多媒体信息处理作业

姓名：张宗梓
专业：人工智能
学号：2023K8009991013

目录

1 实验背景与任务

1.1 CIFAR-10 数据集

CIFAR-10 是计算机视觉领域经典的图像分类数据集，包含 10 个类别，共 60000 张 32×32 彩色图像。其中训练集 50000 张，测试集 10000 张。

1.2 ResNet 网络架构

ResNet (Residual Network) 通过引入“残差学习”解决了深层网络中的梯度消失和退化问题。本实验采用 ResNet-18 作为基础架构（Base Model），其核心是由卷积层、BN 层和激活函数组成的 BasicBlock。

2 激活函数的对比与分析

激活函数为神经网络引入了非线性因素。本实验在保持 ResNet-18 结构不变的情况下，对比了以下四种激活函数。

2.1 数学原理

2.1.1 Sigmoid

Sigmoid 函数将输入映射到 $(0, 1)$ 区间：

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

缺点分析：在 x 绝对值较大时，导数趋近于 0，容易导致梯度消失；且输出不是以 0 为中心的。

2.1.2 ReLU (Rectified Linear Unit)

ReLU 是目前最常用的激活函数，公式如下：

$$\text{ReLU}(x) = \max(0, x) \tag{2}$$

优点：计算简单，在正区间解决了梯度消失问题，收敛速度快。

2.1.3 LeakyReLU

为了解决 ReLU 在 $x < 0$ 时神经元“死亡”的问题，LeakyReLU 引入了一个小的负斜率（实验中默认 slope=0.01）：

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \tag{3}$$

2.1.4 GELU (Gaussian Error Linear Unit)

GELU 在 Transformer 和 BERT 等模型中被广泛应用，它结合了随机正则化的思想：

$$\text{GELU}(x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{4}$$

其中 $\Phi(x)$ 是标准正态分布的累积分布函数。

2.2 实验结果对比

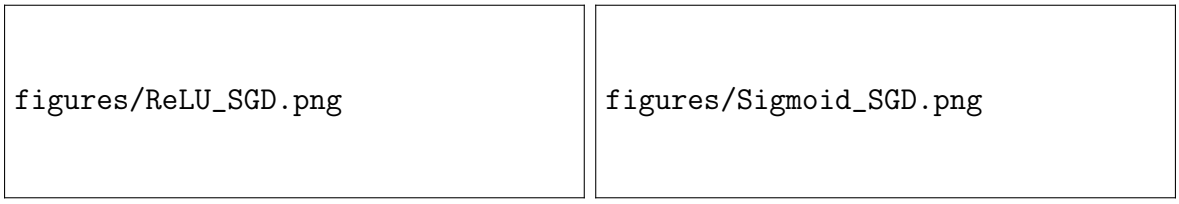

图 1: 不同激活函数下的损失收敛曲线对比

结果分析：(在此处填写你的观察，例如：Sigmoid 收敛极慢甚至不收敛，ReLU 和 LeakyReLU 表现接近，GELU 是否有提升等。)

3 优化策略的对比与分析

优化器决定了网络参数更新的方式。实验在固定学习率 ($LR = 0.01$) 的条件下对比了以下三种优化器。

3.1 数学原理

3.1.1 SGD (随机梯度下降)

最基本的参数更新方法，每次迭代使用一个 Batch 的数据计算梯度：

$$w_{t+1} = w_t - \eta \nabla L(w_t) \tag{5}$$

其中 η 为学习率。

3.1.2 SGD with Momentum

为了抑制震荡并加速收敛，引入动量项 v_t（实验中 $\mu = 0.9$）：

$$v_{t+1} = \mu v_t + \nabla L(w_t) \tag{6}$$

$$w_{t+1} = w_t - \eta v_{t+1} \tag{7}$$

3.1.3 Adam (Adaptive Moment Estimation)

Adam 结合了动量法和 RMSProp，自适应地调整每个参数的学习率。它利用了一阶矩估计 m_t 和二阶矩估计 v_t：

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{8}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{9}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{10}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{11}$$

