成 绩 _____

北京航空航天大學

BEIHANG UNIVERSITY

实验二 手写数字识别

实验报告

院（系）名称	高等理工学院
专 业 名 称	自动化
学 生 学 号	18376251
学 生 姓 名	乐祥立
指 导 教 师	李阳

2021 年 6 月

实验二　手写数字识别

一、实验目的

利用理论课所学的 BP 网络实现手写数字识别

二、实验过程与结果

1. 获取 MNIST 数据

MNIST 手写数据集的获取方式有很多种，主要可以从深度学习框架 Pytorch 或 Tensorflow 中下载，也可以通过手动下载的方式获取。通过学习框架下载的方式可以快速获得已经分好类别的训练集和测试集，训练数据和标签数据。

本文使用 Pytorch 深度学习框架进行数据的下载，具体的下载方式为：

```
train_set = torchvision.datasets.MNIST(
    root=path, train=True,
    transform=torchvision.transforms.ToTensor(), download=True
)
train_loader = dataloader.DataLoader(
    dataset=train_set, shuffle=True, batch_size=train_size
)
```

图 1 获取 MNIST 数据

train_set 获得的是训练数据，第一次使用会进行下载，如果已经存在有文件，则不会再次进行下载。train_loader 会将获得的数据进行打包，方便进行训练，其中 shuffle 功能是将获得的数据打乱顺序。

此外，也可以使用手动获取数据的方式来得到数据集，并通过相关的函数来获取本地的数据并将数据进行划分和标准化。

2. 构建 BP 神经网络

BP 神经网络指的是反向传播网络，即误差反向传播的多层前馈网络。因此在本实验中搭建了使用全连接层的 BP 神经网络和基于 CNN 的 BP 神经网络。其中使用全连接层的 BP 神经网络由输入层、隐藏层、和输出层构成。其中输入层的维度是[batch_size, 784]，隐藏层由两层构成，他们的维度分别是[batch_size, 400]和[batch_size, 200]，输出层的维度是[batch_size, 10]。输出层的 10 代表分类的个数，并且必须要满足大于等于标签中的分类个数，否则会报错。

```python
class BPNNMdel(torch.nn.Module):
    def __init__(self):
        super(BPNNMdel, self).__init__()
        self.layer1 = nn.Sequential(nn.Linear(784, 400), nn.ReLU())
        self.layer2 = nn.Sequential(nn.Linear(400, 200), nn.ReLU())
        self.layer3 = nn.Sequential(nn.Linear(200, 100), nn.ReLU())
        self.layer4 = nn.Sequential(nn.Linear(100, 10))  # 输出维度必须大于标签的维度，即最好大于分类数，否则报错

    def forward(self, img):
        img = self.layer1(img)
        img = self.layer2(img)
        img = self.layer3(img)
        img = self.layer4(img)
        return img
```

图 2 全连接层的 BP 神经网络

基于 CNN 的 BP 神经网络使用两层卷积网络和一层全连接层构成，其中第一个卷积神经网络使用 16 大小为 5 的卷积核，padding=2，使用 ReLU 激活并进行池化采样。第二个卷积网络类似，不同之处在于使用 32 个卷积核。因此输入数据的维度是[batch_size, 1,28,28]，经过第一层之后维度变为[batch_size, 16,14,14]，经过第二层之后维度变为[batch_size, 32,7,7]，之后变化为：[batch_size, 32,7,7] → [batch_size, 32 ∗ 7 ∗ 7] → [batch_size, 10]

```python
class CNNModel(torch.nn.Module):
    def __init__(self):
        super(CNNModel, self).__init__()
        self.CNN1 = nn.Sequential(
            nn.Conv2d(  # input shape (1,28,28)
                in_channels=1, out_channels=16, kernel_size=5,
                stride=1, padding=2
            ),  # output shape (16,28,28)
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2)  # output shape (16,14,14)
        )
        self.CNN2 = nn.Sequential(
            nn.Conv2d(
                in_channels=16, out_channels=32, kernel_size=5,
                stride=1, padding=2
            ),  # output shape (32,14,14)
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2)  # output shape (32,7,7)
        )
        self.out = nn.Linear(32*7*7, 10)

    def forward(self, img):
        img = self.CNN1(img)
        img = self.CNN2(img)
        img = img.reshape(img.size(0), -1)
        out = self.out(img)
        return out
```

