南開大學
Nankai University

实验五

卷积神经网络——肺炎识别

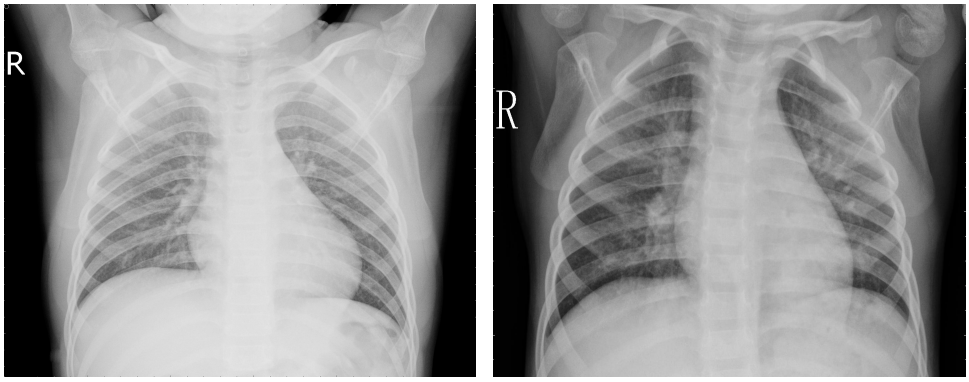
学　　校: 南开大学
学　　院: 人工智能学院
专　　业: 智能科学与技术
实验成员: 2111381 石若川
2113828 郭丰睿
2113467 王腾
2113326 王顺暄

2023 年 12 月 13 日

目录

1 问题简述

医学图像通常具有复杂性和多样性，涵盖了各种器官结构、病变和解剖学变异。传统的图像处理和特征提取方法在处理这种多样性时存在一定的局限性。随着深度学习的发展，针对医学图像的任务出现了一系列创新的神经网络架构，在不同的医学图像任务中取得了显著的成果。深度学习通过学习复杂的特征表示，能够更好地适应医学图像的多样性和复杂性。深度学习在医学图像识别中的成功应用使得图像诊断更为自动化和高效。这对于医生进行快速而准确的初步筛查，提供了重要的辅助。



(a) NORMAL (b) PNEUMONIA

图 1.1: 肺炎 X 光影像

本实验中，我们选择了卷积神经网络（CNN）和 ResNet，对肺炎图像进行识别。基于 chest_xray 数据集，我们采用了一个规模庞大的肺炎图像数据集，其中包含各种肺部疾病的高分辨率图像，以建立一个具有良好泛化能力的模型。为了克服深度神经网络中的梯度问题，我们选择了 ResNet，其残差学习结构有助于减轻梯度消失和梯度爆炸的影响。在模型训练阶段，我们对数据进行了仔细的预处理，包括调整大小和归一化，以确保模型对输入的稳健性。训练过程中，我们采用了优化算法和损失函数，并进行了详细的调优，以获得最佳性能。性能评估阶段涵盖了准确度、召回率等指标，并通过交叉验证验证了模型的鲁棒性。我们还通过可视化工具对模型的决策过程进行了解释性分析。

2 实验目的

搭建深度网络，基于肺炎影像数据集训练模型，对于给出的正常或患病肺部测试影像进行检测分类。在达成现实医学应用的同时，进一步理解图像分类任务中深度卷积网络的工作原理与优势。

3 实验内容

我们分别搭建了自建的卷积神经网络（CNN）和 ResNet 网络，因此实验总体分为两部分。每部分包含以下内容：

1. 搭建网络，使用训练数据集和验证数据集对模型进行训练。

2. 观察训练过程，给出训练与验证的损失和准确率随训练迭代变化曲线。

3. 在测试数据集上测试模型，给出测试准确率与混淆矩阵，展示预测结果，评估模型性能。

4 编译环境

电脑系统	Python 版本	CPU	GPU
Windows11	3.7	Intel i9-12900h	Nvidia GeForce RTX Get3070ti

