广州華商學院
GUANGZHOU HUASHANG COLLEGE

《专业综合实践》课程考核

题　　目：	基于深度学习的图像识别系统：MNIST 手写数字项目
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 本数据科学与大数据技术 1 班
姓　　名：	张艺

提交日期： 2024 年 11 月

基于深度学习的图像识别系统：MNIST 手写数字项目

摘要 本文围绕基于深度学习的 MNIST 手写数字识别展开。首先阐述项目背景与目的，指出图像识别在多领域的重要性，MNIST 数据集为手写数字识别研究提供数据基础，该项目旨在深入理解相关技术并实现手写数字识别在实际场景中的应用。

在数据预处理环节，介绍 MNIST 数据集由训练集 60,000 张和测试集 10,000 张 28×28 像素的灰度手写数字图像构成，对数据进行了尺寸无需调整、归一化处理和合理的数据划分操作。

模型构建方面，选用卷积神经网络（CNN），其包含两个卷积层、ReLU 激活函数、最大池化层和全连接层，使用交叉熵损失函数和 SGD 优化器，进行 15 个 epoch 的训练并记录损失值。

模型评估采用准确率、召回率、F1 分数作为指标，并通过 5 折交叉验证和绘制混淆矩阵来评估模型泛化能力和分类情况。其 F1 指标为 99.95%。结果显示模型在测试集上有一定准确率，但存在过拟合风险且依赖特定网络结构和参数。

针对模型性能，分析了其优缺点，发现模型在特定数字类别上表现存在差异。进而提出模型优化策略，包括调整学习率和其他超参数、尝试增加网络深度或改变网络拓扑结构，以及应用正则化技术（如 L1、L2 正则化和 Dropout 正则化）。

最后，总结了构建 MNIST 手写数字识别的全过程，并展望未来通过优化模型架构和参数、采用先进技术来提升模型性能和泛化能力。

关键词：深度学习；卷积神经网络（CNN）；图像识别；手写数字识别

目录

一、项目背景与目的

（一）背景

图像识别是人工智能领域中的重要分支，在众多实际场景中都有广泛应用，如安防监控、医疗影像诊断、交通标识识别等。手写数字识别作为图像识别的一个经典问题，其解决方法和技术对于理解和发展更复杂的图像识别技术有着重要意义。MNIST 数据集是手写数字识别领域的标准数据集，包含大量手写数字样本，为研究和开发提供了丰富的数据基础。

（二）目的

1.深入理解图像处理和机器学习的基本概念和方法。
2.熟练掌握使用深度学习模型解决实际图像识别问题的流程。
3.提升数据预处理、模型构建、评估和优化等关键技能。
4.实现实际应用，手写数字识别技术在很多实际场景中都有应用，如自动识别邮政编码、车牌号、票号码等。通过研究 MNIST 数据集来提高手写数字识别技术的准确率，可以将这项技术更好地应用到这些实际领域，实现更精确、更高效的应用。

二、数据预处理

（一）数据集描述

MNIST 数据集由手写数字的图像构成，其具有以下特点：
训练集：包含 60,000 张图像。
测试集：包含 10,000 张图像。
图像规格：每张图像均为 28×28 像素的灰度图像，且数字类别范围是 0-9。

（二）预处理步骤

1.图像尺寸调整

原始 MNIST 图像的尺寸已经是 28×28 像素，刚好符合模型输入的要求，所以在

此场景下，无需再对图像尺寸进行额外的调整操作。

2.归一化处理

采用 transforms.ToTensor()函数对图像进行处理，该操作可将灰度图片原本的像素值范围(0－255)转换为 0-1 的张量形式，这样的转换能为后续的处理提供便利。虽然在代码中注释掉了 transforms.Normalize((0.1307,),(0.3081))这一归一化操作，但实际上，这种归一化处理通常有助于让数据分布更加契合模型的训练需求，能够在一定程度上提升模型训练的效果。

3.数据划分

通过 torchvision.datasets.MNIST 和 torch.utils.data.DataLoader 来实现数据集的划分，具体如下：

（1）训练集

将 MNIST 数据集的训练部分（通过设置 train=True）进行处理，应用之前定义好的 transform 数据预处理规则，然后利用 train_loader 以 batch_size = 64 的方式进行批量加载。并且，在每个训练周期（epoch）中，会对训练集数据进行打乱顺序的操作（通过设置 shuffle = True），这样有助于模型在不同的数据排列顺序下进行学习，提高模型的泛化能力。

