山东大学

崇新学堂

2025－2026 学年第 1 学期

实 验 报 告

课程名称：信息基础Ⅱ

实验名称：ResNet-18

专 业 班 级　崇新 23

学 生 姓 名　杨瑞

实 验 时 间　2025/10/13

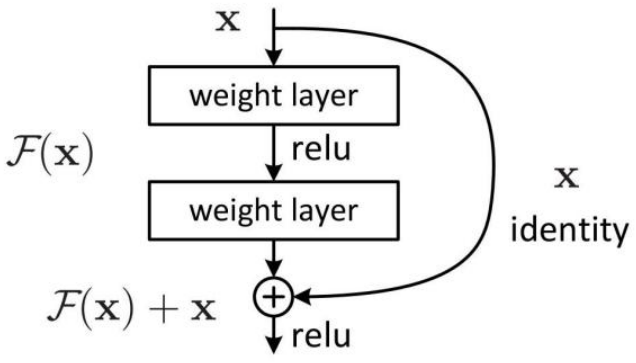
一、知识梳理

1. ResNet 简介

ResNet（Residual Network，残差网络）由何恺明等人于 2015 年提出，论文为 “Deep Residual Learning for Image Recognition”，并在 2016 年 CVPR 获得最佳论文奖。ResNet 在 ImageNet 图像分类比赛中以显著优势取得了冠军，是深度学习史上极具影响力的网络结构之一。

ResNet 的核心思想是通过引入残差学习（Residual Learning）来缓解深层神经网络中的梯度消失和性能退化问题。当网络层数加深时，传统结构容易出现训练误差上升的现象，而残差结构通过在网络中加入跳跃连接（Shortcut Connection），使得信号能够直接跨层传播，从而保持梯度的稳定传递。

2. 残差块（Residual Block）结构



残差块的数学形式为：

$$y = F(x, W_i) + x$$

其中 F(x, W_i) 表示卷积与非线性变换后的结果，x 为输入信号。当输入与输出维度不同步时，使用 1×1 卷积进行通道匹配。

3. ResNet-18 网络结构

ResNet-18 是 ResNet 系列中的一种轻量级版本，总计 18 层可训练参数（不计池化层）。结构如下：

conv1: 7×7 卷积, stride=2, 输出通道 64

conv2_x: 残差块×2, 输出通道 64

conv3_x: 残差块×2, 输出通道 128

conv4_x: 残差块×2, 输出通道 256

conv5_x: 残差块×2, 输出通道 512

平均池化 + 全连接层: 输出 10 类

二、代码构建过程

1. 网络定义

本实验使用 PyTorch 实现 ResNet-18 网络。首先定义残差块（Residual），包括两个 3×3 卷积层与可选的 1×1 卷积层用于维度匹配：

```
class Residual(nn.Module):
    def __init__(self, input_channels, num_channels, use_1x1conv=False, strides=1):
        super().__init__()
        self.conv1 = nn.Conv2d(input_channels, num_channels, kernel_size=3, padding=1, stride=strides)
        self.conv2 = nn.Conv2d(num_channels, num_channels, kernel_size=3, padding=1)
        if use_1x1conv:
            self.conv3 = nn.Conv2d(input_channels, num_channels, kernel_size=1, stride=strides)
        else:
            self.conv3 = None
        self.bn1 = nn.BatchNorm2d(num_channels)
```

```
        self.bn2 = nn.BatchNorm2d(num_channels)
        self.relu = nn.ReLU(inplace=True)



    def forward(self, X):
        #输入 X 先经过第一个卷积层，再经过批量归一化层，最后经过 ReLU 激活函数
        Y = F.relu(self.bn1(self.conv1(X)))
        #Y 再经过第二个卷积层，再经过批量归一化层，最后经过 ReLU 激活函数
        Y = self.bn2(self.conv2(Y))
        if self.conv3:
            X = self.conv3(X)
        #残差连接
        Y += X
        return F.relu(Y)
```

2. 数据预处理

数据集选用 CIFAR-10，包含 60000 张彩色图片（32×32），分为 10 个类别。训练集：50000 张，测试集：10000 张。训练数据进行了随机水平翻转增强，并统一缩放至 224×224 尺寸。

```
# 数据预处理和增强
transform_train = transforms.Compose([
    transforms.Resize(224),  # 调整大小到 224x224
    transforms.RandomHorizontalFlip(),  # 随机水平翻转增强
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

transform_test = transforms.Compose([
    transforms.Resize(224),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])
# CIFAR-10 类别名称
classes = ['airplane', 'automobile', 'bird', 'cat', 'deer',
           'dog', 'frog', 'horse', 'ship', 'truck']
# 加载数据集 - 保存到 d:/Information_HW/Data/CIFAR-10
train_dataset = torchvision.datasets.CIFAR10(
    root='d:/Information_HW/Data/CIFAR-10', train=True, download=True, transform=transform_train)
test_dataset = torchvision.datasets.CIFAR10(
    root='d:/Information_HW/Data/CIFAR-10', train=False, download=True, transform=transform_test)
```

3. 训练设置

学习率 lr = 0.01；批次大小 batch_size = 128；轮数 num_epochs = 20；优化器：SGD + Momentum(0.9)；损失函数：交叉熵（CrossEntropyLoss）。

在后续实验中，为了解决过拟合问题，在优化器中添加了权重衰减项（L2 正则化）：

```
# 调整学习率和批次大小以适应 CIFAR-10
lr, num_epochs, batch_size = 0.01, 20, 128

# 在创建优化器时添加权重衰减
optimizer = torch.optim.SGD(net.parameters(), lr=lr, momentum=0.9, weight_decay=5e-4)  # 添加 L2 正则化
```

