同濟大學
TONGJI UNIVERSITY

《人工智能导论》实验报告

实验报告名称 基于卷积神经网络的 MNIST 手写数字识别

姓 名 林继申

学 号 2250758

学 院 软件学院

专 业 软件工程

教 师 曾 进

完成日期：2024 年 3 月 13 日

摘 要

本实验使用卷积神经网络（CNN）对MNIST手写数字数据集进行训练和测试。通过简单的预处理将图像数据转换为张量，并进行归一化处理。CNN模型包括两个卷积层和两个全连接层，使用了交叉熵损失函数和Adam优化器。训练过程中观察到模型表现出良好的收敛性和稳定性，平均训练损失逐渐减小。在测试集上获得了99.19%的准确率和较小的平均损失，验证了模型在未见数据上的泛化能力和预测准确性。综合考虑训练和测试结果，本实验证明了所构建的CNN模型在手写数字识别任务上表现出色，具有实际应用的潜力和稳定性。

关键词：卷积神经网络，MNIST，手写数字识别，深度学习

1. 实验介绍

本实验基于卷积神经网络（CNN）对MNIST手写数字数据集进行训练和测试。CNN是一种专门用于处理具有网格结构数据（如图像）的深度学习模型，在图像识别任务中表现出色。本实验将使用PyTorch实现一个简单的CNN模型。

2. 数据集介绍

MNIST（Modified National Institute of Standards and Technology）是一个常用的手写数字识别数据集，由美国国家标准与技术研究院（NIST）创建并进行了修改。该数据集包含了大量的手写数字图像，是机器学习领域中最经典的数据集之一。

MNIST数据集包含了60000个训练样本和10000个测试样本。每个样本都是一个28x28像素的灰度图像，表示了0到9之间的单个手写数字。每个图像都对应一个相应的标签，标识了图像所表示的数字。标签是从0到9的整数。

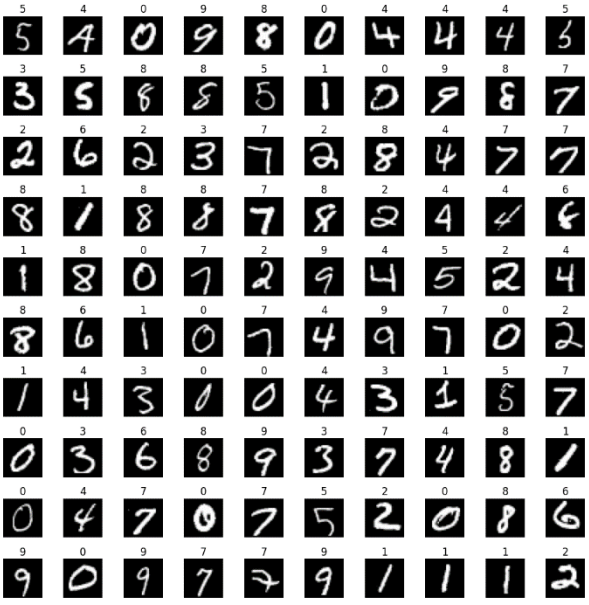

图2.1 部分MNIST手写数字数据集

3. 实验原理

3.1.CNN（卷积神经网络）

卷积神经网络（Convolutional Neural Network，CNN）是一种深度学习模型，专门用于处理具有网格结构的数据，如图像。CNN在图像识别任务中表现出色，因其能够自动学习图像中的特征而广受欢迎。下面

是本实验中使用的CNN结构，共有421642个参数。

3.1.1. 卷积层（Convolutional Layer）

卷积层是CNN的核心组件之一。它通过在图像上滑动卷积核（filter）来提取图像的特征。每个卷积核都包含一组权重，可以学习不同的特征模式。本实验中，我们定义了两个卷积层：

