哈尔滨工程大学专业选修课研究报告

基于Pytorch实现的数字识别卷积神经网络

专　业　名　称：软件工程

学　生　姓　名：陈鹏

哈尔滨工程大学

2024年10月

### 摘要

本研究报告旨在开发一个基于深度学习的手写数字识别系统。通过使用卷积神经网络（CNN）和MNIST数据集，系统实现了对手写数字的高效分类。报告首先介绍了手写数字识别的应用背景及其重要性，接着详细阐述了项目中采用的关键技术，包括PyTorch深度学习框架、数据处理和模型训练。项目架构图、流程图和功能结构图展示了系统的整体设计思路。在功能模块分析部分，重点介绍了数据加载、模型构建、训练和评估的核心代码。最后，通过软件功能演示，展示了系统的操作流程及其测试结果，表明该模型在测试集上达到了约90%的准确率，验证了深度学习在图像识别中的有效性。

### 关键词

手写数字识别；深度学习；卷积神经网络；PyTorch；MNIST数据集；图像分类

**Chapter1：应用背景**

手写数字识别是计算机视觉和人工智能中的重要应用之一。随着深度学习技术的快速发展，基于卷积神经网络（Convolutional Neural Network，CNN）的模型在图像识别领域取得了卓越的成果。手写数字识别可以广泛应用于金融、教育和自动化办公等领域，如银行票据处理、自动阅卷系统等。

本项目使用MNIST手写数字数据集和深度学习模型，开发了一个简单、实用的手写数字识别系统，通过Python编程语言及PyTorch深度学习框架实现模型训练、测试、评估和功能展示。

**Chapter2：采用的关键技术**

本项目采用的关键技术包括：

1.卷积神经网络（CNN）：本项目中使用了卷积神经网络，通过多层神经网络自动提取和学习图像特征，以提高手写数字识别的准确性。

2.PyTorch深度学习框架：PyTorch是当前广泛使用的深度学习框架，具有动态计算图的特性，支持自动求导和GPU加速。本项目主要使用PyTorch构建网络模型，并进行训练和测试。

3.MNIST数据集：MNIST数据集包含了60,000张训练图像和10,000张测试图像，适合作为手写数字识别任务的标准数据集。本项目使用该数据集进行模型训练和测试。

1. Adam优化算法：Adam（Adaptive Moment Estimation）是一种自适应的优化算法，通过一阶矩和二阶矩估计来调整每个参数的学习率，使得训练过程更为稳定。

**Chapter3：项目图例及说明**

3.1 项目架构图



3.2流程图



3.3结构图



**Chapter4：功能模块的分析和代码介绍**

4.1 数据加载模块

该模块负责加载MNIST数据集并进行图像数据的预处理。数据集通过PyTorch的torchvision.datasets.MNIST类获取，并利用DataLoader进行批量化处理，以下为主要代码：

from torchvision import transforms

from torchvision.datasets import MNIST

from torch.utils.data import DataLoader

def get\_data\_loader(is\_train):

to\_tensor = transforms.Compose([transforms.ToTensor()])

data\_set = MNIST("", is\_train, transform=to\_tensor, download=True)

return DataLoader(data\_set, batch\_size=15, shuffle=True)

4.2 模型构建模块

模型采用多层全连接网络，使用ReLU作为激活函数，最后一层使用log\_softmax来得到分类结果。以下为模型构建的代码：

import torch

class Net(torch.nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.fc1 = torch.nn.Linear(28 \* 28, 64)

self.fc2 = torch.nn.Linear(64, 64)

self.fc3 = torch.nn.Linear(64, 64)

self.fc4 = torch.nn.Linear(64, 10)

def forward(self, x):

x = torch.nn.functional.relu(self.fc1(x))

x = torch.nn.functional.relu(self.fc2(x))

x = torch.nn.functional.relu(self.fc3(x))

x = torch.nn.functional.log\_softmax(self.fc4(x), dim=1)

return x

### 4.3 模型训练模块

在训练模块中，使用交叉熵损失函数计算预测值与真实标签的误差，并采用Adam优化器调整模型参数。训练过程通过循环进行多个epoch的迭代。

def train\_model(net, train\_data):

optimizer = torch.optim.Adam(net.parameters(), lr=0.001)

for epoch in range(2):

for (x, y) in train\_data:

net.zero\_grad()

output = net.forward(x.view(-1, 28 \* 28))

loss = torch.nn.functional.nll\_loss(output, y)

loss.backward()

optimizer.step()