5 实验步骤

***Step* 1: 数据准备与预处理**

将肺部影像数据划分为训练、验证、测试集，其中训练集 5216 张，验证集 624 张，测试集 16 张。对数据进行预处理，包含尺寸归一化和随机尺寸归一化等，增强数据。

***Step* 2: 模型搭建**

构建深度卷积网络：其中自建的卷积神经网络包括四个卷积层，在每层卷积后使用 ReLU 作为激活函数，并添加最大池化层进行下采样；ResNet 网络使用 ResNet34 的较浅层结构，首先分别定义残差结构块，基于此构建数量序列为 3-4-6-3 的 ResNet34 网络，使用交叉熵损失函数、Adam 优化器。

***Step* 3: 训练与训练过程评估**

使用训练数据集进行模型训练，在每个 epoch 中使用验证集评估训练性能，以此来保存最优模型权重，共训练 10 轮。在 ResNet 网络训练时，引入官方预训练权重，提升训练效率。在训练过程中同时记录训练损失、训练准确率、验证损失、验证准确率，在训练结束后分别绘制损失和准确率随 epoch 变化曲线，观察训练过程。

***Step* 4: 测试与模型效果评价**

针对测试集进行图像分类，计算测试准确率，同时构建混淆矩阵，展示具体预测结果。

6 网络构建

6.1 传统卷积神经网络

实验中构建了一个包括四层卷积的 CNN 网络，每层卷积的卷积核尺寸为 3×3。在每层卷积后使用 ReLU 作为激活函数，并添加一个最大池化层进行下采样。经过四层卷积后，使用三个全连接层得到分类输出。在各全连接层之间使用丢失率为 50% 的 Dropout，随机使部分神经元失活，增强网络的泛化能力，克服过拟合问题。网络架构如图6.2所示，输入图像尺寸为 224×224，输出为 NORMAL 与 PNEUMONIA 的二分类结果。

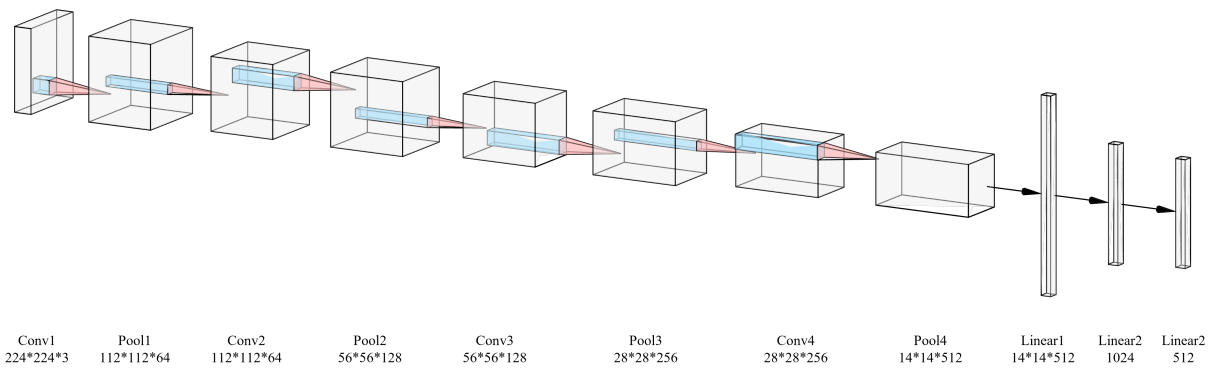


图 6.2: CNN 网络架构

网络构建的代码如下:

```
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 64, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.conv4 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(512 * 14 * 14, 1024)
        self.fc2 = nn.Linear(1024, 512)
        self.fc3 = nn.Linear(512, 2) # 2 classes: NORMAL and PNEUMONIA
        self.dropout = nn.Dropout(0.5)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        x = x.view(-1, 512 * 14 * 14)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = F.relu(self.fc2(x))
        x = self.dropout(x)
        x = self.fc3(x)
        return x
```

