广州華商學院
GUANGZHOU HUASHANG COLLEGE

《专业综合实践Ⅱ》课程考核

题　　目：	基于深度学习的图像识别系统：mnist 手写数据集
学　　院：	人工智能学院
专　　业：	数据科学与大数据技术
年级班别：	21 本 1 班
姓　　名：	黄宇彬
指导教师：	刘盛

提交日期：2024 年 12 月

目录

一、 项目背景与目的

1.1 背景：

1.1.1 图像识别在人工智能领域的重要性：

图像识别作为人工智能的一个关键领域，融合了计算机视觉、机器学习、模式识别等多个学科的研究成果。随着深度学习技术的突破性进展，图像识别技术实现了显著的飞跃，并在众多行业中展现出巨大的应用潜力。这项技术赋予了计算机类似人类的图像识别和理解能力，对于自动化技术、安全监控系统、医疗诊断过程、自动驾驶技术等都具有深远的影响。

1.1.2 MNIST 手写数据集的实际应用场景：

MNIST 数据集是一个包含手写数字的大型数据库，它广泛用于训练各种图像处理系统。这个数据集因其简单性、易用性和丰富的标注信息而成为机器学习领域的“Hello World”。实际应用场景包括：

1.自动表单处理： 在银行、保险等行业，自动读取和识别手写表格。

2.移动应用： 手写数字识别应用，如计算器、笔记应用等。

1.2 目的：

1.2.1 培养学生对图像处理和机器学习的基本理解：

通过这个项目，学生将学习图像识别的基础知识，包括图像的表示、特征提取、分类器设计等。这将帮助他们理解图像识别技术是如何工作的，以及如何应用于实际问题。

1.2.2 训练学生使用深度学习模型解决实际问题的能力：

学生将通过构建和训练深度学习模型来识别手写数字，这不仅能够提升他们的编程技能，还能增强他们解决实际问题的能力。

1.2.3 提高学生的数据预处理、模型构建、评估和优化的技能：

在项目过程中，学生将学习如何进行数据预处理、选择合适的模型架构、

评估模型性能以及对模型进行调优。这些技能对于任何希望在人工智能领域发展的专业人士来说都是至关重要的。

二、 数据预处理

2.1 数据集描述：

MNIST 数据集是机器学习和计算机视觉领域内最知名的数据集之一，由美国国家标准与技术研究院（NIST）提供，后来被 Yann LeCun 等人用于训练多种机器学习模型。该数据集包含 60,000 个训练样本和 10,000 个测试样本，涵盖了手写数字 0 到 9。每个图像都是灰度的，分辨率为 28x28 像素。其特点如下特点：

1.大规模：包含大量的手写数字图像，适合训练统计模型。

2.高质量：图像经过预处理，背景清晰，数字居中。

3.多样性：尽管是手写数字，但书写风格多样，增加了识别的复杂性。

4.标注清晰：每个图像都有一个明确的标签，即对应的数字。

2.2 预处理步骤：

2.2.1 图像尺寸调整：

由于深度学习模型通常需要固定大小的输入，我们需要将所有图像调整到相同的尺寸。对于 MNIST 数据集，图像已经是统一的 28x28 像素，因此不需要额外调整。

2.2.2 归一化处理：

将图像的像素值从[0, 255]缩放到[0, 1]的范围，这有助于模型更快地收敛。在 MNIST 数据集中，可以通过除以 255 来实现。

```
from torchvision import transforms
```

```
# 定义转换操作
transform = transforms.Compose([
    transforms.ToTensor(),  # 将 PIL 图像或 NumPy ndarray 转换为
FloatTensor，并缩放到[0, 1]
    transforms.Normalize((0.5,), (0.5,))  # 归一化到[-1, 1]
])
```

