**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 | **系统结构班** | 专业（方向） | **计算机科学与技术** |
| 学号 | **21307358** | 姓名 | **曾慧蕾** |

# 实验题目

* 运用pytorch框架完成中药图片分类，具体见给出的训练集和测试集(data.zip)。
* 需要画出loss、准确率曲线图。

# 实验内容

1. 算法原理

本实验使用卷积神经网络来完成中药图片分类任务。

**CNN算法原理主要包括以下几个方面：**

* 1. 卷积层：卷积层是CNN的核心层，它通过卷积操作提取图像的特征。卷积操作是指将一个filter应用于输入图像的每个位置，计算出对应位置的输出值。通过使用多个滤波器，卷积层可以提取出不同的特征。
  2. 池化层：池化层用于减小卷积层输出的特征图的大小，同时保留重要的特征。常用的池化操作包括最大池化和平均池化，它们分别取池化窗口内的最大值和平均值作为输出值。
  3. 激活函数：激活函数用于引入非线性因素，增加模型的表达能力。常用的激活函数包括ReLU、Sigmoid和Tanh等，而在多分类问题中常用到softmax。
  4. 全连接层：全连接层用于将卷积层和池化层的输出连接起来，形成最终的分类结果。全连接层通常包括多个神经元，每个神经元对应一个类别，输出该类别的概率。

**训练CNN完成中药图片分类的步骤如下：**

* 1. 数据预处理：将中药图片转换为数字矩阵，并进行归一化处理。
  2. 构建CNN模型：根据中药图片的特点和分类任务的要求，构建合适的CNN模型，包括卷积层、池化层、激活函数和全连接层等。
  3. 模型训练：使用训练集对CNN模型进行训练，通过反向传播算法更新模型参数。
  4. 评估：用测试集对训练好的模型进行评估，计算模型的准确率、评估模型的性能。
  5. 模型优化：根据模型评估结果，对CNN模型进行优化，包括调整模型结构、调整超参数等，提高模型的性能。

在CNN模型中，卷积层和池化层可以提取图像的局部特征，而全连接层可以将这些局部特征组合起来，形成最终的分类结果。因此，在中药图片分类任务中，我们可以使用多个卷积层和池化层来提取不同层次的特征，然后使用全连接层将这些特征组合起来，形成最终的分类结果。

1. 关键代码展示（带注释）

收集图片地址并进行数据预处理，用dataloader载入存储的图片数据并打包成后续可用于CNN训练的数据。

def data\_processing():

    train\_data = dataset('D:\\code\\python\\file\\lab11\\alltrain.txt')

    test\_data = dataset('D:\\code\\python\\file\\lab11\\alltest.txt')

    train\_loader = DataLoader(train\_data,batch\_size=batch\_size,shuffle=True)

    test\_loader = DataLoader(test\_data,batch\_size=batch\_size,shuffle=False)

    return train\_loader, test\_loader

用于处理数据的类：

class dataset(DataLoader):

    def \_\_init\_\_(self,file):

        self.file=file

        self.img=[]

        f=open(file,'r',encoding='utf-8')

        for img\_info in f:

            example=img\_info.rstrip()

            example=example.split('!') #在感叹号前后分割，前为地址后为标签

            self.img.append((example[0],int(example[1])))

    def \_\_getitem\_\_(self,index):

        img\_path,label=self.img[index]

        image=Image.open(img\_path)

        if image.mode!='RGB':

            image=image.convert('RGB')

        image=image.resize((128,128),Image.BILINEAR)

        image=np.array(image).astype('float32')

        image=image.transpose((2,0,1))/255

        label=np.array([label],dtype='int64')

        return image,label

    def \_\_len\_\_(self):

        return len(self.img)

