合肥工业大学

《机器视觉》课程实验

实验题目　课程实验三：学号识别

学生姓名　史皓宇

学　　号　2023217603

专业班级　智能科 23-3 班

一、实验目的

深入理解手写数字识别的核心逻辑，掌握深度学习（CNN）在图像分类任务中的应用原理，明确卷积神经网络从特征提取到分类决策的完整流程，理解卷积、池化、激活函数、正则化等关键模块的作用。

熟练运用 MNIST 数据集进行模型训练与验证，掌握数据集的自动下载、加载、预处理（数据增强、归一化）方法，理解数据增强对提升模型泛化能力的重要性。

掌握手写学号识别的端到端实现流程，包括输入图像预处理（二值化、轮廓提取、噪声过滤、尺寸适配）、单个数字分割与识别、识别结果拼接，明确每个环节对最终识别准确率的影响。

熟练使用 PyTorch 深度学习框架，实现 CNN 模型的构建、训练、优化与保存，掌握优化器（Adadelta）、学习率调度器（StepLR）、损失函数（NLLLoss）的配置与使用，提升深度学习工程实践能力。

完成实验加分项要求：通过深度学习方法实现学号识别，独立完成自定义命名的实验环境配置，掌握虚拟环境创建、依赖库安装、版本兼容性验证的完整流程。

培养问题排查与优化思维，针对学号识别中可能出现的轮廓分割错误、数字识别偏差等问题，理解参数调整（如轮廓面积阈值、

图像缩放比例）的优化逻辑，提升复杂场景下的视觉识别问题解决能力。

二、实验环境

2.1软件环境

编程语言：Python

依赖库：OpenCV（cv2）1.45+、NumPy 1.20+、Matplotlib 3.5+、OS

开发工具：PyCharm 2022

2.2硬件环境

处理器：Intel Core i7-13800H

内存：32GB

显卡：NVIDA RTX 2000 Ada Generation Laptop GPU

三、实验原理

3.1实验环境配置（加分项）

由于我早就已经长期使用anaconda，所以base环境中就已经有常用的各种包。所以要在终端中创建一个shy环境，可以复制anaconda3的base环境，如下图。