6.2 ResNet 网络

在这里我们构建的是 ResNet34 网络，网络整体有 34 层，核心部分以 3-4-6-3 的数量序列连接了 16 个残差块，每个残差块中设置两个卷积层。

构建过程具体而言，先创建基本残差块类，块中含有两个卷积层，每个卷积层之后都有批归一化处理，再使用 ReLU 激活函数。另外并列加入残差连接，将输入直接给向输出。这使得网络能够捕捉输入特征的非线性关系的同时，避免梯度消失问题，为网络的加深提供了基础。残差块 Block 构建为:

```
class Block(nn.Module):
    expansion = 1

    def __init__(self, in_channel, out_channel, stride=1, downsample=None,
        **kwargs):
        super(Block, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=in_channel, out_channels=out_channel,
                               kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channel)
        self.relu = nn.ReLU()
        self.conv2 = nn.Conv2d(in_channels=out_channel, out_channels=out_channel,
                               kernel_size=3, stride=1, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channel)
        self.downsample = downsample

    def forward(self, x):
        identity = x
        if self.downsample is not None:
            identity = self.downsample(x)

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        out += identity
        out = self.relu(out)

        return out
```

进一步构建整体网络，第一层为初始卷积层和批归一化层，包括一个 7x7 的卷积，用于处理输入图像的初始特征提取，随后是批归一化处理和 ReLU 激活函数；之后连接最大池化层，使用 3x3 的最大池化，以步幅为 2 进行降采样，减小空间维度；随后进入残差网络核心，构建四层基本块组，分别

包含不同数量的残差块。每个层级内的残差块的通道数逐渐增加，同时进行空间降采样。最后加入一个全局平均池化层和一个全连接层，用于最终分类。

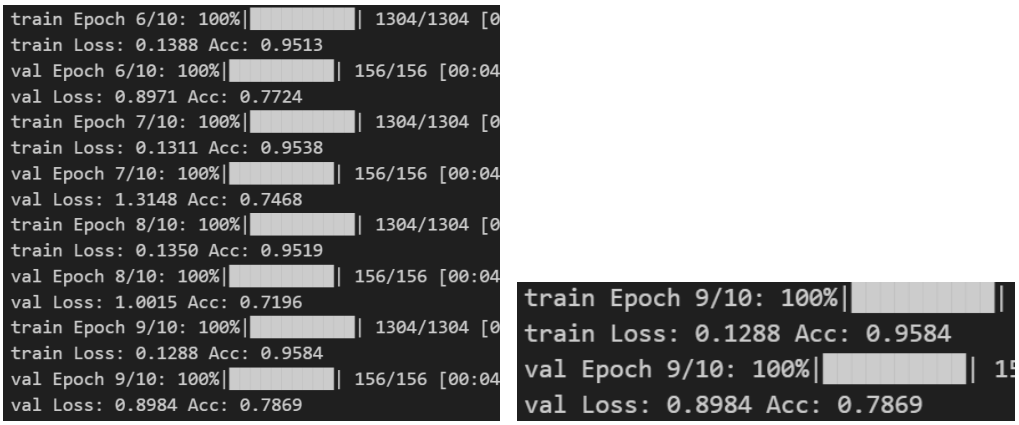
```
class ResNet(nn.Module):

    def __init__(self,block,blocks_num,num_classes=1000,include_top=True,
                 groups=1,width_per_group=64):
        super(ResNet, self).__init__()
        self.include_top = include_top
        self.in_channel = 64

        self.groups = groups
        self.width_per_group = width_per_group

        self.conv1 = nn.Conv2d(3, self.in_channel, kernel_size=7, stride=2,
                               padding=3, bias=False)
        self.bn1 = nn.BatchNorm2d(self.in_channel)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
        self.layer1 = self._make_layer(block, 64, blocks_num[0])
        self.layer2 = self._make_layer(block, 128, blocks_num[1], stride=2)
        self.layer3 = self._make_layer(block, 256, blocks_num[2], stride=2)
        self.layer4 = self._make_layer(block, 512, blocks_num[3], stride=2)
        if self.include_top:
            self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
            self.fc = nn.Linear(512 * block.expansion, num_classes)

        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.kaiming_normal_(m.weight, mode='fan_out', nonlinearity='relu')

    def _make_layer(self, block, channel, block_num, stride=1):
        downsample = None
        if stride != 1 or self.in_channel != channel * block.expansion:
            downsample = nn.Sequential(
                nn.Conv2d(self.in_channel, channel * block.expansion, kernel_size=1,
                    stride=stride, bias=False),
                nn.BatchNorm2d(channel * block.expansion))

        layers = []
        layers.append(block(self.in_channel,channel,downsample=downsample,
        stride=stride,
```

```
                        groups=self.groups,width_per_group=self.width_per_group))
        self.in_channel = channel * block.expansion

        for _ in range(1, block_num):
            layers.append(block(self.in_channel,channel,groups=self.groups,
                                width_per_group=self.width_per_group))

        return nn.Sequential(*layers)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)

        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)

        if self.include_top:
            x = self.avgpool(x)
            x = torch.flatten(x, 1)
            x = self.fc(x)

        return x
```