2.2.3 数据增强：

通过应用各种变换（如旋转、翻转、缩放等）来增加数据的多样性，这有助于模型学习到更加鲁棒的特征。

```
from torchvision.transforms import RandomRotation,
RandomHorizontalFlip, RandomResizedCrop
transform = transforms.Compose([
    RandomRotation(10),  # 随机旋转
    RandomHorizontalFlip(),  # 随机水平翻转
    RandomResizedCrop(28, scale=(0.8, 1.0)),  # 随机裁剪并调整大小
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

2.2.4 划分数据集：

将数据集分为训练集、验证集和测试集，以评估模型的性能并避免过拟合。

```
from torch.utils.data import random_split
# 假设 dataset 是包含完整 MNIST 数据集的 DataLoader
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
train_dataset, test_dataset = random_split(dataset, [train_size,
test_size])
```

通过这些预处理步骤，我们能够为模型提供一个干净、标准化且多样化的数据集，这有助于提高模型的性能和泛化能力。

三、 模型构建

3.1 模型选择：

3.1.1 描述所选择的深度学习模型：

卷积神经网络（CNN）在深度学习领域中广受欢迎，主要用于图像数据的处理。它们模拟了人类视觉系统的工作机制，能够自动地逐层分析图像并提取出有用的特征。CNN 特别适合于图像识别任务，因为它们能够识别图像中的局部模式（通过卷积操作），并通过多层网络结构来提取更抽象的特征。简而言之，CNN 能够从简单到复杂逐步识别图像中的各种特征，使其在图像识别领域具有强大的应用能力。

3.1.2 理论基础：

1.局部感受野：每个卷积神经元只响应输入图像中的一个局部区域，这与生物视觉系统中的受体特性相似。

2.参数共享：在卷积层中，同一个卷积核（滤波器）在整个图像上滑动，这意味着相同的特征检测器可以在整个输入中使用，减少了模型的参数数量。

3.平移不变性：由于卷积操作的特性，CNN 能够学习到具有平移不变性的特征，即相同的特征在图像的不同位置都能被检测到。

4.层次结构：从浅层到深层，CNN 能够从简单的边缘和纹理特征逐渐学习到复杂的形状和对象特征

3.2 模型架构：

3.2.1 详细说明模型的层结构

以下是一个用于 MNIST 数据集的简单 CNN 模型架构示例：

```
import torch.nn as nn
class CNN(nn.Module):
    def __init__(self):
```

```
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32,
kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64,
kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(in_features=64 * 7 * 7,
out_features=128)
        self.fc2 = nn.Linear(in_features=128, out_features=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)  # Flatten the tensor
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    return x
```

3.2.2 描述激活函数、损失函数和优化器的选择：

激活函数：在这个模型中，我们使用 ReLU（Rectified Linear Unit）作为激活函数，因为它能够加速训练过程，并且减少梯度消失的问题。ReLU 函数定义为 f(x) = max(0, x)。

损失函数：对于多分类问题，我们通常使用交叉熵损失（Cross-Entropy Loss）。它测量的是模型预测的概率分布与真实标签的概率分布之间的差异。优化器：我们选择 Adam 优化器，因为它结合了 RMSProp 和 Momentum 两种优化算法的优点，并且能够自动调整学习率，适合大多数深度学习任务。

```
python
import torch.optim as optim
import torch.nn.functional as F
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

通过合理选择激活函数、损失函数和优化器，我们可以构建一个有效的 CNN

模型来处理 MNIST 手写数字识别任务。这些组件共同工作，使得模型能够从数据中学习并做出准确的预测。

四、 模型评估

4.1 评估指标：

在机器学习中，模型评估指标是衡量模型性能的关键。以下是一些常用的评估指标：

1. 准确率（Accuracy）：准确率是最直观的性能指标，它表示模型正确预测的样本数占总样本数的比例。对于分类问题，准确率计算公式为:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

