实验一 手写数字识别

1. 实验要求

本实验旨在构建一个基于卷积神经网络(CNN)的手写数字识别系统，实现对MNIST数据集中0-9数字的自动分类，要求：

- 搭建 PyTorch（或其他框架）环境
- 构建一个规范的卷积神经网络组织结构
- 在 MNIST 手写数字数据集上进行训练和评估，实现测试集准确率达 到 98%及以上

---

2. MNIST数据集介绍

数据规模：

- **训练集**：60,000张手写数字图像
- **测试集**：10,000张手写数字图像
- **图像尺寸**：28×28像素灰度图像
- **类别数量**：10个类别(数字0-9)

数据特点：

- **标准化格式**：所有图像已居中并标准化为28×28尺寸
- **灰度图像**：单通道图像，像素值范围0-255

---

3. 实验设计与实现

3.1 网络结构设计

3.1.1 整体架构

本项目设计了一个两层卷积神经网络，整体架构如下：

```
CNN模型架构 (约85K参数)
├── 输入层
│   └── 28×28×1 灰度图像
├── 第一卷积块
│   ├── Conv2d: 1→16通道，5×5卷积核，padding=2
│   ├── ReLU激活函数
│   └── MaxPool2d: 2×2池化 → 14×14×16
├── 第二卷积块
│   ├── Conv2d: 16→32通道，5×5卷积核，padding=2
│   ├── ReLU激活函数
│   └── MaxPool2d: 2×2池化 → 7×7×32
├── 展平层
│   └── 32×7×7 = 1568维特征向量
└── 全连接层
    └── Linear: 1568 → 10输出(softmax分类)
```

3.1.2 核心代码实现

CNN模型定义：

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        # 第一层卷积+激活+池化
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=1,        # 输入通道数
                out_channels=16,      # 输出通道数
                kernel_size=5,        # 卷积核大小
                stride=1,             # 步长
                padding=2             # 填充数
            ),
            nn.ReLU(),                # 激活函数
            nn.MaxPool2d(2)           # 池化核大小 2x2
        )

        # 第二层卷积+激活+池化
        self.conv2 = nn.Sequential(
            nn.Conv2d(16, 32, 5, 1, 2),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )

        # 全连接层
        self.out = nn.Linear(32 * 7 * 7, 10)  # 输出10个类别(0-9)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(x.size(0), -1)
        output = self.out(x)
        return output
```

3.2 损失函数设计

3.2.1 交叉熵损失函数

采用多分类交叉熵损失函数

```
criterion = nn.CrossEntropyLoss()
loss = criterion(output, target)
```

3.3 优化器设计

3.3.1 Adam优化器配置

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

3.4 训练策略设计

3.4.1 训练流程

```python
def train_epoch(self):
    self.model.train()
    for batch_idx, (data, target) in enumerate(self.train_loader):
        data, target = data.to(self.device), target.to(self.device)

