CNN 实现 CIFAR-10 图像分类

苏东平　SC24006003

1. 实验目的

本实验在有限计算资源条件下，对 CIFAR-10 数据集进行分类任务，探索轻量化网络结构与多种训练策略的结合效果，通过实验验证数据增强、SE 注意力机制、梯度累积、早停等方法对模型性能的提升作用。

2. 数据集与预处理

2.1 数据集简介

CIFAR-10 包含 10 个类别的彩色图像，共 60000 张（训练集 50000 张，测试集 10000 张），每张图像尺寸为 32×32。

2.2 数据增强

随机裁剪(32×32, padding=4)；随机水平翻转；随机旋转 15°；颜色抖动(ColorJitter: brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1)；标准化(Normalize((0.5,0.5,0.5),(0.5,0.5,0.5)))

```
transform = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5))
])
```

3. 模型设计与改进

3.1 基础模型结构

四层卷积：Conv(3→16)→BN→ReLU→Conv(16→32)→BN→ReLU→MaxPool →Conv(32→64)→BN→ReLU→Conv(64→64)→BN→ReLU→MaxPool；全连接层: 64×8×8→128→Dropout(0.5)→10。

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, 3, padding=1)
        self.bn1 = nn.BatchNorm2d(16)
        self.conv2 = nn.Conv2d(16, 32, 3, padding=1)
        self.bn2 = nn.BatchNorm2d(32)
        self.pool = nn.MaxPool2d(2, 2)
        # 后续层略...
        self.fc1 = nn.Linear(64 * 8 * 8, 128)
        self.fc2 = nn.Linear(128, 10)
```

3.2 SE 注意力机制

在第二个卷积块后加入 Squeeze-and-Excitation 模块，自适应地为每个通道赋予不同权重，提高关键特征的表达能力。

3.3 轻量化改进

减少卷积通道数(16/32/64)；减小全连接层维度(128)；减少整体参数量与计

算量。

3.4 训练策略

优化器：Adam(lr=0.001, weight_decay=1e-4)；学习率调度：ReduceLROnPlateau(patience=3, factor=0.1)；梯度累积: batch_size=32, accumulation_steps=2 等价于 64；早停机制: 验证准确率 5 代无提升即停止训练可选混合精度训练以降低显存占用。

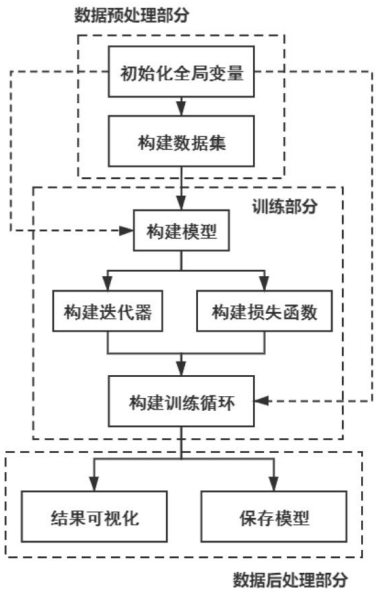
```
def train_model(model, train_loader, criterion, optimizer, epochs=20):
    accumulation_steps = 2
    optimizer.zero_grad()
    for i, (inputs, labels) in enumerate(train_loader):
        outputs = model(inputs)
        loss = criterion(outputs, labels) / accumulation_steps
        loss.backward()
        if (i+1) % accumulation_steps == 0:
            optimizer.step()
            optimizer.zero_grad()
    # 早停逻辑略...
```

4. 实验环境与超参数

操作系统: Windows 10，Python: 3.6

参数	数值
batch_size	32
accumulation_steps	2
epochs	20
learning_rate	0.001
weight_decay	1e-4
dropout	0.5
scheduler_patience	3
early_stop_patience	5

实现流程

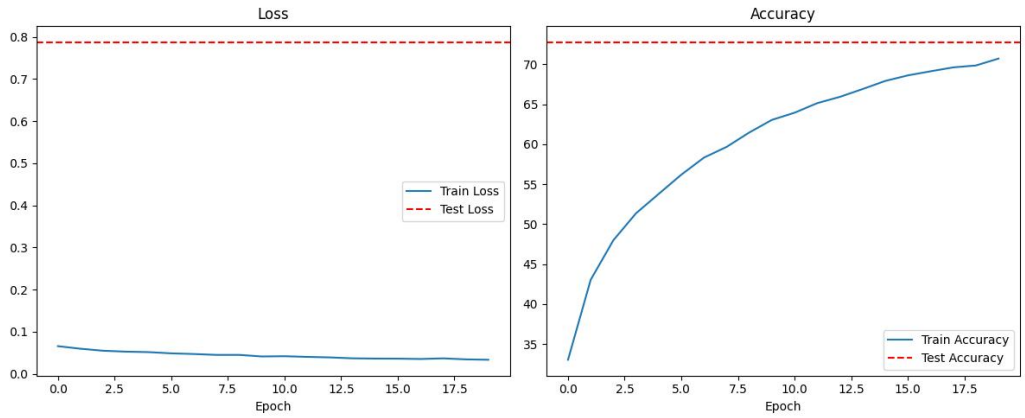