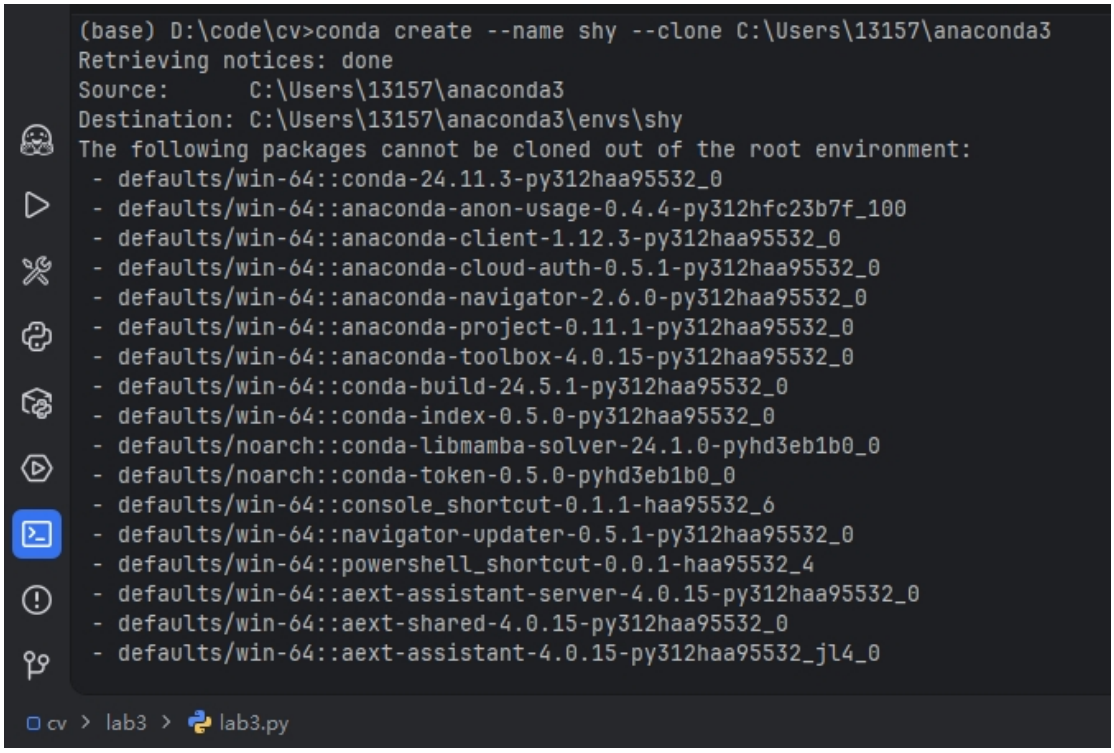

图1 创建shy环境

查看环境列表，然后激活shy环境，之后运行代码即可。

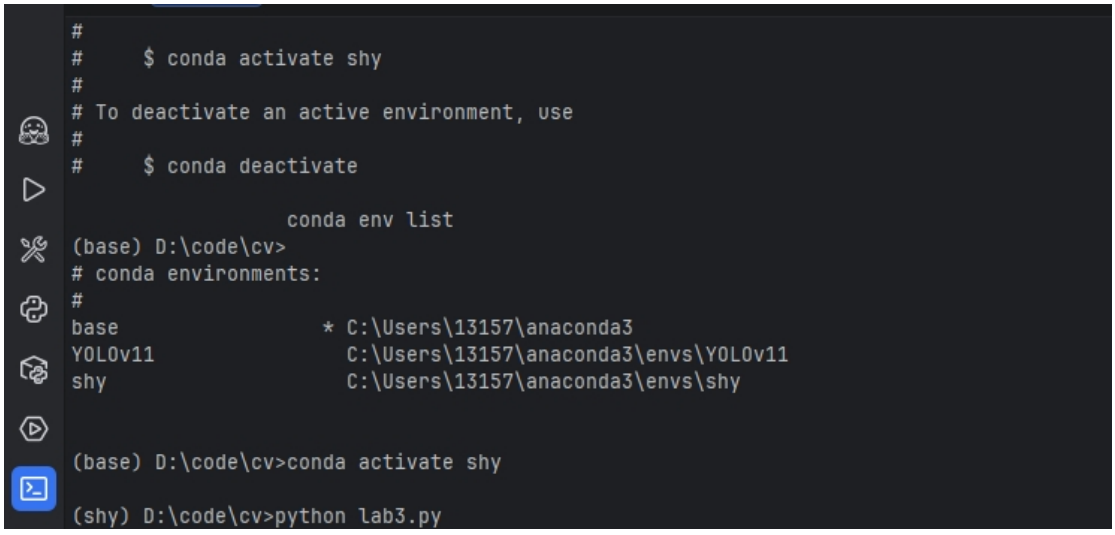

图2 激活shy环境

3.2MNIST 数据集与数据预处理原理

MNIST 数据集特性：MNIST 是手写数字分类任务的经典数据集，包含 60000 张训练集图像和 10000 张测试集图像，每张图像为 28×28 像素的单通道灰度图（黑底白字），标签为 0~9 的阿拉伯数字。

训练集预处理：包含随机旋转（±10°）、随机平移（水平 / 垂直方向 10%）的数据增强操作，目的是提升模型泛化能力，避免过拟合；通过ToTensor()将图像转换为 PyTorch 张量（维度：C×H×W，取值范围 [0,1]），再用Normalize((0.1307,), (0.3081,))进行归一化（MNIST 数据集的均值和标准差，使数据分布更接近标准正态分布，加速模型收敛）。

