20337270_钟海财_实验8

中山大学计算机学院

本科生实验报告

(2021学年春季学期)

课程名称：Artificial Intelligence

教学班级	**20级软工+网安**	**专业（方向）**	**软件工程**
学号	**20337270**	**姓名**	**钟海财**

一、 实验题目

实验任务

□ 运用pytorch框架完成中药图片分类，具体见给出的数据集和测试集。

□ 需要画出loss、准确率曲线图。

二、 实验内容

1. 算法原理

卷积神经网络（CNN）

卷积

不再是对图像中每一个像素做处理，而是对图片上每一小块像素区域做处理，加强了图片中像素的连续性，从而处理的一个图形而不是单个像素点

神经网络

神经网络是一种计算模型，由大量的神经元以及层与层之间的激活函数组成。

卷积神经网络(CNN)

属于人工神经网络的一种，它的权重共享的网络结构显著降低了模型的复杂度，减少了权值的数量。卷积神经网络可以直接将图片作为网络的输入，自动提取特征，并且对图片的变形（如平移、比例缩放、倾斜）等具有高度不变形。

torch.nn 搭建神经网络

卷积神经网络

nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, bias=True)

NOTE: PyTorch 卷积网络输入默认格式为(N, C, H, W)

其中 N 为batch 大小（输入默认batch处理），C 为图像通道数（黑白1维，彩色RGB三维），H和W分别为图像的高度和宽度。 Conv2d 的前两个参数分别为输入和输出的通道数，kernel_size为卷积核大小，stride为步长默认为1，padding 为填充默认0。 一般情况下，计算公式为

- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

网络训练一般步骤

实例化网络net = Net() 后，计算得到 Loss，并定义网络优化器

```
optim = nn.optim.Adam(net.parameters(), lr=lr)
```

