CNN 神经网络 MINST 手写数字识别实验报告

实验过程

搭建 CNN

本次实验使用的神经网络结构如下：

1. 两层卷积层
2. 1层 dropout 层用于正则化
3. 2层全连接
4. 使用 ReLU 作为激活函数
5. 使用 softmax 作为最后一层的激活函数，对输出做逻辑回归预测

搭建的模型如下：

```
# 搭建 CNN
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # 构建 CNN 中的相关层，包括两个 2d 卷积层、一个 dropout 层、两个全连接层
        # 最后的输出是一个十维向量，代表 [1, 10]，通过 softmax 来变化为一个和为 1 的
向量，其中值最大的就是我们识别的结果
        # 比如经过 softmax 的向量为 [0.05, 0.05, 0.1, 0.1, 0.1, 0.1, 0.1,
0.1, 0.1, 0.2]。则识别结果为 9
        self.conv1 = nn.Conv2d(1, 10, kernel_size=5)
        self.conv2 = nn.Conv2d(10, 20, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        # 通过 max_pool + relu 来进行矩阵经过神经网络的变化
        # max_pool 的 size 是 2
        x = func.relu(func.max_pool2d(self.conv1(x), 2))
        x = func.relu(func.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        # reshape 为 320，对应后面的全连接层
        x = x.view(-1, 320)
        x = func.relu(self.fc1(x))
        x = func.dropout(x, training=self.training)
        x = self.fc2(x)
        return func.log_softmax(x, dim=1)
```

训练模型

使用 `DataLoader` 所获得的 MINST 手写数字训练集进行模型训练，这里的逻辑是每次迭代都会使用测试数据对训练好的模型进行准确率的评估。

训练逻辑如下：

```
def train(epoch):
    network.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad()
        output = network(data)
        loss = func.nll_loss(output, target)
        # 反向传播计算损失函数
        loss.backward()
        optimizer.step()
        if batch_idx % log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss:
{:.6f}'.format(epoch, batch_idx * len(data),

len(train_loader.dataset),

100. * batch_idx / len(train_loader),

loss.item()))
            train_losses.append(loss.item())
            train_counter.append((batch_idx * 64) + ((epoch - 1) *
len(train_loader.dataset)))
            # 保存当前训练的模型
            torch.save(network.state_dict(), './model.pth')
            torch.save(optimizer.state_dict(), './optimizer.pth')
```

测试逻辑如下：

```
def test():
    network.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            output = network(data)
            # 计算测试的损失函数
            test_loss += func.nll_loss(output, target,
reduction='sum').item()
            # 预测结果其实就是 soft_max 中值最大的那个下标
            pred = output.data.max(1, keepdim=True)[1]
            # 计算正确率
            correct += pred.eq(target.data.view_as(pred)).sum()
    test_loss /= len(test_loader.dataset)
    test_losses.append(test_loss)
    print('\nTest set: Avg. loss: {:.4f}, Accuracy: {}/{}
({:.0f}%)\n'.format(test_loss, correct,

len(test_loader.dataset),
```

```
    100. * correct / len(

    test_loader.dataset)))
```

设定的 epoch 为 10（但是实际实验下来发现，其实在 4-5 次的时候准确率基本就达到了 98%，后面的迭代有可能会导致模型的过拟合，降低模型的泛化能力，这里可以通过降低学习率来减少每个迭代的拟合程度步长。）

> 但是降低学习率有可能落入局部最优的情况，这是需要通过不断调整参数来进行模型调整的，这里我多次调整之后感觉没有什么差别，最后还是选择了 learning_rate = 0.01 这样的学习率

测试预测

可以通过一些自己的手写数字 or MINST 提供的测试数据可视化以下我们模型的测试结果，具体的结果早起 `handwrite.ipynb` 中有可视化展示，这里就不贴图片了

实验结果

最后的实验结果是训练出来的 model 在测试集上有 98% 的准确率，能够较好的对手写数字图像进行分辨

思考

在做这个作业的过程中，其实还是有比较多疑惑的地方的，比如以下几个点是我在做实验过程中思考到的问题：

1. 如何去设计这个 CNN，他应该有哪一些层，每个层的 in_channels 和 out_channels 应该怎么设置
2. 如何去选择我们的激活函数
3. 如何去确定学习率等超参数

在查阅了一些资料之后，目前对于上面的这些问题我的理解是这样的：

1. 如何去设计一个神经网络其实是取决于这个神经网络的用途，可以去参考一些已有领域的模型设计；同时需要看这个模型是要放在一些泛化能力强的场景还是有特殊用处（即只针对一个特殊场景）服务的，这都会导致你的网络结构不同。
2. 选择激活函数其实还是要看你的用途，看你是要做逻辑回归 / 线性回归 / 分类等情况，在 MINST 手写数字的场景下其实我们要做的就是分类，所以可以用 sigmoid、softmax 这种作为最终的激活函数
3. 如何去确定我的思考是通过不断的尝试和对比实验来进行确定，只有在有对比的情况下才能够知道超参数如何设置是比较好的，这是需要**经验**和**不断的对比重复**

Reference

1. 卷积神经网络（CNN）的结构设计都有哪些思想?
2. How to design deep Convolutional Neural Networks?