测试集预处理：仅包含张量转换与归一化，避免数据增强对测试结果的干扰，确保测试集的真实性。

3.3CNN 模型结构与工作原理

代码中Net类定义的 CNN 模型采用 “卷积提取特征→池化降维→全连接分类” 的经典架构，各层功能如下：

（1）卷积层（Conv2d）：

conv1：输入通道 1（单通道灰度图），输出通道 32，卷积核大小 3×3，步长 1；通过 32 个不同的卷积核对输入图像进行特征提取，捕捉边缘、纹理等低级视觉特征。

conv2：输入通道 32，输出通道 64，卷积核大小 3×3，步长 1；在低级特征基础上提取更复杂的中级特征（如数字的笔画组合）。

激活函数（ReLU）：引入非线性变换，使模型能够拟合复杂的分类函数，缓解梯度消失问题，提升模型表达能力。

（2）池化层（MaxPool2d）：采用 2×2 最大池化，步长默认与核大小一致；在保留关键特征的同时，将特征图尺寸缩小一半（降维），减少计算量，增强模型对图像微小位移的鲁棒性。

（3）正则化层（Dropout）：

dropout1： dropout 概率 0.25，在池化后随机丢弃 25% 的特征图元素，防止模型过度依赖局部特征，缓解过拟合。

dropout2： dropout 概率 0.5，在全连接层后丢弃 50% 的神经元，进一步增强模型泛化能力。

（4）全连接层（Linear）：

fc1：输入维度 9216（conv2 输出特征图经池化后尺寸为 12×12，64×12×12=9216），输出维度 128；将高维特征图转换为一维特征向量，进行高级特征融合。

fc2：输入维度 128，输出维度 10（对应 0~9 共 10 个数字类别）；将融合后的特征映射到类别空间。

（5）输出层（LogSoftmax）：对全连接层输出进行对数 softmax 变换，将输出值转换为概率分布（求和为 1），便于计算交叉熵损失（NLLLoss）。

3.4模型训练与优化原理

模型模式切换：model.train()开启训练模式（启用 Dropout），model.eval()开启评估模式（关闭 Dropout）。

数据加载：通过DataLoader将 MNIST 数据集按批次（batch_size=64）加载，支持 shuffle（训练集打乱）、多线程加载（GPU 模式下），提升训练效率。

梯度下降：优化器（Adadelta）初始化梯度为 0（optimizer.zero_grad()），前向传播计算模型输出与损失（NLLLoss，适用于 LogSoftmax 输出），反向传播（loss.backward()）计算梯度，优化器更新模型参数（optimizer.step()）。

学习率调度：StepLR调度器每轮训练后将学习率乘以 gamma（0.7），逐步降低学习率，使模型在训练后期更稳定地收敛到最优解。

评估指标：测试集上计算平均损失（所有样本损失之和 / 样本总数）与准确率（正确识别样本数 / 总样本数），评估模型泛化能力。

3.5学号识别的端到端流程原理

图像读取与灰度转换：cv2.imread读取彩色学号照片，cv2.COLOR_BGR2GRAY转换为单通道灰度图。

二值化与反转：cv2.threshold采用 OTSU 自适应阈值 + 反转二值化（THRESH_BINARY_INV），将 “白底黑字” 的手写学号转换为 “黑底白字”，与 MNIST 数据集格式一致，便于模型识别。

轮廓检测与过滤：cv2.findContours提取二值图像中的数字轮廓，通过cv2.contourArea过滤面积小于 20 的噪声轮廓（如纸张污渍）。

轮廓排序：按轮廓左上角 x 坐标排序（cv2.boundingRect(c)[0]），确保数字按学号从左到右的顺序排列，避免识别结果顺序错乱。

ROI 裁剪：通过cv2.boundingRect获取每个数字轮廓的边界框，裁剪出单个数字的感兴趣区域（ROI）。

尺寸适配：将 ROI 按比例缩放到 20×20（保证数字纵横比），再填充到 28×28 画布中心（匹配 MNIST 输入尺寸），避免数字变形导致识别误差。

归一化：transforms.ToTensor()将图像转换为张量并归一化到[0,1]，transforms.Normalize采用 MNIST 的均值和标准差，使输入数据分布与训练数据一致。

数字识别与结果拼接：将适配后的单个数字张量输入训练好的 CNN 模型，通过output.argmax(dim=1)获取概率最大的类别（即预测数字），按轮廓排序顺序拼接所有数字，得到最终学号。