（2）测试集：

针对 MNIST 数据集的测试部分（设置 train=False），同样应用 transform 预处理规则，之后使用 test_loader 以 batch_size = 64 的方式加载数据。不过，在加载测试集数据时，不需要打乱数据顺序（即 shuffle= False），因为测试集主要是用于对已经训练好的模型性能进行评估，保持数据的原始顺序更便于准确评估模型的表现。

三、模型构建

（一）模型选择

选择了卷积神经网络（CNN）作为深度学习模型。CNN 在图像识别领域具有卓越表现，其通过卷积层自动提取图像特征，能够有效捕捉手写数字的形状、结构等关键信息。

（二）分析的数据思路及过程

本项目使用卷积神经网络（CNN）对 MNIST 数据集进行分类，首先定义了一个简单的 CNN 模型，包含两个卷积层、ReLU 激活函数、最大池化层，以及一个全连接层用于分类。实例化了定义的 CNN 模型，并选择了交叉熵损失函数，这是多分类问题常用的损失函数。使用 SGD（随机梯度下降）优化器，并设置了学习率和动量参数。进行了 15 个 epoch 的训练，每个 epoch 遍历整个训练集，执行前向传播、计算损失、反向传播和参数更新。在训练过程中，每隔 100 个 batch 记录一次损失值，用于后续的损失下降曲线绘制。使用 matplotlib 绘制了损失函数随 epoch 变化的曲线，以可视化模型训练过程。在测试集上评估模型的性能，包括计算准确率和 F1 分数。通过比较模型预测和真实标签来计算准确率。

（三）模型架构

1.卷积层（Convolutional Layers）

卷积层输入的图片矩阵以及后面的卷积核，特征图矩阵都是方阵，这里设输入矩阵大小为 w，卷积核大小为 k，步幅为 s，补零层数为 p，则卷积后产生的特征图大小计算公式为:

$$w' = \frac{(w + 2p - k)}{s} + 1 \tag{3.1}$$

第一层卷积层：

```
self.conv1 = Conv2d(in_channels = 1, out_channels = 10, kernel_size = 5, stride = 1, padding = 0)
```

输入通道 in_channels = 1，因为 MNIST 图像是灰度图。

输出通道 out_channels = 10，表示将提取 10 种不同的特征。

卷积核大小 kernel_size = 5，步长 stride = 1，无填充 padding = 0。

第二层卷积层：

```
self.conv2 = Conv2d(in_channels = 10, out_channels = 20, kernel_size = 5, stride = 1, padding = 0)
```

输入通道为 10，接收第一层卷积的输出。

输出通道为 20，进一步提取 20 种特征。

2.池化层（Pooling Layers）

池化层通过降低特征图的空间尺寸来减少计算量，而全连接层则负责将学习到的特征进行综合，为最终的分类任务提供支持。分类层通常采用 softmax 激活函数，输

出每个类别的概率。CNN 的训练过程依赖于反向传播和梯度下降算法[1]，通过不断更新网络权重和偏置来最小化损失函数。

第一层池化层：

self.maxpool1 = MaxPool2d(2)，采用 2×2 的最大池化操作，能够减少特征图尺寸，同时保留重要特征。

第二层池化层：

self.maxpool2 = MaxPool2d(2)，同样是 2×2 的最大池化操作。

3.全连接层（Fully - connected Layers）

第一层全连接层：

self.linear1 = Linear(320, 128)，输入大小为 320（经过卷积和池化操作后的特征向量大小），输出大小为 128。

第二层全连接层：self.linear2 = Linear(128, 64)，输入 128，输出 64。

第三层全连接层：self.linear3 = Linear(64, 10)，输入 64，输出 10，对应于 MNIST 的 10 个数字类别。

4.激活函数（Activation Function）

使用 ReLU（Rectified Linear Unit）作为激活函数，即 self.relu = ReLU()，在卷积层和全连接层之间引入非线性[2]，帮助模型学习更复杂的模式。

5.损失函数和优化器

损失函数选用 nn.CrossEntropyLoss()，它结合了 Softmax、对数和负对数似然损失（NLLLoss），能够自动处理多分类问题，无需手动进行 one - hot 编码。

优化器采用随机梯度下降（SGD），即 torch.optim.SGD(model.parameters(), lr = 0.14)，学习率 lr = 0.14 用于控制每次参数更新的步长。