然后定义好网络、损失函数与网络优化器，用于训练网络：

    train\_loader,test\_loader=data\_processing()

    device=torch.device("cuda" if torch.cuda.is\_available() else "cpu")

    net=CNN().to(device) # 定义网络

    criterion=torch.nn.CrossEntropyLoss() #定义损失函数

    optim=torch.optim.Adam(net.parameters(),lr=1e-3) #定义网络优化器

    epochs=2 # 定义epoch

cnn网络定义如下，包括四层卷积层和一层分类层，分类层在最后预测五种中药类型：

class CNN(nn.Module):

    def \_\_init\_\_(self):

        super(CNN,self).\_\_init\_\_()

        self.conv1=nn.Sequential(

            nn.Conv2d(3, 32, 3, 1, 1),#卷积

            nn.BatchNorm2d(32),

            nn.ReLU(inplace=True),#激活函数

            nn.Conv2d(32,32,5,1,2),

            nn.BatchNorm2d(32),

            nn.ReLU(inplace=True),

            nn.MaxPool2d(kernel\_size=2,stride=2),#池化)

        self.conv2=nn.Sequential(

            nn.Conv2d(32,64,3,1,1),

            nn.BatchNorm2d(64),

            nn.ReLU(inplace=True),

            nn.Conv2d(64,64,3,1,1),

            nn.BatchNorm2d(64),

            nn.ReLU(inplace=True),

            nn.MaxPool2d(2,2),)

        self.conv3=nn.Sequential(

            nn.Conv2d(64,128,3,1,1),

            nn.BatchNorm2d(128),

            nn.ReLU(inplace=True),

            nn.Conv2d(128,128,3,1,1),

            nn.BatchNorm2d(128),

            nn.ReLU(inplace=True),

            nn.MaxPool2d(2,2),)

        self.conv4=nn.Sequential(

            nn.Conv2d(128,256,3,1,1),

            nn.BatchNorm2d(256),

            nn.ReLU(inplace=True),

            nn.Conv2d(256,256,3,1,1),

            nn.BatchNorm2d(256),

            nn.ReLU(inplace=True),

            nn.MaxPool2d(2,2),)

        self.classify=nn.Sequential(

            nn.Dropout(0.5), #防止过拟合

            nn.Linear(256\*8\*8,256),

            nn.BatchNorm1d(256),

            nn.ReLU(inplace=True),

            nn.Dropout(0.5),

            nn.Linear(256,128),

            nn.BatchNorm1d(128),

            nn.ReLU(inplace=True),

            nn.Dropout(0.5),

            nn.Linear(128,5),

            nn.BatchNorm1d(5),

            nn.ReLU(inplace=True),)

    def forward(self,x):

        x=self.conv1(x)

        x=self.conv2(x)

        x=self.conv3(x)

        x=self.conv4(x)

        x=x.flatten(1) #展平成一维向量

        x=self.classify(x)

        return x

训练网络的部分可以直接套模型，步骤为：清除梯度、反向传播、优化器更新：

    net.train()

    loss=[]

    accuracy=[]

    for epoch in range(epochs):

        for step,(input,label) in enumerate(train\_loader):

            input=input.reshape(-1,3,128,128)

            output=net(input)

            loss\_one=criterion(output,label.squeeze(dim=1))

            optim.zero\_grad() #在更新前清除上一步的梯度

            loss\_one.backward() #loss反向传播

            optim.step() #优化器更新

            loss.append(loss\_one.item()) #损失值记录

            rate=test(net)

            accuracy.append(rate)

            print("epoch:",epoch,"data id:",step+1,"loss:",loss\_one.item(),"accuracy:",rate)

测试部分：每进行一次实验都会测试一次，从而展现准确率变化：

def test(net):

    correct=0

    total=0

    with torch.no\_grad():

        for \_,(x,y) in enumerate(test\_loader):

            x=x.reshape(-1,3,128,128)

            outputs=net(x)

            \_,pred=torch.max(outputs.data,dim=1)

            pred=pred.reshape(10,1)

            total+=y.size(0)

            correct+=(pred==y).sum().item()