四、实验步骤

4.1实验准备阶段

数据集准备：无需手动下载 MNIST 数据集，代码会自动检测本地是否存在（./data/MNIST），不存在则自动下载并保存到指定目录。

输入图像准备：用手机拍摄手写学号照片（满足 “输入图像要求”），命名为student_id.png，放置于实验代码所在目录。

4.2模型训练流程

训练参数配置：在main函数中设置训练轮次（epochs=10）、批次大小（batch_size=64）、学习率（lr=1.0）、学习率衰减系数（gamma=0.7）等参数。

设备选择：代码自动检测是否支持 CUDA，优先使用 GPU（device="cuda"），否则使用 CPU（device="cpu"），打印当前使用设备。

数据集加载：定义训练集与测试集的预处理 transform，通过datasets.MNIST加载数据集，创建DataLoader用于批量加载数据。

模型与优化器初始化：实例化Net模型并移至指定设备，初始化 Adadelta 优化器与 StepLR 学习率调度器。

模型训练与评估：循环执行 10 轮训练，每轮训练后在测试集上评估模型性能，打印训练进度（轮次、样本占比、损失）与测试结果（平均损失、准确率）。

模型保存：训练完成后，将模型参数保存为mnist_cnn.pt文件，便于后续复用（无需重复训练）。

4.3学号识别流程

图像预处理：调用predict_student_id函数，加载student_id.png，执行灰度转换、二值化反转、轮廓检测与过滤、轮廓排序。

单个数字分割与适配：遍历排序后的数字轮廓，裁剪 ROI 区域，缩放到 20×20 并填充为 28×28，进行张量转换与归一化。

数字预测：将单个数字张量输入模型（开启torch.no_grad()关闭梯度计算，提升推理效率），获取预测结果并转换为字符串。

结果输出：拼接所有数字的预测结果，打印最终预测学号，完成识别任务。

五、实验结果与分析

激活shy环境，之后运行代码.检测到本地有MNIST数据集，不再下载。开始训练。

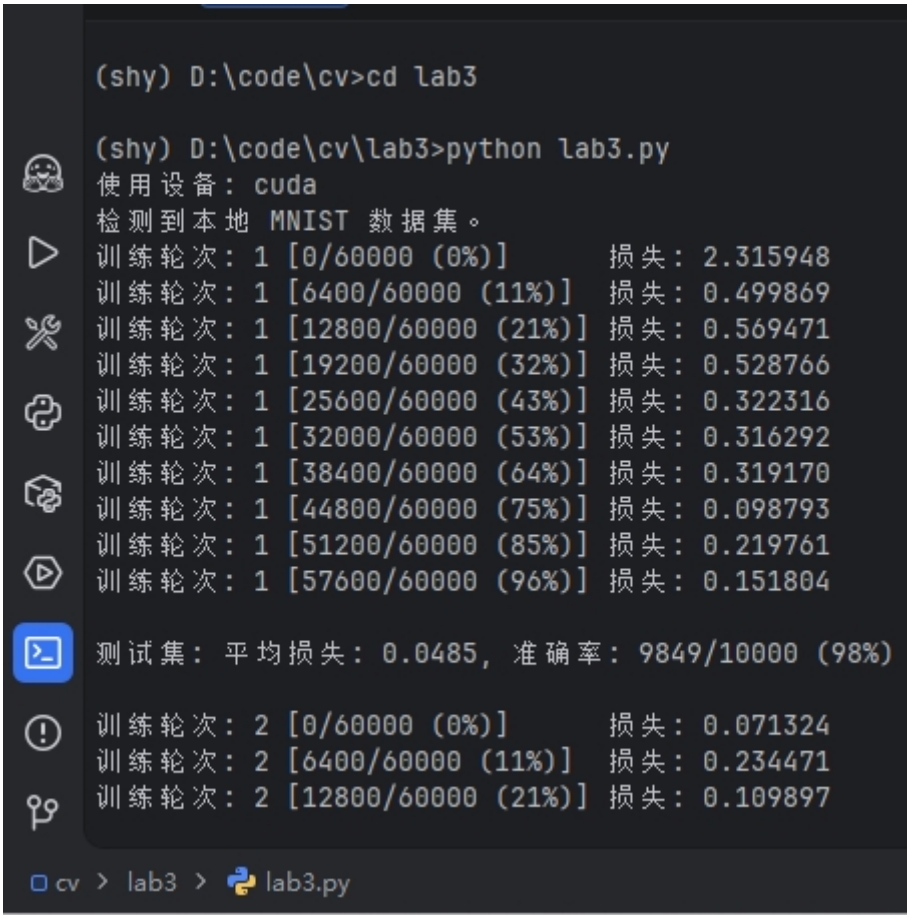

图3 训练过程

可以看到使用显卡（2000Ada）进行了训练

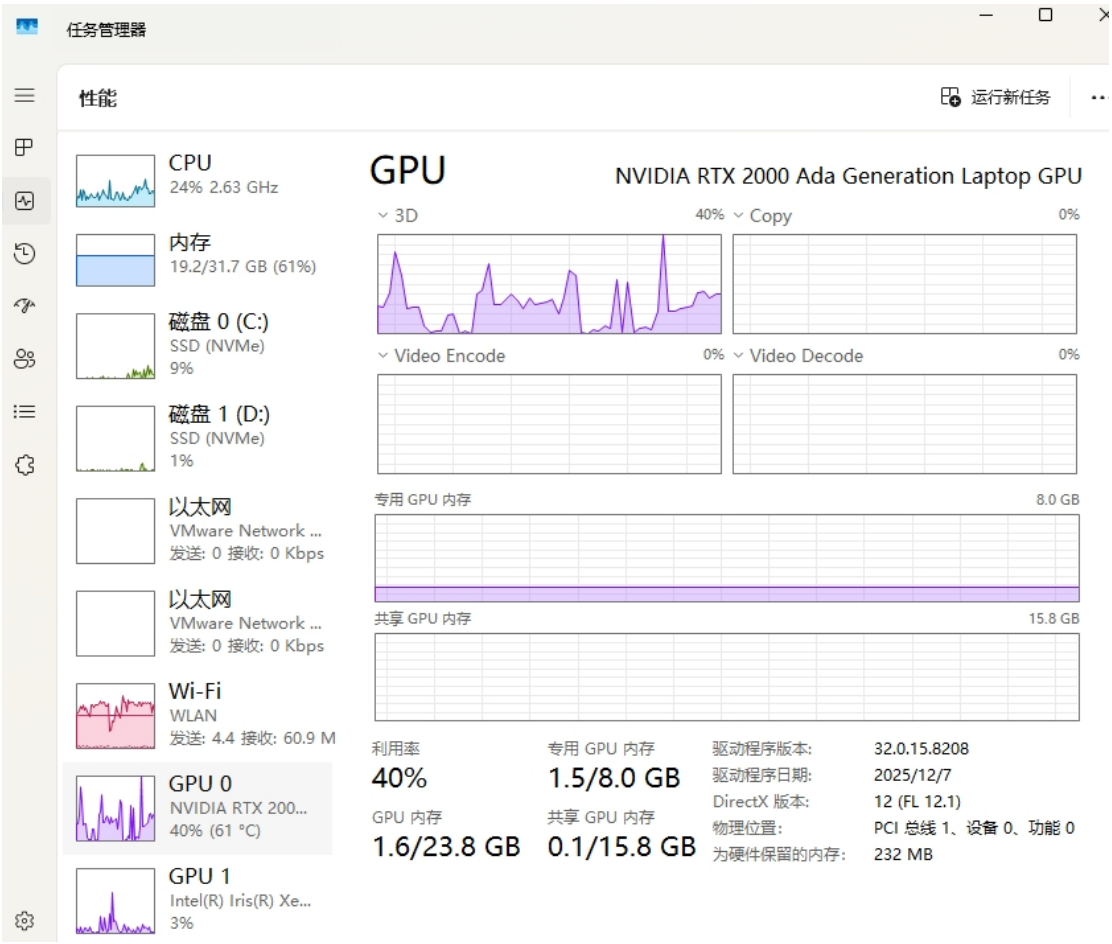

图4 训练占用资源截图

一开始设计的只有3轮，但是手写数字不同于平板上写或是打印电子字体，纸张上容易有噪声，把我学号的2023217603的7识别成了1，所以我改成训练10轮，成功识别。