其中 TP 是真正例，TN 是真负例，FP 是假正例，FN 是假负例。

2. 召回率（Recall）：召回率也称为真正率，它衡量的是模型成功识别的正样本占所有实际正样本的比例。召回率的计算公式为：

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

3. 精确率（Precision）：精确率衡量的是模型预测为正的样本中实际为正样本的比例。精确率的计算公式为：

$$Precision = \frac{TP}{TP+FP} \tag{4.3}$$

4. F1 分数（F1 Score）：F1 分数是精确率和召回率的调和平均值，它试图在精确率和召回率之间找到一个平衡。F1 分数的计算公式为：

$$\text{F1score} = \frac{2TP}{2TP + FP + FN} \tag{4.4}$$

4.2 评估方法：

交叉验证（Cross-Validation）：交叉验证是一种评估模型泛化能力的技术。最常见的是 K 折交叉验证，它将数据集分成 K 个大小相等的子集，每次用一个子集作为测试集，其余 K-1 个子集作为训练集。模型在 K 个不同的训练集和测试集上训练和测试，最终评估指标取 K 次的平均值。这种方法可以减少模型评估结果的方差，提高评估的稳定性和可靠性。

混淆矩阵（Confusion Matrix）：混淆矩阵是一个表格，用于描述分类模型的性能。它显示了每个类别的实际类别与模型预测类别之间的关系。混淆矩阵中，行表示实际类别，列表示预测类别。通过混淆矩阵，我们可以直观地看到模型在各个类别上的分类效果，以及模型可能存在的问题，如某些类别的误分类率特别高。

五、 结果分析与优化

5.1 结果分析：

5.1.1 对比不同模型的性能：

在结果分析阶段，我们通常会比较几种不同的模型或同一模型的不同配置的性能。这可以通过查看每个模型的准确率、召回率、精确率和 F1 分数等指标来完成。比较时，我们不仅要考虑模型的整体性能，还要考虑它们在不同类别上的表现差异。其中使用了 Accuracy 来展示模型预测的结果，如图 5 所示：

```
Epoch 029 train_loss 1.46442 val_loss 1.46518
train took 1168.2078 seconds to execute
Accuracy:99%
test took 2.5026 seconds to execute
```

图 5 Accuracy 结果展示

5.1.2 分析其优缺点：

优点：可能包括高准确率、良好的泛化能力、快速的推理时间等。

缺点：可能包括过拟合、欠拟合、对某些类别的预测性能差等。

5.1.3 讨论模型在特定类别上的表现差异：

有时候，模型可能在某些类别上表现特别好，而在其他类别上表现较差。这可能是由于数据不平衡、模型对某些特征的敏感度不足或者数据预处理的问题。通过分析混淆矩阵，我们可以识别出模型在哪些类别上存在问题，并进一步探究原因。

5.2 模型优化：

5.2.1 根据分析结果调整模型参数：

1.根据结果分析，我们可以调整模型的超参数，如学习率、批次大小、迭代次数等，以改善模型性能。例如，如果模型出现过拟合，我们可以通过增加数据增强、调整网络深度或正则化技术来减少过拟合。