7 实验结果

7.1 传统卷积神经网络

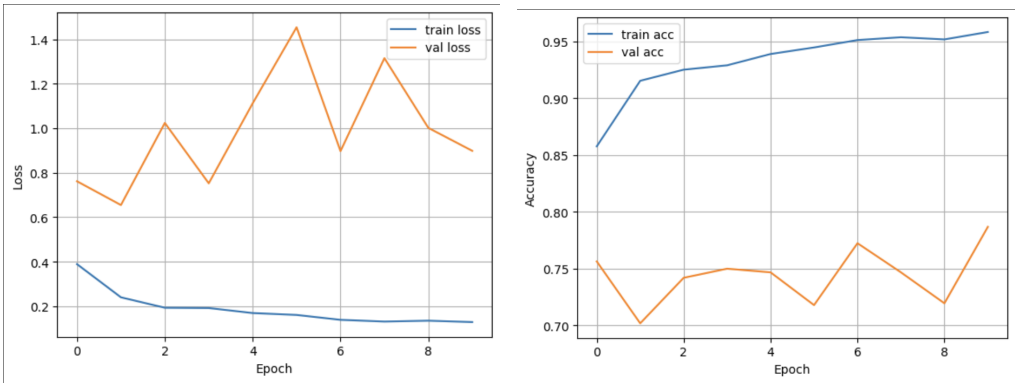
模型训练过程如图7.3(a) 所示。其中最好的一轮 epoch 下，训练准确率为 0.9584，训练损失为 0.1288，验证损失为 0.8984，验证准确率为 0.7869，如图7.3(b) 所示。



(a) 训练过程 (b) 最优 epoch

图 7.3: CNN 模型训练

训练过程中损失和准确率的变化曲线见图7.4。



(a) 损失变化 (b) 准确率变化

图 7.4: ResNet 训练参数曲线

在损失变化图中，训练集损失稳步下降，验证集损失所有波动但最终变化不大；在准确率变化图中，训练准确率稳步上升，而验证准确率波动上升，说明随着训练进行，模型逐步迭代寻找最优分类方法。

