中山大学计算机学院人工智能本科生实验报告

2022学年春季学期

课程名称：Artificial Intelligence

教学班级	人工智能（陈川）	专业（方向）	计算机科学与技术人工智能与大数据
学号	20337025	姓名	崔璨明

一、实验题目

运用pytorch框架完成中药图片分类。

实验要求

1. 设计合适的卷积神经网络结构，选择合适的损失函数以及优化器，利用训练集完成网络训练，并在测试集上计算准确率。
2. 需要可视化训练集上损失曲线图和测试集上准确率曲线图。
3. 需要提交一份简要报告+代码

二、实验内容

1、算法原理

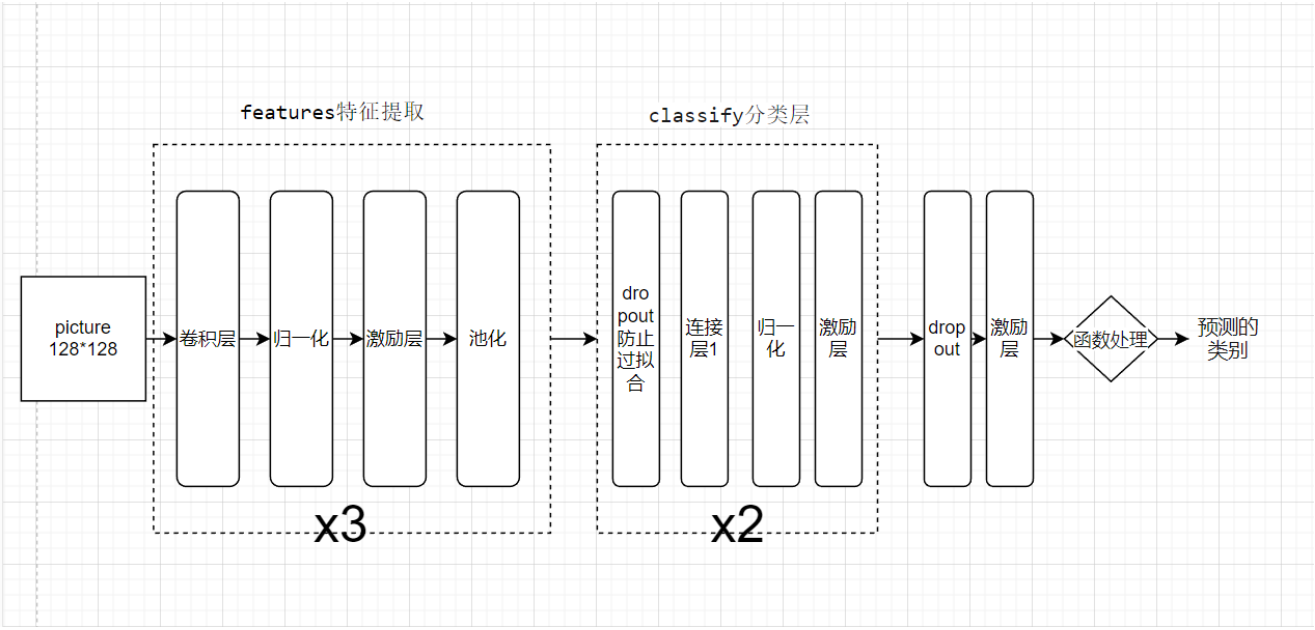
卷积神经网络（CNN）
卷积即不再是对图像中每一个像素做处理，而是对图片上每一小块像素 区域做处理，加强了图片中像素的连续性，从而处理的一个图形而不是单个像素点。在图像识别或分类任务中，卷积神经网络便是一种利用卷积层提取图像特征并进行相应的处理的神经网络，网络经过计算模型，由大量的神经元以及层与层之间的激活函数组成。