三、实验结果

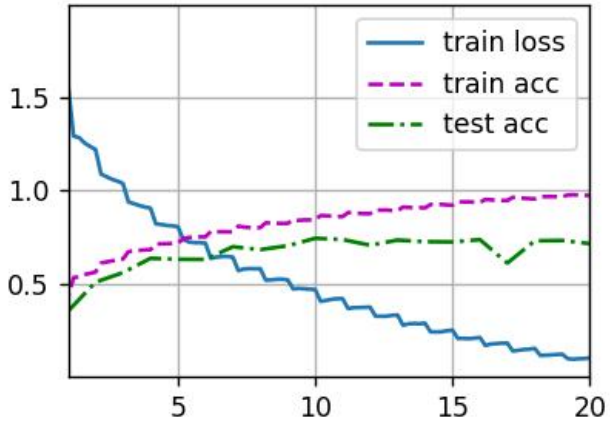
1. 初始训练（未加正则化）

模型在前期训练中表现出过拟合现象：

Loss = 0.100，训练准确率 97.1%，测试准确率 71.3%。

说明模型在训练数据上拟合良好，但泛化能力较弱。

```
loss 0.011, train acc 0.997, test acc 0.918
506.1 examples/sec on cuda:0
```

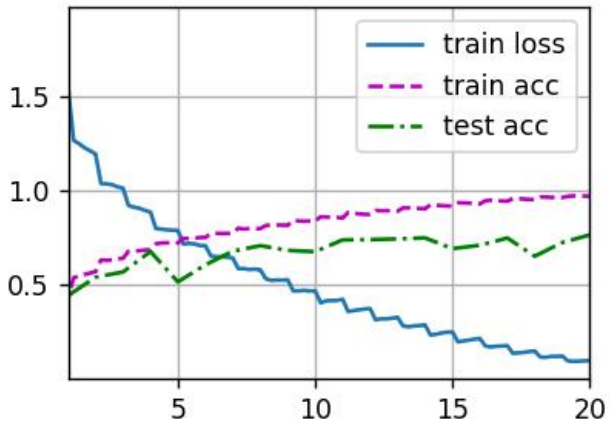


2. 添加 L2 正则化后

在优化器中加入 weight_decay=5e-4 后，模型的测试集准确率有所提升，过拟合程度减轻。

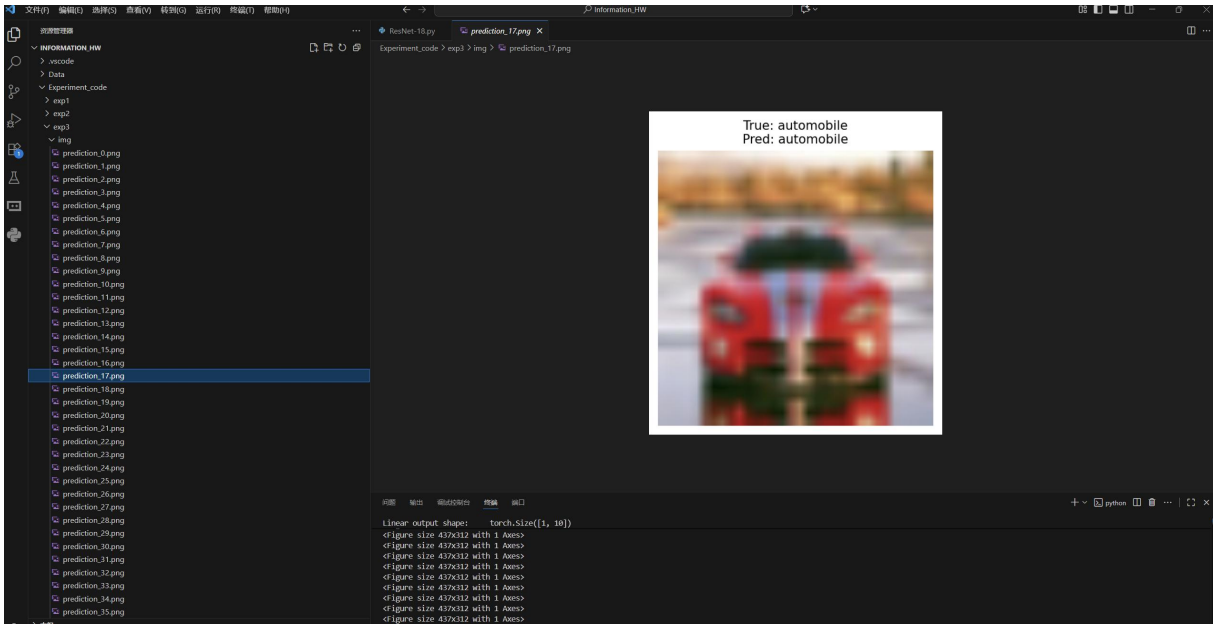
训练准确率约为 96%，测试准确率提升至约 76%–78%。

```
loss 0.099, train acc 0.971, test acc 0.765
444.9 examples/sec on cuda:0
```



四、附加实验

为了验证模型效果，实验中随机选取 50 张测试图像进行预测并保存结果图片，结果显示多数预测正确，少数样本因背景复杂而误判，总体性能稳定。



五、实验总结

1. 本实验实现了 ResNet-18 网络结构，并在 CIFAR-10 数据集上完成训练与测试。

2. 初始训练出现过拟合，表现为训练集精度高而测试集精度低。

3. 通过在优化器中添加 L2 正则化（weight decay），有效缓解过拟合现象，提高模型泛化能力。

4. 实验表明，正则化项能抑制参数过大带来的模型复杂度，从而获得更稳定的性能。

5. 本实验加深了我对 ResNet 结构与正则化方法的理解，也提升了对深度网络训练技巧的掌握。