随后进行模型测试，在测试集上计算得到的测试准确率为 **87.5%**，其混淆矩阵如图7.5。

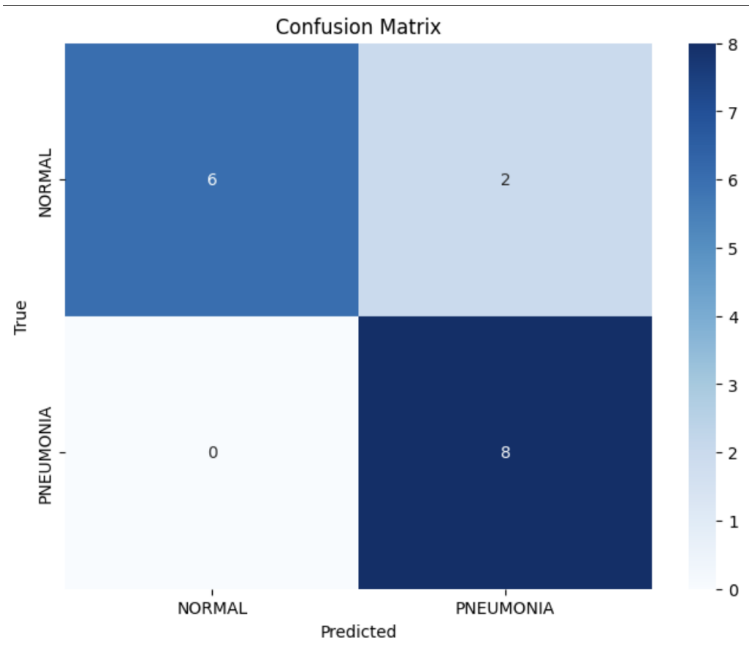


图 7.5: CNN 测试混淆矩阵

具体展示 16 张测试数据图片，上标预测值，括号内为实际值，当预测错误则标红显示，如图7.6。

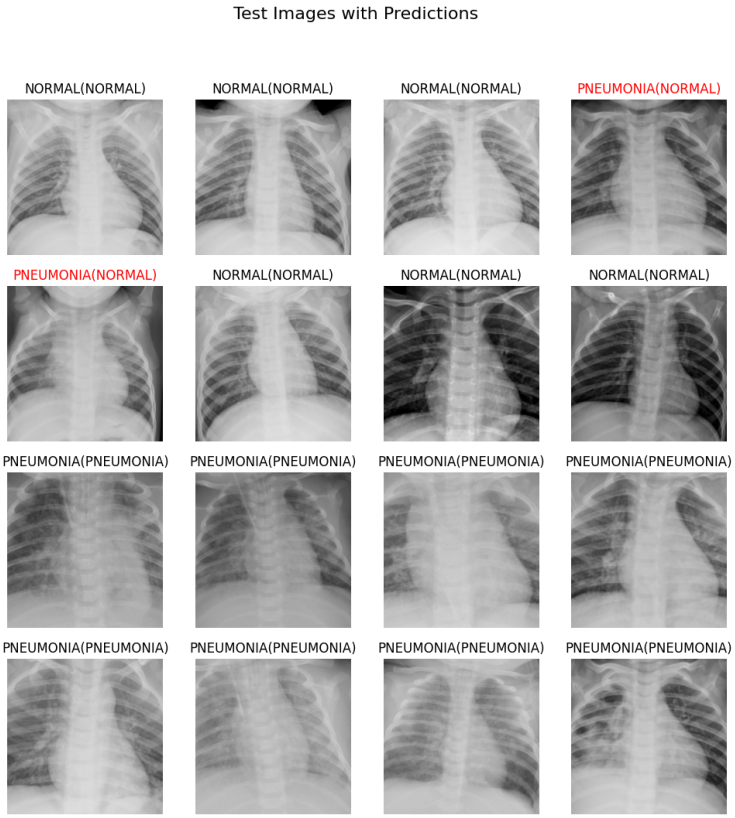
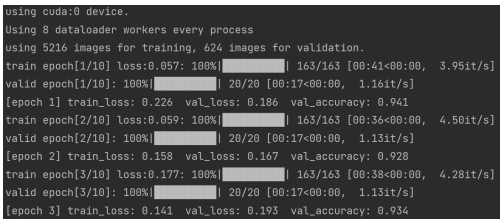


图 7.6: CNN 测试图像

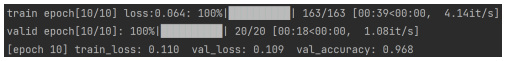
在 10 个 epoch 的训练后，16 张测试集中有两张被预测错误，达到了一个较为满意的准确率，说明肺炎图像的分类识别问题在卷积网络下的效果较优，即 CNN 的网络架构适用于该分类任务。

7.2 ResNet 网络

模型训练过程如图7.7(a) 所示。其中最好的一轮 epoch 下，训练损失为 0.110，验证损失为 0.109，验证准确率为 0.968，如图7.7(b) 所示。