网络训练和实验过程
在该实验中，运用pytorch框架完成中药图片分类的卷积神经网络搭建和实验流程如下：

1. 读入训练集和测试集中的数字图片信息以及对图片预处理
2. 用pytorch搭建神经网络（包括卷积和全连接神经网络）
3. 将一个batch的训练集中的图片输入至神经网络，得到所有中药图片的预测分类概率
4. 根据真实标签和预测标签，利用交叉熵损失函数计算loss值，
5. 并进行梯度下降，并利用调优器进行参数调整
6. 若训练次数（epoch）未到，则返回步骤3
7. 画出loss曲线图
8. 在测试集上进行测试

2、卷积神经网络结构

我设计的卷积神经网络的结构图如下，“X2”和“X3”分别代表需要再经过2组类似结构和经过3组类似结构：



3、算法伪代码

利用训练集中的数据训练卷积神经网络的算法伪代码如下：

```
输入：待训练的神经网络model,计算交叉熵的函数criterion，训练集train_loader,对整个训练集的训练次数epochs_num
输出：训练好的模型model
def Train(epochs_num,model,trian_loader):
    #对整个训练集训练epochs_num次
    for epoch in range(epochs_num):
        for (img,label) in (a batch in trian_loader):
        #img为图像，label为该图像对应的标签
            sample=img.reshape #将图像转为符合输入的格式
            put sample into model #输入图像
            get output from model#得到模型输出的标签
            loss=_criterion(output,label) #计算损失函数交叉熵
            backward #利用梯度下降向后传播
                        update parameters  #更新参数
    return model
```

进行测试的算法伪代码算法如下：

```
输入：训练好的模型model，测试集test_loader
输出：预测结果types，正确率rate
def Test(test_loader,model):
    correct=0
    types=null
    for (img,label) in test_loader:
            output=model(img)
            types.put(output)
            if output==label:
                    correct+=1
```

```
    rate= correct/len(test_loader)
    return types,rate
```

4、损失函数以及优化器

选择交叉熵作为损失函数、采用梯度下降法反向传播进行参数优化，具体实现如下：

```
#交叉熵损失函数
criterion=nn.CrossEntropyLoss().to(device)

#定义模型优化器：输入模型参数，定义初始学习率
optim=torch.optim.Adam(model.parameters(),lr=learn_rate)

#定义学习率调度器：输入包装的模型，定义学习率衰减周期step_size，gamma为衰减的乘法因子
exp_lr_scheduler = lr_scheduler.StepLR(optim, step_size=6, gamma=0.1)
```

5、关键代码展示

首先，为了使图片的输入符合卷积神经网络的输入，我先是对训练集和测试集的中药图片进行了处理，将相同类别的中药图片放在同一个文件夹下，并编写程序`pre_work.py`生成相应的文本文件，然后调用`torch.utils.data`库中的`Dataset`类和`Dataloader`方法载入图片并进行预处理（提取中心、统一尺寸、转化成张量、归一化），得到训练集`train_loader`和测试集`test_loader`，具体代码展示如下：
生成图片对应信息以便读取到`Dataset`类中的程序在附件中有详细说明，在此不做赘述，载入图片并进行预处理、生成数据集的关键代码如下：

```
my_transform = transforms.Compose([
        transforms.CenterCrop(128),
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
        transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))  # 归一
化
    ])
class MyDataset(Dataset):
    def __init__(self, txt_path, transform = None):
        fh = open(txt_path, 'r') #读取 制作好的txt文件的 图片路径和标签到imgs里
        imgs = []
        for line in fh:
            line = line.rstrip()
            words = line.split('!')
            imgs.append((words[0], int(words[1])))
            self.imgs = imgs
            self.transform = transform
    def __getitem__(self, index):
        fn, label = self.imgs[index] #self.imgs是一个list，self.imgs的一个元素是
一个str，包含
        #图片路径，图片标签，这些信息是在init函数中从txt文件中读取的
        # fn是一个图片路径
        img = Image.open(fn).convert('RGB') #利用Image.open对图片进行读取，img类
```

```
型为 Image ， mode='RGB'
        if self.transform is not None:
            img = self.transform(img)
        return img, label
    def __len__(self):
        return len(self.imgs)
train_data = MyDataset('E://VSCODE/py/alltrain.txt',transform=my_transform)
test_data=MyDataset('E://VSCODE/py/alltest.txt',transform=my_transform)
```