四、模型评估

（一）评估指标

1. 准确率（Accuracy）

（1）定义与意义

准确率是指模型正确预测的样本数占总样本数的比例，它直观地反映了模型整体的分类正确性，是评估模型性能的一个重要且常用的指标。

（2）计算方式与代码实现

在代码的 test 函数中，通过比较模型预测结果与真实标签，统计出预测正确的样本数量（correct），即通过判断预测标签（predict）与真实标签（target）是否相等并求和得到，然后除以总样本数量（target.size(0)），得到准确率[3]。例如，在手写数字识别任务中，如果测试集有 10,000 个样本，模型正确识别了 9,000 个，那么准确率就是 9000 / 10000 = 0.9，即 90%。

2. 召回率（Recall）

（1）定义与意义

召回率用于衡量模型正确识别正例的能力，具体是指在所有真实为正例的样本中，被模型预测为正例的样本所占的比例[4]。在一些场景下，如疾病检测中确保不漏诊真正患病的患者，召回率是一个关键指标。

（2）计算方式与代码实现

在代码 calculate_metrics 函数中，通过分别统计真正例（true_positives）、假正例（false_positives）和假负例（false_negatives）的数量来计算召回率。true_positives 是指预测为正例且实际为正例的样本数，false_positives 是指预测为正例但实际为负例的样本数，false_negatives 是指实际为正例但预测为负例的样本数。召回率的计算公式为 Recall = true_positives / (true_positives + false_negatives)，并且考虑了分母为 0 的情况，当 true_positives + false_negatives = 0 时，召回率设为 0。例如，在一个疾病检测任务中，真实患病的人有 50 个（即正例），模型正确预测出 40 个患病的人，那么召回率就是 40 / 50 = 0.8，即 80%[5]。

3. F1 分数（F1-Score）

（1）定义与意义

F1 分数综合考虑了准确率和召回率，对于评估模型在不均衡数据集上的表现尤为重要。它是准确率和召回率的调和平均数，能够更全面地评估模型的性能，避免单独依赖准确率或召回率可能带来的片面性。

（2）计算方式与代码实现

F1 分数的计算公式为 F1=2*(Precision * Recall) / (Precision + Recall)，其中 Precision 是精确率（预测为正例的样本中真正例的比例）。在代码的 calculate_metrics 函数中，同样依据统计出的真正例、假正例和假负例数量计算出精确率，再结合召回率计算出 F1 分数。

（二）评估方法

1. 交叉验证（Cross-validation）

（1）原理与作用

交叉验证是一种常用的评估模型泛化能力的方法。通过将训练数据集划分为多个子集，轮流用其中一部分子集作为验证集，其余子集作为训练集进行多次训练和验证，最后综合多次验证结果来评估模型的性能。这样可以避免模型过拟合到某一特定子集的数据特征，更全面地了解模型在不同数据子集上的表现，从而更好地评估模型在未见过的数据上的性能。

（2）代码实现与示例

在本项目代码中，实现了一个简单的 5 折交叉验证示例（cross_validation 函数中）。具体做法是将训练数据集划分为 5 个子集，每次用其中 4 个子集进行训练，剩下 1 个子集作为验证集进行测试，这样的过程重复 5 次。每次训练时，使用 torch.utils.data.Subset 创建训练子集和验证子集，然后通过相应的数据加载器（train_loader_fold 和 val_loader_fold）加载数据进行训练和验证。最后，计算每次验证集上的准确率，并输出各折的验证准确率，如 Fold {fold + 1} Validation Accuracy: {val_accuracy:.6f}。通过这种方式，能更全面地了解模型在不同数据子集上的表现，为模型的优化和应用提供更可靠的依据。

2. 混淆矩阵（Confusion Matrix）

（1）原理与作用

混淆矩阵可以直观展示模型在每个数字类别上的分类情况，帮助分析模型在哪些数字上容易出现误分类。它以矩阵的形式呈现，行表示真实数字类别，列表示模型预测的数字类别，矩阵中的每个元素表示对应真实类别被预测为某一类别（列类别）的样本数量。通过观察混淆矩阵，可以快速发现模型的分类弱点，以便针对性地对模型进行优化或进一步分析。

