实验一：基于CNN的手写数字识别实验报告

概述

任务描述

本实验的目标是实现一个卷积神经网络（CNN），用于对MNIST手写数字数据集进行分类。该数据集包含0-9共10类手写数字图像， 要求模型能够准确识别输入图像对应的数字类别。

数据集

- **MNIST数据集**：包含60,000张训练图像和10,000张测试图像，每张图像为28×28像素的灰度图。
- **预处理**：图像被归一化为 `[0,1]` 范围，并通过 `Reshape` 操作调整为 `(1,28,28)` 的张量格式以适应卷积层输入。

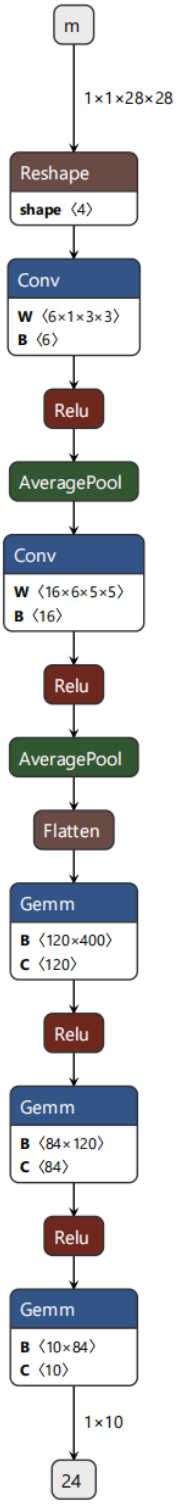
解决方案

设计了一个类似LeNet的卷积神经网络，包含卷积层、激活函数、池化层和全连接层，使用交叉熵损失函数和随机梯度下降（SGD） 优化器进行训练。

解决方案

网络结构设计

网络结构如下图所示：


m
1×1×28×28
Reshape
shape 〈4〉
Conv
W 〈6×1×3×3〉
B 〈6〉
Relu
AveragePool
Conv
W 〈16×6×5×5〉
B 〈16〉
Relu
AveragePool
Flatten
Gemm
B 〈120×400〉
C 〈120〉
Relu
Gemm
B 〈84×120〉
C 〈84〉
Relu
Gemm
B 〈10×84〉
C 〈10〉
1×10
24

具体层设计如下：

```python
class ZyhNet(nn.Module):
    def __init__(self):
        super(ZyhNet, self).__init__()
        # 卷积部分
        self.CNN1 = nn.Sequential(
            nn.Conv2d(1, 6, kernel_size=3, stride=1, padding=1),  # 输出尺寸: 6×28×28
            nn.ReLU(),
            nn.AvgPool2d(kernel_size=2, stride=2)                 # 输出尺寸: 6×14×14
        )
        self.CNN2 = nn.Sequential(
            nn.Conv2d(6, 16, kernel_size=5, stride=1),             # 输出尺寸: 16×10×10
            nn.ReLU(),
            nn.AvgPool2d(kernel_size=2, stride=2)                 # 输出尺寸: 16×5×5
        )
        # 全连接部分
        self.FC1 = nn.Sequential(
            nn.Flatten(),
            nn.Linear(400, 120),    # 16×5×5=400 → 120
            nn.ReLU(),
            nn.Linear(120, 84),
            nn.ReLU(),
            nn.Linear(84, 10)       # 输出10类概率
        )
```

关键参数

- **卷积层1**：6个3×3卷积核，输入通道1，输出通道6。
- **池化层**：平均池化（2×2窗口，步长2）。
- **卷积层2**：16个5×5卷积核，输入通道6，输出通道16。
- **全连接层**：400→120→84→10，每层后接ReLU激活。

损失函数与优化器

- **损失函数**：交叉熵损失（`nn.CrossEntropyLoss`）。
- **优化器**：随机梯度下降（SGD），学习率设为0.01。

```python
loss_fun = nn.CrossEntropyLoss()
optimizer = optim.SGD(net_in.parameters(), lr=lr_in)
```

创新点

1. **平均池化替代最大池化**：使用`AvgPool2d`减少特征图细节丢失。
2. **Xavier参数初始化**：对卷积层和全连接层的权重进行Xavier均匀初始化，加速收敛。

```python
def init(m):
    if isinstance(m, (nn.Linear, nn.Conv2d)):
        nn.init.xavier_uniform_(m.weight)
net.apply(init)
```