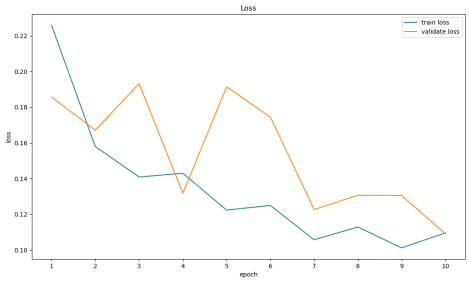
(a) 训练过程



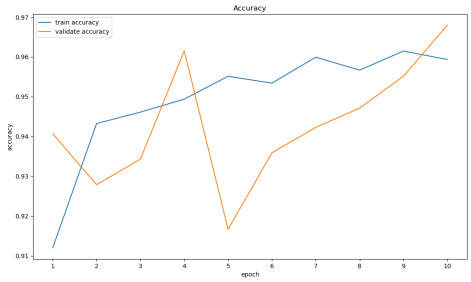
(b) 最优 epoch

图 7.7: ResNet 模型训练

训练过程中损失和准确率的变化曲线见图7.8。



(a) 损失变化



(b) 准确率变化

图 7.8: ResNet 训练参数曲线

在损失变化图中随着训练的进行，虽然数值变化存在一定波动，训练损失和验证损失整体都呈现下降趋势。而验证损失波动幅度大于训练损失，其下降速度也慢于训练损失下降；准确率变化图中两曲线均波动上升，说明随着训练进行，模型逐步寻优迭代，符合模型训练规律。

