation and test sets*

*val\_test\_transforms = transforms.Compose([*

*transforms.Resize((224, 224)), # Resize image to 224x224*

*transforms.ToTensor(), # Convert image to tensor format*

*transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) # Normalize image*

*])*

（1）RandomResizedCrop(224):

对输入图像进行随机裁剪，并调整到224×224的固定大小，避免背景区域对模型训练的影响。同时，这样做可以增加训练样本的多样性，有助于提高模型的泛化能力。

（2）RandomHorizontalFlip():

随机水平翻转模拟实际中的变异情况，因为肺部图像的对称性使得翻转不会改变其诊断特性。有助于减轻模型对图像方向的依赖性，提高鲁棒性。

（3）RandomRotation(15):

在一定角度范围内旋转图像，模拟因拍摄角度不同导致的影像变化。提高模型对拍摄角度变化的适应能力。

（4）RandomAffine(degrees=0, scale=(0.8, 1.2)):

引入随机缩放操作模拟实际中不同影像设备下的图像比例差异。

（5）Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]):

使用ImageNet的均值和标准差对图像进行标准化，确保输入特征分布在相同范围内。加速模型收敛，同时减少梯度爆炸或消失的风险。

# 模型训练与调优

训练脚本及结果图表（准确率、损失曲线等），并解释调优过程中遇到的问题及解决方案。

训练脚本如下：

*model = models.resnet18(pretrained=True)*

*num\_features = model.fc.in\_features*

*model.fc = nn.Linear(num\_features, 2)*

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

*criterion = nn.CrossEntropyLoss()*

*optimizer = optim.Adam(model.parameters(), lr=0.001)*

*def train\_model(model, train\_loader, val\_loader, criterion, optimizer, num\_epochs=10):*

*history = {'train\_loss': [], 'val\_loss': [], 'train\_acc': [], 'val\_acc': []}*

*for epoch in range(num\_epochs):*

*model.train()*

*train\_loss = 0*

*correct = 0*

*total = 0*

*for inputs, labels in train\_loader:*

*inputs, labels = inputs.to(torch.device("cuda" if torch.cuda.is\_available() else "cpu")), labels.to(torch.device("cuda" if torch.cuda.is\_available() else "cpu"))*

*optimizer.zero\_grad()*

*outputs = model(inputs)*

*loss = criterion(outputs, labels)*

*loss.backward()*

*optimizer.step()*

*train\_loss += loss.item() \* inputs.size(0)*

*\_, predicted = outputs.max(1)*

*total += labels.size(0)*

*correct += predicted.eq(labels).sum().item()*

*train\_loss /= len(train\_loader.dataset)*

*train\_acc = correct / total*

*model.eval()*

*val\_loss = 0*

*correct = 0*

*total = 0*

*with torch.no\_grad():*

*for inputs, labels in val\_loader:*

*inputs, labels = inputs.to(torch.device("cuda" if torch.cuda.is\_available() else "cpu")), labels.to(torch.device("cuda" if torch.cuda.is\_available() else "cpu"))*

*outputs = model(inputs)*

*loss = criterion(outputs, labels)*

*val\_loss += loss.item() \* inputs.size(0)*

*\_, predicted = outputs.max(1)*

*total += labels.size(0)*

*correct += predicted.eq(labels).sum().item()*

*val\_loss /= len(val\_loader.dataset)*

*val\_acc = correct / total*

*history['train\_loss'].append(train\_loss)*

*history['val\_loss'].append(val\_loss)*

*history['train\_acc'].append(train\_acc)*

*history['val\_acc'].append(val\_acc)*

*print(f"Epoch {epoch+1}/{num\_epochs}: Train Loss: {train\_loss:.4f}, Train Acc: {train\_acc:.4f}, Val Loss: {val\_loss:.4f}, Val Acc: {val\_acc:.4f}")*

*return history*

*history = train\_model(model, train\_loader, val\_loader, criterion, optimizer, num\_epochs=10)*

*def plot\_history(history):*

*plt.figure(figsize=(12, 4))*

*plt.subplot(1, 2, 1)*

*plt.plot(history['train\_loss'], label='Train Loss')*

*plt.plot(history['val\_loss'], label='Val Loss')*

*plt.legend()*

*plt.title('Loss')*

*plt.subplot(1, 2, 2)*

*plt.plot(history['train\_acc'], label='Train Acc')*

*plt.plot(history['val\_acc'], label='Val Acc')*

*plt.legend()*

*plt.title('Accuracy')*

*plt.show()*

*plot\_history(history)*

*model.eval()*

*y\_true = []*

*y\_pred = []*

*with torch.no\_grad():*

*for inputs, labels in test\_loader:*

*inputs, labels = inputs.to(torch.device("cuda" if torch.cuda.is\_available() else "cpu")), labels.to(torch.device("cuda" if torch.cuda.is\_available() else "cpu"))*

*outputs = model(inputs)*

*\_, predicted = outputs.max(1)*

*y\_true.extend(labels.cpu().numpy())*

*y\_pred.extend(predicted.cpu().numpy())*

*accuracy = accuracy\_score(y\_true, y\_pred)*

*precision = precision\_score(y\_true, y\_pred, average='binary')*

*recall = recall\_score(y\_true, y\_pred, average='binary')*

*f1 = f1\_score(y\_true, y\_pred, average='binary')*

*print(f"Test Accuracy: {accuracy:.4f}")*

*print(f"Precision: {precision:.4f}")*

*print(f"Recall: {recall:.4f}")*

*print(f"F1 Score: {f1:.4f}")*

*print("\nClassification Report:")*

*print(classification\_report(y\_true, y\_pred, target\_names=test\_dataset.classes))*