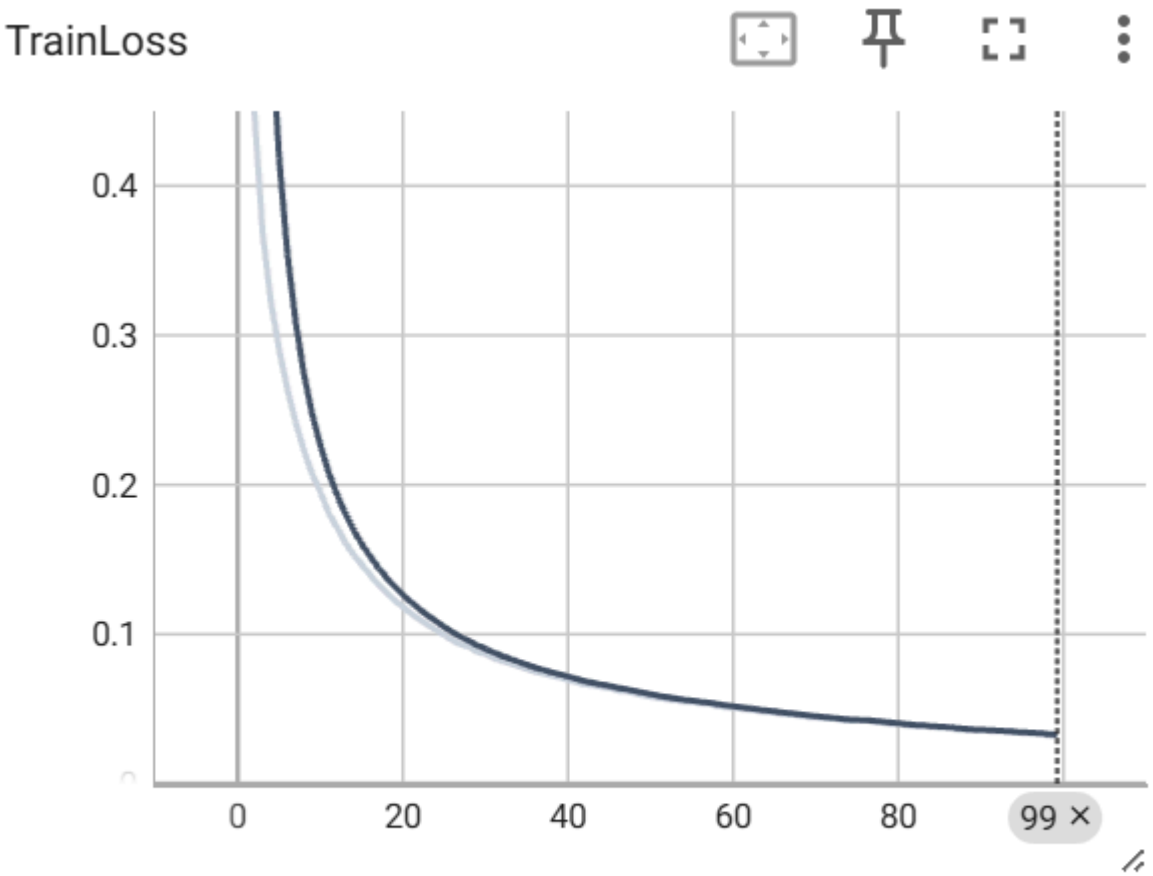
实验分析

训练配置

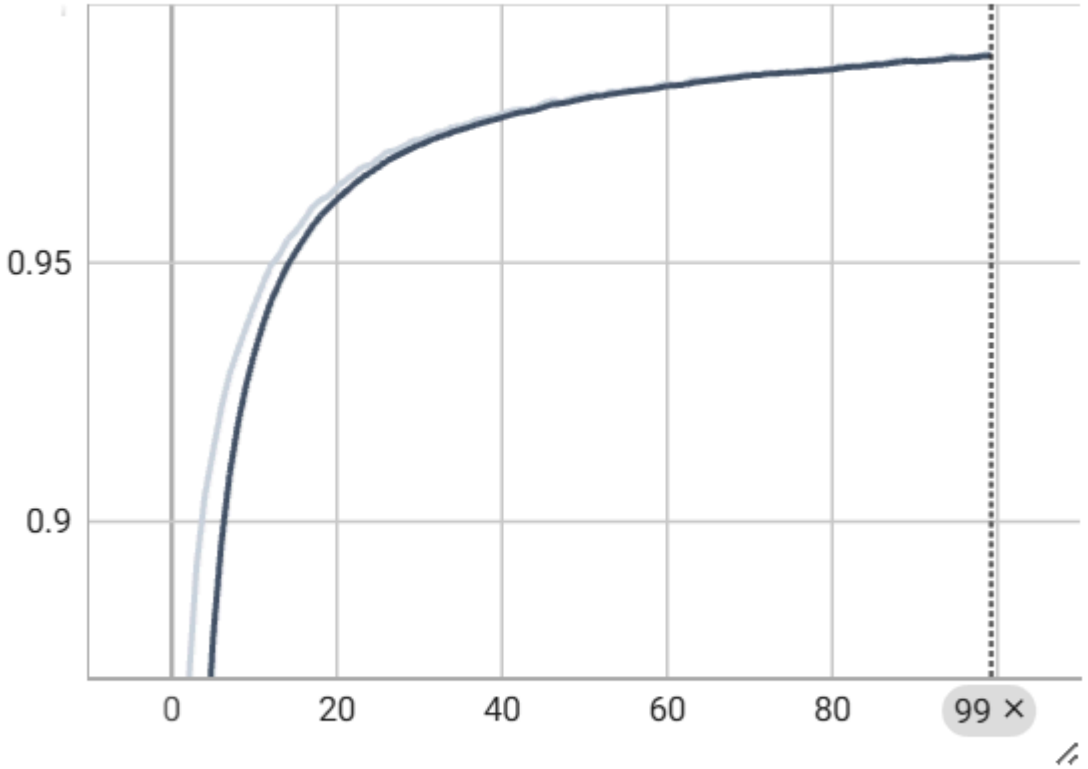
- **设备**：优先使用GPU（CUDA），否则使用CPU。
- **超参数**：批量大小256，训练轮数100，学习率0.01。
- **监控工具**：TensorBoard记录训练损失和准确率。