5.2.2 尝试不同的网络结构或正则化技术以提高性能：

2.网络结构：尝试不同的网络架构，如增加或减少卷积层、改变全连接层的大小，或者引入残差连接（ResNet）等。

3.正则化技术：应用如 L1、L2 正则化、Dropout、Batch Normalization 等技术，以减少过拟合并提高模型的泛化能力。

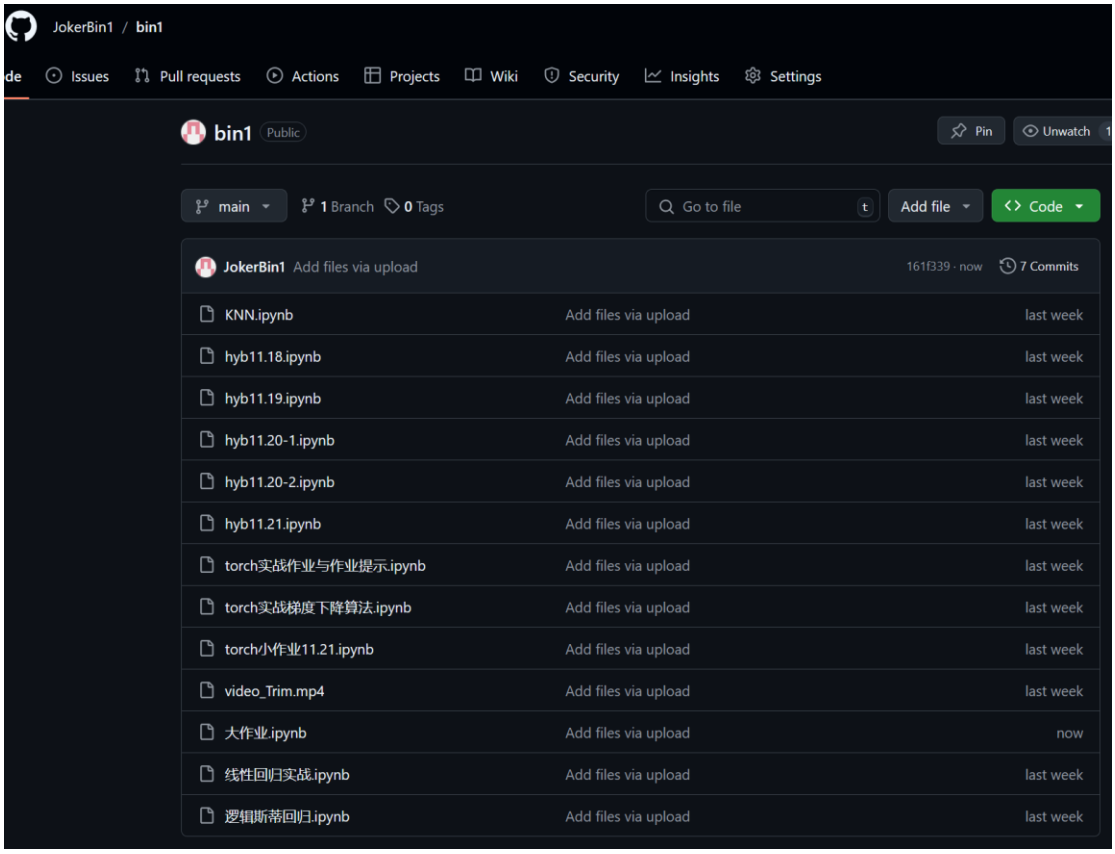
4.数据增强：通过旋转、缩放、裁剪等方法增加训练数据的多样性，可以帮助模型学习到更鲁棒的特征。

5.学习率调度：使用学习率衰减策略，如学习率衰减或周期性调整，以优化训练过程。

六、附录

github 项目链接：https://github.com/JokerBin1/bin1.git

附带仓库目录结构截图：


JokerBin1 / bin1
de
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
bin1
Public
Pin
Unwatch
main
1 Branch
0 Tags
Go to file
t
Add file
Code
JokerBin1 Add files via upload
161f339 · now
7 Commits
KNN.ipynb Add files via upload last week
hyb11.18.ipynb Add files via upload last week
hyb11.19.ipynb Add files via upload last week
hyb11.20-1.ipynb Add files via upload last week
hyb11.20-2.ipynb Add files via upload last week
hyb11.21.ipynb Add files via upload last week
torch实战作业与作业提示.ipynb Add files via upload last week
torch实战梯度下降算法.ipynb Add files via upload last week
torch小作业11.21.ipynb Add files via upload last week
video_Trim.mp4 Add files via upload last week
大作业.ipynb Add files via upload now
线性回归实战.ipynb Add files via upload last week
逻辑斯蒂回归.ipynb Add files via upload last week