运用pytorch框架搭建卷积神经网络的关键代码如下：

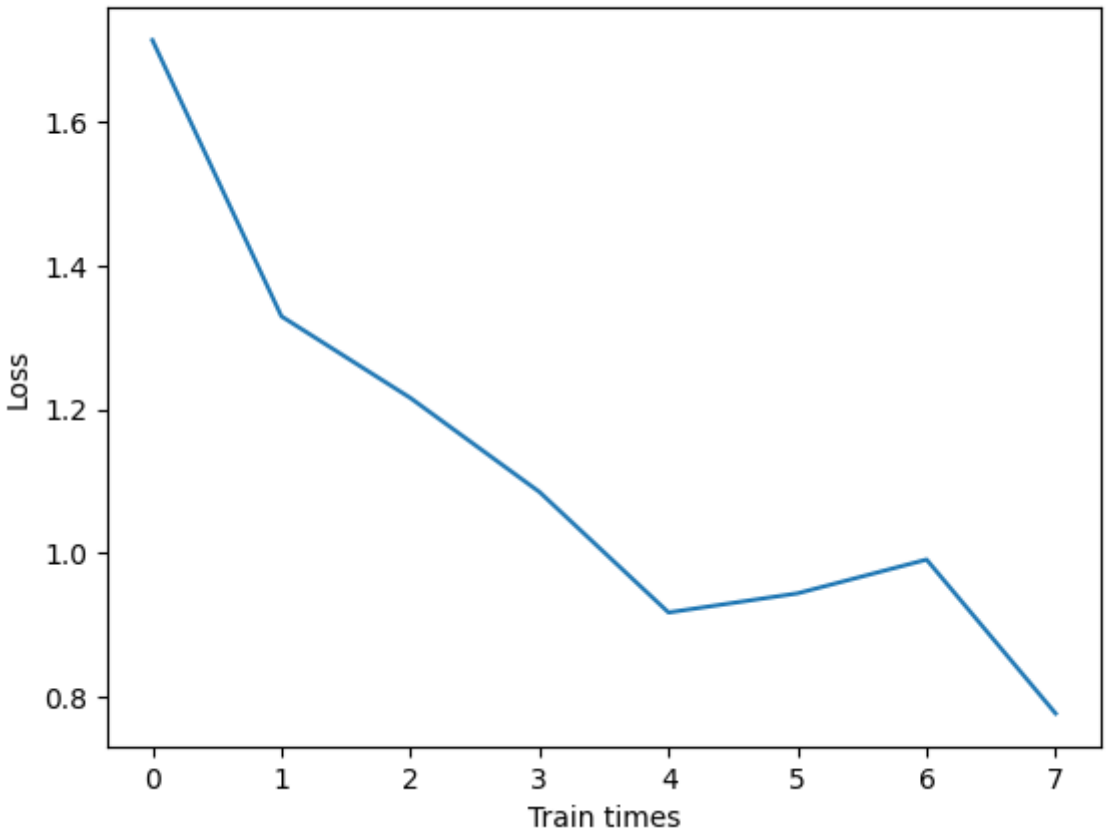
```
class cuixs_convNet(nn.Module):
    def __init__(self):
        super(cuixs_convNet,self).__init__()
        #卷积层
        self.features=nn.Sequential(

nn.Conv2d(in_channels=3,out_channels=64,kernel_size=3,stride=1,padding=1),#卷积
            nn.BatchNorm2d(num_features=64),#添加BatchNorm2d进行数据的归一化处理，
            nn.ReLU(inplace=True),#激活函数，就地操作
            nn.MaxPool2d(kernel_size=2,stride=2),#池化

            nn.Conv2d(64,128,3,1,1),
            nn.BatchNorm2d(128),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2,2),
            nn.Conv2d(128,256,3,1,1),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(4,4),
        )
        #分类层
        self.classify=nn.Sequential(
            nn.Dropout(0.5),#Dropout层,防止过拟合
            nn.Linear(256*8*8,256),#连接层
            nn.BatchNorm1d(256),#归一化
            nn.ReLU(inplace=True),#激活函数

            nn.Dropout(0.5),
            nn.Linear(256,256),
            nn.BatchNorm1d(256),
            nn.ReLU(inplace=True),

            nn.Dropout(0.5),
            nn.Linear(256,5),
        )
    def forward(self,x):
        #特征提取
        x=self.features(x)
        #展平成一维向量
        x=x.flatten(1)
        x=self.classify(x)
        return x
```