实验结果

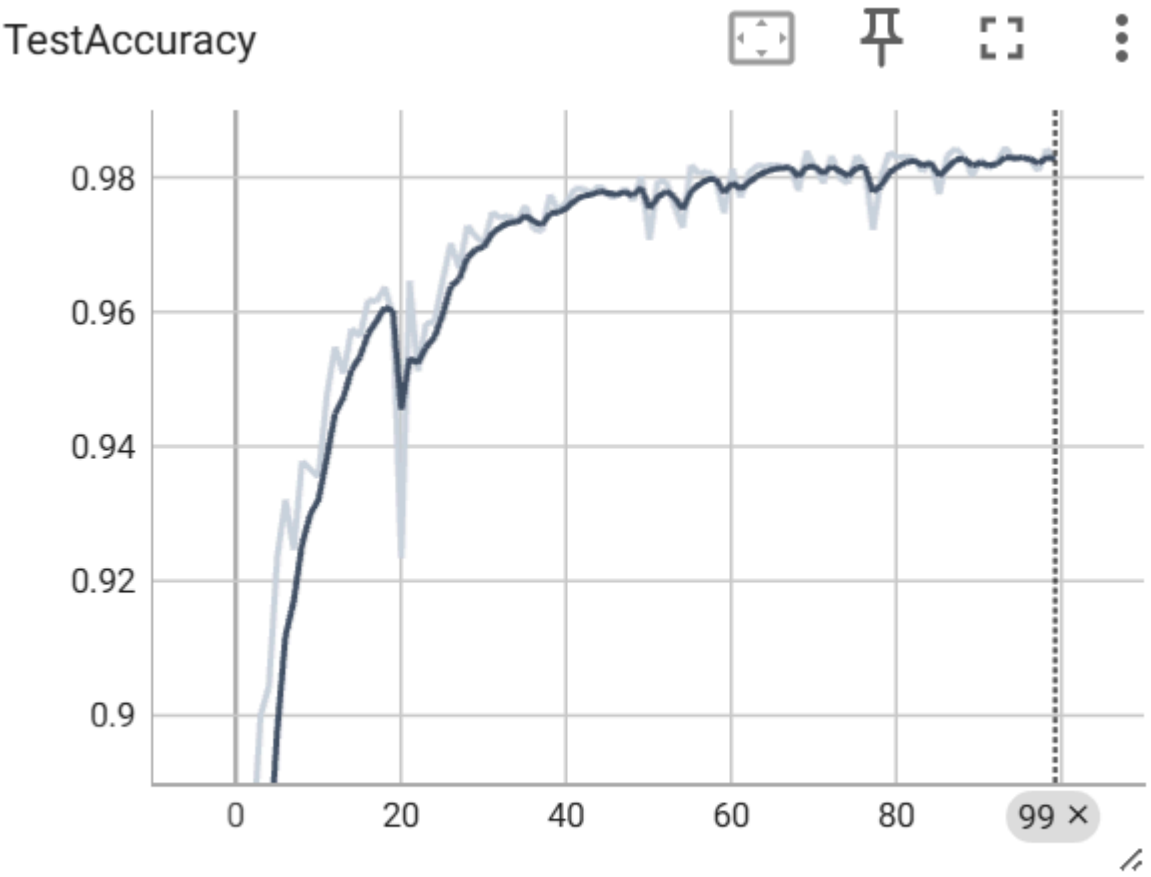
- **训练损失**：最终训练损失降至约0.032。
- **测试准确率**：模型在测试集上达到98%以上的分类准确率（test acc: 0.983）。
- **训练过程可视化**：通过TensorBoard可观察损失和准确率随训练轮数的变化趋势。



TrainAccuracy



Run ↑	Smoothed	Value	Step	Relative
.	0.9901	0.9904	99	51.78 min



Run ↑	Smoothed	Value	Step	Relative
.	0.9828	0.9826	99	51.78 min

总结

1. **成果**：设计的CNN模型在MNIST数据集上表现优异，测试准确率超过98%，验证了网络结构的有效性。
2. **改进方向**：
   - 尝试使用最大池化（`MaxPool2d`）对比性能差异。
   - 引入数据增强（如旋转、平移）提升模型鲁棒性。
   - 调整学习率调度策略（如余弦退火）进一步优化收敛速度。
3. **扩展应用**：本模型可迁移至其他图像分类任务，通过调整输入尺寸和输出类别数适配不同场景。