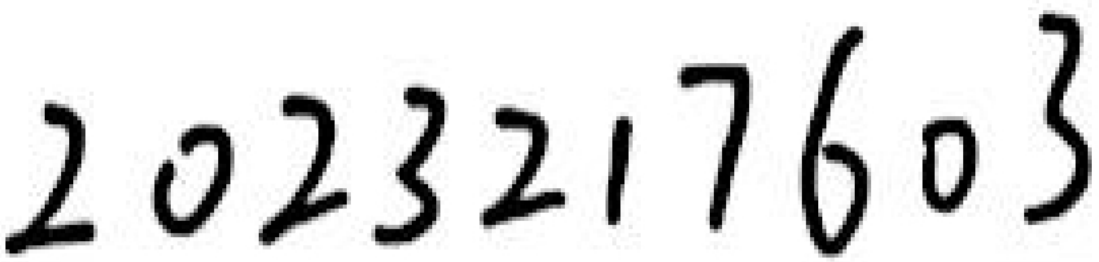

图5 输入图像

训练完成后保存模型，并且正确识别了我的手写学号。

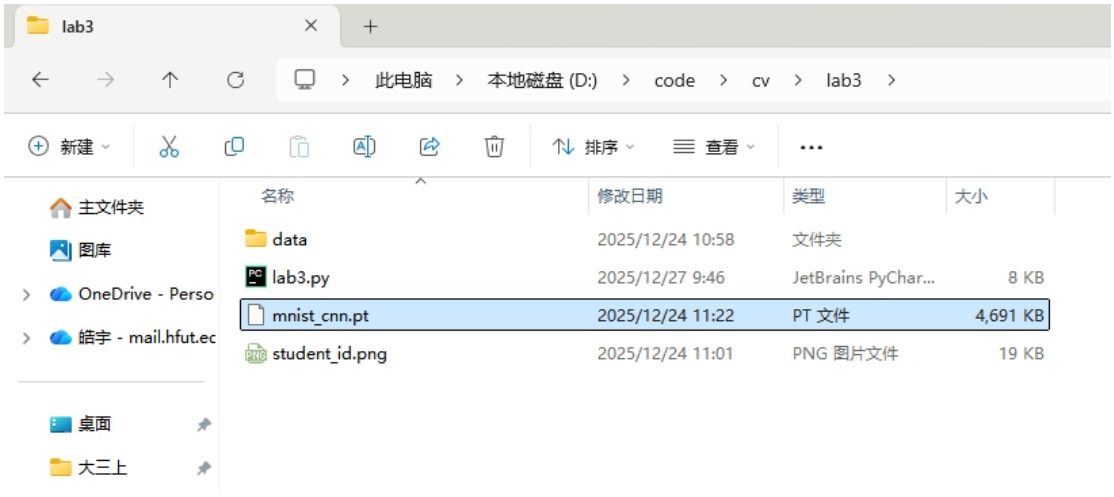

图6 保存的模型

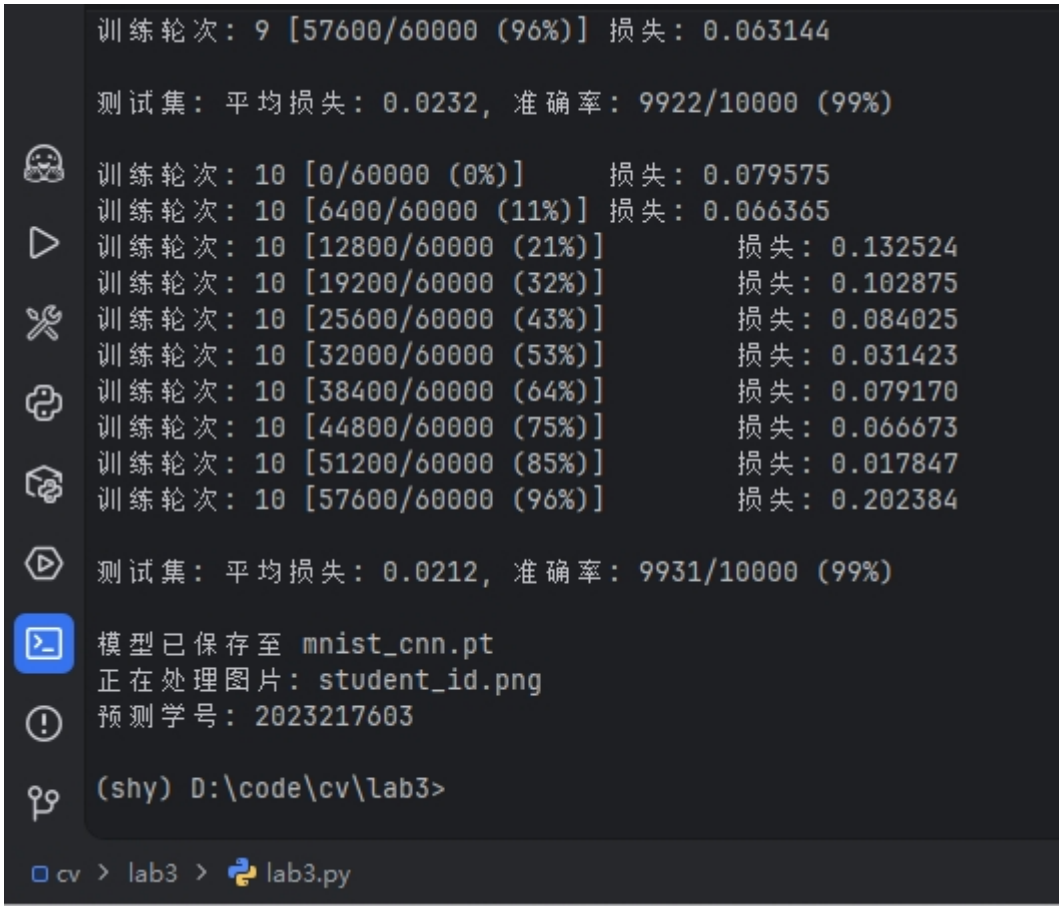

图7 识别成功

六、实验体会

本次实验通过 CNN 实现手写学号识别，让我深刻理解了 “特征提取→特征融合→分类决策” 的深度学习核心逻辑。CNN 的卷积层能够自动学习手写数字的低级特征（边缘、笔画）和高级特征（数字结构），无需手动设计特征提取器，这是其优于传统机器视觉方法的关键。同时，数据增强（随机旋转、平移）和正则化（Dropout）的作用尤为明显 —— 通过数据增强，模型能够适应手写数字的微小形变（如倾斜、位置偏移）；通过 Dropout，有效避免了模型在 MNIST 数据集上的过拟合，使测试集准确率稳定在 99% 以上。这让我认识到，深度学习模型的性能不仅取决于网络结构，合理的预处理与正则化策略同样至关重要。