图 3 基于 CNN 的 BP 神经网络

3. 训练构建的 BP 神经网络

对所构建的两种神经网络分别进行训练可以得到他们在训练集上损失函数和准确度随迭代次数的变化情况如下所示。

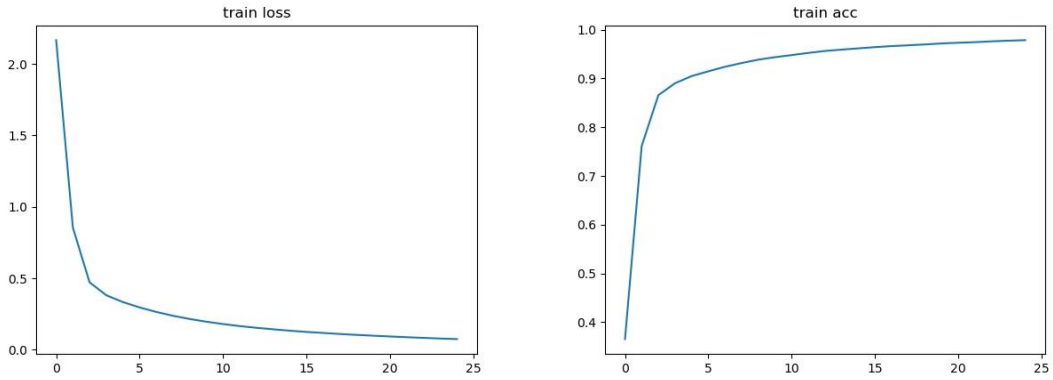

图 4 全连接层 BP 网络训练集的损失函数和准确率变化图

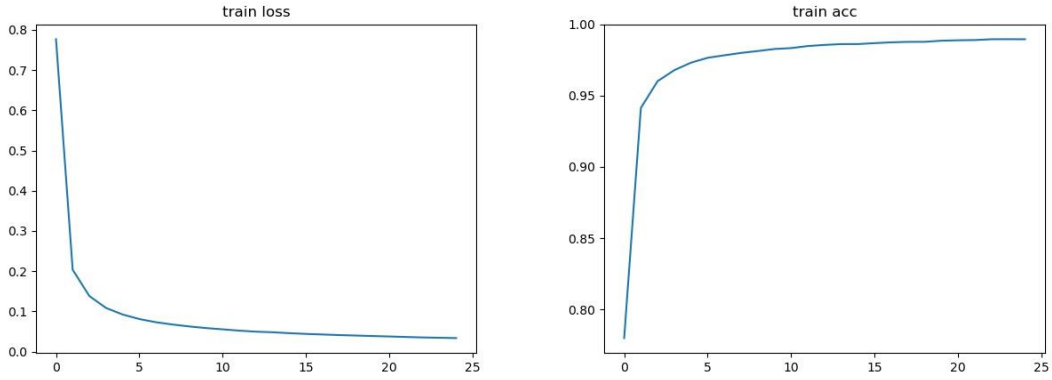

图 5 基于 CNN 的 BP 网络训练集损失函数和准确率变化图

其中，全连接层 BP 网络在经过 25 代训练之后的准确率可以达到 97.86%，而基于 CNN 的网络在同样 25 代训练之后的准确率可以达到 98.85%。

4. 测试 BP 网络

在模型训练的同时可以使用测试集进行测试来考察测试集准确度和损失的变化情况。也可以在保存模型之后再选择部分数据进行测试，考察准确度并进行可视化。

在使用两种模型结构进行训练和测试后可以得到如下的表格：

表 1 BP 网络的损失函数和准确率（训练 25 代）

	Train Loss	Train Accuracy	Test Loss	Test Accuracy
全连接层 BP	0.07437	97.86%	0.09634	97.12%