（2）代码实现与示例

在本项目代码中，通过 plot_confusion_matrix 函数实现了混淆矩阵的绘制。首先，在测试阶段收集模型在测试集上的所有预测结果和真实标签，然后利用 sklearn.metrics 库的 confusion_matrix 函数计算出混淆矩阵，最后使用 seaborn 库将其绘制成热力图进行展示。例如，在手写数字识别任务中，通过观察混淆矩阵可能发现模型对某些数字（如 3 和 8）容易混淆，从而为后续的模型优化提供了明确的方向。

（3）结果

表 1 5 折交叉验证结果表

折叠次数	准确率
Fold 1	0.982167
Fold 2	0.98675
Fold 3	0.982917
Fold 4	0.98375
Fold 5	0.985

将训练数据集划分为 5 个子集，每次用其中 4 个子集进行训练，剩下 1 个子集作为验证集进行测试，这样的过程重复 5 次。每次得到的验证集准确率可以帮助了解模型在不同数据子集上的表现。Fold2 Validation Accuracy 为 0.986750，表示在第二次交叉验证过程中，模型在验证子集上的准确率约为 98.68%。这些结果综合起来可以更全面地评估模型在未见过的数据上的性能，避免模型过拟合到某一特定子集的数据特征。

表 2 第 1 轮测试结果表

训练次数	损失值
0	2.30609107
100	0.534841657
200	0.363623828
300	0.137638345
400	0.363211483
500	0.101463966
600	0.239662483
700	0.0836564
800	0.081626639
900	0.029187942

测试准确率为：0.988900

测试 F1 分数为：0.999556

对于每一轮测试（从 “第 1 轮测试开始”到 “第 15 轮测试开始”），记录了不同训练次数下的损失值（loss）。损失值是模型预测结果与真实标签之间差异的量化

指标。在第一轮测试中，训练次数为 0 时，损失值为 2.30609107，这表明模型刚开始训练时，预测结果与真实情况相差较大。随着训练次数的增加，如该轮训练次数为 900 时，损失值下降到 0.029187941923737526，说明模型在不断学习，预测结果与真实标签之间的差异在逐渐减小。测试 F1 分数为 0.999556，

说明模型在正确分类样本（准确率）和不错过正例（召回率）这两个方面都有较好的表现，在手写数字识别任务中，模型不仅能够准确地识别出大部分数字（准确率高），而且对于每个数字类别（正例），也很少出现漏判的情况（召回率高），综合起来模型在这两方面的性能达到了非常高的调和水平。