神经网络搭建完毕后，利用训练集对卷积神经网络进行训练，并在测试集上进行测试，训练和测试的函数如下：

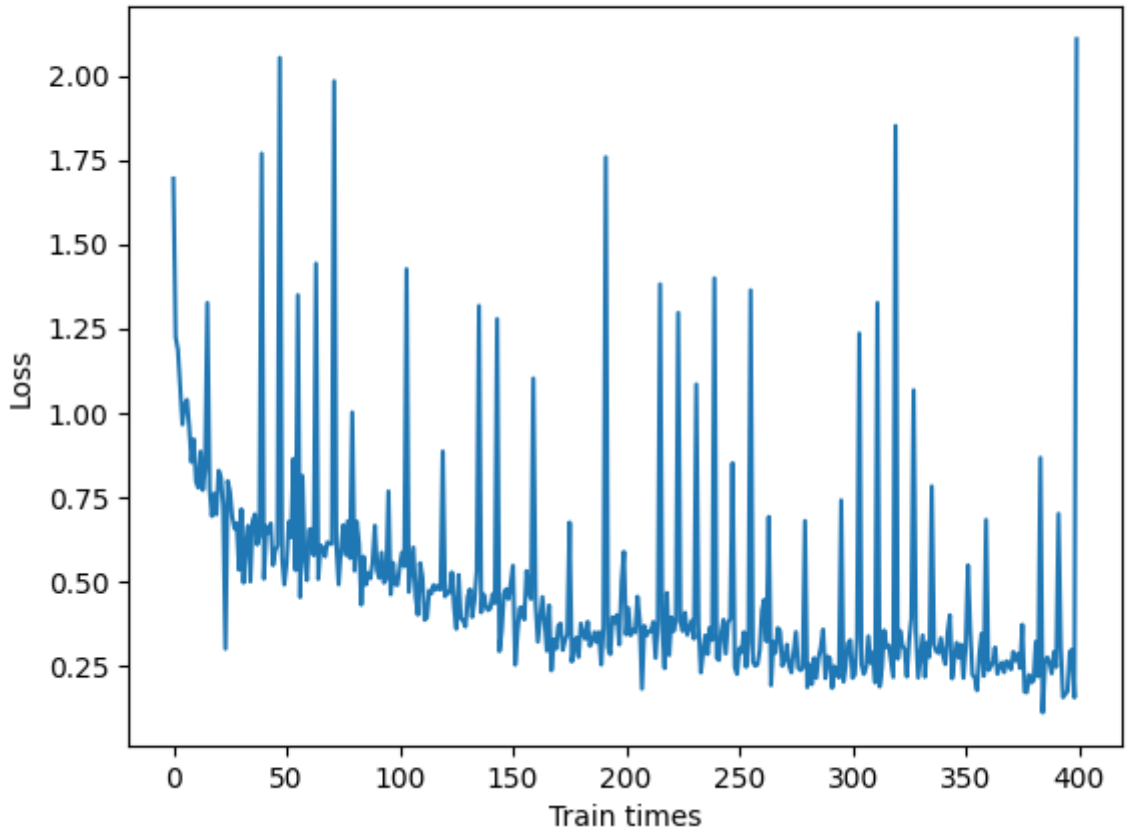
```
def
Train(epochs_num,_model,_device,_criterion,_optim,_exp_lr_scheduler,_trian_load
er):
    _model.train()  #训练模式
    _exp_lr_scheduler.step()    #开始调度
    loss_set=[]
    for epoch in range(epochs_num):#对训练集重复epochs_num次
        for i,(img,label) in enumerate(_trian_loader):
            sample=img.to(device)
            label=label.to(device)
             # 此时样本是一批图片，在CNN的输入中，我们需要将其变为四维，
            # reshape第一个-1 代表自动计算批量图片的数目n
            # 最后reshape得到的结果就是n张图片，每一张图片都是三通道的128 * 128，得
到四维张量
            sample=sample.reshape(-1,3,128,128)
            output=_model(sample)
            #损失函数
            loss=_criterion(output,label)
            #设置内部参数为0
            _optim.zero_grad()
            #向后传播
            loss.backward()
            #参数更新
            _optim.step()
            print("Epoch:{}/{}, step:{}, loss:{:.4f}".format(epoch + 1,
epochs_num, i + 1, loss.item()))
            loss_set.append(loss.item())
    return loss_set

def Test(_test_loader,_model,_device,_criterion):
    _model.eval()#测试模式
    loss=0
    correct=0
    with torch.no_grad():
        for data,target in _test_loader:
            data=data.to(_device)
            target=target.to(_device)
            output=_model(data.reshape(-1,3,128,128))
            loss+=_criterion(output,target).item()
            pred=output.data.max(1,keepdim=True)[1]
            correct+=pred.eq(target.data.view_as(pred)).cpu().sum()
    loss /= len(_test_loader.dataset)
    print('\nAverage loss: {:.4f}, Accuracy: {}/{} ({:.3f}%)\n'.format(
        loss, correct, len(_test_loader.dataset),
        100. * correct / len(_test_loader.dataset)))
```