在更新前，需清除上一步的梯度，即

```
optim.zero_grad()
```

然后 Loss 反向传播：

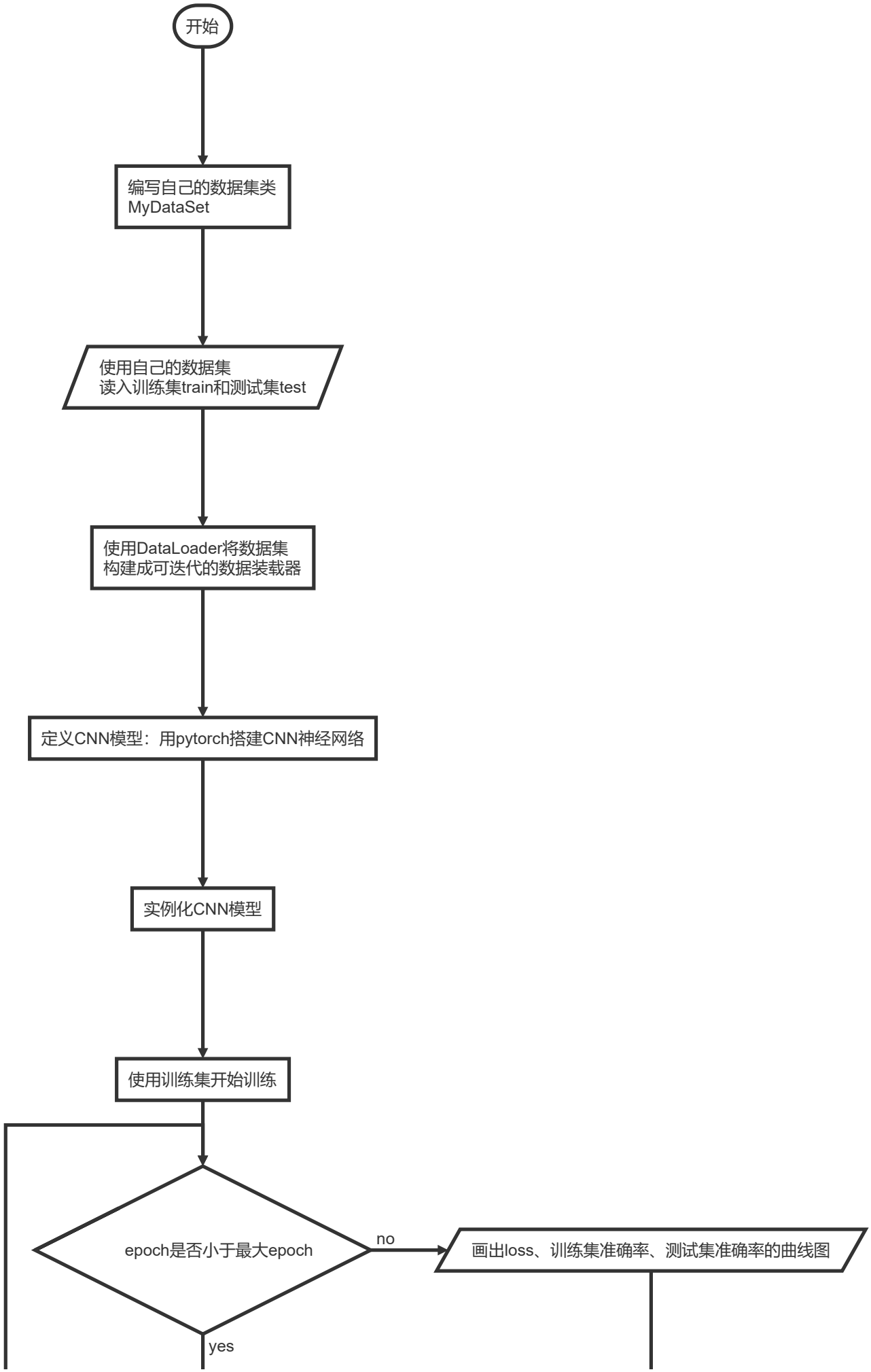
```
loss.backward()
```

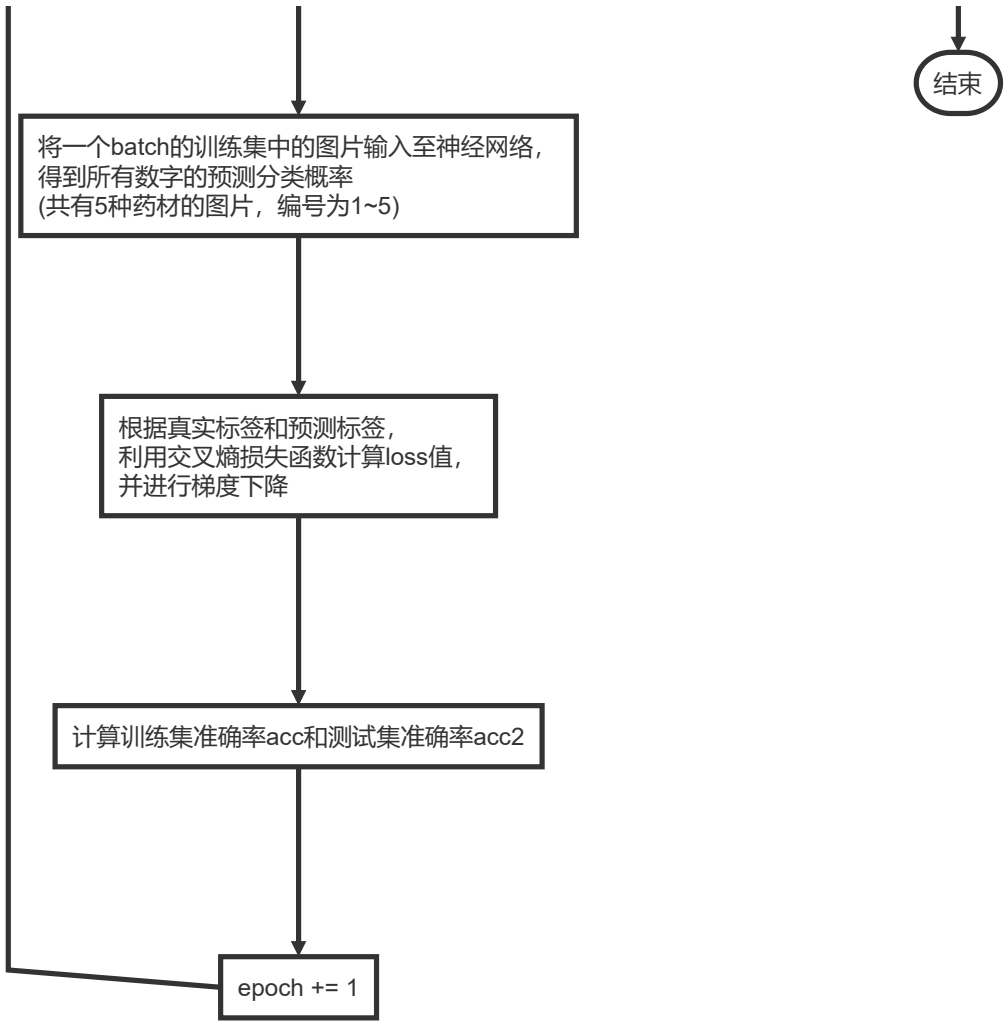
最后优化器更新：

```
optim.step()
```

2. 伪代码/流程图

流程图：


开始
编写自己的数据集类
MyDataSet
使用自己的数据集
读入训练集train和测试集test
使用DataLoader将数据集
构建成可迭代的数据装载器
定义CNN模型：用pytorch搭建CNN神经网络
实例化CNN模型
使用训练集开始训练
epoch是否小于最大epoch
no
yes
画出loss、训练集准确率、测试集准确率的曲线图