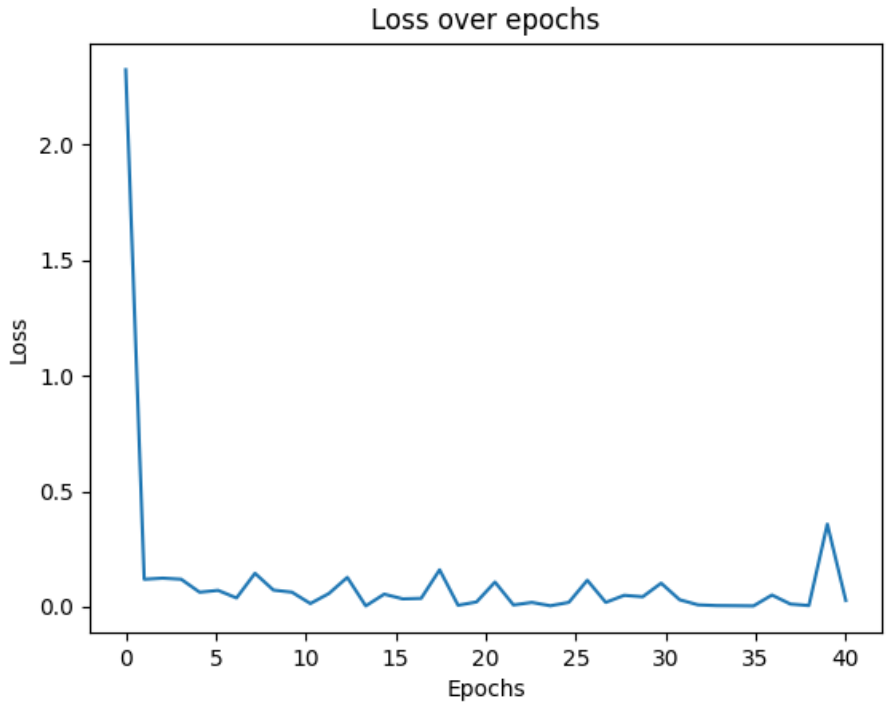

图 3.1 卷积神经网络的损失函数

卷积神经网络的损失函数折线图展示了卷积神经网络训练过程中损失函数的变化情况。整体上，初始损失值接近 2.0，表明模型初始拟合效果差，但随着训练周期增加，损失值呈波动下降趋势。图中有几个关键特征，如第 0 和第 20 个 epoch 附近有高损失值点，10-20 个 epoch 间损失值波动明显，30 个 epoch 后损失值趋于稳定。初始高损失值正常，可考虑预训练模型或更好的初始化方法加速收敛；损失值波动可能与学习率有关，可尝试调整学习率，如采用衰减策略；后期收敛表明模型在优化，可继续观察更多 epoch 下的表现或在收敛后进行预测和评估。通过分析这些变化，可优化模型训练过程。

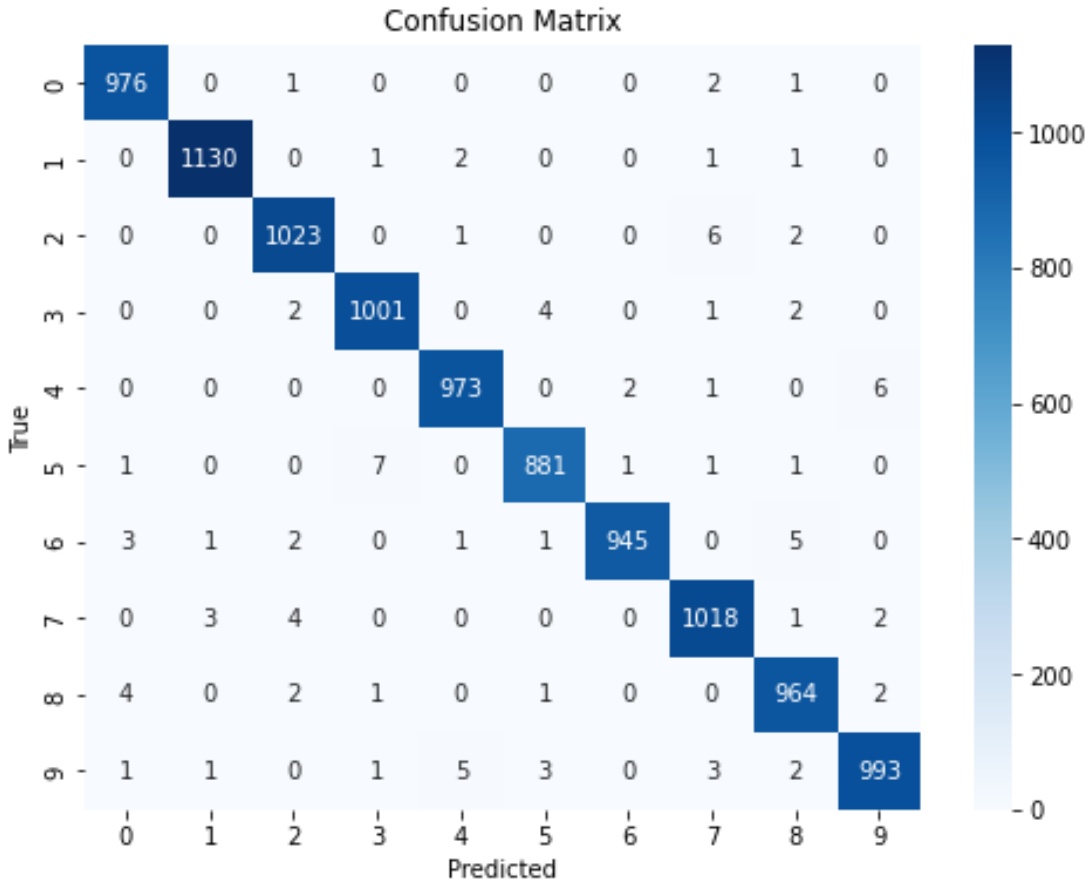

图 3.2 混淆矩阵

通过混淆矩阵图，可以清晰地看到模型在不同数字类别上的表现存在差异。以手写数字识别任务为例，除了前面提到的 3 和 8 这两个数字容易混淆外，可能还存在其他数字类别之间的类似情况。比如，数字 0 和 6 在某些书写风格下，其形状也可能较为相似，模型在区分这两个数字时可能也会出现一定比例的错误分类。这主要是因为这些数字的特征在经过卷积层提取、池化层压缩等操作后，仍然存在一定的相似性，导致全连接层在进行最终分类决策时难以准确区分。