随后进行模型测试，在测试集上计算得到的测试准确率为 **93.75%**，其混淆矩阵如图7.9。

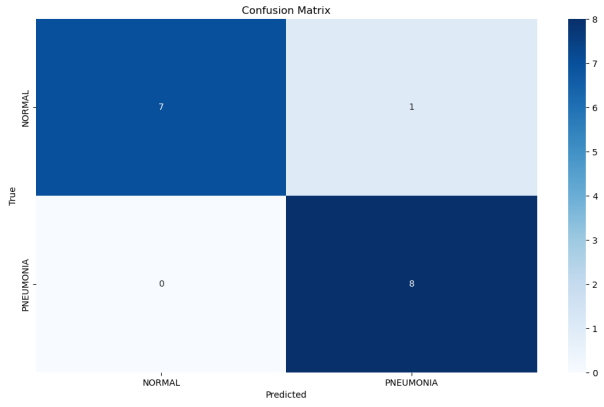


图 7.9: ResNet 测试混淆矩阵

具体展示 16 张测试数据图片，上标预测值，括号内为实际值，当预测错误则标红显示，如图7.10。

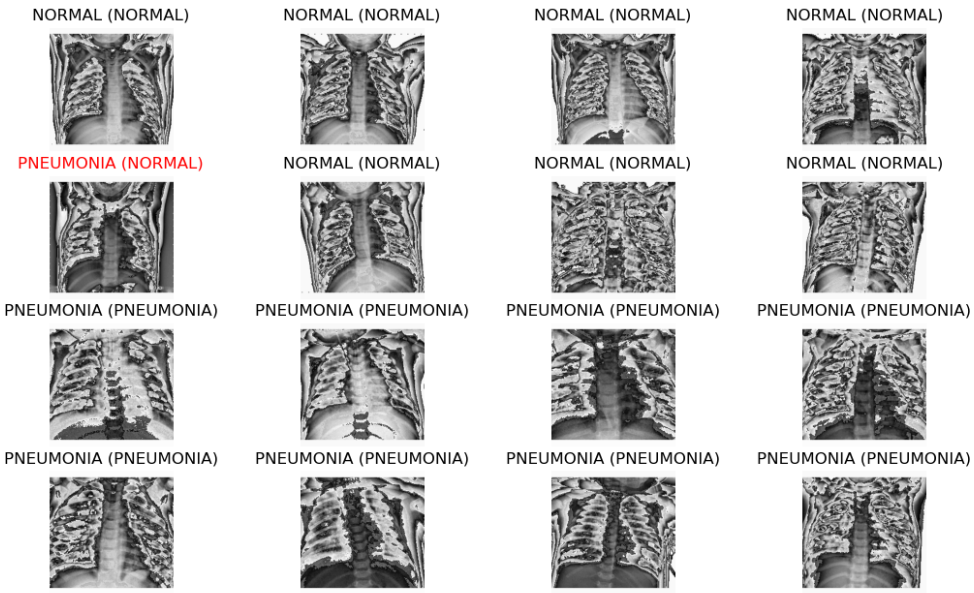


图 7.10: ResNet 测试图像

结果可见，尽管使用 ResNet 网络仅仅进行了 10 轮迭代，模型在测试集上仅有一张图像被预测错误，准确率较高，远高于自建卷积神经网络。这印证了 ResNet 网络在图像分类任务中的优越性，它使用残差结构和残差连接，有效地避免梯度消失和网络退化问题，使得网络构建可以更深更宽，表达能力大幅提升。

8 分析总结

本实验中使用了传统 CNN 网络与 ResNet 网络对 chest_xray 数据集中的肺炎图像进行了分类识别。结果显示，CNN 网络在测试集上的准确率为 87.5%，ResNet 网络在测试集上的准确率为 93.75%。这表明 ResNet 网络对于图像分类问题的解决具有更好的效果。

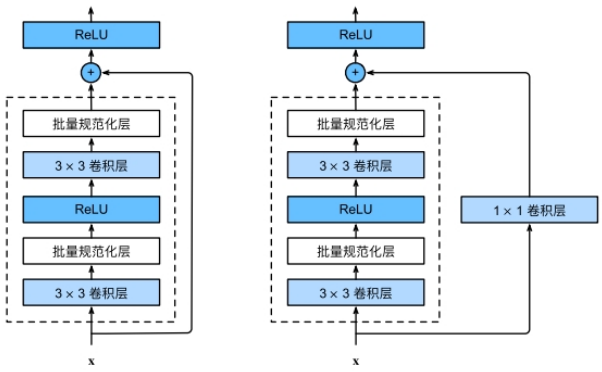


图 8.11: ResNet 网络结构

ResNet 能取得更好的分类效果，是由于 ResNet 引入了残差学习的概念。随着网络深度的增加，梯度消失或梯度爆炸问题会变得更加显著。在传统的深度网络中，通过反向传播训练时，梯度可能变得非常小，导致权重更新几乎不可见，或者变得非常大，导致权重更新过大。ResNet 通过使用残差块，引入了跳跃连接，通过在网络中引入跨层的直接连接，允许信息在网络中更快地传播。这有助于解决深层网络中的梯度消失和梯度爆炸问题，使得网络更容易训练和优化。这使得 ResNet 可以具有更深的网络结构。因此，ResNet 可以捕捉更丰富的图像特征。深层网络可以通过多个卷积层和非线性激活函数逐渐提取图像的高级特征，从而更好地区分不同类别的图像。

通过本实验，我们掌握了卷积神经网络应用于图像分类问题的基本原理；通过搭建传统 CNN 网络与 ResNet 网络，加深了对于卷积神经网络架构的理解；通过比较传统 CNN 网络与 ResNet 网络的分类准确率，体现了 ResNet 更好的分类效果，并解释了 ResNet 网络性能更优的原因。

本实验作为深度学习方法应用于医疗图像分类的简单示例，体现了深度学习方法在医疗图像领域的广阔应用前景。随着技术的不断进步和对医疗图像理解的深入，深度学习在医学图像分类中的应用将继续取得创新性的进展，为医学诊断和治疗提供更有效的支持。

A 附录: 附件说明

文件名	文件内容
my_CNN_4layres.ipynb	4 层 CNN 网络源代码
model.py	ResNet 网络构建源代码
train.py	ResNet 训练源代码
batch_predict.py	ResNet 预测测试源代码
class_indices.json	ResNet 分类类别索引文件
resNet34.pth	ResNet 网络权重

注：ResNet 预训练权重未包含