3. 关键代码展示（带注释）

3.1 我的数据集的类：

```
# 批处理大小
batch_size1 = 50
# 将每个标签转换成对应的数字，因为在使用字符串转换为张量会报错
labels = {'baihe': 1, 'dangshen': 2, 'gouqi': 3, 'huaihua': 4, 'jinyinhua': 5}


# 编写我自己的数据集的类
class MyDataSet(Dataset):
    def __init__(self, root_dir='./data', train_val='train', transform=None):
        # 得到文件夹train_val的路径
        self.data_path = os.path.join(root_dir, train_val)
        # 得到文件夹train_val下的所有照片的路径
        self.image_names = glob.glob(self.data_path + '/*/*.jpg')
```

```
        self.data_transform = transform
        self.train_val = train_val
        # print(self.image_names[0])

    def __len__(self):
        return len(self.image_names)

    def __getitem__(self, item):
        # 通过索引item得到照片的路径
        img_path = self.image_names[item]
        # print(img_path)
        # 通过照片的路径以RGB格式读取照片
        img = Image.open(img_path).convert('RGB')
        # print(img.size)
        image = img
        # 通过路径得到该照片的标签
        label = img_path.split('\\')[-2]
        # 将标签转换为对应的数字
        label = labels[label]
        # 将标签转换为张量
        label = torch.tensor(label)
        if self.data_transform is not None:
            try:
                # 对读取的照片进行处理：如大小缩放为相同，标准化
                image = self.data_transform(img)
            except:
                print('can not load image:{}'.format(img_path))
        return image, label


# 把照片数据转成tensor,并遵从正态分布
transform1 = transforms.Compose([
    # 将图像缩放成84*84大小
    transforms.Resize(84),
    transforms.CenterCrop(84),
    transforms.ToTensor(),
     # 标准化
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# 使用MyDataSet来生成训练集,测试集（手动将文件夹test转换为train的格式，命名为test1）
train_dataset = {x: MyDataSet(
    root_dir=r'D:/桌面文件/pythonAI/lab8/data',
    train_val=x,
    transform=transform1
) for x in ['train', 'test1']}
# 使用DataLoader
train_loader = DataLoader(
    train_dataset['train'],
    batch_size=batch_size1,
    shuffle=True
)
test_loader = DataLoader(
    train_dataset['test1'],
    batch_size=10,  # 只有10张照片
    shuffle=True
)
```

3.2 定义CNN模型

```
# 定义CNN模型，两层
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(  # 输入 3*84*84，batch_size = batch_size1=50
                in_channels=3,
                out_channels=16,
                kernel_size=5,
                stride=1,
                padding=2,
            ),  # 输出(16，84，84)
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),  # 输出(16，42，42)
        )
        self.conv2 = nn.Sequential(  # 输入(16，42，42)
            nn.Conv2d(16, 32, 5, 1, 2),  # 输出(32，42，42)
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),  # 输出(32，21，21)
        )
        self.out = nn.Linear(32 * 21 * 21, batch_size1)
        # 输出(32，21，21)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(-1, 32 * 21 * 21)  # (32，21，21)
        output = self.out(x)
        return output
```

3.3 实例化模型并进行训练

```
# 实例化模型
model = Net()
# 检测显卡是否可用
use_cuda = torch.cuda.is_available()
# 使用GPU加速
if use_cuda:
    model.cuda()

print(model)
# 使用交叉熵损失函数
criterion = nn.CrossEntropyLoss()
# 使用带有动量的随机梯度下降
optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
# 用于存储损失
loss_list = []
# 用于存储在训练集train中的准确率
acc = []
# 用于存储在test的10张照片中的准确率
acc2 = []
epochs = 50
# 总迭代次数
all = epochs * train_dataset['train'].__len__()
# print(all)
# 进行模型训练
```

```
for epoch in range(epochs):
    for batch, (X, y) in enumerate(train_loader):
        if use_cuda:
            X, y = X.cuda(), y.cuda()  # 使用GPU加速
        # 正向传播
        y_pred = model(X)
        # 计算损失
        loss = criterion(y_pred, y)
        # 梯度归零
        optimizer.zero_grad()
        # 反向传播
        loss.backward()
        # 更新参数
        optimizer.step()
        # 每n个batch次看下损失和准确率
        if batch % 3 == 0:  # n = 3（n<=18(902//50)，可以增大n或减小epochs以减少训练时
间）
            loss_list.append(loss.data.item())
            print("epoch=", epoch, "loss------------", loss.data.item())
            # 检验正确率
            total = 0
            correct = 0
            # 不需要计算梯度
            with torch.no_grad():
                for X, y in train_loader:   # 检验在训练集上的准确率
                    if use_cuda:
                        X, y = X.cuda(), y.cuda()
                    y_pred = model(X)
                    # 返回值有两个，第一个是最大的值，第二个是最大值的索引
                    _, predicted = torch.max(y_pred.data, dim=1)
                    total += y.size(0)
                    correct += (predicted == y).sum().item()
                acc.append((100.0 * (correct / total)))
                print("在训练集train中的准确率acc= ", acc[-1], "%")
                total = 0
                correct = 0
                for X, y in test_loader:  # 检验在测试集上的准确率
                    if use_cuda:
                        X, y = X.cuda(), y.cuda()
                    y_pred = model(X)
                    # 返回值有两个，第一个是最大的值，第二个是最大值的索引
                    _, predicted = torch.max(y_pred.data, dim=1)
                    total += y.size(0)
                    correct += (predicted == y).sum().item()
                acc2.append((100.0 * (correct / total)))
                print("在测试集test中的准确率acc2= ", acc2[-1], "%")
```

