《人工智能软件开发与实践》

（ 2023 学年 秋季 学期）

作

业

报

告

学　　号：____

姓　　名：__

班　　级：____

任课教师：_____

作业报告

实验名称：　使用卷积神经网络进进行写数字识别

成绩：

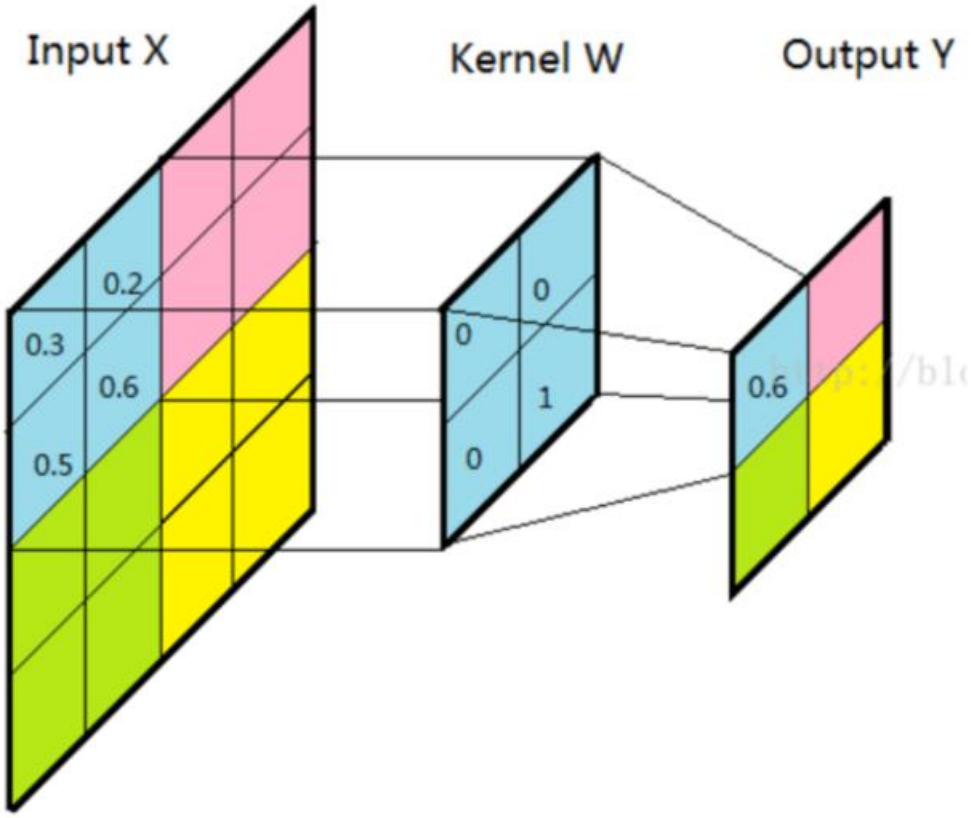
实验类别： 验证/综合型实验　　**实验要求：** 1 人 1 组　**时间：** 2023 年 9 月 6 日

一、　实验目的

使用Pytorch 构建一个简单的卷积神经网络：CNN，并完成一个简单的手写数字识别任务。

二、实验内容

卷积神经网络（英语：Convolutional Neural Network，缩写：CNN）是一种前馈神经网络，它的人工神经元可 以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。 卷积神经网络由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池 化层（pooling layer）。这一结构使得卷积神经网络能够利用输入数据的二维结构。与其他深度学习结构相比， 卷积神经网络在图像和语音识别方面能够给出更好的结果。



设计思路：

① 使用构造 torchvision.datasets 装载 mnist 数据集

② 构建 2 层 CNN 网络

③ 第一层：输入 1 维，输出 32 维，卷积核 2*2

④ 第二层：输入 32 维，输出 64 维，卷积核 2*2

⑤ 迭代训练，到收敛

⑥ 计算封闭测试精度和开发测试精度

三、使用的算法名称（若无，可以不填）

四、程序源码（拷贝至此处，同时作为附件和报告再一份单独的程序）

a） 装载数据

b） 构建多层感知器

为程序按照功能块添加注释（将带有注释的程序，粘贴至此处）

```
import torch
import torch.nn as nn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.utils.data import Dataset

# 装载 mnist 数据集
train_dataset = dsets.MNIST(root='./data', train=True,
transform=transforms.ToTensor(), download=True)
test_dataset = dsets.MNIST(root='./data', train=False,
transform=transforms.ToTensor())

# 卷积核大小
kernel_size1 = 3
kernel_size2 = 3


# 构建 2 层 CNN 网络
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.layer1 = nn.Sequential(    # 第一层卷积核
            nn.Conv2d(1, 32, kernel_size=kernel_size1, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.layer2 = nn.Sequential(    # 第二层卷积核
            nn.Conv2d(32, 64, kernel_size=kernel_size2, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc = nn.Linear(7 * 7 * 64, 10)     # 全连接层
```

```
    def forward(self, x):   # 前向函数
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.reshape(out.size(0), -1)  # 将张量改变形状以便进入全连接层进行处理
        out = self.fc(out)
        return out


# 超参数
num_epochs = 5
batch_size = 64
learning_rate = 0.001

# 数据加载器
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)

# 定义模型为 cnn，loss 计算标准为 CrossEntropy 和优化器
cnn = CNN()
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(cnn.parameters(), lr=learning_rate)

total_step = len(train_loader)  # 确定步数
for epoch in range(num_epochs): # 迭代
    for i, (images, labels) in enumerate(train_loader):
        outputs = cnn(images)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if (i + 1) % 100 == 0:
            print('Epoch [{}/{}], Step [{}/{}], Loss: {:.4f}'
                  .format(epoch + 1, num_epochs, i + 1, total_step, loss.item()))

# 计算封闭测试精度和开发测试精度
with torch.no_grad():
    correct_train = 0
    total_train = 0
    correct_test = 0
    total_test = 0
```

```
for images_train, labels_train in train_loader:
    outputs_train = cnn(images_train)
    _, predicted_train = torch.max(outputs_train.data, 1)
    total_train += labels_train.size(0)
    correct_train += (predicted_train == labels_train).sum().item()

for images_test, labels_test in test_loader:
    outputs_test = cnn(images_test)
    _, predicted_test = torch.max(outputs_test.data, 1)
    total_test += labels_test.size(0)
    correct_test += (predicted_test == labels_test).sum().item()

print('封闭测试精度: {:.2f}%'.format(100 * correct_train / total_train))
print('开放测试精度: {:.2f}%'.format(100 * correct_test / total_test))
```

五、程序运行结果（将程序运行结果的截图拷贝至此处，或者填写实验结果）

对隐层层数、隐层维度、Batch 大小、Epoch 次数等超参进行调参（不需要穷举，只需要填写下面的表格即可，保证每个超参至少取 2 个不同的值），填写对应的封闭测精度和开放测精度。（表格行数不够，可以自行添加）

序号	第一层 卷积核	第一层 卷积核	Batch 大小	封闭精度	开放精度
1	2*2	2*2	100	98.98%	98.33%
2	2*2	3*3	64	99.11%	98.75%
3	3*3	2*2	64	99.22%	98.79%
4	3*3	3*3	64	99.57%	99.00%

六、心得体会和遇到的困难

Python 使用还欠缺熟练度，对机器学习、人工神经网络的基础知识掌握有所欠缺。