学号识别的核心难点在于 “准确分割单个数字” 与 “保持数字顺序”，这两个环节直接决定了最终识别结果的正确性。实验中发现，二值化的阈值选择、轮廓面积的过滤阈值对分割效果影响极大：若二值化阈值不当，会导致数字笔画断裂或背景噪声过多；若轮廓面积阈值过小，会保留纸张污渍等噪声轮廓，干扰识别；若未对轮廓按 x 坐标排序，识别结果会出现顺序错乱。通过调整这些参数，最终实现了单个数字的精准分割与顺序排列，这让我明白，机器视觉任务中 “预处理” 是连接原始图像与模型输入的桥梁，其效果直接影响后续模型推理的准确性。

本次实验熟练掌握了 PyTorch 框架的核心用法，包括模型定义（nn.Module）、数据加载（DataLoader）、训练流程（前向传播→反向传播→参数更新）、模型保存与加载。PyTorch 的动态计算图

特性让调试更加便捷，而 GPU 加速功能使训练效率提升数倍。此外，学习率调度器（StepLR）的使用让我理解了 “动态调整学习率” 的重要性 —— 固定学习率可能导致模型在训练后期震荡不收敛，而逐步降低的学习率能让模型更稳定地逼近最优解。这些工程化的实践经验，为后续复杂深度学习任务（如目标检测、图像分割）奠定了坚实基础。

实验加分项要求的自定义环境配置，让我掌握了虚拟环境的创建、依赖库安装、版本兼容性验证的完整流程。