3.4 画图

```
# 显示loss曲线图，all为总迭代次数 = epochs * 训练集大小(902)
plt.plot(np.linspace(0, all, len(loss_list)), loss_list)
plt.xlabel('迭代次数', fontsize=18)
plt.ylabel('代价', rotation=0, fontsize=18)
plt.title('误差和训练Epoch数', fontsize=18)
plt.show()
# 显示在训练集train的准确率acc曲线图
plt.plot(np.linspace(0, all, len(acc)), acc)
plt.xlabel("迭代次数", fontsize=18)
plt.ylabel("acc", rotation=0, fontsize=18)
```

```
plt.title('训练集train准确率和训练Epoch数', fontsize=18)
plt.show()
# 显示在测试集test的准确率acc2曲线图
plt.plot(np.linspace(0, all, len(acc2)), acc2)
plt.xlabel("迭代次数", fontsize=18)
plt.ylabel("acc2", rotation=0, fontsize=18)
plt.title('测试集test准确率和训练Epoch数', fontsize=18)
plt.show()
```

4. 创新点&优化（如果有）

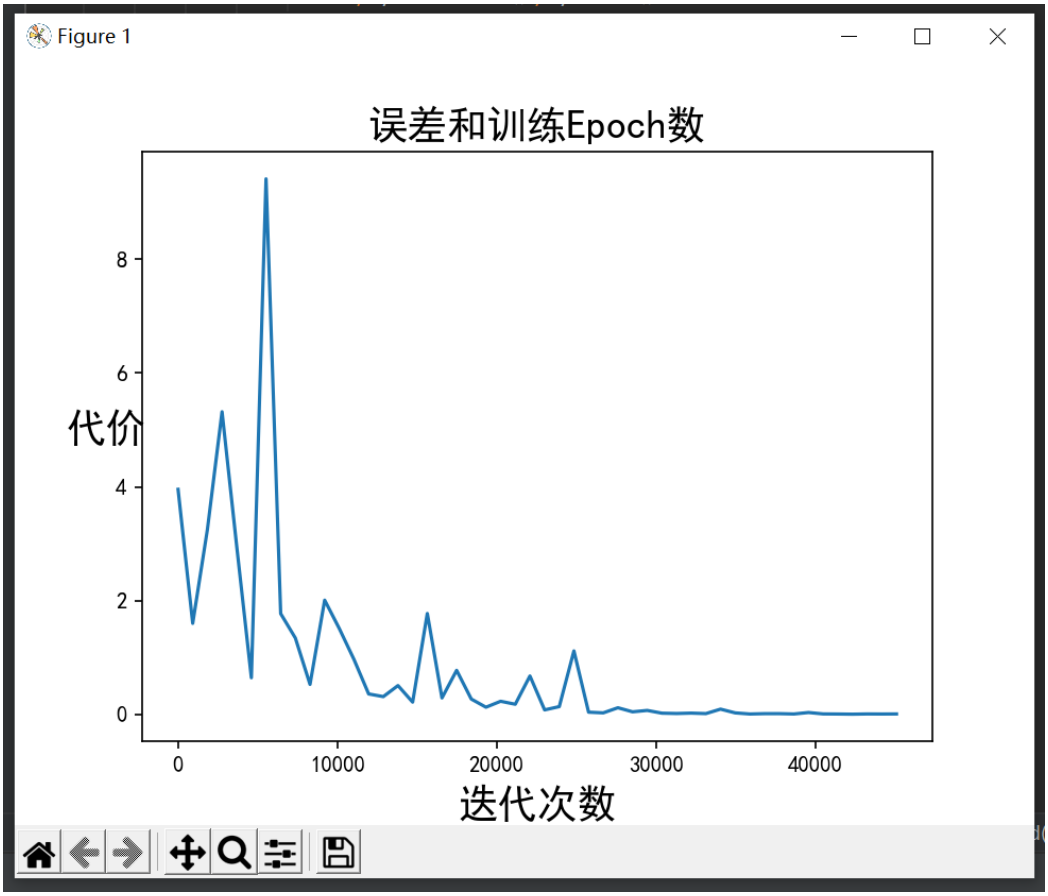
无

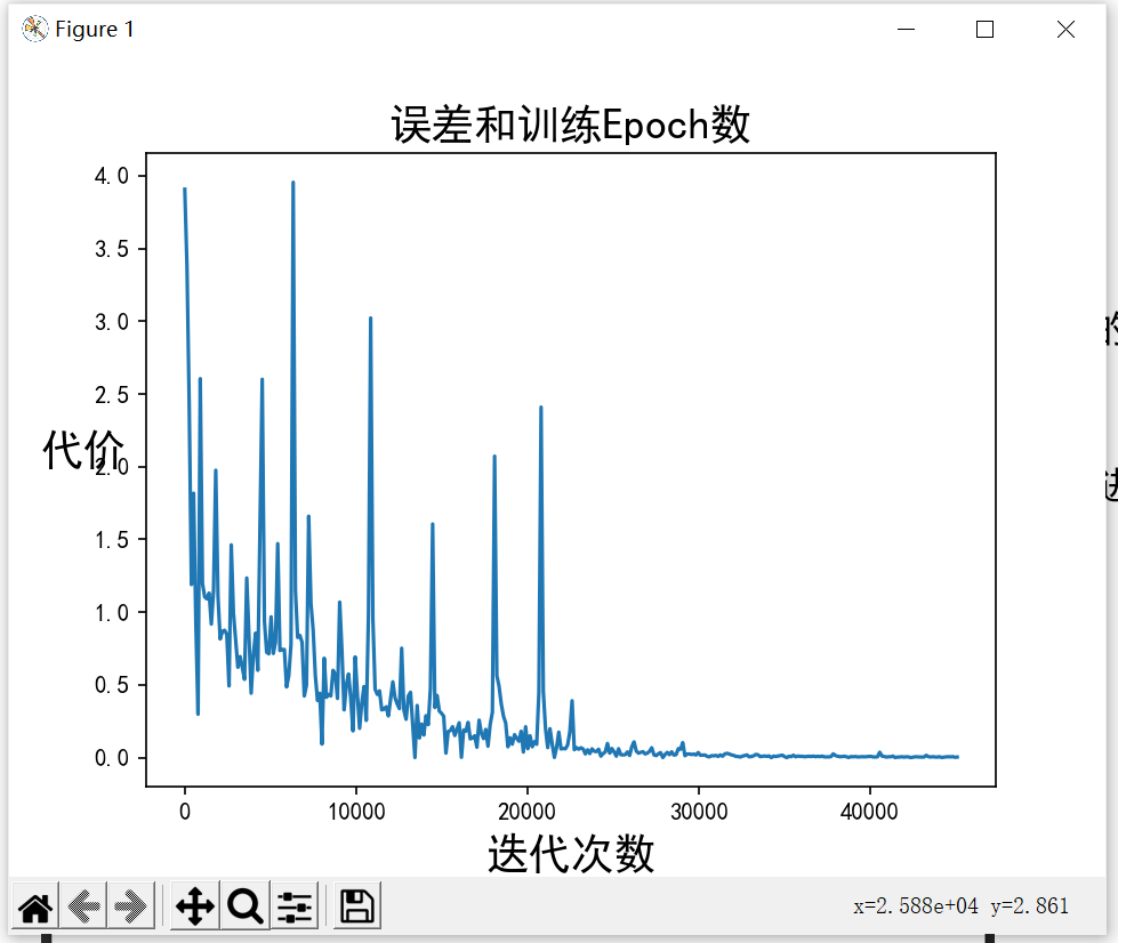
三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

迭代次数 = epoch数 * 训练集大小(902)