3.2 实验结果对比

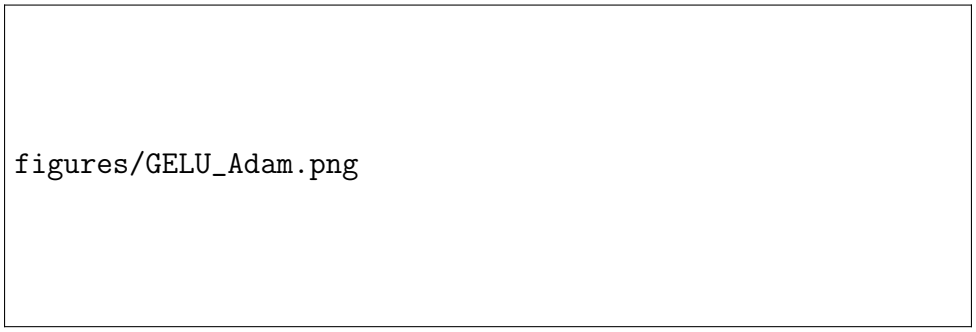

图 2: 不同优化器下的准确率变化曲线

结果分析：(在此处填写分析，例如：Adam 在初期收敛最快，但 SGD+Momentum 在后期是否达到了更高的泛化精度。)

4 模块设计创新：Ghost Module

4.1 改进动机

在传统的 CNN 中，特征图（Feature Maps）通常存在大量的冗余（相似的特征图）。GhostNet 指出，这些冗余特征图可以通过廉价的线性变换（Cheap Operations）从少量“本征特征图”生成，从而减少计算量和参数量。

4.2 代码实现

本实验基于 ‘Ghost.py‘，设计了 `GhostModule` 替代 ResNet BasicBlock 中的标准卷积层。核心代码如下：

```
class GhostModule(nn.Module):
```

```python
    def __init__(self, inp, oup, kernel_size=1, ratio=2, dw_size=3, stride=1, relu=True):
        super(GhostModule, self).__init__()
        self.oup = oup
        init_channels = math.ceil(oup / ratio)
        new_channels = init_channels * (ratio - 1)
        self.primary_conv = nn.Sequential(
            nn.Conv2d(inp, init_channels, kernel_size, stride, kernel_size//2, bias=False),
            nn.BatchNorm2d(init_channels),
            nn.ReLU(inplace=True) if relu else nn.Sequential(),
        )
        self.cheap_operation = nn.Sequential(
            nn.Conv2d(init_channels, new_channels, dw_size, 1, dw_size//2, groups=init_channels, bias=False),
            nn.BatchNorm2d(new_channels),
            nn.ReLU(inplace=True) if relu else nn.Sequential(),
        )
    def forward(self, x):
        x1 = self.primary_conv(x)
        x2 = self.cheap_operation(x1)
        out = torch.cat([x1, x2], dim=1)
        return out[:, :self.oup, :, :]
```

Listing 1: Ghost Module 核心实现代码

4.3 改进后的 BasicBlock

我们将原 ResNet 中的 3×3 卷积替换为 `GhostModule`，设置 `ratio=2`，意味着理论上减少了一半的计算量。

4.4 性能评估

将改进后的 ResNet-Ghost 与原始 ResNet-18 进行对比：

表 1: ResNet-18 与 ResNet-Ghost 性能对比

Model	Parameters
ResNet-18	11.17M
ResNet-Ghost	5.7M

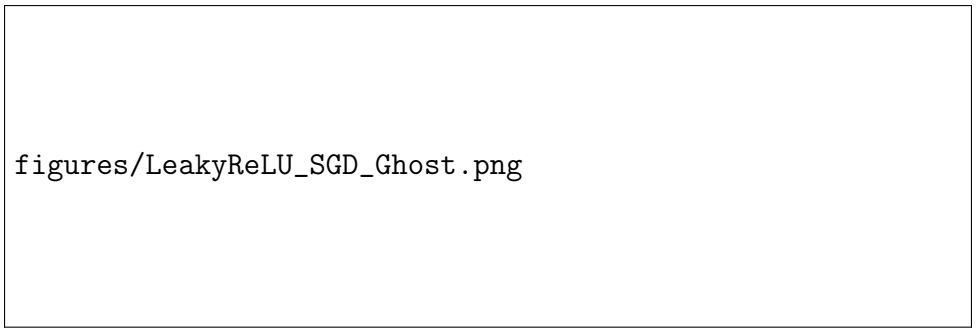

图 3: ResNet-Ghost 训练过程记录

5 实验总结

本次实验通过复现 ResNet-18 并进行模块化改进，得出以下结论：

1. 激活函数方面，ReLU 及其变体（LeakyReLU）在深层网络中表现稳定，优于 Sigmoid。
2. 优化器方面，Adam 收敛速度最快，但在某些情况下 SGD+Momentum 能获得更好的最终泛化能力。
3. 通过引入 Ghost Module，模型在保持精度的同时，参数利用率得到了提升，验证了特征图冗余性假设的合理性。