本实验使用了经典的 ResNet18 作为基础模型，并对其最后一层进行修改以适应二分类任务。输入数据大小为 [batch\_size, 3, 224, 224]，表示每个批次的图像为彩色图像（RGB通道），经过预处理调整为 224×224 像素。初始图像像素值归一化并按通道标准化，以加速训练并提高模型性能。

（1）卷积层 1 (Conv1)：

卷积核大小：7×7，步长为 2。

卷积核数量：64。

Padding：3（保持图像尺寸适中）。

激活函数：ReLU。

输出特征图大小为 [batch\_size, 64, 112, 112]。

（2）最大池化层 (MaxPool)：

池化窗口大小：3×3，步长为 2。

输出特征图大小为 [batch\_size, 64, 56, 56]。

（3）残差模块 (Residual Blocks)：

模型包含 4 个残差模块，每个模块包含若干个卷积层和跳跃连接（Skip Connection），4个模块的参数设置如下：

模块 1：

2 个卷积层，每层卷积核数量为 64，卷积核大小为 3×3，步长为 1。

输出特征图大小为 [batch\_size, 64, 56, 56]。

模块 2：

2 个卷积层，每层卷积核数量为 128，卷积核大小为 3×3，步长为 2。

输出特征图大小为 [batch\_size, 128, 28, 28]。

模块 3：

2 个卷积层，每层卷积核数量为 256，卷积核大小为 3×3，步长为 2。

输出特征图大小为 [batch\_size, 256, 14, 14]。

模块 4：

2 个卷积层，每层卷积核数量为 512，卷积核大小为 3×3，步长为 2。

输出特征图大小为 [batch\_size, 512, 7, 7]。

（4）全局平均池化层 (Global Average Pooling)：

对特征图的每个通道执行全局平均池化，最终得到 [batch\_size, 512] 的特征向量。

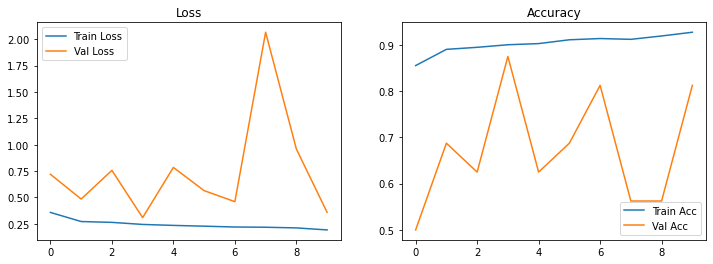
（5）全连接层 (Fully Connected Layer)：

输入维度：512。

输出维度：2（对应于二分类任务：Pneumonia/Normal）。

激活函数：Softmax，用于生成分类概率。

准确率与损失曲线如下：



# 模型评估

（1）训练过程表现：

在10个训练轮次中，模型的训练损失从0.3575逐渐降低到0.1917，训练准确率从85.53%提升至92.73%，表明模型在训练集上能够有效地学习特征。

验证集的损失波动较大，从0.7194下降至0.3583，但中间出现了显著的上升（如第8轮的2.0660）。验证准确率在多轮次中保持波动，最终稳定在81.25%。这种波动可能表明模型对验证集的泛化能力存在一定的不足。

测试集的总体准确率为90.71%，显示出较高的分类能力。

（2）模型表现参数

Precision（精确度）：

正常病例：0.97

肺炎病例：0.88

Recall（召回率）：

正常病例：0.77

肺炎病例：0.99

F1 Score：

正常病例：0.86

肺炎病例：0.93

宏平均（Macro Avg）：

精确度：0.93

召回率：0.88

F1分数：0.90

加权平均（Weighted Avg）：

精确度：0.91

召回率：0.91

F1分数：0.90

（3）模型表现分析

模型的分类准确率较高，测试集准确率达90.71%，表明模型在识别肺炎病例上表现良好，具备实际应用的潜力。特别是在肺炎病例上，召回率达到99%，说明模型几乎能够识别出所有的肺炎病例，具有较高的敏感性，对实际诊断中不遗漏病例非常关键。在肺炎病例中F1分数为0.93，权衡了精确度和召回率，证明模型的分类能力较均衡。

但同时，该模型对正常病例的识别存在不足：正常病例的召回率仅为77%，精确度和F1分数也相对较低。这可能导致部分正常病例被误分类为肺炎，增加了不必要的医疗资源使用。验证集损失波动较大：在验证过程中出现了显著的损失上升（如第8轮的2.0660），表明模型可能存在过拟合问题。对不同类别的权衡不够理想：模型更倾向于识别肺炎病例，对正常病例的判别能力有待进一步优化。

# 代码链接

GitHub链接：<https://github.com/luosheng-00/assignment>

# 总结

本实验旨在利用深度学习技术开发一个基于ResNet18的卷积神经网络模型，自动对胸部X光影像进行二分类（肺炎/正常），以辅助临床诊断。通过对公开的Chest X-Ray Images (Pneumonia)数据集的预处理、模型构建、训练与优化，我们实现了较高的分类准确率，测试集总体表现为90.71%，其中肺炎病例的召回率高达99%，证明了模型在肺炎检测中的敏感性和可靠性。实验中，模型充分利用了数据增强技术（如随机裁剪、翻转和旋转），提升了对不同影像变异的适应性。此外，通过迁移学习方法，利用预训练权重进一步优化了模型的学习效率。然而，实验也暴露了一些不足之处，如对正常病例的识别能力较弱，以及验证集损失波动较大，提示模型可能存在一定程度的过拟合。总体而言，本实验证明了深度学习技术在医疗影像分析领域的可行性与高效性，为开发肺炎自动检测系统提供了有力的技术支持。未来，可通过优化数据分布、增强正则化以及调整模型超参数等方式，进一步提升模型的泛化能力和临床适用性，为推动医疗智能化发展贡献力量。