误差与迭代次数的关系：





训练集train准确率acc与迭代次数的关系：

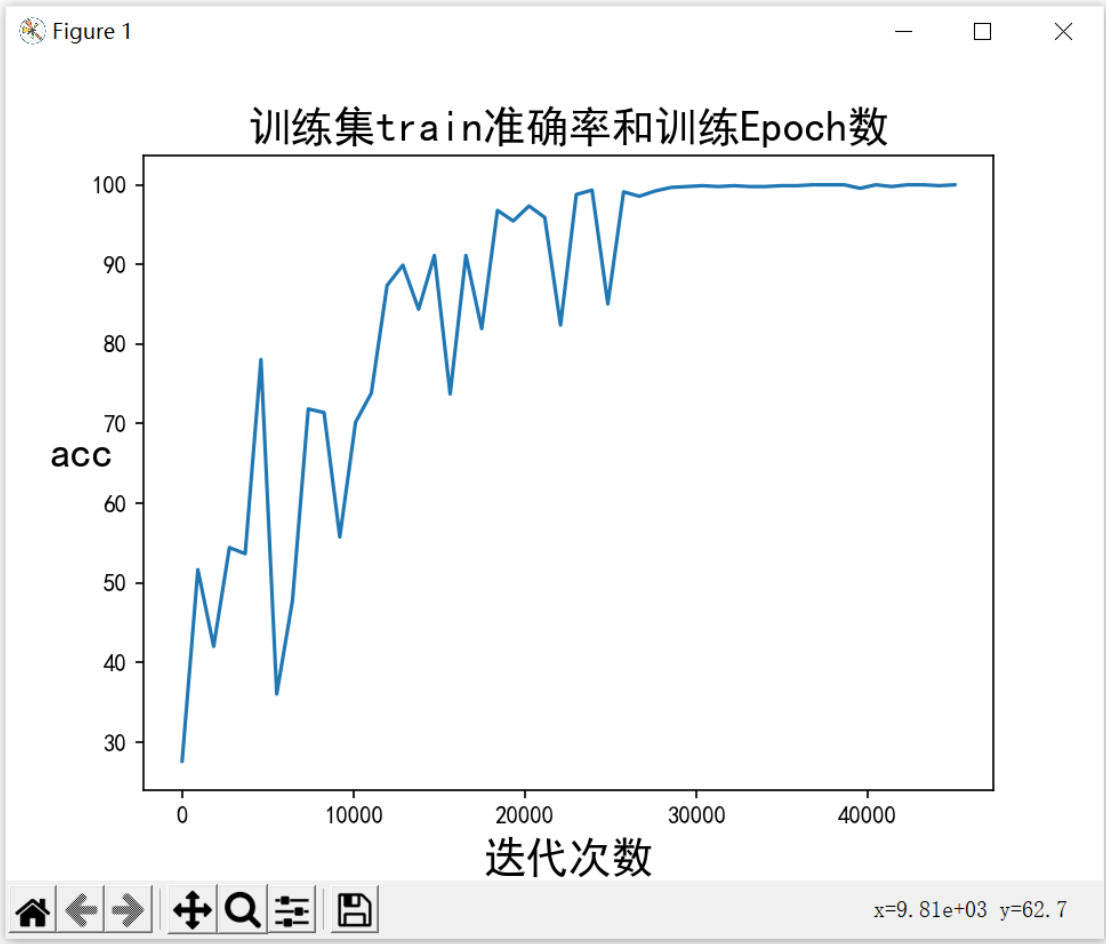


Figure 1
训练集train准确率和训练Epoch数
acc
迭代次数
100
90
80
70
60
50
40
30
0
10000
20000
30000
40000
x=9.81e+03 y=62.7

