**中山大学计算机学院**

**人工智能**

**本科生实验报告**

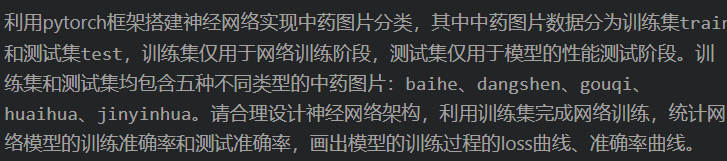
**（2022学年春季学期）**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 教学班级 | **202320346** | 专业（方向） | **计算机科学与技术** |
| 学号 | **21312450** | 姓名 | **林隽哲** |

# 实验题目

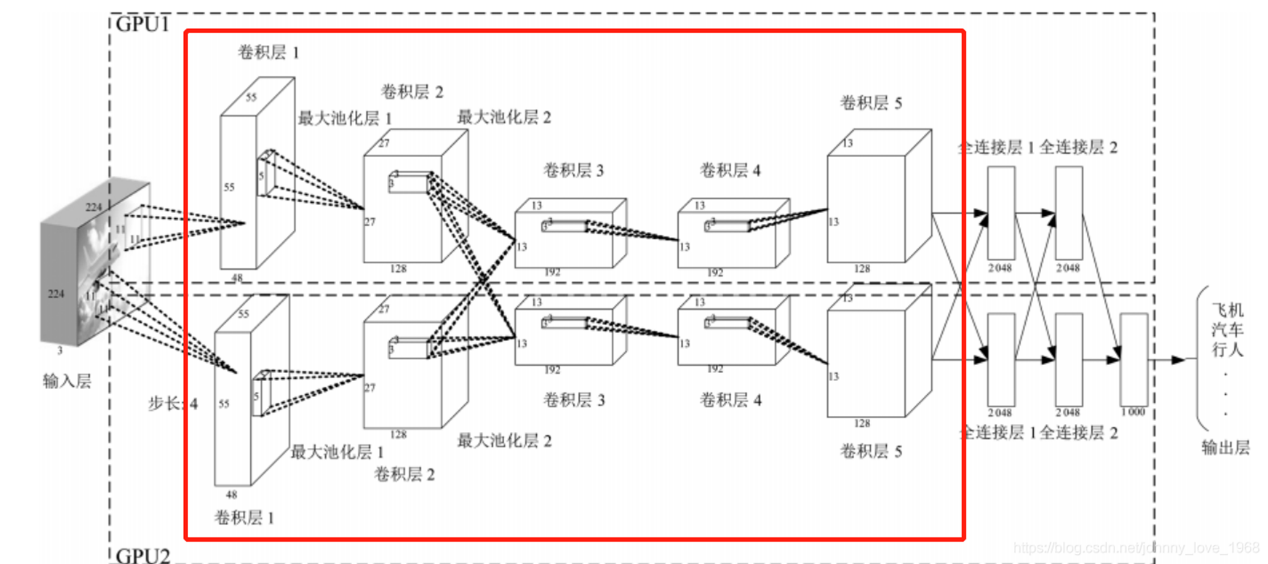
中药图片分类任务



# 实验内容

1. 算法原理

**卷积神经网路（Convolution Neural Network, CNN）**



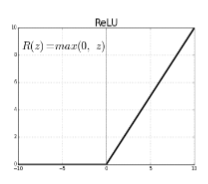
* + 1. **输入层：**输入层接受原始图像数据。本次实验使用的训练数据为彩色的图像，均由三个颜色通道（RGB）组成，因此输入层的输入图像的厚度为3。同时，为了方便计算，需要在数据预处理时统一每张图片的宽高。
    2. **卷积层：**

**箱线图

低可信度描述已自动生成**

卷积层由**torch.nn.Conv2d**实现。卷积层接受5个主要参数：  
·**in\_channels**: 输入维度，第一层的卷积层作为输入层，输入维度为3。随后的卷积层的输入维度取决于上一个最大池化层的输出维度；  
·**out\_channels**: 输出维度，也即为卷积层使用的卷积核的个数；  
·**kernel\_size**: 卷积核的大小；  
·**stride**: 卷积核移动的步长；  
·**padding**：图像的边缘扩充。  
由于我想要实现当一个图像在经过卷积变化后的大小与原来的一致，因此我选择了将每一个卷积层的**stride**都设为了1，此时只要满足 即可。