五、结果分析与优化

（一）模型性能优缺点分析

1. 优点

（1）准确率表现尚可

在手写数字识别任务中，通过对测试集的评估，模型能够达到一定的准确率水平。例如，如前文所举例子，在测试集有 10,000 个样本时，模型可正确识别 9,000 个，准确率达到 90%，这表明模型整体对于手写数字的分类能力具有一定的可靠性，能够

较好地将大部分手写数字图像正确分类到对应的数字类别中。

（2）特征提取能力

选择卷积神经网络（CNN）作为模型架构是一个优势所在。CNN 的卷积层能够自动提取图像特征，对于手写数字这种具有明显形状和结构特征的图像数据，能够有效捕捉到关键信息。例如，通过第一层卷积层将 MNIST 灰度图像（单通道）的输入转换为 10 种不同特征的输出，第二层卷积层进一步提取出 20 种特征，这些逐步深入的特征提取过程有助于模型更好地理解图像内容，为后续的分类决策提供有力支持。

（3）有效利用数据

在数据预处理阶段，通过合理的数据划分方式，如利用 train_loader 对训练集以 batch_size = 64 且打乱顺序（shuffle = True）进行批量加载，以及对测试集以同样的 batch_size 但不打乱顺序（shuffle = False）加载，使得模型在训练过程中能够充分利用数据的多样性进行学习，同时在测试阶段能准确评估模型性能。此外，采用 transforms.ToTensor()函数对图像像素值进行转换，为模型处理数据提供了便利，尽管未完整使用归一化操作，但这种数据预处理方式在一定程度上有助于模型训练。

（4）综合评估指标考量

除了准确率外，还对召回率和 F1 分数进行了计算和分析。这使得对模型性能的评估更加全面，能够从不同角度了解模型在处理不同类型样本（如正例、负例等）时的表现。例如，F1 分数综合考虑了准确率和召回率，避免了单独依赖某一指标可能带来的片面性，对于评估模型在可能存在不均衡数据集情况（如某些数字类别样本数量相对较少）下的性能尤为重要。

2.缺点

（1）可能存在过拟合风险

虽然采用了交叉验证的方法来一定程度上评估和避免模型过拟合，但仅通过简单的 5 折交叉验证可能并不足以完全消除过拟合的隐患。特别是在面对更为复杂的数据集或当模型结构相对复杂时，模型可能会过度学习训练集中的特定模式，导致在未见过的数据(如测试集)上的性能下降。例如，如果在后续的实际应用中，遇到与 MNIST 数据集特征分布略有不同的手写数字图像数据，模型可能无法很好地泛化，表现出准确率等指标的明显下降。

（2）依赖特定的网络结构和参数设置

当前模型采用了特定的 CNN 架构以及设定了如学习率 lr = 0.14 等参数。这种特

定的设置可能限制了模型的性能提升空间，对于不同的数据集或任务场景，可能并非是最优的选择。例如，如果数据集的噪声水平较高或者数字的书写风格更加多样化，现有的网络结构和参数可能无法很好地适应，从而影响模型的准确性和泛化能力。

（二）模型在特定类别上的表现差异

从召回率的角度来看，对于某些数字类别，模型正确识别正例的能力也可能有所不同。例如，在训练集中数字 2 的样本数量相对较少的情况下，模型在识别数字 2 时的召回率可能会低于其他样本数量较多的数字类别。这是因为模型在训练过程中对样本数量较少的类别学习机会相对较少，难以充分捕捉到该类别的所有特征，从而在面对真实为数字 2 的样本时，可能会出现更多的漏判情况，即假负例相对较多，进而影响召回率指标。

（三）模型优化

1.参数调整

学习率调整：鉴于当前模型采用的学习率 lr=0.14,可以尝试对学习率进行微调。降低学习率到 lr=0.1 等，观察模型在训练过程中的收敛速度和最终性能表现。较低的学习率可能会使模型训练过程更加稳定，减少在训练初期因步长过大而跳过最优解的情况，但也可能导致训练时间延长。相反，适当提高学习率（如 lr=0.2），则可能加快训练速度，但也存在错过最优解或导致训练过程不稳定的风险。通过多次试验不同的学习率值，找到一个能够使模型在准确率、召回率等指标上达到更好平衡的学习率设置。