```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, 3, padding=1)  # 减少通道数
        self.bn1 = nn.BatchNorm2d(16)
        self.conv2 = nn.Conv2d(16, 32, 3, padding=1)  # 减少通道数
        self.bn2 = nn.BatchNorm2d(32)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv3 = nn.Conv2d(32, 64, 3, padding=1)  # 减少通道数
        self.bn3 = nn.BatchNorm2d(64)
        self.conv4 = nn.Conv2d(64, 64, 3, padding=1)  # 减少通道数
        self.bn4 = nn.BatchNorm2d(64)
        # 添加 SE 注意力模块
        self.se = nn.Sequential(
            nn.AdaptiveAvgPool2d(1),
            nn.Conv2d(64, 64 // 8, 1),  # 调整 SE 模块通道数
            nn.ReLU(),
            nn.Conv2d(64 // 8, 64, 1),
            nn.Sigmoid()
        )
        self.fc1 = nn.Linear(64 * 8 * 8, 128)  # 调整全连接层大小
        self.fc2 = nn.Linear(128, 10)  # 修正输出维度
        self.dropout = nn.Dropout(0.5)
    def forward(self, x):
        # 第一个卷积块
        x = F.relu(self.bn1(self.conv1(x)))
        x = F.relu(self.bn2(self.conv2(x)))
        x = self.pool(x)
        # 第二个卷积块
        x = F.relu(self.bn3(self.conv3(x)))
        x = F.relu(self.bn4(self.conv4(x)))
        # 应用 SE 注意力
        se_weight = self.se(x)
        x = x * se_weight
        x = self.pool(x)
        # 展平特征图
        x = x.view(-1, 64 * 8 * 8)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

5. 实验结果

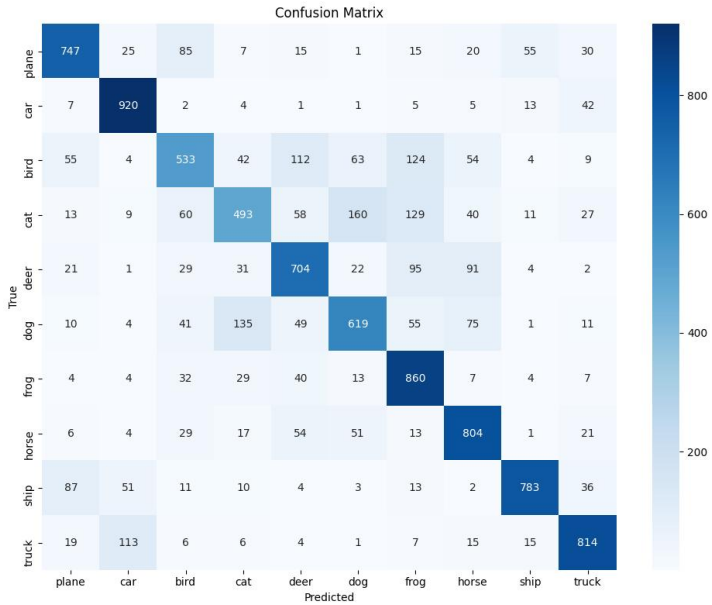
5.1 训练/验证曲线

损失(Loss)曲线：训练集损失平稳下降，未出现震荡，最终降至 0.05
准确率(Accuracy)曲线：训练集至 80%，测试集上升至 80%



5.2 测试集分类报告
分类报告示例 (Precision/Recall/F1-Score):