    return correct/total

1. 创新点&优化（如果有）

在多次重复训练后发现，最终训练结果的准确率受batch\_size影响较大，不同的batch\_size会有不同的训练次数。在多次调整过后选用了最合适的一条：

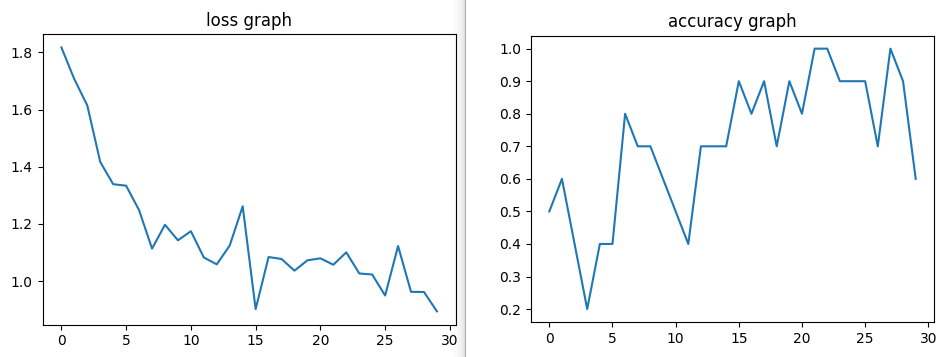
batch\_size=128

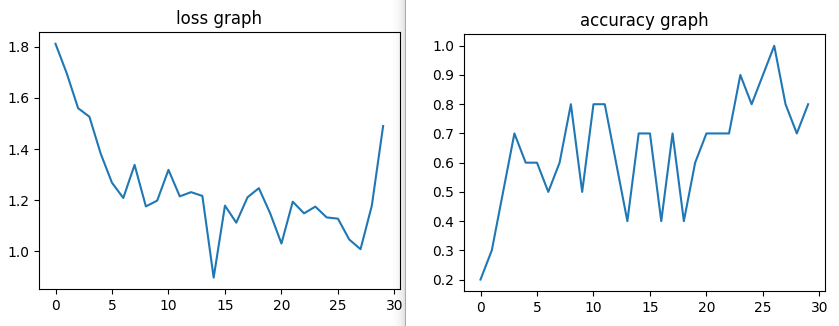
后续展示正确率的变化程度

# 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）

当batch\_size=64，epochs=2时，实验结果如下所示：

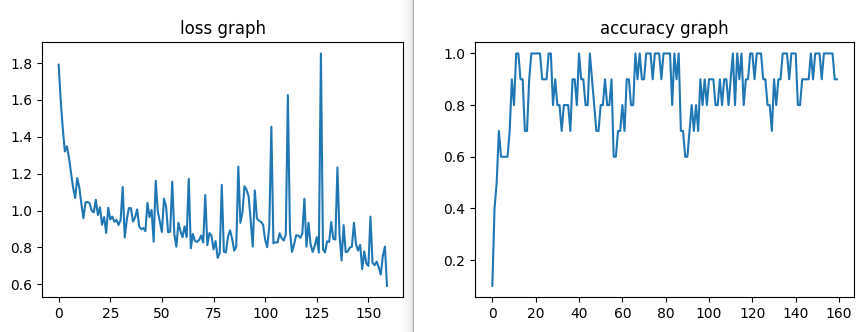


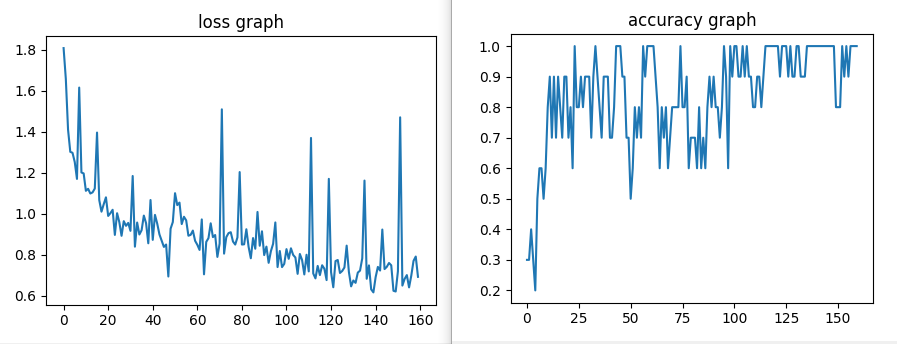


进行两次测试后可见，正确率呈波动上升趋势、loss图则是波动下降趋势，在三十次测试中正确率最高达到过三次100%

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）

当batch\_size=128，epoch=50时：





在训练次数到达10+时，正确率就已经可以达到90%，而当batch\_size=64时几乎在18~23内才能第一次到达90%。可见当调整到合适的batch size时能使模型训练效果提升。

# 参考资料

[基于图像分类网络VGG实现中草药识别 - 飞桨AI Studio (baidu.com)](https://aistudio.baidu.com/aistudio/projectdetail/5738888)

[Pytorch卷积神经网络图像识别-关于数据预处理和模型建立的参考](https://blog.csdn.net/qq_25426559/article/details/122263705)

**超算习堂·课时30-第十六讲实验课**