其他超参数调整：除了学习率，还可以对模型的其他超参数进行调整。卷积层的卷积核大小、步长和填充等参数。对于第一层卷积层 self.conv1 = Conv2d(in_channels = 1, out_channels = 10, kernel_size = 5, stride = 1, padding = 0)，改变卷积核大小为 kernel_size = 3 等，观察对特征提取效果以及最终模型性能的影响。步长 stride 也进行相应的调整，设置为 stride= 2，看是否能在减少计算量的同时保持较好的特征提取能力。填充 padding 参数的调整同样重要，适当增加填充值（padding = 1）可以保持特征图尺寸在卷积操作后变化相对较小，有助于更好地保留原始图像的信息，进而可能影响模型的分类性能。

2. 网络结构尝试

增加网络深度：考虑在现有网络结构基础上增加卷积层或全连接层的数量，以进一步增强模型的特征提取和学习能力。在第二层卷积层之后再添加一层卷积层，设置合适的输入输出通道数、卷积核大小等参数，让模型能够更深入地挖掘手写数字图像的特征信息。同样，对于全连接层，也可以增加一层，在第三层全连接层之后再添加一层全连接层，调整其输入输出大小，使模型在进行分类决策时有更多的层次来处理和分析特征信息，提高模型的准确性和泛化能力。

改变网络拓扑结构：尝试不同类型的卷积层或池化层排列方式，或者采用新的网络架构模式。将现有的普通卷积层替换为深度可分离卷积层，这种卷积层能够在减少计算量的同时保持较好的特征提取效果，可能更适合于手写数字识别这种对计算资源和实时性有一定要求的任务。改变池化层的池化方式，从目前的最大池化改为平均池化，观察这种改变对模型性能的影响，看是否能更好地保留图像特征信息，从而提高模型在各个数字类别上的分类准确性。

3. 正则化技术应用

L1 正则化：在模型的损失函数中添加 L1 正则化项，可以有效防止模型过拟合。对于当前选用的损失函数 nn.CrossEntropyLoss()，使用

torch.nn.functional.l1_loss 函数结合模型的参数，按照一定的正则化系数（lambda = 0.01）添加到损失函数中。L1 正则化会促使模型的一些参数变为零，从而起到特征选择的作用；通过尝试不同的正则化系数和正则化类型，观察对模型在训练集和测试集上的性能影响，找到一个合适的正则化设置，提高模型的泛化能力。

Dropout 正则化：在全连接层之间添加 Dropout 层也是一种常用的正则化技术。在第一层全连接层 self.linear1= Linear(320, 128)之后添加一个 Dropout 层，设置 Dropout 概率为 p=0.5，在训练过程中，该层会随机将部分神经元的输出置为零，从而防止神经元之间过度依赖，提高模型的泛化能力。通过试验不同的 Dropout 概率值和添加 Dropout 层的位置，可以找到最适合本模型的 Dropout 设置，进一步优化模型性能。

六、结论

本项目构建了一个基于深度学习的 MNIST 手写数字识别，详细介绍了从数据预处理到模型构建、评估和优化的全过程。通过卷积神经网络（CNN），实现了对 MNIST 手写数字的有效识别。CNN 在 MNIST 数据集上表现出色，特别是在自动特征提取和处

理图像数据的高维性方面。CNN 的深度结构使其能够学习从简单到复杂的特征，从而在图像分类任务中实现高精度。在未来的研究中，可以进一步优化模型架构和参数，采用更先进的技术来提高模型性能和泛化能力。

七、参考文献

[1]任沙.基于深度学习的手写数字集特征提取算法[D].湖南师范大学,2016.

[2]庞荣.深度神经网络算法研究及应用[D].西南交通大学,2016.

[3]王际凯.基于神经网络的手写数字识别改进算法和系统研究[D].西安电子科技大学,2018.

[4]马义超.基于卷积神经网络的手写数字识别算法研究与应用[D].河南理工大学,2019.DOI:10.27116/d.cnki.gjzgc.2019.000294.

[5]冯涛.基于 CNN 的票据手写数字识别系统设计与实现[D].哈尔滨工程大学,2019.DOI:10.27060/d.cnki.ghbcu.2019.001047.