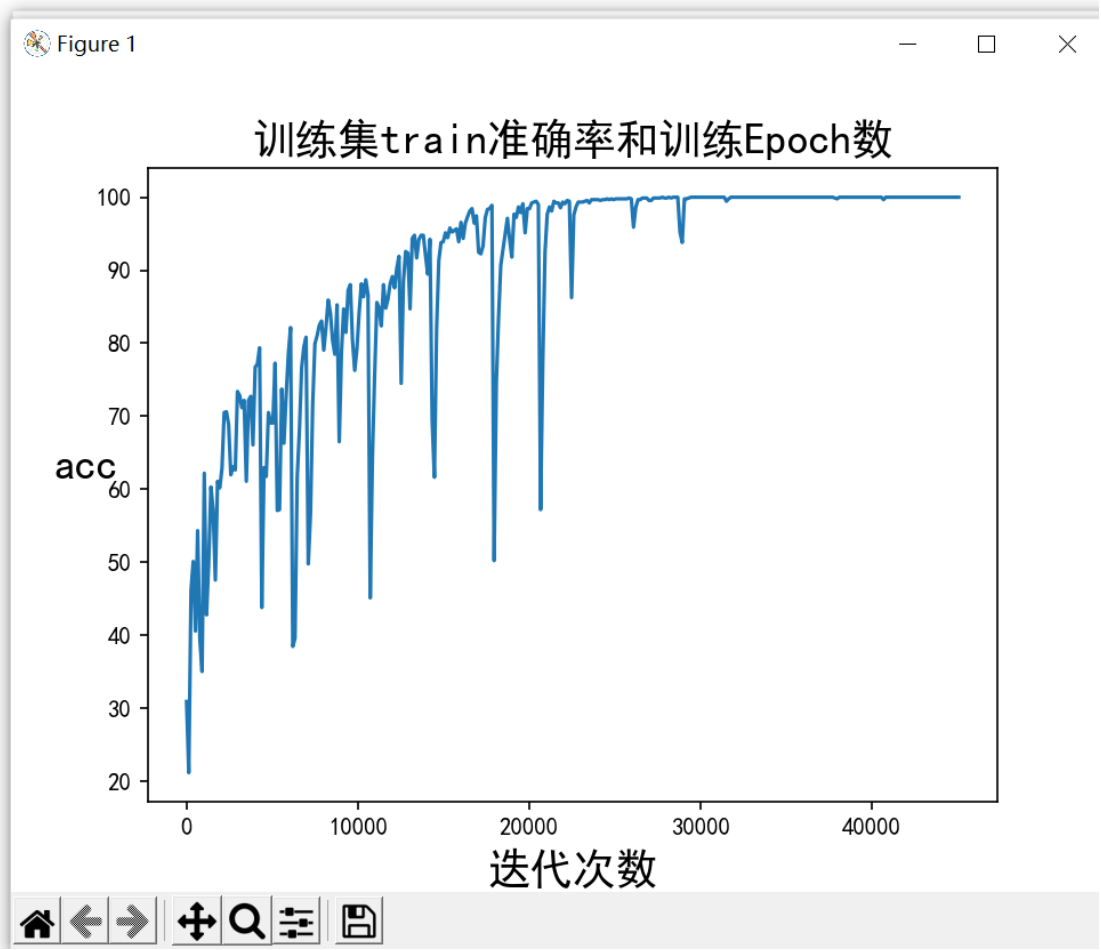
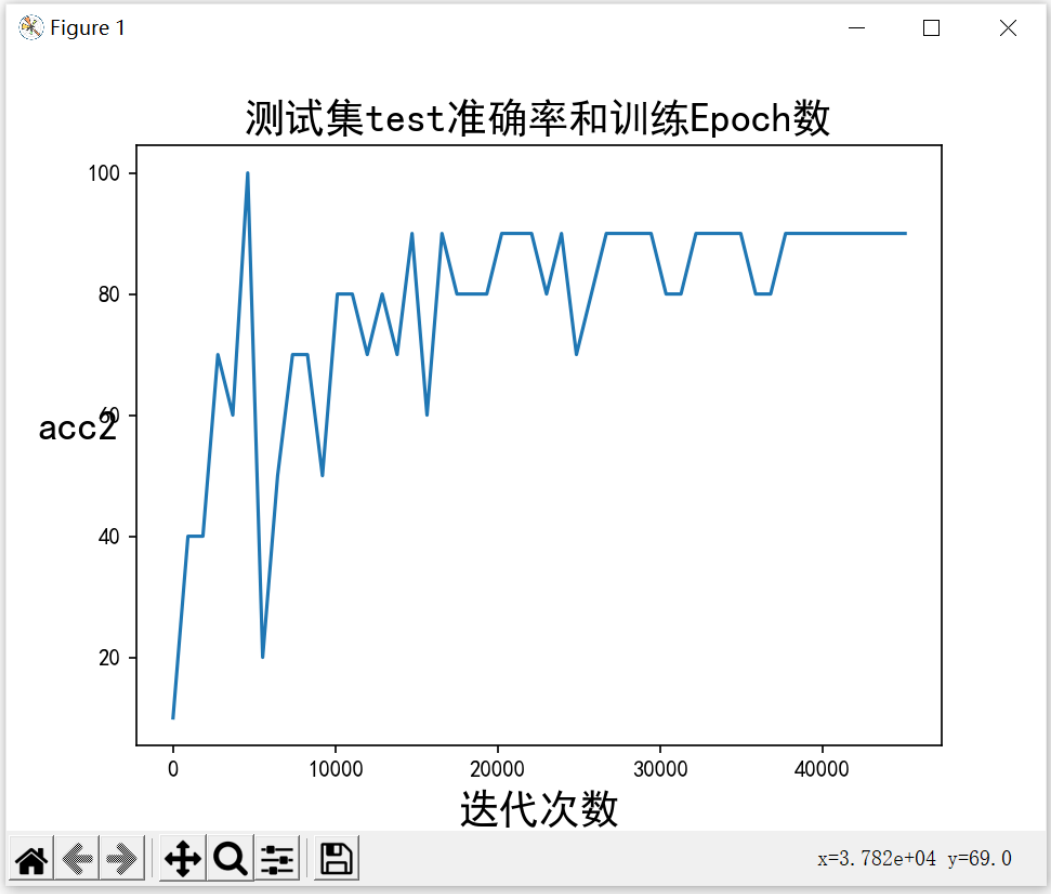