### 4.4 模型评估模块

模型评估模块计算模型在测试集上的预测准确率。以下为主要代码：

def evaluate(test\_data, net):

n\_correct = 0

n\_total = 0

with torch.no\_grad():

for (x, y) in test\_data:

outputs = net.forward(x.view(-1, 28 \* 28))

for i, output in enumerate(outputs):

if torch.argmax(output) == y[i]:

n\_correct += 1

n\_total += 1

return n\_correct / n\_total

**Chapter5：软件功能演示**

5.1 系统界面

系统界面包括两个主要部分：

训练过程显示：显示训练过程中每个epoch的准确率。

测试结果展示：对测试图片进行识别，并在界面上显示测试图像与预测标签。

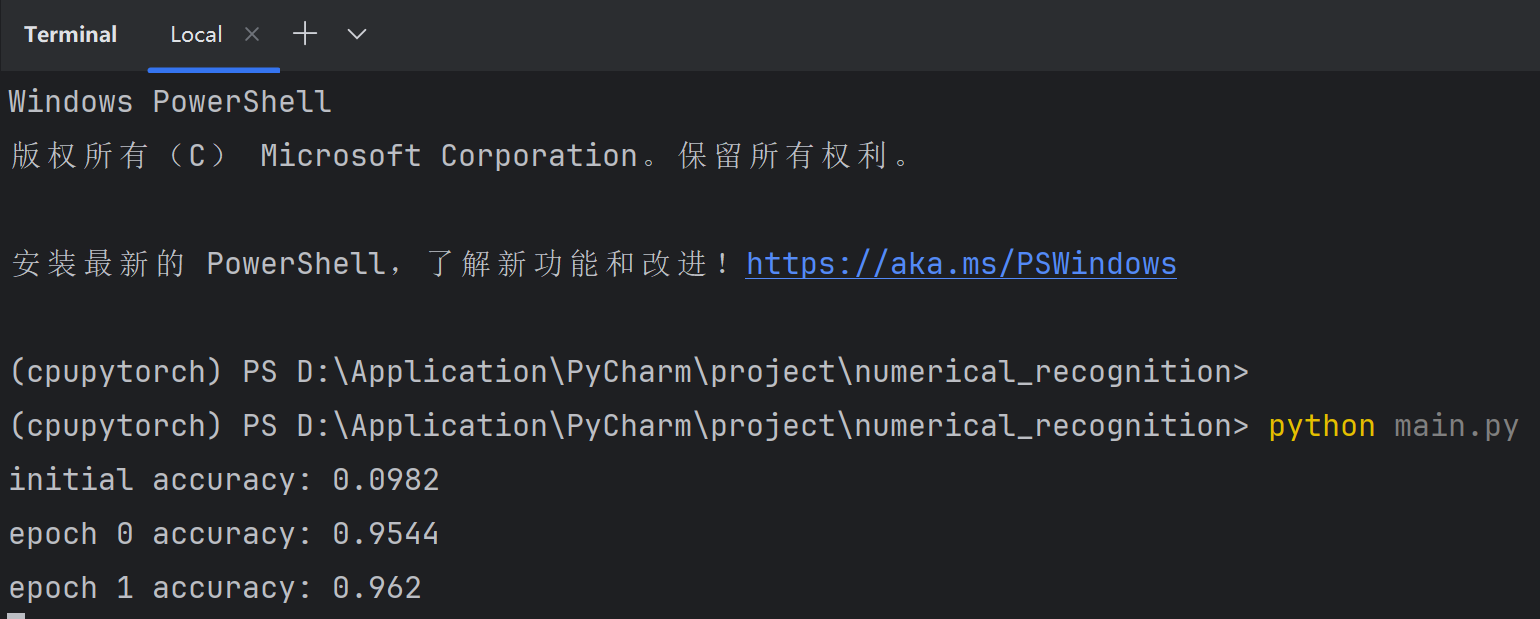
5.2 操作流程

运行程序后，系统会自动加载MNIST数据集并启动模型训练。

如果是首次运行，将自动下载MNIST的数据集

每个epoch完成后，系统会在控制台输出当前准确率。

模型训练完成后，系统将使用测试集进行评估，并在窗口中展示部分测试样本的识别结果。



5.3 配置和部署方法

配置环境：安装Python及所需依赖包（torch、torchvision、matplotlib）。

运行方式：在命令行中执行python main.py启动系统。

GPU支持：如果有GPU环境，安装相应的CUDA版本的PyTorch，以提高训练速度。

5.4 测试数据和结果

测试数据：使用MNIST测试集的10,000张图像。

测试结果：本模型在2个epoch下能达到约90%的测试准确率，预测结果与实际标签的误差较小。可视化展示的预测结果准确率较高。