第一个卷积层：输入通道为1（灰度图像），输出通道为32，卷积核大小为3x3。

第二个卷积层：输入通道为32，输出通道为64，卷积核大小为3x3。

3.1.2. 池化层（Pooling Layer）

池化层用于减少卷积层输出的空间维度，同时保留重要信息。在本实验中，我们采用最大池化（Max Pooling）操作，将每个2x2的区域中的最大值作为输出。

3.1.3. 全连接层（Fully Connected Layer）

全连接层用于将卷积层输出的特征图转换为分类结果。在本实验中，我们定义了两个全连接层：

第一个全连接层：输入特征维度为64x7x7（64个通道，每个通道大小为7x7），输出维度为128。

第二个全连接层：输入维度为128，输出维度为10（对应10个数字类别）。

3.2.损失函数与优化器

在训练CNN模型时，需要定义损失函数来衡量模型输出与真实标签之间的差距，并使用优化器来更新模型参数以最小化损失函数。在本实验中，我们采用以下损失函数和优化器：

损失函数：交叉熵损失函数（Cross Entropy Loss），用于多分类问题的损失计算。

优化器：Adam优化器，一种自适应学习率的优化算法，用于更新模型参数。

```
class CNNModel(nn.Module):
    def __init__(self):
        super(CNNModel, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(64 * 7 * 7, 128)
        self.fc2 = nn.Linear(128, 10)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 64 * 7 * 7)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return F.log_softmax(x, dim=1)
model = CNNModel()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001, weight_decay=1e-5)
```

代码3.1 卷积神经网络模型定义代码段

4. 实验设计

4.1.数据预处理

在实验中，我们对MNIST数据集进行了简单的预处理。首先，我们将图像数据转换为张量，并进行了归一化处理，使像素值在[0, 1]范围内。这有助于加速模型收敛并提高模型性能。

4.2.CNN模型训练

我们使用了包含两个卷积层和两个全连接层的CNN模型。模型的参数初始化采用了默认的方法。在训练过程中，我们采用了批量梯度下降（Batch Gradient Descent）方法，每次从训练集中随机采样一批数据进行模型参数更新。同时，我们使用了Adam优化器来自适应地调整学习率。

```
num_epochs = 20
epoch_losses = []
for epoch in range(num_epochs):
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if i % 100 == 99:
            average_loss = running_loss / 100
            print('[Epoch: %d, Mini-batch: %d] Average Loss: %.5f' % (epoch + 1, i
+ 1, average_loss))
            running_loss = 0.0
    epoch_loss = running_loss / len(train_loader)
    epoch_losses.append(epoch_loss)
    print('Epoch %d completed. Average Loss: %.5f' % (epoch + 1, epoch_loss))
```

代码4.1 卷积神经网络模型训练代码段

4.3. 实验评估

在训练过程中，我们通过监测训练损失的变化来评估模型的训练情况，并在每个epoch结束时计算了平均训练损失。此外，我们在测试集上评估了模型的性能，计算了模型在测试集上的准确率以及平均测试损失。我们还通过绘制损失随epoch变化的曲线和展示部分测试集样本的预测结果，来直观地展示模型的训练情况和性能表现。

5. 实验结果

5.1. 训练过程分析

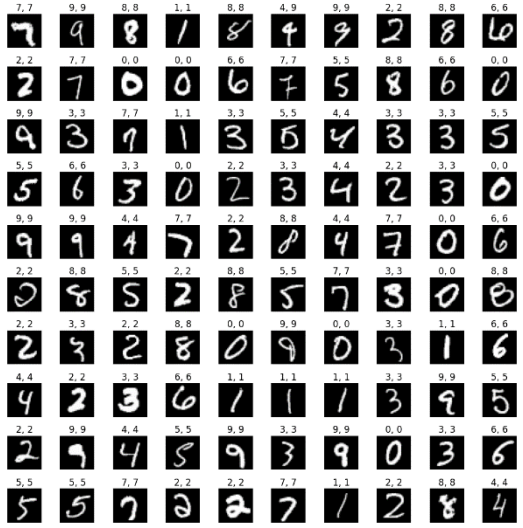

图5.1 部分手写数字数据集标签与预测结果

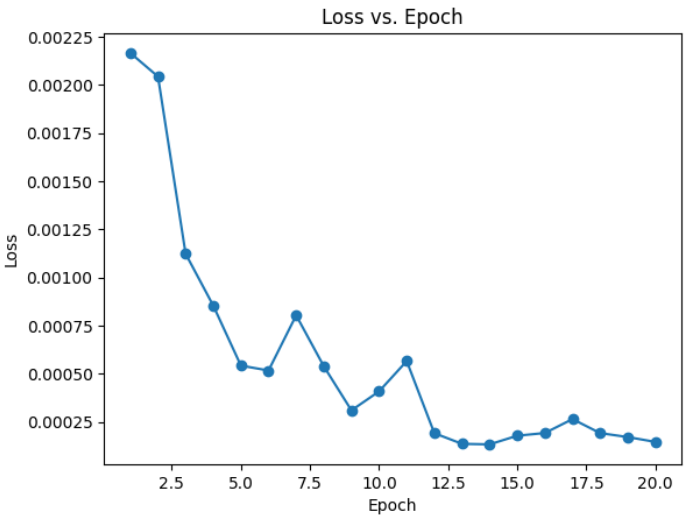

图5.2 CNN模型训练损失随epoch变化曲线

我们观察到模型在训练过程中表现出了良好的收敛性和稳定性。随着训练的进行，平均训练损失逐渐减小，这表明模型对训练数据的拟合效果逐渐提高。在每个epoch内，小批量训练损失也呈现逐渐减小的趋势。这种逐步减小的损失趋势证明了模型在不断优化参数以适应训练数据的过程中的有效性。值得注意的是，随着训练的进行，损失的下降速度逐渐减缓，这表明模型在后续训练中对数据的拟合程度逐渐增加，进一步证实了模型的收敛性。