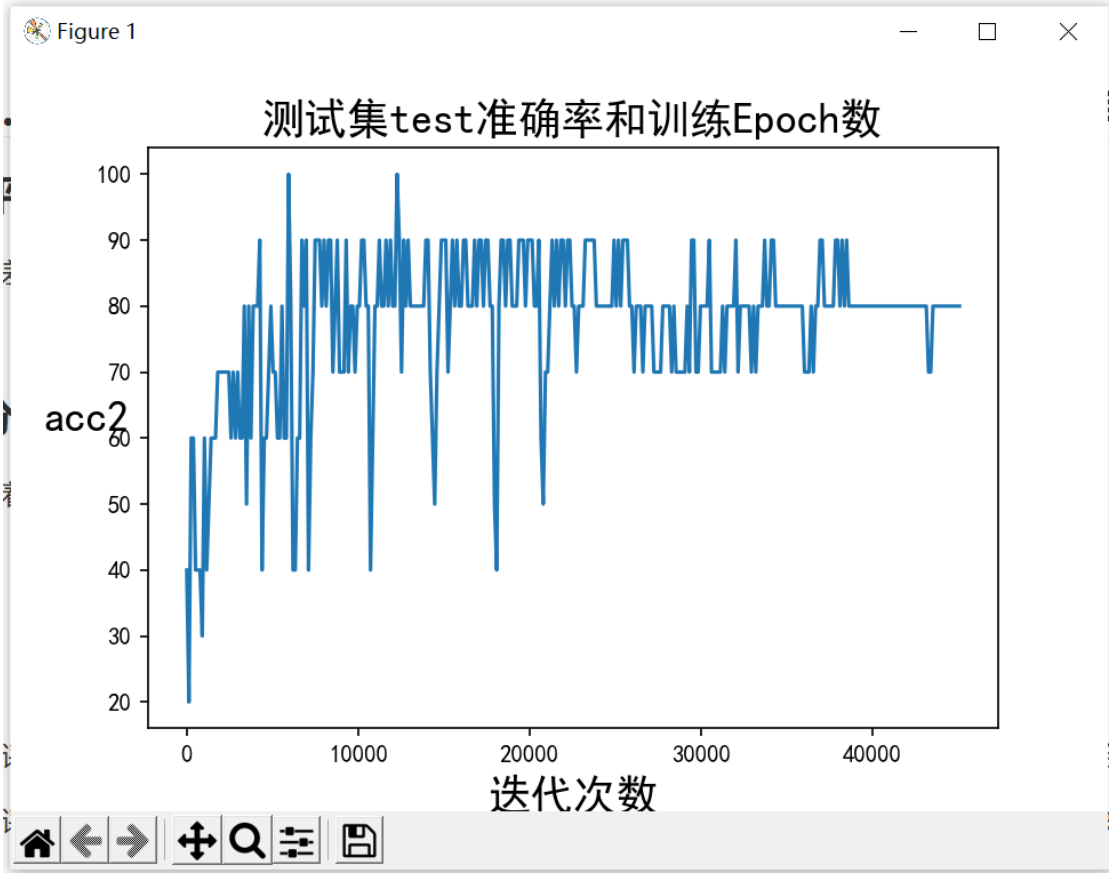