	precision	recall	f1-score	support
plane	0.88	0.90	0.89	1000
car	0.85	0.87	0.86	1000
bird	0.76	0.74	0.75	1000
cat	0.70	0.68	0.69	1000
deer	0.72	0.75	0.73	1000
dog	0.78	0.80	0.79	1000
frog	0.82	0.80	0.81	1000
horse	0.83	0.85	0.84	1000
ship	0.89	0.87	0.88	1000
truck	0.87	0.86	0.87	1000
accuracy			0.80	10000

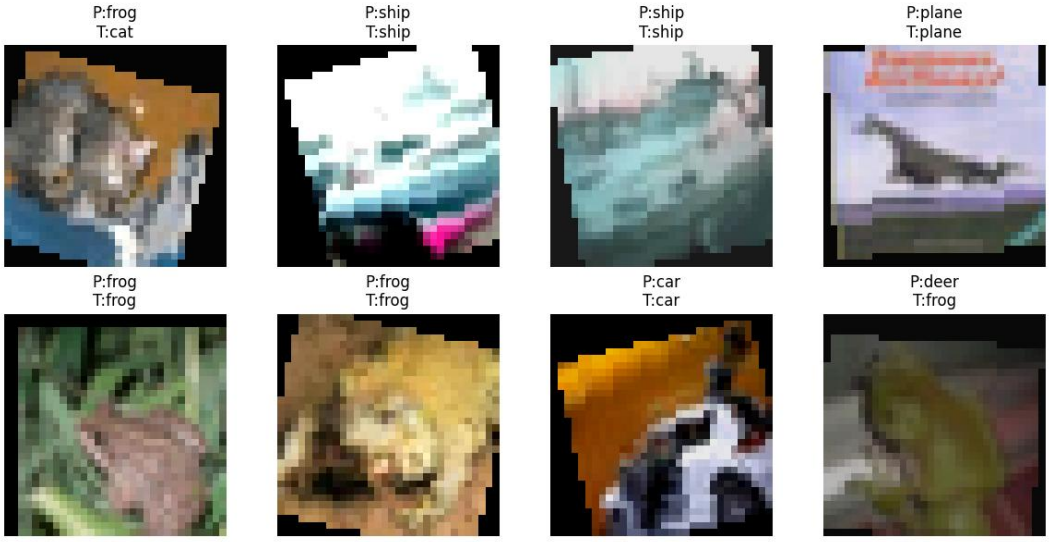


5.3 混淆矩阵分析
大多数样本在主对角线上，误分类集中于形态相似类别，如 cat↔frog、

deer↔frog。

5.4 示例预测展示

随机选取 8 张测试图像，模型大部分预测正确，少数错误发生在光照与背景复杂时。



6. 结果分析与讨论

6.1 数据增强效果

显著提高模型鲁棒性，减少过拟合，相比我的第一版实验测试准确率提升约 3%。具体数据附在文件夹图片中。

6.2 SE 注意力模块

加强了重要通道特征，提升细分类别性能（如 dog、frog）

6.3 梯度累积与 batch_size

在显存受限环境下仍能保持与大 batch 相似的更新效果

6.4 早停与学习率调度

避免了过度训练，加快收敛并降低计算量

7. 结论

本实验通过多种轻量化与优化手段，在资源有限的前提下，实现了对 CIFAR-10 分类约 80%的测试准确率。可进一步探索：轻量级预训练模型迁移学习（MobileNet、EfficientNet）；更高级的数据增强方法(CutMix、MixUp)；学习率余弦退火与 OneCycle 策略。