CNN_BP	0.03358	98.95%	0.03146	98.97%

测试集的损失函数和准确率变化情况如下图所示：

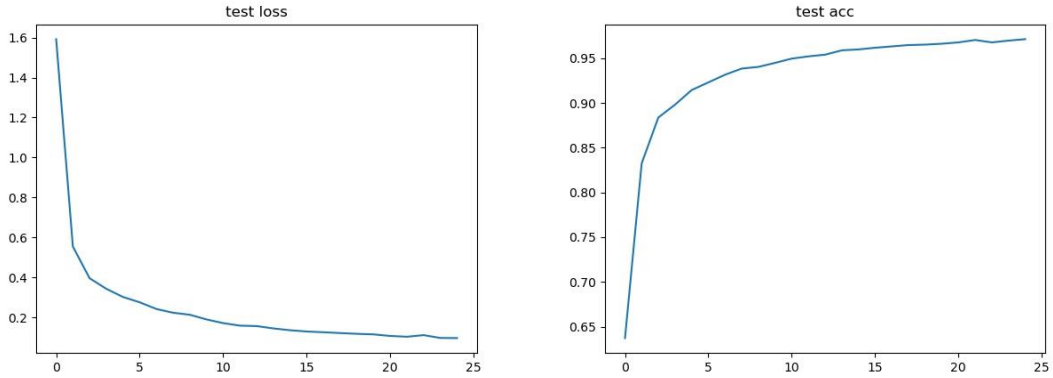

图 6 全连接层 BP 网络测试集的损失函数和准确率变化图

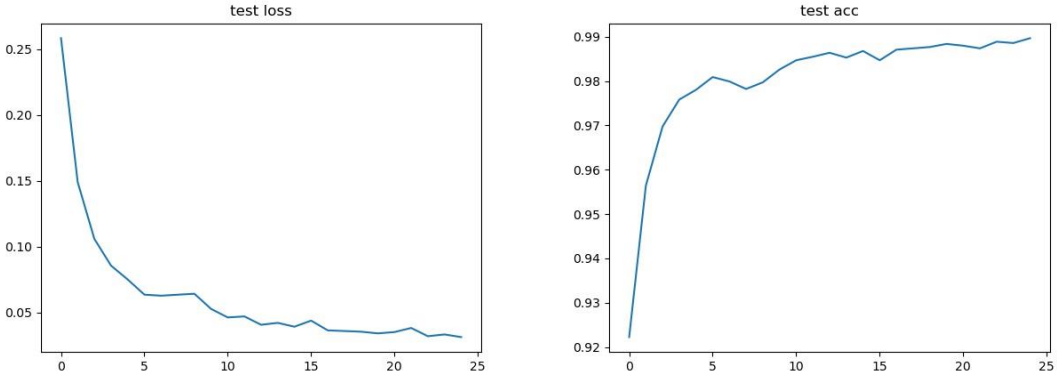

图 7 基于 CNN 的 BP 网络测试集损失函数和准确率变化图

因此可以得到网络准确率和损失函数总的变化图像为：

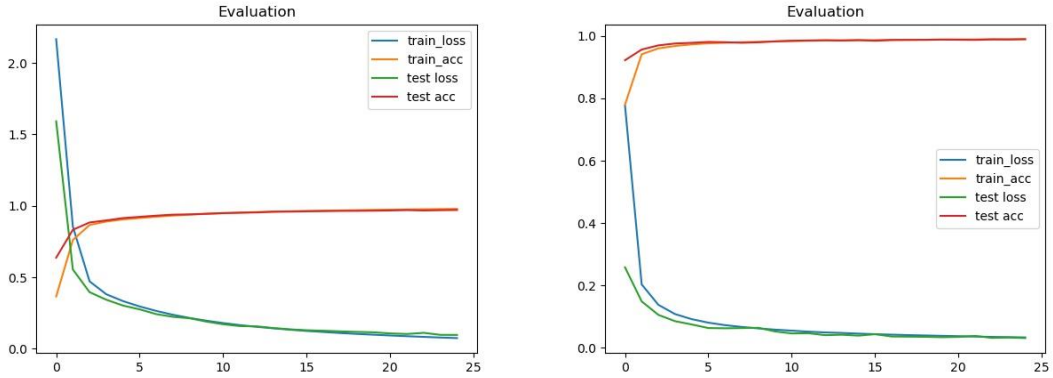

图 8 神经网络损失函数和准确率随迭代次数变化图（左：全连接 右：CNN）

训练完毕后对测试集的结果进行测试可以得到:

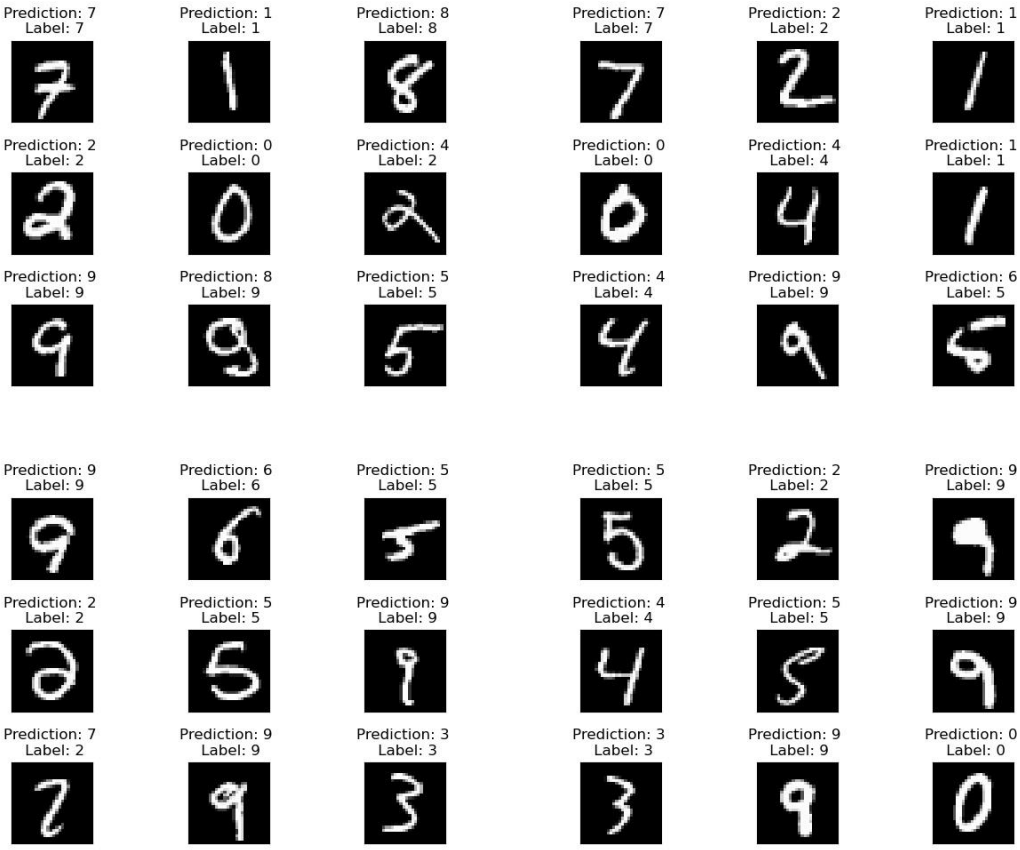

图 9 测试结果可视化

由测试结果可以发现，模型基本上可以准确识别大部分的手写数字，只有个别的数字由于特征不是非常明显甚至兼有其他数字的特征，因此被错误识别。总的来说，模型的训练效果还是不错的。

三、结果分析与实验结论

通过实验可以发现，使用 BP 网络进行训练识别手写数字在一开始基本是随机识别，但是在对整个数据集进行多次训练之后，在训练集和测试集上的准确率都能达到较高的水准并且可以看到，在前几次迭代训练的结果中，损失函数下降的非常快，准确率也快速上升。

此外，两种模型测试集上的损失函数总体上一直在下降，没有出现上升而准确率总体上一直在上升没有出现下降。因此模型并没有出现过拟合的现象，还可以继续迭代进行训练。

对比两种模型可以看到，基于 CNN 的 BP 网络的准确率均高于全连接层的 BP 神经

网络，有较好的训练效果。但是 CNN 训练的时间相较于全连接层较长，在数据较多时可能略显吃力。

四、收获、体会及建议

通过手写数字识别的实验，我对 BP 神经网络的原理和结果有了更加深刻的了解和认识；学习了如何使用深度学习框架加载数据集并进行处理和训练；对搭建神经网络的方法和神经网络相关的保存、评估等方法也有所认识。此外，还对比了两种不同的 BP 神经网络，总的收获还是非常大的。

代码链接：

https://github.com/lerlis/MNIST_BP