三、实验结果

选取epoch=50，batch的大小为128张/批，初始学习率设置为0.001，在学习率调度器中定义学习率衰减周期step_size=6，衰减的乘法因子gamma设置为0.1，在测试集上进行训练，得到在每个epoch中，损失函数的变化图像如下：



在总的训练时间内（50个epoch），损失函数的变化如下：



训练完成后，在训练集上进行测试，得到**正确率为90.687%**，结果如下：

```
Epoch:49/50, step:5, loss:0.1883
Epoch:49/50, step:6, loss:0.2082
Epoch:49/50, step:7, loss:0.2193
Epoch:49/50, step:8, loss:0.6478
Epoch:50/50, step:1, loss:0.1968
Epoch:50/50, step:2, loss:0.2293
Epoch:50/50, step:3, loss:0.2794
Epoch:50/50, step:4, loss:0.2655
Epoch:50/50, step:5, loss:0.3170
Epoch:50/50, step:6, loss:0.3445
Epoch:50/50, step:7, loss:0.4128
Epoch:50/50, step:8, loss:0.1665

Average loss: 0.0020, Accuracy: 818/902 (90.687%)

PS E:\VSCODE\py>
```

在测试集上进行测试，得到结果如下，**正确率为80%**：

```
Epoch:50/50, step:4, loss:0.1756
Epoch:50/50, step:5, loss:0.2928
Epoch:50/50, step:6, loss:0.2994
Epoch:50/50, step:7, loss:0.1570
Epoch:50/50, step:8, loss:2.1087

Average loss: 0.0697, Accuracy: 8/10 (80.000%)

PS E:\VSCODE\py>
```

四、创新点&&优化

- 调用了学习率调度器`lr_scheduler.StepLR`，在训练的过程中更新学习率，达到更好的效果
- 一开始的epoch值选取过小，导致正确率不高，而后面经过多次实验，选取epoch值为50，预测的准确率有了很大的提升。

五、参考资料

- PyTorch.pdf
  -PyTorch 入门实战（四）——利用Torch.nn构建卷积神经网络_一株草的世界的博客-CSDN博客