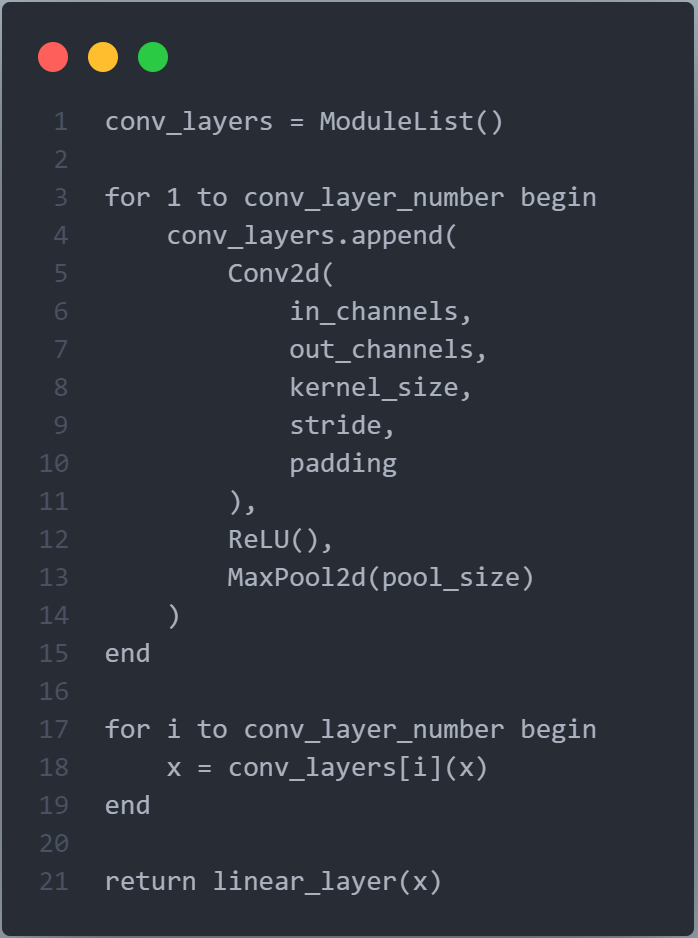
* + 1. **激活函数：**这里使用**ReLU**函数作为激活函数。通过**torch.nn.ReLU**实现。

****

* + 1. **最大池化层：**池化层通过减小特征图的大小来减少计算复杂性。它通过选择池化窗口内的最大值或平均值来实现。这有利于提取最重要的特征。最大池化层通过**torch.nn.Maxpool2d**实现。
    2. **全连接和输出：**最后，全连接层将提取的特征映射转化为网络的最终输出。这可以是一个分类标签、回归值或其他任务的结果。

1. 伪代码

**CNN结构的伪代码**



**训练流程**

**文本

描述已自动生成**

1. 关键代码展示

**CNN模型的实现**

class CNN(nn.Module):

    def \_\_init\_\_(*self*, *img\_size*=256, *in\_channels*=3, *out\_channels*=5, *conv\_layers\_params*=None):

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

        if *conv\_layers\_params* is None:

*conv\_layers\_params* = [

                {'kernel\_num': 16, 'kernel\_size': 5, 'pool\_size': 2},

                {'kernel\_num': 32, 'kernel\_size': 5, 'pool\_size': 2}

            ]

*self*.conv\_layers = nn.ModuleList()

        current\_channels = *in\_channels*

*# Create convolutional layers based on provided parameters*

        for params in *conv\_layers\_params*:

*self*.conv\_layers.append(nn.Sequential(

                nn.Conv2d(

*in\_channels*=current\_channels,

*out\_channels*=params['kernel\_num'],

*kernel\_size*=params['kernel\_size'],

*stride*=1,

*padding*=(params['kernel\_size'] - 1) // 2

                ),

                nn.ReLU(),

                nn.MaxPool2d(params['pool\_size'])

            ))

            current\_channels = params['kernel\_num']

*# Calculate the size of the flattened features after all conv and pool layers*

        feature\_size = *self*.\_calculate\_feature\_size(*img\_size*, *conv\_layers\_params*)

*# Fully connected layer*

*self*.out = nn.Linear(feature\_size, *out\_channels*)

    def \_calculate\_feature\_size(*self*, *img\_size*, *conv\_layers\_params*):

*# Calculate the size of the flattened features after all conv and pool layers*

        for params in *conv\_layers\_params*:

            padding = (params['kernel\_size'] - 1) // 2

*img\_size* = (*img\_size* - params['kernel\_size'] + 2 \* padding) // params['pool\_size'] + 1

        return *img\_size*\*\*2 \* *conv\_layers\_params*[-1]['kernel\_num']

    def forward(*self*, *x*):

*# Pass input through each convolutional layer*

        for conv\_layer in *self*.conv\_layers:

*x* = conv\_layer(*x*)

*# Flatten the output for the fully connected layer*

*x* = *x*.view(*x*.size(0), -1)

*# Pass through the fully connected layer*

        output = *self*.out(*x*)

        return output

**训练部分的实现**

for epoch in range(EPOCH):

    for step, (x, y) in enumerate(train\_loader):

        x = x.to(*device*=device)

        y = y.to(*device*=device)

