# 实验报告：LeNet-5 在 MNIST 数据集上的训练与测试

## 1. 实验目的

实现并测试 LeNet-5 网络在手写数字识别任务上的性能。

## **2. 实验环境**

编程语言：Python 3.8

深度学习框架：TensorFlow 2.4

**3. LeNet-5 网络结构实现**

LeNet-5 网络由两个卷积层、两个池化层和一个全连接层组成，最后是一个 softmax 输出层。

## **4. 实验代码**

# 导入PyTorch库  
import torch  
import torch.nn as nn  
import torch.optim as optim  
import torchvision  
import torchvision.transforms as transforms  
from torch.utils.data import DataLoader  
  
# 定义LeNet-5架构的神经网络类  
class LeNet5(nn.Module):  
 def \_\_init\_\_(self):  
 super(LeNet5, self).\_\_init\_\_()  
 # 第一卷积层：输入1通道（灰度图像），输出6通道，卷积核大小为5x5  
 self.conv1 = nn.Conv2d(1, 6, kernel\_size=5)  
 # 第一池化层：最大池化，池化窗口大小为2x2，步幅为2  
 self.pool1 = nn.MaxPool2d(kernel\_size=2, stride=2)  
 # 第二卷积层：输入6通道，输出16通道，卷积核大小为5x5  
 self.conv2 = nn.Conv2d(6, 16, kernel\_size=5)  
 # 第二池化层：最大池化，池化窗口大小为2x2，步幅为2  
 self.pool2 = nn.MaxPool2d(kernel\_size=2, stride=2)  
 # 第一个全连接层：输入维度是16\*4\*4，输出维度是120  
 self.fc1 = nn.Linear(16 \* 4 \* 4, 120)  
 # 第二个全连接层：输入维度是120，输出维度是84  
 self.fc2 = nn.Linear(120, 84)  
 # 第三个全连接层：输入维度是84，输出维度是10，对应10个类别  
 self.fc3 = nn.Linear(84, 10)  
  
 def forward(self, x):  
 # 前向传播函数定义网络的数据流向  
 x = self.pool1(torch.relu(self.conv1(x)))  
 x = self.pool2(torch.relu(self.conv2(x)))  
 x = x.view(-1, 16 \* 4 \* 4)  
 x = torch.relu(self.fc1(x))  
 x = torch.relu(self.fc2(x))  
 x = self.fc3(x)  
 return x  
  
# 定义数据变换和加载MNIST数据集  
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))])  
  
# 训练数据集  
train\_dataset = torchvision.datasets.MNIST(root='./data', train=True, transform=transform, download=True)  
train\_loader = DataLoader(train\_dataset, batch\_size=64, shuffle=True)  
  
# 测试数据集  
test\_dataset = torchvision.datasets.MNIST(root='./data', train=False, transform=transform, download=True)  
test\_loader = DataLoader(test\_dataset, batch\_size=64, shuffle=False)  
  
# 初始化LeNet-5模型以及定义损失函数和优化器  
net = LeNet5()  
criterion = nn.CrossEntropyLoss() # 交叉熵损失函数，用于分类问题  
optimizer = optim.Adam(net.parameters(), lr=0.001) # Adam优化器，学习率为0.001  
  
# 训练循环  
for epoch in range(5): # 可以根据需要调整训练的轮数  
 running\_loss = 0.0  
 for i, data in enumerate(train\_loader, 0):  
 inputs, labels = data  
 optimizer.zero\_grad() # 清零梯度  
  
 outputs = net(inputs) # 前向传播  
 loss = criterion(outputs, labels) # 计算损失  
 loss.backward() # 反向传播，计算梯度  
 optimizer.step() # 更新权重  
  
 running\_loss += loss.item()  
 print(f"Epoch {epoch + 1}, Loss: {running\_loss / len(train\_loader)}")  
  
print("Finished Training")  
  
# 测试模型  
correct = 0  
total = 0  
with torch.no\_grad():  
 for data in test\_loader:  
 inputs, labels = data  
 outputs = net(inputs) # 前向传播  
 \_, predicted = torch.max(outputs.data, 1) # 找到最大概率的类别  
 total += labels.size(0)  
 correct += (predicted == labels).sum().item()  
  
accuracy = 100 \* correct / total  
print(f"Accuracy on the test set: {accuracy}%")

## **5. 实验结果**

## 

## **6. 结果分析**

模型有很高的准确率，表明模型具有很好的泛化能力。

## **7. 实验总结**

LeNet-5 模型虽然结构简单，但在 MNIST 数据集上表现出色，证明了其在手写数字识别任务上的有效性。模型的训练和测试过程顺利，没有遇到显著的问题。