Figure 1
训练集train准确率和训练Epoch数
acc
迭代次数
100
90
80
70
60
50
40
30
20
0
10000
20000
30000
40000

测试集test准确率acc2与迭代次数的关系：





2. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

评测指标：

误差，训练集train准确率acc，测试集test准确率acc2

分析：

随着迭代次数的增加，误差，acc和acc2都在波动，最终趋于稳定：

其中误差整体趋势为减小到0，最终在迭代次数为30000后收敛到0附近；

acc整体趋势为增大到100%，最终也是在迭代次数为30000后收敛到100%附近；

acc2整体趋势为增大，最终在迭代次数为40000后收敛到90%附近.

由评测指标误差和acc可知，我们的CNN模型对于训练集train的拟合程度非常好，几何完全拟合；

由评测指标acc2可知，我们的CNN模型对于测试集test的预测拟合程度还不错，最终收敛到90%。

四、 思考题

本次实验无思考题。

五、 参考资料

1. 实验文档：14_PyTorch.pdf
2. 课本：《人工智能》(第三版) 清华大学出版社
3. 参考网站：

   pytorch实现CNN模型进行多分类(mnist)*VVeaker*的博客-*CSDN*博客pytorch textcnn多标签分类

   完整版Pytorch训练图像分类--Dataset篇 - 知乎 (zhihu.com)

   【莫烦Python】PyTorch 神经网络哔哩哔哩bilibili