*# forward*

        output = cnn(x)

*# calculate loss*

        loss = loss\_func(output, y)

*# clear gradients for this training step before backward*

        optimizer.zero\_grad()

*# backpropagation, compute gradients*

        loss.backward()

*# apply gradients*

        optimizer.step()

*# Test*

        if step and step % 1 == 0:

            pred\_y, accuracy = test\_model()

            print('Epoch: ', epoch, '| train loss: %.4f' % loss.item(), '| test accuracy: %.2f' % accuracy)

            loss\_record.append(loss.item())

            accuracy\_record.append(accuracy)

else:

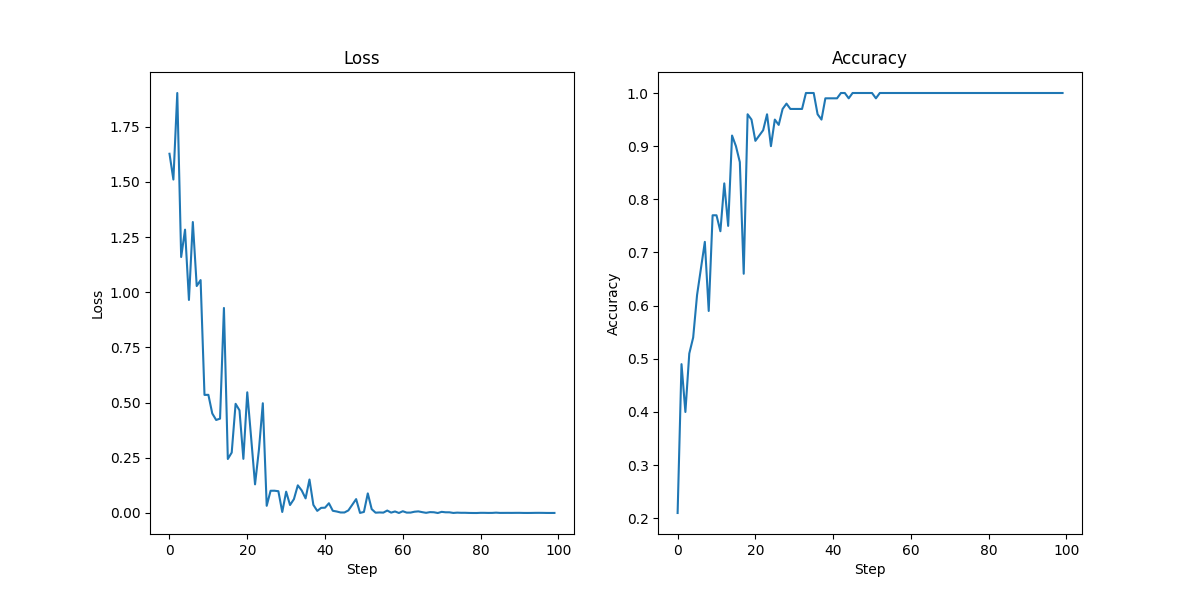
    print(f'Prediction:\t {pred\_y}')

    print(f'Ground Truth:\t {test\_y.cpu().data.numpy()}')

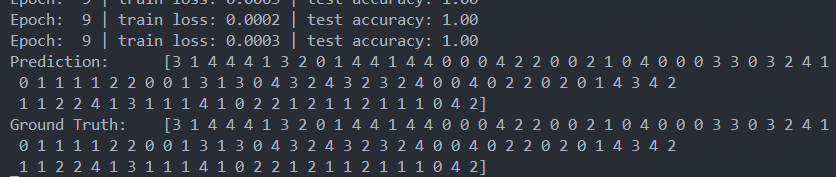
# 实验结果及分析

本次实验中我使用了两个测试集，一个为实验数据中给出的10个数据集，另一个为我在训练数据中随机抽选出的100个数据集。在计算准确率时与画图时，我将会使用较大的数据集。更小的数据集的测试我将会单独给出。

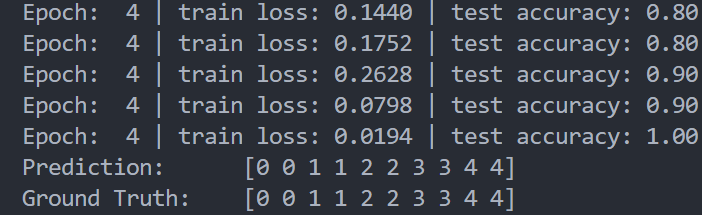
**训练过程的loss曲线、准确率曲线**



**使用大测试集最终的loss与accuracy**

****

**使用小测试集最终的loss与accuracy**

****

# 参考资料

* 理论课课件