        # 梯度清零
        self.optimizer.zero_grad()

        # 前向传播
        output = self.model(data)
        loss = self.criterion(output, target)

        # 反向传播
        loss.backward()
        self.optimizer.step()
```

3.4.2 训练配置

批处理设置：

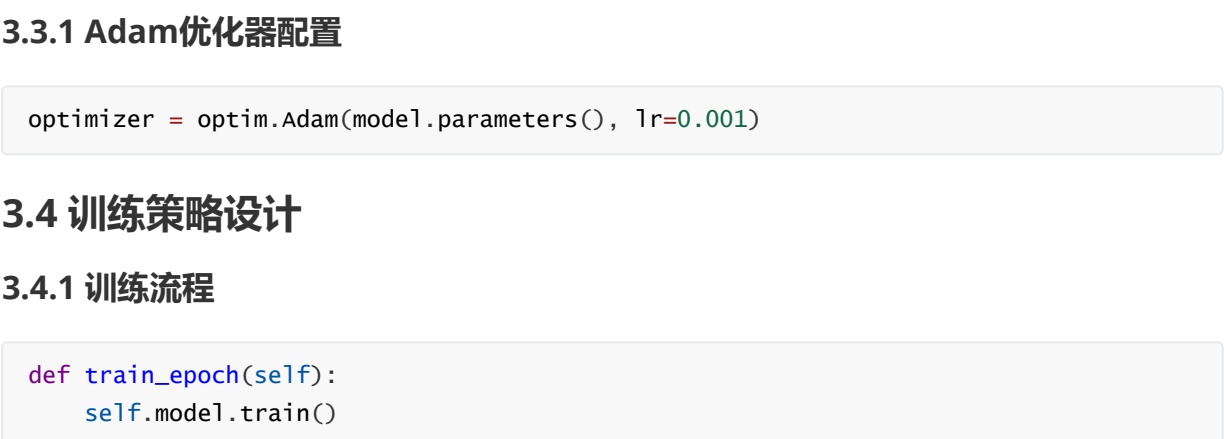
- **批大小**：64
- **并行加载**：2个工作进程

训练轮数：

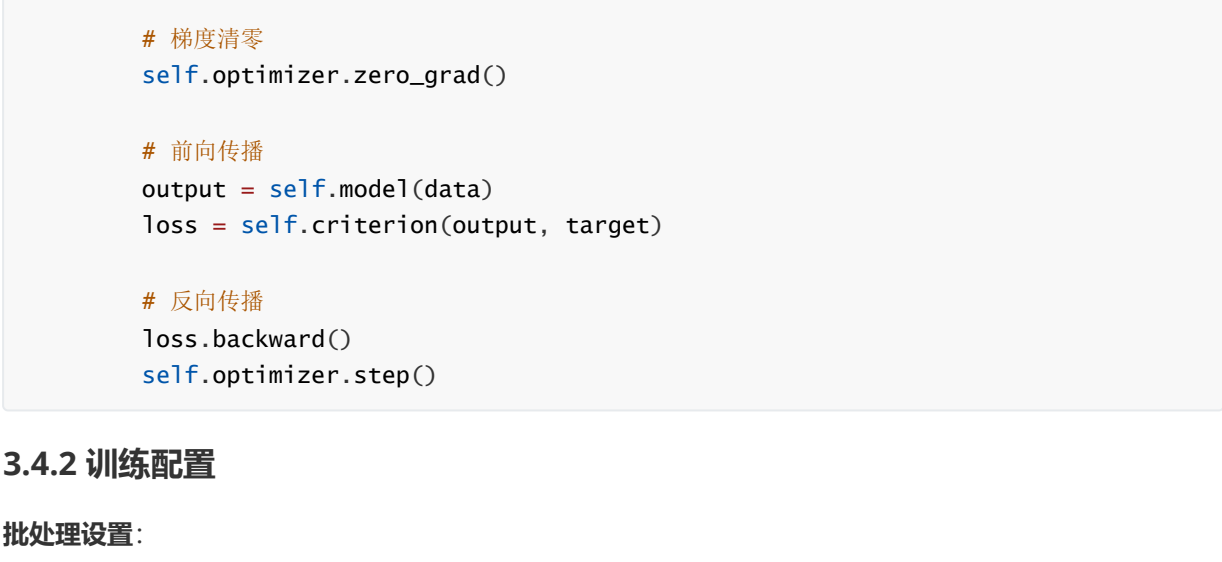
- **总轮数**：15个epoch
- **早停机制**：保存最佳模型，防止性能退化

---

4. 实验结果与分析



如上图所示，训练集准确率和测试集准确率在训练过程中快速上升，在14个epoch后，训练集准确率约为99.5%，测试集准确率约为99.2%，满足实验要求。



如图所示，模型能够精准的识别出测试集中的手写数字，具有较好的泛化能力。