我们观察到在epoch在6到12之间时，训练过程中的平均损失出现了小范围的波动。这种现象可能是由于多种因素相互作用所致。学习率的调整可能导致模型参数更新速度的变化，引起损失的波动。数据批量的影响也是一个重要因素，特定批量数据的特性可能会对模型训练产生影响。此外，模型复杂性以及随机性也可能在一定程度上影响了损失的变化。这种小范围的损失波动通常被认为是正常的，只要整体的训练损失仍然在下降趋势中。

5.2.测试结果分析

在测试集上，我们评估了模型的性能，获得了99.19%的准确率。这说明模型在未见过的数据上具有很好的泛化能力，能够有效地识别手写数字。此外，我们计算了模型在测试集上的平均损失，结果为0.00058，这表明模型的预测结果与真实标签之间的差距很小。综合考虑准确率和损失值，我们可以得出结论，模型在测试集上表现出了出色的性能，验证了其在实际应用中的有效性和稳定性。

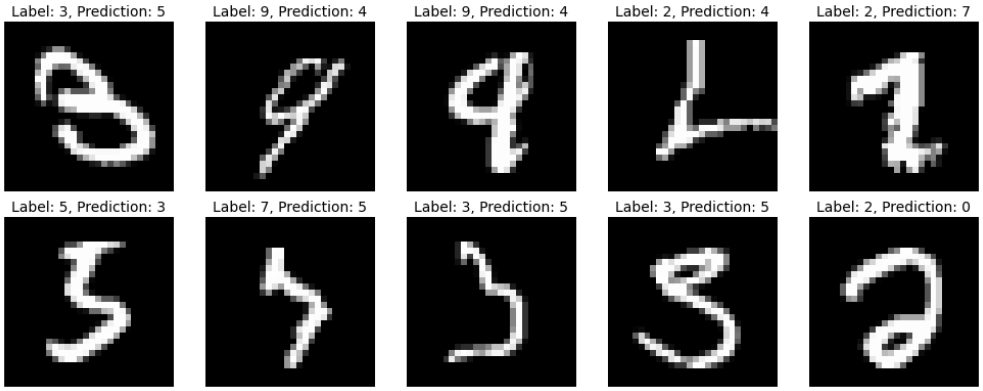

图5.3 部分手写数字数据集标签与错误的预测结果

需要指出的是，许多被模型错误预测的例子，即使对人类来说也可能难以准确识别。因此，我们的模型在实践中的表现非常出色。

6. 总结

本实验采用了CNN模型对MNIST手写数字数据集进行训练和测试。通过对模型训练过程的分析，观察到模型在训练集上表现出良好的收敛性和稳定性，随着训练的进行，平均损失逐渐减小，证明了模型在不断优化参数以适应数据的过程中的有效性。在测试集上评估模型性能，获得了99.19%的准确率和较小的平均损失，验证了模型在未见数据上的泛化能力和预测准确性。综合考虑训练和测试结果，表明所构建的CNN模型在手写数字识别任务上表现出色，具有实际应用的潜力和稳定性，为解决图像分类问题提供了可靠的解决方案。

通过本实验，我对深度学习和卷积神经网络有了更深入的理解。深度学习模型的训练过程需要耐心和细致的调整，同时了解CNN的结构和原理能够更好地指导模型设计和调优。在实验中，我意识到CNN在处理图像任务中的强大能力，以及通过对数据进行适当处理和优化，模型能够在未见过的数据上取得出色的泛化性能。这次实验让我更加确信深度学习技术的潜力和广泛应用的前景，也激励我继续深入学习和探索。

7. 附录

MNIST_Handwritten_Digit_Recognition_Based_on_CNN.ipynb：MNIST手写数字识别代码及运行结果。
utils.py：本实验所需的工具函数，被导入到主代码文件中，用于MNIST手写数字数据集的可视化。
MNIST文件夹：MNIST手写数字数据集，共四个文件，分别为训练数据集和测试数据集的图像与标签。